

GiNaC

1.8.0

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>7</b>
3.1	Class List . . . . .	7
<b>4</b>	<b>File Index</b>	<b>15</b>
4.1	File List . . . . .	15
<b>5</b>	<b>Namespace Documentation</b>	<b>19</b>
5.1	GiNaC Namespace Reference . . . . .	19
5.1.1	Typedef Documentation . . . . .	58
5.1.1.1	archive_node_id . . . . .	58
5.1.1.2	archive_atom . . . . .	58
5.1.1.3	synthesize_func . . . . .	59
5.1.1.4	unarchive_map_t . . . . .	59
5.1.1.5	exvector . . . . .	59
5.1.1.6	exset . . . . .	59
5.1.1.7	exmap . . . . .	59
5.1.1.8	evalffunctype . . . . .	59
5.1.1.9	FUNCP_1P . . . . .	59
5.1.1.10	FUNCP_2P . . . . .	60

5.1.1.11	FUNCP_CUBA . . . . .	60
5.1.1.12	epvector . . . . .	60
5.1.1.13	epp . . . . .	60
5.1.1.14	exprseq . . . . .	60
5.1.1.15	paramset . . . . .	60
5.1.1.16	eval_funcp . . . . .	60
5.1.1.17	evalf_funcp . . . . .	61
5.1.1.18	conjugate_funcp . . . . .	61
5.1.1.19	real_part_funcp . . . . .	61
5.1.1.20	imag_part_funcp . . . . .	61
5.1.1.21	expand_funcp . . . . .	61
5.1.1.22	derivative_funcp . . . . .	61
5.1.1.23	expl_derivative_funcp . . . . .	61
5.1.1.24	power_funcp . . . . .	61
5.1.1.25	series_funcp . . . . .	62
5.1.1.26	print_funcp . . . . .	62
5.1.1.27	info_funcp . . . . .	62
5.1.1.28	eval_funcp_1 . . . . .	62
5.1.1.29	evalf_funcp_1 . . . . .	62
5.1.1.30	conjugate_funcp_1 . . . . .	62
5.1.1.31	real_part_funcp_1 . . . . .	62
5.1.1.32	imag_part_funcp_1 . . . . .	62
5.1.1.33	expand_funcp_1 . . . . .	63
5.1.1.34	derivative_funcp_1 . . . . .	63
5.1.1.35	expl_derivative_funcp_1 . . . . .	63
5.1.1.36	power_funcp_1 . . . . .	63
5.1.1.37	series_funcp_1 . . . . .	63
5.1.1.38	print_funcp_1 . . . . .	63
5.1.1.39	info_funcp_1 . . . . .	63
5.1.1.40	eval_funcp_2 . . . . .	63

5.1.1.41	<a href="#">evalf_funcp_2</a>	64
5.1.1.42	<a href="#">conjugate_funcp_2</a>	64
5.1.1.43	<a href="#">real_part_funcp_2</a>	64
5.1.1.44	<a href="#">imag_part_funcp_2</a>	64
5.1.1.45	<a href="#">expand_funcp_2</a>	64
5.1.1.46	<a href="#">derivative_funcp_2</a>	64
5.1.1.47	<a href="#">expl_derivative_funcp_2</a>	64
5.1.1.48	<a href="#">power_funcp_2</a>	64
5.1.1.49	<a href="#">series_funcp_2</a>	65
5.1.1.50	<a href="#">print_funcp_2</a>	65
5.1.1.51	<a href="#">info_funcp_2</a>	65
5.1.1.52	<a href="#">eval_funcp_3</a>	65
5.1.1.53	<a href="#">evalf_funcp_3</a>	65
5.1.1.54	<a href="#">conjugate_funcp_3</a>	65
5.1.1.55	<a href="#">real_part_funcp_3</a>	65
5.1.1.56	<a href="#">imag_part_funcp_3</a>	65
5.1.1.57	<a href="#">expand_funcp_3</a>	66
5.1.1.58	<a href="#">derivative_funcp_3</a>	66
5.1.1.59	<a href="#">expl_derivative_funcp_3</a>	66
5.1.1.60	<a href="#">power_funcp_3</a>	66
5.1.1.61	<a href="#">series_funcp_3</a>	66
5.1.1.62	<a href="#">print_funcp_3</a>	66
5.1.1.63	<a href="#">info_funcp_3</a>	66
5.1.1.64	<a href="#">eval_funcp_4</a>	67
5.1.1.65	<a href="#">evalf_funcp_4</a>	67
5.1.1.66	<a href="#">conjugate_funcp_4</a>	67
5.1.1.67	<a href="#">real_part_funcp_4</a>	67
5.1.1.68	<a href="#">imag_part_funcp_4</a>	67
5.1.1.69	<a href="#">expand_funcp_4</a>	67
5.1.1.70	<a href="#">derivative_funcp_4</a>	67

5.1.1.71	expl_derivative_funcp_4 . . . . .	67
5.1.1.72	power_funcp_4 . . . . .	68
5.1.1.73	series_funcp_4 . . . . .	68
5.1.1.74	print_funcp_4 . . . . .	68
5.1.1.75	info_funcp_4 . . . . .	68
5.1.1.76	eval_funcp_5 . . . . .	68
5.1.1.77	evalf_funcp_5 . . . . .	68
5.1.1.78	conjugate_funcp_5 . . . . .	68
5.1.1.79	real_part_funcp_5 . . . . .	69
5.1.1.80	imag_part_funcp_5 . . . . .	69
5.1.1.81	expand_funcp_5 . . . . .	69
5.1.1.82	derivative_funcp_5 . . . . .	69
5.1.1.83	expl_derivative_funcp_5 . . . . .	69
5.1.1.84	power_funcp_5 . . . . .	69
5.1.1.85	series_funcp_5 . . . . .	69
5.1.1.86	print_funcp_5 . . . . .	70
5.1.1.87	info_funcp_5 . . . . .	70
5.1.1.88	eval_funcp_6 . . . . .	70
5.1.1.89	evalf_funcp_6 . . . . .	70
5.1.1.90	conjugate_funcp_6 . . . . .	70
5.1.1.91	real_part_funcp_6 . . . . .	70
5.1.1.92	imag_part_funcp_6 . . . . .	70
5.1.1.93	expand_funcp_6 . . . . .	71
5.1.1.94	derivative_funcp_6 . . . . .	71
5.1.1.95	expl_derivative_funcp_6 . . . . .	71
5.1.1.96	power_funcp_6 . . . . .	71
5.1.1.97	series_funcp_6 . . . . .	71
5.1.1.98	print_funcp_6 . . . . .	71
5.1.1.99	info_funcp_6 . . . . .	71
5.1.1.100	eval_funcp_7 . . . . .	72

5.1.1.101 evalf_funcp_7 . . . . .	72
5.1.1.102 conjugate_funcp_7 . . . . .	72
5.1.1.103 real_part_funcp_7 . . . . .	72
5.1.1.104 imag_part_funcp_7 . . . . .	72
5.1.1.105 expand_funcp_7 . . . . .	72
5.1.1.106 derivative_funcp_7 . . . . .	72
5.1.1.107 expl_derivative_funcp_7 . . . . .	73
5.1.1.108 power_funcp_7 . . . . .	73
5.1.1.109 series_funcp_7 . . . . .	73
5.1.1.110 print_funcp_7 . . . . .	73
5.1.1.111 info_funcp_7 . . . . .	73
5.1.1.112 eval_funcp_8 . . . . .	73
5.1.1.113 evalf_funcp_8 . . . . .	73
5.1.1.114 conjugate_funcp_8 . . . . .	74
5.1.1.115 real_part_funcp_8 . . . . .	74
5.1.1.116 imag_part_funcp_8 . . . . .	74
5.1.1.117 expand_funcp_8 . . . . .	74
5.1.1.118 derivative_funcp_8 . . . . .	74
5.1.1.119 expl_derivative_funcp_8 . . . . .	74
5.1.1.120 power_funcp_8 . . . . .	74
5.1.1.121 series_funcp_8 . . . . .	75
5.1.1.122 print_funcp_8 . . . . .	75
5.1.1.123 info_funcp_8 . . . . .	75
5.1.1.124 eval_funcp_9 . . . . .	75
5.1.1.125 evalf_funcp_9 . . . . .	75
5.1.1.126 conjugate_funcp_9 . . . . .	75
5.1.1.127 real_part_funcp_9 . . . . .	75
5.1.1.128 imag_part_funcp_9 . . . . .	76
5.1.1.129 expand_funcp_9 . . . . .	76
5.1.1.130 derivative_funcp_9 . . . . .	76

5.1.1.131 expl_derivative_funcp_9 . . . . .	76
5.1.1.132 power_funcp_9 . . . . .	76
5.1.1.133 series_funcp_9 . . . . .	76
5.1.1.134 print_funcp_9 . . . . .	76
5.1.1.135 info_funcp_9 . . . . .	77
5.1.1.136 eval_funcp_10 . . . . .	77
5.1.1.137 evalf_funcp_10 . . . . .	77
5.1.1.138 conjugate_funcp_10 . . . . .	77
5.1.1.139 real_part_funcp_10 . . . . .	77
5.1.1.140 imag_part_funcp_10 . . . . .	77
5.1.1.141 expand_funcp_10 . . . . .	77
5.1.1.142 derivative_funcp_10 . . . . .	78
5.1.1.143 expl_derivative_funcp_10 . . . . .	78
5.1.1.144 power_funcp_10 . . . . .	78
5.1.1.145 series_funcp_10 . . . . .	78
5.1.1.146 print_funcp_10 . . . . .	78
5.1.1.147 info_funcp_10 . . . . .	78
5.1.1.148 eval_funcp_11 . . . . .	78
5.1.1.149 evalf_funcp_11 . . . . .	79
5.1.1.150 conjugate_funcp_11 . . . . .	79
5.1.1.151 real_part_funcp_11 . . . . .	79
5.1.1.152 imag_part_funcp_11 . . . . .	79
5.1.1.153 expand_funcp_11 . . . . .	79
5.1.1.154 derivative_funcp_11 . . . . .	79
5.1.1.155 expl_derivative_funcp_11 . . . . .	79
5.1.1.156 power_funcp_11 . . . . .	80
5.1.1.157 series_funcp_11 . . . . .	80
5.1.1.158 print_funcp_11 . . . . .	80
5.1.1.159 info_funcp_11 . . . . .	80
5.1.1.160 eval_funcp_12 . . . . .	80



5.1.1.161 evalf_funcp_12 . . . . .	80
5.1.1.162 conjugate_funcp_12 . . . . .	80
5.1.1.163 real_part_funcp_12 . . . . .	81
5.1.1.164 imag_part_funcp_12 . . . . .	81
5.1.1.165 expand_funcp_12 . . . . .	81
5.1.1.166 derivative_funcp_12 . . . . .	81
5.1.1.167 expl_derivative_funcp_12 . . . . .	81
5.1.1.168 power_funcp_12 . . . . .	81
5.1.1.169 series_funcp_12 . . . . .	82
5.1.1.170 print_funcp_12 . . . . .	82
5.1.1.171 info_funcp_12 . . . . .	82
5.1.1.172 eval_funcp_13 . . . . .	82
5.1.1.173 evalf_funcp_13 . . . . .	82
5.1.1.174 conjugate_funcp_13 . . . . .	82
5.1.1.175 real_part_funcp_13 . . . . .	83
5.1.1.176 imag_part_funcp_13 . . . . .	83
5.1.1.177 expand_funcp_13 . . . . .	83
5.1.1.178 derivative_funcp_13 . . . . .	83
5.1.1.179 expl_derivative_funcp_13 . . . . .	83
5.1.1.180 power_funcp_13 . . . . .	83
5.1.1.181 series_funcp_13 . . . . .	84
5.1.1.182 print_funcp_13 . . . . .	84
5.1.1.183 info_funcp_13 . . . . .	84
5.1.1.184 eval_funcp_14 . . . . .	84
5.1.1.185 evalf_funcp_14 . . . . .	84
5.1.1.186 conjugate_funcp_14 . . . . .	84
5.1.1.187 real_part_funcp_14 . . . . .	85
5.1.1.188 imag_part_funcp_14 . . . . .	85
5.1.1.189 expand_funcp_14 . . . . .	85
5.1.1.190 derivative_funcp_14 . . . . .	85

5.1.1.191	<a href="#">expl_derivative_funcp_14</a>	85
5.1.1.192	<a href="#">power_funcp_14</a>	85
5.1.1.193	<a href="#">series_funcp_14</a>	86
5.1.1.194	<a href="#">print_funcp_14</a>	86
5.1.1.195	<a href="#">info_funcp_14</a>	86
5.1.1.196	<a href="#">eval_funcp_exvector</a>	86
5.1.1.197	<a href="#">evalf_funcp_exvector</a>	86
5.1.1.198	<a href="#">conjugate_funcp_exvector</a>	86
5.1.1.199	<a href="#">real_part_funcp_exvector</a>	86
5.1.1.200	<a href="#">imag_part_funcp_exvector</a>	87
5.1.1.201	<a href="#">expand_funcp_exvector</a>	87
5.1.1.202	<a href="#">derivative_funcp_exvector</a>	87
5.1.1.203	<a href="#">expl_derivative_funcp_exvector</a>	87
5.1.1.204	<a href="#">power_funcp_exvector</a>	87
5.1.1.205	<a href="#">series_funcp_exvector</a>	87
5.1.1.206	<a href="#">print_funcp_exvector</a>	87
5.1.1.207	<a href="#">info_funcp_exvector</a>	87
5.1.1.208	<a href="#">exhashmap</a>	88
5.1.1.209	<a href="#">spmap</a>	88
5.1.1.210	<a href="#">lookup_map</a>	88
5.1.1.211	<a href="#">lst</a>	88
5.1.1.212	<a href="#">uintvector</a>	88
5.1.1.213	<a href="#">unsignedvector</a>	88
5.1.1.214	<a href="#">exvectorvector</a>	88
5.1.1.215	<a href="#">sym_desc_vec</a>	89
5.1.1.216	<a href="#">digits_changed_callback</a>	89
5.1.1.217	<a href="#">print_context_class_info</a>	89
5.1.1.218	<a href="#">registered_class_info</a>	89
5.1.2	<a href="#">Enumeration Type Documentation</a>	89
5.1.2.1	<a href="#">anonymous enum</a>	89

5.1.3	Function Documentation	89
5.1.3.1	GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [1/32]	89
5.1.3.2	GINAC_BIND_UNARCHIVER() [1/49]	90
5.1.3.3	GINAC_DECLARE_UNARCHIVER() [1/51]	90
5.1.3.4	write_unsigned()	90
5.1.3.5	read_unsigned()	90
5.1.3.6	operator<<() [1/16]	90
5.1.3.7	operator<<() [2/16]	91
5.1.3.8	operator>>() [1/3]	91
5.1.3.9	operator>>() [2/3]	91
5.1.3.10	find_factory_fcn()	91
5.1.3.11	GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [2/32]	92
5.1.3.12	is_a() [1/3]	92
5.1.3.13	is_exactly_a() [1/2]	92
5.1.3.14	dynallocate() [1/2]	92
5.1.3.15	dynallocate() [2/2]	93
5.1.3.16	GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [3/32]	93
5.1.3.17	print_func< print_dflt >() [1/3]	93
5.1.3.18	GINAC_BIND_UNARCHIVER() [2/49]	93
5.1.3.19	GINAC_BIND_UNARCHIVER() [3/49]	93
5.1.3.20	GINAC_BIND_UNARCHIVER() [4/49]	93
5.1.3.21	GINAC_BIND_UNARCHIVER() [5/49]	94
5.1.3.22	GINAC_BIND_UNARCHIVER() [6/49]	94
5.1.3.23	GINAC_BIND_UNARCHIVER() [7/49]	94
5.1.3.24	GINAC_BIND_UNARCHIVER() [8/49]	94
5.1.3.25	is_dirac_slash()	94
5.1.3.26	base_and_index()	94
5.1.3.27	dirac_ONE()	94
5.1.3.28	get_dim_uint()	95
5.1.3.29	clifford_unit()	95

5.1.3.30	<a href="#">dirac_gamma()</a>	95
5.1.3.31	<a href="#">dirac_gamma5()</a>	96
5.1.3.32	<a href="#">dirac_gammaL()</a>	96
5.1.3.33	<a href="#">dirac_gammaR()</a>	97
5.1.3.34	<a href="#">dirac_slash()</a>	97
5.1.3.35	<a href="#">get_representation_label()</a> [1/2]	97
5.1.3.36	<a href="#">trace_string()</a>	97
5.1.3.37	<a href="#">dirac_trace()</a> [1/3]	98
5.1.3.38	<a href="#">dirac_trace()</a> [2/3]	98
5.1.3.39	<a href="#">dirac_trace()</a> [3/3]	98
5.1.3.40	<a href="#">canonicalize_clifford()</a>	99
5.1.3.41	<a href="#">clifford_star_bar()</a>	99
5.1.3.42	<a href="#">clifford_prime()</a>	99
5.1.3.43	<a href="#">remove_dirac_ONE()</a>	99
5.1.3.44	<a href="#">clifford_max_label()</a>	100
5.1.3.45	<a href="#">clifford_norm()</a>	100
5.1.3.46	<a href="#">clifford_inverse()</a>	100
5.1.3.47	<a href="#">lst_to_clifford()</a> [1/2]	100
5.1.3.48	<a href="#">lst_to_clifford()</a> [2/2]	101
5.1.3.49	<a href="#">get_clifford_comp()</a>	101
5.1.3.50	<a href="#">clifford_to_lst()</a>	101
5.1.3.51	<a href="#">clifford_moebius_map()</a> [1/2]	102
5.1.3.52	<a href="#">clifford_moebius_map()</a> [2/2]	102
5.1.3.53	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [2/51]	103
5.1.3.54	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [3/51]	103
5.1.3.55	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [4/51]	103
5.1.3.56	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [5/51]	103
5.1.3.57	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [6/51]	103
5.1.3.58	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [7/51]	104
5.1.3.59	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [8/51]	104

5.1.3.60	<a href="#">is_clifford_tinfo()</a>	104
5.1.3.61	<a href="#">clifford_bar()</a>	104
5.1.3.62	<a href="#">clifford_star()</a>	104
5.1.3.63	<a href="#">GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</a> [4/32]	105
5.1.3.64	<a href="#">print_func&lt; print_dflt &gt;()</a> [2/3]	105
5.1.3.65	<a href="#">GINAC_BIND_UNARCHIVER()</a> [9/49]	105
5.1.3.66	<a href="#">GINAC_BIND_UNARCHIVER()</a> [10/49]	105
5.1.3.67	<a href="#">GINAC_BIND_UNARCHIVER()</a> [11/49]	105
5.1.3.68	<a href="#">GINAC_BIND_UNARCHIVER()</a> [12/49]	105
5.1.3.69	<a href="#">GINAC_BIND_UNARCHIVER()</a> [13/49]	106
5.1.3.70	<a href="#">permute_free_index_to_front()</a>	106
5.1.3.71	<a href="#">color_ONE()</a>	106
5.1.3.72	<a href="#">color_T()</a>	107
5.1.3.73	<a href="#">color_f()</a>	107
5.1.3.74	<a href="#">color_d()</a>	108
5.1.3.75	<a href="#">color_h()</a>	108
5.1.3.76	<a href="#">is_color_tinfo()</a>	108
5.1.3.77	<a href="#">get_representation_label()</a> [2/2]	109
5.1.3.78	<a href="#">color_trace()</a> [1/3]	109
5.1.3.79	<a href="#">color_trace()</a> [2/3]	109
5.1.3.80	<a href="#">color_trace()</a> [3/3]	110
5.1.3.81	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [9/51]	110
5.1.3.82	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [10/51]	110
5.1.3.83	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [11/51]	110
5.1.3.84	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [12/51]	110
5.1.3.85	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [13/51]	111
5.1.3.86	<a href="#">GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</a> [5/32]	111
5.1.3.87	<a href="#">GINAC_BIND_UNARCHIVER()</a> [14/49]	111
5.1.3.88	<a href="#">GINAC_DECLARE_UNARCHIVER()</a> [14/51]	111
5.1.3.89	<a href="#">crc32()</a>	111

5.1.3.90	<code>are_ex_trivially_equal()</code>	112
5.1.3.91	<code>operator&lt;&lt;()</code> [3/16]	112
5.1.3.92	<code>operator&lt;&lt;()</code> [4/16]	112
5.1.3.93	<code>operator&lt;&lt;()</code> [5/16]	112
5.1.3.94	<code>nops()</code> [1/2]	113
5.1.3.95	<code>expand()</code> [1/2]	113
5.1.3.96	<code>conjugate()</code>	113
5.1.3.97	<code>real_part()</code>	113
5.1.3.98	<code>imag_part()</code>	114
5.1.3.99	<code>has()</code>	114
5.1.3.100	<code>find()</code>	114
5.1.3.101	<code>is_polynomial()</code>	114
5.1.3.102	<code>degree()</code>	115
5.1.3.103	<code>ldegree()</code>	115
5.1.3.104	<code>coeff()</code>	115
5.1.3.105	<code>numer()</code> [1/2]	115
5.1.3.106	<code>denom()</code> [1/2]	116
5.1.3.107	<code>numer_denom()</code>	116
5.1.3.108	<code>normal()</code>	116
5.1.3.109	<code>to_rational()</code>	116
5.1.3.110	<code>to_polynomial()</code>	117
5.1.3.111	<code>collect()</code>	117
5.1.3.112	<code>eval()</code>	117
5.1.3.113	<code>evalf()</code> [1/2]	117
5.1.3.114	<code>evalm()</code>	117
5.1.3.115	<code>eval_integ()</code>	118
5.1.3.116	<code>diff()</code>	118
5.1.3.117	<code>series()</code>	118
5.1.3.118	<code>match()</code>	118
5.1.3.119	<code>simplify_indexed()</code> [1/3]	119

5.1.3.120 <code>simplify_indexed()</code> [2/3]	119
5.1.3.121 <code>symmetrize()</code> [1/4]	119
5.1.3.122 <code>symmetrize()</code> [2/4]	119
5.1.3.123 <code>antisymmetrize()</code> [1/4]	119
5.1.3.124 <code>antisymmetrize()</code> [2/4]	120
5.1.3.125 <code>symmetrize_cyclic()</code> [1/4]	120
5.1.3.126 <code>symmetrize_cyclic()</code> [2/4]	120
5.1.3.127 <code>op()</code>	120
5.1.3.128 <code>lhs()</code>	120
5.1.3.129 <code>rhs()</code>	121
5.1.3.130 <code>is_zero()</code> [1/2]	121
5.1.3.131 <code>swap()</code> [1/2]	121
5.1.3.132 <code>subs()</code> [1/3]	121
5.1.3.133 <code>subs()</code> [2/3]	122
5.1.3.134 <code>subs()</code> [3/3]	122
5.1.3.135 <code>is_a()</code> [2/3]	122
5.1.3.136 <code>is_exactly_a()</code> [2/2]	122
5.1.3.137 <code>ex_to()</code>	122
5.1.3.138 <code>compile_ex()</code> [1/3]	123
5.1.3.139 <code>compile_ex()</code> [2/3]	123
5.1.3.140 <code>compile_ex()</code> [3/3]	124
5.1.3.141 <code>link_ex()</code> [1/3]	124
5.1.3.142 <code>link_ex()</code> [2/3]	125
5.1.3.143 <code>link_ex()</code> [3/3]	125
5.1.3.144 <code>unlink_ex()</code>	125
5.1.3.145 <code>swap()</code> [2/2]	126
5.1.3.146 <code>conjugateepvector()</code>	126
5.1.3.147 <code>factor()</code>	126
5.1.3.148 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [6/32]	127
5.1.3.149 <code>GINAC_DECLARE_UNARCHIVER()</code> [15/51]	127

5.1.3.150 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [7/32]	127
5.1.3.151 GINAC_BIND_UNARCHIVER() [15/49]	127
5.1.3.152 GINAC_DECLARE_UNARCHIVER() [16/51]	127
5.1.3.153 GINAC_BIND_UNARCHIVER() [16/49]	127
5.1.3.154 GINAC_DECLARE_UNARCHIVER() [17/51]	128
5.1.3.155 is_the_function()	128
5.1.3.156 make_hash_seed()	128
5.1.3.157 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [8/32]	128
5.1.3.158 print_func< print_context >()	129
5.1.3.159 GINAC_BIND_UNARCHIVER() [17/49]	129
5.1.3.160 GINAC_BIND_UNARCHIVER() [18/49]	129
5.1.3.161 GINAC_BIND_UNARCHIVER() [19/49]	129
5.1.3.162 is_dummy_pair() [1/2]	129
5.1.3.163 is_dummy_pair() [2/2]	130
5.1.3.164 find_free_and_dummy() [1/2]	130
5.1.3.165 minimal_dim()	130
5.1.3.166 GINAC_DECLARE_UNARCHIVER() [18/51]	131
5.1.3.167 GINAC_DECLARE_UNARCHIVER() [19/51]	131
5.1.3.168 GINAC_DECLARE_UNARCHIVER() [20/51]	131
5.1.3.169 find_free_and_dummy() [2/2]	131
5.1.3.170 find_dummy_indices()	131
5.1.3.171 count_dummy_indices()	132
5.1.3.172 count_free_indices()	132
5.1.3.173 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [9/32]	132
5.1.3.174 GINAC_BIND_UNARCHIVER() [20/49]	132
5.1.3.175 indices_consistent()	133
5.1.3.176 number_of_type()	133
5.1.3.177 rename_dummy_indices()	133
5.1.3.178 find_variant_indices()	133
5.1.3.179 reposition_dummy_indices()	134



5.1.3.180 product_to_exvector()	134
5.1.3.181 idx_symmetrization()	134
5.1.3.182 simplify_indexed() [3/3]	135
5.1.3.183 simplify_indexed_product()	135
5.1.3.184 hasindex()	135
5.1.3.185 get_all_dummy_indices_safely()	136
5.1.3.186 get_all_dummy_indices()	136
5.1.3.187 rename_dummy_indices_uniquely() [1/4]	136
5.1.3.188 rename_dummy_indices_uniquely() [2/4]	136
5.1.3.189 rename_dummy_indices_uniquely() [3/4]	137
5.1.3.190 rename_dummy_indices_uniquely() [4/4]	137
5.1.3.191 expand_dummy_sum()	137
5.1.3.192 GINAC_DECLARE_UNARCHIVER() [21/51]	138
5.1.3.193 conjugate_evalf()	138
5.1.3.194 conjugate_eval()	138
5.1.3.195 conjugate_print_latex()	138
5.1.3.196 conjugate_conjugate()	138
5.1.3.197 conjugate_expl_derivative()	138
5.1.3.198 conjugate_real_part()	139
5.1.3.199 conjugate_imag_part()	139
5.1.3.200 func_arg_info()	139
5.1.3.201 conjugate_info()	139
5.1.3.202 REGISTER_FUNCTION() [1/36]	139
5.1.3.203 real_part_evalf()	140
5.1.3.204 real_part_eval()	140
5.1.3.205 real_part_print_latex()	140
5.1.3.206 real_part_conjugate()	140
5.1.3.207 real_part_real_part()	140
5.1.3.208 real_part_imag_part()	140
5.1.3.209 real_part_expl_derivative()	141

5.1.3.210 REGISTER_FUNCTION() [2/36]	141
5.1.3.211 imag_part_evalf()	141
5.1.3.212 imag_part_eval()	141
5.1.3.213 imag_part_print_latex()	141
5.1.3.214 imag_part_conjugate()	142
5.1.3.215 imag_part_real_part()	142
5.1.3.216 imag_part_imag_part()	142
5.1.3.217 imag_part_expl_derivative()	142
5.1.3.218 REGISTER_FUNCTION() [3/36]	142
5.1.3.219 abs_evalf()	142
5.1.3.220 abs_eval()	143
5.1.3.221 abs_expand()	143
5.1.3.222 abs_expl_derivative()	143
5.1.3.223 abs_print_latex()	143
5.1.3.224 abs_print_csrc_float()	143
5.1.3.225 abs_conjugate()	144
5.1.3.226 abs_real_part()	144
5.1.3.227 abs_imag_part()	144
5.1.3.228 abs_power()	144
5.1.3.229 abs_info()	144
5.1.3.230 REGISTER_FUNCTION() [4/36]	145
5.1.3.231 step_evalf()	145
5.1.3.232 step_eval()	145
5.1.3.233 step_series()	145
5.1.3.234 step_conjugate()	145
5.1.3.235 step_real_part()	146
5.1.3.236 step_imag_part()	146
5.1.3.237 REGISTER_FUNCTION() [5/36]	146
5.1.3.238 csgn_evalf()	146
5.1.3.239 csgn_eval()	146

5.1.3.240 csgn_series()	147
5.1.3.241 csgn_conjugate()	147
5.1.3.242 csgn_real_part()	147
5.1.3.243 csgn_imag_part()	147
5.1.3.244 csgn_power()	147
5.1.3.245 REGISTER_FUNCTION() [6/36]	148
5.1.3.246 eta_evalf()	148
5.1.3.247 eta_eval()	148
5.1.3.248 eta_series()	148
5.1.3.249 eta_conjugate()	148
5.1.3.250 eta_real_part()	149
5.1.3.251 eta_imag_part()	149
5.1.3.252 REGISTER_FUNCTION() [7/36]	149
5.1.3.253 Li2_evalf()	149
5.1.3.254 Li2_eval()	149
5.1.3.255 Li2_deriv()	150
5.1.3.256 Li2_series() [1/2]	150
5.1.3.257 Li2_conjugate()	150
5.1.3.258 REGISTER_FUNCTION() [8/36]	150
5.1.3.259 Li3_eval()	150
5.1.3.260 REGISTER_FUNCTION() [9/36]	151
5.1.3.261 zetaderiv_eval()	151
5.1.3.262 zetaderiv_deriv()	151
5.1.3.263 REGISTER_FUNCTION() [10/36]	151
5.1.3.264 factorial_evalf()	151
5.1.3.265 factorial_eval()	152
5.1.3.266 factorial_print_dflt_latex()	152
5.1.3.267 factorial_conjugate()	152
5.1.3.268 factorial_real_part()	152
5.1.3.269 factorial_imag_part()	152

5.1.3.270 REGISTER_FUNCTION() [11/36]	152
5.1.3.271 binomial_evalf()	153
5.1.3.272 binomial_sym()	153
5.1.3.273 binomial_eval()	153
5.1.3.274 binomial_conjugate()	153
5.1.3.275 binomial_real_part()	153
5.1.3.276 binomial_imag_part()	154
5.1.3.277 REGISTER_FUNCTION() [12/36]	154
5.1.3.278 Order_eval()	154
5.1.3.279 Order_series()	154
5.1.3.280 Order_conjugate()	154
5.1.3.281 Order_real_part()	155
5.1.3.282 Order_imag_part()	155
5.1.3.283 Order_expl_derivative()	155
5.1.3.284 REGISTER_FUNCTION() [13/36]	155
5.1.3.285 Isolve()	155
5.1.3.286 fsolve()	156
5.1.3.287 zeta() [1/3]	156
5.1.3.288 zeta() [2/3]	156
5.1.3.289 is_the_function< zeta_SERIAL >()	157
5.1.3.290 G() [1/2]	157
5.1.3.291 G() [2/2]	157
5.1.3.292 is_the_function< G_SERIAL >()	157
5.1.3.293 psi() [1/4]	157
5.1.3.294 psi() [2/4]	158
5.1.3.295 is_the_function< psi_SERIAL >()	158
5.1.3.296 iterated_integral() [1/2]	158
5.1.3.297 iterated_integral() [2/2]	158
5.1.3.298 is_the_function< iterated_integral_SERIAL >()	158
5.1.3.299 is_order_function()	159

5.1.3.300 convert_H_to_Li()	159
5.1.3.301 EllipticK_evalf()	159
5.1.3.302 EllipticK_eval()	159
5.1.3.303 EllipticK_deriv()	160
5.1.3.304 EllipticK_series()	160
5.1.3.305 EllipticK_print_latex()	160
5.1.3.306 REGISTER_FUNCTION() [14/36]	160
5.1.3.307 EllipticE_evalf()	160
5.1.3.308 EllipticE_eval()	161
5.1.3.309 EllipticE_deriv()	161
5.1.3.310 EllipticE_series()	161
5.1.3.311 EllipticE_print_latex()	161
5.1.3.312 REGISTER_FUNCTION() [15/36]	161
5.1.3.313 iterated_integral_evalf_impl()	162
5.1.3.314 iterated_integral2_evalf()	162
5.1.3.315 iterated_integral3_evalf()	162
5.1.3.316 iterated_integral2_eval()	162
5.1.3.317 iterated_integral3_eval()	163
5.1.3.318 lgamma_evalf()	163
5.1.3.319 lgamma_eval()	163
5.1.3.320 lgamma_deriv()	163
5.1.3.321 lgamma_series()	164
5.1.3.322 lgamma_conjugate()	164
5.1.3.323 REGISTER_FUNCTION() [16/36]	164
5.1.3.324 tgamma_evalf()	164
5.1.3.325 tgamma_eval()	164
5.1.3.326 tgamma_deriv()	165
5.1.3.327 tgamma_series()	165
5.1.3.328 tgamma_conjugate()	165
5.1.3.329 REGISTER_FUNCTION() [17/36]	165

5.1.3.330 beta_evalf()	166
5.1.3.331 beta_eval()	166
5.1.3.332 beta_deriv()	166
5.1.3.333 beta_series()	166
5.1.3.334 REGISTER_FUNCTION() [18/36]	166
5.1.3.335 psi1_evalf()	167
5.1.3.336 psi1_eval()	167
5.1.3.337 psi1_deriv()	167
5.1.3.338 psi1_series()	167
5.1.3.339 psi2_evalf()	168
5.1.3.340 psi2_eval()	168
5.1.3.341 psi2_deriv()	168
5.1.3.342 psi2_series()	168
5.1.3.343 G2_evalf()	169
5.1.3.344 G2_eval()	169
5.1.3.345 G3_evalf()	169
5.1.3.346 G3_eval()	169
5.1.3.347 Li_evalf()	170
5.1.3.348 Li_eval()	170
5.1.3.349 Li_series()	170
5.1.3.350 Li_deriv()	170
5.1.3.351 Li_print_latex()	171
5.1.3.352 REGISTER_FUNCTION() [19/36]	171
5.1.3.353 S_evalf()	171
5.1.3.354 S_eval()	171
5.1.3.355 S_series()	171
5.1.3.356 S_deriv()	172
5.1.3.357 S_print_latex()	172
5.1.3.358 REGISTER_FUNCTION() [20/36]	172
5.1.3.359 H_evalf()	172

5.1.3.360 <code>H_eval()</code> . . . . .	172
5.1.3.361 <code>H_series()</code> . . . . .	173
5.1.3.362 <code>H_deriv()</code> . . . . .	173
5.1.3.363 <code>H_print_latex()</code> . . . . .	173
5.1.3.364 <code>REGISTER_FUNCTION()</code> [21/36] . . . . .	173
5.1.3.365 <code>zeta1_evalf()</code> . . . . .	173
5.1.3.366 <code>zeta1_eval()</code> . . . . .	174
5.1.3.367 <code>zeta1_deriv()</code> . . . . .	174
5.1.3.368 <code>zeta1_print_latex()</code> . . . . .	174
5.1.3.369 <code>zeta2_evalf()</code> . . . . .	174
5.1.3.370 <code>zeta2_eval()</code> . . . . .	174
5.1.3.371 <code>zeta2_deriv()</code> . . . . .	175
5.1.3.372 <code>zeta2_print_latex()</code> . . . . .	175
5.1.3.373 <code>exp_evalf()</code> . . . . .	175
5.1.3.374 <code>exp_eval()</code> . . . . .	175
5.1.3.375 <code>exp_expand()</code> . . . . .	175
5.1.3.376 <code>exp_deriv()</code> . . . . .	176
5.1.3.377 <code>exp_real_part()</code> . . . . .	176
5.1.3.378 <code>exp_imag_part()</code> . . . . .	176
5.1.3.379 <code>exp_conjugate()</code> . . . . .	176
5.1.3.380 <code>exp_power()</code> . . . . .	176
5.1.3.381 <code>REGISTER_FUNCTION()</code> [22/36] . . . . .	177
5.1.3.382 <code>log_evalf()</code> . . . . .	177
5.1.3.383 <code>log_eval()</code> . . . . .	177
5.1.3.384 <code>log_deriv()</code> . . . . .	177
5.1.3.385 <code>log_series()</code> . . . . .	177
5.1.3.386 <code>log_real_part()</code> . . . . .	178
5.1.3.387 <code>log_imag_part()</code> . . . . .	178
5.1.3.388 <code>log_expand()</code> . . . . .	178
5.1.3.389 <code>log_conjugate()</code> . . . . .	178

5.1.3.390 REGISTER_FUNCTION() [23/36]	178
5.1.3.391 sin_evalf()	179
5.1.3.392 sin_eval()	179
5.1.3.393 sin_deriv()	179
5.1.3.394 sin_real_part()	179
5.1.3.395 sin_imag_part()	179
5.1.3.396 sin_conjugate()	180
5.1.3.397 REGISTER_FUNCTION() [24/36]	180
5.1.3.398 cos_evalf()	180
5.1.3.399 cos_eval()	180
5.1.3.400 cos_deriv()	180
5.1.3.401 cos_real_part()	181
5.1.3.402 cos_imag_part()	181
5.1.3.403 cos_conjugate()	181
5.1.3.404 REGISTER_FUNCTION() [25/36]	181
5.1.3.405 tan_evalf()	181
5.1.3.406 tan_eval()	182
5.1.3.407 tan_deriv()	182
5.1.3.408 tan_real_part()	182
5.1.3.409 tan_imag_part()	182
5.1.3.410 tan_series()	182
5.1.3.411 tan_conjugate()	183
5.1.3.412 REGISTER_FUNCTION() [26/36]	183
5.1.3.413 asin_evalf()	183
5.1.3.414 asin_eval()	183
5.1.3.415 asin_deriv()	183
5.1.3.416 asin_conjugate()	184
5.1.3.417 REGISTER_FUNCTION() [27/36]	184
5.1.3.418 acos_evalf()	184
5.1.3.419 acos_eval()	184



5.1.3.420 <code>acos_deriv()</code> . . . . .	184
5.1.3.421 <code>acos_conjugate()</code> . . . . .	185
5.1.3.422 <code>REGISTER_FUNCTION()</code> [28/36] . . . . .	185
5.1.3.423 <code>atan_evalf()</code> . . . . .	185
5.1.3.424 <code>atan_eval()</code> . . . . .	185
5.1.3.425 <code>atan_deriv()</code> . . . . .	185
5.1.3.426 <code>atan_series()</code> . . . . .	186
5.1.3.427 <code>atan_conjugate()</code> . . . . .	186
5.1.3.428 <code>REGISTER_FUNCTION()</code> [29/36] . . . . .	186
5.1.3.429 <code>atan2_evalf()</code> . . . . .	186
5.1.3.430 <code>atan2_eval()</code> . . . . .	186
5.1.3.431 <code>atan2_deriv()</code> . . . . .	187
5.1.3.432 <code>REGISTER_FUNCTION()</code> [30/36] . . . . .	187
5.1.3.433 <code>sinh_evalf()</code> . . . . .	187
5.1.3.434 <code>sinh_eval()</code> . . . . .	187
5.1.3.435 <code>sinh_deriv()</code> . . . . .	187
5.1.3.436 <code>sinh_real_part()</code> . . . . .	188
5.1.3.437 <code>sinh_imag_part()</code> . . . . .	188
5.1.3.438 <code>sinh_conjugate()</code> . . . . .	188
5.1.3.439 <code>REGISTER_FUNCTION()</code> [31/36] . . . . .	188
5.1.3.440 <code>cosh_evalf()</code> . . . . .	188
5.1.3.441 <code>cosh_eval()</code> . . . . .	189
5.1.3.442 <code>cosh_deriv()</code> . . . . .	189
5.1.3.443 <code>cosh_real_part()</code> . . . . .	189
5.1.3.444 <code>cosh_imag_part()</code> . . . . .	189
5.1.3.445 <code>cosh_conjugate()</code> . . . . .	189
5.1.3.446 <code>REGISTER_FUNCTION()</code> [32/36] . . . . .	190
5.1.3.447 <code>tanh_evalf()</code> . . . . .	190
5.1.3.448 <code>tanh_eval()</code> . . . . .	190
5.1.3.449 <code>tanh_deriv()</code> . . . . .	190

5.1.3.450 tanh_series()	190
5.1.3.451 tanh_real_part()	191
5.1.3.452 tanh_imag_part()	191
5.1.3.453 tanh_conjugate()	191
5.1.3.454 REGISTER_FUNCTION() [33/36]	191
5.1.3.455 asinh_evalf()	191
5.1.3.456 asinh_eval()	192
5.1.3.457 asinh_deriv()	192
5.1.3.458 asinh_conjugate()	192
5.1.3.459 REGISTER_FUNCTION() [34/36]	192
5.1.3.460 acosh_evalf()	192
5.1.3.461 acosh_eval()	193
5.1.3.462 acosh_deriv()	193
5.1.3.463 acosh_conjugate()	193
5.1.3.464 REGISTER_FUNCTION() [35/36]	193
5.1.3.465 atanh_evalf()	193
5.1.3.466 atanh_eval()	194
5.1.3.467 atanh_deriv()	194
5.1.3.468 atanh_series()	194
5.1.3.469 atanh_conjugate()	194
5.1.3.470 REGISTER_FUNCTION() [36/36]	194
5.1.3.471 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [10/32]	195
5.1.3.472 subsvalue()	195
5.1.3.473 adaptivesimpson()	195
5.1.3.474 GINAC_BIND_UNARCHIVER() [21/49]	195
5.1.3.475 GINAC_DECLARE_UNARCHIVER() [22/51]	196
5.1.3.476 ifactor()	196
5.1.3.477 is_discriminant_of_quadratic_number_field()	196
5.1.3.478 kronecker_symbol()	196
5.1.3.479 primitive_dirichlet_character()	197

5.1.3.480 <code>dirichlet_character()</code> . . . . .	197
5.1.3.481 <code>generalised_Bernoulli_number()</code> . . . . .	197
5.1.3.482 <code>Bernoulli_polynomial()</code> . . . . .	198
5.1.3.483 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [11/32] . . . . .	198
5.1.3.484 <code>GINAC_BIND_UNARCHIVER()</code> [22/49] . . . . .	198
5.1.3.485 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [12/32] . . . . .	198
5.1.3.486 <code>GINAC_BIND_UNARCHIVER()</code> [23/49] . . . . .	198
5.1.3.487 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [13/32] . . . . .	199
5.1.3.488 <code>GINAC_BIND_UNARCHIVER()</code> [24/49] . . . . .	199
5.1.3.489 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [14/32] . . . . .	199
5.1.3.490 <code>GINAC_BIND_UNARCHIVER()</code> [25/49] . . . . .	199
5.1.3.491 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [15/32] . . . . .	199
5.1.3.492 <code>GINAC_BIND_UNARCHIVER()</code> [26/49] . . . . .	199
5.1.3.493 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [16/32] . . . . .	200
5.1.3.494 <code>GINAC_BIND_UNARCHIVER()</code> [27/49] . . . . .	200
5.1.3.495 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [17/32] . . . . .	200
5.1.3.496 <code>GINAC_BIND_UNARCHIVER()</code> [28/49] . . . . .	200
5.1.3.497 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [18/32] . . . . .	200
5.1.3.498 <code>GINAC_BIND_UNARCHIVER()</code> [29/49] . . . . .	200
5.1.3.499 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [19/32] . . . . .	201
5.1.3.500 <code>GINAC_BIND_UNARCHIVER()</code> [30/49] . . . . .	201
5.1.3.501 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [20/32] . . . . .	201
5.1.3.502 <code>GINAC_BIND_UNARCHIVER()</code> [31/49] . . . . .	201
5.1.3.503 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [21/32] . . . . .	201
5.1.3.504 <code>GINAC_BIND_UNARCHIVER()</code> [32/49] . . . . .	201
5.1.3.505 <code>GINAC_DECLARE_UNARCHIVER()</code> [23/51] . . . . .	202
5.1.3.506 <code>GINAC_DECLARE_UNARCHIVER()</code> [24/51] . . . . .	202
5.1.3.507 <code>GINAC_DECLARE_UNARCHIVER()</code> [25/51] . . . . .	202
5.1.3.508 <code>GINAC_DECLARE_UNARCHIVER()</code> [26/51] . . . . .	202
5.1.3.509 <code>GINAC_DECLARE_UNARCHIVER()</code> [27/51] . . . . .	202

5.1.3.510 GINAC_DECLARE_UNARCHIVER() [28/51]	202
5.1.3.511 GINAC_DECLARE_UNARCHIVER() [29/51]	202
5.1.3.512 GINAC_DECLARE_UNARCHIVER() [30/51]	203
5.1.3.513 GINAC_DECLARE_UNARCHIVER() [31/51]	203
5.1.3.514 GINAC_DECLARE_UNARCHIVER() [32/51]	203
5.1.3.515 GINAC_DECLARE_UNARCHIVER() [33/51]	203
5.1.3.516 GINAC_DECLARE_UNARCHIVER() [34/51]	203
5.1.3.517 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [22/32]	203
5.1.3.518 GINAC_BIND_UNARCHIVER() [33/49]	204
5.1.3.519 lst_to_matrix()	204
5.1.3.520 diag_matrix() [1/2]	204
5.1.3.521 diag_matrix() [2/2]	204
5.1.3.522 unit_matrix() [1/2]	204
5.1.3.523 symbolic_matrix() [1/2]	205
5.1.3.524 reduced_matrix()	205
5.1.3.525 sub_matrix()	205
5.1.3.526 GINAC_DECLARE_UNARCHIVER() [35/51]	205
5.1.3.527 nops() [2/2]	206
5.1.3.528 expand() [2/2]	206
5.1.3.529 evalf() [2/2]	206
5.1.3.530 rows()	206
5.1.3.531 cols()	206
5.1.3.532 transpose()	207
5.1.3.533 determinant()	207
5.1.3.534 trace()	207
5.1.3.535 charpoly()	207
5.1.3.536 inverse() [1/3]	207
5.1.3.537 inverse() [2/3]	208
5.1.3.538 rank() [1/2]	208
5.1.3.539 rank() [2/2]	208

5.1.3.540 unit_matrix() [2/2]	208
5.1.3.541 symbolic_matrix() [2/2]	208
5.1.3.542 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [23/32]	209
5.1.3.543 tryfactsubs()	209
5.1.3.544 algebraic_match_mul_with_mul()	209
5.1.3.545 GINAC_BIND_UNARCHIVER() [34/49]	209
5.1.3.546 GINAC_DECLARE_UNARCHIVER() [36/51]	210
5.1.3.547 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [24/32]	210
5.1.3.548 reeval_ncmul()	210
5.1.3.549 hold_ncmul()	210
5.1.3.550 GINAC_BIND_UNARCHIVER() [35/49]	210
5.1.3.551 GINAC_DECLARE_UNARCHIVER() [37/51]	210
5.1.3.552 get_first_symbol()	210
5.1.3.553 add_symbol()	211
5.1.3.554 collect_symbols()	211
5.1.3.555 get_symbol_stats()	211
5.1.3.556 lcmcoeff()	212
5.1.3.557 lcm_of_coefficients_denominators()	212
5.1.3.558 multiply_lcm()	213
5.1.3.559 quo()	213
5.1.3.560 rem()	214
5.1.3.561 decomp_rational()	214
5.1.3.562 prem()	215
5.1.3.563 sprem()	215
5.1.3.564 divide()	216
5.1.3.565 divide_in_z()	216
5.1.3.566 sr_gcd()	217
5.1.3.567 interpolate()	218
5.1.3.568 heur_gcd_z()	218
5.1.3.569 heur_gcd()	219

5.1.3.570 gcd_pf_pow()	220
5.1.3.571 gcd_pf_mul()	220
5.1.3.572 gcd() <sup>[1/2]</sup>	220
5.1.3.573 gcd_pf_pow_pow()	221
5.1.3.574 lcm() <sup>[1/2]</sup>	221
5.1.3.575 sqrfree_yun()	222
5.1.3.576 sqrfree()	222
5.1.3.577 sqrfree_parfrac()	223
5.1.3.578 replace_with_symbol() <sup>[1/2]</sup>	223
5.1.3.579 replace_with_symbol() <sup>[2/2]</sup>	224
5.1.3.580 frac_cancel()	224
5.1.3.581 find_common_factor()	225
5.1.3.582 collect_common_factors()	225
5.1.3.583 resultant()	226
5.1.3.584 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() <sup>[25/32]</sup>	226
5.1.3.585 make_real_float()	226
5.1.3.586 read_real_float()	227
5.1.3.587 GINAC_BIND_UNARCHIVER() <sup>[36/49]</sup>	227
5.1.3.588 write_real_float()	227
5.1.3.589 print_real_number()	227
5.1.3.590 print_integer_csrc()	228
5.1.3.591 print_real_csrc()	228
5.1.3.592 coerce()	228
5.1.3.593 coerce< int, cln::cl_I >()	229
5.1.3.594 coerce< unsigned int, cln::cl_I >()	229
5.1.3.595 print_real_cl_N()	229
5.1.3.596 exp()	229
5.1.3.597 log()	230
5.1.3.598 sin()	230
5.1.3.599 cos()	231

5.1.3.600 tan()	231
5.1.3.601 asin()	231
5.1.3.602 acos()	232
5.1.3.603 atan() [1/2]	232
5.1.3.604 atan() [2/2]	232
5.1.3.605 sinh()	233
5.1.3.606 cosh()	233
5.1.3.607 tanh()	234
5.1.3.608 asinh()	234
5.1.3.609 acosh()	234
5.1.3.610 atanh()	235
5.1.3.611 Li2_series() [2/2]	235
5.1.3.612 Li2_projection()	235
5.1.3.613 Li2_()	236
5.1.3.614 Li2()	236
5.1.3.615 zeta() [3/3]	236
5.1.3.616 guess_precision()	236
5.1.3.617 lgamma() [1/2]	237
5.1.3.618 lgamma() [2/2]	237
5.1.3.619 tgamma() [1/2]	237
5.1.3.620 tgamma() [2/2]	237
5.1.3.621 psi() [3/4]	238
5.1.3.622 psi() [4/4]	238
5.1.3.623 factorial()	238
5.1.3.624 doublefactorial()	238
5.1.3.625 binomial()	239
5.1.3.626 bernoulli()	239
5.1.3.627 fibonacci()	240
5.1.3.628 abs()	240
5.1.3.629 mod()	241

5.1.3.630 smod()	241
5.1.3.631 irem() [1/2]	241
5.1.3.632 irem() [2/2]	242
5.1.3.633 iquo() [1/2]	242
5.1.3.634 iquo() [2/2]	243
5.1.3.635 gcd() [2/2]	243
5.1.3.636 lcm() [2/2]	244
5.1.3.637 sqrt() [1/2]	244
5.1.3.638 isqrt()	244
5.1.3.639 PiEvalf()	245
5.1.3.640 EulerEvalf()	245
5.1.3.641 CatalanEvalf()	245
5.1.3.642 operator<<() [6/16]	245
5.1.3.643 GINAC_DECLARE_UNARCHIVER() [38/51]	245
5.1.3.644 pow() [1/3]	246
5.1.3.645 inverse() [3/3]	246
5.1.3.646 step()	246
5.1.3.647 csgn()	246
5.1.3.648 is_zero() [2/2]	247
5.1.3.649 is_positive()	247
5.1.3.650 is_negative()	247
5.1.3.651 is_integer()	247
5.1.3.652 is_pos_integer()	247
5.1.3.653 is_nonneg_integer()	248
5.1.3.654 is_even()	248
5.1.3.655 is_odd()	248
5.1.3.656 is_prime()	248
5.1.3.657 is_rational()	248
5.1.3.658 is_real()	249
5.1.3.659 is_cinteger()	249



5.1.3.660 <code>is_crational()</code> . . . . .	249
5.1.3.661 <code>to_int()</code> . . . . .	249
5.1.3.662 <code>to_long()</code> . . . . .	250
5.1.3.663 <code>to_double()</code> . . . . .	250
5.1.3.664 <code>real()</code> . . . . .	250
5.1.3.665 <code>imag()</code> . . . . .	250
5.1.3.666 <code>numer()</code> [2/2] . . . . .	250
5.1.3.667 <code>denom()</code> [2/2] . . . . .	251
5.1.3.668 <code>exadd()</code> . . . . .	251
5.1.3.669 <code>exmul()</code> . . . . .	251
5.1.3.670 <code>exminus()</code> . . . . .	251
5.1.3.671 <code>operator+()</code> [1/4] . . . . .	252
5.1.3.672 <code>operator-()</code> [1/4] . . . . .	252
5.1.3.673 <code>operator*()</code> [1/2] . . . . .	252
5.1.3.674 <code>operator/()</code> [1/2] . . . . .	252
5.1.3.675 <code>operator+()</code> [2/4] . . . . .	252
5.1.3.676 <code>operator-()</code> [2/4] . . . . .	253
5.1.3.677 <code>operator*()</code> [2/2] . . . . .	253
5.1.3.678 <code>operator/()</code> [2/2] . . . . .	253
5.1.3.679 <code>operator+=()</code> [1/2] . . . . .	253
5.1.3.680 <code>operator-=()</code> [1/2] . . . . .	253
5.1.3.681 <code>operator*=()</code> [1/2] . . . . .	254
5.1.3.682 <code>operator/=()</code> [1/2] . . . . .	254
5.1.3.683 <code>operator+=()</code> [2/2] . . . . .	254
5.1.3.684 <code>operator-=()</code> [2/2] . . . . .	254
5.1.3.685 <code>operator*=()</code> [2/2] . . . . .	254
5.1.3.686 <code>operator/=()</code> [2/2] . . . . .	255
5.1.3.687 <code>operator+()</code> [3/4] . . . . .	255
5.1.3.688 <code>operator-()</code> [3/4] . . . . .	255
5.1.3.689 <code>operator+()</code> [4/4] . . . . .	255

5.1.3.690 operator-() [4/4]	255
5.1.3.691 operator++() [1/4]	255
5.1.3.692 operator--() [1/4]	256
5.1.3.693 operator++() [2/4]	256
5.1.3.694 operator--() [2/4]	256
5.1.3.695 operator++() [3/4]	256
5.1.3.696 operator--() [3/4]	257
5.1.3.697 operator++() [4/4]	257
5.1.3.698 operator--() [4/4]	257
5.1.3.699 operator==()	257
5.1.3.700 operator!=()	258
5.1.3.701 operator<()	258
5.1.3.702 operator<=()	258
5.1.3.703 operator>()	258
5.1.3.704 operator>=()	258
5.1.3.705 my_ios_index()	259
5.1.3.706 my_ios_callback()	259
5.1.3.707 get_print_context()	259
5.1.3.708 set_print_context()	259
5.1.3.709 get_print_options()	259
5.1.3.710 set_print_options()	260
5.1.3.711 operator<<() [7/16]	260
5.1.3.712 operator>>() [3/3]	260
5.1.3.713 dflt()	260
5.1.3.714 latex()	260
5.1.3.715 python()	261
5.1.3.716 python_repr()	261
5.1.3.717 tree()	261
5.1.3.718 csrc()	261
5.1.3.719 csrc_float()	261

5.1.3.720 <code>csrc_double()</code> . . . . .	262
5.1.3.721 <code>csrc_cl_N()</code> . . . . .	262
5.1.3.722 <code>index_dimensions()</code> . . . . .	262
5.1.3.723 <code>no_index_dimensions()</code> . . . . .	262
5.1.3.724 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [26/32] . . . . .	262
5.1.3.725 <code>print_sym_pow()</code> . . . . .	263
5.1.3.726 <code>GINAC_BIND_UNARCHIVER()</code> [37/49] . . . . .	263
5.1.3.727 <code>GINAC_DECLARE_UNARCHIVER()</code> [39/51] . . . . .	263
5.1.3.728 <code>pow()</code> [2/3] . . . . .	263
5.1.3.729 <code>pow()</code> [3/3] . . . . .	263
5.1.3.730 <code>sqrt()</code> [2/2] . . . . .	264
5.1.3.731 <code>is_a()</code> [3/3] . . . . .	264
5.1.3.732 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [27/32] . . . . .	264
5.1.3.733 <code>GINAC_BIND_UNARCHIVER()</code> [38/49] . . . . .	264
5.1.3.734 <code>GINAC_DECLARE_UNARCHIVER()</code> [40/51] . . . . .	264
5.1.3.735 <code>series_to_poly()</code> . . . . .	264
5.1.3.736 <code>is_terminating()</code> . . . . .	265
5.1.3.737 <code>make_return_type_t()</code> . . . . .	265
5.1.3.738 <code>set_print_func()</code> [1/2] . . . . .	265
5.1.3.739 <code>set_print_func()</code> [2/2] . . . . .	266
5.1.3.740 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [28/32] . . . . .	266
5.1.3.741 <code>GINAC_BIND_UNARCHIVER()</code> [39/49] . . . . .	266
5.1.3.742 <code>print_operator()</code> . . . . .	266
5.1.3.743 <code>GINAC_DECLARE_UNARCHIVER()</code> [41/51] . . . . .	266
5.1.3.744 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [29/32] . . . . .	267
5.1.3.745 <code>get_default_TeX_name()</code> . . . . .	267
5.1.3.746 <code>GINAC_BIND_UNARCHIVER()</code> [40/49] . . . . .	267
5.1.3.747 <code>GINAC_BIND_UNARCHIVER()</code> [41/49] . . . . .	267
5.1.3.748 <code>GINAC_BIND_UNARCHIVER()</code> [42/49] . . . . .	267
5.1.3.749 <code>GINAC_DECLARE_UNARCHIVER()</code> [42/51] . . . . .	268

5.1.3.750 GINAC_DECLARE_UNARCHIVER() [43/51]	268
5.1.3.751 GINAC_DECLARE_UNARCHIVER() [44/51]	268
5.1.3.752 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [30/32]	268
5.1.3.753 GINAC_BIND_UNARCHIVER() [43/49]	268
5.1.3.754 index0()	268
5.1.3.755 index1()	269
5.1.3.756 index2()	269
5.1.3.757 index3()	269
5.1.3.758 not_symmetric()	269
5.1.3.759 symmetric2()	269
5.1.3.760 symmetric3()	269
5.1.3.761 symmetric4()	270
5.1.3.762 antisymmetric2()	270
5.1.3.763 antisymmetric3()	270
5.1.3.764 antisymmetric4()	270
5.1.3.765 canonicalize()	270
5.1.3.766 symm()	271
5.1.3.767 symmetrize() [3/4]	271
5.1.3.768 antisymmetrize() [3/4]	271
5.1.3.769 symmetrize_cyclic() [3/4]	272
5.1.3.770 GINAC_DECLARE_UNARCHIVER() [45/51]	272
5.1.3.771 sy_none() [1/4]	272
5.1.3.772 sy_none() [2/4]	272
5.1.3.773 sy_none() [3/4]	272
5.1.3.774 sy_none() [4/4]	273
5.1.3.775 sy_symm() [1/4]	273
5.1.3.776 sy_symm() [2/4]	273
5.1.3.777 sy_symm() [3/4]	273
5.1.3.778 sy_symm() [4/4]	273
5.1.3.779 sy_anti() [1/4]	274

5.1.3.780 <code>sy_anti()</code> [2/4]	274
5.1.3.781 <code>sy_anti()</code> [3/4]	274
5.1.3.782 <code>sy_anti()</code> [4/4]	274
5.1.3.783 <code>sy_cycl()</code> [1/4]	274
5.1.3.784 <code>sy_cycl()</code> [2/4]	275
5.1.3.785 <code>sy_cycl()</code> [3/4]	275
5.1.3.786 <code>sy_cycl()</code> [4/4]	275
5.1.3.787 <code>symmetrize()</code> [4/4]	275
5.1.3.788 <code>antisymmetrize()</code> [4/4]	275
5.1.3.789 <code>symmetrize_cyclic()</code> [4/4]	276
5.1.3.790 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [31/32]	276
5.1.3.791 <code>print_func&lt; print_dflt &gt;()</code> [3/3]	276
5.1.3.792 <code>GINAC_BIND_UNARCHIVER()</code> [44/49]	276
5.1.3.793 <code>GINAC_BIND_UNARCHIVER()</code> [45/49]	276
5.1.3.794 <code>GINAC_BIND_UNARCHIVER()</code> [46/49]	276
5.1.3.795 <code>GINAC_BIND_UNARCHIVER()</code> [47/49]	277
5.1.3.796 <code>GINAC_BIND_UNARCHIVER()</code> [48/49]	277
5.1.3.797 <code>delta_tensor()</code>	277
5.1.3.798 <code>metric_tensor()</code>	277
5.1.3.799 <code>lorentz_g()</code>	278
5.1.3.800 <code>spinor_metric()</code>	278
5.1.3.801 <code>epsilon_tensor()</code> [1/2]	279
5.1.3.802 <code>epsilon_tensor()</code> [2/2]	279
5.1.3.803 <code>lorentz_eps()</code>	280
5.1.3.804 <code>GINAC_DECLARE_UNARCHIVER()</code> [46/51]	280
5.1.3.805 <code>GINAC_DECLARE_UNARCHIVER()</code> [47/51]	280
5.1.3.806 <code>GINAC_DECLARE_UNARCHIVER()</code> [48/51]	281
5.1.3.807 <code>GINAC_DECLARE_UNARCHIVER()</code> [49/51]	281
5.1.3.808 <code>GINAC_DECLARE_UNARCHIVER()</code> [50/51]	281
5.1.3.809 <code>log2()</code>	281

5.1.3.810	<code>multinomial_coefficient()</code>	281
5.1.3.811	<code>rotate_left()</code>	282
5.1.3.812	<code>compare_pointers()</code>	282
5.1.3.813	<code>golden_ratio_hash()</code>	282
5.1.3.814	<code>permutation_sign()</code> [1/2]	283
5.1.3.815	<code>permutation_sign()</code> [2/2]	283
5.1.3.816	<code>shaker_sort()</code>	283
5.1.3.817	<code>cyclic_permutation()</code>	283
5.1.3.818	<code>format_index_value()</code> [1/2]	284
5.1.3.819	<code>format_index_value()</code> [2/2]	284
5.1.3.820	<code>operator&lt;&lt;()</code> [8/16]	284
5.1.3.821	<code>operator&lt;&lt;()</code> [9/16]	284
5.1.3.822	<code>operator&lt;&lt;()</code> [10/16]	285
5.1.3.823	<code>operator&lt;&lt;()</code> [11/16]	285
5.1.3.824	<code>operator&lt;&lt;()</code> [12/16]	285
5.1.3.825	<code>operator&lt;&lt;()</code> [13/16]	285
5.1.3.826	<code>operator&lt;&lt;()</code> [14/16]	286
5.1.3.827	<code>operator&lt;&lt;()</code> [15/16]	286
5.1.3.828	<code>operator&lt;&lt;()</code> [16/16]	286
5.1.3.829	<code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [32/32]	286
5.1.3.830	<code>GINAC_BIND_UNARCHIVER()</code> [49/49]	287
5.1.3.831	<code>haswild()</code>	287
5.1.3.832	<code>GINAC_DECLARE_UNARCHIVER()</code> [51/51]	287
5.1.3.833	<code>wild()</code>	287
5.1.4	Variable Documentation	287
5.1.4.1	<code>unarch_table_instance</code>	287
5.1.4.2	<code>map_evalm</code>	288
5.1.4.3	<code>map_eval_integ</code>	288
5.1.4.4	<code>tensor</code>	288
5.1.4.5	<code>Pi</code>	288

5.1.4.6	Euler	288
5.1.4.7	Catalan	289
5.1.4.8	crctab	289
5.1.4.9	library_initializer	289
5.1.4.10	_num0_bp	289
5.1.4.11	idx	289
5.1.4.12	force_include_tgamma	289
5.1.4.13	force_include_zeta1	290
5.1.4.14	l	290
5.1.4.15	Digits	290
5.1.4.16	next_print_context_id	290
5.1.4.17	version_major	290
5.1.4.18	version_minor	291
5.1.4.19	version_micro	291
5.1.4.20	_num_120_p	291
5.1.4.21	_ex_120	291
5.1.4.22	_num_60_p	291
5.1.4.23	_ex_60	291
5.1.4.24	_num_48_p	292
5.1.4.25	_ex_48	292
5.1.4.26	_num_30_p	292
5.1.4.27	_ex_30	292
5.1.4.28	_num_25_p	292
5.1.4.29	_ex_25	292
5.1.4.30	_num_24_p	293
5.1.4.31	_ex_24	293
5.1.4.32	_num_20_p	293
5.1.4.33	_ex_20	293
5.1.4.34	_num_18_p	293
5.1.4.35	_ex_18	293

5.1.4.36	_num_15_p	294
5.1.4.37	_ex_15	294
5.1.4.38	_num_12_p	294
5.1.4.39	_ex_12	294
5.1.4.40	_num_11_p	294
5.1.4.41	_ex_11	294
5.1.4.42	_num_10_p	295
5.1.4.43	_ex_10	295
5.1.4.44	_num_9_p	295
5.1.4.45	_ex_9	295
5.1.4.46	_num_8_p	295
5.1.4.47	_ex_8	295
5.1.4.48	_num_7_p	296
5.1.4.49	_ex_7	296
5.1.4.50	_num_6_p	296
5.1.4.51	_ex_6	296
5.1.4.52	_num_5_p	296
5.1.4.53	_ex_5	296
5.1.4.54	_num_4_p	297
5.1.4.55	_ex_4	297
5.1.4.56	_num_3_p	297
5.1.4.57	_ex_3	297
5.1.4.58	_num_2_p	297
5.1.4.59	_ex_2	297
5.1.4.60	_num_1_p	298
5.1.4.61	_ex_1	298
5.1.4.62	_num_1_2_p	298
5.1.4.63	_ex_1_2	298
5.1.4.64	_num_1_3_p	298
5.1.4.65	_ex_1_3	299



5.1.4.66	<a href="#">_num_1_4_p</a>	299
5.1.4.67	<a href="#">_ex_1_4</a>	299
5.1.4.68	<a href="#">_num0_p</a>	299
5.1.4.69	<a href="#">_ex0</a>	299
5.1.4.70	<a href="#">_num1_4_p</a>	300
5.1.4.71	<a href="#">_ex1_4</a>	300
5.1.4.72	<a href="#">_num1_3_p</a>	300
5.1.4.73	<a href="#">_ex1_3</a>	300
5.1.4.74	<a href="#">_num1_2_p</a>	300
5.1.4.75	<a href="#">_ex1_2</a>	300
5.1.4.76	<a href="#">_num1_p</a>	301
5.1.4.77	<a href="#">_ex1</a>	301
5.1.4.78	<a href="#">_num2_p</a>	301
5.1.4.79	<a href="#">_ex2</a>	302
5.1.4.80	<a href="#">_num3_p</a>	302
5.1.4.81	<a href="#">_ex3</a>	302
5.1.4.82	<a href="#">_num4_p</a>	302
5.1.4.83	<a href="#">_ex4</a>	302
5.1.4.84	<a href="#">_num5_p</a>	303
5.1.4.85	<a href="#">_ex5</a>	303
5.1.4.86	<a href="#">_num6_p</a>	303
5.1.4.87	<a href="#">_ex6</a>	303
5.1.4.88	<a href="#">_num7_p</a>	303
5.1.4.89	<a href="#">_ex7</a>	303
5.1.4.90	<a href="#">_num8_p</a>	304
5.1.4.91	<a href="#">_ex8</a>	304
5.1.4.92	<a href="#">_num9_p</a>	304
5.1.4.93	<a href="#">_ex9</a>	304
5.1.4.94	<a href="#">_num10_p</a>	304
5.1.4.95	<a href="#">_ex10</a>	304

5.1.4.96	<a href="#">_num11_p</a>	305
5.1.4.97	<a href="#">_ex11</a>	305
5.1.4.98	<a href="#">_num12_p</a>	305
5.1.4.99	<a href="#">_ex12</a>	305
5.1.4.100	<a href="#">_num15_p</a>	305
5.1.4.101	<a href="#">_ex15</a>	305
5.1.4.102	<a href="#">_num18_p</a>	306
5.1.4.103	<a href="#">_ex18</a>	306
5.1.4.104	<a href="#">_num20_p</a>	306
5.1.4.105	<a href="#">_ex20</a>	306
5.1.4.106	<a href="#">_num24_p</a>	306
5.1.4.107	<a href="#">_ex24</a>	306
5.1.4.108	<a href="#">_num25_p</a>	307
5.1.4.109	<a href="#">_ex25</a>	307
5.1.4.110	<a href="#">_num30_p</a>	307
5.1.4.111	<a href="#">_ex30</a>	307
5.1.4.112	<a href="#">_num48_p</a>	307
5.1.4.113	<a href="#">_ex48</a>	307
5.1.4.114	<a href="#">_num60_p</a>	308
5.1.4.115	<a href="#">_ex60</a>	308
5.1.4.116	<a href="#">_num120_p</a>	308
5.1.4.117	<a href="#">_ex120</a>	308
5.2	<a href="#">GiNaC::internal Namespace Reference</a>	308
5.3	<a href="#">std Namespace Reference</a>	308
5.3.1	<a href="#">Function Documentation</a>	309
5.3.1.1	<a href="#">swap()</a>	309

<b>6</b>	<b>Class Documentation</b>	<b>311</b>
6.1	GiNaC::internal::_iter_rep Struct Reference	311
6.1.1	Constructor & Destructor Documentation	311
6.1.1.1	_iter_rep()	311
6.1.2	Member Function Documentation	311
6.1.2.1	operator==()	312
6.1.2.2	operator!=()	312
6.1.3	Member Data Documentation	312
6.1.3.1	e	312
6.1.3.2	i	312
6.1.3.3	i_end	312
6.2	GiNaC::_numeric_digits Class Reference	313
6.2.1	Detailed Description	313
6.2.2	Constructor & Destructor Documentation	313
6.2.2.1	_numeric_digits()	314
6.2.3	Member Function Documentation	314
6.2.3.1	operator=()	314
6.2.3.2	operator long()	314
6.2.3.3	print()	314
6.2.3.4	add_callback()	314
6.2.4	Member Data Documentation	315
6.2.4.1	digits	315
6.2.4.2	too_late	315
6.2.4.3	callbacklist	315
6.3	GiNaC::add Class Reference	315
6.3.1	Detailed Description	317
6.3.2	Constructor & Destructor Documentation	317
6.3.2.1	add() [1/6]	317
6.3.2.2	add() [2/6]	317
6.3.2.3	add() [3/6]	318

6.3.2.4	<a href="#">add()</a> [4/6]	318
6.3.2.5	<a href="#">add()</a> [5/6]	318
6.3.2.6	<a href="#">add()</a> [6/6]	318
6.3.3	<a href="#">Member Function Documentation</a>	318
6.3.3.1	<a href="#">precedence()</a>	319
6.3.3.2	<a href="#">info()</a>	319
6.3.3.3	<a href="#">is_polynomial()</a>	319
6.3.3.4	<a href="#">degree()</a>	320
6.3.3.5	<a href="#">ldegree()</a>	320
6.3.3.6	<a href="#">coeff()</a>	320
6.3.3.7	<a href="#">eval()</a>	320
6.3.3.8	<a href="#">evalm()</a>	321
6.3.3.9	<a href="#">series()</a>	321
6.3.3.10	<a href="#">normal()</a>	321
6.3.3.11	<a href="#">integer_content()</a>	322
6.3.3.12	<a href="#">smod()</a>	322
6.3.3.13	<a href="#">max_coefficient()</a>	322
6.3.3.14	<a href="#">conjugate()</a>	323
6.3.3.15	<a href="#">real_part()</a>	323
6.3.3.16	<a href="#">imag_part()</a>	323
6.3.3.17	<a href="#">get_free_indices()</a>	323
6.3.3.18	<a href="#">eval_ncmul()</a>	323
6.3.3.19	<a href="#">derivative()</a>	324
6.3.3.20	<a href="#">return_type()</a>	324
6.3.3.21	<a href="#">return_type_tinfo()</a>	324
6.3.3.22	<a href="#">thisexpairseq()</a> [1/2]	324
6.3.3.23	<a href="#">thisexpairseq()</a> [2/2]	325
6.3.3.24	<a href="#">split_ex_to_pair()</a>	325
6.3.3.25	<a href="#">combine_ex_with_coeff_to_pair()</a>	325
6.3.3.26	<a href="#">combine_pair_with_coeff_to_pair()</a>	325

6.3.3.27	<a href="#">recombine_pair_to_ex()</a>	326
6.3.3.28	<a href="#">expand()</a>	326
6.3.3.29	<a href="#">print_add()</a>	326
6.3.3.30	<a href="#">do_print()</a>	326
6.3.3.31	<a href="#">do_print_latex()</a>	327
6.3.3.32	<a href="#">do_print_csrc()</a>	327
6.3.3.33	<a href="#">do_print_python_repr()</a>	327
6.3.4	<a href="#">Friends And Related Function Documentation</a>	327
6.3.4.1	<a href="#">mul</a>	327
6.3.4.2	<a href="#">power</a>	327
6.4	<a href="#">GiNaC::archive Class Reference</a>	328
6.4.1	<a href="#">Detailed Description</a>	329
6.4.2	<a href="#">Member Typedef Documentation</a>	329
6.4.2.1	<a href="#">inv_at_cit</a>	329
6.4.3	<a href="#">Constructor &amp; Destructor Documentation</a>	329
6.4.3.1	<a href="#">archive() [1/3]</a>	329
6.4.3.2	<a href="#">~archive()</a>	330
6.4.3.3	<a href="#">archive() [2/3]</a>	330
6.4.3.4	<a href="#">archive() [3/3]</a>	330
6.4.4	<a href="#">Member Function Documentation</a>	330
6.4.4.1	<a href="#">archive_ex()</a>	330
6.4.4.2	<a href="#">unarchive_ex() [1/3]</a>	331
6.4.4.3	<a href="#">unarchive_ex() [2/3]</a>	331
6.4.4.4	<a href="#">unarchive_ex() [3/3]</a>	331
6.4.4.5	<a href="#">num_expressions()</a>	332
6.4.4.6	<a href="#">get_top_node()</a>	332
6.4.4.7	<a href="#">clear()</a>	332
6.4.4.8	<a href="#">add_node()</a>	332
6.4.4.9	<a href="#">get_node()</a>	333
6.4.4.10	<a href="#">forget()</a>	333

6.4.4.11	<a href="#">printraw()</a>	333
6.4.4.12	<a href="#">atomize()</a>	333
6.4.4.13	<a href="#">unatomize()</a>	334
6.4.5	<a href="#">Friends And Related Function Documentation</a>	334
6.4.5.1	<a href="#">operator&lt;&lt;</a>	334
6.4.5.2	<a href="#">operator&gt;&gt;</a>	334
6.4.6	<a href="#">Member Data Documentation</a>	334
6.4.6.1	<a href="#">nodes</a>	334
6.4.6.2	<a href="#">exprs</a>	335
6.4.6.3	<a href="#">atoms</a>	335
6.4.6.4	<a href="#">inverse_atoms</a>	335
6.4.6.5	<a href="#">exphtable</a>	335
6.5	<a href="#">GiNaC::archive_node Class Reference</a>	335
6.5.1	<a href="#">Detailed Description</a>	337
6.5.2	<a href="#">Member Typedef Documentation</a>	337
6.5.2.1	<a href="#">propinfovector</a>	337
6.5.2.2	<a href="#">archive_node_cit</a>	338
6.5.3	<a href="#">Member Enumeration Documentation</a>	338
6.5.3.1	<a href="#">property_type</a>	338
6.5.4	<a href="#">Constructor &amp; Destructor Documentation</a>	338
6.5.4.1	<a href="#">archive_node() [1/2]</a>	338
6.5.4.2	<a href="#">archive_node() [2/2]</a>	338
6.5.5	<a href="#">Member Function Documentation</a>	339
6.5.5.1	<a href="#">operator=()</a>	339
6.5.5.2	<a href="#">add_bool()</a>	339
6.5.5.3	<a href="#">add_unsigned()</a>	339
6.5.5.4	<a href="#">add_string()</a>	339
6.5.5.5	<a href="#">add_ex()</a>	340
6.5.5.6	<a href="#">find_bool()</a>	340
6.5.5.7	<a href="#">find_unsigned()</a>	340

6.5.5.8	<a href="#">find_string()</a>	341
6.5.5.9	<a href="#">find_first()</a>	341
6.5.5.10	<a href="#">find_last()</a>	341
6.5.5.11	<a href="#">find_property_range()</a>	341
6.5.5.12	<a href="#">find_ex()</a>	342
6.5.5.13	<a href="#">find_ex_by_loc()</a>	342
6.5.5.14	<a href="#">find_ex_node()</a>	342
6.5.5.15	<a href="#">get_properties()</a>	342
6.5.5.16	<a href="#">unarchive()</a>	343
6.5.5.17	<a href="#">has_same_ex_as()</a>	343
6.5.5.18	<a href="#">has_ex()</a>	343
6.5.5.19	<a href="#">get_ex()</a>	343
6.5.5.20	<a href="#">forget()</a>	343
6.5.5.21	<a href="#">printraw()</a>	344
6.5.6	<a href="#">Friends And Related Function Documentation</a>	344
6.5.6.1	<a href="#">operator&lt;&lt;</a>	344
6.5.6.2	<a href="#">operator&gt;&gt;</a>	344
6.5.7	<a href="#">Member Data Documentation</a>	344
6.5.7.1	<a href="#">a</a>	344
6.5.7.2	<a href="#">props</a>	345
6.5.7.3	<a href="#">has_expression</a>	345
6.5.7.4	<a href="#">e</a>	345
6.6	<a href="#">GiNaC::archive_node::archive_node_cit_range Struct Reference</a>	345
6.6.1	<a href="#">Member Data Documentation</a>	345
6.6.1.1	<a href="#">begin</a>	346
6.6.1.2	<a href="#">end</a>	346
6.7	<a href="#">GiNaC::archive::archived_ex Struct Reference</a>	346
6.7.1	<a href="#">Detailed Description</a>	346
6.7.2	<a href="#">Constructor &amp; Destructor Documentation</a>	346
6.7.2.1	<a href="#">archived_ex()</a> [1/2]	347

6.7.2.2	<a href="#">archived_ex()</a> [2/2]	347
6.7.3	<a href="#">Member Data Documentation</a>	347
6.7.3.1	<a href="#">name</a>	347
6.7.3.2	<a href="#">root</a>	347
6.8	<a href="#">GiNaC::basic Class Reference</a>	348
6.8.1	<a href="#">Detailed Description</a>	351
6.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	351
6.8.2.1	<a href="#">basic()</a> [1/2]	351
6.8.2.2	<a href="#">~basic()</a>	352
6.8.2.3	<a href="#">basic()</a> [2/2]	352
6.8.3	<a href="#">Member Function Documentation</a>	352
6.8.3.1	<a href="#">operator=()</a>	352
6.8.3.2	<a href="#">duplicate()</a>	352
6.8.3.3	<a href="#">eval()</a>	353
6.8.3.4	<a href="#">evalf()</a>	353
6.8.3.5	<a href="#">evalm()</a>	353
6.8.3.6	<a href="#">eval_integ()</a>	353
6.8.3.7	<a href="#">eval_ncmul()</a>	354
6.8.3.8	<a href="#">eval_indexed()</a>	354
6.8.3.9	<a href="#">print()</a>	354
6.8.3.10	<a href="#">dbgprint()</a>	355
6.8.3.11	<a href="#">dbgprinttree()</a>	355
6.8.3.12	<a href="#">precedence()</a>	355
6.8.3.13	<a href="#">info()</a>	355
6.8.3.14	<a href="#">nops()</a>	356
6.8.3.15	<a href="#">op()</a>	356
6.8.3.16	<a href="#">operator[]()</a> [1/4]	356
6.8.3.17	<a href="#">operator[]()</a> [2/4]	356
6.8.3.18	<a href="#">let_op()</a>	357
6.8.3.19	<a href="#">operator[]()</a> [3/4]	357



6.8.3.20	<code>operator[]()</code> [ 4 / 4 ]	357
6.8.3.21	<code>has()</code>	357
6.8.3.22	<code>match()</code>	358
6.8.3.23	<code>match_same_type()</code>	358
6.8.3.24	<code>subs()</code>	358
6.8.3.25	<code>map()</code>	359
6.8.3.26	<code>accept()</code>	359
6.8.3.27	<code>is_polynomial()</code>	359
6.8.3.28	<code>degree()</code>	359
6.8.3.29	<code>ldegree()</code>	360
6.8.3.30	<code>coeff()</code>	360
6.8.3.31	<code>expand()</code>	360
6.8.3.32	<code>collect()</code>	360
6.8.3.33	<code>derivative()</code>	361
6.8.3.34	<code>series()</code>	361
6.8.3.35	<code>normal()</code>	362
6.8.3.36	<code>to_rational()</code>	362
6.8.3.37	<code>to_polynomial()</code>	362
6.8.3.38	<code>integer_content()</code>	363
6.8.3.39	<code>smod()</code>	363
6.8.3.40	<code>max_coefficient()</code>	363
6.8.3.41	<code>get_free_indices()</code>	364
6.8.3.42	<code>add_indexed()</code>	364
6.8.3.43	<code>scalar_mul_indexed()</code>	364
6.8.3.44	<code>contract_with()</code>	365
6.8.3.45	<code>return_type()</code>	366
6.8.3.46	<code>return_type_tinfo()</code>	366
6.8.3.47	<code>conjugate()</code>	366
6.8.3.48	<code>real_part()</code>	366
6.8.3.49	<code>imag_part()</code>	366

6.8.3.50	<code>compare_same_type()</code>	367
6.8.3.51	<code>is_equal_same_type()</code>	367
6.8.3.52	<code>calchash()</code>	367
6.8.3.53	<code>print_dispatch()</code> [1/2]	368
6.8.3.54	<code>print_dispatch()</code> [2/2]	368
6.8.3.55	<code>archive()</code>	368
6.8.3.56	<code>read_archive()</code>	369
6.8.3.57	<code>subs_one_level()</code>	369
6.8.3.58	<code>diff()</code>	369
6.8.3.59	<code>compare()</code>	370
6.8.3.60	<code>is_equal()</code>	370
6.8.3.61	<code>hold()</code>	371
6.8.3.62	<code>gethash()</code>	371
6.8.3.63	<code>setflag()</code>	372
6.8.3.64	<code>clearflag()</code>	372
6.8.3.65	<code>ensure_if_modifiable()</code>	372
6.8.3.66	<code>do_print()</code>	373
6.8.3.67	<code>do_print_tree()</code>	373
6.8.3.68	<code>do_print_python_repr()</code>	373
6.8.4	Friends And Related Function Documentation	373
6.8.4.1	<code>ex</code>	373
6.8.5	Member Data Documentation	373
6.8.5.1	<code>flags</code>	374
6.8.5.2	<code>hashvalue</code>	374
6.9	GiNaC::basic_log_kernel Class Reference	374
6.9.1	Detailed Description	375
6.9.2	Member Function Documentation	375
6.9.2.1	<code>series_coeff_impl()</code>	375
6.9.2.2	<code>do_print()</code>	375
6.10	GiNaC::basic_multi_iterator< T > Class Template Reference	376

6.10.1 Detailed Description . . . . .	377
6.10.2 Constructor & Destructor Documentation . . . . .	377
6.10.2.1 basic_multi_iterator() [1/3] . . . . .	377
6.10.2.2 basic_multi_iterator() [2/3] . . . . .	377
6.10.2.3 basic_multi_iterator() [3/3] . . . . .	378
6.10.2.4 ~basic_multi_iterator() . . . . .	378
6.10.3 Member Function Documentation . . . . .	378
6.10.3.1 size() . . . . .	378
6.10.3.2 overflow() . . . . .	378
6.10.3.3 get_vector() . . . . .	378
6.10.3.4 operator[]() [1/2] . . . . .	379
6.10.3.5 operator[]() [2/2] . . . . .	379
6.10.3.6 operator()() [1/2] . . . . .	379
6.10.3.7 operator()() [2/2] . . . . .	379
6.10.3.8 init() . . . . .	379
6.10.3.9 operator++() . . . . .	380
6.10.4 Friends And Related Function Documentation . . . . .	380
6.10.4.1 operator<< . . . . .	380
6.10.5 Member Data Documentation . . . . .	380
6.10.5.1 N . . . . .	380
6.10.5.2 B . . . . .	380
6.10.5.3 v . . . . .	381
6.10.5.4 flag_overflow . . . . .	381
6.11 GiNaC::basic_partition_generator Class Reference . . . . .	381
6.11.1 Detailed Description . . . . .	382
6.11.2 Constructor & Destructor Documentation . . . . .	382
6.11.2.1 basic_partition_generator() . . . . .	382
6.11.3 Member Data Documentation . . . . .	382
6.11.3.1 mpgen . . . . .	382
6.12 GiNaC::class_info< OPT > Class Template Reference . . . . .	382

6.12.1	Constructor & Destructor Documentation . . . . .	383
6.12.1.1	class_info() . . . . .	383
6.12.2	Member Function Documentation . . . . .	383
6.12.2.1	get_parent() . . . . .	384
6.12.2.2	find() . . . . .	384
6.12.2.3	dump_hierarchy() . . . . .	384
6.12.2.4	dump_tree() . . . . .	384
6.12.2.5	identify_parents() . . . . .	385
6.12.3	Member Data Documentation . . . . .	385
6.12.3.1	options . . . . .	385
6.12.3.2	first . . . . .	385
6.12.3.3	next . . . . .	385
6.12.3.4	parent . . . . .	385
6.12.3.5	parents_identified . . . . .	386
6.13	GiNaC::clifford Class Reference . . . . .	386
6.13.1	Detailed Description . . . . .	387
6.13.2	Constructor & Destructor Documentation . . . . .	387
6.13.2.1	clifford() [1/4] . . . . .	388
6.13.2.2	clifford() [2/4] . . . . .	388
6.13.2.3	clifford() [3/4] . . . . .	388
6.13.2.4	clifford() [4/4] . . . . .	389
6.13.3	Member Function Documentation . . . . .	389
6.13.3.1	precedence() . . . . .	389
6.13.3.2	archive() . . . . .	389
6.13.3.3	read_archive() . . . . .	390
6.13.3.4	eval_ncmul() . . . . .	390
6.13.3.5	match_same_type() . . . . .	390
6.13.3.6	thiscontainer() [1/2] . . . . .	391
6.13.3.7	thiscontainer() [2/2] . . . . .	391
6.13.3.8	return_type() . . . . .	391

6.13.3.9	<a href="#">return_type_tinfo()</a>	391
6.13.3.10	<a href="#">get_representation_label()</a>	391
6.13.3.11	<a href="#">get_metric()</a> [1/2]	391
6.13.3.12	<a href="#">get_metric()</a> [2/2]	392
6.13.3.13	<a href="#">same_metric()</a>	392
6.13.3.14	<a href="#">get_commutator_sign()</a>	392
6.13.3.15	<a href="#">nops()</a>	392
6.13.3.16	<a href="#">op()</a>	393
6.13.3.17	<a href="#">let_op()</a>	393
6.13.3.18	<a href="#">subs()</a>	393
6.13.3.19	<a href="#">do_print_dflt()</a>	393
6.13.3.20	<a href="#">do_print_latex()</a>	394
6.13.3.21	<a href="#">do_print_tree()</a>	394
6.13.4	<a href="#">Member Data Documentation</a>	394
6.13.4.1	<a href="#">representation_label</a>	394
6.13.4.2	<a href="#">metric</a>	394
6.13.4.3	<a href="#">commutator_sign</a>	395
6.14	<a href="#">GiNaC::cliffordunit Class Reference</a>	395
6.14.1	<a href="#">Detailed Description</a>	395
6.14.2	<a href="#">Member Function Documentation</a>	396
6.14.2.1	<a href="#">contract_with()</a>	396
6.14.2.2	<a href="#">do_print()</a>	396
6.14.2.3	<a href="#">do_print_latex()</a>	396
6.15	<a href="#">GiNaC::color Class Reference</a>	396
6.15.1	<a href="#">Detailed Description</a>	397
6.15.2	<a href="#">Constructor &amp; Destructor Documentation</a>	397
6.15.2.1	<a href="#">color()</a> [1/4]	398
6.15.2.2	<a href="#">color()</a> [2/4]	398
6.15.2.3	<a href="#">color()</a> [3/4]	398
6.15.2.4	<a href="#">color()</a> [4/4]	398

6.15.3	Member Function Documentation	399
6.15.3.1	archive()	399
6.15.3.2	read_archive()	399
6.15.3.3	eval_ncmul()	399
6.15.3.4	match_same_type()	400
6.15.3.5	thiscontainer() [1/2]	400
6.15.3.6	thiscontainer() [2/2]	400
6.15.3.7	return_type()	400
6.15.3.8	return_type_tinfo()	401
6.15.3.9	get_representation_label()	401
6.15.4	Member Data Documentation	401
6.15.4.1	representation_label	401
6.16	GiNaC::compare_all_equal< T > Class Template Reference	401
6.16.1	Detailed Description	402
6.16.2	Constructor & Destructor Documentation	402
6.16.2.1	~compare_all_equal()	402
6.16.3	Member Function Documentation	402
6.16.3.1	struct_is_equal()	402
6.16.3.2	struct_compare()	402
6.17	GiNaC::compare_bitwise< T > Class Template Reference	403
6.17.1	Detailed Description	403
6.17.2	Constructor & Destructor Documentation	403
6.17.2.1	~compare_bitwise()	403
6.17.3	Member Function Documentation	403
6.17.3.1	struct_is_equal()	403
6.17.3.2	struct_compare()	404
6.18	GiNaC::compare_std_less< T > Class Template Reference	404
6.18.1	Detailed Description	404
6.18.2	Constructor & Destructor Documentation	404
6.18.2.1	~compare_std_less()	404

6.18.3	Member Function Documentation	404
6.18.3.1	struct_is_equal()	405
6.18.3.2	struct_compare()	405
6.19	GiNaC::composition_generator Class Reference	405
6.19.1	Detailed Description	406
6.19.2	Constructor & Destructor Documentation	406
6.19.2.1	composition_generator()	406
6.19.3	Member Function Documentation	406
6.19.3.1	get()	406
6.19.3.2	next()	406
6.19.4	Member Data Documentation	406
6.19.4.1	cmgen	407
6.19.4.2	atend	407
6.19.4.3	trivial	407
6.19.4.4	composition	407
6.19.4.5	current_updated	407
6.20	GiNaC::const_iterator Class Reference	408
6.20.1	Constructor & Destructor Documentation	409
6.20.1.1	const_iterator() [1/2]	409
6.20.1.2	const_iterator() [2/2]	409
6.20.2	Member Function Documentation	409
6.20.2.1	operator*()	409
6.20.2.2	operator->()	409
6.20.2.3	operator[]()	409
6.20.2.4	operator++() [1/2]	410
6.20.2.5	operator++() [2/2]	410
6.20.2.6	operator+=()	410
6.20.2.7	operator+()	410
6.20.2.8	operator--() [1/2]	410
6.20.2.9	operator--() [2/2]	410

6.20.2.10 operator==( )	411
6.20.2.11 operator-( )	411
6.20.2.12 operator==( )	411
6.20.2.13 operator!=( )	411
6.20.2.14 operator<( )	411
6.20.2.15 operator>( )	411
6.20.2.16 operator<=( )	412
6.20.2.17 operator>=( )	412
6.20.3 Friends And Related Function Documentation	412
6.20.3.1 ex	412
6.20.3.2 const_preorder_iterator	412
6.20.3.3 const_postorder_iterator	412
6.20.3.4 operator+	412
6.20.3.5 operator-	413
6.20.4 Member Data Documentation	413
6.20.4.1 e	413
6.20.4.2 i	413
6.21 GiNaC::const_postorder_iterator Class Reference	413
6.21.1 Constructor & Destructor Documentation	414
6.21.1.1 const_postorder_iterator() [1/2]	414
6.21.1.2 const_postorder_iterator() [2/2]	414
6.21.2 Member Function Documentation	414
6.21.2.1 operator*( )	414
6.21.2.2 operator->( )	414
6.21.2.3 operator++( ) [1/2]	415
6.21.2.4 operator++( ) [2/2]	415
6.21.2.5 operator==( )	415
6.21.2.6 operator!=( )	415
6.21.2.7 descend( )	415
6.21.2.8 increment( )	415



6.21.3	Member Data Documentation . . . . .	416
6.21.3.1	s . . . . .	416
6.22	GiNaC::const_preorder_iterator Class Reference . . . . .	416
6.22.1	Constructor & Destructor Documentation . . . . .	417
6.22.1.1	const_preorder_iterator() [1/2] . . . . .	417
6.22.1.2	const_preorder_iterator() [2/2] . . . . .	417
6.22.2	Member Function Documentation . . . . .	417
6.22.2.1	operator*() . . . . .	417
6.22.2.2	operator->() . . . . .	417
6.22.2.3	operator++() [1/2] . . . . .	417
6.22.2.4	operator++() [2/2] . . . . .	418
6.22.2.5	operator==() . . . . .	418
6.22.2.6	operator"!="() . . . . .	418
6.22.2.7	increment() . . . . .	418
6.22.3	Member Data Documentation . . . . .	418
6.22.3.1	s . . . . .	418
6.23	GiNaC::constant Class Reference . . . . .	419
6.23.1	Detailed Description . . . . .	420
6.23.2	Constructor & Destructor Documentation . . . . .	420
6.23.2.1	constant() [1/2] . . . . .	420
6.23.2.2	constant() [2/2] . . . . .	421
6.23.3	Member Function Documentation . . . . .	421
6.23.3.1	info() . . . . .	421
6.23.3.2	evalf() . . . . .	421
6.23.3.3	is_polynomial() . . . . .	422
6.23.3.4	conjugate() . . . . .	422
6.23.3.5	real_part() . . . . .	422
6.23.3.6	imag_part() . . . . .	422
6.23.3.7	archive() . . . . .	422
6.23.3.8	read_archive() . . . . .	423

6.23.3.9	<a href="#">derivative()</a>	423
6.23.3.10	<a href="#">is_equal_same_type()</a>	423
6.23.3.11	<a href="#">calchash()</a>	424
6.23.3.12	<a href="#">do_print()</a>	424
6.23.3.13	<a href="#">do_print_tree()</a>	424
6.23.3.14	<a href="#">do_print_latex()</a>	424
6.23.3.15	<a href="#">do_print_python_repr()</a>	424
6.23.4	<a href="#">Member Data Documentation</a>	425
6.23.4.1	<a href="#">name</a>	425
6.23.4.2	<a href="#">TeX_name</a>	425
6.23.4.3	<a href="#">ef</a>	425
6.23.4.4	<a href="#">number</a>	425
6.23.4.5	<a href="#">serial</a>	425
6.23.4.6	<a href="#">next_serial</a>	426
6.23.4.7	<a href="#">domain</a>	426
6.24	<a href="#">GiNaC::container&lt; C &gt; Class Template Reference</a>	426
6.24.1	<a href="#">Detailed Description</a>	428
6.24.2	<a href="#">Member Typedef Documentation</a>	429
6.24.2.1	<a href="#">STLT</a>	429
6.24.2.2	<a href="#">const_iterator</a>	429
6.24.2.3	<a href="#">const_reverse_iterator</a>	429
6.24.3	<a href="#">Constructor &amp; Destructor Documentation</a>	429
6.24.3.1	<a href="#">container() [1/4]</a>	429
6.24.3.2	<a href="#">container() [2/4]</a>	429
6.24.3.3	<a href="#">container() [3/4]</a>	430
6.24.3.4	<a href="#">container() [4/4]</a>	430
6.24.4	<a href="#">Member Function Documentation</a>	430
6.24.4.1	<a href="#">get_default_flags()</a>	430
6.24.4.2	<a href="#">get_open_delim()</a>	430
6.24.4.3	<a href="#">get_close_delim()</a>	430

6.24.4.4	<a href="#">info()</a>	431
6.24.4.5	<a href="#">precedence()</a>	431
6.24.4.6	<a href="#">nops()</a>	431
6.24.4.7	<a href="#">op()</a>	432
6.24.4.8	<a href="#">let_op()</a>	432
6.24.4.9	<a href="#">subs()</a>	432
6.24.4.10	<a href="#">read_archive()</a>	433
6.24.4.11	<a href="#">archive()</a>	433
6.24.4.12	<a href="#">conjugate()</a>	433
6.24.4.13	<a href="#">real_part()</a>	434
6.24.4.14	<a href="#">imag_part()</a>	434
6.24.4.15	<a href="#">is_equal_same_type()</a>	434
6.24.4.16	<a href="#">thiscontainer()</a> [1/2]	435
6.24.4.17	<a href="#">thiscontainer()</a> [2/2]	435
6.24.4.18	<a href="#">printseq()</a>	435
6.24.4.19	<a href="#">sort_()</a> [1/2]	436
6.24.4.20	<a href="#">sort_()</a> [2/2]	436
6.24.4.21	<a href="#">unique_()</a> [1/2]	436
6.24.4.22	<a href="#">prepend()</a>	436
6.24.4.23	<a href="#">append()</a>	436
6.24.4.24	<a href="#">remove_first()</a>	437
6.24.4.25	<a href="#">remove_last()</a>	437
6.24.4.26	<a href="#">remove_all()</a>	437
6.24.4.27	<a href="#">sort()</a>	437
6.24.4.28	<a href="#">unique()</a>	437
6.24.4.29	<a href="#">begin()</a>	438
6.24.4.30	<a href="#">end()</a>	438
6.24.4.31	<a href="#">rbegin()</a>	438
6.24.4.32	<a href="#">rend()</a>	438
6.24.4.33	<a href="#">do_print()</a>	439

6.24.4.34	<code>do_print_tree()</code>	439
6.24.4.35	<code>do_print_python()</code>	439
6.24.4.36	<code>do_print_python_repr()</code>	439
6.24.4.37	<code>subchildren()</code>	439
6.24.4.38	<code>unique_()</code> [2/2]	440
6.25	<code>GiNaC::container_storage&lt; C &gt;</code> Class Template Reference	440
6.25.1	Detailed Description	441
6.25.2	Member Typedef Documentation	441
6.25.2.1	STLT	441
6.25.3	Constructor & Destructor Documentation	441
6.25.3.1	<code>container_storage()</code> [1/4]	441
6.25.3.2	<code>container_storage()</code> [2/4]	441
6.25.3.3	<code>container_storage()</code> [3/4]	441
6.25.3.4	<code>container_storage()</code> [4/4]	442
6.25.3.5	<code>~container_storage()</code>	442
6.25.4	Member Function Documentation	442
6.25.4.1	<code>reserve()</code> [1/4]	442
6.25.4.2	<code>reserve()</code> [2/4]	442
6.25.4.3	<code>reserve()</code> [3/4]	442
6.25.4.4	<code>reserve()</code> [4/4]	443
6.25.5	Member Data Documentation	443
6.25.5.1	<code>seq</code>	443
6.26	<code>GiNaC::composition_generator::coolmulti</code> Struct Reference	443
6.26.1	Constructor & Destructor Documentation	444
6.26.1.1	<code>coolmulti()</code>	444
6.26.1.2	<code>~coolmulti()</code>	444
6.26.2	Member Function Documentation	444
6.26.2.1	<code>next_permutation()</code>	444
6.26.2.2	<code>finished()</code>	444
6.26.3	Member Data Documentation	445

6.26.3.1	head	445
6.26.3.2	i	445
6.26.3.3	after_i	445
6.27	GiNaC::derivative_map_function Struct Reference	445
6.27.1	Detailed Description	446
6.27.2	Constructor & Destructor Documentation	446
6.27.2.1	derivative_map_function()	446
6.27.3	Member Function Documentation	446
6.27.3.1	operator()()	446
6.27.4	Member Data Documentation	446
6.27.4.1	s	446
6.28	GiNaC::determinant_algo Class Reference	447
6.28.1	Detailed Description	447
6.28.2	Member Enumeration Documentation	447
6.28.2.1	anonymous enum	447
6.29	GiNaC::diracgamma Class Reference	448
6.29.1	Detailed Description	448
6.29.2	Member Function Documentation	448
6.29.2.1	contract_with()	449
6.29.2.2	do_print()	449
6.29.2.3	do_print_latex()	449
6.30	GiNaC::diracgamma5 Class Reference	449
6.30.1	Detailed Description	450
6.30.2	Member Function Documentation	450
6.30.2.1	conjugate()	450
6.30.2.2	do_print()	450
6.30.2.3	do_print_latex()	450
6.31	GiNaC::diracgammaL Class Reference	451
6.31.1	Detailed Description	451
6.31.2	Member Function Documentation	451

6.31.2.1	<a href="#">conjugate()</a>	451
6.31.2.2	<a href="#">do_print()</a>	452
6.31.2.3	<a href="#">do_print_latex()</a>	452
6.32	<a href="#">GiNaC::diracgammaR Class Reference</a>	452
6.32.1	<a href="#">Detailed Description</a>	453
6.32.2	<a href="#">Member Function Documentation</a>	453
6.32.2.1	<a href="#">conjugate()</a>	453
6.32.2.2	<a href="#">do_print()</a>	453
6.32.2.3	<a href="#">do_print_latex()</a>	453
6.33	<a href="#">GiNaC::diracone Class Reference</a>	453
6.33.1	<a href="#">Detailed Description</a>	454
6.33.2	<a href="#">Member Function Documentation</a>	454
6.33.2.1	<a href="#">do_print()</a>	454
6.33.2.2	<a href="#">do_print_latex()</a>	454
6.34	<a href="#">GiNaC::do_taylor Class Reference</a>	454
6.34.1	<a href="#">Detailed Description</a>	454
6.35	<a href="#">GiNaC::domain Class Reference</a>	455
6.35.1	<a href="#">Detailed Description</a>	455
6.35.2	<a href="#">Member Enumeration Documentation</a>	455
6.35.2.1	<a href="#">anonymous enum</a>	455
6.36	<a href="#">GiNaC::dunno Class Reference</a>	455
6.36.1	<a href="#">Detailed Description</a>	455
6.37	<a href="#">GiNaC::Ebar_kernel Class Reference</a>	456
6.37.1	<a href="#">Detailed Description</a>	457
6.37.2	<a href="#">Constructor &amp; Destructor Documentation</a>	457
6.37.2.1	<a href="#">Ebar_kernel()</a>	457
6.37.3	<a href="#">Member Function Documentation</a>	457
6.37.3.1	<a href="#">nops()</a>	457
6.37.3.2	<a href="#">op()</a>	457
6.37.3.3	<a href="#">let_op()</a>	458

6.37.3.4	<a href="#">is_numeric()</a>	458
6.37.3.5	<a href="#">get_numerical_value()</a>	458
6.37.3.6	<a href="#">series_coeff_impl()</a>	458
6.37.3.7	<a href="#">do_print()</a>	459
6.37.4	<a href="#">Member Data Documentation</a>	459
6.37.4.1	<a href="#">n</a>	459
6.37.4.2	<a href="#">m</a>	459
6.37.4.3	<a href="#">x</a>	459
6.37.4.4	<a href="#">y</a>	459
6.38	<a href="#">GiNaC::Eisenstein_h_kernel Class Reference</a>	460
6.38.1	<a href="#">Detailed Description</a>	461
6.38.2	<a href="#">Constructor &amp; Destructor Documentation</a>	461
6.38.2.1	<a href="#">Eisenstein_h_kernel()</a>	461
6.38.3	<a href="#">Member Function Documentation</a>	461
6.38.3.1	<a href="#">series()</a>	461
6.38.3.2	<a href="#">nops()</a>	462
6.38.3.3	<a href="#">op()</a>	462
6.38.3.4	<a href="#">let_op()</a>	462
6.38.3.5	<a href="#">is_numeric()</a>	462
6.38.3.6	<a href="#">Laurent_series()</a>	463
6.38.3.7	<a href="#">get_numerical_value()</a>	463
6.38.3.8	<a href="#">uses_Laurent_series()</a>	463
6.38.3.9	<a href="#">coefficient_a0()</a>	463
6.38.3.10	<a href="#">coefficient_an()</a>	464
6.38.3.11	<a href="#">q_expansion_modular_form()</a>	464
6.38.3.12	<a href="#">do_print()</a>	464
6.38.4	<a href="#">Member Data Documentation</a>	464
6.38.4.1	<a href="#">k</a>	464
6.38.4.2	<a href="#">N</a>	465
6.38.4.3	<a href="#">r</a>	465

6.38.4.4	s	465
6.38.4.5	C_norm	465
6.39	GiNaC::Eisenstein_kernel Class Reference	465
6.39.1	Detailed Description	466
6.39.2	Constructor & Destructor Documentation	467
6.39.2.1	Eisenstein_kernel()	467
6.39.3	Member Function Documentation	467
6.39.3.1	series()	467
6.39.3.2	nops()	467
6.39.3.3	op()	468
6.39.3.4	let_op()	468
6.39.3.5	is_numeric()	468
6.39.3.6	Laurent_series()	468
6.39.3.7	get_numerical_value()	469
6.39.3.8	uses_Laurent_series()	469
6.39.3.9	q_expansion_modular_form()	469
6.39.3.10	do_print()	469
6.39.4	Member Data Documentation	469
6.39.4.1	k	470
6.39.4.2	N	470
6.39.4.3	a	470
6.39.4.4	b	470
6.39.4.5	K	470
6.39.4.6	C_norm	470
6.40	GiNaC::composition_generator::coolmulti::element Struct Reference	471
6.40.1	Constructor & Destructor Documentation	471
6.40.1.1	element()	471
6.40.1.2	~element()	471
6.40.2	Member Data Documentation	471
6.40.2.1	value	471



6.40.2.2	next	472
6.41	GiNaC::ELi_kernel Class Reference	472
6.41.1	Detailed Description	473
6.41.2	Constructor & Destructor Documentation	473
6.41.2.1	ELi_kernel()	473
6.41.3	Member Function Documentation	473
6.41.3.1	nops()	473
6.41.3.2	op()	473
6.41.3.3	let_op()	474
6.41.3.4	is_numeric()	474
6.41.3.5	get_numerical_value()	474
6.41.3.6	series_coeff_impl()	474
6.41.3.7	do_print()	475
6.41.4	Member Data Documentation	475
6.41.4.1	n	475
6.41.4.2	m	475
6.41.4.3	x	475
6.41.4.4	y	475
6.42	std::equal_to< GiNaC::ex > Struct Template Reference	476
6.42.1	Detailed Description	476
6.42.2	Member Function Documentation	476
6.42.2.1	operator()	476
6.43	GiNaC::error_and_integral Struct Reference	476
6.43.1	Constructor & Destructor Documentation	477
6.43.1.1	error_and_integral()	477
6.43.2	Member Data Documentation	477
6.43.2.1	error	477
6.43.2.2	integral	477
6.44	GiNaC::error_and_integral_is_less Struct Reference	477
6.44.1	Member Function Documentation	477

6.44.1.1	<a href="#">operator()</a>	478
6.45	<a href="#">GiNaC::eval_integ_map_function Struct Reference</a>	478
6.45.1	<a href="#">Detailed Description</a>	478
6.45.2	<a href="#">Member Function Documentation</a>	478
6.45.2.1	<a href="#">operator()</a>	478
6.46	<a href="#">GiNaC::evalf_map_function Struct Reference</a>	479
6.46.1	<a href="#">Detailed Description</a>	479
6.46.2	<a href="#">Member Function Documentation</a>	479
6.46.2.1	<a href="#">operator()</a>	479
6.47	<a href="#">GiNaC::evalm_map_function Struct Reference</a>	479
6.47.1	<a href="#">Detailed Description</a>	480
6.47.2	<a href="#">Member Function Documentation</a>	480
6.47.2.1	<a href="#">operator()</a>	480
6.48	<a href="#">GiNaC::ex Class Reference</a>	480
6.48.1	<a href="#">Detailed Description</a>	484
6.48.2	<a href="#">Constructor &amp; Destructor Documentation</a>	484
6.48.2.1	<a href="#">ex() [1/10]</a>	484
6.48.2.2	<a href="#">ex() [2/10]</a>	484
6.48.2.3	<a href="#">ex() [3/10]</a>	485
6.48.2.4	<a href="#">ex() [4/10]</a>	485
6.48.2.5	<a href="#">ex() [5/10]</a>	485
6.48.2.6	<a href="#">ex() [6/10]</a>	485
6.48.2.7	<a href="#">ex() [7/10]</a>	485
6.48.2.8	<a href="#">ex() [8/10]</a>	485
6.48.2.9	<a href="#">ex() [9/10]</a>	486
6.48.2.10	<a href="#">ex() [10/10]</a>	486
6.48.3	<a href="#">Member Function Documentation</a>	486
6.48.3.1	<a href="#">swap()</a>	486
6.48.3.2	<a href="#">begin()</a>	486
6.48.3.3	<a href="#">end()</a>	487

6.48.3.4	<code>preorder_begin()</code>	487
6.48.3.5	<code>preorder_end()</code>	487
6.48.3.6	<code>postorder_begin()</code>	487
6.48.3.7	<code>postorder_end()</code>	487
6.48.3.8	<code>eval()</code>	487
6.48.3.9	<code>evalf()</code>	488
6.48.3.10	<code>evalm()</code>	488
6.48.3.11	<code>eval_ncmul()</code>	488
6.48.3.12	<code>eval_integ()</code>	488
6.48.3.13	<code>print()</code>	489
6.48.3.14	<code>dbgprint()</code>	489
6.48.3.15	<code>dbgprinttree()</code>	489
6.48.3.16	<code>info()</code>	490
6.48.3.17	<code>nops()</code>	490
6.48.3.18	<code>op()</code>	491
6.48.3.19	<code>operator[]()</code> [1/4]	491
6.48.3.20	<code>operator[]()</code> [2/4]	491
6.48.3.21	<code>let_op()</code>	491
6.48.3.22	<code>operator[]()</code> [3/4]	492
6.48.3.23	<code>operator[]()</code> [4/4]	492
6.48.3.24	<code>lhs()</code>	492
6.48.3.25	<code>rhs()</code>	492
6.48.3.26	<code>conjugate()</code>	492
6.48.3.27	<code>real_part()</code>	493
6.48.3.28	<code>imag_part()</code>	493
6.48.3.29	<code>has()</code>	493
6.48.3.30	<code>find()</code>	493
6.48.3.31	<code>match()</code> [1/2]	494
6.48.3.32	<code>match()</code> [2/2]	494
6.48.3.33	<code>subs()</code> [1/3]	494

6.48.3.34 subs() [2/3]	495
6.48.3.35 subs() [3/3]	495
6.48.3.36 map() [1/2]	495
6.48.3.37 map() [2/2]	495
6.48.3.38 accept()	496
6.48.3.39 traverse_preorder()	496
6.48.3.40 traverse_postorder()	496
6.48.3.41 traverse()	496
6.48.3.42 is_polynomial()	497
6.48.3.43 degree()	497
6.48.3.44 ldegree()	497
6.48.3.45 coeff()	497
6.48.3.46 lcoeff()	498
6.48.3.47 tcoeff()	498
6.48.3.48 expand()	498
6.48.3.49 collect()	498
6.48.3.50 diff()	498
6.48.3.51 series()	499
6.48.3.52 normal()	500
6.48.3.53 to_rational()	500
6.48.3.54 to_polynomial()	501
6.48.3.55 numer()	501
6.48.3.56 denom()	501
6.48.3.57 numer_denom()	502
6.48.3.58 unit()	502
6.48.3.59 content()	503
6.48.3.60 integer_content()	503
6.48.3.61 primpart() [1/2]	503
6.48.3.62 primpart() [2/2]	504
6.48.3.63 unitcontprim()	504

6.48.3.64 smod()	505
6.48.3.65 max_coefficient()	505
6.48.3.66 get_free_indices()	506
6.48.3.67 simplify_indexed() [1/2]	506
6.48.3.68 simplify_indexed() [2/2]	506
6.48.3.69 compare()	507
6.48.3.70 is_equal()	507
6.48.3.71 is_zero()	508
6.48.3.72 is_zero_matrix()	508
6.48.3.73 symmetrize() [1/2]	508
6.48.3.74 symmetrize() [2/2]	508
6.48.3.75 antisymmetrize() [1/2]	509
6.48.3.76 antisymmetrize() [2/2]	509
6.48.3.77 symmetrize_cyclic() [1/2]	509
6.48.3.78 symmetrize_cyclic() [2/2]	509
6.48.3.79 return_type()	510
6.48.3.80 return_type_tinfo()	510
6.48.3.81 gethash()	510
6.48.3.82 construct_from_basic()	510
6.48.3.83 construct_from_int()	511
6.48.3.84 construct_from_uint()	511
6.48.3.85 construct_from_long()	511
6.48.3.86 construct_from_ulong()	511
6.48.3.87 construct_from_longlong()	512
6.48.3.88 construct_from_ulonglong()	512
6.48.3.89 construct_from_double()	512
6.48.3.90 construct_from_string_and_lst()	512
6.48.3.91 makewritable()	512
6.48.3.92 share()	513
6.48.4 Friends And Related Function Documentation	513

6.48.4.1	<a href="#">archive_node</a>	513
6.48.4.2	<a href="#">are_ex_trivially_equal</a>	513
6.48.4.3	<a href="#">ex_to</a>	513
6.48.4.4	<a href="#">is_a</a>	514
6.48.4.5	<a href="#">is_exactly_a</a>	514
6.48.5	<a href="#">Member Data Documentation</a>	514
6.48.5.1	<a href="#">bp</a>	514
6.49	<a href="#">GiNaC::ex_base_is_less Struct Reference</a>	515
6.49.1	<a href="#">Member Function Documentation</a>	515
6.49.1.1	<a href="#">operator()()</a>	515
6.50	<a href="#">GiNaC::ex_is_equal Struct Reference</a>	515
6.50.1	<a href="#">Member Function Documentation</a>	515
6.50.1.1	<a href="#">operator()()</a>	515
6.51	<a href="#">GiNaC::ex_is_less Struct Reference</a>	516
6.51.1	<a href="#">Member Function Documentation</a>	516
6.51.1.1	<a href="#">operator()()</a>	516
6.52	<a href="#">GiNaC::ex_swap Struct Reference</a>	516
6.52.1	<a href="#">Member Function Documentation</a>	516
6.52.1.1	<a href="#">operator()()</a>	516
6.53	<a href="#">GiNaC::expair Class Reference</a>	517
6.53.1	<a href="#">Detailed Description</a>	517
6.53.2	<a href="#">Constructor &amp; Destructor Documentation</a>	517
6.53.2.1	<a href="#">expair() [1/2]</a>	517
6.53.2.2	<a href="#">expair() [2/2]</a>	518
6.53.3	<a href="#">Member Function Documentation</a>	518
6.53.3.1	<a href="#">is_equal()</a>	518
6.53.3.2	<a href="#">is_less()</a>	518
6.53.3.3	<a href="#">compare()</a>	518
6.53.3.4	<a href="#">print()</a>	519
6.53.3.5	<a href="#">is_canonical_numeric()</a>	519

6.53.3.6	<a href="#">swap()</a>	519
6.53.3.7	<a href="#">conjugate()</a>	519
6.53.4	<a href="#">Member Data Documentation</a>	519
6.53.4.1	<a href="#">rest</a>	519
6.53.4.2	<a href="#">coeff</a>	520
6.54	<a href="#">GiNaC::expair_is_less Struct Reference</a>	520
6.54.1	<a href="#">Detailed Description</a>	520
6.54.2	<a href="#">Member Function Documentation</a>	520
6.54.2.1	<a href="#">operator()()</a>	520
6.55	<a href="#">GiNaC::expair_rest_is_less Struct Reference</a>	521
6.55.1	<a href="#">Detailed Description</a>	521
6.55.2	<a href="#">Member Function Documentation</a>	521
6.55.2.1	<a href="#">operator()()</a>	521
6.56	<a href="#">GiNaC::expair_swap Struct Reference</a>	521
6.56.1	<a href="#">Member Function Documentation</a>	521
6.56.1.1	<a href="#">operator()()</a>	522
6.57	<a href="#">GiNaC::expairseq Class Reference</a>	522
6.57.1	<a href="#">Detailed Description</a>	524
6.57.2	<a href="#">Constructor &amp; Destructor Documentation</a>	524
6.57.2.1	<a href="#">expairseq() [1/4]</a>	524
6.57.2.2	<a href="#">expairseq() [2/4]</a>	524
6.57.2.3	<a href="#">expairseq() [3/4]</a>	524
6.57.2.4	<a href="#">expairseq() [4/4]</a>	524
6.57.3	<a href="#">Member Function Documentation</a>	525
6.57.3.1	<a href="#">precedence()</a>	525
6.57.3.2	<a href="#">info()</a>	525
6.57.3.3	<a href="#">nops()</a>	525
6.57.3.4	<a href="#">op()</a>	526
6.57.3.5	<a href="#">map()</a>	526
6.57.3.6	<a href="#">eval()</a>	526

6.57.3.7	<a href="#">to_rational()</a>	526
6.57.3.8	<a href="#">to_polynomial()</a>	527
6.57.3.9	<a href="#">match()</a>	527
6.57.3.10	<a href="#">subs()</a>	527
6.57.3.11	<a href="#">conjugate()</a>	527
6.57.3.12	<a href="#">archive()</a>	528
6.57.3.13	<a href="#">read_archive()</a>	528
6.57.3.14	<a href="#">is_equal_same_type()</a>	528
6.57.3.15	<a href="#">return_type()</a>	529
6.57.3.16	<a href="#">calchash()</a>	529
6.57.3.17	<a href="#">expand()</a>	529
6.57.3.18	<a href="#">thisexpairseq()</a> [1/2]	529
6.57.3.19	<a href="#">thisexpairseq()</a> [2/2]	530
6.57.3.20	<a href="#">printseq()</a>	530
6.57.3.21	<a href="#">printpair()</a>	530
6.57.3.22	<a href="#">split_ex_to_pair()</a>	530
6.57.3.23	<a href="#">combine_ex_with_coeff_to_pair()</a>	530
6.57.3.24	<a href="#">combine_pair_with_coeff_to_pair()</a>	531
6.57.3.25	<a href="#">recombine_pair_to_ex()</a>	531
6.57.3.26	<a href="#">expair_needs_further_processing()</a>	531
6.57.3.27	<a href="#">default_overall_coeff()</a>	531
6.57.3.28	<a href="#">combine_overall_coeff()</a> [1/2]	531
6.57.3.29	<a href="#">combine_overall_coeff()</a> [2/2]	532
6.57.3.30	<a href="#">can_make_flat()</a>	532
6.57.3.31	<a href="#">do_print()</a>	532
6.57.3.32	<a href="#">do_print_tree()</a>	532
6.57.3.33	<a href="#">construct_from_2_ex()</a>	532
6.57.3.34	<a href="#">construct_from_2_expairseq()</a>	532
6.57.3.35	<a href="#">construct_from_expairseq_ex()</a>	533
6.57.3.36	<a href="#">construct_from_exvector()</a>	533



6.57.3.37 <code>construct_from_epvector()</code> [1/2]	533
6.57.3.38 <code>construct_from_epvector()</code> [2/2]	533
6.57.3.39 <code>make_flat()</code> [1/2]	533
6.57.3.40 <code>make_flat()</code> [2/2]	533
6.57.3.41 <code>canonicalize()</code>	534
6.57.3.42 <code>combine_same_terms_sorted_seq()</code>	534
6.57.3.43 <code>is_canonical()</code>	534
6.57.3.44 <code>expandchildren()</code>	534
6.57.3.45 <code>evalchildren()</code>	534
6.57.3.46 <code>subschildren()</code>	534
6.57.4 Member Data Documentation	534
6.57.4.1 <code>seq</code>	535
6.57.4.2 <code>overall_coeff</code>	535
6.58 <code>GiNaC::expand_map_function</code> Struct Reference	535
6.58.1 Detailed Description	536
6.58.2 Constructor & Destructor Documentation	536
6.58.2.1 <code>expand_map_function()</code>	536
6.58.3 Member Function Documentation	536
6.58.3.1 <code>operator()()</code>	536
6.58.4 Member Data Documentation	536
6.58.4.1 <code>options</code>	536
6.59 <code>GiNaC::expand_options</code> Class Reference	537
6.59.1 Detailed Description	537
6.59.2 Member Enumeration Documentation	537
6.59.2.1 <code>anonymous enum</code>	537
6.60 <code>GiNaC::factor_options</code> Class Reference	537
6.60.1 Detailed Description	538
6.60.2 Member Enumeration Documentation	538
6.60.2.1 <code>anonymous enum</code>	538
6.61 <code>GiNaC::fail</code> Class Reference	538

6.61.1	Member Function Documentation	538
6.61.1.1	return_type()	539
6.61.1.2	do_print()	539
6.62	GiNaC::fderivative Class Reference	539
6.62.1	Detailed Description	540
6.62.2	Constructor & Destructor Documentation	541
6.62.2.1	fderivative() [1/3]	541
6.62.2.2	fderivative() [2/3]	541
6.62.2.3	fderivative() [3/3]	541
6.62.3	Member Function Documentation	542
6.62.3.1	print()	542
6.62.3.2	eval()	542
6.62.3.3	series()	542
6.62.3.4	thiscontainer() [1/2]	543
6.62.3.5	thiscontainer() [2/2]	543
6.62.3.6	archive()	543
6.62.3.7	read_archive()	543
6.62.3.8	derivative()	544
6.62.3.9	is_equal_same_type()	544
6.62.3.10	match_same_type()	544
6.62.3.11	derivatives()	545
6.62.3.12	do_print()	545
6.62.3.13	do_print_latex()	545
6.62.3.14	do_print_csrc()	545
6.62.3.15	do_print_tree()	546
6.62.4	Member Data Documentation	546
6.62.4.1	parameter_set	546
6.63	GiNaC::function Class Reference	546
6.63.1	Detailed Description	549
6.63.2	Constructor & Destructor Documentation	549

6.63.2.1	<a href="#">function()</a> [1/18]	549
6.63.2.2	<a href="#">function()</a> [2/18]	549
6.63.2.3	<a href="#">function()</a> [3/18]	549
6.63.2.4	<a href="#">function()</a> [4/18]	549
6.63.2.5	<a href="#">function()</a> [5/18]	550
6.63.2.6	<a href="#">function()</a> [6/18]	550
6.63.2.7	<a href="#">function()</a> [7/18]	550
6.63.2.8	<a href="#">function()</a> [8/18]	550
6.63.2.9	<a href="#">function()</a> [9/18]	551
6.63.2.10	<a href="#">function()</a> [10/18]	551
6.63.2.11	<a href="#">function()</a> [11/18]	551
6.63.2.12	<a href="#">function()</a> [12/18]	552
6.63.2.13	<a href="#">function()</a> [13/18]	552
6.63.2.14	<a href="#">function()</a> [14/18]	552
6.63.2.15	<a href="#">function()</a> [15/18]	553
6.63.2.16	<a href="#">function()</a> [16/18]	553
6.63.2.17	<a href="#">function()</a> [17/18]	553
6.63.2.18	<a href="#">function()</a> [18/18]	553
6.63.3	<a href="#">Member Function Documentation</a>	553
6.63.3.1	<a href="#">print()</a>	553
6.63.3.2	<a href="#">precedence()</a>	554
6.63.3.3	<a href="#">expand()</a>	554
6.63.3.4	<a href="#">eval()</a>	555
6.63.3.5	<a href="#">evalf()</a>	555
6.63.3.6	<a href="#">eval_ncmul()</a>	555
6.63.3.7	<a href="#">calchash()</a>	555
6.63.3.8	<a href="#">series()</a>	556
6.63.3.9	<a href="#">thiscontainer()</a> [1/2]	556
6.63.3.10	<a href="#">thiscontainer()</a> [2/2]	556
6.63.3.11	<a href="#">conjugate()</a>	556

6.63.3.12	<a href="#">real_part()</a>	557
6.63.3.13	<a href="#">imag_part()</a>	557
6.63.3.14	<a href="#">archive()</a>	557
6.63.3.15	<a href="#">read_archive()</a>	557
6.63.3.16	<a href="#">info()</a>	558
6.63.3.17	<a href="#">derivative()</a>	558
6.63.3.18	<a href="#">is_equal_same_type()</a>	558
6.63.3.19	<a href="#">match_same_type()</a>	559
6.63.3.20	<a href="#">return_type()</a>	559
6.63.3.21	<a href="#">return_type_tinfo()</a>	559
6.63.3.22	<a href="#">pderivative()</a>	559
6.63.3.23	<a href="#">expl_derivative()</a>	560
6.63.3.24	<a href="#">registered_functions()</a>	560
6.63.3.25	<a href="#">lookup_remember_table()</a>	560
6.63.3.26	<a href="#">store_remember_table()</a>	560
6.63.3.27	<a href="#">power()</a>	560
6.63.3.28	<a href="#">register_new()</a>	561
6.63.3.29	<a href="#">find_function()</a>	561
6.63.3.30	<a href="#">get_registered_functions()</a>	561
6.63.3.31	<a href="#">get_serial()</a>	561
6.63.3.32	<a href="#">get_name()</a>	561
6.63.4	<a href="#">Friends And Related Function Documentation</a>	562
6.63.4.1	<a href="#">remember_table_entry</a>	562
6.63.5	<a href="#">Member Data Documentation</a>	562
6.63.5.1	<a href="#">current_serial</a>	562
6.63.5.2	<a href="#">serial</a>	562
6.64	<a href="#">GiNaC::function_options Class Reference</a>	562
6.64.1	<a href="#">Constructor &amp; Destructor Documentation</a>	567
6.64.1.1	<a href="#">function_options() [1/3]</a>	568
6.64.1.2	<a href="#">function_options() [2/3]</a>	568

6.64.1.3	<a href="#">function_options()</a> [3/3]	568
6.64.1.4	<a href="#">~function_options()</a>	568
6.64.2	<a href="#">Member Function Documentation</a>	568
6.64.2.1	<a href="#">initialize()</a>	568
6.64.2.2	<a href="#">dummy()</a>	569
6.64.2.3	<a href="#">set_name()</a>	569
6.64.2.4	<a href="#">latex_name()</a>	569
6.64.2.5	<a href="#">eval_func()</a> [1/15]	569
6.64.2.6	<a href="#">eval_func()</a> [2/15]	569
6.64.2.7	<a href="#">eval_func()</a> [3/15]	569
6.64.2.8	<a href="#">eval_func()</a> [4/15]	570
6.64.2.9	<a href="#">eval_func()</a> [5/15]	570
6.64.2.10	<a href="#">eval_func()</a> [6/15]	570
6.64.2.11	<a href="#">eval_func()</a> [7/15]	570
6.64.2.12	<a href="#">eval_func()</a> [8/15]	570
6.64.2.13	<a href="#">eval_func()</a> [9/15]	570
6.64.2.14	<a href="#">eval_func()</a> [10/15]	571
6.64.2.15	<a href="#">eval_func()</a> [11/15]	571
6.64.2.16	<a href="#">eval_func()</a> [12/15]	571
6.64.2.17	<a href="#">eval_func()</a> [13/15]	571
6.64.2.18	<a href="#">eval_func()</a> [14/15]	571
6.64.2.19	<a href="#">evalf_func()</a> [1/15]	571
6.64.2.20	<a href="#">evalf_func()</a> [2/15]	572
6.64.2.21	<a href="#">evalf_func()</a> [3/15]	572
6.64.2.22	<a href="#">evalf_func()</a> [4/15]	572
6.64.2.23	<a href="#">evalf_func()</a> [5/15]	572
6.64.2.24	<a href="#">evalf_func()</a> [6/15]	572
6.64.2.25	<a href="#">evalf_func()</a> [7/15]	572
6.64.2.26	<a href="#">evalf_func()</a> [8/15]	573
6.64.2.27	<a href="#">evalf_func()</a> [9/15]	573

6.64.2.28 evalf_func() [10/15] . . . . .	573
6.64.2.29 evalf_func() [11/15] . . . . .	573
6.64.2.30 evalf_func() [12/15] . . . . .	573
6.64.2.31 evalf_func() [13/15] . . . . .	573
6.64.2.32 evalf_func() [14/15] . . . . .	574
6.64.2.33 conjugate_func() [1/15] . . . . .	574
6.64.2.34 conjugate_func() [2/15] . . . . .	574
6.64.2.35 conjugate_func() [3/15] . . . . .	574
6.64.2.36 conjugate_func() [4/15] . . . . .	574
6.64.2.37 conjugate_func() [5/15] . . . . .	574
6.64.2.38 conjugate_func() [6/15] . . . . .	575
6.64.2.39 conjugate_func() [7/15] . . . . .	575
6.64.2.40 conjugate_func() [8/15] . . . . .	575
6.64.2.41 conjugate_func() [9/15] . . . . .	575
6.64.2.42 conjugate_func() [10/15] . . . . .	575
6.64.2.43 conjugate_func() [11/15] . . . . .	575
6.64.2.44 conjugate_func() [12/15] . . . . .	576
6.64.2.45 conjugate_func() [13/15] . . . . .	576
6.64.2.46 conjugate_func() [14/15] . . . . .	576
6.64.2.47 real_part_func() [1/15] . . . . .	576
6.64.2.48 real_part_func() [2/15] . . . . .	576
6.64.2.49 real_part_func() [3/15] . . . . .	576
6.64.2.50 real_part_func() [4/15] . . . . .	577
6.64.2.51 real_part_func() [5/15] . . . . .	577
6.64.2.52 real_part_func() [6/15] . . . . .	577
6.64.2.53 real_part_func() [7/15] . . . . .	577
6.64.2.54 real_part_func() [8/15] . . . . .	577
6.64.2.55 real_part_func() [9/15] . . . . .	577
6.64.2.56 real_part_func() [10/15] . . . . .	578
6.64.2.57 real_part_func() [11/15] . . . . .	578

6.64.2.58 <code>real_part_func()</code> [12/15]	578
6.64.2.59 <code>real_part_func()</code> [13/15]	578
6.64.2.60 <code>real_part_func()</code> [14/15]	578
6.64.2.61 <code>imag_part_func()</code> [1/15]	578
6.64.2.62 <code>imag_part_func()</code> [2/15]	579
6.64.2.63 <code>imag_part_func()</code> [3/15]	579
6.64.2.64 <code>imag_part_func()</code> [4/15]	579
6.64.2.65 <code>imag_part_func()</code> [5/15]	579
6.64.2.66 <code>imag_part_func()</code> [6/15]	579
6.64.2.67 <code>imag_part_func()</code> [7/15]	579
6.64.2.68 <code>imag_part_func()</code> [8/15]	580
6.64.2.69 <code>imag_part_func()</code> [9/15]	580
6.64.2.70 <code>imag_part_func()</code> [10/15]	580
6.64.2.71 <code>imag_part_func()</code> [11/15]	580
6.64.2.72 <code>imag_part_func()</code> [12/15]	580
6.64.2.73 <code>imag_part_func()</code> [13/15]	580
6.64.2.74 <code>imag_part_func()</code> [14/15]	581
6.64.2.75 <code>expand_func()</code> [1/15]	581
6.64.2.76 <code>expand_func()</code> [2/15]	581
6.64.2.77 <code>expand_func()</code> [3/15]	581
6.64.2.78 <code>expand_func()</code> [4/15]	581
6.64.2.79 <code>expand_func()</code> [5/15]	581
6.64.2.80 <code>expand_func()</code> [6/15]	582
6.64.2.81 <code>expand_func()</code> [7/15]	582
6.64.2.82 <code>expand_func()</code> [8/15]	582
6.64.2.83 <code>expand_func()</code> [9/15]	582
6.64.2.84 <code>expand_func()</code> [10/15]	582
6.64.2.85 <code>expand_func()</code> [11/15]	582
6.64.2.86 <code>expand_func()</code> [12/15]	583
6.64.2.87 <code>expand_func()</code> [13/15]	583

6.64.2.88	<code>expand_func()</code>	[14/15]	583
6.64.2.89	<code>derivative_func()</code>	[1/15]	583
6.64.2.90	<code>derivative_func()</code>	[2/15]	583
6.64.2.91	<code>derivative_func()</code>	[3/15]	583
6.64.2.92	<code>derivative_func()</code>	[4/15]	584
6.64.2.93	<code>derivative_func()</code>	[5/15]	584
6.64.2.94	<code>derivative_func()</code>	[6/15]	584
6.64.2.95	<code>derivative_func()</code>	[7/15]	584
6.64.2.96	<code>derivative_func()</code>	[8/15]	584
6.64.2.97	<code>derivative_func()</code>	[9/15]	584
6.64.2.98	<code>derivative_func()</code>	[10/15]	585
6.64.2.99	<code>derivative_func()</code>	[11/15]	585
6.64.2.100	<code>derivative_func()</code>	[12/15]	585
6.64.2.101	<code>derivative_func()</code>	[13/15]	585
6.64.2.102	<code>derivative_func()</code>	[14/15]	585
6.64.2.103	<code>expl_derivative_func()</code>	[1/15]	585
6.64.2.104	<code>expl_derivative_func()</code>	[2/15]	586
6.64.2.105	<code>expl_derivative_func()</code>	[3/15]	586
6.64.2.106	<code>expl_derivative_func()</code>	[4/15]	586
6.64.2.107	<code>expl_derivative_func()</code>	[5/15]	586
6.64.2.108	<code>expl_derivative_func()</code>	[6/15]	586
6.64.2.109	<code>expl_derivative_func()</code>	[7/15]	586
6.64.2.110	<code>expl_derivative_func()</code>	[8/15]	587
6.64.2.111	<code>expl_derivative_func()</code>	[9/15]	587
6.64.2.112	<code>expl_derivative_func()</code>	[10/15]	587
6.64.2.113	<code>expl_derivative_func()</code>	[11/15]	587
6.64.2.114	<code>expl_derivative_func()</code>	[12/15]	587
6.64.2.115	<code>expl_derivative_func()</code>	[13/15]	587
6.64.2.116	<code>expl_derivative_func()</code>	[14/15]	588
6.64.2.117	<code>power_func()</code>	[1/15]	588



6.64.2.118	<a href="#">power_func()</a> [2/15]	588
6.64.2.119	<a href="#">power_func()</a> [3/15]	588
6.64.2.120	<a href="#">power_func()</a> [4/15]	588
6.64.2.121	<a href="#">power_func()</a> [5/15]	588
6.64.2.122	<a href="#">power_func()</a> [6/15]	589
6.64.2.123	<a href="#">power_func()</a> [7/15]	589
6.64.2.124	<a href="#">power_func()</a> [8/15]	589
6.64.2.125	<a href="#">power_func()</a> [9/15]	589
6.64.2.126	<a href="#">power_func()</a> [10/15]	589
6.64.2.127	<a href="#">power_func()</a> [11/15]	589
6.64.2.128	<a href="#">power_func()</a> [12/15]	590
6.64.2.129	<a href="#">power_func()</a> [13/15]	590
6.64.2.130	<a href="#">power_func()</a> [14/15]	590
6.64.2.131	<a href="#">series_func()</a> [1/15]	590
6.64.2.132	<a href="#">series_func()</a> [2/15]	590
6.64.2.133	<a href="#">series_func()</a> [3/15]	590
6.64.2.134	<a href="#">series_func()</a> [4/15]	591
6.64.2.135	<a href="#">series_func()</a> [5/15]	591
6.64.2.136	<a href="#">series_func()</a> [6/15]	591
6.64.2.137	<a href="#">series_func()</a> [7/15]	591
6.64.2.138	<a href="#">series_func()</a> [8/15]	591
6.64.2.139	<a href="#">series_func()</a> [9/15]	591
6.64.2.140	<a href="#">series_func()</a> [10/15]	592
6.64.2.141	<a href="#">series_func()</a> [11/15]	592
6.64.2.142	<a href="#">series_func()</a> [12/15]	592
6.64.2.143	<a href="#">series_func()</a> [13/15]	592
6.64.2.144	<a href="#">series_func()</a> [14/15]	592
6.64.2.145	<a href="#">info_func()</a> [1/15]	592
6.64.2.146	<a href="#">info_func()</a> [2/15]	593
6.64.2.147	<a href="#">info_func()</a> [3/15]	593

6.64.2.148	info_func()	[ 4/15]	593
6.64.2.149	info_func()	[ 5/15]	593
6.64.2.150	info_func()	[ 6/15]	593
6.64.2.151	info_func()	[ 7/15]	593
6.64.2.152	info_func()	[ 8/15]	594
6.64.2.153	info_func()	[ 9/15]	594
6.64.2.154	info_func()	[10/15]	594
6.64.2.155	info_func()	[11/15]	594
6.64.2.156	info_func()	[12/15]	594
6.64.2.157	info_func()	[13/15]	594
6.64.2.158	info_func()	[14/15]	595
6.64.2.159	eval_func()	[15/15]	595
6.64.2.160	evalf_func()	[15/15]	595
6.64.2.161	conjugate_func()	[15/15]	595
6.64.2.162	real_part_func()	[15/15]	595
6.64.2.163	imag_part_func()	[15/15]	595
6.64.2.164	expand_func()	[15/15]	596
6.64.2.165	derivative_func()	[15/15]	596
6.64.2.166	expl_derivative_func()	[15/15]	596
6.64.2.167	power_func()	[15/15]	596
6.64.2.168	series_func()	[15/15]	596
6.64.2.169	info_func()	[15/15]	596
6.64.2.170	print_func()	[ 1/15]	597
6.64.2.171	print_func()	[ 2/15]	597
6.64.2.172	print_func()	[ 3/15]	597
6.64.2.173	print_func()	[ 4/15]	597
6.64.2.174	print_func()	[ 5/15]	597
6.64.2.175	print_func()	[ 6/15]	598
6.64.2.176	print_func()	[ 7/15]	598
6.64.2.177	print_func()	[ 8/15]	598

6.64.2.178	<a href="#">print_func()</a> [9/15]	598
6.64.2.179	<a href="#">print_func()</a> [10/15]	598
6.64.2.180	<a href="#">print_func()</a> [11/15]	599
6.64.2.181	<a href="#">print_func()</a> [12/15]	599
6.64.2.182	<a href="#">print_func()</a> [13/15]	599
6.64.2.183	<a href="#">print_func()</a> [14/15]	599
6.64.2.184	<a href="#">print_func()</a> [15/15]	599
6.64.2.185	<a href="#">set_return_type()</a>	600
6.64.2.186	<a href="#">do_not_evalf_params()</a>	600
6.64.2.187	<a href="#">remember()</a>	600
6.64.2.188	<a href="#">overloaded()</a>	600
6.64.2.189	<a href="#">set_symmetry()</a>	600
6.64.2.190	<a href="#">get_name()</a>	601
6.64.2.191	<a href="#">get_nparams()</a>	601
6.64.2.192	<a href="#">has_derivative()</a>	601
6.64.2.193	<a href="#">has_power()</a>	601
6.64.2.194	<a href="#">test_and_set_nparams()</a>	601
6.64.2.195	<a href="#">set_print_func()</a>	601
6.64.3	<a href="#">Friends And Related Function Documentation</a>	602
6.64.3.1	<a href="#">function</a>	602
6.64.3.2	<a href="#">fderivative</a>	602
6.64.4	<a href="#">Member Data Documentation</a>	602
6.64.4.1	<a href="#">name</a>	602
6.64.4.2	<a href="#">TeX_name</a>	602
6.64.4.3	<a href="#">nparams</a>	602
6.64.4.4	<a href="#">eval_f</a>	603
6.64.4.5	<a href="#">evalf_f</a>	603
6.64.4.6	<a href="#">conjugate_f</a>	603
6.64.4.7	<a href="#">real_part_f</a>	603
6.64.4.8	<a href="#">imag_part_f</a>	603

6.64.4.9	<a href="#">expand_f</a>	603
6.64.4.10	<a href="#">derivative_f</a>	604
6.64.4.11	<a href="#">expl_derivative_f</a>	604
6.64.4.12	<a href="#">power_f</a>	604
6.64.4.13	<a href="#">series_f</a>	604
6.64.4.14	<a href="#">print_dispatch_table</a>	604
6.64.4.15	<a href="#">info_f</a>	604
6.64.4.16	<a href="#">evalf_params_first</a>	605
6.64.4.17	<a href="#">use_return_type</a>	605
6.64.4.18	<a href="#">return_type</a>	605
6.64.4.19	<a href="#">return_type_tinfo</a>	605
6.64.4.20	<a href="#">use_remember</a>	605
6.64.4.21	<a href="#">remember_size</a>	605
6.64.4.22	<a href="#">remember_assoc_size</a>	606
6.64.4.23	<a href="#">remember_strategy</a>	606
6.64.4.24	<a href="#">eval_use_exvector_args</a>	606
6.64.4.25	<a href="#">evalf_use_exvector_args</a>	606
6.64.4.26	<a href="#">conjugate_use_exvector_args</a>	606
6.64.4.27	<a href="#">real_part_use_exvector_args</a>	606
6.64.4.28	<a href="#">imag_part_use_exvector_args</a>	607
6.64.4.29	<a href="#">expand_use_exvector_args</a>	607
6.64.4.30	<a href="#">derivative_use_exvector_args</a>	607
6.64.4.31	<a href="#">expl_derivative_use_exvector_args</a>	607
6.64.4.32	<a href="#">power_use_exvector_args</a>	607
6.64.4.33	<a href="#">series_use_exvector_args</a>	607
6.64.4.34	<a href="#">print_use_exvector_args</a>	608
6.64.4.35	<a href="#">info_use_exvector_args</a>	608
6.64.4.36	<a href="#">functions_with_same_name</a>	608
6.64.4.37	<a href="#">symtree</a>	608
6.65	<a href="#">GiNaC::G2_SERIAL Class Reference</a>	608

6.65.1 Detailed Description . . . . .	609
6.65.2 Member Data Documentation . . . . .	609
6.65.2.1 serial . . . . .	609
6.66 GiNaC::G3_SERIAL Class Reference . . . . .	609
6.66.1 Detailed Description . . . . .	609
6.66.2 Member Data Documentation . . . . .	609
6.66.2.1 serial . . . . .	610
6.67 GiNaC::gcd_options Struct Reference . . . . .	610
6.67.1 Detailed Description . . . . .	610
6.67.2 Member Enumeration Documentation . . . . .	610
6.67.2.1 anonymous enum . . . . .	610
6.68 GiNaC::gcdheu_failed Class Reference . . . . .	611
6.68.1 Detailed Description . . . . .	611
6.69 GiNaC::has_distance< T > Class Template Reference . . . . .	611
6.69.1 Detailed Description . . . . .	611
6.69.2 Member Typedef Documentation . . . . .	612
6.69.2.1 yes_type . . . . .	612
6.69.2.2 no_type . . . . .	612
6.69.3 Member Enumeration Documentation . . . . .	612
6.69.3.1 anonymous enum . . . . .	612
6.69.4 Member Function Documentation . . . . .	612
6.69.4.1 test() [1/2] . . . . .	612
6.69.4.2 test() [2/2] . . . . .	613
6.70 GiNaC::has_options Class Reference . . . . .	613
6.70.1 Detailed Description . . . . .	613
6.70.2 Member Enumeration Documentation . . . . .	613
6.70.2.1 anonymous enum . . . . .	613
6.71 std::hash< GiNaC::ex > Struct Template Reference . . . . .	613
6.71.1 Detailed Description . . . . .	614
6.71.2 Member Function Documentation . . . . .	614

6.71.2.1	<code>operator()</code>	614
6.72	GiNaC::idx Class Reference	614
6.72.1	Detailed Description	616
6.72.2	Constructor & Destructor Documentation	616
6.72.2.1	<code>idx()</code>	616
6.72.3	Member Function Documentation	616
6.72.3.1	<code>info()</code>	616
6.72.3.2	<code>nops()</code>	617
6.72.3.3	<code>op()</code>	617
6.72.3.4	<code>map()</code>	617
6.72.3.5	<code>evalf()</code>	618
6.72.3.6	<code>subs()</code>	618
6.72.3.7	<code>archive()</code>	618
6.72.3.8	<code>read_archive()</code>	619
6.72.3.9	<code>derivative()</code>	619
6.72.3.10	<code>match_same_type()</code>	619
6.72.3.11	<code>calchash()</code>	620
6.72.3.12	<code>is_dummy_pair_same_type()</code>	620
6.72.3.13	<code>get_value()</code>	620
6.72.3.14	<code>is_numeric()</code>	620
6.72.3.15	<code>is_symbolic()</code>	621
6.72.3.16	<code>get_dim()</code>	621
6.72.3.17	<code>is_dim_numeric()</code>	621
6.72.3.18	<code>is_dim_symbolic()</code>	621
6.72.3.19	<code>replace_dim()</code>	622
6.72.3.20	<code>minimal_dim()</code>	622
6.72.3.21	<code>print_index()</code>	622
6.72.3.22	<code>do_print()</code>	622
6.72.3.23	<code>do_print_csrc()</code>	623
6.72.3.24	<code>do_print_latex()</code>	623

6.72.3.25 <code>do_print_tree()</code> . . . . .	623
6.72.4 Member Data Documentation . . . . .	623
6.72.4.1 <code>value</code> . . . . .	623
6.72.4.2 <code>dim</code> . . . . .	624
6.73 <code>GiNaC::idx_is_equal_ignore_dim</code> Struct Reference . . . . .	624
6.73.1 Member Function Documentation . . . . .	624
6.73.1.1 <code>operator()</code> . . . . .	624
6.74 <code>GiNaC::indexed</code> Class Reference . . . . .	624
6.74.1 Detailed Description . . . . .	626
6.74.2 Constructor & Destructor Documentation . . . . .	626
6.74.2.1 <code>indexed()</code> [1/13] . . . . .	626
6.74.2.2 <code>indexed()</code> [2/13] . . . . .	627
6.74.2.3 <code>indexed()</code> [3/13] . . . . .	627
6.74.2.4 <code>indexed()</code> [4/13] . . . . .	628
6.74.2.5 <code>indexed()</code> [5/13] . . . . .	628
6.74.2.6 <code>indexed()</code> [6/13] . . . . .	629
6.74.2.7 <code>indexed()</code> [7/13] . . . . .	629
6.74.2.8 <code>indexed()</code> [8/13] . . . . .	630
6.74.2.9 <code>indexed()</code> [9/13] . . . . .	630
6.74.2.10 <code>indexed()</code> [10/13] . . . . .	631
6.74.2.11 <code>indexed()</code> [11/13] . . . . .	631
6.74.2.12 <code>indexed()</code> [12/13] . . . . .	631
6.74.2.13 <code>indexed()</code> [13/13] . . . . .	631
6.74.3 Member Function Documentation . . . . .	632
6.74.3.1 <code>precedence()</code> . . . . .	632
6.74.3.2 <code>info()</code> . . . . .	632
6.74.3.3 <code>eval()</code> . . . . .	632
6.74.3.4 <code>real_part()</code> . . . . .	633
6.74.3.5 <code>imag_part()</code> . . . . .	633
6.74.3.6 <code>get_free_indices()</code> . . . . .	633

6.74.3.7	<a href="#">archive()</a>	633
6.74.3.8	<a href="#">read_archive()</a>	634
6.74.3.9	<a href="#">derivative()</a>	634
6.74.3.10	<a href="#">thiscontainer()</a> [1/2]	634
6.74.3.11	<a href="#">thiscontainer()</a> [2/2]	634
6.74.3.12	<a href="#">return_type()</a>	635
6.74.3.13	<a href="#">return_type_tinfo()</a>	635
6.74.3.14	<a href="#">expand()</a>	635
6.74.3.15	<a href="#">all_index_values_are()</a>	635
6.74.3.16	<a href="#">get_indices()</a>	636
6.74.3.17	<a href="#">get_dummy_indices()</a> [1/2]	636
6.74.3.18	<a href="#">get_dummy_indices()</a> [2/2]	636
6.74.3.19	<a href="#">has_dummy_index_for()</a>	636
6.74.3.20	<a href="#">get_symmetry()</a>	636
6.74.3.21	<a href="#">printindices()</a>	637
6.74.3.22	<a href="#">print_indexed()</a>	637
6.74.3.23	<a href="#">do_print()</a>	637
6.74.3.24	<a href="#">do_print_latex()</a>	637
6.74.3.25	<a href="#">do_print_tree()</a>	637
6.74.3.26	<a href="#">validate()</a>	638
6.74.4	<a href="#">Friends And Related Function Documentation</a>	638
6.74.4.1	<a href="#">simplify_indexed</a>	638
6.74.4.2	<a href="#">simplify_indexed_product</a>	638
6.74.4.3	<a href="#">reposition_dummy_indices</a>	638
6.74.5	<a href="#">Member Data Documentation</a>	639
6.74.5.1	<a href="#">symtree</a>	639
6.75	<a href="#">GiNaC::info_flags Class Reference</a>	639
6.75.1	<a href="#">Detailed Description</a>	640
6.75.2	<a href="#">Member Enumeration Documentation</a>	640
6.75.2.1	<a href="#">anonymous enum</a>	640



6.76 GiNaC::integral Class Reference . . . . .	641
6.76.1 Detailed Description . . . . .	642
6.76.2 Constructor & Destructor Documentation . . . . .	642
6.76.2.1 integral() . . . . .	642
6.76.3 Member Function Documentation . . . . .	642
6.76.3.1 precedence() . . . . .	643
6.76.3.2 eval() . . . . .	643
6.76.3.3 evalf() . . . . .	643
6.76.3.4 degree() . . . . .	643
6.76.3.5 ldegree() . . . . .	644
6.76.3.6 eval_ncmul() . . . . .	644
6.76.3.7 nops() . . . . .	644
6.76.3.8 op() . . . . .	644
6.76.3.9 let_op() . . . . .	645
6.76.3.10 expand() . . . . .	645
6.76.3.11 get_free_indices() . . . . .	645
6.76.3.12 return_type() . . . . .	645
6.76.3.13 return_type_tinfo() . . . . .	646
6.76.3.14 conjugate() . . . . .	646
6.76.3.15 eval_integ() . . . . .	646
6.76.3.16 archive() . . . . .	646
6.76.3.17 read_archive() . . . . .	647
6.76.3.18 derivative() . . . . .	647
6.76.3.19 series() . . . . .	647
6.76.3.20 do_print() . . . . .	648
6.76.3.21 do_print_latex() . . . . .	648
6.76.4 Member Data Documentation . . . . .	648
6.76.4.1 max_integration_level . . . . .	648
6.76.4.2 relative_integration_error . . . . .	648
6.76.4.3 x . . . . .	648

6.76.4.4	a	649
6.76.4.5	b	649
6.76.4.6	f	649
6.77	GiNaC::integration_kernel Class Reference	649
6.77.1	Detailed Description	651
6.77.2	Member Function Documentation	651
6.77.2.1	series()	651
6.77.2.2	has_trailing_zero()	652
6.77.2.3	is_numeric()	652
6.77.2.4	Laurent_series()	652
6.77.2.5	get_numerical_value()	652
6.77.2.6	uses_Laurent_series()	653
6.77.2.7	series_coeff_impl()	653
6.77.2.8	get_cache_size()	653
6.77.2.9	set_cache_step()	653
6.77.2.10	get_series_coeff()	653
6.77.2.11	series_coeff()	654
6.77.2.12	get_numerical_value_impl()	654
6.77.2.13	do_print()	654
6.77.3	Member Data Documentation	654
6.77.3.1	cache_step_size	654
6.77.3.2	series_vec	655
6.78	GiNaC::is_not_a_clifford Struct Reference	655
6.78.1	Detailed Description	655
6.78.2	Member Function Documentation	655
6.78.2.1	operator()	655
6.79	GiNaC::is_summation_idx Struct Reference	655
6.79.1	Member Function Documentation	655
6.79.1.1	operator()	656
6.80	GiNaC::iterated_integral2_SERIAL Class Reference	656

6.80.1 Detailed Description . . . . .	656
6.80.2 Member Data Documentation . . . . .	656
6.80.2.1 serial . . . . .	656
6.81 GiNaC::iterated_integral3_SERIAL Class Reference . . . . .	657
6.81.1 Detailed Description . . . . .	657
6.81.2 Member Data Documentation . . . . .	657
6.81.2.1 serial . . . . .	657
6.82 GiNaC::Kronecker_dtau_kernel Class Reference . . . . .	657
6.82.1 Detailed Description . . . . .	658
6.82.2 Constructor & Destructor Documentation . . . . .	658
6.82.2.1 Kronecker_dtau_kernel() . . . . .	658
6.82.3 Member Function Documentation . . . . .	659
6.82.3.1 nops() . . . . .	659
6.82.3.2 op() . . . . .	659
6.82.3.3 let_op() . . . . .	659
6.82.3.4 is_numeric() . . . . .	659
6.82.3.5 get_numerical_value() . . . . .	660
6.82.3.6 series_coeff_impl() . . . . .	660
6.82.3.7 do_print() . . . . .	660
6.82.4 Member Data Documentation . . . . .	660
6.82.4.1 n . . . . .	660
6.82.4.2 z . . . . .	661
6.82.4.3 K . . . . .	661
6.82.4.4 C_norm . . . . .	661
6.83 GiNaC::Kronecker_dz_kernel Class Reference . . . . .	661
6.83.1 Detailed Description . . . . .	662
6.83.2 Constructor & Destructor Documentation . . . . .	662
6.83.2.1 Kronecker_dz_kernel() . . . . .	662
6.83.3 Member Function Documentation . . . . .	663
6.83.3.1 nops() . . . . .	663

6.83.3.2	<a href="#">op()</a>	663
6.83.3.3	<a href="#">let_op()</a>	663
6.83.3.4	<a href="#">is_numeric()</a>	663
6.83.3.5	<a href="#">get_numerical_value()</a>	664
6.83.3.6	<a href="#">series_coeff_impl()</a>	664
6.83.3.7	<a href="#">do_print()</a>	664
6.83.4	<a href="#">Member Data Documentation</a>	664
6.83.4.1	<a href="#">n</a>	664
6.83.4.2	<a href="#">z_j</a>	665
6.83.4.3	<a href="#">tau</a>	665
6.83.4.4	<a href="#">K</a>	665
6.83.4.5	<a href="#">C_norm</a>	665
6.84	<a href="#">GiNaC::lanczos_coeffs Class Reference</a>	665
6.84.1	<a href="#">Constructor &amp; Destructor Documentation</a>	666
6.84.1.1	<a href="#">lanczos_coeffs()</a>	666
6.84.2	<a href="#">Member Function Documentation</a>	666
6.84.2.1	<a href="#">sufficiently_accurate()</a>	666
6.84.2.2	<a href="#">get_order()</a>	666
6.84.2.3	<a href="#">calc_lanczos_A()</a>	666
6.84.3	<a href="#">Member Data Documentation</a>	667
6.84.3.1	<a href="#">coeffs</a>	667
6.84.3.2	<a href="#">current_vector</a>	667
6.85	<a href="#">std::less&lt; GiNaC::ptr&lt; T &gt; &gt; Struct Template Reference</a>	667
6.85.1	<a href="#">Detailed Description</a>	667
6.85.2	<a href="#">Member Function Documentation</a>	667
6.85.2.1	<a href="#">operator()()</a>	668
6.86	<a href="#">GiNaC::library_init Class Reference</a>	668
6.86.1	<a href="#">Detailed Description</a>	668
6.86.2	<a href="#">Constructor &amp; Destructor Documentation</a>	669
6.86.2.1	<a href="#">library_init()</a>	669

6.86.2.2	<a href="#">~library_init()</a>	669
6.86.3	<a href="#">Member Function Documentation</a>	669
6.86.3.1	<a href="#">init_unarchivers()</a>	670
6.86.4	<a href="#">Member Data Documentation</a>	670
6.86.4.1	<a href="#">count</a>	670
6.87	<a href="#">GiNaC::make_flat_inserter Class Reference</a>	670
6.87.1	<a href="#">Detailed Description</a>	671
6.87.2	<a href="#">Constructor &amp; Destructor Documentation</a>	671
6.87.2.1	<a href="#">make_flat_inserter()</a> [1/2]	671
6.87.2.2	<a href="#">make_flat_inserter()</a> [2/2]	671
6.87.3	<a href="#">Member Function Documentation</a>	671
6.87.3.1	<a href="#">handle_factor()</a>	671
6.87.3.2	<a href="#">combine_indices()</a>	672
6.87.4	<a href="#">Member Data Documentation</a>	672
6.87.4.1	<a href="#">do_renaming</a>	672
6.87.4.2	<a href="#">used_indices</a>	672
6.88	<a href="#">GiNaC::map_function Struct Reference</a>	672
6.88.1	<a href="#">Detailed Description</a>	673
6.88.2	<a href="#">Member Typedef Documentation</a>	673
6.88.2.1	<a href="#">argument_type</a>	673
6.88.2.2	<a href="#">result_type</a>	674
6.88.3	<a href="#">Constructor &amp; Destructor Documentation</a>	674
6.88.3.1	<a href="#">~map_function()</a>	674
6.88.4	<a href="#">Member Function Documentation</a>	674
6.88.4.1	<a href="#">operator()()</a>	674
6.89	<a href="#">GiNaC::matrix Class Reference</a>	674
6.89.1	<a href="#">Detailed Description</a>	677
6.89.2	<a href="#">Constructor &amp; Destructor Documentation</a>	677
6.89.2.1	<a href="#">matrix()</a> [1/5]	677
6.89.2.2	<a href="#">matrix()</a> [2/5]	677

6.89.2.3	<a href="#">matrix()</a> [3/5]	678
6.89.2.4	<a href="#">matrix()</a> [4/5]	678
6.89.2.5	<a href="#">matrix()</a> [5/5]	678
6.89.3	<a href="#">Member Function Documentation</a>	678
6.89.3.1	<a href="#">nops()</a>	678
6.89.3.2	<a href="#">op()</a>	679
6.89.3.3	<a href="#">let_op()</a>	679
6.89.3.4	<a href="#">evalm()</a>	679
6.89.3.5	<a href="#">subs()</a>	679
6.89.3.6	<a href="#">eval_indexed()</a>	680
6.89.3.7	<a href="#">add_indexed()</a>	680
6.89.3.8	<a href="#">scalar_mul_indexed()</a>	680
6.89.3.9	<a href="#">contract_with()</a>	680
6.89.3.10	<a href="#">conjugate()</a>	681
6.89.3.11	<a href="#">real_part()</a>	681
6.89.3.12	<a href="#">imag_part()</a>	681
6.89.3.13	<a href="#">archive()</a>	681
6.89.3.14	<a href="#">read_archive()</a>	682
6.89.3.15	<a href="#">match_same_type()</a>	682
6.89.3.16	<a href="#">return_type()</a>	682
6.89.3.17	<a href="#">rows()</a>	683
6.89.3.18	<a href="#">cols()</a>	683
6.89.3.19	<a href="#">add()</a>	683
6.89.3.20	<a href="#">sub()</a>	683
6.89.3.21	<a href="#">mul()</a> [1/2]	684
6.89.3.22	<a href="#">mul()</a> [2/2]	684
6.89.3.23	<a href="#">mul_scalar()</a>	684
6.89.3.24	<a href="#">pow()</a>	685
6.89.3.25	<a href="#">operator()</a> [1/2]	685
6.89.3.26	<a href="#">operator()</a> [2/2]	685

6.89.3.27 set()	686
6.89.3.28 transpose()	686
6.89.3.29 determinant()	686
6.89.3.30 trace()	687
6.89.3.31 charpoly()	687
6.89.3.32 inverse() <sup>[1/2]</sup>	688
6.89.3.33 inverse() <sup>[2/2]</sup>	688
6.89.3.34 solve()	689
6.89.3.35 rank() <sup>[1/2]</sup>	690
6.89.3.36 rank() <sup>[2/2]</sup>	690
6.89.3.37 is_zero_matrix()	690
6.89.3.38 determinant_minor()	690
6.89.3.39 echelon_form()	691
6.89.3.40 gauss_elimination()	691
6.89.3.41 division_free_elimination()	691
6.89.3.42 fraction_free_elimination()	692
6.89.3.43 markowitz_elimination()	692
6.89.3.44 pivot()	693
6.89.3.45 print_elements()	693
6.89.3.46 do_print()	693
6.89.3.47 do_print_latex()	694
6.89.3.48 do_print_python_repr()	694
6.89.4 Member Data Documentation	694
6.89.4.1 row	694
6.89.4.2 col	694
6.89.4.3 m	695
6.90 GiNaC::minkmetric Class Reference	695
6.90.1 Detailed Description	696
6.90.2 Constructor & Destructor Documentation	696
6.90.2.1 minkmetric()	696

6.90.3	Member Function Documentation	696
6.90.3.1	info()	696
6.90.3.2	eval_indexed()	697
6.90.3.3	archive()	697
6.90.3.4	read_archive()	697
6.90.3.5	return_type()	697
6.90.3.6	do_print()	698
6.90.3.7	do_print_latex()	698
6.90.4	Member Data Documentation	698
6.90.4.1	pos_sig	698
6.91	GiNaC::modular_form_kernel Class Reference	698
6.91.1	Detailed Description	699
6.91.2	Constructor & Destructor Documentation	699
6.91.2.1	modular_form_kernel()	700
6.91.3	Member Function Documentation	700
6.91.3.1	series()	700
6.91.3.2	nops()	700
6.91.3.3	op()	700
6.91.3.4	let_op()	701
6.91.3.5	is_numeric()	701
6.91.3.6	Laurent_series()	701
6.91.3.7	get_numerical_value()	701
6.91.3.8	uses_Laurent_series()	702
6.91.3.9	q_expansion_modular_form()	702
6.91.3.10	do_print()	702
6.91.4	Member Data Documentation	702
6.91.4.1	k	702
6.91.4.2	P	702
6.91.4.3	C_norm	703
6.92	GiNaC::basic_partition_generator::mpartition2 Struct Reference	703



6.92.1	Constructor & Destructor Documentation . . . . .	703
6.92.1.1	mpartition2() . . . . .	703
6.92.2	Member Function Documentation . . . . .	703
6.92.2.1	next_partition() . . . . .	704
6.92.3	Member Data Documentation . . . . .	704
6.92.3.1	x . . . . .	704
6.92.3.2	n . . . . .	704
6.92.3.3	m . . . . .	704
6.93	GiNaC::mul Class Reference . . . . .	705
6.93.1	Detailed Description . . . . .	707
6.93.2	Constructor & Destructor Documentation . . . . .	707
6.93.2.1	mul() [1/7] . . . . .	707
6.93.2.2	mul() [2/7] . . . . .	707
6.93.2.3	mul() [3/7] . . . . .	707
6.93.2.4	mul() [4/7] . . . . .	708
6.93.2.5	mul() [5/7] . . . . .	708
6.93.2.6	mul() [6/7] . . . . .	708
6.93.2.7	mul() [7/7] . . . . .	708
6.93.3	Member Function Documentation . . . . .	708
6.93.3.1	precedence() . . . . .	709
6.93.3.2	info() . . . . .	709
6.93.3.3	is_polynomial() . . . . .	709
6.93.3.4	degree() . . . . .	709
6.93.3.5	ldegree() . . . . .	710
6.93.3.6	coeff() . . . . .	710
6.93.3.7	has() . . . . .	710
6.93.3.8	eval() . . . . .	711
6.93.3.9	evalf() . . . . .	711
6.93.3.10	real_part() . . . . .	711
6.93.3.11	imag_part() . . . . .	711

6.93.3.12 evalm()	712
6.93.3.13 series()	712
6.93.3.14 normal()	712
6.93.3.15 integer_content()	713
6.93.3.16 smod()	713
6.93.3.17 max_coefficient()	713
6.93.3.18 get_free_indices()	714
6.93.3.19 conjugate()	714
6.93.3.20 derivative()	714
6.93.3.21 eval_ncmul()	714
6.93.3.22 return_type()	715
6.93.3.23 return_type_tinfo()	715
6.93.3.24 thisexpairseq() [1/2]	715
6.93.3.25 thisexpairseq() [2/2]	715
6.93.3.26 split_ex_to_pair()	715
6.93.3.27 combine_ex_with_coeff_to_pair()	716
6.93.3.28 combine_pair_with_coeff_to_pair()	716
6.93.3.29 recombine_pair_to_ex()	716
6.93.3.30 expair_needs_further_processing()	716
6.93.3.31 default_overall_coeff()	717
6.93.3.32 combine_overall_coeff() [1/2]	717
6.93.3.33 combine_overall_coeff() [2/2]	717
6.93.3.34 can_make_flat()	717
6.93.3.35 expand()	718
6.93.3.36 algebraic_subs_mul()	718
6.93.3.37 find_real_imag()	718
6.93.3.38 print_overall_coeff()	718
6.93.3.39 do_print()	719
6.93.3.40 do_print_latex()	719
6.93.3.41 do_print_csrc()	719

6.93.3.42	<a href="#">do_print_python_repr()</a>	719
6.93.3.43	<a href="#">can_be_further_expanded()</a>	719
6.93.3.44	<a href="#">expandchildren()</a>	720
6.93.4	<a href="#">Friends And Related Function Documentation</a>	720
6.93.4.1	<a href="#">add</a>	720
6.93.4.2	<a href="#">ncmul</a>	720
6.93.4.3	<a href="#">power</a>	720
6.94	<a href="#">GiNaC::multi_iterator_counter&lt; T &gt; Class Template Reference</a>	721
6.94.1	<a href="#">Detailed Description</a>	721
6.94.2	<a href="#">Constructor &amp; Destructor Documentation</a>	722
6.94.2.1	<a href="#">multi_iterator_counter() [1/3]</a>	722
6.94.2.2	<a href="#">multi_iterator_counter() [2/3]</a>	722
6.94.2.3	<a href="#">multi_iterator_counter() [3/3]</a>	722
6.94.3	<a href="#">Member Function Documentation</a>	722
6.94.3.1	<a href="#">init()</a>	722
6.94.3.2	<a href="#">operator++()</a>	723
6.94.4	<a href="#">Friends And Related Function Documentation</a>	723
6.94.4.1	<a href="#">operator&lt;&lt;</a>	723
6.95	<a href="#">GiNaC::multi_iterator_counter_indv&lt; T &gt; Class Template Reference</a>	723
6.95.1	<a href="#">Detailed Description</a>	724
6.95.2	<a href="#">Constructor &amp; Destructor Documentation</a>	724
6.95.2.1	<a href="#">multi_iterator_counter_indv() [1/3]</a>	724
6.95.2.2	<a href="#">multi_iterator_counter_indv() [2/3]</a>	725
6.95.2.3	<a href="#">multi_iterator_counter_indv() [3/3]</a>	725
6.95.3	<a href="#">Member Function Documentation</a>	725
6.95.3.1	<a href="#">init()</a>	725
6.95.3.2	<a href="#">operator++()</a>	725
6.95.4	<a href="#">Friends And Related Function Documentation</a>	726
6.95.4.1	<a href="#">operator&lt;&lt;</a>	726
6.95.5	<a href="#">Member Data Documentation</a>	726

6.95.5.1	Nv	726
6.96	GiNaC::multi_iterator_ordered< T > Class Template Reference	726
6.96.1	Detailed Description	727
6.96.2	Constructor & Destructor Documentation	727
6.96.2.1	multi_iterator_ordered() [1/3]	727
6.96.2.2	multi_iterator_ordered() [2/3]	728
6.96.2.3	multi_iterator_ordered() [3/3]	728
6.96.3	Member Function Documentation	728
6.96.3.1	init()	728
6.96.3.2	operator++()	728
6.96.4	Friends And Related Function Documentation	729
6.96.4.1	operator<<	729
6.97	GiNaC::multi_iterator_ordered_eq< T > Class Template Reference	729
6.97.1	Detailed Description	730
6.97.2	Constructor & Destructor Documentation	730
6.97.2.1	multi_iterator_ordered_eq() [1/3]	730
6.97.2.2	multi_iterator_ordered_eq() [2/3]	730
6.97.2.3	multi_iterator_ordered_eq() [3/3]	731
6.97.3	Member Function Documentation	731
6.97.3.1	init()	731
6.97.3.2	operator++()	731
6.97.4	Friends And Related Function Documentation	731
6.97.4.1	operator<<	732
6.98	GiNaC::multi_iterator_ordered_eq_indv< T > Class Template Reference	732
6.98.1	Detailed Description	733
6.98.2	Constructor & Destructor Documentation	733
6.98.2.1	multi_iterator_ordered_eq_indv() [1/3]	733
6.98.2.2	multi_iterator_ordered_eq_indv() [2/3]	733
6.98.2.3	multi_iterator_ordered_eq_indv() [3/3]	733
6.98.3	Member Function Documentation	734

6.98.3.1	<a href="#">init()</a>	734
6.98.3.2	<a href="#">operator++()</a>	734
6.98.4	<a href="#">Friends And Related Function Documentation</a>	734
6.98.4.1	<a href="#">operator&lt;&lt;</a>	734
6.98.5	<a href="#">Member Data Documentation</a>	734
6.98.5.1	<a href="#">Nv</a>	735
6.99	<a href="#">GiNaC::multi_iterator_permutation&lt; T &gt; Class Template Reference</a>	735
6.99.1	<a href="#">Detailed Description</a>	736
6.99.2	<a href="#">Constructor &amp; Destructor Documentation</a>	736
6.99.2.1	<a href="#">multi_iterator_permutation() [1/3]</a>	736
6.99.2.2	<a href="#">multi_iterator_permutation() [2/3]</a>	736
6.99.2.3	<a href="#">multi_iterator_permutation() [3/3]</a>	737
6.99.3	<a href="#">Member Function Documentation</a>	737
6.99.3.1	<a href="#">init()</a>	737
6.99.3.2	<a href="#">operator++()</a>	737
6.99.3.3	<a href="#">get_sign()</a>	738
6.99.4	<a href="#">Friends And Related Function Documentation</a>	738
6.99.4.1	<a href="#">operator&lt;&lt;</a>	738
6.100	<a href="#">GiNaC::multi_iterator_shuffle&lt; T &gt; Class Template Reference</a>	738
6.100.1	<a href="#">Detailed Description</a>	739
6.100.2	<a href="#">Constructor &amp; Destructor Documentation</a>	739
6.100.2.1	<a href="#">multi_iterator_shuffle() [1/2]</a>	739
6.100.2.2	<a href="#">multi_iterator_shuffle() [2/2]</a>	740
6.100.3	<a href="#">Member Function Documentation</a>	740
6.100.3.1	<a href="#">init()</a>	740
6.100.3.2	<a href="#">operator++()</a>	740
6.100.4	<a href="#">Friends And Related Function Documentation</a>	740
6.100.4.1	<a href="#">operator&lt;&lt;</a>	741
6.100.5	<a href="#">Member Data Documentation</a>	741
6.100.5.1	<a href="#">N_internal</a>	741

6.100.5.2 v_internal . . . . .	741
6.100.5.3 v_orig . . . . .	741
6.101 GiNaC::multi_iterator_shuffle_prime< T > Class Template Reference . . . . .	742
6.101.1 Detailed Description . . . . .	742
6.101.2 Constructor & Destructor Documentation . . . . .	742
6.101.2.1 multi_iterator_shuffle_prime() [1/2] . . . . .	743
6.101.2.2 multi_iterator_shuffle_prime() [2/2] . . . . .	743
6.101.3 Member Function Documentation . . . . .	743
6.101.3.1 init() . . . . .	743
6.101.4 Friends And Related Function Documentation . . . . .	743
6.101.4.1 operator<< . . . . .	743
6.102 GiNaC::multiple_polylog_kernel Class Reference . . . . .	744
6.102.1 Detailed Description . . . . .	744
6.102.2 Constructor & Destructor Documentation . . . . .	745
6.102.2.1 multiple_polylog_kernel() . . . . .	745
6.102.3 Member Function Documentation . . . . .	745
6.102.3.1 nops() . . . . .	745
6.102.3.2 op() . . . . .	745
6.102.3.3 let_op() . . . . .	745
6.102.3.4 is_numeric() . . . . .	746
6.102.3.5 series_coeff_impl() . . . . .	746
6.102.3.6 do_print() . . . . .	746
6.102.4 Member Data Documentation . . . . .	746
6.102.4.1 z . . . . .	746
6.103 GiNaC::ncmul Class Reference . . . . .	747
6.103.1 Detailed Description . . . . .	748
6.103.2 Constructor & Destructor Documentation . . . . .	748
6.103.2.1 ncmul() [1/7] . . . . .	748
6.103.2.2 ncmul() [2/7] . . . . .	748
6.103.2.3 ncmul() [3/7] . . . . .	749

6.103.2.4 ncmul() [ 4 / 7 ] . . . . .	749
6.103.2.5 ncmul() [ 5 / 7 ] . . . . .	749
6.103.2.6 ncmul() [ 6 / 7 ] . . . . .	749
6.103.2.7 ncmul() [ 7 / 7 ] . . . . .	749
6.103.3 Member Function Documentation . . . . .	749
6.103.3.1 precedence() . . . . .	750
6.103.3.2 info() . . . . .	750
6.103.3.3 degree() . . . . .	750
6.103.3.4 ldegree() . . . . .	750
6.103.3.5 expand() . . . . .	751
6.103.3.6 coeff() . . . . .	751
6.103.3.7 eval() . . . . .	751
6.103.3.8 evalm() . . . . .	752
6.103.3.9 get_free_indices() . . . . .	752
6.103.3.10 thiscontainer() [ 1 / 2 ] . . . . .	752
6.103.3.11 thiscontainer() [ 2 / 2 ] . . . . .	752
6.103.3.12 conjugate() . . . . .	752
6.103.3.13 real_part() . . . . .	753
6.103.3.14 imag_part() . . . . .	753
6.103.3.15 derivative() . . . . .	753
6.103.3.16 return_type() . . . . .	753
6.103.3.17 return_type_tinfo() . . . . .	754
6.103.3.18 do_print() . . . . .	754
6.103.3.19 do_print_csrc() . . . . .	754
6.103.3.20 count_factors() . . . . .	754
6.103.3.21 append_factors() . . . . .	754
6.103.3.22 expandchildren() . . . . .	755
6.103.3.23 get_factors() . . . . .	755
6.103.4 Friends And Related Function Documentation . . . . .	755
6.103.4.1 power . . . . .	755

6.103.4.2 reeval_ncmul . . . . .	755
6.103.4.3 hold_ncmul . . . . .	755
6.104GiNaC::normal_map_function Struct Reference . . . . .	756
6.104.1 Detailed Description . . . . .	756
6.104.2 Member Function Documentation . . . . .	756
6.104.2.1 operator()() . . . . .	756
6.105GiNaC::numeric Class Reference . . . . .	757
6.105.1 Detailed Description . . . . .	760
6.105.2 Constructor & Destructor Documentation . . . . .	760
6.105.2.1 numeric() [1/10] . . . . .	760
6.105.2.2 numeric() [2/10] . . . . .	761
6.105.2.3 numeric() [3/10] . . . . .	761
6.105.2.4 numeric() [4/10] . . . . .	761
6.105.2.5 numeric() [5/10] . . . . .	761
6.105.2.6 numeric() [6/10] . . . . .	761
6.105.2.7 numeric() [7/10] . . . . .	761
6.105.2.8 numeric() [8/10] . . . . .	762
6.105.2.9 numeric() [9/10] . . . . .	762
6.105.2.10numeric() [10/10] . . . . .	762
6.105.3 Member Function Documentation . . . . .	762
6.105.3.1 precedence() . . . . .	763
6.105.3.2 info() . . . . .	763
6.105.3.3 is_polynomial() . . . . .	763
6.105.3.4 degree() . . . . .	764
6.105.3.5 ldegree() . . . . .	764
6.105.3.6 coeff() . . . . .	764
6.105.3.7 has() . . . . .	764
6.105.3.8 eval() . . . . .	765
6.105.3.9 evalf() . . . . .	765
6.105.3.10subs() . . . . .	765



6.105.3.11	<code>normal()</code>	766
6.105.3.12	<code>to_rational()</code>	766
6.105.3.13	<code>to_polynomial()</code>	766
6.105.3.14	<code>integer_content()</code>	767
6.105.3.15	<code>smod()</code>	767
6.105.3.16	<code>max_coefficient()</code>	768
6.105.3.17	<code>conjugate()</code>	768
6.105.3.18	<code>real_part()</code>	768
6.105.3.19	<code>imag_part()</code>	769
6.105.3.20	<code>archive()</code>	769
6.105.3.21	<code>read_archive()</code>	769
6.105.3.22	<code>derivative()</code>	769
6.105.3.23	<code>s_equal_same_type()</code>	770
6.105.3.24	<code>calchash()</code>	770
6.105.3.25	<code>add()</code>	770
6.105.3.26	<code>sub()</code>	771
6.105.3.27	<code>mul()</code>	771
6.105.3.28	<code>div()</code>	771
6.105.3.29	<code>power()</code>	772
6.105.3.30	<code>add_dyn()</code>	772
6.105.3.31	<code>sub_dyn()</code>	772
6.105.3.32	<code>mul_dyn()</code>	772
6.105.3.33	<code>div_dyn()</code>	772
6.105.3.34	<code>power_dyn()</code>	773
6.105.3.35	<code>operator=()</code> [1/6]	773
6.105.3.36	<code>operator=()</code> [2/6]	773
6.105.3.37	<code>operator=()</code> [3/6]	773
6.105.3.38	<code>operator=()</code> [4/6]	774
6.105.3.39	<code>operator=()</code> [5/6]	774
6.105.3.40	<code>operator=()</code> [6/6]	774

6.105.3.41inverse()	774
6.105.3.42step()	774
6.105.3.43csgn()	775
6.105.3.44compare()	775
6.105.3.45s_equal()	775
6.105.3.46s_zero()	776
6.105.3.47s_positive()	776
6.105.3.48s_negative()	776
6.105.3.49s_integer()	776
6.105.3.50s_pos_integer()	777
6.105.3.51is_nonneg_integer()	777
6.105.3.52s_even()	777
6.105.3.53s_odd()	777
6.105.3.54s_prime()	778
6.105.3.55s_rational()	778
6.105.3.56s_real()	778
6.105.3.57s_cinteger()	778
6.105.3.58s_crational()	779
6.105.3.59operator==(())	779
6.105.3.60operator!=(())	779
6.105.3.61operator<()	779
6.105.3.62operator<=()	779
6.105.3.63operator>()	781
6.105.3.64operator>=()	781
6.105.3.65to_int()	781
6.105.3.66to_long()	782
6.105.3.67to_double()	782
6.105.3.68to_cl_N()	782
6.105.3.69real()	782
6.105.3.70mag()	783

6.105.3.71	<code>numer()</code> . . . . .	783
6.105.3.72	<code>denom()</code> . . . . .	783
6.105.3.73	<code>int_length()</code> . . . . .	783
6.105.3.74	<code>print_numeric()</code> . . . . .	784
6.105.3.75	<code>do_print()</code> . . . . .	784
6.105.3.76	<code>do_print_latex()</code> . . . . .	784
6.105.3.77	<code>do_print_csrc()</code> . . . . .	784
6.105.3.78	<code>do_print_csrc_cl_N()</code> . . . . .	785
6.105.3.79	<code>do_print_tree()</code> . . . . .	785
6.105.3.80	<code>do_print_python_repr()</code> . . . . .	785
6.105.4	Member Data Documentation . . . . .	785
6.105.4.1	<code>value</code> . . . . .	785
6.106	<code>GiNaC::op0_is_equal</code> Struct Reference . . . . .	786
6.106.1	Member Function Documentation . . . . .	786
6.106.1.1	<code>operator&gt;()</code> . . . . .	786
6.107	<code>GiNaC::partition_generator</code> Class Reference . . . . .	786
6.107.1	Detailed Description . . . . .	787
6.107.2	Constructor & Destructor Documentation . . . . .	787
6.107.2.1	<code>partition_generator()</code> . . . . .	787
6.107.3	Member Function Documentation . . . . .	787
6.107.3.1	<code>get()</code> . . . . .	787
6.107.3.2	<code>next()</code> . . . . .	787
6.107.4	Member Data Documentation . . . . .	787
6.107.4.1	<code>partition</code> . . . . .	788
6.107.4.2	<code>current_updated</code> . . . . .	788
6.108	<code>GiNaC::partition_with_zero_parts_generator</code> Class Reference . . . . .	788
6.108.1	Detailed Description . . . . .	789
6.108.2	Constructor & Destructor Documentation . . . . .	789
6.108.2.1	<code>partition_with_zero_parts_generator()</code> . . . . .	789
6.108.3	Member Function Documentation . . . . .	789

6.108.3.1 get()	789
6.108.3.2 next()	789
6.108.4 Member Data Documentation	789
6.108.4.1 m	790
6.108.4.2 partition	790
6.108.4.3 current_updated	790
6.109GiNaC::pointer_to_map_function Class Reference	790
6.109.1 Constructor & Destructor Documentation	791
6.109.1.1 pointer_to_map_function()	791
6.109.2 Member Function Documentation	791
6.109.2.1 operator()()	791
6.109.3 Member Data Documentation	791
6.109.3.1 ptr	791
6.110GiNaC::pointer_to_map_function_1arg< T1 > Class Template Reference	791
6.110.1 Constructor & Destructor Documentation	792
6.110.1.1 pointer_to_map_function_1arg()	792
6.110.2 Member Function Documentation	792
6.110.2.1 operator()()	792
6.110.3 Member Data Documentation	792
6.110.3.1 ptr	793
6.110.3.2 arg1	793
6.111GiNaC::pointer_to_map_function_2args< T1, T2 > Class Template Reference	793
6.111.1 Constructor & Destructor Documentation	793
6.111.1.1 pointer_to_map_function_2args()	794
6.111.2 Member Function Documentation	794
6.111.2.1 operator()()	794
6.111.3 Member Data Documentation	794
6.111.3.1 ptr	794
6.111.3.2 arg1	794
6.111.3.3 arg2	795

6.112GiNaC::pointer_to_map_function_3args< T1, T2, T3 > Class Template Reference . . . . .	795
6.112.1 Constructor & Destructor Documentation . . . . .	795
6.112.1.1 pointer_to_map_function_3args() . . . . .	795
6.112.2 Member Function Documentation . . . . .	796
6.112.2.1 operator>() . . . . .	796
6.112.3 Member Data Documentation . . . . .	796
6.112.3.1 ptr . . . . .	796
6.112.3.2 arg1 . . . . .	796
6.112.3.3 arg2 . . . . .	796
6.112.3.4 arg3 . . . . .	797
6.113GiNaC::pointer_to_member_to_map_function< C > Class Template Reference . . . . .	797
6.113.1 Constructor & Destructor Documentation . . . . .	797
6.113.1.1 pointer_to_member_to_map_function() . . . . .	797
6.113.2 Member Function Documentation . . . . .	798
6.113.2.1 operator>() . . . . .	798
6.113.3 Member Data Documentation . . . . .	798
6.113.3.1 ptr . . . . .	798
6.113.3.2 c . . . . .	798
6.114GiNaC::pointer_to_member_to_map_function_1arg< C, T1 > Class Template Reference . . . . .	798
6.114.1 Constructor & Destructor Documentation . . . . .	799
6.114.1.1 pointer_to_member_to_map_function_1arg() . . . . .	799
6.114.2 Member Function Documentation . . . . .	799
6.114.2.1 operator>() . . . . .	799
6.114.3 Member Data Documentation . . . . .	799
6.114.3.1 ptr . . . . .	800
6.114.3.2 c . . . . .	800
6.114.3.3 arg1 . . . . .	800
6.115GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 > Class Template Reference . . . . .	800
6.115.1 Constructor & Destructor Documentation . . . . .	801
6.115.1.1 pointer_to_member_to_map_function_2args() . . . . .	801

6.115.2 Member Function Documentation . . . . .	801
6.115.2.1 operator()() . . . . .	801
6.115.3 Member Data Documentation . . . . .	801
6.115.3.1 ptr . . . . .	802
6.115.3.2 c . . . . .	802
6.115.3.3 arg1 . . . . .	802
6.115.3.4 arg2 . . . . .	802
6.116GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 > Class Template Reference	802
6.116.1 Constructor & Destructor Documentation . . . . .	803
6.116.1.1 pointer_to_member_to_map_function_3args() . . . . .	803
6.116.2 Member Function Documentation . . . . .	803
6.116.2.1 operator()() . . . . .	803
6.116.3 Member Data Documentation . . . . .	804
6.116.3.1 ptr . . . . .	804
6.116.3.2 c . . . . .	804
6.116.3.3 arg1 . . . . .	804
6.116.3.4 arg2 . . . . .	804
6.116.3.5 arg3 . . . . .	804
6.117GiNaC::pole_error Class Reference . . . . .	805
6.117.1 Detailed Description . . . . .	805
6.117.2 Constructor & Destructor Documentation . . . . .	805
6.117.2.1 pole_error() . . . . .	805
6.117.3 Member Function Documentation . . . . .	805
6.117.3.1 degree() . . . . .	806
6.117.4 Member Data Documentation . . . . .	806
6.117.4.1 deg . . . . .	806
6.118GiNaC::possymbol Class Reference . . . . .	806
6.118.1 Detailed Description . . . . .	807
6.118.2 Constructor & Destructor Documentation . . . . .	807
6.118.2.1 possymbol() [1/3] . . . . .	807

6.118.2.2 possymbol() [2/3]	807
6.118.2.3 possymbol() [3/3]	807
6.118.3 Member Function Documentation	807
6.118.3.1 get_domain()	807
6.118.3.2 duplicate()	808
6.119GiNaC::power Class Reference	808
6.119.1 Detailed Description	810
6.119.2 Constructor & Destructor Documentation	810
6.119.2.1 power() [1/2]	810
6.119.2.2 power() [2/2]	810
6.119.3 Member Function Documentation	810
6.119.3.1 precedence()	811
6.119.3.2 info()	811
6.119.3.3 nops()	811
6.119.3.4 op()	811
6.119.3.5 map()	812
6.119.3.6 is_polynomial()	812
6.119.3.7 degree()	812
6.119.3.8 ldegree()	812
6.119.3.9 coeff()	813
6.119.3.10 eval()	813
6.119.3.11 evalf()	814
6.119.3.12 evalm()	814
6.119.3.13 series()	814
6.119.3.14 subs()	815
6.119.3.15 has()	815
6.119.3.16 normal()	815
6.119.3.17 to_rational()	816
6.119.3.18 to_polynomial()	816
6.119.3.19 conjugate()	816

6.119.3.20	<a href="#">real_part()</a>	816
6.119.3.21	<a href="#">imag_part()</a>	817
6.119.3.22	<a href="#">archive()</a>	817
6.119.3.23	<a href="#">read_archive()</a>	817
6.119.3.24	<a href="#">derivative()</a>	817
6.119.3.25	<a href="#">eval_ncmul()</a>	818
6.119.3.26	<a href="#">return_type()</a>	818
6.119.3.27	<a href="#">return_type_tinfo()</a>	818
6.119.3.28	<a href="#">expand()</a>	818
6.119.3.29	<a href="#">print_power()</a>	819
6.119.3.30	<a href="#">do_print_dflt()</a>	819
6.119.3.31	<a href="#">do_print_latex()</a>	819
6.119.3.32	<a href="#">do_print_csrc()</a>	819
6.119.3.33	<a href="#">do_print_python()</a>	820
6.119.3.34	<a href="#">do_print_python_repr()</a>	820
6.119.3.35	<a href="#">do_print_csrc_cl_N()</a>	820
6.119.3.36	<a href="#">expand_add()</a>	820
6.119.3.37	<a href="#">expand_add_2()</a>	821
6.119.3.38	<a href="#">expand_mul()</a>	821
6.119.4	<a href="#">Friends And Related Function Documentation</a>	821
6.119.4.1	<a href="#">mul</a>	821
6.119.5	<a href="#">Member Data Documentation</a>	822
6.119.5.1	<a href="#">basis</a>	822
6.119.5.2	<a href="#">exponent</a>	822
6.120	<a href="#">GiNaC::print_context Class Reference</a>	822
6.120.1	<a href="#">Detailed Description</a>	823
6.120.2	<a href="#">Constructor &amp; Destructor Documentation</a>	823
6.120.2.1	<a href="#">print_context()</a>	823
6.120.2.2	<a href="#">~print_context()</a>	823
6.120.3	<a href="#">Member Data Documentation</a>	823



6.120.3.1 s . . . . .	823
6.120.3.2 options . . . . .	824
6.121GiNaC::print_context_options Class Reference . . . . .	824
6.121.1 Detailed Description . . . . .	824
6.121.2 Constructor & Destructor Documentation . . . . .	824
6.121.2.1 print_context_options() . . . . .	825
6.121.3 Member Function Documentation . . . . .	825
6.121.3.1 get_name() . . . . .	825
6.121.3.2 get_parent_name() . . . . .	825
6.121.3.3 get_id() . . . . .	825
6.121.4 Member Data Documentation . . . . .	825
6.121.4.1 name . . . . .	825
6.121.4.2 parent_name . . . . .	826
6.121.4.3 id . . . . .	826
6.122GiNaC::print_csrc Class Reference . . . . .	826
6.122.1 Detailed Description . . . . .	826
6.122.2 Constructor & Destructor Documentation . . . . .	827
6.122.2.1 print_csrc() . . . . .	827
6.123GiNaC::print_csrc_cl_N Class Reference . . . . .	827
6.123.1 Detailed Description . . . . .	827
6.123.2 Constructor & Destructor Documentation . . . . .	827
6.123.2.1 print_csrc_cl_N() . . . . .	828
6.124GiNaC::print_csrc_double Class Reference . . . . .	828
6.124.1 Detailed Description . . . . .	828
6.124.2 Constructor & Destructor Documentation . . . . .	828
6.124.2.1 print_csrc_double() . . . . .	829
6.125GiNaC::print_csrc_float Class Reference . . . . .	829
6.125.1 Detailed Description . . . . .	829
6.125.2 Constructor & Destructor Documentation . . . . .	829
6.125.2.1 print_csrc_float() . . . . .	830

6.126GiNaC::print_dflt Class Reference . . . . .	830
6.126.1 Detailed Description . . . . .	830
6.126.2 Constructor & Destructor Documentation . . . . .	830
6.126.2.1 print_dflt() . . . . .	830
6.127GiNaC::print_functor Class Reference . . . . .	831
6.127.1 Detailed Description . . . . .	831
6.127.2 Constructor & Destructor Documentation . . . . .	831
6.127.2.1 print_functor() [1/5] . . . . .	831
6.127.2.2 print_functor() [2/5] . . . . .	831
6.127.2.3 print_functor() [3/5] . . . . .	832
6.127.2.4 print_functor() [4/5] . . . . .	832
6.127.2.5 print_functor() [5/5] . . . . .	832
6.127.3 Member Function Documentation . . . . .	832
6.127.3.1 operator=() . . . . .	832
6.127.3.2 operator>() . . . . .	832
6.127.3.3 is_valid() . . . . .	833
6.127.4 Member Data Documentation . . . . .	833
6.127.4.1 impl . . . . .	833
6.128GiNaC::print_functor_impl Class Reference . . . . .	833
6.128.1 Detailed Description . . . . .	833
6.128.2 Constructor & Destructor Documentation . . . . .	834
6.128.2.1 ~print_functor_impl() . . . . .	834
6.128.3 Member Function Documentation . . . . .	834
6.128.3.1 duplicate() . . . . .	834
6.128.3.2 operator>() . . . . .	834
6.129GiNaC::print_latex Class Reference . . . . .	834
6.129.1 Detailed Description . . . . .	835
6.129.2 Constructor & Destructor Documentation . . . . .	835
6.129.2.1 print_latex() . . . . .	835
6.130GiNaC::print_memfun_handler< T, C > Class Template Reference . . . . .	835

6.130.1 Detailed Description . . . . .	836
6.130.2 Member Typedef Documentation . . . . .	836
6.130.2.1 F . . . . .	836
6.130.3 Constructor & Destructor Documentation . . . . .	836
6.130.3.1 print_memfun_handler() . . . . .	836
6.130.4 Member Function Documentation . . . . .	836
6.130.4.1 duplicate() . . . . .	837
6.130.4.2 operator>() . . . . .	837
6.130.5 Member Data Documentation . . . . .	837
6.130.5.1 f . . . . .	837
6.131 GiNaC::print_options Class Reference . . . . .	837
6.131.1 Detailed Description . . . . .	838
6.131.2 Member Enumeration Documentation . . . . .	838
6.131.2.1 anonymous enum . . . . .	838
6.132 GiNaC::print_ptrfun_handler< T, C > Class Template Reference . . . . .	838
6.132.1 Detailed Description . . . . .	839
6.132.2 Member Typedef Documentation . . . . .	839
6.132.2.1 F . . . . .	839
6.132.3 Constructor & Destructor Documentation . . . . .	839
6.132.3.1 print_ptrfun_handler() . . . . .	839
6.132.4 Member Function Documentation . . . . .	839
6.132.4.1 duplicate() . . . . .	839
6.132.4.2 operator>() . . . . .	840
6.132.5 Member Data Documentation . . . . .	840
6.132.5.1 f . . . . .	840
6.133 GiNaC::print_python Class Reference . . . . .	840
6.133.1 Detailed Description . . . . .	841
6.133.2 Constructor & Destructor Documentation . . . . .	841
6.133.2.1 print_python() . . . . .	841
6.134 GiNaC::print_python_repr Class Reference . . . . .	841

6.134.1 Detailed Description . . . . .	841
6.134.2 Constructor & Destructor Documentation . . . . .	842
6.134.2.1 print_python_repr() . . . . .	842
6.135GiNaC::print_tree Class Reference . . . . .	842
6.135.1 Detailed Description . . . . .	842
6.135.2 Constructor & Destructor Documentation . . . . .	842
6.135.2.1 print_tree() [1/2] . . . . .	843
6.135.2.2 print_tree() [2/2] . . . . .	843
6.135.3 Member Data Documentation . . . . .	843
6.135.3.1 delta_indent . . . . .	843
6.136GiNaC::archive_node::property Struct Reference . . . . .	843
6.136.1 Detailed Description . . . . .	844
6.136.2 Constructor & Destructor Documentation . . . . .	844
6.136.2.1 property() [1/2] . . . . .	844
6.136.2.2 property() [2/2] . . . . .	844
6.136.3 Member Data Documentation . . . . .	844
6.136.3.1 type . . . . .	844
6.136.3.2 name . . . . .	844
6.136.3.3 value . . . . .	845
6.137GiNaC::archive_node::property_info Struct Reference . . . . .	845
6.137.1 Detailed Description . . . . .	845
6.137.2 Constructor & Destructor Documentation . . . . .	845
6.137.2.1 property_info() [1/2] . . . . .	846
6.137.2.2 property_info() [2/2] . . . . .	846
6.137.3 Member Data Documentation . . . . .	846
6.137.3.1 type . . . . .	846
6.137.3.2 name . . . . .	846
6.137.3.3 count . . . . .	846
6.138GiNaC::pseries Class Reference . . . . .	847
6.138.1 Detailed Description . . . . .	849

6.138.2 Constructor & Destructor Documentation . . . . .	849
6.138.2.1 pseries() [1/2] . . . . .	849
6.138.2.2 pseries() [2/2] . . . . .	850
6.138.3 Member Function Documentation . . . . .	850
6.138.3.1 precedence() . . . . .	850
6.138.3.2 nops() . . . . .	850
6.138.3.3 op() . . . . .	850
6.138.3.4 degree() . . . . .	851
6.138.3.5 ldegree() . . . . .	851
6.138.3.6 coeff() . . . . .	851
6.138.3.7 collect() . . . . .	852
6.138.3.8 eval() . . . . .	852
6.138.3.9 evalf() . . . . .	852
6.138.3.10series() . . . . .	852
6.138.3.11subs() . . . . .	853
6.138.3.12normal() . . . . .	853
6.138.3.13expand() . . . . .	853
6.138.3.14conjugate() . . . . .	854
6.138.3.15real_part() . . . . .	854
6.138.3.16imag_part() . . . . .	854
6.138.3.17eval_integ() . . . . .	854
6.138.3.18evalm() . . . . .	854
6.138.3.19archive() . . . . .	855
6.138.3.20read_archive() . . . . .	855
6.138.3.21derivative() . . . . .	855
6.138.3.22get_var() . . . . .	856
6.138.3.23get_point() . . . . .	856
6.138.3.24convert_to_poly() . . . . .	856
6.138.3.25s_compatible_to() . . . . .	856
6.138.3.26s_zero() . . . . .	857

6.138.3.27	<a href="#">s_terminating()</a>	857
6.138.3.28	<a href="#">coeffop()</a>	857
6.138.3.29	<a href="#">exponop()</a>	857
6.138.3.30	<a href="#">add_series()</a>	857
6.138.3.31	<a href="#">mul_const()</a>	858
6.138.3.32	<a href="#">mul_series()</a>	858
6.138.3.33	<a href="#">power_const()</a>	859
6.138.3.34	<a href="#">shift_exponents()</a>	859
6.138.3.35	<a href="#">print_series()</a>	859
6.138.3.36	<a href="#">do_print()</a>	860
6.138.3.37	<a href="#">do_print_latex()</a>	860
6.138.3.38	<a href="#">do_print_tree()</a>	860
6.138.3.39	<a href="#">do_print_python()</a>	860
6.138.3.40	<a href="#">do_print_python_repr()</a>	860
6.138.4	<a href="#">Member Data Documentation</a>	861
6.138.4.1	<a href="#">seq</a>	861
6.138.4.2	<a href="#">var</a>	861
6.138.4.3	<a href="#">point</a>	861
6.139	<a href="#">GiNaC::psi1_SERIAL Class Reference</a>	861
6.139.1	<a href="#">Detailed Description</a>	862
6.139.2	<a href="#">Member Data Documentation</a>	862
6.139.2.1	<a href="#">serial</a>	862
6.140	<a href="#">GiNaC::psi2_SERIAL Class Reference</a>	862
6.140.1	<a href="#">Detailed Description</a>	863
6.140.2	<a href="#">Member Data Documentation</a>	863
6.140.2.1	<a href="#">serial</a>	863
6.141	<a href="#">GiNaC::ptr&lt; T &gt; Class Template Reference</a>	863
6.141.1	<a href="#">Detailed Description</a>	864
6.141.2	<a href="#">Constructor &amp; Destructor Documentation</a>	864
6.141.2.1	<a href="#">ptr()</a> [1/3]	864

6.141.2.2 ptr() [2/3]	864
6.141.2.3 ptr() [3/3]	865
6.141.2.4 ~ptr()	865
6.141.3 Member Function Documentation	865
6.141.3.1 operator=()	865
6.141.3.2 operator*()	865
6.141.3.3 operator->()	865
6.141.3.4 makewritable()	865
6.141.3.5 swap()	866
6.141.3.6 operator==()	866
6.141.3.7 operator!=()	866
6.141.4 Friends And Related Function Documentation	866
6.141.4.1 std::less< ptr< T > >	866
6.141.4.2 get_pointer	866
6.141.4.3 operator== [1/2]	867
6.141.4.4 operator!= [1/2]	867
6.141.4.5 operator== [2/2]	867
6.141.4.6 operator!= [2/2]	867
6.141.4.7 operator<<	867
6.141.5 Member Data Documentation	867
6.141.5.1 p	868
6.142GiNaC::realsymbol Class Reference	868
6.142.1 Detailed Description	868
6.142.2 Constructor & Destructor Documentation	869
6.142.2.1 realsymbol() [1/3]	869
6.142.2.2 realsymbol() [2/3]	869
6.142.2.3 realsymbol() [3/3]	869
6.142.3 Member Function Documentation	869
6.142.3.1 get_domain()	869
6.142.3.2 conjugate()	869

6.142.3.3 <code>real_part()</code> . . . . .	870
6.142.3.4 <code>imag_part()</code> . . . . .	870
6.142.3.5 <code>duplicate()</code> . . . . .	870
6.143 <code>GiNaC::refcounted</code> Class Reference . . . . .	871
6.143.1 Detailed Description . . . . .	871
6.143.2 Constructor & Destructor Documentation . . . . .	872
6.143.2.1 <code>refcounted()</code> . . . . .	872
6.143.3 Member Function Documentation . . . . .	872
6.143.3.1 <code>add_reference()</code> . . . . .	872
6.143.3.2 <code>remove_reference()</code> . . . . .	872
6.143.3.3 <code>get_refcount()</code> . . . . .	872
6.143.3.4 <code>set_refcount()</code> . . . . .	872
6.143.4 Member Data Documentation . . . . .	873
6.143.4.1 <code>refcount</code> . . . . .	873
6.144 <code>GiNaC::registered_class_options</code> Class Reference . . . . .	873
6.144.1 Detailed Description . . . . .	874
6.144.2 Constructor & Destructor Documentation . . . . .	874
6.144.2.1 <code>registered_class_options()</code> . . . . .	874
6.144.3 Member Function Documentation . . . . .	874
6.144.3.1 <code>get_name()</code> . . . . .	874
6.144.3.2 <code>get_parent_name()</code> . . . . .	874
6.144.3.3 <code>get_id()</code> . . . . .	874
6.144.3.4 <code>get_print_dispatch_table()</code> . . . . .	875
6.144.3.5 <code>print_func()</code> [1/3] . . . . .	875
6.144.3.6 <code>print_func()</code> [2/3] . . . . .	875
6.144.3.7 <code>print_func()</code> [3/3] . . . . .	875
6.144.3.8 <code>set_print_func()</code> . . . . .	875
6.144.4 Member Data Documentation . . . . .	876
6.144.4.1 <code>name</code> . . . . .	876
6.144.4.2 <code>parent_name</code> . . . . .	876



6.144.4.3 tinfo_key . . . . .	876
6.144.4.4 print_dispatch_table . . . . .	876
6.145GiNaC::relational Class Reference . . . . .	877
6.145.1 Detailed Description . . . . .	878
6.145.2 Member Typedef Documentation . . . . .	878
6.145.2.1 safe_bool . . . . .	878
6.145.3 Member Enumeration Documentation . . . . .	878
6.145.3.1 operators . . . . .	878
6.145.4 Constructor & Destructor Documentation . . . . .	879
6.145.4.1 relational() . . . . .	879
6.145.5 Member Function Documentation . . . . .	879
6.145.5.1 precedence() . . . . .	879
6.145.5.2 info() . . . . .	879
6.145.5.3 nops() . . . . .	880
6.145.5.4 op() . . . . .	880
6.145.5.5 map() . . . . .	880
6.145.5.6 subs() . . . . .	880
6.145.5.7 archive() . . . . .	881
6.145.5.8 read_archive() . . . . .	881
6.145.5.9 eval_ncmul() . . . . .	881
6.145.5.10match_same_type() . . . . .	881
6.145.5.11return_type() . . . . .	882
6.145.5.12return_type_tinfo() . . . . .	882
6.145.5.13calchash() . . . . .	882
6.145.5.14do_print() . . . . .	882
6.145.5.15do_print_python_repr() . . . . .	883
6.145.5.16hs() . . . . .	883
6.145.5.17rhs() . . . . .	883
6.145.5.18make_safe_bool() . . . . .	883
6.145.5.19operator safe_bool() . . . . .	883

6.145.5.20operator"() . . . . .	884
6.145.6 Member Data Documentation . . . . .	884
6.145.6.1 lh . . . . .	884
6.145.6.2 rh . . . . .	884
6.145.6.3 o . . . . .	884
6.146GiNaC::remember_strategies Class Reference . . . . .	884
6.146.1 Detailed Description . . . . .	885
6.146.2 Member Enumeration Documentation . . . . .	885
6.146.2.1 anonymous enum . . . . .	885
6.147GiNaC::remember_table Class Reference . . . . .	885
6.147.1 Detailed Description . . . . .	886
6.147.2 Constructor & Destructor Documentation . . . . .	886
6.147.2.1 remember_table() [1/2] . . . . .	886
6.147.2.2 remember_table() [2/2] . . . . .	887
6.147.3 Member Function Documentation . . . . .	887
6.147.3.1 lookup_entry() . . . . .	887
6.147.3.2 add_entry() . . . . .	887
6.147.3.3 clear_all_entries() . . . . .	887
6.147.3.4 show_statistics() . . . . .	887
6.147.3.5 remember_tables() . . . . .	888
6.147.3.6 init_table() . . . . .	888
6.147.4 Member Data Documentation . . . . .	888
6.147.4.1 table_size . . . . .	888
6.147.4.2 max_assoc_size . . . . .	888
6.147.4.3 remember_strategy . . . . .	888
6.148GiNaC::remember_table_entry Class Reference . . . . .	889
6.148.1 Detailed Description . . . . .	889
6.148.2 Constructor & Destructor Documentation . . . . .	889
6.148.2.1 remember_table_entry() . . . . .	889
6.148.3 Member Function Documentation . . . . .	890

6.148.3.1 <code>is_equal()</code> . . . . .	890
6.148.3.2 <code>get_result()</code> . . . . .	890
6.148.3.3 <code>get_last_access()</code> . . . . .	890
6.148.3.4 <code>get_successful_hits()</code> . . . . .	890
6.148.4 Member Data Documentation . . . . .	890
6.148.4.1 <code>hashvalue</code> . . . . .	890
6.148.4.2 <code>seq</code> . . . . .	891
6.148.4.3 <code>result</code> . . . . .	891
6.148.4.4 <code>last_access</code> . . . . .	891
6.148.4.5 <code>successful_hits</code> . . . . .	891
6.148.4.6 <code>access_counter</code> . . . . .	891
6.149 <code>GiNaC::remember_table_list</code> Class Reference . . . . .	892
6.149.1 Detailed Description . . . . .	892
6.149.2 Constructor & Destructor Documentation . . . . .	892
6.149.2.1 <code>remember_table_list()</code> . . . . .	892
6.149.3 Member Function Documentation . . . . .	892
6.149.3.1 <code>add_entry()</code> . . . . .	893
6.149.3.2 <code>lookup_entry()</code> . . . . .	893
6.149.4 Member Data Documentation . . . . .	893
6.149.4.1 <code>max_assoc_size</code> . . . . .	893
6.149.4.2 <code>remember_strategy</code> . . . . .	893
6.150 <code>GiNaC::return_type_t</code> Struct Reference . . . . .	893
6.150.1 Detailed Description . . . . .	894
6.150.2 Member Function Documentation . . . . .	894
6.150.2.1 <code>operator&lt;()</code> . . . . .	894
6.150.2.2 <code>operator==()</code> . . . . .	894
6.150.2.3 <code>operator!=()</code> . . . . .	895
6.150.3 Member Data Documentation . . . . .	895
6.150.3.1 <code>tinfo</code> . . . . .	895
6.150.3.2 <code>rl</code> . . . . .	895

6.151GiNaC::return_types Class Reference . . . . .	895
6.151.1 Member Enumeration Documentation . . . . .	895
6.151.1.1 anonymous enum . . . . .	895
6.152GiNaC::relational::safe_bool_helper Struct Reference . . . . .	896
6.152.1 Member Function Documentation . . . . .	896
6.152.1.1 nonnull() . . . . .	896
6.153GiNaC::scalar_products Class Reference . . . . .	896
6.153.1 Detailed Description . . . . .	897
6.153.2 Member Function Documentation . . . . .	897
6.153.2.1 add() [1/2] . . . . .	897
6.153.2.2 add() [2/2] . . . . .	898
6.153.2.3 add_vectors() . . . . .	898
6.153.2.4 clear() . . . . .	898
6.153.2.5 is_defined() . . . . .	898
6.153.2.6 evaluate() . . . . .	899
6.153.2.7 debugprint() . . . . .	899
6.153.3 Member Data Documentation . . . . .	899
6.153.3.1 spm . . . . .	899
6.154GiNaC::series_options Class Reference . . . . .	899
6.154.1 Detailed Description . . . . .	899
6.154.2 Member Enumeration Documentation . . . . .	899
6.154.2.1 anonymous enum . . . . .	899
6.155GiNaC::solve_algo Class Reference . . . . .	900
6.155.1 Detailed Description . . . . .	900
6.155.2 Member Enumeration Documentation . . . . .	900
6.155.2.1 anonymous enum . . . . .	900
6.156GiNaC::spinidx Class Reference . . . . .	901
6.156.1 Detailed Description . . . . .	902
6.156.2 Constructor & Destructor Documentation . . . . .	902
6.156.2.1 spinidx() . . . . .	902

6.156.3 Member Function Documentation . . . . .	903
6.156.3.1 is_dummy_pair_same_type() . . . . .	903
6.156.3.2 conjugate() . . . . .	903
6.156.3.3 archive() . . . . .	903
6.156.3.4 read_archive() . . . . .	904
6.156.3.5 match_same_type() . . . . .	904
6.156.3.6 is_dotted() . . . . .	904
6.156.3.7 is_undotted() . . . . .	905
6.156.3.8 toggle_dot() . . . . .	905
6.156.3.9 toggle_variance_dot() . . . . .	905
6.156.3.10do_print() . . . . .	905
6.156.3.11do_print_latex() . . . . .	905
6.156.3.12do_print_tree() . . . . .	906
6.156.4 Member Data Documentation . . . . .	906
6.156.4.1 dotted . . . . .	906
6.157GiNaC::spinmetric Class Reference . . . . .	906
6.157.1 Detailed Description . . . . .	907
6.157.2 Member Function Documentation . . . . .	907
6.157.2.1 info() . . . . .	907
6.157.2.2 eval_indexed() . . . . .	908
6.157.2.3 contract_with() . . . . .	908
6.157.2.4 do_print() . . . . .	908
6.157.2.5 do_print_latex() . . . . .	908
6.158GiNaC::spmapkey Class Reference . . . . .	909
6.158.1 Constructor & Destructor Documentation . . . . .	909
6.158.1.1 spmapkey() [1/2] . . . . .	909
6.158.1.2 spmapkey() [2/2] . . . . .	909
6.158.2 Member Function Documentation . . . . .	909
6.158.2.1 operator==() . . . . .	910
6.158.2.2 operator<() . . . . .	910

6.158.2.3 <code>debugprint()</code> . . . . .	910
6.158.3 Member Data Documentation . . . . .	910
6.158.3.1 <code>v1</code> . . . . .	910
6.158.3.2 <code>v2</code> . . . . .	910
6.158.3.3 <code>dim</code> . . . . .	911
6.159 <code>GiNaC::status_flags</code> Class Reference . . . . .	911
6.159.1 Detailed Description . . . . .	911
6.159.2 Member Enumeration Documentation . . . . .	911
6.159.2.1 <code>anonymous enum</code> . . . . .	911
6.160 <code>GiNaC::structure&lt; T, ComparisonPolicy &gt;</code> Class Template Reference . . . . .	912
6.160.1 Detailed Description . . . . .	914
6.160.2 Constructor & Destructor Documentation . . . . .	914
6.160.2.1 <code>structure()</code> . . . . .	914
6.160.3 Member Function Documentation . . . . .	914
6.160.3.1 <code>get_class_name()</code> . . . . .	914
6.160.3.2 <code>eval()</code> . . . . .	915
6.160.3.3 <code>evalm()</code> . . . . .	915
6.160.3.4 <code>eval_ncmul()</code> . . . . .	915
6.160.3.5 <code>eval_indexed()</code> . . . . .	915
6.160.3.6 <code>print()</code> . . . . .	915
6.160.3.7 <code>precedence()</code> . . . . .	916
6.160.3.8 <code>info()</code> . . . . .	916
6.160.3.9 <code>nops()</code> . . . . .	916
6.160.3.10 <code>op()</code> . . . . .	917
6.160.3.11 <code>operator[]()</code> [ 1 / 4 ] . . . . .	917
6.160.3.12 <code>operator[]()</code> [ 2 / 4 ] . . . . .	917
6.160.3.13 <code>set_op()</code> . . . . .	917
6.160.3.14 <code>operator[]()</code> [ 3 / 4 ] . . . . .	917
6.160.3.15 <code>operator[]()</code> [ 4 / 4 ] . . . . .	918
6.160.3.16 <code>has()</code> . . . . .	918

6.160.3.17	<a href="#">match()</a>	918
6.160.3.18	<a href="#">match_same_type()</a>	919
6.160.3.19	<a href="#">subs()</a>	919
6.160.3.20	<a href="#">map()</a>	919
6.160.3.21	<a href="#">degree()</a>	920
6.160.3.22	<a href="#">degree()</a>	920
6.160.3.23	<a href="#">coeff()</a>	920
6.160.3.24	<a href="#">expand()</a>	920
6.160.3.25	<a href="#">collect()</a>	920
6.160.3.26	<a href="#">derivative()</a>	921
6.160.3.27	<a href="#">series()</a>	921
6.160.3.28	<a href="#">normal()</a>	922
6.160.3.29	<a href="#">to_rational()</a>	922
6.160.3.30	<a href="#">to_polynomial()</a>	922
6.160.3.31	<a href="#">integer_content()</a>	923
6.160.3.32	<a href="#">mod()</a>	923
6.160.3.33	<a href="#">max_coefficient()</a>	923
6.160.3.34	<a href="#">get_free_indices()</a>	924
6.160.3.35	<a href="#">add_indexed()</a>	924
6.160.3.36	<a href="#">scalar_mul_indexed()</a>	924
6.160.3.37	<a href="#">contract_with()</a>	925
6.160.3.38	<a href="#">return_type()</a>	926
6.160.3.39	<a href="#">return_type_tinfo()</a>	926
6.160.3.40	<a href="#">s_equal_same_type()</a>	926
6.160.3.41	<a href="#">calchash()</a>	926
6.160.3.42	<a href="#">operator-&gt;()</a>	927
6.160.3.43	<a href="#">get_struct()</a> [1/2]	927
6.160.3.44	<a href="#">get_struct()</a> [2/2]	927
6.160.4	Member Data Documentation	927
6.160.4.1	<a href="#">obj</a>	927

6.161 GiNaC::su3d Class Reference . . . . .	928
6.161.1 Detailed Description . . . . .	928
6.161.2 Member Function Documentation . . . . .	928
6.161.2.1 eval_indexed() . . . . .	929
6.161.2.2 contract_with() . . . . .	929
6.161.2.3 return_type() . . . . .	929
6.161.2.4 do_print() . . . . .	929
6.161.2.5 do_print_latex() . . . . .	930
6.162 GiNaC::su3f Class Reference . . . . .	930
6.162.1 Detailed Description . . . . .	930
6.162.2 Member Function Documentation . . . . .	931
6.162.2.1 eval_indexed() . . . . .	931
6.162.2.2 contract_with() . . . . .	931
6.162.2.3 return_type() . . . . .	931
6.162.2.4 do_print() . . . . .	931
6.162.2.5 do_print_latex() . . . . .	932
6.163 GiNaC::su3one Class Reference . . . . .	932
6.163.1 Detailed Description . . . . .	932
6.163.2 Member Function Documentation . . . . .	932
6.163.2.1 do_print() . . . . .	933
6.163.2.2 do_print_latex() . . . . .	933
6.164 GiNaC::su3t Class Reference . . . . .	933
6.164.1 Detailed Description . . . . .	934
6.164.2 Member Function Documentation . . . . .	934
6.164.2.1 contract_with() . . . . .	934
6.164.2.2 do_print() . . . . .	934
6.164.2.3 do_print_latex() . . . . .	934
6.165 GiNaC::subs_options Class Reference . . . . .	934
6.165.1 Detailed Description . . . . .	935
6.165.2 Member Enumeration Documentation . . . . .	935



6.165.2.1 anonymous enum . . . . .	935
6.166GiNaC::sy_is_less Class Reference . . . . .	935
6.166.1 Constructor & Destructor Documentation . . . . .	936
6.166.1.1 sy_is_less() . . . . .	936
6.166.2 Member Function Documentation . . . . .	936
6.166.2.1 operator>() . . . . .	936
6.166.3 Member Data Documentation . . . . .	936
6.166.3.1 v . . . . .	936
6.167GiNaC::sy_swap Class Reference . . . . .	936
6.167.1 Constructor & Destructor Documentation . . . . .	937
6.167.1.1 sy_swap() . . . . .	937
6.167.2 Member Function Documentation . . . . .	937
6.167.2.1 operator>() . . . . .	937
6.167.3 Member Data Documentation . . . . .	937
6.167.3.1 v . . . . .	937
6.167.3.2 swapped . . . . .	938
6.168GiNaC::sym_desc Struct Reference . . . . .	938
6.168.1 Detailed Description . . . . .	938
6.168.2 Constructor & Destructor Documentation . . . . .	939
6.168.2.1 sym_desc() . . . . .	939
6.168.3 Member Function Documentation . . . . .	939
6.168.3.1 operator<() . . . . .	939
6.168.4 Member Data Documentation . . . . .	939
6.168.4.1 sym . . . . .	939
6.168.4.2 deg_a . . . . .	939
6.168.4.3 deg_b . . . . .	940
6.168.4.4 ldeg_a . . . . .	940
6.168.4.5 ldeg_b . . . . .	940
6.168.4.6 max_deg . . . . .	940
6.168.4.7 max_lcnops . . . . .	940

6.169GiNaC::symbol Class Reference . . . . .	941
6.169.1 Detailed Description . . . . .	942
6.169.2 Constructor & Destructor Documentation . . . . .	942
6.169.2.1 symbol() [1/2] . . . . .	942
6.169.2.2 symbol() [2/2] . . . . .	943
6.169.3 Member Function Documentation . . . . .	943
6.169.3.1 info() . . . . .	943
6.169.3.2 eval() . . . . .	943
6.169.3.3 evalf() . . . . .	944
6.169.3.4 series() . . . . .	944
6.169.3.5 subs() . . . . .	944
6.169.3.6 normal() . . . . .	945
6.169.3.7 to_rational() . . . . .	945
6.169.3.8 to_polynomial() . . . . .	945
6.169.3.9 conjugate() . . . . .	945
6.169.3.10real_part() . . . . .	946
6.169.3.11imag_part() . . . . .	946
6.169.3.12s_polynomial() . . . . .	946
6.169.3.13archive() . . . . .	946
6.169.3.14read_archive() . . . . .	947
6.169.3.15derivative() . . . . .	947
6.169.3.16s_equal_same_type() . . . . .	947
6.169.3.17calchash() . . . . .	948
6.169.3.18get_domain() . . . . .	948
6.169.3.19set_name() . . . . .	948
6.169.3.20set_TeX_name() . . . . .	948
6.169.3.21get_name() . . . . .	949
6.169.3.22get_TeX_name() . . . . .	949
6.169.3.23do_print() . . . . .	949
6.169.3.24do_print_latex() . . . . .	949

6.169.3.25do_print_tree()	949
6.169.3.26do_print_python_repr()	950
6.169.4 Member Data Documentation	950
6.169.4.1 serial	950
6.169.4.2 name	950
6.169.4.3 TeX_name	950
6.169.4.4 next_serial	951
6.170GiNaC::symbolset Class Reference	951
6.170.1 Constructor & Destructor Documentation	951
6.170.1.1 symbolset()	951
6.170.2 Member Function Documentation	951
6.170.2.1 insert_symbols()	952
6.170.2.2 has()	952
6.170.3 Member Data Documentation	952
6.170.3.1 s	952
6.171GiNaC::symmetry Class Reference	952
6.171.1 Detailed Description	954
6.171.2 Member Enumeration Documentation	954
6.171.2.1 symmetry_type	954
6.171.3 Constructor & Destructor Documentation	954
6.171.3.1 symmetry() [1/2]	954
6.171.3.2 symmetry() [2/2]	955
6.171.4 Member Function Documentation	955
6.171.4.1 archive()	955
6.171.4.2 read_archive()	955
6.171.4.3 calchash()	956
6.171.4.4 get_type()	956
6.171.4.5 set_type()	956
6.171.4.6 add()	956
6.171.4.7 validate()	957

6.171.4.8 has_symmetry()	957
6.171.4.9 has_nonsymmetric()	957
6.171.4.10 has_cyclic()	957
6.171.4.11 do_print()	957
6.171.4.12 do_print_tree()	958
6.171.5 Friends And Related Function Documentation	958
6.171.5.1 sy_is_less	958
6.171.5.2 sy_swap	958
6.171.5.3 canonicalize	958
6.171.6 Member Data Documentation	958
6.171.6.1 type	959
6.171.6.2 indices	959
6.171.6.3 children	959
6.172 GiNaC::symminfo Class Reference	959
6.172.1 Detailed Description	960
6.172.2 Constructor & Destructor Documentation	960
6.172.2.1 symminfo() [1/2]	960
6.172.2.2 symminfo() [2/2]	960
6.172.3 Member Data Documentation	960
6.172.3.1 symmterm	960
6.172.3.2 coeff	961
6.172.3.3 orig	961
6.172.3.4 num	961
6.173 GiNaC::symminfo_is_less_by_orig Class Reference	961
6.173.1 Member Function Documentation	961
6.173.1.1 operator()()	961
6.174 GiNaC::symminfo_is_less_by_symmterm Class Reference	962
6.174.1 Member Function Documentation	962
6.174.1.1 operator()()	962
6.175 GiNaC::tensdelta Class Reference	962

6.175.1 Detailed Description	963
6.175.2 Member Function Documentation	963
6.175.2.1 info()	963
6.175.2.2 eval_indexed()	963
6.175.2.3 contract_with()	964
6.175.2.4 return_type()	964
6.175.2.5 do_print()	964
6.175.2.6 do_print_latex()	964
6.176GiNaC::tensepsilon Class Reference	965
6.176.1 Detailed Description	966
6.176.2 Constructor & Destructor Documentation	966
6.176.2.1 tensepsilon()	966
6.176.3 Member Function Documentation	966
6.176.3.1 info()	966
6.176.3.2 eval_indexed()	966
6.176.3.3 contract_with()	967
6.176.3.4 archive()	967
6.176.3.5 read_archive()	967
6.176.3.6 return_type()	967
6.176.3.7 do_print()	968
6.176.3.8 do_print_latex()	968
6.176.4 Member Data Documentation	968
6.176.4.1 minkowski	968
6.176.4.2 pos_sig	968
6.177GiNaC::tensmetric Class Reference	969
6.177.1 Detailed Description	969
6.177.2 Member Function Documentation	969
6.177.2.1 info()	970
6.177.2.2 eval_indexed()	970
6.177.2.3 contract_with()	970

6.177.2.4 return_type() . . . . .	971
6.177.2.5 do_print() . . . . .	971
6.178GiNaC::tensor Class Reference . . . . .	971
6.178.1 Detailed Description . . . . .	972
6.178.2 Member Function Documentation . . . . .	972
6.178.2.1 return_type() . . . . .	972
6.178.2.2 replace_contr_index() . . . . .	972
6.179GiNaC::terminfo Class Reference . . . . .	973
6.179.1 Detailed Description . . . . .	973
6.179.2 Constructor & Destructor Documentation . . . . .	973
6.179.2.1 terminfo() . . . . .	973
6.179.3 Member Data Documentation . . . . .	973
6.179.3.1 orig . . . . .	973
6.179.3.2 symm . . . . .	974
6.180GiNaC::terminfo_is_less Class Reference . . . . .	974
6.180.1 Member Function Documentation . . . . .	974
6.180.1.1 operator>() . . . . .	974
6.181GiNaC::class_info< OPT >::tree_node Struct Reference . . . . .	974
6.181.1 Constructor & Destructor Documentation . . . . .	975
6.181.1.1 tree_node() . . . . .	975
6.181.2 Member Function Documentation . . . . .	975
6.181.2.1 add_child() . . . . .	975
6.181.3 Member Data Documentation . . . . .	975
6.181.3.1 children . . . . .	975
6.181.3.2 info . . . . .	975
6.182GiNaC::unarchive_table_t Class Reference . . . . .	976
6.182.1 Constructor & Destructor Documentation . . . . .	976
6.182.1.1 unarchive_table_t() . . . . .	976
6.182.1.2 ~unarchive_table_t() . . . . .	976
6.182.2 Member Function Documentation . . . . .	976

6.182.2.1 find()	976
6.182.2.2 insert()	977
6.182.3 Member Data Documentation	977
6.182.3.1 usecount	977
6.182.3.2 unarch_map	977
6.183GiNaC::user_defined_kernel Class Reference	977
6.183.1 Detailed Description	978
6.183.2 Constructor & Destructor Documentation	978
6.183.2.1 user_defined_kernel()	978
6.183.3 Member Function Documentation	978
6.183.3.1 nops()	979
6.183.3.2 op()	979
6.183.3.3 let_op()	979
6.183.3.4 is_numeric()	979
6.183.3.5 Laurent_series()	980
6.183.3.6 uses_Laurent_series()	980
6.183.3.7 do_print()	980
6.183.4 Member Data Documentation	980
6.183.4.1 f	980
6.183.4.2 x	981
6.184GiNaC::varidx Class Reference	981
6.184.1 Detailed Description	982
6.184.2 Constructor & Destructor Documentation	982
6.184.2.1 varidx()	982
6.184.3 Member Function Documentation	982
6.184.3.1 is_dummy_pair_same_type()	983
6.184.3.2 archive()	983
6.184.3.3 read_archive()	983
6.184.3.4 match_same_type()	984
6.184.3.5 is_covariant()	984

6.184.3.6 <code>is_contravariant()</code> . . . . .	984
6.184.3.7 <code>toggle_variance()</code> . . . . .	984
6.184.3.8 <code>do_print()</code> . . . . .	985
6.184.3.9 <code>do_print_tree()</code> . . . . .	985
6.184.4 Member Data Documentation . . . . .	985
6.184.4.1 <code>covariant</code> . . . . .	985
6.185GiNaC::visitor Class Reference . . . . .	985
6.185.1 Detailed Description . . . . .	986
6.185.2 Constructor & Destructor Documentation . . . . .	986
6.185.2.1 <code>~visitor()</code> . . . . .	986
6.186GiNaC::wildcard Class Reference . . . . .	986
6.186.1 Detailed Description . . . . .	987
6.186.2 Constructor & Destructor Documentation . . . . .	987
6.186.2.1 <code>wildcard()</code> . . . . .	987
6.186.3 Member Function Documentation . . . . .	987
6.186.3.1 <code>match()</code> . . . . .	987
6.186.3.2 <code>archive()</code> . . . . .	988
6.186.3.3 <code>read_archive()</code> . . . . .	988
6.186.3.4 <code>calchash()</code> . . . . .	988
6.186.3.5 <code>get_label()</code> . . . . .	988
6.186.3.6 <code>do_print()</code> . . . . .	989
6.186.3.7 <code>do_print_tree()</code> . . . . .	989
6.186.3.8 <code>do_print_python_repr()</code> . . . . .	989
6.186.4 Member Data Documentation . . . . .	989
6.186.4.1 <code>label</code> . . . . .	989
6.187GiNaC::zeta1_SERIAL Class Reference . . . . .	990
6.187.1 Detailed Description . . . . .	990
6.187.2 Member Data Documentation . . . . .	990
6.187.2.1 <code>serial</code> . . . . .	990
6.188GiNaC::zeta2_SERIAL Class Reference . . . . .	990
6.188.1 Detailed Description . . . . .	991
6.188.2 Member Data Documentation . . . . .	991
6.188.2.1 <code>serial</code> . . . . .	991



<b>7 File Documentation</b>	<b>993</b>
7.1 add.cpp File Reference	993
7.1.1 Detailed Description	993
7.2 add.h File Reference	994
7.2.1 Detailed Description	994
7.3 archive.cpp File Reference	994
7.3.1 Detailed Description	995
7.4 archive.h File Reference	995
7.4.1 Detailed Description	996
7.4.2 Macro Definition Documentation	996
7.4.2.1 GINAC_DECLARE_UNARCHIVER	997
7.4.2.2 GINAC_BIND_UNARCHIVER	997
7.5 assertion.h File Reference	998
7.5.1 Detailed Description	998
7.5.2 Macro Definition Documentation	998
7.5.2.1 GINAC_ASSERT	998
7.6 basic.cpp File Reference	999
7.6.1 Detailed Description	1000
7.7 basic.h File Reference	1000
7.7.1 Detailed Description	1001
7.8 class_info.h File Reference	1001
7.8.1 Detailed Description	1002
7.9 clifford.cpp File Reference	1002
7.9.1 Detailed Description	1004
7.10 clifford.h File Reference	1004
7.10.1 Detailed Description	1006
7.11 color.cpp File Reference	1006
7.11.1 Detailed Description	1007
7.11.2 Macro Definition Documentation	1007
7.11.2.1 TEST_PERMUTATION	1008

7.11.2.2	CMPINDICES	1008
7.12	color.h File Reference	1008
7.12.1	Detailed Description	1009
7.13	compiler.h File Reference	1009
7.13.1	Detailed Description	1009
7.13.2	Macro Definition Documentation	1010
7.13.2.1	unlikely	1010
7.13.2.2	likely	1010
7.13.2.3	attribute_deprecated	1010
7.14	constant.cpp File Reference	1010
7.14.1	Detailed Description	1011
7.15	constant.h File Reference	1011
7.15.1	Detailed Description	1012
7.16	container.h File Reference	1012
7.16.1	Detailed Description	1012
7.17	crc32.h File Reference	1012
7.17.1	Detailed Description	1013
7.18	ex.cpp File Reference	1013
7.18.1	Detailed Description	1013
7.19	ex.h File Reference	1013
7.19.1	Detailed Description	1015
7.20	excompiler.cpp File Reference	1016
7.20.1	Detailed Description	1016
7.21	excompiler.h File Reference	1016
7.21.1	Detailed Description	1017
7.22	expair.cpp File Reference	1017
7.22.1	Detailed Description	1018
7.23	expair.h File Reference	1018
7.23.1	Detailed Description	1018
7.24	expairseq.cpp File Reference	1019

7.24.1 Detailed Description . . . . .	1019
7.25 expairseq.h File Reference . . . . .	1019
7.25.1 Detailed Description . . . . .	1020
7.26 exprseq.cpp File Reference . . . . .	1020
7.26.1 Detailed Description . . . . .	1020
7.27 exprseq.h File Reference . . . . .	1020
7.27.1 Detailed Description . . . . .	1021
7.28 factor.cpp File Reference . . . . .	1021
7.28.1 Detailed Description . . . . .	1022
7.28.2 Macro Definition Documentation . . . . .	1022
7.28.2.1 DCOUT . . . . .	1022
7.28.2.2 DCOUTVAR . . . . .	1022
7.28.2.3 DCOUT2 . . . . .	1022
7.28.2.4 USE_SAME_DEGREE_FACTOR . . . . .	1023
7.28.3 Variable Documentation . . . . .	1023
7.28.3.1 value . . . . .	1023
7.28.3.2 r . . . . .	1023
7.28.3.3 c . . . . .	1024
7.28.3.4 m . . . . .	1025
7.28.3.5 lr . . . . .	1025
7.28.3.6 cache . . . . .	1025
7.28.3.7 factors . . . . .	1025
7.28.3.8 one . . . . .	1025
7.28.3.9 n . . . . .	1026
7.28.3.10 len . . . . .	1026
7.28.3.11 last . . . . .	1026
7.28.3.12 k . . . . .	1027
7.28.3.13 poly . . . . .	1027
7.28.3.14 x . . . . .	1027
7.28.3.15 evalpoint . . . . .	1028

7.28.3.16 R	1028
7.28.3.17 syms	1028
7.28.3.18 options	1029
7.29 factor.h File Reference	1029
7.29.1 Detailed Description	1029
7.30 fail.cpp File Reference	1029
7.30.1 Detailed Description	1030
7.31 fail.h File Reference	1030
7.31.1 Detailed Description	1030
7.32 fderivative.cpp File Reference	1031
7.32.1 Detailed Description	1031
7.33 fderivative.h File Reference	1031
7.33.1 Detailed Description	1032
7.34 flags.h File Reference	1032
7.34.1 Detailed Description	1032
7.35 function.cpp File Reference	1033
7.35.1 Detailed Description	1033
7.36 function.h File Reference	1033
7.36.1 Detailed Description	1040
7.36.2 Macro Definition Documentation	1040
7.36.2.1 DECLARE_FUNCTION_1P	1040
7.36.2.2 DECLARE_FUNCTION_2P	1040
7.36.2.3 DECLARE_FUNCTION_3P	1041
7.36.2.4 DECLARE_FUNCTION_4P	1041
7.36.2.5 DECLARE_FUNCTION_5P	1041
7.36.2.6 DECLARE_FUNCTION_6P	1042
7.36.2.7 DECLARE_FUNCTION_7P	1042
7.36.2.8 DECLARE_FUNCTION_8P	1042
7.36.2.9 DECLARE_FUNCTION_9P	1043
7.36.2.10 DECLARE_FUNCTION_10P	1043

7.36.2.11 DECLARE_FUNCTION_11P . . . . .	1043
7.36.2.12 DECLARE_FUNCTION_12P . . . . .	1044
7.36.2.13 DECLARE_FUNCTION_13P . . . . .	1044
7.36.2.14 DECLARE_FUNCTION_14P . . . . .	1044
7.36.2.15 REGISTER_FUNCTION . . . . .	1045
7.36.2.16 is_ex_the_function . . . . .	1045
7.37 ginac.h File Reference . . . . .	1045
7.37.1 Detailed Description . . . . .	1046
7.38 hash_map.h File Reference . . . . .	1046
7.38.1 Detailed Description . . . . .	1046
7.39 hash_seed.h File Reference . . . . .	1046
7.39.1 Detailed Description . . . . .	1047
7.40 idx.cpp File Reference . . . . .	1047
7.40.1 Detailed Description . . . . .	1048
7.41 idx.h File Reference . . . . .	1048
7.41.1 Detailed Description . . . . .	1049
7.42 indexed.cpp File Reference . . . . .	1049
7.42.1 Detailed Description . . . . .	1050
7.43 indexed.h File Reference . . . . .	1051
7.43.1 Detailed Description . . . . .	1051
7.44 inifcns.cpp File Reference . . . . .	1052
7.44.1 Detailed Description . . . . .	1055
7.45 inifcns.h File Reference . . . . .	1055
7.45.1 Detailed Description . . . . .	1056
7.46 inifcns_elliptic.cpp File Reference . . . . .	1056
7.46.1 Detailed Description . . . . .	1057
7.47 inifcns_gamma.cpp File Reference . . . . .	1058
7.47.1 Detailed Description . . . . .	1059
7.48 inifcns_nstdsums.cpp File Reference . . . . .	1059
7.48.1 Detailed Description . . . . .	1060

7.49	<a href="#">inifcns_trans.cpp File Reference</a>	1061
7.49.1	<a href="#">Detailed Description</a>	1064
7.50	<a href="#">integral.cpp File Reference</a>	1064
7.50.1	<a href="#">Detailed Description</a>	1064
7.51	<a href="#">integral.h File Reference</a>	1065
7.51.1	<a href="#">Detailed Description</a>	1065
7.52	<a href="#">integration_kernel.cpp File Reference</a>	1065
7.52.1	<a href="#">Detailed Description</a>	1067
7.52.2	<a href="#">Variable Documentation</a>	1067
7.52.2.1	<a href="#">qbar</a>	1067
7.52.2.2	<a href="#">order</a>	1067
7.52.2.3	<a href="#">cache_vec</a>	1067
7.52.2.4	<a href="#">x</a>	1067
7.53	<a href="#">integration_kernel.h File Reference</a>	1068
7.53.1	<a href="#">Detailed Description</a>	1069
7.54	<a href="#">lst.cpp File Reference</a>	1069
7.54.1	<a href="#">Detailed Description</a>	1069
7.55	<a href="#">lst.h File Reference</a>	1069
7.55.1	<a href="#">Detailed Description</a>	1070
7.56	<a href="#">matrix.cpp File Reference</a>	1070
7.56.1	<a href="#">Detailed Description</a>	1071
7.57	<a href="#">matrix.h File Reference</a>	1071
7.57.1	<a href="#">Detailed Description</a>	1072
7.58	<a href="#">mul.cpp File Reference</a>	1072
7.58.1	<a href="#">Detailed Description</a>	1073
7.59	<a href="#">mul.h File Reference</a>	1073
7.59.1	<a href="#">Detailed Description</a>	1074
7.60	<a href="#">ncmul.cpp File Reference</a>	1074
7.60.1	<a href="#">Detailed Description</a>	1074
7.61	<a href="#">ncmul.h File Reference</a>	1075

7.61.1 Detailed Description . . . . .	1075
7.62 normal.cpp File Reference . . . . .	1075
7.62.1 Detailed Description . . . . .	1077
7.62.2 Macro Definition Documentation . . . . .	1077
7.62.2.1 FAST_COMPARE . . . . .	1078
7.62.2.2 USE_REMEMBER . . . . .	1078
7.62.2.3 USE_TRIAL_DIVISION . . . . .	1078
7.62.2.4 STATISTICS . . . . .	1078
7.63 normal.h File Reference . . . . .	1078
7.63.1 Detailed Description . . . . .	1079
7.64 numeric.cpp File Reference . . . . .	1079
7.64.1 Detailed Description . . . . .	1082
7.65 numeric.h File Reference . . . . .	1083
7.65.1 Detailed Description . . . . .	1085
7.66 operators.cpp File Reference . . . . .	1085
7.66.1 Detailed Description . . . . .	1087
7.67 operators.h File Reference . . . . .	1087
7.67.1 Detailed Description . . . . .	1089
7.68 power.cpp File Reference . . . . .	1089
7.68.1 Detailed Description . . . . .	1089
7.69 power.h File Reference . . . . .	1090
7.69.1 Detailed Description . . . . .	1090
7.70 print.cpp File Reference . . . . .	1090
7.70.1 Detailed Description . . . . .	1091
7.71 print.h File Reference . . . . .	1091
7.71.1 Detailed Description . . . . .	1092
7.71.2 Macro Definition Documentation . . . . .	1092
7.71.2.1 GINAC_DECLARE_PRINT_CONTEXT_COMMON . . . . .	1092
7.71.2.2 GINAC_DECLARE_PRINT_CONTEXT_BASE . . . . .	1093
7.71.2.3 GINAC_DECLARE_PRINT_CONTEXT . . . . .	1093

7.71.2.4	GINAC_IMPLEMENT_PRINT_CONTEXT . . . . .	1093
7.72	pseries.cpp File Reference . . . . .	1094
7.72.1	Detailed Description . . . . .	1094
7.73	pseries.h File Reference . . . . .	1094
7.73.1	Detailed Description . . . . .	1095
7.74	ptr.h File Reference . . . . .	1095
7.74.1	Detailed Description . . . . .	1095
7.75	registrar.cpp File Reference . . . . .	1096
7.75.1	Detailed Description . . . . .	1096
7.76	registrar.h File Reference . . . . .	1096
7.76.1	Detailed Description . . . . .	1097
7.76.2	Macro Definition Documentation . . . . .	1097
7.76.2.1	GINAC_DECLARE_REGISTERED_CLASS_COMMON . . . . .	1098
7.76.2.2	GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS . . . . .	1098
7.76.2.3	GINAC_DECLARE_REGISTERED_CLASS . . . . .	1098
7.76.2.4	GINAC_IMPLEMENT_REGISTERED_CLASS . . . . .	1099
7.76.2.5	GINAC_IMPLEMENT_REGISTERED_CLASS_OPT . . . . .	1099
7.76.2.6	GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T . . . . .	1099
7.77	relational.cpp File Reference . . . . .	1100
7.77.1	Detailed Description . . . . .	1100
7.78	relational.h File Reference . . . . .	1100
7.78.1	Detailed Description . . . . .	1101
7.79	remember.cpp File Reference . . . . .	1101
7.79.1	Detailed Description . . . . .	1101
7.80	remember.h File Reference . . . . .	1101
7.80.1	Detailed Description . . . . .	1102
7.81	structure.h File Reference . . . . .	1102
7.81.1	Detailed Description . . . . .	1102
7.82	symbol.cpp File Reference . . . . .	1103
7.82.1	Detailed Description . . . . .	1103



7.83	symbol.h File Reference	1103
7.83.1	Detailed Description	1104
7.84	symmetry.cpp File Reference	1104
7.84.1	Detailed Description	1105
7.85	symmetry.h File Reference	1105
7.85.1	Detailed Description	1106
7.86	tensor.cpp File Reference	1107
7.86.1	Detailed Description	1108
7.87	tensor.h File Reference	1108
7.87.1	Detailed Description	1109
7.88	utils.cpp File Reference	1109
7.88.1	Detailed Description	1111
7.89	utils.h File Reference	1111
7.89.1	Detailed Description	1113
7.89.2	Macro Definition Documentation	1113
7.89.2.1	DEFAULT_CTOR	1113
7.89.2.2	DEFAULT_COMPARE	1113
7.89.2.3	DEFAULT_PRINT	1113
7.89.2.4	DEFAULT_PRINT_LATEX	1114
7.90	utils_multi_iterator.h File Reference	1114
7.90.1	Detailed Description	1116
7.91	version.h File Reference	1116
7.91.1	Detailed Description	1116
7.91.2	Macro Definition Documentation	1116
7.91.2.1	GINACLIB_MAJOR_VERSION	1117
7.91.2.2	GINACLIB_MINOR_VERSION	1117
7.91.2.3	GINACLIB_MICRO_VERSION	1117
7.91.2.4	GINAC_LT_CURRENT	1117
7.91.2.5	GINAC_LT_REVISION	1117
7.91.2.6	GINAC_LT_AGE	1117
7.91.2.7	GINACLIB_ARCHIVE_VERSION	1117
7.91.2.8	GINACLIB_ARCHIVE_AGE	1118
7.91.2.9	GINACLIB_STR_HELPER	1118
7.91.2.10	GINACLIB_STR	1118
7.91.2.11	GINACLIB_VERSION	1118
7.92	wildcard.cpp File Reference	1118
7.92.1	Detailed Description	1119
7.93	wildcard.h File Reference	1119
7.93.1	Detailed Description	1119



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">GiNaC</a> . . . . .	19
<a href="#">GiNaC::internal</a> . . . . .	308
<a href="#">std</a> . . . . .	308



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

GiNaC::internal::_iter_rep . . . . .	311
GiNaC::_numeric_digits . . . . .	313
GiNaC::archive . . . . .	328
GiNaC::archive_node . . . . .	335
GiNaC::archive_node::archive_node_cit_range . . . . .	345
GiNaC::archive::archived_ex . . . . .	346
GiNaC::basic_multi_iterator< T > . . . . .	376
GiNaC::multi_iterator_counter< T > . . . . .	721
GiNaC::multi_iterator_counter_indv< T > . . . . .	723
GiNaC::multi_iterator_ordered< T > . . . . .	726
GiNaC::multi_iterator_ordered_eq< T > . . . . .	729
GiNaC::multi_iterator_ordered_eq_indv< T > . . . . .	732
GiNaC::multi_iterator_permutation< T > . . . . .	735
GiNaC::multi_iterator_shuffle< T > . . . . .	738
GiNaC::multi_iterator_shuffle_prime< T > . . . . .	742
GiNaC::basic_partition_generator . . . . .	381
GiNaC::partition_generator . . . . .	786
GiNaC::partition_with_zero_parts_generator . . . . .	788
GiNaC::class_info< OPT > . . . . .	382
GiNaC::compare_all_equal< T > . . . . .	401
GiNaC::compare_bitwise< T > . . . . .	403
GiNaC::compare_std_less< T > . . . . .	404
ComparisonPolicy	
GiNaC::structure< T, ComparisonPolicy > . . . . .	912
GiNaC::composition_generator . . . . .	405
GiNaC::container_storage< C > . . . . .	440
GiNaC::container< C > . . . . .	426
GiNaC::function . . . . .	546
GiNaC::fderivative . . . . .	539
GiNaC::indexed . . . . .	624
GiNaC::clifford . . . . .	386
GiNaC::color . . . . .	396
GiNaC::ncmul . . . . .	747
GiNaC::composition_generator::coolmulti . . . . .	443

GiNaC::determinant_algo . . . . .	447
GiNaC::do_taylor . . . . .	454
GiNaC::domain . . . . .	455
domain_error	
GiNaC::pole_error . . . . .	805
GiNaC::dunno . . . . .	455
GiNaC::composition_generator::coolmulti::element . . . . .	471
std::equal_to< GiNaC::ex > . . . . .	476
GiNaC::error_and_integral . . . . .	476
GiNaC::error_and_integral_is_less . . . . .	477
GiNaC::ex . . . . .	480
GiNaC::ex_base_is_less . . . . .	515
GiNaC::ex_is_equal . . . . .	515
GiNaC::ex_is_less . . . . .	516
GiNaC::ex_swap . . . . .	516
GiNaC::expair . . . . .	517
GiNaC::expair_is_less . . . . .	520
GiNaC::expair_rest_is_less . . . . .	521
GiNaC::expair_swap . . . . .	521
GiNaC::expand_options . . . . .	537
GiNaC::factor_options . . . . .	537
GiNaC::function_options . . . . .	562
GiNaC::G2_SERIAL . . . . .	608
GiNaC::G3_SERIAL . . . . .	609
GiNaC::gcd_options . . . . .	610
GiNaC::gcdheu_failed . . . . .	611
GiNaC::has_distance< T > . . . . .	611
GiNaC::has_options . . . . .	613
std::hash< GiNaC::ex > . . . . .	613
GiNaC::idx_is_equal_ignore_dim . . . . .	624
GiNaC::info_flags . . . . .	639
GiNaC::is_not_a_clifford . . . . .	655
GiNaC::is_summation_idx . . . . .	655
GiNaC::iterated_integral2_SERIAL . . . . .	656
GiNaC::iterated_integral3_SERIAL . . . . .	657
iterator	
GiNaC::const_iterator . . . . .	408
GiNaC::const_postorder_iterator . . . . .	413
GiNaC::const_preorder_iterator . . . . .	416
GiNaC::lanczos_coeffs . . . . .	665
std::less< GiNaC::ptr< T > > . . . . .	667
GiNaC::library_init . . . . .	668
list	
GiNaC::remember_table_list . . . . .	892
GiNaC::make_flat_inserter . . . . .	670
GiNaC::map_function . . . . .	672
GiNaC::derivative_map_function . . . . .	445
GiNaC::eval_integ_map_function . . . . .	478
GiNaC::evalf_map_function . . . . .	479
GiNaC::evalm_map_function . . . . .	479
GiNaC::expand_map_function . . . . .	535
GiNaC::normal_map_function . . . . .	756
GiNaC::pointer_to_map_function . . . . .	790
GiNaC::pointer_to_map_function_1arg< T1 > . . . . .	791
GiNaC::pointer_to_map_function_2args< T1, T2 > . . . . .	793
GiNaC::pointer_to_map_function_3args< T1, T2, T3 > . . . . .	795
GiNaC::pointer_to_member_to_map_function< C > . . . . .	797
GiNaC::pointer_to_member_to_map_function_1arg< C, T1 > . . . . .	798

GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 > . . . . .	800
GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 > . . . . .	802
GiNaC::basic_partition_generator::mpartition2 . . . . .	703
GiNaC::op0_is_equal . . . . .	786
GiNaC::print_context . . . . .	822
GiNaC::print_csrc . . . . .	826
GiNaC::print_csrc_cl_N . . . . .	827
GiNaC::print_csrc_double . . . . .	828
GiNaC::print_csrc_float . . . . .	829
GiNaC::print_dflt . . . . .	830
GiNaC::print_latex . . . . .	834
GiNaC::print_python . . . . .	840
GiNaC::print_python_repr . . . . .	841
GiNaC::print_tree . . . . .	842
GiNaC::print_context_options . . . . .	824
GiNaC::print_functor . . . . .	831
GiNaC::print_functor_impl . . . . .	833
GiNaC::print_memfun_handler< T, C > . . . . .	835
GiNaC::print_ptrfun_handler< T, C > . . . . .	838
GiNaC::print_options . . . . .	837
GiNaC::archive_node::property . . . . .	843
GiNaC::archive_node::property_info . . . . .	845
GiNaC::psi1_SERIAL . . . . .	861
GiNaC::psi2_SERIAL . . . . .	862
GiNaC::ptr< T > . . . . .	863
GiNaC::ptr< GiNaC::basic > . . . . .	863
GiNaC::refcounted . . . . .	871
GiNaC::basic . . . . .	348
GiNaC::constant . . . . .	419
GiNaC::container< C > . . . . .	426
GiNaC::expairseq . . . . .	522
GiNaC::add . . . . .	315
GiNaC::mul . . . . .	705
GiNaC::fail . . . . .	538
GiNaC::idx . . . . .	614
GiNaC::varidx . . . . .	981
GiNaC::spinidx . . . . .	901
GiNaC::integral . . . . .	641
GiNaC::integration_kernel . . . . .	649
GiNaC::basic_log_kernel . . . . .	374
GiNaC::Ebar_kernel . . . . .	456
GiNaC::Eisenstein_h_kernel . . . . .	460
GiNaC::Eisenstein_kernel . . . . .	465
GiNaC::ELi_kernel . . . . .	472
GiNaC::Kronecker_dtau_kernel . . . . .	657
GiNaC::Kronecker_dz_kernel . . . . .	661
GiNaC::modular_form_kernel . . . . .	698
GiNaC::multiple_polylog_kernel . . . . .	744
GiNaC::user_defined_kernel . . . . .	977
GiNaC::matrix . . . . .	674
GiNaC::numeric . . . . .	757
GiNaC::power . . . . .	808
GiNaC::pseries . . . . .	847
GiNaC::relational . . . . .	877
GiNaC::structure< T, ComparisonPolicy > . . . . .	912
GiNaC::symbol . . . . .	941
GiNaC::realsymbol . . . . .	868

GiNaC::possymbol . . . . .	806
GiNaC::symmetry . . . . .	952
GiNaC::tensor . . . . .	971
GiNaC::cliffordunit . . . . .	395
GiNaC::diracgamma . . . . .	448
GiNaC::diracgamma5 . . . . .	449
GiNaC::diracgammaL . . . . .	451
GiNaC::diracgammaR . . . . .	452
GiNaC::diracone . . . . .	453
GiNaC::su3d . . . . .	928
GiNaC::su3f . . . . .	930
GiNaC::su3one . . . . .	932
GiNaC::su3t . . . . .	933
GiNaC::tensdelta . . . . .	962
GiNaC::tensepsilon . . . . .	965
GiNaC::tensmetric . . . . .	969
GiNaC::minkmetric . . . . .	695
GiNaC::spinmetric . . . . .	906
GiNaC::wildcard . . . . .	986
GiNaC::registered_class_options . . . . .	873
GiNaC::remember_strategies . . . . .	884
GiNaC::remember_table_entry . . . . .	889
GiNaC::return_type_t . . . . .	893
GiNaC::return_types . . . . .	895
GiNaC::relational::safe_bool_helper . . . . .	896
GiNaC::scalar_products . . . . .	896
GiNaC::series_options . . . . .	899
GiNaC::solve_algo . . . . .	900
GiNaC::spmapkey . . . . .	909
GiNaC::status_flags . . . . .	911
GiNaC::subs_options . . . . .	934
GiNaC::sy_is_less . . . . .	935
GiNaC::sy_swap . . . . .	936
GiNaC::sym_desc . . . . .	938
GiNaC::symbolset . . . . .	951
GiNaC::symminfo . . . . .	959
GiNaC::symminfo_is_less_by_orig . . . . .	961
GiNaC::symminfo_is_less_by_symmterm . . . . .	962
GiNaC::terminfo . . . . .	973
GiNaC::terminfo_is_less . . . . .	974
GiNaC::class_info< OPT >::tree_node . . . . .	974
GiNaC::unarchive_table_t . . . . .	976
vector	
GiNaC::remember_table . . . . .	885
GiNaC::visitor . . . . .	985
GiNaC::zeta1_SERIAL . . . . .	990
GiNaC::zeta2_SERIAL . . . . .	990



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">GiNaC::internal::_iter_rep</a>	311
<a href="#">GiNaC::_numeric_digits</a>	
This class is used to instantiate a global singleton object Digits which behaves just like Maple's	
Digits	313
<a href="#">GiNaC::add</a>	
Sum of expressions	315
<a href="#">GiNaC::archive</a>	
This class holds archived versions of <a href="#">GiNaC</a> expressions (class <a href="#">ex</a> )	328
<a href="#">GiNaC::archive_node</a>	
This class stores all properties needed to record/retrieve the state of one object of class <a href="#">basic</a>	
(or a derived class)	335
<a href="#">GiNaC::archive_node::archive_node_cit_range</a>	345
<a href="#">GiNaC::archive::archived_ex</a>	
Archived expression descriptor	346
<a href="#">GiNaC::basic</a>	
This class is the ABC (abstract base class) of <a href="#">GiNaC</a> 's class hierarchy	348
<a href="#">GiNaC::basic_log_kernel</a>	
The basic integration kernel with a logarithmic singularity at the origin	374
<a href="#">GiNaC::basic_multi_iterator&lt; T &gt;</a>	
Basic_multi_iterator is a base class	376
<a href="#">GiNaC::basic_partition_generator</a>	
Base class for generating all bounded combinatorial partitions of an integer n with exactly m	
parts in non-decreasing order	381
<a href="#">GiNaC::class_info&lt; OPT &gt;</a>	382
<a href="#">GiNaC::clifford</a>	
This class holds an object representing an element of the Clifford algebra (the Dirac gamma	
matrices)	386
<a href="#">GiNaC::cliffordunit</a>	
This class represents the Clifford algebra generators (units)	395
<a href="#">GiNaC::color</a>	
This class holds a generator T_a or the unity element of the Lie algebra of SU(3), as used for	
calculations in quantum chromodynamics	396
<a href="#">GiNaC::compare_all_equal&lt; T &gt;</a>	
Comparison policy: all structures of one type are equal	401
<a href="#">GiNaC::compare_bitwise&lt; T &gt;</a>	
Comparison policy: use bit-wise comparison to compare structures	403

<a href="#">GiNaC::compare_std_less&lt; T &gt;</a>	
Comparison policy: use <code>std::equal_to</code> / <code>std::less</code> (defaults to operators <code>==</code> and <code>&lt;</code> ) to compare structures	404
<a href="#">GiNaC::composition_generator</a>	
Generate all compositions of a partition of an integer $n$ , starting with the compositions which has non-decreasing order	405
<a href="#">GiNaC::const_iterator</a>	408
<a href="#">GiNaC::const_postorder_iterator</a>	413
<a href="#">GiNaC::const_preorder_iterator</a>	416
<a href="#">GiNaC::constant</a>	
This class holds constants, symbols with specific numerical value	419
<a href="#">GiNaC::container&lt; C &gt;</a>	
Wrapper template for making <a href="#">GiNaC</a> classes out of STL containers	426
<a href="#">GiNaC::container_storage&lt; C &gt;</a>	
Helper template for encapsulating the <a href="#">reserve()</a> mechanics of STL containers	440
<a href="#">GiNaC::composition_generator::coolmulti</a>	443
<a href="#">GiNaC::derivative_map_function</a>	
Function object to be applied by <a href="#">basic::derivative()</a>	445
<a href="#">GiNaC::determinant_algo</a>	
Switch to control algorithm for determinant computation	447
<a href="#">GiNaC::diracgamma</a>	
This class represents the Dirac gamma Lorentz vector	448
<a href="#">GiNaC::diracgamma5</a>	
This class represents the Dirac gamma5 object which anticommutes with all other gammas	449
<a href="#">GiNaC::diracgammaL</a>	
This class represents the Dirac gammaL object which behaves like $1/2 (1-\text{gamma5})$	451
<a href="#">GiNaC::diracgammaR</a>	
This class represents the Dirac gammaL object which behaves like $1/2 (1+\text{gamma5})$	452
<a href="#">GiNaC::diracone</a>	
This class represents the Clifford algebra unity element	453
<a href="#">GiNaC::do_taylor</a>	
Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe	454
<a href="#">GiNaC::domain</a>	
Domain of an object	455
<a href="#">GiNaC::dunno</a>	
Exception class thrown by functions to signal unimplemented functionality so the expression may just be <code>.hold()</code>	455
<a href="#">GiNaC::Ebar_kernel</a>	
The Ebar-kernel	456
<a href="#">GiNaC::Eisenstein_h_kernel</a>	
The kernel corresponding to the Eisenstein series $h_{k,N,\tau,s}(\tau)$	460
<a href="#">GiNaC::Eisenstein_kernel</a>	
The kernel corresponding to the Eisenstein series $E_{k,N,a,b,K}(\tau)$	465
<a href="#">GiNaC::composition_generator::coolmulti::element</a>	471
<a href="#">GiNaC::ELi_kernel</a>	
The ELi-kernel	472
<a href="#">std::equal_to&lt; GiNaC::ex &gt;</a>	
Specialization of <code>std::equal_to()</code> for <code>ex</code> objects	476
<a href="#">GiNaC::error_and_integral</a>	476
<a href="#">GiNaC::error_and_integral_is_less</a>	477
<a href="#">GiNaC::eval_integ_map_function</a>	
Function object to be applied by <a href="#">basic::eval_integ()</a>	478
<a href="#">GiNaC::evalf_map_function</a>	
Function object to be applied by <a href="#">basic::evalf()</a>	479
<a href="#">GiNaC::evalm_map_function</a>	
Function object to be applied by <a href="#">basic::evalm()</a>	479

<a href="#">GiNaC::ex</a>	
Lightweight wrapper for <a href="#">GiNaC</a> 's symbolic objects	480
<a href="#">GiNaC::ex_base_is_less</a>	515
<a href="#">GiNaC::ex_is_equal</a>	515
<a href="#">GiNaC::ex_is_less</a>	516
<a href="#">GiNaC::ex_swap</a>	516
<a href="#">GiNaC::expair</a>	
A pair of expressions	517
<a href="#">GiNaC::expair_is_less</a>	
Function object for insertion into third argument of STL's sort() etc	520
<a href="#">GiNaC::expair_rest_is_less</a>	
Function object not caring about the numerical coefficients for insertion into third argument of STL's sort()	521
<a href="#">GiNaC::expair_swap</a>	521
<a href="#">GiNaC::expairseq</a>	
A sequence of class expair	522
<a href="#">GiNaC::expand_map_function</a>	
Function object to be applied by <a href="#">basic::expand()</a>	535
<a href="#">GiNaC::expand_options</a>	
Flags to control the behavior of <a href="#">expand()</a>	537
<a href="#">GiNaC::factor_options</a>	
Flags to control the polynomial factorization	537
<a href="#">GiNaC::fail</a>	538
<a href="#">GiNaC::fderivative</a>	
This class represents the (abstract) derivative of a symbolic function	539
<a href="#">GiNaC::function</a>	
The class function is used to implement builtin functions like sin, cos..	546
<a href="#">GiNaC::function_options</a>	562
<a href="#">GiNaC::G2_SERIAL</a>	
Generalized multiple polylogarithm	608
<a href="#">GiNaC::G3_SERIAL</a>	
Generalized multiple polylogarithm with explicit imaginary parts	609
<a href="#">GiNaC::gcd_options</a>	
Flags to control the behavior of <a href="#">gcd()</a> and friends	610
<a href="#">GiNaC::gcdheu_failed</a>	
Exception thrown by <a href="#">heur_gcd()</a> to signal failure	611
<a href="#">GiNaC::has_distance&lt; T &gt;</a>	
SFINAE test for distance	611
<a href="#">GiNaC::has_options</a>	
Flags to control the behavior of <a href="#">has()</a>	613
<a href="#">std::hash&lt; GiNaC::ex &gt;</a>	
Specialization of std::hash() for ex objects	613
<a href="#">GiNaC::idx</a>	
This class holds one index of an indexed object	614
<a href="#">GiNaC::idx_is_equal_ignore_dim</a>	624
<a href="#">GiNaC::indexed</a>	
This class holds an indexed expression	624
<a href="#">GiNaC::info_flags</a>	
Possible attributes an object can have	639
<a href="#">GiNaC::integral</a>	
Symbolic integral	641
<a href="#">GiNaC::integration_kernel</a>	
The base class for integration kernels for iterated integrals	649
<a href="#">GiNaC::is_not_a_clifford</a>	
Predicate for finding non-clifford objects	655
<a href="#">GiNaC::is_summation_idx</a>	655
<a href="#">GiNaC::iterated_integral2_SERIAL</a>	
Complete elliptic integral of the first kind	656

<a href="#">GiNaC::iterated_integral3_SERIAL</a>	
Iterated integral with explicit truncation	657
<a href="#">GiNaC::Kronecker_dtau_kernel</a>	
The kernel corresponding to integrating the Kronecker coefficient function $g^{(n)}(z_j, K\tau)$ in $\tau$ (or equivalently in $\bar{q}$ )	657
<a href="#">GiNaC::Kronecker_dz_kernel</a>	
The kernel corresponding to integrating the Kronecker coefficient function $g^{(n-1)}(z - z_j, K\tau)$ in $z$	661
<a href="#">GiNaC::lanczos_coeffs</a>	665
<a href="#">std::less&lt; GiNaC::ptr&lt; T &gt; &gt;</a>	
Specialization of <code>std::less</code> for <code>ptr&lt;T&gt;</code> to enable ordering of <code>ptr&lt;T&gt;</code> objects (e.g	667
<a href="#">GiNaC::library_init</a>	
Helper class to initialize the library	668
<a href="#">GiNaC::make_flat_inserter</a>	
Class to handle the renaming of dummy indices	670
<a href="#">GiNaC::map_function</a>	
Function object for <code>map()</code>	672
<a href="#">GiNaC::matrix</a>	
Symbolic matrices	674
<a href="#">GiNaC::minkmetric</a>	
This class represents a Minkowski metric tensor	695
<a href="#">GiNaC::modular_form_kernel</a>	
A kernel corresponding to a polynomial in Eisenstein series	698
<a href="#">GiNaC::basic_partition_generator::mpartition2</a>	703
<a href="#">GiNaC::mul</a>	
Product of expressions	705
<a href="#">GiNaC::multi_iterator_counter&lt; T &gt;</a>	
The class <code>multi_iterator_counter</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	
$B \leq i_j < N$	
	721
<a href="#">GiNaC::multi_iterator_counter_indv&lt; T &gt;</a>	
The class <code>multi_iterator_counter_indv</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	
$B \leq i_j < N_j$	
	723
<a href="#">GiNaC::multi_iterator_ordered&lt; T &gt;</a>	
The class <code>multi_iterator_ordered</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	
$B \leq i_j < N$	
and	
$i_j < i_{j+1}.$	
It is assumed that $k > 0$ and $N - B \geq k$	726
<a href="#">GiNaC::multi_iterator_ordered_eq&lt; T &gt;</a>	
The class <code>multi_iterator_ordered_eq</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	
$B \leq i_j < N$	
and	
$i_j \leq i_{j+1}.$	
It is assumed that $k > 0$	729
<a href="#">GiNaC::multi_iterator_ordered_eq_indv&lt; T &gt;</a>	
The class <code>multi_iterator_ordered_eq_indv</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	
$B \leq i_j < N_j$	
and	
$i_j \leq i_{j+1}.$	
	732

[GiNaC::multi\\_iterator\\_permutation< T >](#)

The class [multi\\_iterator\\_permutation](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , for which

$$B \leq i_j < N$$

and

$$i_i \neq i_j$$

In particular, if  $N - B = k$ , [multi\\_iterator\\_permutation](#) loops over all permutations of  $k$  elements [735](#)

[GiNaC::multi\\_iterator\\_shuffle< T >](#)

The class [multi\\_iterator\\_shuffle](#) defines a multi\_iterator, which runs over all shuffles of a and b [738](#)

[GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#)

The class [multi\\_iterator\\_shuffle\\_prime](#) defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b) . . . . . [742](#)

[GiNaC::multiple\\_polylog\\_kernel](#)

The integration kernel for multiple polylogarithms . . . . . [744](#)

[GiNaC::ncmul](#)

Non-commutative product of expressions . . . . . [747](#)

[GiNaC::normal\\_map\\_function](#)

Function object to be applied by [basic::normal\(\)](#) . . . . . [756](#)

[GiNaC::numeric](#)

This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy . . . . . [757](#)

[GiNaC::op0\\_is\\_equal](#) . . . . . [786](#)

[GiNaC::partition\\_generator](#)

Generate all bounded combinatorial partitions of an integer n with exactly m parts (not including zero parts) in non-decreasing order . . . . . [786](#)

[GiNaC::partition\\_with\\_zero\\_parts\\_generator](#)

Generate all bounded combinatorial partitions of an integer n with exactly m parts (including zero parts) in non-decreasing order . . . . . [788](#)

[GiNaC::pointer\\_to\\_map\\_function](#) . . . . . [790](#)

[GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >](#) . . . . . [791](#)

[GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >](#) . . . . . [793](#)

[GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >](#) . . . . . [795](#)

[GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >](#) . . . . . [797](#)

[GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >](#) . . . . . [798](#)

[GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >](#) . . . . . [800](#)

[GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >](#) . . . . . [802](#)

[GiNaC::pole\\_error](#)

Exception class thrown when a singularity is encountered . . . . . [805](#)

[GiNaC::possymbol](#)

Specialization of symbol to real positive domain . . . . . [806](#)

[GiNaC::power](#)

This class holds a two-component object, a basis and an exponent representing exponentiation [808](#)

[GiNaC::print\\_context](#)

Base class for print\_contexts . . . . . [822](#)

[GiNaC::print\\_context\\_options](#)

This class stores information about a registered [print\\_context](#) class . . . . . [824](#)

[GiNaC::print\\_csrc](#)

Base context for C source output . . . . . [826](#)

[GiNaC::print\\_csrc\\_cl\\_N](#)

Context for C source output using CLN numbers . . . . . [827](#)

[GiNaC::print\\_csrc\\_double](#)

Context for C source output using double precision . . . . . [828](#)

[GiNaC::print\\_csrc\\_float](#)

Context for C source output using float precision . . . . . [829](#)

[GiNaC::print\\_dflt](#)

Context for default (ginsh-parsable) output . . . . . [830](#)

[GiNaC::print\\_functor](#)

This class represents a print method for a certain algebraic class and [print\\_context](#) type . . . . [831](#)

<a href="#">GiNaC::print_funcutor_impl</a>	
Base class for <a href="#">print_funcutor</a> handlers . . . . .	833
<a href="#">GiNaC::print_latex</a>	
Context for latex-parsable output . . . . .	834
<a href="#">GiNaC::print_memfun_handler&lt; T, C &gt;</a>	
Print_funcutor handler for member functions of class T, context type C . . . . .	835
<a href="#">GiNaC::print_options</a>	
Flags to control the behavior of a <a href="#">print_context</a> . . . . .	837
<a href="#">GiNaC::print_ptrfun_handler&lt; T, C &gt;</a>	
Print_funcutor handler for pointer-to-functions of class T, context type C . . . . .	838
<a href="#">GiNaC::print_python</a>	
Context for python pretty-print output . . . . .	840
<a href="#">GiNaC::print_python_repr</a>	
Context for python-parsable output . . . . .	841
<a href="#">GiNaC::print_tree</a>	
Context for tree-like output for debugging . . . . .	842
<a href="#">GiNaC::archive_node::property</a>	
Archived property (data type, name and associated data) . . . . .	843
<a href="#">GiNaC::archive_node::property_info</a>	
Information about a stored property . . . . .	845
<a href="#">GiNaC::pseries</a>	
This class holds a extended truncated power series (positive and negative integer powers) . . . . .	847
<a href="#">GiNaC::psi1_SERIAL</a>	
Polylogarithm and multiple polylogarithm . . . . .	861
<a href="#">GiNaC::psi2_SERIAL</a>	
Derivatives of Psi-function (aka polygamma-functions) . . . . .	862
<a href="#">GiNaC::ptr&lt; T &gt;</a>	
Class of (intrusively) reference-counted pointers that support copy-on-write semantics . . . . .	863
<a href="#">GiNaC::realsymbol</a>	
Specialization of symbol to real domain . . . . .	868
<a href="#">GiNaC::refcounted</a>	
Base class for reference-counted objects . . . . .	871
<a href="#">GiNaC::registered_class_options</a>	
This class stores information about a registered <a href="#">GiNaC</a> class . . . . .	873
<a href="#">GiNaC::relational</a>	
This class holds a relation consisting of two expressions and a logical relation between them . . . . .	877
<a href="#">GiNaC::remember_strategies</a>	
Strategies how to clean up the function remember cache . . . . .	884
<a href="#">GiNaC::remember_table</a>	
The remember table is organized like an n-fold associative cache in a microprocessor . . . . .	885
<a href="#">GiNaC::remember_table_entry</a>	
A single entry in the remember table of a function . . . . .	889
<a href="#">GiNaC::remember_table_list</a>	
A list of entries in the remember table having some least significant bits of the hashvalue in common . . . . .	892
<a href="#">GiNaC::return_type_t</a>	
To distinguish between different kinds of non-commutative objects . . . . .	893
<a href="#">GiNaC::return_types</a>	
. . . . .	895
<a href="#">GiNaC::relational::safe_bool_helper</a>	
. . . . .	896
<a href="#">GiNaC::scalar_products</a>	
Helper class for storing information about known scalar products which are to be automatically replaced by <a href="#">simplify_indexed()</a> . . . . .	896
<a href="#">GiNaC::series_options</a>	
Flags to control series expansion . . . . .	899
<a href="#">GiNaC::solve_algo</a>	
Switch to control algorithm for linear system solving . . . . .	900
<a href="#">GiNaC::spinidx</a>	
This class holds a spinor index that can be dotted or undotted and that also has a variance . . . . .	901

<a href="#">GiNaC::spinmetric</a>	
This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors . . . . .	906
<a href="#">GiNaC::spmapkey</a> . . . . .	909
<a href="#">GiNaC::status_flags</a>	
Flags to store information about the state of an object . . . . .	911
<a href="#">GiNaC::structure&lt; T, ComparisonPolicy &gt;</a>	
Wrapper template for making <a href="#">GiNaC</a> classes out of C++ structures . . . . .	912
<a href="#">GiNaC::su3d</a>	
This class represents the tensor of symmetric su(3) structure constants . . . . .	928
<a href="#">GiNaC::su3f</a>	
This class represents the tensor of antisymmetric su(3) structure constants . . . . .	930
<a href="#">GiNaC::su3one</a>	
This class represents the su(3) unity element . . . . .	932
<a href="#">GiNaC::su3t</a>	
This class represents an su(3) generator . . . . .	933
<a href="#">GiNaC::subs_options</a>	
Flags to control the behavior of <a href="#">subs()</a> . . . . .	934
<a href="#">GiNaC::sy_is_less</a> . . . . .	935
<a href="#">GiNaC::sy_swap</a> . . . . .	936
<a href="#">GiNaC::sym_desc</a>	
This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b" . . . . .	938
<a href="#">GiNaC::symbol</a>	
Basic CAS symbol . . . . .	941
<a href="#">GiNaC::symbolset</a> . . . . .	951
<a href="#">GiNaC::symmetry</a>	
This class describes the symmetry of a group of indices . . . . .	952
<a href="#">GiNaC::symminfo</a>	
This structure stores the individual symmetrized terms obtained during the simplification of sums . . . . .	959
<a href="#">GiNaC::symminfo_is_less_by_orig</a> . . . . .	961
<a href="#">GiNaC::symminfo_is_less_by_symmterm</a> . . . . .	962
<a href="#">GiNaC::tensdelta</a>	
This class represents the delta tensor . . . . .	962
<a href="#">GiNaC::tensepsilon</a>	
This class represents the totally antisymmetric epsilon tensor . . . . .	965
<a href="#">GiNaC::tensmetric</a>	
This class represents a general metric tensor which can be used to raise/lower indices . . . . .	969
<a href="#">GiNaC::tensor</a>	
This class holds one of <a href="#">GiNaC</a> 's predefined special tensors such as the delta and the metric tensors . . . . .	971
<a href="#">GiNaC::terminfo</a>	
This structure stores the original and symmetrized versions of terms obtained during the simplification of sums . . . . .	973
<a href="#">GiNaC::terminfo_is_less</a> . . . . .	974
<a href="#">GiNaC::class_info&lt; OPT &gt;::tree_node</a> . . . . .	974
<a href="#">GiNaC::unarchive_table_t</a> . . . . .	976
<a href="#">GiNaC::user_defined_kernel</a>	
A user-defined integration kernel . . . . .	977
<a href="#">GiNaC::varidx</a>	
This class holds an index with a variance (co- or contravariant) . . . . .	981
<a href="#">GiNaC::visitor</a>	
Degenerate base class for visitors . . . . .	985
<a href="#">GiNaC::wildcard</a>	
This class acts as a wildcard for <a href="#">subs()</a> , <a href="#">match()</a> , <a href="#">has()</a> and <a href="#">find()</a> . . . . .	986
<a href="#">GiNaC::zeta1_SERIAL</a>	
Complex conjugate . . . . .	990

[GiNaC::zeta2\\_SERIAL](#)Alternating Euler sum or colored MZV . . . . . [990](#)



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">add.cpp</a>	Implementation of <a href="#">GiNaC</a> 's sums of expressions . . . . .	993
<a href="#">add.h</a>	Interface to <a href="#">GiNaC</a> 's sums of expressions . . . . .	994
<a href="#">archive.cpp</a>	Archiving of <a href="#">GiNaC</a> expressions . . . . .	994
<a href="#">archive.h</a>	Archiving of <a href="#">GiNaC</a> expressions . . . . .	995
<a href="#">assertion.h</a>	Assertion macro definition . . . . .	998
<a href="#">basic.cpp</a>	Implementation of <a href="#">GiNaC</a> 's ABC . . . . .	999
<a href="#">basic.h</a>	Interface to <a href="#">GiNaC</a> 's ABC . . . . .	1000
<a href="#">class_info.h</a>	Helper templates to provide per-class information for class hierarchies . . . . .	1001
<a href="#">clifford.cpp</a>	Implementation of <a href="#">GiNaC</a> 's clifford algebra (Dirac gamma) objects . . . . .	1002
<a href="#">clifford.h</a>	Interface to <a href="#">GiNaC</a> 's clifford algebra (Dirac gamma) objects . . . . .	1004
<a href="#">color.cpp</a>	Implementation of <a href="#">GiNaC</a> 's color (SU(3) Lie algebra) objects . . . . .	1006
<a href="#">color.h</a>	Interface to <a href="#">GiNaC</a> 's color (SU(3) Lie algebra) objects . . . . .	1008
<a href="#">compiler.h</a>	Definition of optimizing macros . . . . .	1009
<a href="#">constant.cpp</a>	Implementation of <a href="#">GiNaC</a> 's constant types and some special constants . . . . .	1010
<a href="#">constant.h</a>	Interface to <a href="#">GiNaC</a> 's constant types and some special constants . . . . .	1011
<a href="#">container.h</a>	Wrapper template for making <a href="#">GiNaC</a> classes out of STL containers . . . . .	1012
<a href="#">crc32.h</a>	CRC32 hash function . . . . .	1012
<a href="#">ex.cpp</a>	Implementation of <a href="#">GiNaC</a> 's light-weight expression handles . . . . .	1013

<a href="#">ex.h</a>	Interface to <a href="#">GiNaC</a> 's light-weight expression handles	1013
<a href="#">excompiler.cpp</a>	Functions to facilitate the conversion of a <a href="#">ex</a> to a function pointer suited for fast numerical integration	1016
<a href="#">excompiler.h</a>	Functions to facilitate the conversion of a <a href="#">ex</a> to a function pointer suited for fast numerical integration	1016
<a href="#">expair.cpp</a>	Implementation of expression pairs (building blocks of <a href="#">expairseq</a> )	1017
<a href="#">expair.h</a>	Definition of expression pairs (building blocks of <a href="#">expairseq</a> )	1018
<a href="#">expairseq.cpp</a>	Implementation of sequences of expression pairs	1019
<a href="#">expairseq.h</a>	Interface to sequences of expression pairs	1019
<a href="#">exprseq.cpp</a>	Implementation of <a href="#">GiNaC</a> 's <a href="#">exprseq</a>	1020
<a href="#">exprseq.h</a>	Definition of <a href="#">GiNaC</a> 's <a href="#">exprseq</a>	1020
<a href="#">factor.cpp</a>	Polynomial factorization (implementation)	1021
<a href="#">factor.h</a>	Polynomial factorization	1029
<a href="#">fail.cpp</a>	Implementation of class signaling failure of operation	1029
<a href="#">fail.h</a>	Interface to class signaling failure of operation	1030
<a href="#">fderivative.cpp</a>	Implementation of abstract derivatives of functions	1031
<a href="#">fderivative.h</a>	Interface to abstract derivatives of functions	1031
<a href="#">flags.h</a>	Collection of all flags used through the <a href="#">GiNaC</a> framework	1032
<a href="#">function.cpp</a>	Implementation of class of symbolic functions	1033
<a href="#">function.h</a>	Interface to class of symbolic functions	1033
<a href="#">ginac.h</a>	This include file includes all other public <a href="#">GiNaC</a> headers	1045
<a href="#">hash_map.h</a>	Replacement for <code>map&lt;&gt;</code> using hash tables	1046
<a href="#">hash_seed.h</a>	Type-specific hash seed	1046
<a href="#">idx.cpp</a>	Implementation of <a href="#">GiNaC</a> 's indices	1047
<a href="#">idx.h</a>	Interface to <a href="#">GiNaC</a> 's indices	1048
<a href="#">indexed.cpp</a>	Implementation of <a href="#">GiNaC</a> 's indexed expressions	1049
<a href="#">indexed.h</a>	Interface to <a href="#">GiNaC</a> 's indexed expressions	1051
<a href="#">inifcns.cpp</a>	Implementation of <a href="#">GiNaC</a> 's initially known functions	1052
<a href="#">inifcns.h</a>	Interface to <a href="#">GiNaC</a> 's initially known functions	1055
<a href="#">inifcns_elliptic.cpp</a>	Implementation of some special functions related to elliptic curves	1056

<a href="#">inifcns_gamma.cpp</a>	Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff	1058
<a href="#">inifcns_nstdsums.cpp</a>	Implementation of some special functions that have a representation as nested sums	1059
<a href="#">inifcns_trans.cpp</a>	Implementation of transcendental (and trigonometric and hyperbolic) functions	1061
<a href="#">integral.cpp</a>	Implementation of <a href="#">GiNaC</a> 's symbolic integral	1064
<a href="#">integral.h</a>	Interface to <a href="#">GiNaC</a> 's symbolic integral	1065
<a href="#">integration_kernel.cpp</a>	Implementation of <a href="#">GiNaC</a> 's integration kernels for iterated integrals	1065
<a href="#">integration_kernel.h</a>	Interface to <a href="#">GiNaC</a> 's integration kernels for iterated integrals	1068
<a href="#">lst.cpp</a>	Implementation of <a href="#">GiNaC</a> 's <code>lst</code>	1069
<a href="#">lst.h</a>	Definition of <a href="#">GiNaC</a> 's <code>lst</code>	1069
<a href="#">matrix.cpp</a>	Implementation of symbolic matrices	1070
<a href="#">matrix.h</a>	Interface to symbolic matrices	1071
<a href="#">mul.cpp</a>	Implementation of <a href="#">GiNaC</a> 's products of expressions	1072
<a href="#">mul.h</a>	Interface to <a href="#">GiNaC</a> 's products of expressions	1073
<a href="#">ncmul.cpp</a>	Implementation of <a href="#">GiNaC</a> 's non-commutative products of expressions	1074
<a href="#">ncmul.h</a>	Interface to <a href="#">GiNaC</a> 's non-commutative products of expressions	1075
<a href="#">normal.cpp</a>	This file implements several functions that work on univariate and multivariate polynomials and rational functions	1075
<a href="#">normal.h</a>	This file defines several functions that work on univariate and multivariate polynomials and rational functions	1078
<a href="#">numeric.cpp</a>	This file contains the interface to the underlying bignum package	1079
<a href="#">numeric.h</a>	Makes the interface to the underlying bignum package available	1083
<a href="#">operators.cpp</a>	Implementation of <a href="#">GiNaC</a> 's overloaded operators	1085
<a href="#">operators.h</a>	Interface to <a href="#">GiNaC</a> 's overloaded operators	1087
<a href="#">power.cpp</a>	Implementation of <a href="#">GiNaC</a> 's symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ )	1089
<a href="#">power.h</a>	Interface to <a href="#">GiNaC</a> 's symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ )	1090
<a href="#">print.cpp</a>	Implementation of helper classes for expression output	1090
<a href="#">print.h</a>	Definition of helper classes for expression output	1091
<a href="#">pseries.cpp</a>	Implementation of class for extended truncated power series and methods for series expansion	1094
<a href="#">pseries.h</a>	Interface to class for extended truncated power series	1094
<a href="#">ptr.h</a>	Reference-counted pointer template	1095

<a href="#">registrar.cpp</a>	
GiNaC's class registrar (for class basic and all classes derived from it)	1096
<a href="#">registrar.h</a>	
GiNaC's class registrar (for class basic and all classes derived from it)	1096
<a href="#">relational.cpp</a>	
Implementation of relations between expressions	1100
<a href="#">relational.h</a>	
Interface to relations between expressions	1100
<a href="#">remember.cpp</a>	
Implementation of helper classes for using the remember option in GiNaC functions	1101
<a href="#">remember.h</a>	
Interface to helper classes for using the remember option in GiNaC functions	1101
<a href="#">structure.h</a>	
Wrapper template for making GiNaC classes out of C++ structures	1102
<a href="#">symbol.cpp</a>	
Implementation of GiNaC's symbolic objects	1103
<a href="#">symbol.h</a>	
Interface to GiNaC's symbolic objects	1103
<a href="#">symmetry.cpp</a>	
Implementation of GiNaC's symmetry definitions	1104
<a href="#">symmetry.h</a>	
Interface to GiNaC's symmetry definitions	1105
<a href="#">tensor.cpp</a>	
Implementation of GiNaC's special tensors	1107
<a href="#">tensor.h</a>	
Interface to GiNaC's special tensors	1108
<a href="#">utils.cpp</a>	
Implementation of several small and furry utilities needed within GiNaC but not of any interest to the user of the library	1109
<a href="#">utils.h</a>	
Interface to several small and furry utilities needed within GiNaC but not of any interest to the user of the library	1111
<a href="#">utils_multi_iterator.h</a>	
Utilities for summing over multiple indices	1114
<a href="#">version.h</a>	
GiNaC library version information	1116
<a href="#">wildcard.cpp</a>	
Implementation of GiNaC's wildcard objects	1118
<a href="#">wildcard.h</a>	
Interface to GiNaC's wildcard objects	1119

## Chapter 5

# Namespace Documentation

### 5.1 GiNaC Namespace Reference

#### Namespaces

- [internal](#)

#### Classes

- class [\\_numeric\\_digits](#)  
*This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.*
- class [add](#)  
*Sum of expressions.*
- class [archive](#)  
*This class holds archived versions of [GiNaC](#) expressions (class *ex*).*
- class [archive\\_node](#)  
*This class stores all properties needed to record/retrieve the state of one object of class *basic* (or a derived class).*
- class [basic](#)  
*This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.*
- class [basic\\_log\\_kernel](#)  
*The basic integration kernel with a logarithmic singularity at the origin.*
- class [basic\\_multi\\_iterator](#)  
*[basic\\_multi\\_iterator](#) is a base class.*
- class [basic\\_partition\\_generator](#)  
*Base class for generating all bounded combinatorial partitions of an integer *n* with exactly *m* parts in non-decreasing order.*
- class [class\\_info](#)
- class [clifford](#)  
*This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).*
- class [cliffordunit](#)  
*This class represents the Clifford algebra generators (units).*
- class [color](#)  
*This class holds a generator  $T_a$  or the unity element of the Lie algebra of  $SU(3)$ , as used for calculations in quantum chromodynamics.*
- class [compare\\_all\\_equal](#)

- Comparison policy: all structures of one type are equal.*

  - class [compare\\_bitwise](#)

*Comparison policy: use bit-wise comparison to compare structures.*
- class [compare\\_std\\_less](#)

*Comparison policy: use `std::equal_to`/`std::less` (defaults to operators `==` and `<`) to compare structures.*
- class [composition\\_generator](#)

*Generate all compositions of a partition of an integer  $n$ , starting with the compositions which has non-decreasing order.*
- class [const\\_iterator](#)
  - class [const\\_postorder\\_iterator](#)
  - class [const\\_preorder\\_iterator](#)
  - class [constant](#)

*This class holds constants, symbols with specific numerical value.*
- class [container](#)

*Wrapper template for making [GiNaC](#) classes out of STL containers.*
- class [container\\_storage](#)

*Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.*
- struct [derivative\\_map\\_function](#)

*Function object to be applied by [basic::derivative\(\)](#).*
- class [determinant\\_algo](#)

*Switch to control algorithm for determinant computation.*
- class [diracgamma](#)

*This class represents the Dirac gamma Lorentz vector.*
- class [diracgamma5](#)

*This class represents the Dirac gamma5 object which anticommutes with all other gammas.*
- class [diracgammaL](#)

*This class represents the Dirac gammaL object which behaves like  $1/2 (1-\text{gamma5})$ .*
- class [diracgammaR](#)

*This class represents the Dirac gammaL object which behaves like  $1/2 (1+\text{gamma5})$ .*
- class [diracone](#)

*This class represents the Clifford algebra unity element.*
- class [do\\_taylor](#)

*Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.*
- class [domain](#)

*Domain of an object.*
- class [dunno](#)

*Exception class thrown by functions to signal unimplemented functionality so the expression may just be `.hold()`*
- class [Ebar\\_kernel](#)

*The Ebar-kernel.*
- class [Eisenstein\\_h\\_kernel](#)

*The kernel corresponding to the Eisenstein series  $h_{k,N,\tau,s}(\tau)$ .*
- class [Eisenstein\\_kernel](#)

*The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .*
- class [ELi\\_kernel](#)

*The ELi-kernel.*
- struct [error\\_and\\_integral](#)
  - struct [error\\_and\\_integral\\_is\\_less](#)
  - struct [eval\\_integ\\_map\\_function](#)

*Function object to be applied by [basic::eval\\_integ\(\)](#).*
- struct [evalf\\_map\\_function](#)

- Function object to be applied by [basic::evalf\(\)](#).
- struct [evalm\\_map\\_function](#)
  - Function object to be applied by [basic::evalm\(\)](#).
- class [ex](#)
  - Lightweight wrapper for [GiNaC](#)'s symbolic objects.
- struct [ex\\_base\\_is\\_less](#)
- struct [ex\\_is\\_equal](#)
- struct [ex\\_is\\_less](#)
- struct [ex\\_swap](#)
- class [expair](#)
  - A pair of expressions.
- struct [expair\\_is\\_less](#)
  - Function object for insertion into third argument of STL's [sort\(\)](#) etc.
- struct [expair\\_rest\\_is\\_less](#)
  - Function object not caring about the numerical coefficients for insertion into third argument of STL's [sort\(\)](#).
- struct [expair\\_swap](#)
- class [expairseq](#)
  - A sequence of class [expair](#).
- struct [expand\\_map\\_function](#)
  - Function object to be applied by [basic::expand\(\)](#).
- class [expand\\_options](#)
  - Flags to control the behavior of [expand\(\)](#).
- class [factor\\_options](#)
  - Flags to control the polynomial factorization.
- class [fail](#)
- class [fderivative](#)
  - This class represents the (abstract) derivative of a symbolic function.
- class [function](#)
  - The class [function](#) is used to implement builtin functions like [sin](#), [cos](#)...
- class [function\\_options](#)
- class [G2\\_SERIAL](#)
  - Generalized multiple polylogarithm.
- class [G3\\_SERIAL](#)
  - Generalized multiple polylogarithm with explicit imaginary parts.
- struct [gcd\\_options](#)
  - Flags to control the behavior of [gcd\(\)](#) and friends.
- class [gcdheu\\_failed](#)
  - Exception thrown by [heur\\_gcd\(\)](#) to signal failure.
- class [has\\_distance](#)
  - SFINAE test for distance.
- class [has\\_options](#)
  - Flags to control the behavior of [has\(\)](#).
- class [idx](#)
  - This class holds one index of an indexed object.
- struct [idx\\_is\\_equal\\_ignore\\_dim](#)
- class [indexed](#)
  - This class holds an indexed expression.
- class [info\\_flags](#)
  - Possible attributes an object can have.
- class [integral](#)
  - Symbolic integral.

- class [integration\\_kernel](#)  
The base class for integration kernels for iterated integrals.
- struct [is\\_not\\_a\\_clifford](#)  
Predicate for finding non-clifford objects.
- struct [is\\_summation\\_idx](#)
- class [iterated\\_integral2\\_SERIAL](#)  
Complete elliptic integral of the first kind.
- class [iterated\\_integral3\\_SERIAL](#)  
Iterated integral with explicit truncation.
- class [Kronecker\\_dtau\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).
- class [Kronecker\\_dz\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .
- class [lanczos\\_coeffs](#)
- class [library\\_init](#)  
Helper class to initialize the library.
- class [make\\_flat\\_inserter](#)  
Class to handle the renaming of dummy indices.
- struct [map\\_function](#)  
Function object for `map()`.
- class [matrix](#)  
Symbolic matrices.
- class [minkmetric](#)  
This class represents a Minkowski metric tensor.
- class [modular\\_form\\_kernel](#)  
A kernel corresponding to a polynomial in Eisenstein series.
- class [mul](#)  
Product of expressions.
- class [multi\\_iterator\\_counter](#)  
The class [multi\\_iterator\\_counter](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that
 
$$B \leq i_j < N$$
- class [multi\\_iterator\\_counter\\_indv](#)  
The class [multi\\_iterator\\_counter\\_indv](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that
 
$$B \leq i_j < N_j$$
- class [multi\\_iterator\\_ordered](#)  
The class [multi\\_iterator\\_ordered](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that
 
$$B \leq i_j < N$$

and

$$i_j < i_{j+1}.$$

It is assumed that  $k > 0$  and  $N - B \geq k$ .
- class [multi\\_iterator\\_ordered\\_eq](#)  
The class [multi\\_iterator\\_ordered\\_eq](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that
 
$$B \leq i_j < N$$

and

$$i_j \leq i_{j+1}.$$

It is assumed that  $k > 0$ .



- class [multi\\_iterator\\_ordered\\_eq\\_indv](#)

The class [multi\\_iterator\\_ordered\\_eq\\_indv](#) defines a [multi\\_iterator](#)  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N_j$$

and

$$i_j \leq i_{j+1}.$$

- class [multi\\_iterator\\_permutation](#)

The class [multi\\_iterator\\_permutation](#) defines a [multi\\_iterator](#)  $(i_1, i_2, \dots, i_k)$ , for which

$$B \leq i_j < N$$

and

$$i_i \neq i_j$$

In particular, if  $N - B = k$ , [multi\\_iterator\\_permutation](#) loops over all permutations of  $k$  elements.

- class [multi\\_iterator\\_shuffle](#)

The class [multi\\_iterator\\_shuffle](#) defines a [multi\\_iterator](#), which runs over all shuffles of  $a$  and  $b$ .

- class [multi\\_iterator\\_shuffle\\_prime](#)

The class [multi\\_iterator\\_shuffle\\_prime](#) defines a [multi\\_iterator](#), which runs over all shuffles of  $a$  and  $b$ , excluding the first one  $(a,b)$ .

- class [multiple\\_polylog\\_kernel](#)

The integration kernel for multiple polylogarithms.

- class [ncmul](#)

Non-commutative product of expressions.

- struct [normal\\_map\\_function](#)

Function object to be applied by [basic::normal\(\)](#).

- class [numeric](#)

This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.

- struct [op0\\_is\\_equal](#)

- class [partition\\_generator](#)

Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (not including zero parts) in non-decreasing order.

- class [partition\\_with\\_zero\\_parts\\_generator](#)

Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (including zero parts) in non-decreasing order.

- class [pointer\\_to\\_map\\_function](#)

- class [pointer\\_to\\_map\\_function\\_1arg](#)

- class [pointer\\_to\\_map\\_function\\_2args](#)

- class [pointer\\_to\\_map\\_function\\_3args](#)

- class [pointer\\_to\\_member\\_to\\_map\\_function](#)

- class [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg](#)

- class [pointer\\_to\\_member\\_to\\_map\\_function\\_2args](#)

- class [pointer\\_to\\_member\\_to\\_map\\_function\\_3args](#)

- class [pole\\_error](#)

Exception class thrown when a singularity is encountered.

- class [possymbol](#)

Specialization of symbol to real positive domain.

- class [power](#)

This class holds a two-component object, a basis and an exponent representing exponentiation.

- class [print\\_context](#)

Base class for [print\\_contexts](#).

- class [print\\_context\\_options](#)

This class stores information about a registered [print\\_context](#) class.

- class [print\\_csrc](#)

- Base context for C source output.*

  - class [print\\_csrc\\_cl\\_N](#)

*Context for C source output using CLN numbers.*
  - class [print\\_csrc\\_double](#)

*Context for C source output using double precision.*
  - class [print\\_csrc\\_float](#)

*Context for C source output using float precision.*
  - class [print\\_dflt](#)

*Context for default (ginsh-parsable) output.*
  - class [print\\_functor](#)

*This class represents a print method for a certain algebraic class and [print\\_context](#) type.*
  - class [print\\_functor\\_impl](#)

*Base class for [print\\_functor](#) handlers.*
  - class [print\\_latex](#)

*Context for latex-parsable output.*
  - class [print\\_memfun\\_handler](#)

*[print\\_functor](#) handler for member functions of class T, context type C*
  - class [print\\_options](#)

*Flags to control the behavior of a [print\\_context](#).*
  - class [print\\_ptrfun\\_handler](#)

*[print\\_functor](#) handler for pointer-to-functions of class T, context type C*
  - class [print\\_python](#)

*Context for python pretty-print output.*
  - class [print\\_python\\_repr](#)

*Context for python-parsable output.*
  - class [print\\_tree](#)

*Context for tree-like output for debugging.*
  - class [pseries](#)

*This class holds a extended truncated power series (positive and negative integer powers).*
  - class [psi1\\_SERIAL](#)

*Polylogarithm and multiple polylogarithm.*
  - class [psi2\\_SERIAL](#)

*Derivatives of Psi-function (aka polygamma-functions).*
  - class [ptr](#)

*Class of (intrusively) reference-counted pointers that support copy-on-write semantics.*
  - class [realsymbol](#)

*Specialization of symbol to real domain.*
  - class [refcounted](#)

*Base class for reference-counted objects.*
  - class [registered\\_class\\_options](#)

*This class stores information about a registered [GiNaC](#) class.*
  - class [relational](#)

*This class holds a relation consisting of two expressions and a logical relation between them.*
  - class [remember\\_strategies](#)

*Strategies how to clean up the function remember cache.*
  - class [remember\\_table](#)

*The remember table is organized like an n-fold associative cache in a microprocessor.*
  - class [remember\\_table\\_entry](#)

*A single entry in the remember table of a function.*
  - class [remember\\_table\\_list](#)

*A list of entries in the remember table having some least significant bits of the hashvalue in common.*

- struct [return\\_type\\_t](#)  
*To distinguish between different kinds of non-commutative objects.*
- class [return\\_types](#)
- class [scalar\\_products](#)  
*Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).*
- class [series\\_options](#)  
*Flags to control series expansion.*
- class [solve\\_algo](#)  
*Switch to control algorithm for linear system solving.*
- class [spinidx](#)  
*This class holds a spinor index that can be dotted or undotted and that also has a variance.*
- class [spinmetric](#)  
*This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.*
- class [smapkey](#)
- class [status\\_flags](#)  
*Flags to store information about the state of an object.*
- class [structure](#)  
*Wrapper template for making [GiNaC](#) classes out of C++ structures.*
- class [su3d](#)  
*This class represents the tensor of symmetric  $su(3)$  structure constants.*
- class [su3f](#)  
*This class represents the tensor of antisymmetric  $su(3)$  structure constants.*
- class [su3one](#)  
*This class represents the  $su(3)$  unity element.*
- class [su3t](#)  
*This class represents an  $su(3)$  generator.*
- class [subs\\_options](#)  
*Flags to control the behavior of [subs\(\)](#).*
- class [sy\\_is\\_less](#)
- class [sy\\_swap](#)
- struct [sym\\_desc](#)  
*This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".*
- class [symbol](#)  
*Basic CAS symbol.*
- class [symbolset](#)
- class [symmetry](#)  
*This class describes the symmetry of a group of indices.*
- class [symminfo](#)  
*This structure stores the individual symmetrized terms obtained during the simplification of sums.*
- class [symminfo\\_is\\_less\\_by\\_orig](#)
- class [symminfo\\_is\\_less\\_by\\_symmterm](#)
- class [tensdelta](#)  
*This class represents the delta tensor.*
- class [tensepsilon](#)  
*This class represents the totally antisymmetric epsilon tensor.*
- class [tensmetric](#)  
*This class represents a general metric tensor which can be used to raise/lower indices.*
- class [tensor](#)  
*This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.*

- class [terminfo](#)  
*This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.*
- class [terminfo\\_is\\_less](#)
- class [unarchive\\_table\\_t](#)
- class [user\\_defined\\_kernel](#)  
*A user-defined integration kernel.*
- class [varidx](#)  
*This class holds an index with a variance (co- or contravariant).*
- class [visitor](#)  
*Degenerate base class for visitors.*
- class [wildcard](#)  
*This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).*
- class [zeta1\\_SERIAL](#)  
*Complex conjugate.*
- class [zeta2\\_SERIAL](#)  
*Alternating Euler sum or colored MZV.*

## Typedefs

- typedef unsigned [archive\\_node\\_id](#)  
*Numerical ID value to refer to an [archive\\_node](#).*
- typedef unsigned [archive\\_atom](#)  
*Numerical ID value to refer to a string.*
- typedef [basic](#) [basic](#) [synthesize\\_func](#) (\*)
- typedef std::map< std::string, [synthesize\\_func](#) > [unarchive\\_map\\_t](#)
- typedef std::vector< [ex](#) > [exvector](#)
- typedef std::set< [ex](#), [ex\\_is\\_less](#) > [exset](#)
- typedef std::map< [ex](#), [ex](#), [ex\\_is\\_less](#) > [exmap](#)
- typedef [ex](#) [evalfunctype](#) (\*)
- typedef double [FUNC\\_P1](#) (double)  
*Function pointer with one function parameter.*
- typedef double [FUNC\\_P2](#) (double, double)  
*Function pointer with two function parameters.*
- typedef void [FUNC\\_CUBA](#) (const int \*, const double[], const int \*, double[])  
*Function pointer for use with the CUBA library (<http://www.feynarts.de/cuba>).*
- typedef std::vector< [expair](#) > [epvector](#)  
*expair-vector*
- typedef [epvector](#)::iterator [epp](#)  
*expair-vector pointer*
- typedef [container](#) < std::vector > [exprseq](#)
- typedef std::multiset< unsigned > [paramset](#)
- typedef [ex](#) [eval\\_funcp](#) (\*)
- typedef [ex](#) [evalf\\_funcp](#) (\*)
- typedef [ex](#) [conjugate\\_funcp](#) (\*)
- typedef [ex](#) [real\\_part\\_funcp](#) (\*)
- typedef [ex](#) [imag\\_part\\_funcp](#) (\*)
- typedef [ex](#) [expand\\_funcp](#) (\*)
- typedef [ex](#) [derivative\\_funcp](#) (\*)
- typedef [ex](#) [expl\\_derivative\\_funcp](#) (\*)
- typedef [ex](#) [power\\_funcp](#) (\*)
- typedef [ex](#) [series\\_funcp](#) (\*)

- typedef void(\* [print\\_funcp](#)) ()
- typedef bool(\* [info\\_funcp](#)) ()
- typedef [ex](#)(\* [eval\\_funcp\\_1](#)) (const [ex](#) &)
- typedef [ex](#)(\* [evalf\\_funcp\\_1](#)) (const [ex](#) &)
- typedef [ex](#)(\* [conjugate\\_funcp\\_1](#)) (const [ex](#) &)
- typedef [ex](#)(\* [real\\_part\\_funcp\\_1](#)) (const [ex](#) &)
- typedef [ex](#)(\* [imag\\_part\\_funcp\\_1](#)) (const [ex](#) &)
- typedef [ex](#)(\* [expand\\_funcp\\_1](#)) (const [ex](#) &, unsigned)
- typedef [ex](#)(\* [derivative\\_funcp\\_1](#)) (const [ex](#) &, unsigned)
- typedef [ex](#)(\* [expl\\_derivative\\_funcp\\_1](#)) (const [ex](#) &, const [symbol](#) &)
- typedef [ex](#)(\* [power\\_funcp\\_1](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [series\\_funcp\\_1](#)) (const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void(\* [print\\_funcp\\_1](#)) (const [ex](#) &, const [print\\_context](#) &)
- typedef bool(\* [info\\_funcp\\_1](#)) (const [ex](#) &, unsigned)
- typedef [ex](#)(\* [eval\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [evalf\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [conjugate\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [real\\_part\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [imag\\_part\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [expand\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [derivative\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [expl\\_derivative\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &, const [symbol](#) &)
- typedef [ex](#)(\* [power\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [series\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void(\* [print\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &, const [print\\_context](#) &)
- typedef bool(\* [info\\_funcp\\_2](#)) (const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [eval\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [evalf\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [conjugate\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [real\\_part\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [imag\\_part\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [expand\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [derivative\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [expl\\_derivative\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [symbol](#) &)
- typedef [ex](#)(\* [power\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [series\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void(\* [print\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [print\\_context](#) &)
- typedef bool(\* [info\\_funcp\\_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [eval\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [evalf\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [conjugate\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [real\\_part\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [imag\\_part\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [expand\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [derivative\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [expl\\_derivative\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [symbol](#) &)
- typedef [ex](#)(\* [power\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [series\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void(\* [print\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [print\\_context](#) &)
- typedef bool(\* [info\\_funcp\\_4](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [eval\\_funcp\\_5](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [evalf\\_funcp\\_5](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [conjugate\\_funcp\\_5](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [real\\_part\\_funcp\\_5](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [imag\\_part\\_funcp\\_5](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)



- [illegible]







- [illegible]

- typedef [ex](#)(\* [evalf\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [conjugate\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [real\\_part\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [imag\\_part\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [expand\\_funcp\\_exvector](#)) (const [exvector](#) &, unsigned)
- typedef [ex](#)(\* [derivative\\_funcp\\_exvector](#)) (const [exvector](#) &, unsigned)
- typedef [ex](#)(\* [expl\\_derivative\\_funcp\\_exvector](#)) (const [exvector](#) &, const [symbol](#) &)
- typedef [ex](#)(\* [power\\_funcp\\_exvector](#)) (const [exvector](#) &, const [ex](#) &)
- typedef [ex](#)(\* [series\\_funcp\\_exvector](#)) (const [exvector](#) &, const [relational](#) &, int, unsigned)
- typedef void(\* [print\\_funcp\\_exvector](#)) (const [exvector](#) &, const [print\\_context](#) &)
- typedef bool(\* [info\\_funcp\\_exvector](#)) (const [exvector](#) &, unsigned)
- template<typename T, class Hash = std::hash<[ex](#)>, class KeyEqual = std::equal\_to<[ex](#)>, class Allocator = std::allocator<std::pair<const [ex](#), T>>>>  
using [exhashmap](#) = std::unordered\_map<[ex](#), T, Hash, KeyEqual, Allocator >
- typedef std::map< [spmapkey](#), [ex](#) > [spmap](#)
- typedef map< [error\\_and\\_integral](#), [ex](#), [error\\_and\\_integral\\_is\\_less](#) > [lookup\\_map](#)
- typedef [container](#)< std::list > [lst](#)
- typedef std::vector< std::size\_t > [uintvector](#)
- typedef std::vector< unsigned > [unsignedvector](#)
- typedef std::vector< [exvector](#) > [exvectorvector](#)
- typedef std::vector< [sym\\_desc](#) > [sym\\_desc\\_vec](#)
- typedef void(\* [digits\\_changed\\_callback](#)) (long)  
*Function pointer to implement callbacks in the case 'Digits' gets changed.*
- typedef class\_info< [print\\_context\\_options](#) > [print\\_context\\_class\\_info](#)
- typedef class\_info< [registered\\_class\\_options](#) > [registered\\_class\\_info](#)

## Enumerations

- enum { [callback\\_registered](#) = 1 }

## Functions

- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (add, [expairseq](#), [print\\_func](#)< [print\\_context](#) >(&add::do\_print). [print\\_func](#)< [print\\_latex](#) >(&add::do\_print\_latex). [print\\_func](#)< [print\\_csrc](#) >(&add::do\_print\_csrc). [print\\_func](#)< [print\\_tree](#) >(&add::do\_print\_tree). [print\\_func](#)< [print\\_python\\_repr](#) >(&add::do\_print\_python\_repr))  
[add](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) (add)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (add)
- static void [write\\_unsigned](#) (std::ostream &os, unsigned val)  
*Write unsigned integer quantity to stream.*
- static unsigned [read\\_unsigned](#) (std::istream &is)  
*Read unsigned integer quantity from stream.*
- std::ostream & [operator<<](#) (std::ostream &os, const [archive\\_node](#) &n)  
*Write [archive\\_node](#) to binary data stream.*
- std::ostream & [operator<<](#) (std::ostream &os, const [archive](#) &ar)  
*Write archive to binary data stream.*
- std::istream & [operator>>](#) (std::istream &is, [archive\\_node](#) &n)  
*Read [archive\\_node](#) from binary data stream.*
- std::istream & [operator>>](#) (std::istream &is, [archive](#) &ar)  
*Read archive from binary data stream.*
- static [synthesize\\_func find\\_factory\\_fcn](#) (const std::string &name)

- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`basic`, `void`, `print_func`< `print_context` >(&`basic::do_print`), `print_func`< `print_tree` >(&`basic::do_print_tree`), `print_func`< `print_python_repr` >(&`basic::do_print_python_repr`))  
`basic`  
*basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by `duplicate()`), so it can copy the `tinfo_key` and the hash value.*
- `template<class T >`  
`bool is_a` (`const basic &obj`)  
*Check if obj is a T, including base classes.*
- `template<class T >`  
`bool is_exactly_a` (`const basic &obj`)  
*Check if obj is a T, not including base classes.*
- `template<class B , typename... Args>`  
`B & dynallocate` (`Args &&... args`)  
*Constructs a new (class basic or derived) B object on the heap.*
- `template<class B >`  
`B & dynallocate` (`std::initializer_list< ex > il`)  
*Constructs a new (class basic or derived) B object on the heap.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`clifford`, `indexed`, `print_func`< `print_dflt` >(&`clifford::do_print_dflt`), `print_func`< `print_latex` >(&`clifford::do_print_latex`), `print_func`< `print_tree` >(&`clifford::do_print_tree`)) `GI↔`  
`NAC_IMPLEMENT_REGISTERED_CLASS_OPT`(`diracone`  
  - `print_func`< `print_dflt` > (&`diracone::do_print`), `print_func`< `print_latex` >(&`diracone`
- `GINAC_BIND_UNARCHIVER` (`clifford`)
- `GINAC_BIND_UNARCHIVER` (`cliffordunit`)
- `GINAC_BIND_UNARCHIVER` (`diracone`)
- `GINAC_BIND_UNARCHIVER` (`diracgamma`)
- `GINAC_BIND_UNARCHIVER` (`diracgamma5`)
- `GINAC_BIND_UNARCHIVER` (`diracgammaL`)
- `GINAC_BIND_UNARCHIVER` (`diracgammaR`)
- `static bool is_dirac_slash` (`const ex &seq0`)
- `static void base_and_index` (`const ex &c`, `ex &b`, `ex &i`)  
*This function decomposes  $\gamma \sim \mu \rightarrow (1, \mu)$  and  $a \rightarrow (a.ix, ix)$*
- `ex dirac_ONE` (`unsigned char rl=0`)  
*Create a Clifford unity object.*
- `static unsigned get_dim_uint` (`const ex &e`)
- `ex clifford_unit` (`const ex &mu`, `const ex &metr`, `unsigned char rl=0`)  
*Create a Clifford unit object.*
- `ex dirac_gamma` (`const ex &mu`, `unsigned char rl=0`)  
*Create a Dirac gamma object.*
- `ex dirac_gamma5` (`unsigned char rl=0`)  
*Create a Dirac gamma5 object.*
- `ex dirac_gammaL` (`unsigned char rl=0`)  
*Create a Dirac gammaL object.*
- `ex dirac_gammaR` (`unsigned char rl=0`)  
*Create a Dirac gammaR object.*
- `ex dirac_slash` (`const ex &e`, `const ex &dim`, `unsigned char rl=0`)  
*Create a term of the form  $e_\mu * \gamma \sim \mu$  with a unique index  $\mu$ .*
- `static unsigned char get_representation_label` (`const return_type_t &ti`)  
*Extract representation label from tinfo key (as returned by `return_type_tinfo()`).*
- `static ex trace_string` (`exvector::const_iterator ix`, `size_t num`)  
*Take trace of a string of an even number of Dirac gammas given a vector of indices.*
- `ex dirac_trace` (`const ex &e`, `const std::set< unsigned char > &rls`, `const ex &trONE=4`)  
*Calculate dirac traces over the specified set of representation labels.*
- `ex dirac_trace` (`const ex &e`, `const lst &rls`, `const ex &trONE=4`)

- Calculate dirac traces over the specified list of representation labels.*

  - [ex dirac\\_trace](#) (const [ex](#) &e, unsigned char rl=0, const [ex](#) &trONE=4)
- Calculate the trace of an expression containing gamma objects with a specified representation label.*

  - [ex canonicalize\\_clifford](#) (const [ex](#) &e)
- Bring all products of clifford objects in an expression into a canonical order.*

  - [ex clifford\\_star\\_bar](#) (const [ex](#) &e, bool do\_bar, unsigned [options](#))
- An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).*

  - [ex clifford\\_prime](#) (const [ex](#) &e)
- Automorphism of the Clifford algebra, simply changes signs of all clifford units.*

  - [ex remove\\_dirac\\_ONE](#) (const [ex](#) &e, unsigned char rl=0, unsigned [options](#)=0)
- Replaces dirac\_ONE's (with a representation\_label no less than rl) in e with 1.*

  - int [clifford\\_max\\_label](#) (const [ex](#) &e, bool ignore\_ONE=false)
- Returns the maximal representation label of a clifford object if e contains at least one, otherwise returns -1.*

  - [ex clifford\\_norm](#) (const [ex](#) &e)
- Calculation of the norm in the Clifford algebra.*

  - [ex clifford\\_inverse](#) (const [ex](#) &e)
- Calculation of the inverse in the Clifford algebra.*

  - [ex lst\\_to\\_clifford](#) (const [ex](#) &v, const [ex](#) &mu, const [ex](#) &metr, unsigned char rl=0)
- List or vector conversion into the Clifford vector.*

  - [ex lst\\_to\\_clifford](#) (const [ex](#) &v, const [ex](#) &e)
- static [ex get\\_clifford\\_comp](#) (const [ex](#) &e, const [ex](#) &c, bool root=true)*
- Auxiliary structure to define a function for stripping one Clifford unit from vectors.*

  - [lst clifford\\_to\\_lst](#) (const [ex](#) &e, const [ex](#) &c, bool algebraic=true)
- An inverse function to [lst\\_to\\_clifford\(\)](#).*

  - [ex clifford\\_moebius\\_map](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &c, const [ex](#) &d, const [ex](#) &v, const [ex](#) &G, unsigned char rl=0)
- Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b\c d) in linear spaces with arbitrary signature.*

  - [ex clifford\\_moebius\\_map](#) (const [ex](#) &M, const [ex](#) &v, const [ex](#) &G, unsigned char rl=0)
- The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([clifford](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([diracone](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([cliffordunit](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([diracgamma](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([diracgamma5](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([diracgammaL](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([diracgammaR](#))
- bool [is\\_clifford\\_tinfo](#) (const [return\\_type\\_t](#) &ti)
- Check whether a given [return\\_type\\_t](#) object (as returned by [return\\_type\\_tinfo\(\)](#)) is that of a clifford object (with an arbitrary representation label).*
- [ex clifford\\_bar](#) (const [ex](#) &e)
- Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.*
- [ex clifford\\_star](#) (const [ex](#) &e)
- Reversion of the Clifford algebra, reverse the order of all clifford units in ncmul.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([su3one](#), [tensor](#), print\_func< [print\\_dflt](#) >(&[su3one::do\\_print](#)), print\_func< [print\\_latex](#) >(&[su3one::do\\_print\\_latex](#))) [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([su3t](#), [PT](#)([su3t](#)), print\_func< [print\\_dflt](#) >(&[su3t::do\\_print](#)), print\_func< [print\\_latex](#) >(&[su3t](#)))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([color](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([su3one](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([su3t](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([su3f](#))

- [GINAC\\_BIND\\_UNARCHIVER](#) ([su3d](#))
- static [ex permute\\_free\\_index\\_to\\_front](#) (const [exvector](#) &iv3, const [exvector](#) &iv2, int &sig)  
*Given a vector iv3 of three indices and a vector iv2 of two indices that is a subset of iv3, return the (free) index that is in iv3 but not in iv2 and the sign introduced by permuting that index to the front.*
- [ex color\\_ONE](#) (unsigned char rl=0)  
*Create the su(3) unity element.*
- [ex color\\_T](#) (const [ex](#) &a, unsigned char rl=0)  
*Create an su(3) generator.*
- [ex color\\_f](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &c)  
*Create an su(3) antisymmetric structure constant.*
- [ex color\\_d](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &c)  
*Create an su(3) symmetric structure constant.*
- [ex color\\_h](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &c)  
*This returns the linear combination d.a.b.c+I\*f.a.b.c.*
- static bool [is\\_color\\_tinfo](#) (const [return\\_type\\_t](#) &ti)  
*Check whether a given tinfo key (as returned by [return\\_type\\_tinfo\(\)](#)) is that of a color object (with an arbitrary representation label).*
- static unsigned char [get\\_representation\\_label](#) (const [return\\_type\\_t](#) &ti)  
*Extract representation label from tinfo key (as returned by [return\\_type\\_tinfo\(\)](#)).*
- [ex color\\_trace](#) (const [ex](#) &e, const std::set< unsigned char > &rls)  
*Calculate color traces over the specified set of representation labels.*
- [ex color\\_trace](#) (const [ex](#) &e, const [lst](#) &rl)  
*Calculate color traces over the specified list of representation labels.*
- [ex color\\_trace](#) (const [ex](#) &e, unsigned char rl=0)  
*Calculate the trace of an expression containing color objects with a specified representation label.*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([color](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3one](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3t](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3f](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3d](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (constant, basic, print\_func< [print\\_context](#) >(&constant::do\_print). print\_func< [print\\_latex](#) >(&constant::do\_print\_latex). print\_func< [print\\_tree](#) >(&constant::do\_print\_tree). print\_func< [print\\_python\\_repr](#) >(&constant::do\_print\_python\_repr)) const ant
- [GINAC\\_BIND\\_UNARCHIVER](#) (constant)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (constant)
- static unsigned [crc32](#) (const char \*c, const unsigned [len](#), const unsigned [crcinit](#))
- bool [are\\_ex\\_trivially\\_equal](#) (const [ex](#) &e1, const [ex](#) &e2)  
*Compare two objects of class quickly without doing a deep tree traversal.*
- std::ostream & [operator<<](#) (std::ostream &os, const [exvector](#) &e)
- std::ostream & [operator<<](#) (std::ostream &os, const [exset](#) &e)
- std::ostream & [operator<<](#) (std::ostream &os, const [exmap](#) &e)
- size\_t [nops](#) (const [ex](#) &thisex)
- [ex expand](#) (const [ex](#) &thisex, unsigned [options](#)=0)
- [ex conjugate](#) (const [ex](#) &thisex)
- [ex real\\_part](#) (const [ex](#) &thisex)
- [ex imag\\_part](#) (const [ex](#) &thisex)
- bool [has](#) (const [ex](#) &thisex, const [ex](#) &pattern, unsigned [options](#)=0)
- bool [find](#) (const [ex](#) &thisex, const [ex](#) &pattern, [exset](#) &found)
- bool [is\\_polynomial](#) (const [ex](#) &thisex, const [ex](#) &vars)
- int [degree](#) (const [ex](#) &thisex, const [ex](#) &s)
- int [ldegree](#) (const [ex](#) &thisex, const [ex](#) &s)
- [ex coeff](#) (const [ex](#) &thisex, const [ex](#) &s, int n=1)
- [ex numer](#) (const [ex](#) &thisex)

- `ex denom` (const `ex` &thisex)
- `ex numer_denom` (const `ex` &thisex)
- `ex normal` (const `ex` &thisex)
- `ex to_rational` (const `ex` &thisex, `exmap` &repl)
- `ex to_polynomial` (const `ex` &thisex, `exmap` &repl)
- `ex collect` (const `ex` &thisex, const `ex` &s, bool distributed=false)
- `ex eval` (const `ex` &thisex)
- `ex evalf` (const `ex` &thisex)
- `ex evalm` (const `ex` &thisex)
- `ex eval_integ` (const `ex` &thisex)
- `ex diff` (const `ex` &thisex, const `symbol` &s, unsigned nth=1)
- `ex series` (const `ex` &thisex, const `ex` &r, int order, unsigned options=0)
- `bool match` (const `ex` &thisex, const `ex` &pattern, `exmap` &repl\_lst)
- `ex simplify_indexed` (const `ex` &thisex, unsigned options=0)
- `ex simplify_indexed` (const `ex` &thisex, const `scalar_products` &sp, unsigned options=0)
- `ex symmetrize` (const `ex` &thisex)
- `ex symmetrize` (const `ex` &thisex, const `lst` &l)
- `ex antisymmetrize` (const `ex` &thisex)
- `ex antisymmetrize` (const `ex` &thisex, const `lst` &l)
- `ex symmetrize_cyclic` (const `ex` &thisex)
- `ex symmetrize_cyclic` (const `ex` &thisex, const `lst` &l)
- `ex op` (const `ex` &thisex, size\_t i)
- `ex lhs` (const `ex` &thisex)
- `ex rhs` (const `ex` &thisex)
- `bool is_zero` (const `ex` &thisex)
- `void swap` (`ex` &e1, `ex` &e2)
- `ex subs` (const `ex` &thisex, const `exmap` &m, unsigned options=0)
- `ex subs` (const `ex` &thisex, const `lst` &ls, const `lst` &lr, unsigned options=0)
- `ex subs` (const `ex` &thisex, const `ex` &e, unsigned options=0)
- `template<class T >`  
`bool is_a` (const `ex` &obj)  
*Check if ex is a handle to a T, including base classes.*
- `template<class T >`  
`bool is_exactly_a` (const `ex` &obj)  
*Check if ex is a handle to a T, not including base classes.*
- `template<class T >`  
`const T & ex_to` (const `ex` &e)  
*Return a reference to the basic-derived class T object embedded in an expression.*
- `void compile_ex` (const `ex` &expr, const `symbol` &sym, `FUNCP_1P` &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- `void compile_ex` (const `ex` &expr, const `symbol` &sym1, const `symbol` &sym2, `FUNCP_2P` &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- `void compile_ex` (const `lst` &exprs, const `lst` &syms, `FUNCP_CUBA` &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- `void link_ex` (const std::string filename, `FUNCP_1P` &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_1P to the contained function.*
- `void link_ex` (const std::string filename, `FUNCP_2P` &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_2P to the contained function.*
- `void link_ex` (const std::string filename, `FUNCP_CUBA` &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_CUBA to the contained function.*
- `void unlink_ex` (const std::string filename)  
*Closes all linked .so files that have the supplied filename.*

- void [swap](#) ([expair](#) &e1, [expair](#) &e2)
- [epvector](#) \* [conjugateepvector](#) (const [epvector](#) &)  
*Complex conjugate every element of an epvector.*
- [ex factor](#) (const [ex](#) &[poly](#), unsigned [options](#))  
*Interface function to the outside world.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([fail](#), [basic](#), print\_func< [print\\_context](#) >(&[fail::do\\_print](#)), print\_func< [print\\_tree](#) >(&[fail::do\\_print\\_tree](#))) [GINAC\\_BIND\\_UNARCHIVER](#)([fail](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([fail](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([fderivative](#), [function](#), print\_func< [print\\_context](#) >(&[fderivative::do\\_print](#)), print\_func< [print\\_latex](#) >(&[fderivative::do\\_print\\_latex](#)), print\_func< [print\\_csrg](#) >(&[fderivative::do\\_print\\_csrg](#)), print\_func< [print\\_tree](#) >(&[fderivative::do\\_print\\_tree](#))) [fderivative](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([fderivative](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([fderivative](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([function](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([function](#))
- template<typename T >  
bool [is\\_the\\_function](#) (const [ex](#) &x)
- static unsigned [make\\_hash\\_seed](#) (const std::type\_info &tinfo)  
*We need a hash function which gives different values for objects of different types.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([idx](#), [basic](#), print\_func< [print\\_context](#) >(&[idx::do\\_print](#)), print\_func< [print\\_latex](#) >(&[idx::do\\_print\\_latex](#)), print\_func< [print\\_csrg](#) >(&[idx::do\\_print\\_csrg](#)), print\_func< [print\\_tree](#) >(&[idx::do\\_print\\_tree](#))) [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)([varidx](#)  
print\_func< [print\\_context](#) > (&[varidx::do\\_print](#)), print\_func< [print\\_latex](#) >(&[varidx](#)  
[GINAC\\_BIND\\_UNARCHIVER](#) ([idx](#))  
[GINAC\\_BIND\\_UNARCHIVER](#) ([varidx](#))  
[GINAC\\_BIND\\_UNARCHIVER](#) ([spinidx](#))
- bool [is\\_dummy\\_pair](#) (const [idx](#) &i1, const [idx](#) &i2)  
*Check whether two indices form a dummy pair.*
- bool [is\\_dummy\\_pair](#) (const [ex](#) &e1, const [ex](#) &e2)  
*Check whether two expressions form a dummy index pair.*
- void [find\\_free\\_and\\_dummy](#) ([exvector::const\\_iterator](#) it, [exvector::const\\_iterator](#) itend, [exvector](#) &out\_free, [exvector](#) &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- [ex minimal\\_dim](#) (const [ex](#) &dim1, const [ex](#) &dim2)  
*Return the minimum of two index dimensions.*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([idx](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([varidx](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([spinidx](#))
- void [find\\_free\\_and\\_dummy](#) (const [exvector](#) &v, [exvector](#) &out\_free, [exvector](#) &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- void [find\\_dummy\\_indices](#) (const [exvector](#) &v, [exvector](#) &out\_dummy)  
*Given a vector of indices, find the dummy indices.*
- size\_t [count\\_dummy\\_indices](#) (const [exvector](#) &v)  
*Count the number of dummy index pairs in an index vector.*
- size\_t [count\\_free\\_indices](#) (const [exvector](#) &v)  
*Count the number of dummy index pairs in an index vector.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([indexed](#), [exprseq](#), print\_func< [print\\_context](#) >(&[indexed::do\\_print](#)), print\_func< [print\\_latex](#) >(&[indexed::do\\_print\\_latex](#)), print\_func< [print\\_tree](#) >(&[indexed::do\\_print\\_tree](#))) [indexed](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([indexed](#))
- static bool [indices\\_consistent](#) (const [exvector](#) &v1, const [exvector](#) &v2)



*Check whether two sorted index vectors are consistent (i.e.*

- `template<class T >`  
`size_t number_of_type` (const `exvector` &`v`)
- `template<class T >`  
`static ex rename_dummy_indices` (const `ex` &`e`, `exvector` &`global_dummy_indices`, `exvector` &`local_dummy_`↵  
`_indices`)  
*Rename dummy indices in an expression.*
- `static void find_variant_indices` (const `exvector` &`v`, `exvector` &`variant_indices`)  
*Given a set of indices, extract those of class `varidx`.*
- `bool reposition_dummy_indices` (`ex` &`e`, `exvector` &`variant_dummy_indices`, `exvector` &`moved_indices`)  
*Raise/lower dummy indices in a single indexed objects to canonicalize their variance.*
- `static void product_to_exvector` (const `ex` &`e`, `exvector` &`v`, bool &`non_commutative`)
- `template<class T >`  
`ex idx_symmetrization` (const `ex` &`r`, const `exvector` &`local_dummy_indices`)
- `ex simplify_indexed` (const `ex` &`e`, `exvector` &`free_indices`, `exvector` &`dummy_indices`, const `scalar_products` &`sp`)  
*Simplify indexed expression, return list of free indices.*
- `ex simplify_indexed_product` (const `ex` &`e`, `exvector` &`free_indices`, `exvector` &`dummy_indices`, const `scalar_products` &`sp`)  
*Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.*
- `bool hasindex` (const `ex` &`x`, const `ex` &`sym`)
- `exvector get_all_dummy_indices_safely` (const `ex` &`e`)  
*More reliable version of the form.*
- `exvector get_all_dummy_indices` (const `ex` &`e`)  
*Returns all dummy indices from the `exvector`.*
- `lst rename_dummy_indices_uniquely` (const `exvector` &`va`, const `exvector` &`vb`)  
*Similar to above, where `va` and `vb` are the same and the return value is a list of two lists for substitution in `b`.*
- `ex rename_dummy_indices_uniquely` (const `exvector` &`va`, const `exvector` &`vb`, const `ex` &`b`)  
*Same as above, where `va` and `vb` contain the indices of `a` and `b` and are sorted.*
- `ex rename_dummy_indices_uniquely` (const `ex` &`a`, const `ex` &`b`)  
*Returns `b` with all dummy indices, which are common with `a`, renamed.*
- `ex rename_dummy_indices_uniquely` (`exvector` &`va`, const `ex` &`b`, bool `modify_va=false`)  
*Returns `b` with all dummy indices, which are listed in `va`, renamed if `modify_va` is set to `TRUE` all dummy indices of `b` will be appended to `va`.*
- `ex expand_dummy_sum` (const `ex` &`e`, bool `subs_idx=false`)  
*This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.*
- `GINAC_DECLARE_UNARCHIVER` (indexed)
- `static ex conjugate_evalf` (const `ex` &`arg`)
- `static ex conjugate_eval` (const `ex` &`arg`)
- `static void conjugate_print_latex` (const `ex` &`arg`, const `print_context` &`c`)
- `static ex conjugate_conjugate` (const `ex` &`arg`)
- `static ex conjugate_expl_derivative` (const `ex` &`arg`, const `symbol` &`s`)
- `static ex conjugate_real_part` (const `ex` &`arg`)
- `static ex conjugate_imag_part` (const `ex` &`arg`)
- `static bool func_arg_info` (const `ex` &`arg`, unsigned `inf`)
- `static bool conjugate_info` (const `ex` &`arg`, unsigned `inf`)
- `REGISTER_FUNCTION` (`conjugate_function`, `eval_func(conjugate_eval)`. `evalf_func(conjugate_evalf)`.  
`expl_derivative_func(conjugate_expl_derivative)`. `info_func(conjugate_info)`. `print_func<` `print_latex`  
`>(conjugate_print_latex)`. `conjugate_func(conjugate_conjugate)`. `real_part_func(conjugate_real_part)`.  
`imag_part_func(conjugate_imag_part)`. `set_name("conjugate","conjugate")`)
- `static ex real_part_evalf` (const `ex` &`arg`)
- `static ex real_part_eval` (const `ex` &`arg`)



- static void `real_part_print_latex` (const `ex` &arg, const `print_context` &c)
- static `ex` `real_part_conjugate` (const `ex` &arg)
- static `ex` `real_part_real_part` (const `ex` &arg)
- static `ex` `real_part_imag_part` (const `ex` &arg)
- static `ex` `real_part_expl_derivative` (const `ex` &arg, const `symbol` &s)
- `REGISTER_FUNCTION` (`real_part_function`, `eval_func`(`real_part_eval`). `evalf_func`(`real_part_evalf`). `expl_←_derivative_func`(`real_part_expl_derivative`). `print_func`< `print_latex` >( `real_part_print_latex`). `conjugate_←_func`(`real_part_conjugate`). `real_part_func`(`real_part_real_part`). `imag_part_func`(`real_part_imag_part`). `set_name`("real\_part", "real\_part"))
- static `ex` `imag_part_evalf` (const `ex` &arg)
- static `ex` `imag_part_eval` (const `ex` &arg)
- static void `imag_part_print_latex` (const `ex` &arg, const `print_context` &c)
- static `ex` `imag_part_conjugate` (const `ex` &arg)
- static `ex` `imag_part_real_part` (const `ex` &arg)
- static `ex` `imag_part_imag_part` (const `ex` &arg)
- static `ex` `imag_part_expl_derivative` (const `ex` &arg, const `symbol` &s)
- `REGISTER_FUNCTION` (`imag_part_function`, `eval_func`(`imag_part_eval`). `evalf_func`(`imag_part_evalf`). `expl_derivative_func`(`imag_part_expl_derivative`). `print_func`< `print_latex` >( `imag_part_print_latex`). `conjugate_func`(`imag_part_conjugate`). `real_part_func`(`imag_part_real_part`). `imag_part_func`(`imag_part_imag_part`). `set_name`("imag\_part", "imag\_part"))
- static `ex` `abs_evalf` (const `ex` &arg)
- static `ex` `abs_eval` (const `ex` &arg)
- static `ex` `abs_expand` (const `ex` &arg, unsigned `options`)
- static `ex` `abs_expl_derivative` (const `ex` &arg, const `symbol` &s)
- static void `abs_print_latex` (const `ex` &arg, const `print_context` &c)
- static void `abs_print_csrc_float` (const `ex` &arg, const `print_context` &c)
- static `ex` `abs_conjugate` (const `ex` &arg)
- static `ex` `abs_real_part` (const `ex` &arg)
- static `ex` `abs_imag_part` (const `ex` &arg)
- static `ex` `abs_power` (const `ex` &arg, const `ex` &exp)
- bool `abs_info` (const `ex` &arg, unsigned inf)
- `REGISTER_FUNCTION` (`abs`, `eval_func`(`abs_eval`). `evalf_func`(`abs_evalf`). `expand_func`(`abs_expand`). `expl_derivative_func`(`abs_expl_derivative`). `info_func`(`abs_info`). `print_func`< `print_latex` >( `abs_print_latex`). `print_func`< `print_csrc_float` >( `abs_print_csrc_float`). `print_func`< `print_csrc_double` >( `abs_print_csrc_float`). `conjugate_func`(`abs_conjugate`). `real_part_func`(`abs_real_part`). `imag_part_func`(`abs_imag_part`). `power_←_func`(`abs_power`))
- static `ex` `step_evalf` (const `ex` &arg)
- static `ex` `step_eval` (const `ex` &arg)
- static `ex` `step_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex` `step_conjugate` (const `ex` &arg)
- static `ex` `step_real_part` (const `ex` &arg)
- static `ex` `step_imag_part` (const `ex` &arg)
- `REGISTER_FUNCTION` (`step`, `eval_func`(`step_eval`). `evalf_func`(`step_evalf`). `series_func`(`step_series`). `conjugate_func`(`step_conjugate`). `real_part_func`(`step_real_part`). `imag_part_func`(`step_imag_part`))
- static `ex` `csgn_evalf` (const `ex` &arg)
- static `ex` `csgn_eval` (const `ex` &arg)
- static `ex` `csgn_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex` `csgn_conjugate` (const `ex` &arg)
- static `ex` `csgn_real_part` (const `ex` &arg)
- static `ex` `csgn_imag_part` (const `ex` &arg)
- static `ex` `csgn_power` (const `ex` &arg, const `ex` &exp)
- `REGISTER_FUNCTION` (`csgn`, `eval_func`(`csgn_eval`). `evalf_func`(`csgn_evalf`). `series_func`(`csgn_series`). `conjugate_func`(`csgn_conjugate`). `real_part_func`(`csgn_real_part`). `imag_part_func`(`csgn_imag_part`). `power_func`(`csgn_power`))
- static `ex` `eta_evalf` (const `ex` &x, const `ex` &y)

- static `ex eta_eval` (const `ex` &`x`, const `ex` &`y`)
  - static `ex eta_series` (const `ex` &`x`, const `ex` &`y`, const `relational` &`rel`, int `order`, unsigned `options`)
  - static `ex eta_conjugate` (const `ex` &`x`, const `ex` &`y`)
  - static `ex eta_real_part` (const `ex` &`x`, const `ex` &`y`)
  - static `ex eta_imag_part` (const `ex` &`x`, const `ex` &`y`)
  - `REGISTER_FUNCTION` (`eta`, `eval_func(eta_eval)`. `evalf_func(eta_evalf)`. `series_func(eta_series)`. `latex_name`("\eta"). `set_symmetry(sy_symm(0, 1))`. `conjugate_func(eta_conjugate)`. `real_part_func(eta_real_part)`. `imag_part_func(eta_imag_part)`)
  - static `ex Li2_evalf` (const `ex` &`x`)
  - static `ex Li2_eval` (const `ex` &`x`)
  - static `ex Li2_deriv` (const `ex` &`x`, unsigned `deriv_param`)
  - static `ex Li2_series` (const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
  - static `ex Li2_conjugate` (const `ex` &`x`)
  - `REGISTER_FUNCTION` (`Li2`, `eval_func(Li2_eval)`. `evalf_func(Li2_evalf)`. `derivative_func(Li2_deriv)`. `series_func(Li2_series)`. `conjugate_func(Li2_conjugate)`. `latex_name`("\mathrm{Li}\_2"))
  - static `ex Li3_eval` (const `ex` &`x`)
  - `REGISTER_FUNCTION` (`Li3`, `eval_func(Li3_eval)`. `latex_name`("\mathrm{Li}\_3"))
  - static `ex zetaderiv_eval` (const `ex` &`n`, const `ex` &`x`)
  - static `ex zetaderiv_deriv` (const `ex` &`n`, const `ex` &`x`, unsigned `deriv_param`)
  - `REGISTER_FUNCTION` (`zetaderiv`, `eval_func(zetaderiv_eval)`. `derivative_func(zetaderiv_deriv)`. `latex_name`("eta^\mathrm{rime}"))
  - static `ex factorial_evalf` (const `ex` &`x`)
  - static `ex factorial_eval` (const `ex` &`x`)
  - static void `factorial_print_dflt_latex` (const `ex` &`x`, const `print_context` &`c`)
  - static `ex factorial_conjugate` (const `ex` &`x`)
  - static `ex factorial_real_part` (const `ex` &`x`)
  - static `ex factorial_imag_part` (const `ex` &`x`)
  - `REGISTER_FUNCTION` (`factorial`, `eval_func(factorial_eval)`. `evalf_func(factorial_evalf)`. `print_func`<`print_dflt`>(`factorial_print_dflt_latex`). `print_func`<`print_latex`>(`factorial_print_dflt_latex`). `conjugate_func`<`factorial_conjugate`>(`factorial_conjugate`). `real_part_func`<`factorial_real_part`>(`factorial_real_part`). `imag_part_func`<`factorial_imag_part`>(`factorial_imag_part`))
  - static `ex binomial_evalf` (const `ex` &`x`, const `ex` &`y`)
  - static `ex binomial_sym` (const `ex` &`x`, const `numeric` &`y`)
  - static `ex binomial_eval` (const `ex` &`x`, const `ex` &`y`)
  - static `ex binomial_conjugate` (const `ex` &`x`, const `ex` &`y`)
  - static `ex binomial_real_part` (const `ex` &`x`, const `ex` &`y`)
  - static `ex binomial_imag_part` (const `ex` &`x`, const `ex` &`y`)
  - `REGISTER_FUNCTION` (`binomial`, `eval_func(binomial_eval)`. `evalf_func(binomial_evalf)`. `conjugate_func`<`binomial_conjugate`>(`binomial_conjugate`). `real_part_func`<`binomial_real_part`>(`binomial_real_part`). `imag_part_func`<`binomial_imag_part`>(`binomial_imag_part`))
  - static `ex Order_eval` (const `ex` &`x`)
  - static `ex Order_series` (const `ex` &`x`, const `relational` &`r`, int `order`, unsigned `options`)
  - static `ex Order_conjugate` (const `ex` &`x`)
  - static `ex Order_real_part` (const `ex` &`x`)
  - static `ex Order_imag_part` (const `ex` &`x`)
  - static `ex Order_expl_derivative` (const `ex` &`arg`, const `symbol` &`s`)
  - `REGISTER_FUNCTION` (`Order`, `eval_func(Order_eval)`. `series_func(Order_series)`. `latex_name`("\mathrm{O}"). `expl_derivative_func`<`Order_expl_derivative`>(`Order_expl_derivative`). `conjugate_func`<`Order_conjugate`>(`Order_conjugate`). `real_part_func`<`Order_real_part`>(`Order_real_part`). `imag_part_func`<`Order_imag_part`>(`Order_imag_part`))
  - `ex Isolve` (const `ex` &`eqns`, const `ex` &`symbols`, unsigned `options=solve_algo::automatic`)
- Factorial function.*
- const `numeric fsolve` (const `ex` &`f`, const `symbol` &`x`, const `numeric` &`x1`, const `numeric` &`x2`)
- Find a real root of real-valued function f(x) numerically within a given interval.*
- template<typename T1 >  
  `function zeta` (const T1 &p1)
  - template<typename T1 , typename T2 >  
  `function zeta` (const T1 &p1, const T2 &p2)

- `template<>`  
`bool is_the_function< zeta_SERIAL > (const ex &x)`
- `template<typename T1 , typename T2 >`  
`function G (const T1 &x, const T2 &y)`
- `template<typename T1 , typename T2 , typename T3 >`  
`function G (const T1 &x, const T2 &s, const T3 &y)`
- `template<>`  
`bool is_the_function< G_SERIAL > (const ex &x)`
- `template<typename T1 >`  
`function psi (const T1 &p1)`
- `template<typename T1 , typename T2 >`  
`function psi (const T1 &p1, const T2 &p2)`
- `template<>`  
`bool is_the_function< psi_SERIAL > (const ex &x)`
- `template<typename T1 , typename T2 >`  
`function iterated_integral (const T1 &kernel_lst, const T2 &lambda)`
- `template<typename T1 , typename T2 , typename T3 >`  
`function iterated_integral (const T1 &kernel_lst, const T2 &lambda, const T3 &N_trunc)`
- `template<>`  
`bool is_the_function< iterated_integral_SERIAL > (const ex &x)`
- `bool is_order_function (const ex &e)`  
*Check whether a function is the Order (O(n)) function.*
- `ex convert_H_to_Li (const ex &parameterlst, const ex &arg)`  
*Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding GiNaC functions.*
- `static ex EllipticK_evalf (const ex &k)`
- `static ex EllipticK_eval (const ex &k)`
- `static ex EllipticK_deriv (const ex &k, unsigned deriv_param)`
- `static ex EllipticK_series (const ex &k, const relational &rel, int order, unsigned options)`
- `static void EllipticK_print_latex (const ex &k, const print_context &c)`
- `REGISTER_FUNCTION (EllipticK, evalf_func(EllipticK_evalf). eval_func(EllipticK_eval). derivative_↔  
func(EllipticK_deriv). series_func(EllipticK_series). print_func< print_latex >(EllipticK_print_latex). do_↔  
not_evalf_params())`
- `static ex EllipticE_evalf (const ex &k)`
- `static ex EllipticE_eval (const ex &k)`
- `static ex EllipticE_deriv (const ex &k, unsigned deriv_param)`
- `static ex EllipticE_series (const ex &k, const relational &rel, int order, unsigned options)`
- `static void EllipticE_print_latex (const ex &k, const print_context &c)`
- `REGISTER_FUNCTION (EllipticE, evalf_func(EllipticE_evalf). eval_func(EllipticE_eval). derivative_↔  
func(EllipticE_deriv). series_func(EllipticE_series). print_func< print_latex >(EllipticE_print_latex). do_↔  
not_evalf_params())`
- `static ex iterated_integral_evalf_impl (const ex &kernel_lst, const ex &lambda, const ex &N_trunc)`
- `static ex iterated_integral2_evalf (const ex &kernel_lst, const ex &lambda)`
- `static ex iterated_integral3_evalf (const ex &kernel_lst, const ex &lambda, const ex &N_trunc)`
- `static ex iterated_integral2_eval (const ex &kernel_lst, const ex &lambda)`
- `static ex iterated_integral3_eval (const ex &kernel_lst, const ex &lambda, const ex &N_trunc)`
- `static ex lgamma_evalf (const ex &x)`
- `static ex lgamma_eval (const ex &x)`  
*Evaluation of  $\lgamma(x)$ , the natural logarithm of the Gamma function.*
- `static ex lgamma_deriv (const ex &x, unsigned deriv_param)`
- `static ex lgamma_series (const ex &arg, const relational &rel, int order, unsigned options)`
- `static ex lgamma_conjugate (const ex &x)`
- `REGISTER_FUNCTION (lgamma, evalf_func(lgamma_evalf). evalf_func(lgamma_evalf). derivative_↔  
func(lgamma_deriv). series_func(lgamma_series). conjugate_func(lgamma_conjugate). latex_name("log  
\gamma"))`

- static `ex` `tgamma_evalf` (const `ex` &`x`)
- static `ex` `tgamma_eval` (const `ex` &`x`)

*Evaluation of  $\text{tgamma}(x)$ , the true Gamma function.*

- static `ex` `tgamma_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `tgamma_series` (const `ex` &`arg`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `tgamma_conjugate` (const `ex` &`x`)
- `REGISTER_FUNCTION` (`tgamma`, `eval_func(tgamma_eval)`. `evalf_func(tgamma_evalf)`. `derivative_func(tgamma_deriv)`. `series_func(tgamma_series)`. `conjugate_func(tgamma_conjugate)`. `latex_name("amma")`)
- static `ex` `beta_evalf` (const `ex` &`x`, const `ex` &`y`)
- static `ex` `beta_eval` (const `ex` &`x`, const `ex` &`y`)
- static `ex` `beta_deriv` (const `ex` &`x`, const `ex` &`y`, unsigned `deriv_param`)
- static `ex` `beta_series` (const `ex` &`arg1`, const `ex` &`arg2`, const `relational` &`rel`, int `order`, unsigned `options`)
- `REGISTER_FUNCTION` (`beta`, `eval_func(beta_eval)`. `evalf_func(beta_evalf)`. `derivative_func(beta_deriv)`. `series_func(beta_series)`. `latex_name("\mathrm{B}")`. `set_symmetry(sy_symm(0, 1))`)
- static `ex` `psi1_evalf` (const `ex` &`x`)
- static `ex` `psi1_eval` (const `ex` &`x`)

*Evaluation of digamma-function  $\text{psi}(x)$ .*

- static `ex` `psi1_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `psi1_series` (const `ex` &`arg`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `psi2_evalf` (const `ex` &`n`, const `ex` &`x`)
- static `ex` `psi2_eval` (const `ex` &`n`, const `ex` &`x`)

*Evaluation of polygamma-function  $\text{psi}(n,x)$ .*

- static `ex` `psi2_deriv` (const `ex` &`n`, const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `psi2_series` (const `ex` &`n`, const `ex` &`arg`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `G2_evalf` (const `ex` &`x_`, const `ex` &`y`)
- static `ex` `G2_eval` (const `ex` &`x_`, const `ex` &`y`)
- static `ex` `G3_evalf` (const `ex` &`x_`, const `ex` &`s_`, const `ex` &`y`)
- static `ex` `G3_eval` (const `ex` &`x_`, const `ex` &`s_`, const `ex` &`y`)
- static `ex` `Li_evalf` (const `ex` &`m_`, const `ex` &`x_`)
- static `ex` `Li_eval` (const `ex` &`m_`, const `ex` &`x_`)
- static `ex` `Li_series` (const `ex` &`m`, const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `Li_deriv` (const `ex` &`m_`, const `ex` &`x_`, unsigned `deriv_param`)
- static void `Li_print_latex` (const `ex` &`m_`, const `ex` &`x_`, const `print_context` &`c`)
- `REGISTER_FUNCTION` (`Li`, `evalf_func(Li_evalf)`. `eval_func(Li_eval)`. `series_func(Li_series)`. `derivative_func(Li_deriv)`. `print_func< print_latex >(Li_print_latex)`. `do_not_evalf_params()`)
- static `ex` `S_evalf` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`)
- static `ex` `S_eval` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`)
- static `ex` `S_series` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `S_deriv` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`, unsigned `deriv_param`)
- static void `S_print_latex` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`, const `print_context` &`c`)
- `REGISTER_FUNCTION` (`S`, `evalf_func(S_evalf)`. `eval_func(S_eval)`. `series_func(S_series)`. `derivative_func(S_deriv)`. `print_func< print_latex >(S_print_latex)`. `do_not_evalf_params()`)
- static `ex` `H_evalf` (const `ex` &`x1`, const `ex` &`x2`)
- static `ex` `H_eval` (const `ex` &`m_`, const `ex` &`x`)
- static `ex` `H_series` (const `ex` &`m`, const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `H_deriv` (const `ex` &`m_`, const `ex` &`x`, unsigned `deriv_param`)
- static void `H_print_latex` (const `ex` &`m_`, const `ex` &`x`, const `print_context` &`c`)
- `REGISTER_FUNCTION` (`H`, `evalf_func(H_evalf)`. `eval_func(H_eval)`. `series_func(H_series)`. `derivative_func(H_deriv)`. `print_func< print_latex >(H_print_latex)`. `do_not_evalf_params()`)
- static `ex` `zeta1_evalf` (const `ex` &`x`)
- static `ex` `zeta1_eval` (const `ex` &`m`)
- static `ex` `zeta1_deriv` (const `ex` &`m`, unsigned `deriv_param`)
- static void `zeta1_print_latex` (const `ex` &`m_`, const `print_context` &`c`)

- static [ex](#) [zeta2\\_evalf](#) (const [ex](#) &x, const [ex](#) &s)
- static [ex](#) [zeta2\\_eval](#) (const [ex](#) &m, const [ex](#) &s\_)
- static [ex](#) [zeta2\\_deriv](#) (const [ex](#) &m, const [ex](#) &s, unsigned deriv\_param)
- static void [zeta2\\_print\\_latex](#) (const [ex](#) &m\_, const [ex](#) &s\_, const [print\\_context](#) &c)
- static [ex](#) [exp\\_evalf](#) (const [ex](#) &x)
- static [ex](#) [exp\\_eval](#) (const [ex](#) &x)
- static [ex](#) [exp\\_expand](#) (const [ex](#) &arg, unsigned options)
- static [ex](#) [exp\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex](#) [exp\\_real\\_part](#) (const [ex](#) &x)
- static [ex](#) [exp\\_imag\\_part](#) (const [ex](#) &x)
- static [ex](#) [exp\\_conjugate](#) (const [ex](#) &x)
- static [ex](#) [exp\\_power](#) (const [ex](#) &x, const [ex](#) &a)
- [REGISTER\\_FUNCTION](#) ([exp](#), eval\_func([exp\\_eval](#)). evalf\_func([exp\\_evalf](#)). expand\_func([exp\\_expand](#)). derivative\_func([exp\\_deriv](#)). real\_part\_func([exp\\_real\\_part](#)). imag\_part\_func([exp\\_imag\\_part](#)). conjugate\_func([exp\\_conjugate](#)). power\_func([exp\\_power](#)). latex\_name("xp"))
- static [ex](#) [log\\_evalf](#) (const [ex](#) &x)
- static [ex](#) [log\\_eval](#) (const [ex](#) &x)
- static [ex](#) [log\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex](#) [log\\_series](#) (const [ex](#) &arg, const [relational](#) &rel, int order, unsigned options)
- static [ex](#) [log\\_real\\_part](#) (const [ex](#) &x)
- static [ex](#) [log\\_imag\\_part](#) (const [ex](#) &x)
- static [ex](#) [log\\_expand](#) (const [ex](#) &arg, unsigned options)
- static [ex](#) [log\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) ([log](#), eval\_func([log\\_eval](#)). evalf\_func([log\\_evalf](#)). expand\_func([log\\_expand](#)). derivative\_func([log\\_deriv](#)). series\_func([log\\_series](#)). real\_part\_func([log\\_real\\_part](#)). imag\_part\_func([log\\_imag\\_part](#)). conjugate\_func([log\\_conjugate](#)). latex\_name("\n"))
- static [ex](#) [sin\\_evalf](#) (const [ex](#) &x)
- static [ex](#) [sin\\_eval](#) (const [ex](#) &x)
- static [ex](#) [sin\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex](#) [sin\\_real\\_part](#) (const [ex](#) &x)
- static [ex](#) [sin\\_imag\\_part](#) (const [ex](#) &x)
- static [ex](#) [sin\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) ([sin](#), eval\_func([sin\\_eval](#)). evalf\_func([sin\\_evalf](#)). derivative\_func([sin\\_deriv](#)). real\_part\_func([sin\\_real\\_part](#)). imag\_part\_func([sin\\_imag\\_part](#)). conjugate\_func([sin\\_conjugate](#)). latex\_name("\sin"))
- static [ex](#) [cos\\_evalf](#) (const [ex](#) &x)
- static [ex](#) [cos\\_eval](#) (const [ex](#) &x)
- static [ex](#) [cos\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex](#) [cos\\_real\\_part](#) (const [ex](#) &x)
- static [ex](#) [cos\\_imag\\_part](#) (const [ex](#) &x)
- static [ex](#) [cos\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) ([cos](#), eval\_func([cos\\_eval](#)). evalf\_func([cos\\_evalf](#)). derivative\_func([cos\\_deriv](#)). real\_part\_func([cos\\_real\\_part](#)). imag\_part\_func([cos\\_imag\\_part](#)). conjugate\_func([cos\\_conjugate](#)). latex\_name("\cos"))
- static [ex](#) [tan\\_evalf](#) (const [ex](#) &x)
- static [ex](#) [tan\\_eval](#) (const [ex](#) &x)
- static [ex](#) [tan\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex](#) [tan\\_real\\_part](#) (const [ex](#) &x)
- static [ex](#) [tan\\_imag\\_part](#) (const [ex](#) &x)
- static [ex](#) [tan\\_series](#) (const [ex](#) &x, const [relational](#) &rel, int order, unsigned options)
- static [ex](#) [tan\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) ([tan](#), eval\_func([tan\\_eval](#)). evalf\_func([tan\\_evalf](#)). derivative\_func([tan\\_deriv](#)). series\_func([tan\\_series](#)). real\_part\_func([tan\\_real\\_part](#)). imag\_part\_func([tan\\_imag\\_part](#)). conjugate\_func([tan\\_conjugate](#)). latex\_name("\tan"))
- static [ex](#) [asin\\_evalf](#) (const [ex](#) &x)

- static [ex asin\\_eval](#) (const [ex](#) &[x](#))
- static [ex asin\\_deriv](#) (const [ex](#) &[x](#), unsigned deriv\_param)
- static [ex asin\\_conjugate](#) (const [ex](#) &[x](#))
- [REGISTER\\_FUNCTION](#) ([asin](#), eval\_func([asin\\_eval](#)). evalf\_func([asin\\_evalf](#)). derivative\_func([asin\\_deriv](#)). conjugate\_func([asin\\_conjugate](#)). latex\_name("rcsin"))
- static [ex acos\\_evalf](#) (const [ex](#) &[x](#))
- static [ex acos\\_eval](#) (const [ex](#) &[x](#))
- static [ex acos\\_deriv](#) (const [ex](#) &[x](#), unsigned deriv\_param)
- static [ex acos\\_conjugate](#) (const [ex](#) &[x](#))
- [REGISTER\\_FUNCTION](#) ([acos](#), eval\_func([acos\\_eval](#)). evalf\_func([acos\\_evalf](#)). derivative\_func([acos\\_deriv](#)). conjugate\_func([acos\\_conjugate](#)). latex\_name("rccos"))
- static [ex atan\\_evalf](#) (const [ex](#) &[x](#))
- static [ex atan\\_eval](#) (const [ex](#) &[x](#))
- static [ex atan\\_deriv](#) (const [ex](#) &[x](#), unsigned deriv\_param)
- static [ex atan\\_series](#) (const [ex](#) &arg, const [relational](#) &rel, int [order](#), unsigned [options](#))
- static [ex atan\\_conjugate](#) (const [ex](#) &[x](#))
- [REGISTER\\_FUNCTION](#) ([atan](#), eval\_func([atan\\_eval](#)). evalf\_func([atan\\_evalf](#)). derivative\_func([atan\\_deriv](#)). series\_func([atan\\_series](#)). conjugate\_func([atan\\_conjugate](#)). latex\_name("rctan"))
- static [ex atan2\\_evalf](#) (const [ex](#) &y, const [ex](#) &[x](#))
- static [ex atan2\\_eval](#) (const [ex](#) &y, const [ex](#) &[x](#))
- static [ex atan2\\_deriv](#) (const [ex](#) &y, const [ex](#) &[x](#), unsigned deriv\_param)
- [REGISTER\\_FUNCTION](#) ([atan2](#), eval\_func([atan2\\_eval](#)). evalf\_func([atan2\\_evalf](#)). derivative\_func([atan2\\_deriv](#)))
- static [ex sinh\\_evalf](#) (const [ex](#) &[x](#))
- static [ex sinh\\_eval](#) (const [ex](#) &[x](#))
- static [ex sinh\\_deriv](#) (const [ex](#) &[x](#), unsigned deriv\_param)
- static [ex sinh\\_real\\_part](#) (const [ex](#) &[x](#))
- static [ex sinh\\_imag\\_part](#) (const [ex](#) &[x](#))
- static [ex sinh\\_conjugate](#) (const [ex](#) &[x](#))
- [REGISTER\\_FUNCTION](#) ([sinh](#), eval\_func([sinh\\_eval](#)). evalf\_func([sinh\\_evalf](#)). derivative\_func([sinh\\_deriv](#)). real\_part\_func([sinh\\_real\\_part](#)). imag\_part\_func([sinh\\_imag\\_part](#)). conjugate\_func([sinh\\_conjugate](#)). latex\_name("inh"))
- static [ex cosh\\_evalf](#) (const [ex](#) &[x](#))
- static [ex cosh\\_eval](#) (const [ex](#) &[x](#))
- static [ex cosh\\_deriv](#) (const [ex](#) &[x](#), unsigned deriv\_param)
- static [ex cosh\\_real\\_part](#) (const [ex](#) &[x](#))
- static [ex cosh\\_imag\\_part](#) (const [ex](#) &[x](#))
- static [ex cosh\\_conjugate](#) (const [ex](#) &[x](#))
- [REGISTER\\_FUNCTION](#) ([cosh](#), eval\_func([cosh\\_eval](#)). evalf\_func([cosh\\_evalf](#)). derivative\_func([cosh\\_deriv](#)). real\_part\_func([cosh\\_real\\_part](#)). imag\_part\_func([cosh\\_imag\\_part](#)). conjugate\_func([cosh\\_conjugate](#)). latex\_name("osh"))
- static [ex tanh\\_evalf](#) (const [ex](#) &[x](#))
- static [ex tanh\\_eval](#) (const [ex](#) &[x](#))
- static [ex tanh\\_deriv](#) (const [ex](#) &[x](#), unsigned deriv\_param)
- static [ex tanh\\_series](#) (const [ex](#) &[x](#), const [relational](#) &rel, int [order](#), unsigned [options](#))
- static [ex tanh\\_real\\_part](#) (const [ex](#) &[x](#))
- static [ex tanh\\_imag\\_part](#) (const [ex](#) &[x](#))
- static [ex tanh\\_conjugate](#) (const [ex](#) &[x](#))
- [REGISTER\\_FUNCTION](#) ([tanh](#), eval\_func([tanh\\_eval](#)). evalf\_func([tanh\\_evalf](#)). derivative\_func([tanh\\_deriv](#)). series\_func([tanh\\_series](#)). real\_part\_func([tanh\\_real\\_part](#)). imag\_part\_func([tanh\\_imag\\_part](#)). conjugate\_func([tanh\\_conjugate](#)). latex\_name("anh"))
- static [ex asinh\\_evalf](#) (const [ex](#) &[x](#))
- static [ex asinh\\_eval](#) (const [ex](#) &[x](#))
- static [ex asinh\\_deriv](#) (const [ex](#) &[x](#), unsigned deriv\_param)
- static [ex asinh\\_conjugate](#) (const [ex](#) &[x](#))



- `REGISTER_FUNCTION` (`asinh`, `eval_func(asinh_eval)`, `evalf_func(asinh_evalf)`, `derivative_func(asinh_deriv)`, `conjugate_func(asinh_conjugate)`)
- static `ex acosh_evalf` (const `ex` &`x`)
- static `ex acosh_eval` (const `ex` &`x`)
- static `ex acosh_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex acosh_conjugate` (const `ex` &`x`)
- `REGISTER_FUNCTION` (`acosh`, `eval_func(acosh_eval)`, `evalf_func(acosh_evalf)`, `derivative_func(acosh_deriv)`, `conjugate_func(acosh_conjugate)`)
- static `ex atanh_evalf` (const `ex` &`x`)
- static `ex atanh_eval` (const `ex` &`x`)
- static `ex atanh_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex atanh_series` (const `ex` &`arg`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex atanh_conjugate` (const `ex` &`x`)
- `REGISTER_FUNCTION` (`atanh`, `eval_func(atanh_eval)`, `evalf_func(atanh_evalf)`, `derivative_func(atanh_deriv)`, `series_func(atanh_series)`, `conjugate_func(atanh_conjugate)`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`integral`, `basic`, `print_func< print_dflt >(&integral::do_print)`, `print_func< print_python >(&integral::do_print)`, `print_func< print_latex >(&integral::do_print_latex)`) `integral`
- `ex subsvalue` (const `ex` &`var`, const `ex` &`value`, const `ex` &`fun`)
- `ex adaptivesimpson` (const `ex` &`x`, const `ex` &`a_in`, const `ex` &`b_in`, const `ex` &`f`, const `ex` &`error`)  
*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*
- `GINAC_BIND_UNARCHIVER` (`integral`)
- `GINAC_DECLARE_UNARCHIVER` (`integral`)
- `ex ifactor` (const `numeric` &`n`)  
*Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $lst( lst(p1,...,pr), lst(a1,...,ar) )$  such that  $n = p1^{a1} \dots$*
- `bool is_discriminant_of_quadratic_number_field` (const `numeric` &`n`)  
*Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.*
- `numeric kronecker_symbol` (const `numeric` &`a`, const `numeric` &`n`)  
*Returns the Kronecker symbol  $a$ : integer  $n$ : integer.*
- `numeric primitive_dirichlet_character` (const `numeric` &`n`, const `numeric` &`a`)  
*Defines a primitive Dirichlet character through the Kronecker symbol.*
- `numeric dirichlet_character` (const `numeric` &`n`, const `numeric` &`a`, const `numeric` &`N`)  
*Defines a Dirichlet character through the Kronecker symbol.*
- `numeric generalised_Bernoulli_number` (const `numeric` &`k`, const `numeric` &`b`)  
*The generalised Bernoulli number.*
- `ex Bernoulli_polynomial` (const `numeric` &`k`, const `ex` &`x`)  
*The Bernoulli polynomials.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`integration_kernel`, `basic`, `print_func< print_context >(&integration_kernel::do_print)`) `integration_kernel`
- `GINAC_BIND_UNARCHIVER` (`integration_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`basic_log_kernel`, `integration_kernel`, `print_func< print_context >(&basic_log_kernel::do_print)`) `basic_log_kernel`
- `GINAC_BIND_UNARCHIVER` (`basic_log_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`multiple_polylog_kernel`, `integration_kernel`, `print_func< print_context >(&multiple_polylog_kernel::do_print)`) `multiple_polylog_kernel`
- `GINAC_BIND_UNARCHIVER` (`multiple_polylog_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`ELi_kernel`, `integration_kernel`, `print_func< print_context >(&ELi_kernel::do_print)`) `ELi_kernel`
- `GINAC_BIND_UNARCHIVER` (`ELi_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`Ebar_kernel`, `integration_kernel`, `print_func< print_context >(&Ebar_kernel::do_print)`) `Ebar_kernel`
- `GINAC_BIND_UNARCHIVER` (`Ebar_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`Kronecker_dtau_kernel`, `integration_kernel`, `print_func< print_context >(&Kronecker_dtau_kernel::do_print)`) `Kronecker_dtau_kernel`

- [GINAC\\_BIND\\_UNARCHIVER](#) ([Kronecker\\_dtau\\_kernel](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([Kronecker\\_dz\\_kernel](#), [integration\\_kernel](#), [print\\_func< print\\_context >\(&Kronecker\\_dz\\_kernel::do\\_print\)](#)) [Kronecker\\_dz\\_kernel](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([Kronecker\\_dz\\_kernel](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([Eisenstein\\_kernel](#), [integration\\_kernel](#), [print\\_func< print\\_context >\(&Eisenstein\\_kernel::do\\_print\)](#)) [Eisenstein\\_kernel](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([Eisenstein\\_kernel](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([Eisenstein\\_h\\_kernel](#), [integration\\_kernel](#), [print\\_func< print\\_context >\(&Eisenstein\\_h\\_kernel::do\\_print\)](#)) [Eisenstein\\_h\\_kernel](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([Eisenstein\\_h\\_kernel](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([modular\\_form\\_kernel](#), [integration\\_kernel](#), [print\\_func< print\\_context >\(&modular\\_form\\_kernel::do\\_print\)](#)) [modular\\_form\\_kernel](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([modular\\_form\\_kernel](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([user\\_defined\\_kernel](#), [integration\\_kernel](#), [print\\_func< print\\_context >\(&user\\_defined\\_kernel::do\\_print\)](#)) [user\\_defined\\_kernel](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([user\\_defined\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([integration\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([basic\\_log\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([multiple\\_polylog\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([ELi\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Ebar\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Kronecker\\_dtau\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Kronecker\\_dz\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Eisenstein\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Eisenstein\\_h\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([modular\\_form\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([user\\_defined\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([lst](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([matrix](#), [basic](#), [print\\_func< print\\_context >\(&matrix::do\\_print\)](#)). [print\\_func< print\\_latex >\(&matrix::do\\_print\\_latex\)](#). [print\\_func< print\\_tree >\(&matrix::do\\_print\\_tree\)](#). [print\\_func< print\\_python\\_repr >\(&matrix::do\\_print\\_python\\_repr\)](#)) [matrix](#)
- *Default ctor.*
- [GINAC\\_BIND\\_UNARCHIVER](#) ([matrix](#))
- [ex lst\\_to\\_matrix](#) (const [lst](#) &l)
- *Convert list of lists to matrix.*
- [ex diag\\_matrix](#) (const [lst](#) &l)
- *Convert list of diagonal elements to matrix.*
- [ex diag\\_matrix](#) (std::initializer\_list< [ex](#) > l)
- [ex unit\\_matrix](#) (unsigned [r](#), unsigned [c](#))
- *Create an r times c unit matrix.*
- [ex symbolic\\_matrix](#) (unsigned [r](#), unsigned [c](#), const std::string &base\_name, const std::string &tex\_base\_name)
- *Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- [ex reduced\\_matrix](#) (const [matrix](#) &m, unsigned [r](#), unsigned [c](#))
- *Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*
- [ex sub\\_matrix](#) (const [matrix](#) &m, unsigned [r](#), unsigned [nr](#), unsigned [c](#), unsigned [nc](#))
- *Return the nr times nc submatrix starting at position r, c of matrix m.*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([matrix](#))
- [size\\_t nops](#) (const [matrix](#) &m)
- [ex expand](#) (const [matrix](#) &m, unsigned [options](#)=0)
- [ex evalf](#) (const [matrix](#) &m)
- unsigned [rows](#) (const [matrix](#) &m)
- unsigned [cols](#) (const [matrix](#) &m)



- [matrix transpose](#) (const [matrix](#) &m)
- [ex determinant](#) (const [matrix](#) &m, unsigned [options=determinant\\_algo::automatic](#))
- [ex trace](#) (const [matrix](#) &m)
- [ex charpoly](#) (const [matrix](#) &m, const [ex](#) &lambda)
- [matrix inverse](#) (const [matrix](#) &m)
- [matrix inverse](#) (const [matrix](#) &m, unsigned algo)
- unsigned [rank](#) (const [matrix](#) &m)
- unsigned [rank](#) (const [matrix](#) &m, unsigned [solve\\_algo](#))
- [ex unit\\_matrix](#) (unsigned x)  
*Create a x times x unit matrix.*
- [ex symbolic\\_matrix](#) (unsigned r, unsigned c, const std::string &base\_name)  
*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (mul, [expairseq](#), [print\\_func](#)< [print\\_context](#) >(&mul::do\_print). [print\\_func](#)< [print\\_latex](#) >(&mul::do\_print\_latex). [print\\_func](#)< [print\\_csrc](#) >(&mul::do\_print\_csrc). [print\\_func](#)< [print\\_tree](#) >(&mul::do\_print\_tree). [print\\_func](#)< [print\\_python\\_repr](#) >(&mul::do\_print\_python\_repr)) mul
- bool [tryfactsubs](#) (const [ex](#) &origfactor, const [ex](#) &patternfactor, int &nummatches, [exmap](#) &repls)
- bool [algebraic\\_match\\_mul\\_with\\_mul](#) (const mul &e, const [ex](#) &pat, [exmap](#) &repls, int factor, int &nummatches, const std::vector< bool > &subsed, std::vector< bool > &matched)  
*Checks whether e matches to the pattern pat and the (possibly to be updated) list of replacements repls.*
- [GINAC\\_BIND\\_UNARCHIVER](#) (mul)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (mul)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (ncmul, [exprseq](#), [print\\_func](#)< [print\\_context](#) >(&ncmul::do\_print). [print\\_func](#)< [print\\_tree](#) >(&ncmul::do\_print\_tree). [print\\_func](#)< [print\\_csrc](#) >(&ncmul::do\_print\_csrc). [print\\_func](#)< [print\\_python\\_repr](#) >(&ncmul::do\_print\_csrc)) ncmul
- [ex reeval\\_ncmul](#) (const [exvector](#) &v)
- [ex hold\\_ncmul](#) (const [exvector](#) &v)
- [GINAC\\_BIND\\_UNARCHIVER](#) (ncmul)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (ncmul)
- static bool [get\\_first\\_symbol](#) (const [ex](#) &e, [ex](#) &x)  
*Return pointer to first symbol found in expression.*
- static void [add\\_symbol](#) (const [ex](#) &s, [sym\\_desc\\_vec](#) &v)
- static void [collect\\_symbols](#) (const [ex](#) &e, [sym\\_desc\\_vec](#) &v)
- static void [get\\_symbol\\_stats](#) (const [ex](#) &a, const [ex](#) &b, [sym\\_desc\\_vec](#) &v)  
*Collect statistical information about symbols in polynomials.*
- static [numeric lcmcoeff](#) (const [ex](#) &e, const [numeric](#) &l)
- static [numeric lcm\\_of\\_coefficients\\_denominators](#) (const [ex](#) &e)  
*Compute LCM of denominators of coefficients of a polynomial.*
- static [ex multiply\\_lcm](#) (const [ex](#) &e, const [numeric](#) &lcm)  
*Bring polynomial from  $Q[X]$  to  $Z[X]$  by multiplying in the previously determined LCM of the coefficient's denominators.*
- [ex quo](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex rem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex decomp\\_rational](#) (const [ex](#) &a, const [ex](#) &x)  
*Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .*
- [ex prem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex sprem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- bool [divide](#) (const [ex](#) &a, const [ex](#) &b, [ex](#) &q, bool check\_args)  
*Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Q[X]$ .*

- static bool `divide_in_z` (const `ex` &a, const `ex` &b, `ex` &q, `sym_desc_vec::const_iterator` var)  
*Exact polynomial division of  $a(X)$  by  $b(X)$  in  $\mathbb{Z}[X]$ .*
- static `ex` `sr_gcd` (const `ex` &a, const `ex` &b, `sym_desc_vec::const_iterator` var)  
*Compute GCD of multivariate polynomials using the subresultant PRS algorithm.*
- static `ex` `interpolate` (const `ex` &gamma, const `numeric` &xi, const `ex` &x, int degree\_hint=1)  
*xi-adic polynomial interpolation*
- static bool `heur_gcd_z` (`ex` &res, const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, `sym_desc_vec::const_iterator` var)  
*Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
- static bool `heur_gcd` (`ex` &res, const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, `sym_desc_vec::const_iterator` var)  
*Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
- static `ex` `gcd_pf_pow` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
- static `ex` `gcd_pf_mul` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
- `ex` `gcd` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, bool check\_args, unsigned `options`)  
*Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $\mathbb{Z}[X]$ .*
- static `ex` `gcd_pf_pow_pow` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
- `ex` `lcm` (const `ex` &a, const `ex` &b, bool check\_args)  
*Compute LCM (Least Common Multiple) of multivariate polynomials in  $\mathbb{Z}[X]$ .*
- static `epvector` `sqrfree_yun` (const `ex` &a, const `symbol` &x)  
*Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.*
- `ex` `sqrfree` (const `ex` &a, const `lst` &l)  
*Compute a square-free factorization of a multivariate polynomial in  $\mathbb{Q}[X]$ .*
- `ex` `sqrfree_parfrac` (const `ex` &a, const `symbol` &x)  
*Compute square-free partial fraction decomposition of rational function  $a(x)$ .*
- static `ex` `replace_with_symbol` (const `ex` &e, `exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier)  
*Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
- static `ex` `replace_with_symbol` (const `ex` &e, `exmap` &repl)  
*Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
- static `ex` `frac_cancel` (const `ex` &n, const `ex` &d)  
*Fraction cancellation.*
- static `ex` `find_common_factor` (const `ex` &e, `ex` &factor, `exmap` &repl)  
*Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).*
- `ex` `collect_common_factors` (const `ex` &e)  
*Collect common factors in sums.*
- `ex` `resultant` (const `ex` &e1, const `ex` &e2, const `ex` &s)  
*Resultant of two expressions e1,e2 with respect to symbol s.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`numeric`, `basic`, `print_func`< `print_context` >(&`numeric::do_print`).  
`print_func`< `print_latex` >(&`numeric::do_print_latex`). `print_func`< `print_csrc` >(&`numeric::do_print_csrc`).  
`print_func`< `print_csrc_cl_N` >(&`numeric::do_print_csrc_cl_N`). `print_func`< `print_tree` >(&`numeric::do_print_tree`).  
`print_func`< `print_python_repr` >(&`numeric::do_print_python_repr`)) `numeric`  
*default ctor.*
- static const `cln::cl_F` `make_real_float` (const `cln::cl_idcoded_float` &dec)  
*Construct a floating point number from sign, mantissa, and exponent.*
- static const `cln::cl_F` `read_real_float` (`std::istream` &s)  
*Read serialized floating point number.*
- `GINAC_BIND_UNARCHIVER` (`numeric`)
- static void `write_real_float` (`std::ostream` &s, const `cln::cl_R` &n)
- static void `print_real_number` (const `print_context` &c, const `cln::cl_R` &x)  
*Helper function to print a real number in a nicer way than is CLN's default.*
- static void `print_integer_csrc` (const `print_context` &c, const `cln::cl_I` &x)  
*Helper function to print integer number in C++ source format.*
- static void `print_real_csrc` (const `print_context` &c, const `cln::cl_R` &x)

- Helper function to print real number in C++ source format.*
- `template<typename T1 , typename T2 >`  
`static bool coerce (T1 &dst, const T2 &arg)`
- `template<>`  
`bool coerce< int, cln::cl_I > (int &dst, const cln::cl_I &arg)`  
*Check if CLN integer can be converted into int.*
- `template<>`  
`bool coerce< unsigned int, cln::cl_I > (unsigned int &dst, const cln::cl_I &arg)`
- `static void print_real_cl_N (const print_context &c, const cln::cl_R &x)`  
*Helper function to print real number in C++ source format using cl\_N types.*
- `const numeric exp (const numeric &x)`  
*Exponential function.*
- `const numeric log (const numeric &x)`  
*Natural logarithm.*
- `const numeric sin (const numeric &x)`  
*Numeric sine (trigonometric function).*
- `const numeric cos (const numeric &x)`  
*Numeric cosine (trigonometric function).*
- `const numeric tan (const numeric &x)`  
*Numeric tangent (trigonometric function).*
- `const numeric asin (const numeric &x)`  
*Numeric inverse sine (trigonometric function).*
- `const numeric acos (const numeric &x)`  
*Numeric inverse cosine (trigonometric function).*
- `const numeric atan (const numeric &x)`  
*Numeric arcustangent.*
- `const numeric atan (const numeric &y, const numeric &x)`  
*Numeric arcustangent of two arguments, analytically continued in a suitable way.*
- `const numeric sinh (const numeric &x)`  
*Numeric hyperbolic sine (trigonometric function).*
- `const numeric cosh (const numeric &x)`  
*Numeric hyperbolic cosine (trigonometric function).*
- `const numeric tanh (const numeric &x)`  
*Numeric hyperbolic tangent (trigonometric function).*
- `const numeric asinh (const numeric &x)`  
*Numeric inverse hyperbolic sine (trigonometric function).*
- `const numeric acosh (const numeric &x)`  
*Numeric inverse hyperbolic cosine (trigonometric function).*
- `const numeric atanh (const numeric &x)`  
*Numeric inverse hyperbolic tangent (trigonometric function).*
- `static cln::cl_N Li2_series (const cln::cl_N &x, const cln::float_format_t &prec)`  
*Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.*
- `static cln::cl_N Li2_projection (const cln::cl_N &x, const cln::float_format_t &prec)`  
*Folds Li2's argument inside a small rectangle to enhance convergence.*
- `const cln::cl_N Li2_ (const cln::cl_N &value)`  
*Numeric evaluation of Dilogarithm.*
- `const numeric Li2 (const numeric &x)`
- `const numeric zeta (const numeric &x)`  
*Numeric evaluation of Riemann's Zeta function.*
- `static cln::float_format_t guess_precision (const cln::cl_N &x)`
- `const cln::cl_N lgamma (const cln::cl_N &x)`

*The Gamma function.*

- `const numeric lgamma` (`const numeric &x`)
- `const cln::cl_N tgamma` (`const cln::cl_N &x`)
- `const numeric tgamma` (`const numeric &x`)
- `const numeric psi` (`const numeric &x`)

*The psi function (aka polygamma function).*

- `const numeric psi` (`const numeric &n`, `const numeric &x`)

*The psi functions (aka polygamma functions).*

- `const numeric factorial` (`const numeric &n`)

*Factorial combinatorial function.*

- `const numeric doublefactorial` (`const numeric &n`)

*The double factorial combinatorial function.*

- `const numeric binomial` (`const numeric &n`, `const numeric &k`)

*The Binomial coefficients.*

- `const numeric bernoulli` (`const numeric &nn`)

*Bernoulli number.*

- `const numeric fibonacci` (`const numeric &n`)

*Fibonacci number.*

- `const numeric abs` (`const numeric &x`)

*Absolute value.*

- `const numeric mod` (`const numeric &a`, `const numeric &b`)

*Modulus (in positive representation).*

- `const numeric smod` (`const numeric &a_`, `const numeric &b_`)

*Modulus (in symmetric representation).*

- `const numeric irem` (`const numeric &a`, `const numeric &b`)

*Numeric integer remainder.*

- `const numeric irem` (`const numeric &a`, `const numeric &b`, `numeric &q`)

*Numeric integer remainder.*

- `const numeric iquo` (`const numeric &a`, `const numeric &b`)

*Numeric integer quotient.*

- `const numeric iquo` (`const numeric &a`, `const numeric &b`, `numeric &r`)

*Numeric integer quotient.*

- `const numeric gcd` (`const numeric &a`, `const numeric &b`)

*Greatest Common Divisor.*

- `const numeric lcm` (`const numeric &a`, `const numeric &b`)

*Least Common Multiple.*

- `const numeric sqrt` (`const numeric &x`)

*Numeric square root.*

- `const numeric isqrt` (`const numeric &x`)

*Integer numeric square root.*

- `ex PiEvalf` ()

*Floating point evaluation of Archimedes' constant Pi.*

- `ex EulerEvalf` ()

*Floating point evaluation of Euler's constant gamma.*

- `ex CatalanEvalf` ()

*Floating point evaluation of Catalan's constant.*

- `std::ostream & operator<<` (`std::ostream &os`, `const _numeric_digits &e`)
- `GINAC_DECLARE_UNARCHIVER` (`numeric`)
- `const numeric pow` (`const numeric &x`, `const numeric &y`)
- `const numeric inverse` (`const numeric &x`)
- `numeric step` (`const numeric &x`)

- `int csgn (const numeric &x)`
- `bool is_zero (const numeric &x)`
- `bool is_positive (const numeric &x)`
- `bool is_negative (const numeric &x)`
- `bool is_integer (const numeric &x)`
- `bool is_pos_integer (const numeric &x)`
- `bool is_nonneg_integer (const numeric &x)`
- `bool is_even (const numeric &x)`
- `bool is_odd (const numeric &x)`
- `bool is_prime (const numeric &x)`
- `bool is_rational (const numeric &x)`
- `bool is_real (const numeric &x)`
- `bool is_cinteger (const numeric &x)`
- `bool is_crational (const numeric &x)`
- `int to_int (const numeric &x)`
- `long to_long (const numeric &x)`
- `double to_double (const numeric &x)`
- `const numeric real (const numeric &x)`
- `const numeric imag (const numeric &x)`
- `const numeric numer (const numeric &x)`
- `const numeric denom (const numeric &x)`
- `static const ex exadd (const ex &lh, const ex &rh)`  
*Used internally by `operator+()` to add two ex objects.*
- `static const ex exmul (const ex &lh, const ex &rh)`  
*Used internally by `operator*()` to multiply two ex objects.*
- `static const ex exminus (const ex &lh)`  
*Used internally by `operator-()` and friends to change the sign of an argument.*
- `const ex operator+ (const ex &lh, const ex &rh)`
- `const ex operator- (const ex &lh, const ex &rh)`
- `const ex operator* (const ex &lh, const ex &rh)`
- `const ex operator/ (const ex &lh, const ex &rh)`
- `const numeric operator+ (const numeric &lh, const numeric &rh)`
- `const numeric operator- (const numeric &lh, const numeric &rh)`
- `const numeric operator* (const numeric &lh, const numeric &rh)`
- `const numeric operator/ (const numeric &lh, const numeric &rh)`
- `ex & operator+= (ex &lh, const ex &rh)`
- `ex & operator-= (ex &lh, const ex &rh)`
- `ex & operator*= (ex &lh, const ex &rh)`
- `ex & operator/= (ex &lh, const ex &rh)`
- `numeric & operator+= (numeric &lh, const numeric &rh)`
- `numeric & operator-= (numeric &lh, const numeric &rh)`
- `numeric & operator*= (numeric &lh, const numeric &rh)`
- `numeric & operator/= (numeric &lh, const numeric &rh)`
- `const ex operator+ (const ex &lh)`
- `const ex operator- (const ex &lh)`
- `const numeric operator+ (const numeric &lh)`
- `const numeric operator- (const numeric &lh)`
- `ex & operator++ (ex &rh)`  
*Expression prefix increment.*
- `ex & operator-- (ex &rh)`  
*Expression prefix decrement.*
- `const ex operator++ (ex &lh, int)`  
*Expression postfix increment.*
- `const ex operator-- (ex &lh, int)`

*Expression postfix decrement.*

- `numeric & operator++ (numeric &rh)`

*Numeric prefix increment.*

- `numeric & operator-- (numeric &rh)`

*Numeric prefix decrement.*

- `const numeric operator++ (numeric &lh, int)`

*Numeric postfix increment.*

- `const numeric operator-- (numeric &lh, int)`

*Numeric postfix decrement.*

- `const relational operator== (const ex &lh, const ex &rh)`
- `const relational operator!= (const ex &lh, const ex &rh)`
- `const relational operator< (const ex &lh, const ex &rh)`
- `const relational operator<= (const ex &lh, const ex &rh)`
- `const relational operator> (const ex &lh, const ex &rh)`
- `const relational operator>= (const ex &lh, const ex &rh)`
- `static int my_ios_index ()`
- `static void my_ios_callback (std::ios_base::event ev, std::ios_base &s, int i)`
- `static print_context * get_print_context (std::ios_base &s)`
- `static void set_print_context (std::ios_base &s, const print_context &c)`
- `static unsigned get_print_options (std::ios_base &s)`
- `static void set_print_options (std::ostream &s, unsigned options)`
- `std::ostream & operator<< (std::ostream &os, const ex &e)`
- `std::istream & operator>> (std::istream &is, ex &e)`
- `std::ostream & dflt (std::ostream &os)`
- `std::ostream & latex (std::ostream &os)`
- `std::ostream & python (std::ostream &os)`
- `std::ostream & python_repr (std::ostream &os)`
- `std::ostream & tree (std::ostream &os)`
- `std::ostream & csrc (std::ostream &os)`
- `std::ostream & csrc_float (std::ostream &os)`
- `std::ostream & csrc_double (std::ostream &os)`
- `std::ostream & csrc_cl_N (std::ostream &os)`
- `std::ostream & index_dimensions (std::ostream &os)`
- `std::ostream & no_index_dimensions (std::ostream &os)`
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (power, basic, print_func< print_dflt >(&power::do_print_dflt). print_func< print_latex >(&power::do_print_latex). print_func< print_csrc >(&power::do_print_csrc). print_func< print_python >(&power::do_print_python). print_func< print_python_repr >(&power::do_print_python_repr). print_func< print_csrc_cl_N >(&power::do_print_csrc_cl_N)) power`
- `static void print_sym_pow (const print_context &c, const symbol &x, int exp)`
- `GINAC_BIND_UNARCHIVER (power)`
- `GINAC_DECLARE_UNARCHIVER (power)`
- `ex pow (const ex &b, const ex &e)`

*Symbolic exponentiation.*

- `template<typename T1 , typename T2 >  
ex pow (const T1 &b, const T2 &e)`
- `ex sqrt (const ex &a)`

*Square root expression.*

- `template<class T >  
bool is_a (const print_context &obj)`

*Check if obj is a T, including base classes.*

- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (pseries, basic, print_func< print_context >(&pseries::do_print). print_func< print_latex >(&pseries::do_print_latex). print_func< print_tree >(&pseries::do_print_tree). print_func< print_python >(&pseries::do_print_python). print_func< print_python_repr >(&pseries::do_print_python_repr)) pseries`

- [GINAC\\_BIND\\_UNARCHIVER](#) ([pseries](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([pseries](#))
- [ex series\\_to\\_poly](#) (const [ex](#) &[e](#))
 

*Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.*
- bool [is\\_terminating](#) (const [pseries](#) &[s](#))
- template<typename T >
 [return\\_type\\_t make\\_return\\_type\\_t](#) (const unsigned [rl](#)=0)
- template<class Alg, class Ctx, class T, class C >
 void [set\\_print\\_func](#) (void f(const T &, const C &[c](#), unsigned))
 

*Add or replace a print method.*
- template<class Alg, class Ctx, class T, class C >
 void [set\\_print\\_func](#) (void(T::\*f)(const C &, unsigned))
 

*Add or replace a print method.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([relational](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[relational::do\\_print](#)),
 [print\\_func](#)< [print\\_tree](#) >(&[relational::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[relational::do\\_print\\_python\\_repr](#)))
 [relational](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([relational](#))
- static void [print\\_operator](#) (const [print\\_context](#) &[c](#), [relational::operators](#) o)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([relational](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([symbol](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[symbol::do\\_print](#)),
 [print\\_func](#)< [print\\_latex](#) >(&[symbol::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_tree](#) >(&[symbol::do\\_print\\_tree](#)),
 [print\\_func](#)< [print\\_python\\_repr](#) >(&[symbol::do\\_print\\_python\\_repr](#)))
 [symbol](#)
- static const std::string & [get\\_default\\_TeX\\_name](#) (const std::string &[name](#))
 

*Return default TeX name for symbol.*
- [GINAC\\_BIND\\_UNARCHIVER](#) ([symbol](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([realsymbol](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([possymbol](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([symbol](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([realsymbol](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([possymbol](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([symmetry](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[symmetry::do\\_print](#)),
 [print\\_func](#)< [print\\_tree](#) >(&[symmetry::do\\_print\\_tree](#)))
 [symmetry](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([symmetry](#))
- static const [symmetry](#) & [index0](#) ()
- static const [symmetry](#) & [index1](#) ()
- static const [symmetry](#) & [index2](#) ()
- static const [symmetry](#) & [index3](#) ()
- const [symmetry](#) & [not\\_symmetric](#) ()
- const [symmetry](#) & [symmetric2](#) ()
- const [symmetry](#) & [symmetric3](#) ()
- const [symmetry](#) & [symmetric4](#) ()
- const [symmetry](#) & [antisymmetric2](#) ()
- const [symmetry](#) & [antisymmetric3](#) ()
- const [symmetry](#) & [antisymmetric4](#) ()
- int [canonicalize](#) ([exvector::iterator](#) v, const [symmetry](#) &[symm](#))
 

*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- static [ex symm](#) (const [ex](#) &[e](#), [exvector::const\\_iterator](#) first, [exvector::const\\_iterator](#) [last](#), bool asymmetric)
- [ex symmetrize](#) (const [ex](#) &[e](#), [exvector::const\\_iterator](#) first, [exvector::const\\_iterator](#) [last](#))
 

*Symmetrize expression over a set of objects (symbols, indices).*
- [ex antisymmetrize](#) (const [ex](#) &[e](#), [exvector::const\\_iterator](#) first, [exvector::const\\_iterator](#) [last](#))
 

*Antisymmetrize expression over a set of objects (symbols, indices).*
- [ex symmetrize\\_cyclic](#) (const [ex](#) &[e](#), [exvector::const\\_iterator](#) first, [exvector::const\\_iterator](#) [last](#))
 

*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*



- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([symmetry](#))
- [symmetry sy\\_none](#) ()
- [symmetry sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry sy\\_symm](#) ()
- [symmetry sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry sy\\_anti](#) ()
- [symmetry sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry sy\\_cycl](#) ()
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [ex symmetrize](#) (const [ex](#) &e, const [exvector](#) &v)  
*Symmetrize expression over a set of objects (symbols, indices).*
- [ex antisymmetrize](#) (const [ex](#) &e, const [exvector](#) &v)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- [ex symmetrize\\_cyclic](#) (const [ex](#) &e, const [exvector](#) &v)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([tensdelta](#), [tensor](#), print\_func< [print\\_dflt](#) >(&[tensdelta::do\\_print](#)), print\_func< [print\\_latex](#) >(&[tensdelta::do\\_print\\_latex](#))) [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)([tensmetric](#), [print\\_func](#)< [print\\_dflt](#) >(&[tensmetric::do\\_print](#)), print\_func< [print\\_latex](#) >(&[tensmetric](#)))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([minkmetric](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([tensepsilon](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([tensdelta](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([tensmetric](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([spinmetric](#))
- [ex delta\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a delta tensor with specified indices.*
- [ex metric\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a symmetric metric tensor with specified indices.*
- [ex lorentz\\_g](#) (const [ex](#) &i1, const [ex](#) &i2, bool pos\_sig=false)  
*Create a Minkowski metric tensor with specified indices.*
- [ex spinor\\_metric](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a spinor metric tensor with specified indices.*
- [ex epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create an epsilon tensor in a Euclidean space with two indices.*
- [ex epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Create an epsilon tensor in a Euclidean space with three indices.*
- [ex lorentz\\_eps](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4, bool pos\_sig=false)  
*Create an epsilon tensor in a Minkowski space with four indices.*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensdelta](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensmetric](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([minkmetric](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([spinmetric](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensepsilon](#))
- [unsigned log2](#) (unsigned [n](#))  
*Integer binary logarithm.*



- const [numeric\\_multinomial\\_coefficient](#) (const std::vector< unsigned > &p)  
*Compute the multinomial coefficient  $n!/(p_1! * p_2! * \dots * p_k!)$  where  $n = p_1 + p_2 + \dots + p_k$ , i.e.*
- unsigned [rotate\\_left](#) (unsigned n)  
*Rotate bits of unsigned value by one bit to the left.*
- template<class T >  
int [compare\\_pointers](#) (const T \*a, const T \*b)  
*Compare two pointers (just to establish some sort of canonical order).*
- unsigned [golden\\_ratio\\_hash](#) (uintptr\_t n)  
*Truncated multiplication with golden ratio, for computing hash values.*
- template<class It >  
int [permutation\\_sign](#) (It first, It last)
- template<class It, class Cmp, class Swap >  
int [permutation\\_sign](#) (It first, It last, Cmp comp, Swap swapit)
- template<class It, class Cmp, class Swap >  
void [shaker\\_sort](#) (It first, It last, Cmp comp, Swap swapit)
- template<class It, class Swap >  
void [cyclic\\_permutation](#) (It first, It last, It new\_first, Swap swapit)
- template<typename T >  
std::enable\_if< [has\\_distance](#)< T >::value, typename std::iterator\_traits< T >::difference\_type >::type  
[format\\_index\\_value](#) (const T &a, const T &b)  
*For printing a multi-index: If the templates are used, where T is an iterator, printing the address where the iterator points to is not meaningful.*
- template<typename T >  
std::enable\_if<![has\\_distance](#)< T >::value, T >::type [format\\_index\\_value](#) (const T &a, const T &b)  
*For all other cases we simply print the value.*
- template<class T >  
std::ostream & [operator<<](#) (std::ostream &os, const [basic\\_multi\\_iterator](#)< T > &v)  
*Output operator.*
- template<class T >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_ordered](#)< T > &v)  
*Output operator.*
- template<class T >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_ordered\\_eq](#)< T > &v)  
*Output operator.*
- template<class T >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_ordered\\_eq\\_indv](#)< T > &v)  
*Output operator.*
- template<class T >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_counter](#)< T > &v)  
*Output operator.*
- template<class T >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_counter\\_indv](#)< T > &v)  
*Output operator.*
- template<class T >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_permutation](#)< T > &v)  
*Output operator.*
- template<class T >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_shuffle](#)< T > &v)  
*Output operator.*
- template<class T >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_shuffle\\_prime](#)< T > &v)  
*Output operator.*

- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`wildcard`, `basic`, `print_func`< `print_context` >(&`wildcard::do_print`).  
`print_func`< `print_tree` >(&`wildcard::do_print_tree`). `print_func`< `print_python_repr` >(&`wildcard::do_print_python_repr`)  
`wildcard`
- `GINAC_BIND_UNARCHIVER` (`wildcard`)
- `bool haswild` (`const ex &x`)  
*Check whether x has a wildcard anywhere as a subexpression.*
- `GINAC_DECLARE_UNARCHIVER` (`wildcard`)
- `ex wild` (`unsigned label=0`)  
*Create a wildcard object with the specified label.*

## Variables

- static `unarchive_table_t` `unarch_table_instance`
- `GiNaC::evalm_map_function` `map_evalm`
- `GiNaC::eval_integ_map_function` `map_eval_integ`
- `tensor`
- `const constant Pi` ("Pi", `PiEvalf`, "\i", `domain::positive`)  
*Pi.*
- `const constant Euler` ("Euler", `EulerEvalf`, "\amma\_E", `domain::positive`)  
*Euler's constant.*
- `const constant Catalan` ("Catalan", `CatalanEvalf`, "G", `domain::positive`)  
*Catalan's constant.*
- static `unsigned const crctab` [256]
- static `library_init` `library_initializer`  
*For construction of flyweights, etc.*
- `const basic * _num0_bp`
- `idx`
- `unsigned force_include_tgamma` = `tgamma_SERIAL::serial`
- `unsigned force_include_zeta1` = `zeta1_SERIAL::serial`
- `const numeric I` = `numeric(cln::complex(cln::cl_I(0),cln::cl_I(1)))`  
*Imaginary unit.*
- `_numeric_digits` `Digits`  
*Accuracy in decimal digits.*
- `unsigned next_print_context_id` = 0  
*Next free ID for `print_context` types.*
- `const int version_major` = `GINACLIB_MAJOR_VERSION`
- `const int version_minor` = `GINACLIB_MINOR_VERSION`
- `const int version_micro` = `GINACLIB_MICRO_VERSION`
- `const numeric * _num_120_p`
- `const ex _ex_120` = `_ex_120`
- `const numeric * _num_60_p`
- `const ex _ex_60` = `_ex_60`
- `const numeric * _num_48_p`
- `const ex _ex_48` = `_ex_48`
- `const numeric * _num_30_p`
- `const ex _ex_30` = `_ex_30`
- `const numeric * _num_25_p`
- `const ex _ex_25` = `_ex_25`
- `const numeric * _num_24_p`
- `const ex _ex_24` = `_ex_24`
- `const numeric * _num_20_p`
- `const ex _ex_20` = `_ex_20`

- const [numeric](#) \* [\\_num\\_18\\_p](#)
- const [ex](#) [\\_ex\\_18](#) = [\\_ex\\_18](#)
- const [numeric](#) \* [\\_num\\_15\\_p](#)
- const [ex](#) [\\_ex\\_15](#) = [\\_ex\\_15](#)
- const [numeric](#) \* [\\_num\\_12\\_p](#)
- const [ex](#) [\\_ex\\_12](#) = [\\_ex\\_12](#)
- const [numeric](#) \* [\\_num\\_11\\_p](#)
- const [ex](#) [\\_ex\\_11](#) = [\\_ex\\_11](#)
- const [numeric](#) \* [\\_num\\_10\\_p](#)
- const [ex](#) [\\_ex\\_10](#) = [\\_ex\\_10](#)
- const [numeric](#) \* [\\_num\\_9\\_p](#)
- const [ex](#) [\\_ex\\_9](#) = [\\_ex\\_9](#)
- const [numeric](#) \* [\\_num\\_8\\_p](#)
- const [ex](#) [\\_ex\\_8](#) = [\\_ex\\_8](#)
- const [numeric](#) \* [\\_num\\_7\\_p](#)
- const [ex](#) [\\_ex\\_7](#) = [\\_ex\\_7](#)
- const [numeric](#) \* [\\_num\\_6\\_p](#)
- const [ex](#) [\\_ex\\_6](#) = [\\_ex\\_6](#)
- const [numeric](#) \* [\\_num\\_5\\_p](#)
- const [ex](#) [\\_ex\\_5](#) = [\\_ex\\_5](#)
- const [numeric](#) \* [\\_num\\_4\\_p](#)
- const [ex](#) [\\_ex\\_4](#) = [\\_ex\\_4](#)
- const [numeric](#) \* [\\_num\\_3\\_p](#)
- const [ex](#) [\\_ex\\_3](#) = [\\_ex\\_3](#)
- const [numeric](#) \* [\\_num\\_2\\_p](#)
- const [ex](#) [\\_ex\\_2](#) = [\\_ex\\_2](#)
- const [numeric](#) \* [\\_num\\_1\\_p](#)
- const [ex](#) [\\_ex\\_1](#) = [\\_ex\\_1](#)
- const [numeric](#) \* [\\_num\\_1\\_2\\_p](#)
- const [ex](#) [\\_ex\\_1\\_2](#) = [\\_ex\\_1\\_2](#)
- const [numeric](#) \* [\\_num\\_1\\_3\\_p](#)
- const [ex](#) [\\_ex\\_1\\_3](#) = [\\_ex\\_1\\_3](#)
- const [numeric](#) \* [\\_num\\_1\\_4\\_p](#)
- const [ex](#) [\\_ex\\_1\\_4](#) = [\\_ex\\_1\\_4](#)
- const [numeric](#) \* [\\_num0\\_p](#)
- const [ex](#) [\\_ex0](#) = [\\_ex0](#)
- const [numeric](#) \* [\\_num1\\_4\\_p](#)
- const [ex](#) [\\_ex1\\_4](#) = [\\_ex1\\_4](#)
- const [numeric](#) \* [\\_num1\\_3\\_p](#)
- const [ex](#) [\\_ex1\\_3](#) = [\\_ex1\\_3](#)
- const [numeric](#) \* [\\_num1\\_2\\_p](#)
- const [ex](#) [\\_ex1\\_2](#) = [\\_ex1\\_2](#)
- const [numeric](#) \* [\\_num1\\_p](#)
- const [ex](#) [\\_ex1](#) = [\\_ex1](#)
- const [numeric](#) \* [\\_num2\\_p](#)
- const [ex](#) [\\_ex2](#) = [\\_ex2](#)
- const [numeric](#) \* [\\_num3\\_p](#)
- const [ex](#) [\\_ex3](#) = [\\_ex3](#)
- const [numeric](#) \* [\\_num4\\_p](#)
- const [ex](#) [\\_ex4](#) = [\\_ex4](#)
- const [numeric](#) \* [\\_num5\\_p](#)
- const [ex](#) [\\_ex5](#) = [\\_ex5](#)
- const [numeric](#) \* [\\_num6\\_p](#)
- const [ex](#) [\\_ex6](#) = [\\_ex6](#)
- const [numeric](#) \* [\\_num7\\_p](#)

- const [ex\\_ex7](#) = [\\_ex7](#)
- const [numeric](#) \* [\\_num8\\_p](#)
- const [ex\\_ex8](#) = [\\_ex8](#)
- const [numeric](#) \* [\\_num9\\_p](#)
- const [ex\\_ex9](#) = [\\_ex9](#)
- const [numeric](#) \* [\\_num10\\_p](#)
- const [ex\\_ex10](#) = [\\_ex10](#)
- const [numeric](#) \* [\\_num11\\_p](#)
- const [ex\\_ex11](#) = [\\_ex11](#)
- const [numeric](#) \* [\\_num12\\_p](#)
- const [ex\\_ex12](#) = [\\_ex12](#)
- const [numeric](#) \* [\\_num15\\_p](#)
- const [ex\\_ex15](#) = [\\_ex15](#)
- const [numeric](#) \* [\\_num18\\_p](#)
- const [ex\\_ex18](#) = [\\_ex18](#)
- const [numeric](#) \* [\\_num20\\_p](#)
- const [ex\\_ex20](#) = [\\_ex20](#)
- const [numeric](#) \* [\\_num24\\_p](#)
- const [ex\\_ex24](#) = [\\_ex24](#)
- const [numeric](#) \* [\\_num25\\_p](#)
- const [ex\\_ex25](#) = [\\_ex25](#)
- const [numeric](#) \* [\\_num30\\_p](#)
- const [ex\\_ex30](#) = [\\_ex30](#)
- const [numeric](#) \* [\\_num48\\_p](#)
- const [ex\\_ex48](#) = [\\_ex48](#)
- const [numeric](#) \* [\\_num60\\_p](#)
- const [ex\\_ex60](#) = [\\_ex60](#)
- const [numeric](#) \* [\\_num120\\_p](#)
- const [ex\\_ex120](#) = [\\_ex120](#)

### 5.1.1 Typedef Documentation

#### 5.1.1.1 `archive_node_id`

```
typedef unsigned GiNaC::archive\_node\_id
```

Numerical ID value to refer to an [archive\\_node](#).

#### 5.1.1.2 `archive_atom`

```
typedef unsigned GiNaC::archive\_atom
```

Numerical ID value to refer to a string.

#### 5.1.1.3 synthesise\_func

```
typedef basic>(* GiNaC::synthesise_func) ()
```

#### 5.1.1.4 unarchive\_map\_t

```
typedef std::map<std::string, synthesise_func> GiNaC::unarchive_map_t
```

#### 5.1.1.5 exvector

```
typedef std::vector<ex> GiNaC::exvector
```

#### 5.1.1.6 exset

```
typedef std::set<ex, ex_is_less> GiNaC::exset
```

#### 5.1.1.7 exmap

```
typedef std::map<ex, ex, ex_is_less> GiNaC::exmap
```

#### 5.1.1.8 evalffunctype

```
typedef ex(* GiNaC::evalffunctype) ()
```

#### 5.1.1.9 FUNCP\_1P

```
typedef double(* GiNaC::FUNCP_1P) (double)
```

Function pointer with one function parameter.

#### 5.1.1.10 FUNCP\_2P

```
typedef double(* GiNaC::FUNCP_2P) (double, double)
```

Function pointer with two function parameters.

#### 5.1.1.11 FUNCP\_CUBA

```
typedef void(* GiNaC::FUNCP_CUBA) (const int *, const double[], const int *, double[])
```

Function pointer for use with the CUBA library (<http://www.feynarts.de/cuba>).

#### 5.1.1.12 epvector

```
typedef std::vector<expair> GiNaC::epvector
```

expair-vector

#### 5.1.1.13 epp

```
typedef epvector::iterator GiNaC::epp
```

expair-vector pointer

#### 5.1.1.14 exprseq

```
typedef container<std::vector> GiNaC::exprseq
```

#### 5.1.1.15 paramset

```
typedef std::multiset<unsigned> GiNaC::paramset
```

#### 5.1.1.16 eval\_funcp

```
typedef ex(* GiNaC::eval_funcp) ()
```

#### 5.1.1.17 evalf\_funcp

```
typedef ex(* GiNaC::evalf_funcp) ()
```

#### 5.1.1.18 conjugate\_funcp

```
typedef ex(* GiNaC::conjugate_funcp) ()
```

#### 5.1.1.19 real\_part\_funcp

```
typedef ex(* GiNaC::real_part_funcp) ()
```

#### 5.1.1.20 imag\_part\_funcp

```
typedef ex(* GiNaC::imag_part_funcp) ()
```

#### 5.1.1.21 expand\_funcp

```
typedef ex(* GiNaC::expand_funcp) ()
```

#### 5.1.1.22 derivative\_funcp

```
typedef ex(* GiNaC::derivative_funcp) ()
```

#### 5.1.1.23 expl\_derivative\_funcp

```
typedef ex(* GiNaC::expl_derivative_funcp) ()
```

#### 5.1.1.24 power\_funcp

```
typedef ex(* GiNaC::power_funcp) ()
```

**5.1.1.25 series\_funcp**

```
typedef ex(* GiNaC::series_funcp) ()
```

**5.1.1.26 print\_funcp**

```
typedef void(* GiNaC::print_funcp) ()
```

**5.1.1.27 info\_funcp**

```
typedef bool(* GiNaC::info_funcp) ()
```

**5.1.1.28 eval\_funcp\_1**

```
typedef ex(* GiNaC::eval_funcp_1) (const ex &)
```

**5.1.1.29 evalf\_funcp\_1**

```
typedef ex(* GiNaC::evalf_funcp_1) (const ex &)
```

**5.1.1.30 conjugate\_funcp\_1**

```
typedef ex(* GiNaC::conjugate_funcp_1) (const ex &)
```

**5.1.1.31 real\_part\_funcp\_1**

```
typedef ex(* GiNaC::real_part_funcp_1) (const ex &)
```

**5.1.1.32 imag\_part\_funcp\_1**

```
typedef ex(* GiNaC::imag_part_funcp_1) (const ex &)
```



#### 5.1.1.33 expand\_funcp\_1

```
typedef ex(* GiNaC::expand_funcp_1) (const ex &, unsigned)
```

#### 5.1.1.34 derivative\_funcp\_1

```
typedef ex(* GiNaC::derivative_funcp_1) (const ex &, unsigned)
```

#### 5.1.1.35 expl\_derivative\_funcp\_1

```
typedef ex(* GiNaC::expl_derivative_funcp_1) (const ex &, const symbol &)
```

#### 5.1.1.36 power\_funcp\_1

```
typedef ex(* GiNaC::power_funcp_1) (const ex &, const ex &)
```

#### 5.1.1.37 series\_funcp\_1

```
typedef ex(* GiNaC::series_funcp_1) (const ex &, const relational &, int, unsigned)
```

#### 5.1.1.38 print\_funcp\_1

```
typedef void(* GiNaC::print_funcp_1) (const ex &, const print_context &)
```

#### 5.1.1.39 info\_funcp\_1

```
typedef bool(* GiNaC::info_funcp_1) (const ex &, unsigned)
```

#### 5.1.1.40 eval\_funcp\_2

```
typedef ex(* GiNaC::eval_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.41 evalf\_funcp\_2

```
typedef ex(* GiNaC::evalf_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.42 conjugate\_funcp\_2

```
typedef ex(* GiNaC::conjugate_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.43 real\_part\_funcp\_2

```
typedef ex(* GiNaC::real_part_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.44 imag\_part\_funcp\_2

```
typedef ex(* GiNaC::imag_part_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.45 expand\_funcp\_2

```
typedef ex(* GiNaC::expand_funcp_2) (const ex &, const ex &, unsigned)
```

#### 5.1.1.46 derivative\_funcp\_2

```
typedef ex(* GiNaC::derivative_funcp_2) (const ex &, const ex &, unsigned)
```

#### 5.1.1.47 expl\_derivative\_funcp\_2

```
typedef ex(* GiNaC::expl_derivative_funcp_2) (const ex &, const ex &, const symbol &)
```

#### 5.1.1.48 power\_funcp\_2

```
typedef ex(* GiNaC::power_funcp_2) (const ex &, const ex &, const ex &)
```

#### 5.1.1.49 series\_funcp\_2

```
typedef ex(* GiNaC::series_funcp_2) (const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.50 print\_funcp\_2

```
typedef void(* GiNaC::print_funcp_2) (const ex &, const ex &, const print_context &)
```

#### 5.1.1.51 info\_funcp\_2

```
typedef bool(* GiNaC::info_funcp_2) (const ex &, const ex &, unsigned)
```

#### 5.1.1.52 eval\_funcp\_3

```
typedef ex(* GiNaC::eval_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.53 evalf\_funcp\_3

```
typedef ex(* GiNaC::evalf_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.54 conjugate\_funcp\_3

```
typedef ex(* GiNaC::conjugate_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.55 real\_part\_funcp\_3

```
typedef ex(* GiNaC::real_part_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.56 imag\_part\_funcp\_3

```
typedef ex(* GiNaC::imag_part_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.57 expand\_funcp\_3

```
typedef ex(* GiNaC::expand_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.58 derivative\_funcp\_3

```
typedef ex(* GiNaC::derivative_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.59 expl\_derivative\_funcp\_3

```
typedef ex(* GiNaC::expl_derivative_funcp_3) (const ex &, const ex &, const ex &, const symbol  
&)
```

#### 5.1.1.60 power\_funcp\_3

```
typedef ex(* GiNaC::power_funcp_3) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.61 series\_funcp\_3

```
typedef ex(* GiNaC::series_funcp_3) (const ex &, const ex &, const ex &, const relational &,  
int, unsigned)
```

#### 5.1.1.62 print\_funcp\_3

```
typedef void(* GiNaC::print_funcp_3) (const ex &, const ex &, const ex &, const print_context  
&)
```

#### 5.1.1.63 info\_funcp\_3

```
typedef bool(* GiNaC::info_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.64 eval\_funcp\_4

```
typedef ex(* GiNaC::eval_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.65 evalf\_funcp\_4

```
typedef ex(* GiNaC::evalf_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.66 conjugate\_funcp\_4

```
typedef ex(* GiNaC::conjugate_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.67 real\_part\_funcp\_4

```
typedef ex(* GiNaC::real_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.68 imag\_part\_funcp\_4

```
typedef ex(* GiNaC::imag_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.69 expand\_funcp\_4

```
typedef ex(* GiNaC::expand_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.70 derivative\_funcp\_4

```
typedef ex(* GiNaC::derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.71 expl\_derivative\_funcp\_4

```
typedef ex(* GiNaC::expl_derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &,  
const symbol &)
```

#### 5.1.1.72 power\_funcp\_4

```
typedef ex(* GiNaC::power_funcp_4) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.73 series\_funcp\_4

```
typedef ex(* GiNaC::series_funcp_4) (const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.74 print\_funcp\_4

```
typedef void(* GiNaC::print_funcp_4) (const ex &, const ex &, const ex &, const ex &, const print_context &)
```

#### 5.1.1.75 info\_funcp\_4

```
typedef bool(* GiNaC::info_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.76 eval\_funcp\_5

```
typedef ex(* GiNaC::eval_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.77 evalf\_funcp\_5

```
typedef ex(* GiNaC::evalf_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.78 conjugate\_funcp\_5

```
typedef ex(* GiNaC::conjugate_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.79 real\_part\_funcp\_5

```
typedef ex(* GiNaC::real_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.80 imag\_part\_funcp\_5

```
typedef ex(* GiNaC::imag_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.81 expand\_funcp\_5

```
typedef ex(* GiNaC::expand_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.82 derivative\_funcp\_5

```
typedef ex(* GiNaC::derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.83 expl\_derivative\_funcp\_5

```
typedef ex(* GiNaC::expl_derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

#### 5.1.1.84 power\_funcp\_5

```
typedef ex(* GiNaC::power_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.85 series\_funcp\_5

```
typedef ex(* GiNaC::series_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.86 print\_funcp\_5

```
typedef void(* GiNaC::print_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const print_context &)
```

#### 5.1.1.87 info\_funcp\_5

```
typedef bool(* GiNaC::info_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex  
&, unsigned)
```

#### 5.1.1.88 eval\_funcp\_6

```
typedef ex(* GiNaC::eval_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &,  
const ex &)
```

#### 5.1.1.89 evalf\_funcp\_6

```
typedef ex(* GiNaC::evalf_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &)
```

#### 5.1.1.90 conjugate\_funcp\_6

```
typedef ex(* GiNaC::conjugate_funcp_6) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```

#### 5.1.1.91 real\_part\_funcp\_6

```
typedef ex(* GiNaC::real_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```

#### 5.1.1.92 imag\_part\_funcp\_6

```
typedef ex(* GiNaC::imag_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```



#### 5.1.1.93 expand\_funcp\_6

```
typedef ex(* GiNaC::expand_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, unsigned)
```

#### 5.1.1.94 derivative\_funcp\_6

```
typedef ex(* GiNaC::derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, unsigned)
```

#### 5.1.1.95 expl\_derivative\_funcp\_6

```
typedef ex(* GiNaC::expl_derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const symbol &)
```

#### 5.1.1.96 power\_funcp\_6

```
typedef ex(* GiNaC::power_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &)
```

#### 5.1.1.97 series\_funcp\_6

```
typedef ex(* GiNaC::series_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.98 print\_funcp\_6

```
typedef void(* GiNaC::print_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const print_context &)
```

#### 5.1.1.99 info\_funcp\_6

```
typedef bool(* GiNaC::info_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, unsigned)
```

#### 5.1.1.100 eval\_funcp\_7

```
typedef ex(* GiNaC::eval_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &)
```

#### 5.1.1.101 evalf\_funcp\_7

```
typedef ex(* GiNaC::evalf_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &)
```

#### 5.1.1.102 conjugate\_funcp\_7

```
typedef ex(* GiNaC::conjugate_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &)
```

#### 5.1.1.103 real\_part\_funcp\_7

```
typedef ex(* GiNaC::real_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &)
```

#### 5.1.1.104 imag\_part\_funcp\_7

```
typedef ex(* GiNaC::imag_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &)
```

#### 5.1.1.105 expand\_funcp\_7

```
typedef ex(* GiNaC::expand_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, unsigned)
```

#### 5.1.1.106 derivative\_funcp\_7

```
typedef ex(* GiNaC::derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.107 expl\_derivative\_funcp\_7

```
typedef ex(* GiNaC::expl_derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const symbol &)
```

#### 5.1.1.108 power\_funcp\_7

```
typedef ex(* GiNaC::power_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &)
```

#### 5.1.1.109 series\_funcp\_7

```
typedef ex(* GiNaC::series_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.110 print\_funcp\_7

```
typedef void(* GiNaC::print_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const print_context &)
```

#### 5.1.1.111 info\_funcp\_7

```
typedef bool(* GiNaC::info_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, unsigned)
```

#### 5.1.1.112 eval\_funcp\_8

```
typedef ex(* GiNaC::eval_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &)
```

#### 5.1.1.113 evalf\_funcp\_8

```
typedef ex(* GiNaC::evalf_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &)
```

#### 5.1.1.114 conjugate\_funcp\_8

```
typedef ex(* GiNaC::conjugate_funcp_8) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.115 real\_part\_funcp\_8

```
typedef ex(* GiNaC::real_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.116 imag\_part\_funcp\_8

```
typedef ex(* GiNaC::imag_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.117 expand\_funcp\_8

```
typedef ex(* GiNaC::expand_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.118 derivative\_funcp\_8

```
typedef ex(* GiNaC::derivative_funcp_8) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.119 expl\_derivative\_funcp\_8

```
typedef ex(* GiNaC::expl_derivative_funcp_8) (const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &, const ex &, const symbol &)
```

#### 5.1.1.120 power\_funcp\_8

```
typedef ex(* GiNaC::power_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.121 series\_funcp\_8

```
typedef ex(* GiNaC::series_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.122 print\_funcp\_8

```
typedef void(* GiNaC::print_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const print_context &)
```

#### 5.1.1.123 info\_funcp\_8

```
typedef bool(* GiNaC::info_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.124 eval\_funcp\_9

```
typedef ex(* GiNaC::eval_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.125 evalf\_funcp\_9

```
typedef ex(* GiNaC::evalf_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.126 conjugate\_funcp\_9

```
typedef ex(* GiNaC::conjugate_funcp_9) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.127 real\_part\_funcp\_9

```
typedef ex(* GiNaC::real_part_funcp_9) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.128 imag\_part\_funcp\_9**

```
typedef ex(* GiNaC::imag_part_funcp_9) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.129 expand\_funcp\_9**

```
typedef ex(* GiNaC::expand_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.130 derivative\_funcp\_9**

```
typedef ex(* GiNaC::derivative_funcp_9) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.131 expl\_derivative\_funcp\_9**

```
typedef ex(* GiNaC::expl_derivative_funcp_9) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

**5.1.1.132 power\_funcp\_9**

```
typedef ex(* GiNaC::power_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.133 series\_funcp\_9**

```
typedef ex(* GiNaC::series_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

**5.1.1.134 print\_funcp\_9**

```
typedef void(* GiNaC::print_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

#### 5.1.1.135 info\_funcp\_9

```
typedef bool(* GiNaC::info_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.136 eval\_funcp\_10

```
typedef ex(* GiNaC::eval_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.137 evalf\_funcp\_10

```
typedef ex(* GiNaC::evalf_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.138 conjugate\_funcp\_10

```
typedef ex(* GiNaC::conjugate_funcp_10) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.139 real\_part\_funcp\_10

```
typedef ex(* GiNaC::real_part_funcp_10) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.140 imag\_part\_funcp\_10

```
typedef ex(* GiNaC::imag_part_funcp_10) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.141 expand\_funcp\_10

```
typedef ex(* GiNaC::expand_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.142 derivative\_funcp\_10**

```
typedef ex(* GiNaC::derivative_funcp_10) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.143 expl\_derivative\_funcp\_10**

```
typedef ex(* GiNaC::expl_derivative_funcp_10) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

**5.1.1.144 power\_funcp\_10**

```
typedef ex(* GiNaC::power_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.145 series\_funcp\_10**

```
typedef ex(* GiNaC::series_funcp_10) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int,
unsigned)
```

**5.1.1.146 print\_funcp\_10**

```
typedef void(* GiNaC::print_funcp_10) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

**5.1.1.147 info\_funcp\_10**

```
typedef bool(* GiNaC::info_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.148 eval\_funcp\_11**

```
typedef ex(* GiNaC::eval_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```



#### 5.1.1.149 evalf\_funcp\_11

```
typedef ex(* GiNaC::evalf_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.150 conjugate\_funcp\_11

```
typedef ex(* GiNaC::conjugate_funcp_11) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.151 real\_part\_funcp\_11

```
typedef ex(* GiNaC::real_part_funcp_11) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.152 imag\_part\_funcp\_11

```
typedef ex(* GiNaC::imag_part_funcp_11) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.153 expand\_funcp\_11

```
typedef ex(* GiNaC::expand_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.154 derivative\_funcp\_11

```
typedef ex(* GiNaC::derivative_funcp_11) (const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.155 expl\_derivative\_funcp\_11

```
typedef ex(* GiNaC::expl_derivative_funcp_11) (const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
symbol &)
```

#### 5.1.1.156 power\_funcp\_11

```
typedef ex(* GiNaC::power_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.157 series\_funcp\_11

```
typedef ex(* GiNaC::series_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.158 print\_funcp\_11

```
typedef void(* GiNaC::print_funcp_11) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context  
&)
```

#### 5.1.1.159 info\_funcp\_11

```
typedef bool(* GiNaC::info_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.160 eval\_funcp\_12

```
typedef ex(* GiNaC::eval_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.161 evalf\_funcp\_12

```
typedef ex(* GiNaC::evalf_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.162 conjugate\_funcp\_12

```
typedef ex(* GiNaC::conjugate_funcp_12) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

Generated by Doxygen

#### 5.1.1.169 series\_funcp\_12

```
typedef ex(* GiNaC::series_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
relational &, int, unsigned)
```

#### 5.1.1.170 print\_funcp\_12

```
typedef void(* GiNaC::print_funcp_12) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
const print_context &)
```

#### 5.1.1.171 info\_funcp\_12

```
typedef bool(* GiNaC::info_funcp_12) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
unsigned)
```

#### 5.1.1.172 eval\_funcp\_13

```
typedef ex(* GiNaC::eval_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &)
```

#### 5.1.1.173 evalf\_funcp\_13

```
typedef ex(* GiNaC::evalf_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &)
```

#### 5.1.1.174 conjugate\_funcp\_13

```
typedef ex(* GiNaC::conjugate_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
const ex &)
```

**5.1.1.175 real\_part\_funcp\_13**

```
typedef ex(* GiNaC::real_part_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &  
const ex &)
```

**5.1.1.176 imag\_part\_funcp\_13**

```
typedef ex(* GiNaC::imag_part_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &  
const ex &)
```

**5.1.1.177 expand\_funcp\_13**

```
typedef ex(* GiNaC::expand_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, unsigned)
```

**5.1.1.178 derivative\_funcp\_13**

```
typedef ex(* GiNaC::derivative_funcp_13) (const ex &, const ex &, const ex &, const ex &  
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, unsigned)
```

**5.1.1.179 expl\_derivative\_funcp\_13**

```
typedef ex(* GiNaC::expl_derivative_funcp_13) (const ex &, const ex &, const ex &, const ex &  
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const symbol &)
```

**5.1.1.180 power\_funcp\_13**

```
typedef ex(* GiNaC::power_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```

**5.1.1.181 series\_funcp\_13**

```
typedef ex(* GiNaC::series_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const relational &, int, unsigned)
```

**5.1.1.182 print\_funcp\_13**

```
typedef void(* GiNaC::print_funcp_13) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const print_context &)
```

**5.1.1.183 info\_funcp\_13**

```
typedef bool(* GiNaC::info_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, unsigned)
```

**5.1.1.184 eval\_funcp\_14**

```
typedef ex(* GiNaC::eval_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

**5.1.1.185 evalf\_funcp\_14**

```
typedef ex(* GiNaC::evalf_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

**5.1.1.186 conjugate\_funcp\_14**

```
typedef ex(* GiNaC::conjugate_funcp_14) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

#### 5.1.1.187 real\_part\_funcp\_14

```
typedef ex(* GiNaC::real_part_funcp_14) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```

#### 5.1.1.188 imag\_part\_funcp\_14

```
typedef ex(* GiNaC::imag_part_funcp_14) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```

#### 5.1.1.189 expand\_funcp\_14

```
typedef ex(* GiNaC::expand_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, unsigned)
```

#### 5.1.1.190 derivative\_funcp\_14

```
typedef ex(* GiNaC::derivative_funcp_14) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
&, const ex &, const ex &, unsigned)
```

#### 5.1.1.191 expl\_derivative\_funcp\_14

```
typedef ex(* GiNaC::expl_derivative_funcp_14) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
&, const ex &, const ex &, const symbol &)
```

#### 5.1.1.192 power\_funcp\_14

```
typedef ex(* GiNaC::power_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &)
```

#### 5.1.1.193 series\_funcp\_14

```
typedef ex(* GiNaC::series_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.194 print\_funcp\_14

```
typedef void(* GiNaC::print_funcp_14) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const print_context &)
```

#### 5.1.1.195 info\_funcp\_14

```
typedef bool(* GiNaC::info_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, unsigned)
```

#### 5.1.1.196 eval\_funcp\_exvector

```
typedef ex(* GiNaC::eval_funcp_exvector) (const exvector &)
```

#### 5.1.1.197 evalf\_funcp\_exvector

```
typedef ex(* GiNaC::evalf_funcp_exvector) (const exvector &)
```

#### 5.1.1.198 conjugate\_funcp\_exvector

```
typedef ex(* GiNaC::conjugate_funcp_exvector) (const exvector &)
```

#### 5.1.1.199 real\_part\_funcp\_exvector

```
typedef ex(* GiNaC::real_part_funcp_exvector) (const exvector &)
```



**5.1.1.200 imag\_part\_funcp\_exvector**

```
typedef ex(* GiNaC::imag_part_funcp_exvector) (const exvector &)
```

**5.1.1.201 expand\_funcp\_exvector**

```
typedef ex(* GiNaC::expand_funcp_exvector) (const exvector &, unsigned)
```

**5.1.1.202 derivative\_funcp\_exvector**

```
typedef ex(* GiNaC::derivative_funcp_exvector) (const exvector &, unsigned)
```

**5.1.1.203 expl\_derivative\_funcp\_exvector**

```
typedef ex(* GiNaC::expl_derivative_funcp_exvector) (const exvector &, const symbol &)
```

**5.1.1.204 power\_funcp\_exvector**

```
typedef ex(* GiNaC::power_funcp_exvector) (const exvector &, const ex &)
```

**5.1.1.205 series\_funcp\_exvector**

```
typedef ex(* GiNaC::series_funcp_exvector) (const exvector &, const relational &, int, unsigned)
```

**5.1.1.206 print\_funcp\_exvector**

```
typedef void(* GiNaC::print_funcp_exvector) (const exvector &, const print\_context &)
```

**5.1.1.207 info\_funcp\_exvector**

```
typedef bool(* GiNaC::info_funcp_exvector) (const exvector &, unsigned)
```

**5.1.1.208 exhashmap**

```
template<typename T , class Hash = std::hash<ex>, class KeyEqual = std::equal_to<ex>, class
Allocator = std::allocator<std::pair<const ex, T>>>
using GiNaC::exhashmap = typedef std::unordered_map<ex, T, Hash, KeyEqual, Allocator>
```

**5.1.1.209 spmap**

```
typedef std::map<spmapkey, ex> GiNaC::spmap
```

**5.1.1.210 lookup\_map**

```
typedef map<error_and_integral, ex, error_and_integral_is_less> GiNaC::lookup_map
```

**5.1.1.211 lst**

```
typedef container< std::list > GiNaC::lst
```

**5.1.1.212 uintvector**

```
typedef std::vector<std::size_t> GiNaC::uintvector
```

**5.1.1.213 unsignedvector**

```
typedef std::vector<unsigned> GiNaC::unsignedvector
```

**5.1.1.214 exvectorvector**

```
typedef std::vector<exvector> GiNaC::exvectorvector
```

5.1.1.215 `sym_desc_vec`

```
typedef std::vector<sym_desc> GiNaC::sym_desc_vec
```

5.1.1.216 `digits_changed_callback`

```
typedef void(* GiNaC::digits_changed_callback) (long)
```

Function pointer to implement callbacks in the case 'Digits' gets changed.

Main purpose of such callbacks is to adjust look-up tables of certain functions to the new precision. Parameter contains the signed difference between new Digits and old Digits.

5.1.1.217 `print_context_class_info`

```
typedef class_info<print_context_options> GiNaC::print_context_class_info
```

5.1.1.218 `registered_class_info`

```
typedef class_info<registered_class_options> GiNaC::registered_class_info
```

## 5.1.2 Enumeration Type Documentation

## 5.1.2.1 anonymous enum

```
anonymous enum
```

Enumerator

callback_registered	
---------------------	--

## 5.1.3 Function Documentation

5.1.3.1 `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [1/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    add ,
```

```

        expairseq ,
        print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
print_tree. print_func< print_python_repr > &::do_print_python_repr )

```

#### 5.1.3.2 GINAC\_BIND\_UNARCHIVER() [1/49]

```

GiNaC::GINAC_BIND_UNARCHIVER (
    add )

```

#### 5.1.3.3 GINAC\_DECLARE\_UNARCHIVER() [1/51]

```

GiNaC::GINAC_DECLARE_UNARCHIVER (
    add )

```

#### 5.1.3.4 write\_unsigned()

```

static void GiNaC::write_unsigned (
    std::ostream & os,
    unsigned val ) [static]

```

Write unsigned integer quantity to stream.

Referenced by operator<<().

#### 5.1.3.5 read\_unsigned()

```

static unsigned GiNaC::read_unsigned (
    std::istream & is ) [static]

```

Read unsigned integer quantity from stream.

Referenced by operator>>().

#### 5.1.3.6 operator<<() [1/16]

```

std::ostream& GiNaC::operator<< (
    std::ostream & os,
    const archive_node & n )

```

Write [archive\\_node](#) to binary data stream.

References [n](#), and [write\\_unsigned\(\)](#).

#### 5.1.3.7 operator<<() [2/16]

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const archive & ar )
```

Write archive to binary data stream.

References GiNaC::archive::atoms, GiNaC::archive::exprs, GINACLIB\_ARCHIVE\_VERSION, GiNaC::archive::nodes, and write\_unsigned().

#### 5.1.3.8 operator>>() [1/3]

```
std::istream& GiNaC::operator>> (
    std::istream & is,
    archive_node & n )
```

Read archive\_node from binary data stream.

References n, and read\_unsigned().

#### 5.1.3.9 operator>>() [2/3]

```
std::istream & GiNaC::operator>> (
    std::istream & is,
    archive & ar )
```

Read archive from binary data stream.

References GiNaC::archive::atoms, GiNaC::archive::exprs, GINACLIB\_ARCHIVE\_AGE, GINACLIB\_ARCHIVE\_VERSION, GiNaC::archive::inverse\_atoms, GiNaC::archive::nodes, and read\_unsigned().

#### 5.1.3.10 find\_factory\_fcn()

```
static synthesize_func GiNaC::find_factory_fcn (
    const std::string & name ) [static]
```

References GiNaC::unarchive\_table\_t::find().

Referenced by GiNaC::archive\_node::unarchive().

#### 5.1.3.11 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [2/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    basic ,
    void ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print←
_tree. print_func< print_python_repr > &::do_print_python_repr )
```

basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by duplicate()), so it can copy the tinfo\_key and the hash value.

#### 5.1.3.12 is\_a() [1/3]

```
template<class T >
bool GiNaC::is_a (
    const basic & obj ) [inline]
```

Check if obj is a T, including base classes.

Referenced by GiNaC::container< C >::subs().

#### 5.1.3.13 is\_exactly\_a() [1/2]

```
template<class T >
bool GiNaC::is_exactly_a (
    const basic & obj ) [inline]
```

Check if obj is a T, not including base classes.

#### 5.1.3.14 dynallocate() [1/2]

```
template<class B , typename... Args>
B& GiNaC::dynallocate (
    Args &&... args ) [inline]
```

Constructs a new (class basic or derived) B object on the heap.

This function picks the object's ctor based on the given argument types.

This helps the constructor of ex from basic (or a derived class B) because then the constructor doesn't have to duplicate the object onto the heap. See [ex::construct\\_from\\_basic\(const basic &\)](#) for more information.

References GiNaC::status\_flags::dynallocated.

**5.1.3.15** `dynallocate()` [2/2]

```
template<class B >
B& GiNaC::dynallocate (
    std::initializer_list< ex > il ) [inline]
```

Constructs a new (class basic or derived) B object on the heap.

This function is needed for [GiNaC](#) classes which have public ctors from initializer lists of expressions (which are not a type and not captured by the variadic template version).

References `GiNaC::status_flags::dynallocated`.

**5.1.3.16** `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [3/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    clifford ,
    indexed ,
    print_func< print_dflt > &::do_print_dflt. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree )
```

**5.1.3.17** `print_func< print_dflt >()` [1/3]

```
GiNaC::print_func< print_dflt > (
    &diracone::do_print ) &
```

**5.1.3.18** `GINAC_BIND_UNARCHIVER()` [2/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    clifford )
```

**5.1.3.19** `GINAC_BIND_UNARCHIVER()` [3/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    cliffordunit )
```

**5.1.3.20** `GINAC_BIND_UNARCHIVER()` [4/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracone )
```

**5.1.3.21 GINAC\_BIND\_UNARCHIVER()** [5/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgamma )
```

**5.1.3.22 GINAC\_BIND\_UNARCHIVER()** [6/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgamma5 )
```

**5.1.3.23 GINAC\_BIND\_UNARCHIVER()** [7/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgammaL )
```

**5.1.3.24 GINAC\_BIND\_UNARCHIVER()** [8/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgammaR )
```

**5.1.3.25 is\_dirac\_slash()**

```
static bool GiNaC::is_dirac_slash (
    const ex & seq0 ) [static]
```

Referenced by `GiNaC::clifford::do_print_dflt()`, and `GiNaC::clifford::do_print_latex()`.

**5.1.3.26 base\_and\_index()**

```
static void GiNaC::base_and_index (
    const ex & c,
    ex & b,
    ex & i ) [static]
```

This function decomposes  $\gamma_\mu \rightarrow (1, \mu)$  and  $a \rightarrow (a.ix, ix)$

**5.1.3.27 dirac\_ONE()**

```
ex GiNaC::dirac_ONE (
    unsigned char r1 = 0 )
```

Create a Clifford unity object.



## Parameters

<i>rl</i>	Representation label
-----------	----------------------

## Returns

newly constructed object

Referenced by `GiNaC::add::coeff()`.

5.1.3.28 `get_dim_uint()`

```
static unsigned GiNaC::get_dim_uint (
    const ex & e ) [static]
```

5.1.3.29 `clifford_unit()`

```
ex GiNaC::clifford_unit (
    const ex & mu,
    const ex & metr,
    unsigned char rl = 0 )
```

Create a Clifford unit object.

## Parameters

<i>mu</i>	Index (must be of class <code>varidx</code> or a derived class)
<i>metr</i>	Metric (should be indexed, <code>tensmetric</code> or a derived class, or a matrix)
<i>rl</i>	Representation label

## Returns

newly constructed Clifford unit object

5.1.3.30 `dirac_gamma()`

```
ex GiNaC::dirac_gamma (
    const ex & mu,
    unsigned char rl = 0 )
```

Create a Dirac gamma object.

**Parameters**

<i>mu</i>	Index (must be of class <code>varidx</code> or a derived class)
<i>rl</i>	Representation label

**Returns**

newly constructed gamma object

**5.1.3.31 `dirac_gamma5()`**

```
ex GiNaC::dirac_gamma5 (
    unsigned char rl = 0 )
```

Create a Dirac gamma5 object.

**Parameters**

<i>rl</i>	Representation label
-----------	----------------------

**Returns**

newly constructed object

**5.1.3.32 `dirac_gammaL()`**

```
ex GiNaC::dirac_gammaL (
    unsigned char rl = 0 )
```

Create a Dirac gammaL object.

**Parameters**

<i>rl</i>	Representation label
-----------	----------------------

**Returns**

newly constructed object

5.1.3.33 `dirac_gammaR()`

```
ex GiNaC::dirac_gammaR (
    unsigned char rl = 0 )
```

Create a Dirac gammaR object.

## Parameters

<i>rl</i>	Representation label
-----------	----------------------

## Returns

newly constructed object

5.1.3.34 `dirac_slash()`

```
ex GiNaC::dirac_slash (
    const ex & e,
    const ex & dim,
    unsigned char rl = 0 )
```

Create a term of the form  $e_\mu * \gamma_\mu$  with a unique index  $\mu$ .

## Parameters

<i>e</i>	Original expression
<i>dim</i>	Dimension of index
<i>rl</i>	Representation label

5.1.3.35 `get_representation_label()` [1/2]

```
static unsigned char GiNaC::get_representation_label (
    const return_type_t & ti ) [static]
```

Extract representation label from tinfo key (as returned by `return_type_tinfo()`).

Referenced by `color_trace()`, `GiNaC::su3f::contract_with()`, and `GiNaC::su3d::contract_with()`.

5.1.3.36 `trace_string()`

```
static ex GiNaC::trace_string (
    exvector::const_iterator ix,
    size_t num ) [static]
```

Take trace of a string of an even number of Dirac gammas given a vector of indices.

**5.1.3.37** `dirac_trace()` [1/3]

```
ex GiNaC::dirac_trace (
    const ex & e,
    const std::set< unsigned char > & rls,
    const ex & trONE = 4 )
```

Calculate dirac traces over the specified set of representation labels.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when  $\gamma_5$  is involved.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rls</i>	Set of representation labels
<i>trONE</i>	Expression to be returned as the trace of the unit matrix

**5.1.3.38** `dirac_trace()` [2/3]

```
ex GiNaC::dirac_trace (
    const ex & e,
    const lst & rll,
    const ex & trONE = 4 )
```

Calculate dirac traces over the specified list of representation labels.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when  $\gamma_5$  is involved.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rll</i>	List of representation labels
<i>trONE</i>	Expression to be returned as the trace of the unit matrix

**5.1.3.39** `dirac_trace()` [3/3]

```
ex GiNaC::dirac_trace (
    const ex & e,
    unsigned char rl = 0,
    const ex & trONE = 4 )
```

Calculate the trace of an expression containing gamma objects with a specified representation label.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when  $\gamma_5$  is involved.

## Parameters

<i>e</i>	Expression to take the trace of
<i>rl</i>	Representation label
<i>trONE</i>	Expression to be returned as the trace of the unit matrix

## 5.1.3.40 canonicalize\_clifford()

```
ex GiNaC::canonicalize_clifford (
    const ex & e )
```

Bring all products of clifford objects in an expression into a canonical order.

This is not necessarily the most simple form but it will allow to check two expressions for equality.

## 5.1.3.41 clifford\_star\_bar()

```
ex GiNaC::clifford_star_bar (
    const ex & e,
    bool do_bar,
    unsigned options )
```

An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).

Referenced by [clifford\\_bar\(\)](#), and [clifford\\_star\(\)](#).

## 5.1.3.42 clifford\_prime()

```
ex GiNaC::clifford_prime (
    const ex & e )
```

Automorphism of the Clifford algebra, simply changes signs of all clifford units.

## 5.1.3.43 remove\_dirac\_ONE()

```
ex GiNaC::remove_dirac_ONE (
    const ex & e,
    unsigned char rl = 0,
    unsigned options = 0 )
```

Replaces `dirac_ONE`'s (with a `representation_label` no less than `rl`) in `e` with 1.

For the default value `rl = 0` remove all of them. Aborts if `e` contains any `clifford_unit` with `representation_label` to be removed.

## Parameters

<i>e</i>	Expression to be processed
<i>rl</i>	Value of representation label
<i>options</i>	Defines some internal use

## 5.1.3.44 clifford\_max\_label()

```
int GiNaC::clifford_max_label (
    const ex & e,
    bool ignore_ONE = false )
```

Returns the maximal representation label of a clifford object if e contains at least one, otherwise returns -1.

## Parameters

<i>e</i>	Expression to be processed defines if clifford_ONE should be ignored in the search
----------	--

Referenced by GiNaC::add::coeff().

## 5.1.3.45 clifford\_norm()

```
ex GiNaC::clifford_norm (
    const ex & e )
```

Calculation of the norm in the Clifford algebra.

## 5.1.3.46 clifford\_inverse()

```
ex GiNaC::clifford_inverse (
    const ex & e )
```

Calculation of the inverse in the Clifford algebra.

## 5.1.3.47 lst\_to\_clifford() [1/2]

```
ex GiNaC::lst_to_clifford (
    const ex & v,
    const ex & mu,
    const ex & metr,
    unsigned char rl = 0 )
```

List or vector conversion into the Clifford vector.

## Parameters

<i>v</i>	List or vector of coordinates
<i>mu</i>	Index (must be of class varidx or a derived class)
<i>metr</i>	Metric (should be indexed, tensmetric or a derived class, or a matrix)
<i>rl</i>	Representation label
<i>e</i>	Clifford unit object

## Returns

Clifford vector with given components

5.1.3.48 `lst_to_clifford()` [2/2]

```
ex GiNaC::lst_to_clifford (
    const ex & v,
    const ex & e )
```

5.1.3.49 `get_clifford_comp()`

```
static ex GiNaC::get_clifford_comp (
    const ex & e,
    const ex & c,
    bool root = true ) [static]
```

Auxiliary structure to define a function for stripping one Clifford unit from vectors.

Used in [clifford\\_to\\_lst\(\)](#).

5.1.3.50 `clifford_to_lst()`

```
lst GiNaC::clifford_to_lst (
    const ex & e,
    const ex & c,
    bool algebraic = true )
```

An inverse function to [lst\\_to\\_clifford\(\)](#).

For given Clifford vector extracts its components with respect to given Clifford unit. Obtained components may contain Clifford units with a different metric. Extraction is based on the algebraic formula  $(e * c.i + c.i * e) / \text{pow}(e.i, 2)$  for non-degenerate cases (i.e. neither  $\text{pow}(e.i, 2) = 0$ ).

## Parameters

<i>e</i>	Clifford expression to be decomposed into components
<i>c</i>	Clifford unit defining the metric for splitting (should have numeric dimension of indices)
<i>algebraic</i>	Use algebraic or symbolic algorithm for extractions

**Returns**

List of components of a Clifford vector

**5.1.3.51 clifford\_moebius\_map()** [1/2]

```
ex GiNaC::clifford_moebius_map (
    const ex & a,
    const ex & b,
    const ex & c,
    const ex & d,
    const ex & v,
    const ex & G,
    unsigned char rl = 0 )
```

Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b\c d) in linear spaces with arbitrary signature.

The expression is  $(a * x + b)/(c * x + d)$ , where x is a vector build from list v with metric G. (see Jan Cnops. An introduction to {D}irac operators on manifolds, v.24 of Progress in Mathematical Physics. Birkhauser Boston Inc., Boston, MA, 2002.)

**Parameters**

<i>a</i>	(1,1) entry of the defining matrix
<i>b</i>	(1,2) entry of the defining matrix
<i>c</i>	(2,1) entry of the defining matrix
<i>d</i>	(2,2) entry of the defining matrix
<i>v</i>	Vector to be transformed
<i>G</i>	Metric of the surrounding space, may be a Clifford unit then the next parameter is ignored
<i>rl</i>	Representation label

**Returns**

List of components of the transformed vector

**5.1.3.52 clifford\_moebius\_map()** [2/2]

```
ex GiNaC::clifford_moebius_map (
    const ex & M,
    const ex & v,
    const ex & G,
    unsigned char rl = 0 )
```

The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.



## Parameters

$M$	the defining matrix
$v$	Vector to be transformed
$G$	Metric of the surrounding space, may be a Clifford unit then the next parameter is ignored
$rl$	Representation label

## Returns

List of components of the transformed vector

## 5.1.3.53 GINAC\_DECLARE\_UNARCHIVER() [2/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    clifford )
```

## 5.1.3.54 GINAC\_DECLARE\_UNARCHIVER() [3/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracone )
```

## 5.1.3.55 GINAC\_DECLARE\_UNARCHIVER() [4/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    cliffordunit )
```

## 5.1.3.56 GINAC\_DECLARE\_UNARCHIVER() [5/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgamma )
```

## 5.1.3.57 GINAC\_DECLARE\_UNARCHIVER() [6/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgamma5 )
```

**5.1.3.58 GINAC\_DECLARE\_UNARCHIVER()** [7/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgammaL )
```

**5.1.3.59 GINAC\_DECLARE\_UNARCHIVER()** [8/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgammaR )
```

**5.1.3.60 is\_clifford\_tinfo()**

```
bool GiNaC::is_clifford_tinfo (
    const return_type_t & ti ) [inline]
```

Check whether a given [return\\_type\\_t](#) object (as returned by `return_type_tinfo()`) is that of a clifford object (with an arbitrary representation label).

**Parameters**

<i>ti</i>	tinfo key
-----------	-----------

References `GiNaC::return_type_t::tinfo`.

Referenced by `GiNaC::ncmul::conjugate()`.

**5.1.3.61 clifford\_bar()**

```
ex GiNaC::clifford_bar (
    const ex & e ) [inline]
```

Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.

References `clifford_star_bar()`.

**5.1.3.62 clifford\_star()**

```
ex GiNaC::clifford_star (
    const ex & e ) [inline]
```

Reversion of the Clifford algebra, reverse the order of all clifford units in `ncmul`.

References `clifford_star_bar()`.

**5.1.3.63 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [4/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    su3one ,
    tensor ,
    print_func< print_dflt > &::do_print. print_func< print_latex > &::do_print_↵
    latex )
```

**5.1.3.64 print\_func< print\_dflt >()** [2/3]

```
GiNaC::print_func< print_dflt > (
    &su3t::do_print ) &
```

**5.1.3.65 GINAC\_BIND\_UNARCHIVER()** [9/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    color )
```

**5.1.3.66 GINAC\_BIND\_UNARCHIVER()** [10/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3one )
```

**5.1.3.67 GINAC\_BIND\_UNARCHIVER()** [11/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3t )
```

**5.1.3.68 GINAC\_BIND\_UNARCHIVER()** [12/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3f )
```

### 5.1.3.69 GINAC\_BIND\_UNARCHIVER() [13/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3d )
```

### 5.1.3.70 permute\_free\_index\_to\_front()

```
static ex GiNaC::permute_free_index_to_front (
    const exvector & iv3,
    const exvector & iv2,
    int & sig ) [static]
```

Given a vector *iv3* of three indices and a vector *iv2* of two indices that is a subset of *iv3*, return the (free) index that is in *iv3* but not in *iv2* and the sign introduced by permuting that index to the front.

#### Parameters

<i>iv3</i>	Vector of 3 indices
<i>iv2</i>	Vector of 2 indices, must be a subset of <i>iv3</i>
<i>sig</i>	Returns sign introduced by index permutation

#### Returns

the free index (the one that is in *iv3* but not in *iv2*)

References GINAC\_ASSERT, and TEST\_PERMUTATION.

Referenced by GiNaC::su3f::contract\_with(), and GiNaC::su3d::contract\_with().

### 5.1.3.71 color\_ONE()

```
ex GiNaC::color_ONE (
    unsigned char r1 = 0 )
```

Create the su(3) unity element.

This is an indexed object, although it has no indices.

#### Parameters

<i>r1</i>	Representation label
-----------	----------------------

#### Returns

newly constructed unity element

Referenced by `GiNaC::su3t::contract_with()`.

#### 5.1.3.72 `color_T()`

```
ex GiNaC::color_T (
    const ex & a,
    unsigned char rl = 0 )
```

Create an  $su(3)$  generator.

##### Parameters

<i>a</i>	Index
<i>rl</i>	Representation label

##### Returns

newly constructed unity generator

Referenced by `color_trace()`, `GiNaC::su3f::contract_with()`, and `GiNaC::su3d::contract_with()`.

#### 5.1.3.73 `color_f()`

```
ex GiNaC::color_f (
    const ex & a,
    const ex & b,
    const ex & c )
```

Create an  $su(3)$  antisymmetric structure constant.

##### Parameters

<i>a</i>	First index
<i>b</i>	Second index
<i>c</i>	Third index

##### Returns

newly constructed structure constant

References `antisymmetric3()`, and `c`.

Referenced by `color_h()`.

5.1.3.74 `color_d()`

```
ex GiNaC::color_d (
    const ex & a,
    const ex & b,
    const ex & c )
```

Create an  $su(3)$  symmetric structure constant.

## Parameters

<i>a</i>	First index
<i>b</i>	Second index
<i>c</i>	Third index

## Returns

newly constructed structure constant

References `c`, and `symmetric3()`.

Referenced by `color_h()`.

5.1.3.75 `color_h()`

```
ex GiNaC::color_h (
    const ex & a,
    const ex & b,
    const ex & c )
```

This returns the linear combination  $d.a.b.c + l.f.a.b.c$ .

References `c`, `color_d()`, `color_f()`, and `l`.

Referenced by `color_trace()`.

5.1.3.76 `is_color_tinfo()`

```
static bool GiNaC::is_color_tinfo (
    const return_type_t & ti ) [static]
```

Check whether a given tinfo key (as returned by `return_type_tinfo()`) is that of a color object (with an arbitrary representation label).

References `GiNaC::return_type_t::tinfo`.

Referenced by `color_trace()`.

**5.1.3.77** `get_representation_label()` [2/2]

```
static unsigned char GiNaC::get_representation_label (
    const return_type_t & ti ) [static]
```

Extract representation label from tinfo key (as returned by `return_type_tinfo()`).

References `GiNaC::return_type_t::rl`.

**5.1.3.78** `color_trace()` [1/3]

```
ex GiNaC::color_trace (
    const ex & e,
    const std::set< unsigned char > & rls )
```

Calculate color traces over the specified set of representation labels.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rls</i>	Set of representation labels

References `_ex0`, `_ex1`, `_ex3`, `color_h()`, `color_T()`, `delta_tensor()`, `GiNaC::ex::expand()`, `get_representation_label()`, `is_color_tinfo()`, `GiNaC::ex::map()`, `GiNaC::ex::nops()`, `GiNaC::ex::op()`, and `GiNaC::ex::return_type_tinfo()`.

Referenced by `color_trace()`, and `GiNaC::su3t::contract_with()`.

**5.1.3.79** `color_trace()` [2/3]

```
ex GiNaC::color_trace (
    const ex & e,
    const lst & rl1 )
```

Calculate color traces over the specified list of representation labels.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rl1</i>	List of representation labels

References `color_trace()`, and `GiNaC::info_flags::nonnegint`.

**5.1.3.80 color\_trace()** [3/3]

```
ex GiNaC::color_trace (
    const ex & e,
    unsigned char rl = 0 )
```

Calculate the trace of an expression containing color objects with a specified representation label.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rl</i>	Representation label

References color\_trace().

**5.1.3.81 GINAC\_DECLARE\_UNARCHIVER()** [9/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    color )
```

**5.1.3.82 GINAC\_DECLARE\_UNARCHIVER()** [10/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3one )
```

**5.1.3.83 GINAC\_DECLARE\_UNARCHIVER()** [11/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3t )
```

**5.1.3.84 GINAC\_DECLARE\_UNARCHIVER()** [12/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3f )
```



**5.1.3.85 GINAC\_DECLARE\_UNARCHIVER()** [13/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3d )
```

**5.1.3.86 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [5/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    constant ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python_repr > &↵
::do_print_python_repr ) const
```

References `GiNaC::status_flags::evaluated`, and `GiNaC::status_flags::expanded`.

**5.1.3.87 GINAC\_BIND\_UNARCHIVER()** [14/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    constant )
```

**5.1.3.88 GINAC\_DECLARE\_UNARCHIVER()** [14/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    constant )
```

**5.1.3.89 crc32()**

```
static unsigned GiNaC::crc32 (
    const char * c,
    const unsigned len,
    const unsigned crcinit ) [static]
```

References `c`, `crctab`, and `len`.

### 5.1.3.90 are\_ex\_trivially\_equal()

```
bool GiNaC::are_ex_trivially_equal (
    const ex & e1,
    const ex & e2 ) [inline]
```

Compare two objects of class quickly without doing a deep tree traversal.

#### Returns

"true" if they are equal "false" if equality cannot be established quickly (e1 and e2 may still be equal, in this case).

References GiNaC::ex::bp.

Referenced by GiNaC::integral::conjugate(), GiNaC::pseries::conjugate(), GiNaC::matrix::conjugate(), GiNaC::add::conjugate(), GiNaC::mul::conjugate(), GiNaC::power::conjugate(), GiNaC::expair::conjugate(), GiNaC::container< C >::conjugate(), GiNaC::pseries::eval\_integ(), GiNaC::integral::evalf(), GiNaC::pseries::evalm(), GiNaC::integral::expand(), GiNaC::power::expand(), GiNaC::indexed::expand(), GiNaC::ncmul::expandchildren(), GiNaC::mul::expandchildren(), GiNaC::make\_flat\_inserter::make\_flat\_inserter(), GiNaC::idx::map(), GiNaC::power::map(), GiNaC::relational::map(), GiNaC::basic::map(), GiNaC::const\_iterator::operator==(), GiNaC::internal::\_iter\_rep::operator==(), GiNaC::idx::subs(), GiNaC::relational::subs(), GiNaC::power::subs(), GiNaC::clifford::subs(), GiNaC::basic::subs(), and GiNaC::container< C >::subschildren().

### 5.1.3.91 operator<<() [3/16]

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const exvector & e )
```

References get\_print\_context().

### 5.1.3.92 operator<<() [4/16]

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const exset & e )
```

References get\_print\_context().

### 5.1.3.93 operator<<() [5/16]

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const exmap & e )
```

References get\_print\_context().

**5.1.3.94 nops()** [1/2]

```
size_t GiNaC::nops (
    const ex & thisex ) [inline]
```

References `GiNaC::ex::nops()`.

Referenced by `GiNaC::mul::algebraic_subs_mul()`, `GiNaC::basic::calchash()`, `GiNaC::basic::derivative()`, `GiNaC::mul::do_print_python_repr()`, `GiNaC::container< C >::do_print_tree()`, `GiNaC::basic::do_print_tree()`, `GiNaC::basic::eval_integ()`, `GiNaC::basic::evalf()`, `GiNaC::basic::evalm()`, `GiNaC::basic::expand()`, `GiNaC::mul::has()`, `GiNaC::basic::has()`, `GiNaC::container< C >::let_op()`, `GiNaC::basic::map()`, `GiNaC::basic::match()`, `GiNaC::clifford::nops()`, `GiNaC::container< C >::op()`, and `GiNaC::basic::subs()`.

**5.1.3.95 expand()** [1/2]

```
ex GiNaC::expand (
    const ex & thisex,
    unsigned options = 0 ) [inline]
```

References `GiNaC::ex::expand()`, and `options`.

Referenced by `GiNaC::basic::collect()`, `divide()`, `divide_in_z()`, `GiNaC::indexed::expand()`, `GiNaC::structure< T, ComparisonPolicy >::expand()`, `GiNaC::function::expand()`, `gcd()`, `gcd_pf_pow()`, `log_expand()`, `prem()`, `quo()`, `rem()`, and `sprem()`.

**5.1.3.96 conjugate()**

```
ex GiNaC::conjugate (
    const ex & thisex ) [inline]
```

References `GiNaC::ex::conjugate()`.

Referenced by `GiNaC::numeric::conjugate()`, `conjugate_evalf()`, and `conjugate_expl_derivative()`.

**5.1.3.97 real\_part()**

```
ex GiNaC::real_part (
    const ex & thisex ) [inline]
```

References `GiNaC::ex::real_part()`.

Referenced by `cos_imag_part()`, `cos_real_part()`, `cosh_imag_part()`, `cosh_real_part()`, `exp_imag_part()`, `exp_real_part()`, `log_imag_part()`, `real_part_expl_derivative()`, `sin_imag_part()`, `sin_real_part()`, `sinh_imag_part()`, `sinh_real_part()`, `tan_imag_part()`, `tan_real_part()`, `tanh_imag_part()`, and `tanh_real_part()`.

### 5.1.3.98 `imag_part()`

```
ex GiNaC::imag_part (
    const ex & thisex ) [inline]
```

References `GiNaC::ex::imag_part()`.

Referenced by `cos_imag_part()`, `cos_real_part()`, `cosh_imag_part()`, `cosh_real_part()`, `exp_imag_part()`, `exp_real_part()`, `imag_part_expl_derivative()`, `log_imag_part()`, `sin_imag_part()`, `sin_real_part()`, `sinh_imag_part()`, `sinh_real_part()`, `tan_imag_part()`, `tan_real_part()`, `tanh_imag_part()`, and `tanh_real_part()`.

### 5.1.3.99 `has()`

```
bool GiNaC::has (
    const ex & thisex,
    const ex & pattern,
    unsigned options = 0 ) [inline]
```

References `GiNaC::ex::has()`, and `options`.

Referenced by `GiNaC::mul::has()`, `GiNaC::structure< T, ComparisonPolicy >::has()`, and `GiNaC::basic::is_polynomial()`.

### 5.1.3.100 `find()`

```
bool GiNaC::find (
    const ex & thisex,
    const ex & pattern,
    exset & found ) [inline]
```

References `GiNaC::ex::find()`.

### 5.1.3.101 `is_polynomial()`

```
bool GiNaC::is_polynomial (
    const ex & thisex,
    const ex & vars ) [inline]
```

References `GiNaC::ex::is_polynomial()`.

## 5.1.3.102 degree()

```
int GiNaC::degree (
    const ex & thisex,
    const ex & s ) [inline]
```

References GiNaC::ex::degree().

Referenced by GiNaC::basic::collect(), GiNaC::structure< T, ComparisonPolicy >::degree(), and replace\_with\_↵symbol().

## 5.1.3.103 ldegree()

```
int GiNaC::ldegree (
    const ex & thisex,
    const ex & s ) [inline]
```

References GiNaC::ex::ldegree().

Referenced by GiNaC::basic::collect(), and GiNaC::structure< T, ComparisonPolicy >::ldegree().

## 5.1.3.104 coeff()

```
ex GiNaC::coeff (
    const ex & thisex,
    const ex & s,
    int n = 1 ) [inline]
```

References GiNaC::ex::coeff(), and n.

Referenced by GiNaC::mul::coeff(), GiNaC::structure< T, ComparisonPolicy >::coeff(), GiNaC::basic::collect(), GiNaC::mul::eval(), GiNaC::make\_flat\_inserter::handle\_factor(), iterated\_integral\_evalf\_impl(), log\_series(), simplify\_indexed(), and sqrfree\_parfrac().

## 5.1.3.105 numer() [1/2]

```
ex GiNaC::numer (
    const ex & thisex ) [inline]
```

References GiNaC::ex::numer().

Referenced by decomp\_rational(), GiNaC::add::do\_print\_csrc(), GiNaC::power::imag\_part(), GiNaC::add::integer↵\_content(), print\_real\_csrc(), GiNaC::power::real\_part(), replace\_with\_symbol(), and sqrfree\_parfrac().

**5.1.3.106** `denom()` [1/2]

```
ex GiNaC::denom (
    const ex & thisex ) [inline]
```

References `GiNaC::ex::denom()`.

Referenced by `decomp_rational()`, `GiNaC::add::do_print_csrc()`, `GiNaC::add::integer_content()`, `print_real_csrc()`, `replace_with_symbol()`, and `sqrfree_parfrac()`.

**5.1.3.107** `numer_denom()`

```
ex GiNaC::numer_denom (
    const ex & thisex ) [inline]
```

References `GiNaC::ex::numer_denom()`.

Referenced by `decomp_rational()`, and `sqrfree_parfrac()`.

**5.1.3.108** `normal()`

```
ex GiNaC::normal (
    const ex & thisex ) [inline]
```

References `GiNaC::ex::normal()`.

Referenced by `fsolve()`, `GiNaC::structure< T, ComparisonPolicy >::normal()`, `GiNaC::normal_map_function↔::operator()`, and `replace_with_symbol()`.

**5.1.3.109** `to_rational()`

```
ex GiNaC::to_rational (
    const ex & thisex,
    exmap & repl ) [inline]
```

References `GiNaC::ex::to_rational()`.

Referenced by `GiNaC::structure< T, ComparisonPolicy >::to_rational()`.

#### 5.1.3.110 to\_polynomial()

```
ex GiNaC::to_polynomial (
    const ex & thisex,
    exmap & repl ) [inline]
```

References GiNaC::ex::to\_polynomial().

Referenced by GiNaC::structure< T, ComparisonPolicy >::to\_polynomial().

#### 5.1.3.111 collect()

```
ex GiNaC::collect (
    const ex & thisex,
    const ex & s,
    bool distributed = false ) [inline]
```

References GiNaC::ex::collect().

Referenced by GiNaC::structure< T, ComparisonPolicy >::collect(), and GiNaC::basic::collect().

#### 5.1.3.112 eval()

```
ex GiNaC::eval (
    const ex & thisex ) [inline]
```

References GiNaC::ex::eval().

#### 5.1.3.113 evalf() [1/2]

```
ex GiNaC::evalf (
    const ex & thisex ) [inline]
```

References GiNaC::ex::evalf().

Referenced by beta\_eval(), eta\_evalf(), H\_evalf(), GiNaC::evalf\_map\_function::operator()(), and zeta2\_evalf().

#### 5.1.3.114 evalm()

```
ex GiNaC::evalm (
    const ex & thisex ) [inline]
```

References GiNaC::ex::evalm().

Referenced by GiNaC::structure< T, ComparisonPolicy >::evalm(), and GiNaC::evalm\_map\_function::operator()().

**5.1.3.115 eval\_integ()**

```
ex GiNaC::eval_integ (
    const ex & thisex ) [inline]
```

References GiNaC::ex::eval\_integ().

Referenced by GiNaC::eval\_integ\_map\_function::operator()().

**5.1.3.116 diff()**

```
ex GiNaC::diff (
    const ex & thisex,
    const symbol & s,
    unsigned nth = 1 ) [inline]
```

References GiNaC::ex::diff().

Referenced by GiNaC::derivative\_map\_function::operator()().

**5.1.3.117 series()**

```
ex GiNaC::series (
    const ex & thisex,
    const ex & r,
    int order,
    unsigned options = 0 ) [inline]
```

References options, order, r, and GiNaC::ex::series().

Referenced by atan\_series(), atanh\_series(), lgamma\_series(), Li2\_series(), log\_series(), psi1\_series(), GiNaC::structure< T, ComparisonPolicy >::series(), and tgamma\_series().

**5.1.3.118 match()**

```
bool GiNaC::match (
    const ex & thisex,
    const ex & pattern,
    exmap & repl_lst ) [inline]
```

References GiNaC::ex::match().

Referenced by GiNaC::basic::has(), GiNaC::structure< T, ComparisonPolicy >::match(), and GiNaC::basic::substitute\_one\_level().



**5.1.3.119 simplify\_indexed()** [1/3]

```
ex GiNaC::simplify_indexed (
    const ex & thisex,
    unsigned options = 0 ) [inline]
```

References options, and GiNaC::ex::simplify\_indexed().

Referenced by GiNaC::ex::simplify\_indexed(), simplify\_indexed(), and simplify\_indexed\_product().

**5.1.3.120 simplify\_indexed()** [2/3]

```
ex GiNaC::simplify_indexed (
    const ex & thisex,
    const scalar_products & sp,
    unsigned options = 0 ) [inline]
```

References options, and GiNaC::ex::simplify\_indexed().

**5.1.3.121 symmetrize()** [1/4]

```
ex GiNaC::symmetrize (
    const ex & thisex ) [inline]
```

References GiNaC::ex::symmetrize().

Referenced by idx\_symmetrization(), symmetrize(), GiNaC::ex::symmetrize(), and symmetrize\_cyclic().

**5.1.3.122 symmetrize()** [2/4]

```
ex GiNaC::symmetrize (
    const ex & thisex,
    const lst & l ) [inline]
```

References GiNaC::ex::symmetrize().

**5.1.3.123 antisymmetrize()** [1/4]

```
ex GiNaC::antisymmetrize (
    const ex & thisex ) [inline]
```

References GiNaC::ex::antisymmetrize().

Referenced by antisymmetrize(), and GiNaC::ex::antisymmetrize().

**5.1.3.124 antisymmetrize()** [2/4]

```
ex GiNaC::antisymmetrize (
    const ex & thisex,
    const lst & l ) [inline]
```

References GiNaC::ex::antisymmetrize().

**5.1.3.125 symmetrize\_cyclic()** [1/4]

```
ex GiNaC::symmetrize_cyclic (
    const ex & thisex ) [inline]
```

References GiNaC::ex::symmetrize\_cyclic().

Referenced by GiNaC::ex::symmetrize\_cyclic().

**5.1.3.126 symmetrize\_cyclic()** [2/4]

```
ex GiNaC::symmetrize_cyclic (
    const ex & thisex,
    const lst & l ) [inline]
```

References GiNaC::ex::symmetrize\_cyclic().

**5.1.3.127 op()**

```
ex GiNaC::op (
    const ex & thisex,
    size_t i ) [inline]
```

References GiNaC::ex::op().

Referenced by GiNaC::mul::algebraic\_subs\_mul(), GiNaC::basic::calchash(), GiNaC::mul::do\_print\_python\_repr(), GiNaC::basic::do\_print\_tree(), GiNaC::basic::has(), GiNaC::basic::map(), GiNaC::basic::match(), GiNaC::clifford::op(), GiNaC::structure< T, ComparisonPolicy >::op(), GiNaC::basic::operator[](), and GiNaC::basic::subs().

**5.1.3.128 lhs()**

```
ex GiNaC::lhs (
    const ex & thisex ) [inline]
```

References GiNaC::ex::lhs().

Referenced by std::less< GiNaC::ptr< T > >::operator()().

**5.1.3.129 rhs()**

```
ex GiNaC::rhs (
    const ex & thisex ) [inline]
```

References `GiNaC::ex::rhs()`.

Referenced by `Isolve()`, `GiNaC::ptr< GiNaC::basic >::operator!=()`, `std::less< GiNaC::ptr< T > >::operator()`, `GiNaC::ptr< GiNaC::basic >::operator==()`, `GiNaC::matrix::solve()`, and `sqrfree_parfrac()`.

**5.1.3.130 is\_zero()** [1/2]

```
bool GiNaC::is_zero (
    const ex & thisex ) [inline]
```

References `GiNaC::ex::is_zero()`.

Referenced by `GiNaC::matrix::determinant()`, `GiNaC::matrix::determinant_minor()`, `GiNaC::power::eval()`, `GiNaC::tensepsilon::eval_indexed()`, `GiNaC::matrix::fraction_free_elimination()`, `GiNaC::matrix::gauss_elimination()`, `GiNaC::matrix::markowitz_elimination()`, `GiNaC::matrix::mul()`, `GiNaC::relational::operator safe_bool()`, `GiNaC::matrix::pivot()`, `GiNaC::clifford::same_metric()`, and `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`.

**5.1.3.131 swap()** [1/2]

```
void GiNaC::swap (
    ex & e1,
    ex & e2 ) [inline]
```

References `GiNaC::ex::swap()`.

Referenced by `permutation_sign()`, and `GiNaC::matrix::pivot()`.

**5.1.3.132 subs()** [1/3]

```
ex GiNaC::subs (
    const ex & thisex,
    const exmap & m,
    unsigned options = 0 ) [inline]
```

References `m`, `options`, and `GiNaC::ex::subs()`.

Referenced by `GiNaC::clifford::subs()`, and `GiNaC::structure< T, ComparisonPolicy >::subs()`.

**5.1.3.133 subs()** [2/3]

```
ex GiNaC::subs (
    const ex & thisex,
    const lst & ls,
    const lst & lr,
    unsigned options = 0 ) [inline]
```

References `lr`, `options`, and `GiNaC::ex::subs()`.

**5.1.3.134 subs()** [3/3]

```
ex GiNaC::subs (
    const ex & thisex,
    const ex & e,
    unsigned options = 0 ) [inline]
```

References `options`, and `GiNaC::ex::subs()`.

**5.1.3.135 is\_a()** [2/3]

```
template<class T >
bool GiNaC::is_a (
    const ex & obj ) [inline]
```

Check if `ex` is a handle to a `T`, including base classes.

References `GiNaC::ex::bp`.

**5.1.3.136 is\_exactly\_a()** [2/2]

```
template<class T >
bool GiNaC::is_exactly_a (
    const ex & obj ) [inline]
```

Check if `ex` is a handle to a `T`, not including base classes.

References `GiNaC::ex::bp`.

**5.1.3.137 ex\_to()**

```
template<class T >
const T& GiNaC::ex_to (
    const ex & e ) [inline]
```

Return a reference to the basic-derived class `T` object embedded in an expression.

This is fast but unsafe: the result is undefined if the expression does not contain a `T` object at its top level. Hence, you should generally check the type of `e` first. Also, you shouldn't cache the returned reference because `GiNaC`'s garbage collector may destroy the referenced object any time it's used in another expression.

## Parameters

<i>e</i>	expression
----------	------------

## Returns

reference to object of class T

## See also

[is\\_exactly\\_a<class T>\(\)](#)

References GiNaC::ex::bp, and GINAC\_ASSERT.

## 5.1.3.138 compile\_ex() [1/3]

```
void GiNaC::compile_ex (
    const ex & expr,
    const symbol & sym,
    FUNCP\_1P & fp,
    const std::string filename = "" )
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type FUNCP\_1P.

## Parameters

<i>expr</i>	Expression to be compiled
<i>sym</i>	Symbol from the expression to become the function parameter
<i>fp</i>	Returned function pointer
<i>filename</i>	Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted

## 5.1.3.139 compile\_ex() [2/3]

```
void GiNaC::compile_ex (
    const ex & expr,
    const symbol & sym1,
    const symbol & sym2,
    FUNCP\_2P & fp,
    const std::string filename = "" )
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type FUNCP\_2P.

## Parameters

<i>expr</i>	Expression to be compiled
<i>sym</i>	Symbol from the expression to become the function parameter
<i>fp</i>	Returned function pointer
<i>filename</i>	Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted

5.1.3.140 `compile_ex()` [3/3]

```
void GiNaC::compile_ex (
    const lst & exprs,
    const lst & syms,
    FUNCP_CUBA & fp,
    const std::string filename = "" )
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type FUNCP\_CUBA.

## Parameters

<i>expr</i>	Expression to be compiled
<i>sym</i>	Symbol from the expression to become the function parameter
<i>fp</i>	Returned function pointer
<i>filename</i>	Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted

5.1.3.141 `link_ex()` [1/3]

```
void GiNaC::link_ex (
    const std::string filename,
    FUNCP_1P & fp )
```

Opens an existing so-file and returns a function pointer of type FUNCP\_1P to the contained function.

The so-file has to be generated by `compile_ex` in advance.

## Parameters

<i>filename</i>	Name of the so-file to open and link
<i>fp</i>	Returned function pointer

**5.1.3.142** `link_ex()` [2/3]

```
void GiNaC::link_ex (
    const std::string filename,
    FUNC_P_2P & fp )
```

Opens an existing so-file and returns a function pointer of type FUNC\_P\_2P to the contained function.

The so-file has to be generated by compile\_ex in advance.

**Parameters**

<i>filename</i>	Name of the so-file to open and link
<i>fp</i>	Returned function pointer

**5.1.3.143** `link_ex()` [3/3]

```
void GiNaC::link_ex (
    const std::string filename,
    FUNC_P_CUBA & fp )
```

Opens an existing so-file and returns a function pointer of type FUNC\_P\_CUBA to the contained function.

The so-file has to be generated by compile\_ex in advance.

**Parameters**

<i>filename</i>	Name of the so-file to open and link
<i>fp</i>	Returned function pointer

**5.1.3.144** `unlink_ex()`

```
void GiNaC::unlink_ex (
    const std::string filename )
```

Closes all linked .so files that have the supplied filename.

**Parameters**

<i>filename</i>	Name of the so-file to close
-----------------	------------------------------

**5.1.3.145** `swap()` [2/2]

```
void GiNaC::swap (
    expair & e1,
    expair & e2 ) [inline]
```

References `GiNaC::expair::swap()`.

**5.1.3.146** `conjugateepvector()`

```
epvector* GiNaC::conjugateepvector (
    const epvector & )
```

Complex conjugate every element of an `epvector`.

Returns zero if this does not change anything.

Referenced by `GiNaC::pseries::conjugate()`.

**5.1.3.147** `factor()`

```
ex GiNaC::factor (
    const ex & poly,
    unsigned options )
```

Interface function to the outside world.

Factorizes univariate and multivariate polynomials.

It uses `factor1()` on each of the explicitly present factors of `poly`.

The default option is `factor_options::polynomial`, which means that `factor()` will only factorize an expression if it is a proper polynomial (i.e. the flag `info_flags::polynomial` is set). Given the option `factor_options::all`, `factor()` will factorize all subexpressions, e.g. polynomials containing functions or polynomials inside function arguments.

**Parameters**

in	<i>poly</i>	expression to factorize
in	<i>option</i>	options to influence the factorization

**Returns**

factorized expression

References `options`, `poly`, and `pow()`.

Referenced by `algebraic_match_mul_with_mul()`, `collect_common_factors()`, `GiNaC::ncmul::eval()`, `GiNaC::power::expand_add()`, `GiNaC::mul::expandchildren()`, `find_common_factor()`, `GiNaC::mul::find_real_imag()`, `GiNaC::mul::info()`, `GiNaC::mul::series()`, and `sqrfree_parfrac()`.



**5.1.3.148 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [6/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    fail ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print↵
    _tree )
```

**5.1.3.149 GINAC\_DECLARE\_UNARCHIVER()** [15/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    fail )
```

**5.1.3.150 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [7/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    fderivative ,
    function ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
    print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
    print_tree )
```

**5.1.3.151 GINAC\_BIND\_UNARCHIVER()** [15/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    fderivative )
```

**5.1.3.152 GINAC\_DECLARE\_UNARCHIVER()** [16/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    fderivative )
```

**5.1.3.153 GINAC\_BIND\_UNARCHIVER()** [16/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    function )
```

**5.1.3.154 GINAC\_DECLARE\_UNARCHIVER()** [17/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    function )
```

**5.1.3.155 is\_the\_function()**

```
template<typename T >
bool GiNaC::is_the_function (
    const ex & x ) [inline]
```

References x.

**5.1.3.156 make\_hash\_seed()**

```
static unsigned GiNaC::make_hash_seed (
    const std::type_info & tinfo ) [inline], [static]
```

We need a hash function which gives different values for objects of different types.

Hence we need some unique integer for each type. Fortunately, standard C++ RTTI class 'type\_info' stores a pointer to mangled type name. Normally this pointer is the same for all objects of the same type (although it changes from run to run), so it can be used for computing hashes. However, on some platforms (such as woe32) the pointer returned by type\_info::name() might be different even for objects of the same type! Hence we need to resort to comparing string representation of the (mangled) type names. This is quite expensive, so we compare crc32 hashes of those strings. We might get more hash collisions (and slower evaluation as a result), but being a bit slower is much better than being wrong.

References golden\_ratio\_hash().

Referenced by GiNaC::wildcard::calchash(), GiNaC::idx::calchash(), GiNaC::symbol::calchash(), GiNaC::relational::calchash(), GiNaC::symmetry::calchash(), GiNaC::basic::calchash(), and GiNaC::function::calchash().

**5.1.3.157 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [8/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    idx ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_print_latex.
    print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_print_tree )
```

**5.1.3.158** `print_func< print_context >()`

```
GiNaC::print_func< print_context > (
    &varidx::do_print ) &
```

**5.1.3.159** `GINAC_BIND_UNARCHIVER()` [17/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    idx )
```

**5.1.3.160** `GINAC_BIND_UNARCHIVER()` [18/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    varidx )
```

**5.1.3.161** `GINAC_BIND_UNARCHIVER()` [19/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    spinidx )
```

**5.1.3.162** `is_dummy_pair()` [1/2]

```
bool GiNaC::is_dummy_pair (
    const idx & i1,
    const idx & i2 )
```

Check whether two indices form a dummy pair.

References `GiNaC::idx::is_dummy_pair_same_type()`.

Referenced by `GiNaC::matrix::contract_with()`, `GiNaC::spinmetric::contract_with()`, `GiNaC::matrix::eval_indexed()`, `GiNaC::tensdelta::eval_indexed()`, `find_free_and_dummy()`, `GiNaC::indexed::has_dummy_index_for()`, `is_dummy↔_pair()`, `GiNaC::is_summation_idx::operator()`, `GiNaC::tensor::replace_contr_index()`, and `reposition_dummy↔_indices()`.

**5.1.3.163 is\_dummy\_pair()** [2/2]

```
bool GiNaC::is_dummy_pair (
    const ex & e1,
    const ex & e2 )
```

Check whether two expressions form a dummy index pair.

References `is_dummy_pair()`.

**5.1.3.164 find\_free\_and\_dummy()** [1/2]

```
void GiNaC::find_free_and_dummy (
    exvector::const_iterator it,
    exvector::const_iterator itend,
    exvector & out_free,
    exvector & out_dummy )
```

Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).

**Parameters**

<i>it</i>	Pointer to start of index vector
<i>itend</i>	Pointer to end of index vector
<i>out_free</i>	Vector of free indices (returned, sorted)
<i>out_dummy</i>	Vector of dummy indices (returned, sorted)

References `is_dummy_pair()`, `last`, and `shaker_sort()`.

Referenced by `GiNaC::su3d::contract_with()`, `count_dummy_indices()`, `count_free_indices()`, `find_dummy_↵indices()`, `find_free_and_dummy()`, `get_all_dummy_indices_safely()`, `GiNaC::indexed::get_dummy_indices()`, `Gi↵NaC::ncmul::get_free_indices()`, `GiNaC::mul::get_free_indices()`, `GiNaC::indexed::get_free_indices()`, `simplify_↵indexed()`, and `simplify_indexed_product()`.

**5.1.3.165 minimal\_dim()**

```
ex GiNaC::minimal_dim (
    const ex & dim1,
    const ex & dim2 )
```

Return the minimum of two index dimensions.

If this is undecidable, throw an exception. Numeric dimensions are always considered "smaller" than symbolic dimensions.

References `GiNaC::ex::is_equal()`.

Referenced by `GiNaC::idx::minimal_dim()`, and `simplify_indexed_product()`.

**5.1.3.166** GINAC\_DECLARE\_UNARCHIVER() [18/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    idx )
```

**5.1.3.167** GINAC\_DECLARE\_UNARCHIVER() [19/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    varidx )
```

**5.1.3.168** GINAC\_DECLARE\_UNARCHIVER() [20/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    spinidx )
```

**5.1.3.169** find\_free\_and\_dummy() [2/2]

```
void GiNaC::find_free_and_dummy (
    const exvector & v,
    exvector & out_free,
    exvector & out_dummy ) [inline]
```

Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).

**Parameters**

<i>v</i>	Index vector
<i>out_free</i>	Vector of free indices (returned, sorted)
<i>out_dummy</i>	Vector of dummy indices (returned, sorted)

References `find_free_and_dummy()`.

**5.1.3.170** find\_dummy\_indices()

```
void GiNaC::find_dummy_indices (
    const exvector & v,
    exvector & out_dummy ) [inline]
```

Given a vector of indices, find the dummy indices.

## Parameters

<i>v</i>	Index vector
<i>out_dummy</i>	Vector of dummy indices (returned, sorted)

References `find_free_and_dummy()`.

Referenced by `GiNaC::indexed::get_dummy_indices()`.

#### 5.1.3.171 `count_dummy_indices()`

```
size_t GiNaC::count_dummy_indices (
    const exvector & v ) [inline]
```

Count the number of dummy index pairs in an index vector.

References `find_free_and_dummy()`.

#### 5.1.3.172 `count_free_indices()`

```
size_t GiNaC::count_free_indices (
    const exvector & v ) [inline]
```

Count the number of dummy index pairs in an index vector.

References `find_free_and_dummy()`.

#### 5.1.3.173 `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [9/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    indexed ,
    exprseq ,
    print_func< print\_context > &::do_print. print_func< print\_latex > &::do_↵
    print_latex. print_func< print\_tree > &::do_print_tree )
```

#### 5.1.3.174 `GINAC_BIND_UNARCHIVER()` [20/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    indexed )
```

## 5.1.3.175 indices\_consistent()

```
static bool GiNaC::indices_consistent (
    const exvector & v1,
    const exvector & v2 ) [static]
```

Check whether two sorted index vectors are consistent (i.e. equal).

Referenced by `GiNaC::add::get_free_indices()`, and `simplify_indexed()`.

## 5.1.3.176 number\_of\_type()

```
template<class T >
size_t GiNaC::number_of_type (
    const exvector & v )
```

## 5.1.3.177 rename\_dummy\_indices()

```
template<class T >
static ex GiNaC::rename_dummy_indices (
    const ex & e,
    exvector & global_dummy_indices,
    exvector & local_dummy_indices ) [static]
```

Rename dummy indices in an expression.

## Parameters

<i>e</i>	Expression to work on
<i>local_dummy_indices</i>	The set of dummy indices that appear in the expression "e"
<i>global_dummy_indices</i>	The set of dummy indices that have appeared before and which we would like to use in "e", too. This gets updated by the function

References `GiNaC::ex::begin()`, `GINAC_ASSERT`, `GiNaC::subs_options::no_pattern`, `shaker_sort()`, and `GiNaC::ex::subs()`.

## 5.1.3.178 find\_variant\_indices()

```
static void GiNaC::find_variant_indices (
    const exvector & v,
    exvector & variant_indices ) [static]
```

Given a set of indices, extract those of class `varidx`.

Referenced by `simplify_indexed()`, and `simplify_indexed_product()`.

#### 5.1.3.179 `reposition_dummy_indices()`

```
bool GiNaC::reposition_dummy_indices (
    ex & e,
    exvector & variant_dummy_indices,
    exvector & moved_indices )
```

Raise/lower dummy indices in a single indexed objects to canonicalize their variance.

##### Parameters

<i>e</i>	Object to work on
<i>variant_dummy_indices</i>	The set of indices that might need repositioning (will be changed by this function)
<i>moved_indices</i>	The set of indices that have been repositioned (will be changed by this function)

##### Returns

true if 'e' was changed

References `is_dummy_pair()`, `k`, `m`, `GiNaC::subs_options::no_pattern`, `GiNaC::ex::nops()`, `GiNaC::ex::op()`, and `GiNaC::ex::subs()`.

Referenced by `simplify_indexed()`, and `simplify_indexed_product()`.

#### 5.1.3.180 `product_to_exvector()`

```
static void GiNaC::product_to_exvector (
    const ex & e,
    exvector & v,
    bool & non_commutative ) [static]
```

References `_ex2`, `GINAC_ASSERT`, `GiNaC::ex::is_equal()`, `GiNaC::ex::nops()`, and `GiNaC::ex::op()`.

Referenced by `get_all_dummy_indices()`, and `simplify_indexed_product()`.

#### 5.1.3.181 `idx_symmetrization()`

```
template<class T >
ex GiNaC::idx_symmetrization (
    const ex & r,
    const exvector & local_dummy_indices )
```

References `r`, and `symmetrize()`.



**5.1.3.182** `simplify_indexed()` [3/3]

```
ex GiNaC::simplify_indexed (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar_products & sp )
```

Simplify indexed expression, return list of free indices.

References `_ex2`, `GiNaC::ex::coeff()`, `coeff()`, `GiNaC::ex::expand()`, `find_free_and_dummy()`, `find_variant_indices()`, `hasindex()`, `indices_consistent()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `k`, `GiNaC::ex::nops()`, `GiNaC::ex::op()`, `reposition_dummy_indices()`, `GiNaC::container_storage< C >::seq`, `simplify_indexed()`, and `simplify_indexed_product()`.

**5.1.3.183** `simplify_indexed_product()`

```
ex GiNaC::simplify_indexed_product (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar_products & sp )
```

Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.

References `_ex0`, `_ex1`, `GiNaC::return_types::commutative`, `GiNaC::scalar_products::evaluate()`, `find_free_and_dummy()`, `find_variant_indices()`, `GINAC_ASSERT`, `GiNaC::scalar_products::is_defined()`, `GiNaC::ex::is_zero()`, `minimal_dim()`, `product_to_exvector()`, `r`, `reposition_dummy_indices()`, and `simplify_indexed()`.

Referenced by `simplify_indexed()`.

**5.1.3.184** `hasindex()`

```
bool GiNaC::hasindex (
    const ex & x,
    const ex & sym )
```

References `GiNaC::ex::nops()`, `GiNaC::ex::op()`, and `x`.

Referenced by `simplify_indexed()`.

**5.1.3.185 get\_all\_dummy\_indices\_safely()**

```
exvector GiNaC::get_all_dummy_indices_safely (
    const ex & e )
```

More reliable version of the form.

The former assumes that e is an expanded expression.

References find\_free\_and\_dummy(), GiNaC::ex::get\_free\_indices(), GiNaC::ex::nops(), and GiNaC::ex::op().

Referenced by GiNaC::mul::expand(), GiNaC::make\_flat\_inserter::handle\_factor(), and rename\_dummy\_indices\_uniquely().

**5.1.3.186 get\_all\_dummy\_indices()**

```
exvector GiNaC::get_all_dummy_indices (
    const ex & e )
```

Returns all dummy indices from the exvector.

Returns all dummy indices from the expression.

References product\_to\_exvector().

Referenced by expand\_dummy\_sum(), and GiNaC::power::expand\_mul().

**5.1.3.187 rename\_dummy\_indices\_uniquely()** [1/4]

```
lst GiNaC::rename_dummy_indices_uniquely (
    const exvector & va,
    const exvector & vb )
```

Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.

References GiNaC::container\_storage< C >::reserve().

Referenced by GiNaC::ncmul::expand(), GiNaC::mul::expand(), GiNaC::power::expand\_mul(), GiNaC::make\_flat\_inserter::handle\_factor(), and rename\_dummy\_indices\_uniquely().

**5.1.3.188 rename\_dummy\_indices\_uniquely()** [2/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
    const exvector & va,
    const exvector & vb,
    const ex & b )
```

Same as above, where va and vb contain the indices of a and b and are sorted.

References GiNaC::subs\_options::no\_index\_renaming, GiNaC::subs\_options::no\_pattern, GiNaC::ex::nops(), GiNaC::container< C >::op(), rename\_dummy\_indices\_uniquely(), and GiNaC::ex::subs().

5.1.3.189 `rename_dummy_indices_uniquely()` [3/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
    const ex & a,
    const ex & b )
```

Returns `b` with all dummy indices, which are common with `a`, renamed.

References `get_all_dummy_indices_safely()`, `GiNaC::subs_options::no_index_renaming`, `GiNaC::subs_options::no_pattern`, `rename_dummy_indices_uniquely()`, and `GiNaC::ex::subs()`.

5.1.3.190 `rename_dummy_indices_uniquely()` [4/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
    exvector & va,
    const ex & b,
    bool modify_va )
```

Returns `b` with all dummy indices, which are listed in `va`, renamed if `modify_va` is set to `TRUE` all dummy indices of `b` will be appended to `va`.

References `get_all_dummy_indices_safely()`, `GiNaC::subs_options::no_index_renaming`, `GiNaC::subs_options::no_pattern`, `rename_dummy_indices_uniquely()`, and `GiNaC::ex::subs()`.

5.1.3.191 `expand_dummy_sum()`

```
ex GiNaC::expand_dummy_sum (
    const ex & e,
    bool subs_idx = false )
```

This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.

Optionally all indices with a variance will be substituted by indices with the corresponding numeric values without variance.

## Parameters

<code>e</code>	the given expression
<code>subs_idx</code>	indicates if variance of dummy indices should be neglected

References `GiNaC::ex::expand()`, `get_all_dummy_indices()`, `idx`, `GiNaC::ex::map()`, `GiNaC::info_flags::nonnegint`, `GiNaC::ex::subs()`, and `to_int()`.

**5.1.3.192 GINAC\_DECLARE\_UNARCHIVER()** [21/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    indexed )
```

**5.1.3.193 conjugate\_evalf()**

```
static ex GiNaC::conjugate_evalf (
    const ex & arg ) [static]
```

References conjugate().

**5.1.3.194 conjugate\_eval()**

```
static ex GiNaC::conjugate_eval (
    const ex & arg ) [static]
```

References GiNaC::ex::conjugate().

**5.1.3.195 conjugate\_print\_latex()**

```
static void GiNaC::conjugate_print_latex (
    const ex & arg,
    const print_context & c ) [static]
```

References c, and GiNaC::ex::print().

**5.1.3.196 conjugate\_conjugate()**

```
static ex GiNaC::conjugate_conjugate (
    const ex & arg ) [static]
```

**5.1.3.197 conjugate\_expl\_derivative()**

```
static ex GiNaC::conjugate_expl_derivative (
    const ex & arg,
    const symbol & s ) [static]
```

References conjugate(), GiNaC::ex::diff(), GiNaC::basic::hold(), GiNaC::symbol::info(), and GiNaC::info\_flags::real.

**5.1.3.198 conjugate\_real\_part()**

```
static ex GiNaC::conjugate_real_part (
    const ex & arg ) [static]
```

References [GiNaC::ex::real\\_part\(\)](#).

**5.1.3.199 conjugate\_imag\_part()**

```
static ex GiNaC::conjugate_imag_part (
    const ex & arg ) [static]
```

References [GiNaC::ex::imag\\_part\(\)](#).

**5.1.3.200 func\_arg\_info()**

```
static bool GiNaC::func_arg_info (
    const ex & arg,
    unsigned inf ) [static]
```

References [GiNaC::info\\_flags::cinteger](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::odd](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::prime](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [GiNaC::info\\_flags::real](#).

Referenced by [conjugate\\_info\(\)](#).

**5.1.3.201 conjugate\_info()**

```
static bool GiNaC::conjugate_info (
    const ex & arg,
    unsigned inf ) [static]
```

References [func\\_arg\\_info\(\)](#).

**5.1.3.202 REGISTER\_FUNCTION()** [1/36]

```
GiNaC::REGISTER_FUNCTION (
    conjugate_function ,
    eval_func(conjugate\_eval). evalf_func(conjugate\_evalf). expl_derivative_↵
func(conjugate\_expl\_derivative). info_func(conjugate\_info). print_func< print\_latex >(conjugate\_print\_latex
conjugate_func(conjugate\_conjugate). real_part_func(conjugate\_real\_part). imag_part_func(conjugate\_imag\_part
set_name("conjugate", "conjugate") )
```

#### 5.1.3.203 `real_part_evalf()`

```
static ex GiNaC::real_part_evalf (
    const ex & arg ) [static]
```

References `real()`.

#### 5.1.3.204 `real_part_eval()`

```
static ex GiNaC::real_part_eval (
    const ex & arg ) [static]
```

References `GiNaC::ex::real_part()`.

#### 5.1.3.205 `real_part_print_latex()`

```
static void GiNaC::real_part_print_latex (
    const ex & arg,
    const print_context & c ) [static]
```

References `c`, and `GiNaC::ex::print()`.

#### 5.1.3.206 `real_part_conjugate()`

```
static ex GiNaC::real_part_conjugate (
    const ex & arg ) [static]
```

#### 5.1.3.207 `real_part_real_part()`

```
static ex GiNaC::real_part_real_part (
    const ex & arg ) [static]
```

#### 5.1.3.208 `real_part_imag_part()`

```
static ex GiNaC::real_part_imag_part (
    const ex & arg ) [static]
```

**5.1.3.209 real\_part\_expl\_derivative()**

```
static ex GiNaC::real_part_expl_derivative (
    const ex & arg,
    const symbol & s ) [static]
```

References `GiNaC::ex::diff()`, `GiNaC::basic::hold()`, `GiNaC::symbol::info()`, `GiNaC::info_flags::real`, and `real_part()`.

**5.1.3.210 REGISTER\_FUNCTION()** [2/36]

```
GiNaC::REGISTER_FUNCTION (
    real_part_function ,
    eval_func(real_part_eval). evalf_func(real_part_evalf). expl_derivative_↔
func(real_part_expl_derivative). print_func< print_latex >(real_part_print_latex). conjugate_↔
_func(real_part_conjugate). real_part_func(real_part_real_part). imag_part_func(real_part_imag_part).
set_name("real_part", "real_part") )
```

**5.1.3.211 imag\_part\_evalf()**

```
static ex GiNaC::imag_part_evalf (
    const ex & arg ) [static]
```

References `imag()`.

**5.1.3.212 imag\_part\_eval()**

```
static ex GiNaC::imag_part_eval (
    const ex & arg ) [static]
```

References `GiNaC::ex::imag_part()`.

**5.1.3.213 imag\_part\_print\_latex()**

```
static void GiNaC::imag_part_print_latex (
    const ex & arg,
    const print_context & c ) [static]
```

References `c`, and `GiNaC::ex::print()`.

**5.1.3.214 imag\_part\_conjugate()**

```
static ex GiNaC::imag_part_conjugate (
    const ex & arg ) [static]
```

**5.1.3.215 imag\_part\_real\_part()**

```
static ex GiNaC::imag_part_real_part (
    const ex & arg ) [static]
```

**5.1.3.216 imag\_part\_imag\_part()**

```
static ex GiNaC::imag_part_imag_part (
    const ex & arg ) [static]
```

**5.1.3.217 imag\_part\_expl\_derivative()**

```
static ex GiNaC::imag_part_expl_derivative (
    const ex & arg,
    const symbol & s ) [static]
```

References [GiNaC::ex::diff\(\)](#), [GiNaC::basic::hold\(\)](#), [imag\\_part\(\)](#), [GiNaC::symbol::info\(\)](#), and [GiNaC::info\\_flags::real](#).

**5.1.3.218 REGISTER\_FUNCTION()** [3/36]

```
GiNaC::REGISTER_FUNCTION (
    imag_part_function ,
    eval_func(imag\_part\_eval). evalf_func(imag\_part\_evalf). expl_derivative_↔
func(imag\_part\_expl\_derivative). print_func< print\_latex >(imag\_part\_print\_latex). conjugate_↔
_func(imag\_part\_conjugate). real_part_func(imag\_part\_real\_part). imag_part_func(imag\_part\_imag\_part).
set_name("imag_part", "imag\_part") )
```

**5.1.3.219 abs\_evalf()**

```
static ex GiNaC::abs_evalf (
    const ex & arg ) [static]
```

References [abs\(\)](#), and [GiNaC::basic::hold\(\)](#).



**5.1.3.220 abs\_eval()**

```
static ex GiNaC::abs_eval (
    const ex & arg ) [static]
```

References `abs()`, `exp()`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `is_ex_the_function`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::nonnegative`, `GiNaC::ex::op()`, `GiNaC::info_flags::positive`, `pow()`, `GiNaC::info_flags::real`, `GiNaC::ex::real_part()`, and `step()`.

**5.1.3.221 abs\_expand()**

```
static ex GiNaC::abs_expand (
    const ex & arg,
    unsigned options ) [static]
```

References `abs()`, `GiNaC::ex::begin()`, `GiNaC::ex::end()`, `GiNaC::ex::expand()`, `GiNaC::expand_options::expand_function_args`, `GiNaC::expand_options::expand_transcendental`, `GiNaC::status_flags::expanded`, `GiNaC::basic::hold()`, `GiNaC::ex::nops()`, and `options`.

**5.1.3.222 abs\_expl\_derivative()**

```
static ex GiNaC::abs_expl_derivative (
    const ex & arg,
    const symbol & s ) [static]
```

References `abs()`, `GiNaC::ex::conjugate()`, and `GiNaC::ex::diff()`.

**5.1.3.223 abs\_print\_latex()**

```
static void GiNaC::abs_print_latex (
    const ex & arg,
    const print_context & c ) [static]
```

References `c`, and `GiNaC::ex::print()`.

**5.1.3.224 abs\_print\_csrc\_float()**

```
static void GiNaC::abs_print_csrc_float (
    const ex & arg,
    const print_context & c ) [static]
```

References `c`, and `GiNaC::ex::print()`.

**5.1.3.225 abs\_conjugate()**

```
static ex GiNaC::abs_conjugate (
    const ex & arg ) [static]
```

References `abs()`, and `GiNaC::basic::hold()`.

**5.1.3.226 abs\_real\_part()**

```
static ex GiNaC::abs_real_part (
    const ex & arg ) [static]
```

References `abs()`, and `GiNaC::basic::hold()`.

**5.1.3.227 abs\_imag\_part()**

```
static ex GiNaC::abs_imag_part (
    const ex & arg ) [static]
```

**5.1.3.228 abs\_power()**

```
static ex GiNaC::abs_power (
    const ex & arg,
    const ex & exp ) [static]
```

References `abs()`, `GiNaC::ex::conjugate()`, `GiNaC::info_flags::even`, `exp()`, `GiNaC::basic::hold()`, `GiNaC::numeric::info()`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `is_even()`, `pow()`, and `GiNaC::info_flags::real`.

**5.1.3.229 abs\_info()**

```
bool GiNaC::abs_info (
    const ex & arg,
    unsigned inf )
```

References `GiNaC::info_flags::even`, `GiNaC::info_flags::has_indices`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::nonnegative`, `GiNaC::info_flags::nonnegint`, `GiNaC::info_flags::odd`, `GiNaC::info_flags::positive`, `GiNaC::info_flags::prime`, and `GiNaC::info_flags::real`.

**5.1.3.230 REGISTER\_FUNCTION()** [4/36]

```
GiNaC::REGISTER_FUNCTION (
    abs ,
    eval_func(abs_eval). evalf_func(abs_evalf). expand_func(abs_expand). expl_↵
    derivative_func(abs_expl_derivative). info_func(abs_info). print_func< print_latex >(abs_print_latex).
    print_func< print_csrc_float >(abs_print_csrc_float). print_func< print_csrc_double >(abs_print_csrc_float)
    conjugate_func(abs_conjugate). real_part_func(abs_real_part). imag_part_func(abs_imag_part).
    power_func(abs_power) )
```

**5.1.3.231 step\_evalf()**

```
static ex GiNaC::step_evalf (
    const ex & arg ) [static]
```

References `GiNaC::basic::hold()`, and `step()`.

**5.1.3.232 step\_eval()**

```
static ex GiNaC::step_eval (
    const ex & arg ) [static]
```

References `GiNaC::basic::hold()`, `I`, `GiNaC::numeric::imag()`, `GiNaC::numeric::is_real()`, `GiNaC::numeric::is_zero()`, `GiNaC::ex::nops()`, `GiNaC::ex::op()`, `GiNaC::numeric::real()`, and `step()`.

**5.1.3.233 step\_series()**

```
static ex GiNaC::step_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex0`, `GiNaC::ex::info()`, `GiNaC::numeric::is_zero()`, `GiNaC::subs_options::no_pattern`, `GiNaC::info_↵`  
`flags::numeric`, `options`, `real()`, `step()`, `GiNaC::ex::subs()`, and `GiNaC::series_options::suppress_branchcut`.

**5.1.3.234 step\_conjugate()**

```
static ex GiNaC::step_conjugate (
    const ex & arg ) [static]
```

References `GiNaC::basic::hold()`, and `step()`.

**5.1.3.235 step\_real\_part()**

```
static ex GiNaC::step_real_part (
    const ex & arg ) [static]
```

References GiNaC::basic::hold(), and step().

**5.1.3.236 step\_imag\_part()**

```
static ex GiNaC::step_imag_part (
    const ex & arg ) [static]
```

**5.1.3.237 REGISTER\_FUNCTION()** [5/36]

```
GiNaC::REGISTER_FUNCTION (
    step ,
    eval_func(step_eval). evalf_func(step_evalf). series_func(step_series). conjugate↵
    _func(step_conjugate). real_part_func(step_real_part). imag_part_func(step_imag_part) )
```

**5.1.3.238 csgn\_evalf()**

```
static ex GiNaC::csgn_evalf (
    const ex & arg ) [static]
```

References csgn().

**5.1.3.239 csgn\_eval()**

```
static ex GiNaC::csgn_eval (
    const ex & arg ) [static]
```

References csgn(), I, GiNaC::numeric::imag(), GiNaC::numeric::is\_real(), GiNaC::numeric::is\_zero(), GiNaC::ex↵  
::nops(), GiNaC::ex::op(), and GiNaC::numeric::real().

#### 5.1.3.240 csgn\_series()

```
static ex GiNaC::csgn_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex0`, `csgn()`, `GiNaC::ex::info()`, `GiNaC::numeric::is_zero()`, `GiNaC::subs_options::no_pattern`, `GiNaC::info_flags::numeric`, `options`, `real()`, `GiNaC::ex::subs()`, and `GiNaC::series_options::suppress_branchcut`.

#### 5.1.3.241 csgn\_conjugate()

```
static ex GiNaC::csgn_conjugate (
    const ex & arg ) [static]
```

References `csgn()`.

#### 5.1.3.242 csgn\_real\_part()

```
static ex GiNaC::csgn_real_part (
    const ex & arg ) [static]
```

References `csgn()`.

#### 5.1.3.243 csgn\_imag\_part()

```
static ex GiNaC::csgn_imag_part (
    const ex & arg ) [static]
```

#### 5.1.3.244 csgn\_power()

```
static ex GiNaC::csgn_power (
    const ex & arg,
    const ex & exp ) [static]
```

References `_ex2`, `csgn()`, `exp()`, `GiNaC::basic::hold()`, `GiNaC::numeric::info()`, `is_integer()`, and `GiNaC::info_flags::positive`.

**5.1.3.245 REGISTER\_FUNCTION()** [6/36]

```
GiNaC::REGISTER_FUNCTION (
    csgn ,
    eval_func(csgn_eval). evalf_func(csgn_evalf). series_func(csgn_series). conjugate↵
    _func(csgn_conjugate). real_part_func(csgn_real_part). imag_part_func(csgn_imag_part). power↵
    _func(csgn_power) )
```

**5.1.3.246 eta\_evalf()**

```
static ex GiNaC::eta_evalf (
    const ex & x,
    const ex & y ) [static]
```

References `_ex0`, `csgn()`, `evalf()`, `I`, `imag()`, `GiNaC::ex::info()`, `GiNaC::numeric::is_negative()`, `GiNaC::numeric::is_↵`  
`real()`, `GiNaC::info_flags::numeric`, `Pi`, `GiNaC::info_flags::positive`, and `x`.

**5.1.3.247 eta\_eval()**

```
static ex GiNaC::eta_eval (
    const ex & x,
    const ex & y ) [static]
```

References `_ex0`, `csgn()`, `I`, `imag()`, `GiNaC::ex::info()`, `GiNaC::numeric::is_negative()`, `GiNaC::numeric::is_real()`,  
`GiNaC::info_flags::numeric`, `Pi`, `GiNaC::info_flags::positive`, and `x`.

**5.1.3.248 eta\_series()**

```
static ex GiNaC::eta_series (
    const ex & x,
    const ex & y,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex0`, `GiNaC::ex::info()`, `GiNaC::info_flags::negative`, `GiNaC::subs_options::no_pattern`, `GiNaC::info_↵`  
`_flags::numeric`, `GiNaC::ex::subs()`, and `x`.

**5.1.3.249 eta\_conjugate()**

```
static ex GiNaC::eta_conjugate (
    const ex & x,
    const ex & y ) [static]
```

References `x`.

**5.1.3.250 eta\_real\_part()**

```
static ex GiNaC::eta_real_part (
    const ex & x,
    const ex & y ) [static]
```

**5.1.3.251 eta\_imag\_part()**

```
static ex GiNaC::eta_imag_part (
    const ex & x,
    const ex & y ) [static]
```

References `GiNaC::basic::hold()`, `I`, and `x`.

**5.1.3.252 REGISTER\_FUNCTION()** [7/36]

```
GiNaC::REGISTER_FUNCTION (
    eta ,
    eval_func(eta_eval). evalf_func(eta_evalf). series_func(eta_series). latex_↵
name("\a"). set_symmetry(sy_symm(0, 1)). conjugate_func(eta_conjugate). real_part_func(eta_real_part).
imag_part_func(eta_imag_part) )
```

**5.1.3.253 Li2\_evalf()**

```
static ex GiNaC::Li2_evalf (
    const ex & x ) [static]
```

References `GiNaC::basic::hold()`, `Li2()`, and `x`.

**5.1.3.254 Li2\_eval()**

```
static ex GiNaC::Li2_eval (
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `_ex12`, `_ex1_2`, `_ex2`, `_ex6`, `_ex_1`, `_ex_1_2`, `_ex_48`, `Catalan`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `I`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `Li2()`, `log()`, `GiNaC::info_↵`  
`flags::numeric`, `Pi`, and `x`.

**5.1.3.255 Li2\_deriv()**

```
static ex GiNaC::Li2_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex1`, `GINAC_ASSERT`, `log()`, and `x`.

**5.1.3.256 Li2\_series()** [1/2]

```
static ex GiNaC::Li2_series (
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex0`, `_ex1`, `_ex2`, `_num2_p`, `l`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `is_real()`, `GiNaC::ex::is_zero()`, `GiNaC::relational::lhs()`, `Li2()`, `log()`, `GiNaC::subs_options::no_pattern`, `GiNaC::ex::nops()`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `options`, `order`, `Pi`, `pow()`, `GiNaC::relational::rhs()`, `GiNaC::ex::series()`, `series()`, `GiNaC::ex::subs()`, `GiNaC::series_options::suppress_branchcut`, `x`, and `zeta()`.

Referenced by `Li2_projection()`.

**5.1.3.257 Li2\_conjugate()**

```
static ex GiNaC::Li2_conjugate (
    const ex & x ) [static]
```

References `_num1_p`, `GiNaC::ex::conjugate()`, `GiNaC::basic::hold()`, `GiNaC::ex::imag_part()`, `GiNaC::ex::info()`, `GiNaC::ex::is_zero()`, `Li2()`, `GiNaC::info_flags::negative`, and `x`.

**5.1.3.258 REGISTER\_FUNCTION()** [8/36]

```
GiNaC::REGISTER_FUNCTION (
    Li2 ,
    eval_func(Li2_eval). evalf_func(Li2_evalf). derivative_func(Li2_deriv). series←
_func(Li2_series). conjugate_func(Li2_conjugate). latex_name("\thrm{Li}_2" ) )
```

**5.1.3.259 Li3\_eval()**

```
static ex GiNaC::Li3_eval (
    const ex & x ) [static]
```

References `GiNaC::ex::is_zero()`, and `x`.



**5.1.3.260 REGISTER\_FUNCTION()** [9/36]

```
GiNaC::REGISTER_FUNCTION (
    Li3 ,
    eval_func(Li3_eval). latex_name("\thrm{Li}_3") )
```

**5.1.3.261 zetaderiv\_eval()**

```
static ex GiNaC::zetaderiv_eval (
    const ex & n,
    const ex & x ) [static]
```

References `GiNaC::basic::hold()`, `n`, `GiNaC::info_flags::numeric`, `x`, and `zeta()`.

**5.1.3.262 zetaderiv\_deriv()**

```
static ex GiNaC::zetaderiv_deriv (
    const ex & n,
    const ex & x,
    unsigned deriv_param ) [static]
```

References `GINAC_ASSERT`, `n`, and `x`.

**5.1.3.263 REGISTER\_FUNCTION()** [10/36]

```
GiNaC::REGISTER_FUNCTION (
    zetaderiv ,
    eval_func(zetaderiv_eval). derivative_func(zetaderiv_deriv). latex_name("\ta^{\ime}")
)
```

**5.1.3.264 factorial\_evalf()**

```
static ex GiNaC::factorial_evalf (
    const ex & x ) [static]
```

References `factorial()`, `GiNaC::basic::hold()`, and `x`.

**5.1.3.265 factorial\_eval()**

```
static ex GiNaC::factorial_eval (
    const ex & x ) [static]
```

References factorial(), GiNaC::basic::hold(), and x.

**5.1.3.266 factorial\_print\_dflt\_latex()**

```
static void GiNaC::factorial_print_dflt_latex (
    const ex & x,
    const print_context & c ) [static]
```

References c, GiNaC::ex::print(), and x.

**5.1.3.267 factorial\_conjugate()**

```
static ex GiNaC::factorial_conjugate (
    const ex & x ) [static]
```

References factorial(), GiNaC::basic::hold(), and x.

**5.1.3.268 factorial\_real\_part()**

```
static ex GiNaC::factorial_real_part (
    const ex & x ) [static]
```

References factorial(), GiNaC::basic::hold(), and x.

**5.1.3.269 factorial\_imag\_part()**

```
static ex GiNaC::factorial_imag_part (
    const ex & x ) [static]
```

**5.1.3.270 REGISTER\_FUNCTION()** [11/36]

```
GiNaC::REGISTER_FUNCTION (
    factorial ,
    eval_func(factorial_eval). evalf_func(factorial_evalf). print_func< print_dflt
>(factorial_print_dflt_latex). print_func< print_latex >(factorial_print_dflt_latex). conjugate←
_func(factorial_conjugate). real_part_func(factorial_real_part). imag_part_func(factorial_imag_part)
)
```

**5.1.3.271 binomial\_evalf()**

```
static ex GiNaC::binomial_evalf (  
    const ex & x,  
    const ex & y ) [static]
```

References binomial(), GiNaC::basic::hold(), and x.

**5.1.3.272 binomial\_sym()**

```
static ex GiNaC::binomial_sym (  
    const ex & x,  
    const numeric & y ) [static]
```

References \_ex0, \_ex1, binomial(), GiNaC::ex::expand(), GiNaC::basic::hold(), GiNaC::numeric::is\_integer(), GiNaC::numeric::is\_nonneg\_integer(), GiNaC::numeric::to\_int(), and x.

Referenced by binomial\_evalf().

**5.1.3.273 binomial\_eval()**

```
static ex GiNaC::binomial_eval (  
    const ex & x,  
    const ex & y ) [static]
```

References binomial(), binomial\_sym(), GiNaC::basic::hold(), is\_integer(), and x.

**5.1.3.274 binomial\_conjugate()**

```
static ex GiNaC::binomial_conjugate (  
    const ex & x,  
    const ex & y ) [static]
```

References binomial(), GiNaC::basic::hold(), and x.

**5.1.3.275 binomial\_real\_part()**

```
static ex GiNaC::binomial_real_part (  
    const ex & x,  
    const ex & y ) [static]
```

References binomial(), GiNaC::basic::hold(), and x.

**5.1.3.276 binomial\_imag\_part()**

```
static ex GiNaC::binomial_imag_part (
    const ex & x,
    const ex & y ) [static]
```

**5.1.3.277 REGISTER\_FUNCTION()** [12/36]

```
GiNaC::REGISTER_FUNCTION (
    binomial ,
    eval_func(binomial_eval) . evalf_func(binomial_evalf) . conjugate_func(binomial_conjugate) .
    real_part_func(binomial_real_part) . imag_part_func(binomial_imag_part) )
```

**5.1.3.278 Order\_eval()**

```
static ex GiNaC::Order_eval (
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `GiNaC::ex::is_zero()`, `m`, and `x`.

**5.1.3.279 Order\_series()**

```
static ex GiNaC::Order_series (
    const ex & x,
    const relational & r,
    int order,
    unsigned options ) [static]
```

References `_ex1`, `GINAC_ASSERT`, `GiNaC::ex::ldegree()`, `order`, `r`, and `x`.

**5.1.3.280 Order\_conjugate()**

```
static ex GiNaC::Order_conjugate (
    const ex & x ) [static]
```

References `x`.

**5.1.3.281 Order\_real\_part()**

```
static ex GiNaC::Order_real_part (
    const ex & x ) [static]
```

References x.

**5.1.3.282 Order\_imag\_part()**

```
static ex GiNaC::Order_imag_part (
    const ex & x ) [static]
```

References GiNaC::ex::info(), GiNaC::info\_flags::real, and x.

**5.1.3.283 Order\_expl\_derivative()**

```
static ex GiNaC::Order_expl_derivative (
    const ex & arg,
    const symbol & s ) [static]
```

References GiNaC::ex::diff().

**5.1.3.284 REGISTER\_FUNCTION()** [13/36]

```
GiNaC::REGISTER_FUNCTION (
    Order ,
    eval_func(Order_eval). series_func(Order_series). latex_name("\\thcal{O}").
    expl_derivative_func(Order_expl_derivative). conjugate_func(Order_conjugate). real_part_←
    func(Order_real_part). imag_part_func(Order_imag_part) )
```

**5.1.3.285 Isolve()**

```
ex GiNaC::lsolve (
    const ex & eqns,
    const ex & symbols,
    unsigned options = solve_algo::automatic )
```

Factorial function.

Binomial function. Order term function (for truncated power series).

References GiNaC::container< C >::append(), c, GiNaC::ex::coeff(), GiNaC::matrix::cols(), GiNaC::ex::expand(), GiNaC::info\_flags::exprseq, GINAC\_ASSERT, GiNaC::symbolset::has(), GiNaC::ex::info(), GiNaC::info\_flags::list, GiNaC::ex::nops(), GiNaC::ex::op(), options, r, GiNaC::info\_flags::relation\_equal, rhs(), GiNaC::matrix::rows(), GiNaC::matrix::solve(), GiNaC::info\_flags::symbol, and syms.

**5.1.3.286 fsolve()**

```
const numeric GiNaC::fsolve (
    const ex & f,
    const symbol & x,
    const numeric & x1,
    const numeric & x2 )
```

Find a real root of real-valued function  $f(x)$  numerically within a given interval.

The function must change sign across interval. Uses Newton- Raphson method combined with bisection in order to guarantee convergence.

**Parameters**

$f$	Function $f(x)$
$x$	Symbol $f(x)$
$x1$	lower interval limit
$x2$	upper interval limit

**Exceptions**

<i>runtime_error</i>	(if interval is invalid).
----------------------	---------------------------

References `GiNaC::ex::diff()`, `GiNaC::ex::evalf()`, `GiNaC::numeric::is_real()`, `is_real()`, `GiNaC::numeric::is_zero()`, `GiNaC::ex::lhs()`, `normal()`, `GiNaC::ex::rhs()`, `GiNaC::ex::subs()`, and `x`.

**5.1.3.287 zeta()** [1/3]

```
template<typename T1 >
function GiNaC::zeta (
    const T1 & p1 ) [inline]
```

References `GiNaC::zeta1_SERIAL::serial`.

Referenced by `Li2_series()`, `Li_eval()`, `psi2_eval()`, `S_eval()`, `zeta1_eval()`, `zeta1_evalf()`, `zeta2_eval()`, `zeta2_evalf()`, and `zetaderiv_eval()`.

**5.1.3.288 zeta()** [2/3]

```
template<typename T1 , typename T2 >
function GiNaC::zeta (
    const T1 & p1,
    const T2 & p2 ) [inline]
```

References `GiNaC::zeta2_SERIAL::serial`.

**5.1.3.289 is\_the\_function< zeta\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< zeta_SERIAL > (
    const ex & x ) [inline]
```

References x.

**5.1.3.290 G() [1/2]**

```
template<typename T1 , typename T2 >
function GiNaC::G (
    const T1 & x,
    const T2 & y ) [inline]
```

References GiNaC::G2\_SERIAL::serial, and x.

Referenced by G2\_eval(), G2\_evalf(), G3\_eval(), and G3\_evalf().

**5.1.3.291 G() [2/2]**

```
template<typename T1 , typename T2 , typename T3 >
function GiNaC::G (
    const T1 & x,
    const T2 & s,
    const T3 & y ) [inline]
```

References GiNaC::G3\_SERIAL::serial, and x.

**5.1.3.292 is\_the\_function< G\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< G_SERIAL > (
    const ex & x ) [inline]
```

References x.

**5.1.3.293 psi() [1/4]**

```
template<typename T1 >
function GiNaC::psi (
    const T1 & p1 ) [inline]
```

References GiNaC::psi1\_SERIAL::serial.

Referenced by beta\_deriv(), lgamma\_deriv(), psi1\_deriv(), psi1\_eval(), psi1\_evalf(), psi1\_series(), psi2\_deriv(), psi2\_eval(), psi2\_evalf(), psi2\_series(), sr\_gcd(), and tgamma\_deriv().

**5.1.3.294** `psi()` [2/4]

```
template<typename T1 , typename T2 >
function GiNaC::psi (
    const T1 & p1,
    const T2 & p2 ) [inline]
```

References GiNaC::psi2\_SERIAL::serial.

**5.1.3.295** `is_the_function< psi_SERIAL >()`

```
template<>
bool GiNaC::is_the_function< psi_SERIAL > (
    const ex & x ) [inline]
```

References x.

**5.1.3.296** `iterated_integral()` [1/2]

```
template<typename T1 , typename T2 >
function GiNaC::iterated_integral (
    const T1 & kernel_lst,
    const T2 & lambda ) [inline]
```

References GiNaC::iterated\_integral2\_SERIAL::serial.

Referenced by iterated\_integral2\_eval(), iterated\_integral3\_eval(), and iterated\_integral\_evalf\_impl().

**5.1.3.297** `iterated_integral()` [2/2]

```
template<typename T1 , typename T2 , typename T3 >
function GiNaC::iterated_integral (
    const T1 & kernel_lst,
    const T2 & lambda,
    const T3 & N_trunc ) [inline]
```

References GiNaC::iterated\_integral3\_SERIAL::serial.

**5.1.3.298** `is_the_function< iterated_integral_SERIAL >()`

```
template<>
bool GiNaC::is_the_function< iterated_integral_SERIAL > (
    const ex & x ) [inline]
```

References x.



**5.1.3.299 is\_order\_function()**

```
bool GiNaC::is_order_function (
    const ex & e ) [inline]
```

Check whether a function is the Order ( $O(n)$ ) function.

References [is\\_ex\\_the\\_function](#).

Referenced by [GiNaC::pseries::add\\_series\(\)](#), [GiNaC::pseries::convert\\_to\\_poly\(\)](#), [GiNaC::pseries::derivative\(\)](#), [GiNaC::pseries::is\\_terminating\(\)](#), [GiNaC::pseries::mul\\_const\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::pseries::op\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::pseries::pseries\(\)](#), and [GiNaC::integral::series\(\)](#).

**5.1.3.300 convert\_H\_to\_Li()**

```
ex GiNaC::convert_H_to_Li (
    const ex & parameterlst,
    const ex & arg )
```

Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding [GiNaC](#) functions.

References [m](#), and [x](#).

**5.1.3.301 EllipticK\_evalf()**

```
static ex GiNaC::EllipticK_evalf (
    const ex & k ) [static]
```

References [GiNaC::ex::evalf\(\)](#), [k](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [sqrt\(\)](#).

**5.1.3.302 EllipticK\_eval()**

```
static ex GiNaC::EllipticK_eval (
    const ex & k ) [static]
```

References [\\_ex0](#), [GiNaC::info\\_flags::crational](#), [GiNaC::constant::evalf\(\)](#), [k](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

**5.1.3.303 EllipticK\_deriv()**

```
static ex GiNaC::EllipticK_deriv (
    const ex & k,
    unsigned deriv_param ) [static]
```

References k.

**5.1.3.304 EllipticK\_series()**

```
static ex GiNaC::EllipticK_series (
    const ex & k,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References \_ex0, \_ex1, \_ex\_1, binomial(), k, GiNaC::subs\_options::no\_pattern, order, Pi, pow(), GiNaC::ex↵::series(), and GiNaC::ex::subs().

**5.1.3.305 EllipticK\_print\_latex()**

```
static void GiNaC::EllipticK_print_latex (
    const ex & k,
    const print_context & c ) [static]
```

References c, and k.

**5.1.3.306 REGISTER\_FUNCTION()** [14/36]

```
GiNaC::REGISTER_FUNCTION (
    EllipticK ,
    evalf_func(EllipticK_evalf). eval_func(EllipticK_eval). derivative_func(EllipticK_deriv).
    series_func(EllipticK_series). print_func< print_latex >(EllipticK_print_latex). do_not_↵
    evalf_params() )
```

**5.1.3.307 EllipticE\_evalf()**

```
static ex GiNaC::EllipticE_evalf (
    const ex & k ) [static]
```

References GiNaC::ex::evalf(), k, GiNaC::info\_flags::numeric, Pi, and sqrt().

**5.1.3.308 EllipticE\_eval()**

```
static ex GiNaC::EllipticE_eval (
    const ex & k ) [static]
```

References `_ex0`, `_ex1`, `_ex_1`, `GiNaC::info_flags::crational`, `GiNaC::constant::evalf()`, `k`, `GiNaC::info_flags::numeric`, and `Pi`.

**5.1.3.309 EllipticE\_deriv()**

```
static ex GiNaC::EllipticE_deriv (
    const ex & k,
    unsigned deriv_param ) [static]
```

References `k`.

**5.1.3.310 EllipticE\_series()**

```
static ex GiNaC::EllipticE_series (
    const ex & k,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex0`, `_ex1`, `_ex_1`, `binomial()`, `k`, `GiNaC::subs_options::no_pattern`, `order`, `Pi`, `pow()`, `GiNaC::ex::series()`, and `GiNaC::ex::subs()`.

**5.1.3.311 EllipticE\_print\_latex()**

```
static void GiNaC::EllipticE_print_latex (
    const ex & k,
    const print_context & c ) [static]
```

References `c`, and `k`.

**5.1.3.312 REGISTER\_FUNCTION()** [15/36]

```
GiNaC::REGISTER_FUNCTION (
    EllipticE ,
    evalf_func(EllipticE_evalf). eval_func(EllipticE_eval). derivative_func(EllipticE_deriv).
    series_func(EllipticE_series). print_func< print_latex >(EllipticE_print_latex). do_not_evalf_params() )
```

#### 5.1.3.313 iterated\_integral\_evalf\_impl()

```
static ex GiNaC::iterated_integral_evalf_impl (
    const ex & kernel_lst,
    const ex & lambda,
    const ex & N_trunc ) [static]
```

References `coeff()`, `Digits`, `GiNaC::ex::evalf()`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `iterated_integral()`, `GiNaC::info_flags::list`, `GiNaC::info_flags::nonnegint`, `GiNaC::info_flags::numeric`, `one`, and `to_int()`.

Referenced by `iterated_integral2_evalf()`, and `iterated_integral3_evalf()`.

#### 5.1.3.314 iterated\_integral2\_evalf()

```
static ex GiNaC::iterated_integral2_evalf (
    const ex & kernel_lst,
    const ex & lambda ) [static]
```

References `iterated_integral_evalf_impl()`.

#### 5.1.3.315 iterated\_integral3\_evalf()

```
static ex GiNaC::iterated_integral3_evalf (
    const ex & kernel_lst,
    const ex & lambda,
    const ex & N_trunc ) [static]
```

References `iterated_integral_evalf_impl()`.

#### 5.1.3.316 iterated\_integral2\_eval()

```
static ex GiNaC::iterated_integral2_eval (
    const ex & kernel_lst,
    const ex & lambda ) [static]
```

References `GiNaC::info_flags::crational`, `GiNaC::function::evalf()`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `iterated_integral()`, and `GiNaC::info_flags::numeric`.

## 5.1.3.317 iterated\_integral3\_eval()

```
static ex GiNaC::iterated_integral3_eval (
    const ex & kernel_lst,
    const ex & lambda,
    const ex & N_trunc ) [static]
```

References [GiNaC::info\\_flags::crational](#), [GiNaC::function::evalf\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [iterated\\_integral\(\)](#), and [GiNaC::info\\_flags::numeric](#).

## 5.1.3.318 lgamma\_evalf()

```
static ex GiNaC::lgamma_evalf (
    const ex & x ) [static]
```

References [lgamma\(\)](#), and [x](#).

## 5.1.3.319 lgamma\_eval()

```
static ex GiNaC::lgamma_eval (
    const ex & x ) [static]
```

Evaluation of [lgamma\(x\)](#), the natural logarithm of the Gamma function.

Handles integer arguments as a special case.

## Exceptions

<a href="#">GiNaC::pole_error("lgamma_eval()")</a>	logarithmic pole",0)
--	----------------------

References [\\_ex\\_1](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [lgamma\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), and [x](#).

## 5.1.3.320 lgamma\_deriv()

```
static ex GiNaC::lgamma_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [GINAC\\_ASSERT](#), [psi\(\)](#), and [x](#).

5.1.3.321 `lgamma_series()`

```
static ex GiNaC::lgamma_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex1`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `lgamma()`, `log()`, `m`, `GiNaC::subs_options::no_↵`  
`pattern`, `options`, `order`, `GiNaC::info_flags::positive`, `series()`, and `GiNaC::ex::subs()`.

5.1.3.322 `lgamma_conjugate()`

```
static ex GiNaC::lgamma_conjugate (
    const ex & x ) [static]
```

References `GiNaC::ex::conjugate()`, `GiNaC::ex::imag_part()`, `GiNaC::ex::info()`, `GiNaC::ex::is_zero()`, `lgamma()`,  
`GiNaC::info_flags::positive`, and `x`.

5.1.3.323 `REGISTER_FUNCTION()` [16/36]

```
GiNaC::REGISTER_FUNCTION (
    lgamma ,
    eval_func(lgamma_eval) . evalf_func(lgamma_evalf) . derivative_func(lgamma_deriv) .
    series_func(lgamma_series) . conjugate_func(lgamma_conjugate) . latex_name("\g \mma" )
```

5.1.3.324 `tgamma_evalf()`

```
static ex GiNaC::tgamma_evalf (
    const ex & x ) [static]
```

References `tgamma()`, and `x`.

5.1.3.325 `tgamma_eval()`

```
static ex GiNaC::tgamma_eval (
    const ex & x ) [static]
```

Evaluation of `tgamma(x)`, the true Gamma function.

Knows about integer arguments, half-integer arguments and that's it. Somebody ought to provide some good numerical evaluation some day...

## Exceptions

<code>pole_error("tgamma_eval() simple pole",0)</code>
--

References `_num1_2_p`, `_num1_p`, `_num2_p`, `_num_2_p`, `abs()`, `doublefactorial()`, `factorial()`, `GiNaC::ex::info()`, `GiNaC::numeric::is_even()`, `GiNaC::numeric::is_integer()`, `GiNaC::numeric::is_positive()`, `n`, `GiNaC::info_flags::numeric`, `Pi`, `pow()`, `sqrt()`, `tgamma()`, and `x`.

5.1.3.326 `tgamma_deriv()`

```
static ex GiNaC::tgamma_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `GINAC_ASSERT`, `psi()`, `tgamma()`, and `x`.

5.1.3.327 `tgamma_series()`

```
static ex GiNaC::tgamma_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex1`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `m`, `GiNaC::subs_options::no_pattern`, `options`, `order`, `GiNaC::info_flags::positive`, `series()`, `GiNaC::ex::subs()`, and `tgamma()`.

5.1.3.328 `tgamma_conjugate()`

```
static ex GiNaC::tgamma_conjugate (
    const ex & x ) [static]
```

References `GiNaC::ex::conjugate()`, `tgamma()`, and `x`.

5.1.3.329 `REGISTER_FUNCTION()` [17/36]

```
GiNaC::REGISTER_FUNCTION (
    tgamma ,
    eval_func(tgamma_eval). evalf_func(tgamma_evalf). derivative_func(tgamma_deriv).
    series_func(tgamma_series). conjugate_func(tgamma_conjugate). latex_name("\mma" )
```

**5.1.3.330 beta\_evalf()**

```
static ex GiNaC::beta_evalf (
    const ex & x,
    const ex & y ) [static]
```

References `exp()`, `lgamma()`, and `x`.

**5.1.3.331 beta\_eval()**

```
static ex GiNaC::beta_eval (
    const ex & x,
    const ex & y ) [static]
```

References `_ex0`, `_ex1`, `_num_1_p`, `evalf()`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `GiNaC::numeric::is_integer()`, `is_integer()`, `GiNaC::numeric::is_negative()`, `is_positive()`, `is_rational()`, `GiNaC::numeric::is_real()`, `is_real()`, `GiNaC::info_flags::numeric`, `pow()`, `tgamma()`, and `x`.

**5.1.3.332 beta\_deriv()**

```
static ex GiNaC::beta_deriv (
    const ex & x,
    const ex & y,
    unsigned deriv_param ) [static]
```

References `GINAC_ASSERT`, `psi()`, and `x`.

**5.1.3.333 beta\_series()**

```
static ex GiNaC::beta_series (
    const ex & arg1,
    const ex & arg2,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `GINAC_ASSERT`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `GiNaC::relational::lhs()`, `GiNaC::subs_options::no_pattern`, `options`, `order`, `GiNaC::info_flags::positive`, `GiNaC::ex::subs()`, and `tgamma()`.

**5.1.3.334 REGISTER\_FUNCTION()** [18/36]

```
GiNaC::REGISTER_FUNCTION (
    beta ,
    eval_func(beta_eval). evalf_func(beta_evalf). derivative_func(beta_deriv).
    series_func(beta_series). latex_name("\mathrm{B}"). set_symmetry(sy_symm(0, 1)) )
```



5.1.3.335 `psi1_evalf()`

```
static ex GiNaC::psi1_evalf (
    const ex & x ) [static]
```

References `GiNaC::basic::hold()`, `psi()`, and `x`.

5.1.3.336 `psi1_eval()`

```
static ex GiNaC::psi1_eval (
    const ex & x ) [static]
```

Evaluation of digamma-function  $\psi(x)$ .

Somebody ought to provide some good numerical evaluation some day...

References `_ex1_2`, `_ex2`, `_num2_p`, `_num_1_p`, Euler, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `GiNaC::numeric::inverse()`, `GiNaC::numeric::is_integer()`, `is_integer()`, `GiNaC::numeric::is_positive()`, `log()`, `GiNaC::info_flags::numeric`, `pow()`, `psi()`, and `x`.

5.1.3.337 `psi1_deriv()`

```
static ex GiNaC::psi1_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex1`, `GINAC_ASSERT`, `psi()`, and `x`.

5.1.3.338 `psi1_series()`

```
static ex GiNaC::psi1_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex1`, `_ex_1`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `m`, `GiNaC::subs_options::no_pattern`, `options`, `order`, `GiNaC::info_flags::positive`, `psi()`, `series()`, and `GiNaC::ex::subs()`.

5.1.3.339 `psi2_evalf()`

```
static ex GiNaC::psi2_evalf (
    const ex & n,
    const ex & x ) [static]
```

References `GiNaC::basic::hold()`, `n`, `psi()`, and `x`.

5.1.3.340 `psi2_eval()`

```
static ex GiNaC::psi2_eval (
    const ex & n,
    const ex & x ) [static]
```

Evaluation of polygamma-function  $\psi(n,x)$ .

Somebody ought to provide some good numerical evaluation some day...

References `_ex1`, `_ex1_2`, `_ex_1`, `_num1_2_p`, `_num1_p`, `_num2_p`, `_num_1_p`, `factorial()`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `is_integer()`, `log()`, `m`, `n`, `GiNaC::info_flags::numeric`, `GiNaC::info_flags::posint`, `pow()`, `psi()`, `tgamma()`, `x`, and `zeta()`.

5.1.3.341 `psi2_deriv()`

```
static ex GiNaC::psi2_deriv (
    const ex & n,
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex1`, `GINAC_ASSERT`, `n`, `psi()`, and `x`.

5.1.3.342 `psi2_series()`

```
static ex GiNaC::psi2_series (
    const ex & n,
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex1`, `_ex_1`, `factorial()`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `m`, `n`, `GiNaC::subs_options::no_↵_pattern`, `options`, `order`, `GiNaC::info_flags::positive`, `psi()`, and `GiNaC::ex::subs()`.

**5.1.3.343 G2\_evalf()**

```
static ex GiNaC::G2_evalf (
    const ex & x_,
    const ex & y ) [static]
```

References `_ex0`, `_ex1`, `factorial()`, `G()`, `GiNaC::basic::hold()`, `imag()`, `GiNaC::ex::info()`, `is_real()`, `log()`, `GiNaC::ex::nops()`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `GiNaC::info_flags::positive`, `pow()`, and `x`.

**5.1.3.344 G2\_eval()**

```
static ex GiNaC::G2_eval (
    const ex & x_,
    const ex & y ) [static]
```

References `_ex0`, `_ex1`, `GiNaC::info_flags::crational`, `factorial()`, `G()`, `GiNaC::basic::hold()`, `imag()`, `GiNaC::ex::info()`, `log()`, `GiNaC::ex::nops()`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `GiNaC::info_flags::positive`, `pow()`, and `x`.

**5.1.3.345 G3\_evalf()**

```
static ex GiNaC::G3_evalf (
    const ex & x_,
    const ex & s_,
    const ex & y ) [static]
```

References `_ex0`, `_ex1`, `GiNaC::ex::begin()`, `GiNaC::container< C >::begin()`, `GiNaC::ex::end()`, `factorial()`, `G()`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `log()`, `GiNaC::container< C >::nops()`, `GiNaC::ex::nops()`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `GiNaC::info_flags::positive`, `pow()`, `GiNaC::info_flags::real`, and `x`.

**5.1.3.346 G3\_eval()**

```
static ex GiNaC::G3_eval (
    const ex & x_,
    const ex & s_,
    const ex & y ) [static]
```

References `_ex0`, `_ex1`, `GiNaC::ex::begin()`, `GiNaC::container< C >::begin()`, `GiNaC::info_flags::crational`, `GiNaC::ex::end()`, `factorial()`, `G()`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `log()`, `GiNaC::container< C >::nops()`, `GiNaC::ex::nops()`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `GiNaC::info_flags::positive`, `pow()`, `GiNaC::info_flags::real`, and `x`.

## 5.1.3.347 Li\_evalf()

```
static ex GiNaC::Li_evalf (
    const ex & m_,
    const ex & x_ ) [static]
```

References `_ex0`, `_ex1`, `GiNaC::ex::begin()`, `GiNaC::ex::evalf()`, `GiNaC::ex::info()`, `m`, `GiNaC::ex::nops()`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `GiNaC::info_flags::posint`, `to_int()`, and `x`.

## 5.1.3.348 Li\_eval()

```
static ex GiNaC::Li_eval (
    const ex & m_,
    const ex & x_ ) [static]
```

References `_ex0`, `_ex1`, `_ex2`, `_ex_1`, `_ex_48`, `GiNaC::container< C >::append()`, `GiNaC::ex::begin()`, `Catalan`, `GiNaC::info_flags::crational`, `GINAC_ASSERT`, `l`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `log()`, `m`, `GiNaC::ex::nops()`, `GiNaC::info_flags::numeric`, `Pi`, `GiNaC::info_flags::posint`, `pow()`, `to_int()`, `x`, and `zeta()`.

## 5.1.3.349 Li\_series()

```
static ex GiNaC::Li_series (
    const ex & m,
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex1`, `GiNaC::ex::info()`, `GiNaC::ex::is_zero()`, `m`, `GiNaC::subs_options::no_pattern`, `GiNaC::info_flags::numeric`, `order`, `pow()`, `GiNaC::ex::series()`, `GiNaC::ex::subs()`, and `x`.

## 5.1.3.350 Li\_deriv()

```
static ex GiNaC::Li_deriv (
    const ex & m_,
    const ex & x_,
    unsigned deriv_param ) [static]
```

References `_ex0`, `GINAC_ASSERT`, `m`, `GiNaC::ex::nops()`, `GiNaC::ex::op()`, and `x`.

**5.1.3.351 Li\_print\_latex()**

```
static void GiNaC::Li_print_latex (
    const ex & m_,
    const ex & x_,
    const print_context & c ) [static]
```

References `GiNaC::ex::begin()`, `c`, `GiNaC::ex::end()`, `m`, and `x`.

**5.1.3.352 REGISTER\_FUNCTION()** [19/36]

```
GiNaC::REGISTER_FUNCTION (
    Li ,
    evalf_func(Li_evalf). eval_func(Li_eval). series_func(Li_series). derivative_↵
func(Li_deriv). print_func< print_latex >(Li_print_latex). do_not_evalf_params() )
```

**5.1.3.353 S\_evalf()**

```
static ex GiNaC::S_evalf (
    const ex & n,
    const ex & p,
    const ex & x ) [static]
```

References `GiNaC::ex::evalf()`, `GiNaC::ex::info()`, `n`, `GiNaC::info_flags::posint`, `to_int()`, and `x`.

**5.1.3.354 S\_eval()**

```
static ex GiNaC::S_eval (
    const ex & n,
    const ex & p,
    const ex & x ) [static]
```

References `_ex0`, `GiNaC::info_flags::crational`, `factorial()`, `GiNaC::ex::info()`, `log()`, `m`, `n`, `GiNaC::info_flags::numeric`, `GiNaC::info_flags::posint`, `pow()`, `to_int()`, `x`, and `zeta()`.

**5.1.3.355 S\_series()**

```
static ex GiNaC::S_series (
    const ex & n,
    const ex & p,
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex1`, `GiNaC::ex::info()`, `GiNaC::ex::is_zero()`, `n`, `GiNaC::subs_options::no_pattern`, `GiNaC::info_flags::↵`  
`numeric`, `options`, `order`, `GiNaC::info_flags::posint`, `pow()`, `GiNaC::ex::series()`, `GiNaC::ex::subs()`, and `x`.

**5.1.3.356 S\_deriv()**

```
static ex GiNaC::S_deriv (
    const ex & n,
    const ex & p,
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex0`, `GINAC_ASSERT`, `n`, and `x`.

**5.1.3.357 S\_print\_latex()**

```
static void GiNaC::S_print_latex (
    const ex & n,
    const ex & p,
    const ex & x,
    const print_context & c ) [static]
```

References `c`, `n`, `GiNaC::ex::print()`, and `x`.

**5.1.3.358 REGISTER\_FUNCTION()** [20/36]

```
GiNaC::REGISTER_FUNCTION (
    S ,
    evalf_func(S_evalf). eval_func(S_eval). series_func(S_series). derivative_↵
func(S_deriv). print_func< print_latex >(S_print_latex). do_not_evalf_params() )
```

**5.1.3.359 H\_evalf()**

```
static ex GiNaC::H_evalf (
    const ex & x1,
    const ex & x2 ) [static]
```

References `abs()`, `GiNaC::container< C >::begin()`, `GiNaC::container< C >::end()`, `GiNaC::ex::evalf()`, `evalf()`, `l`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `m`, `GiNaC::container< C >::nops()`, `GiNaC::ex::nops()`, `GiNaC::↵`  
`::container< C >::op()`, `GiNaC::ex::op()`, `Pi`, `GiNaC::ex::subs()`, `to_int()`, and `x`.

**5.1.3.360 H\_eval()**

```
static ex GiNaC::H_eval (
    const ex & m_,
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `_ex_1`, `GiNaC::info_flags::crational`, `GiNaC::ex::evalf()`, `factorial()`, `GiNaC::ex::info()`, `Gi↵`  
`NaC::info_flags::integer`, `log()`, `m`, `n`, `GiNaC::info_flags::numeric`, `pow()`, `step()`, and `x`.

**5.1.3.361 H\_series()**

```
static ex GiNaC::H_series (
    const ex & m,
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [m](#), and [x](#).

**5.1.3.362 H\_deriv()**

```
static ex GiNaC::H_deriv (
    const ex & m_,
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GINAC\\_ASSERT](#), [m](#), and [x](#).

**5.1.3.363 H\_print\_latex()**

```
static void GiNaC::H_print_latex (
    const ex & m_,
    const ex & x,
    const print\_context & c ) [static]
```

References [c](#), [m](#), [GiNaC::ex::print\(\)](#), and [x](#).

**5.1.3.364 REGISTER\_FUNCTION()** [21/36]

```
GiNaC::REGISTER_FUNCTION (
    H ,
    evalf_func(H\_evalf) . eval_func(H\_eval) . series_func(H\_series) . derivative_↵
func(H\_deriv) . print_func< print\_latex >(H\_print\_latex) . do_not_evalf_params() )
```

**5.1.3.365 zeta1\_evalf()**

```
static ex GiNaC::zeta1_evalf (
    const ex & x ) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [Digits](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags](#)↵  
::posint, [r](#), [x](#), and [zeta\(\)](#).

Referenced by [zeta1\\_eval\(\)](#).

**5.1.3.366 zeta1\_eval()**

```
static ex GiNaC::zeta1_eval (
    const ex & m ) [static]
```

References `_ex0`, `_ex1_2`, `_num1_p`, `_num2_p`, `abs()`, `bernoulli()`, `GiNaC::info_flags::crational`, `factorial()`, `GiNaC::basic::hold()`, `GiNaC::numeric::info()`, `GiNaC::numeric::is_equal()`, `GiNaC::numeric::is_integer()`, `GiNaC::numeric::is_zero()`, `m`, `GiNaC::info_flags::numeric`, `GiNaC::info_flags::odd`, `Pi`, `GiNaC::info_flags::posint`, `pow()`, `zeta()`, and `zeta1_evalf()`.

**5.1.3.367 zeta1\_deriv()**

```
static ex GiNaC::zeta1_deriv (
    const ex & m,
    unsigned deriv_param ) [static]
```

References `_ex0`, `_ex1`, `GINAC_ASSERT`, and `m`.

**5.1.3.368 zeta1\_print\_latex()**

```
static void GiNaC::zeta1_print_latex (
    const ex & m_,
    const print_context & c ) [static]
```

References `c`, `m`, and `GiNaC::ex::print()`.

**5.1.3.369 zeta2\_evalf()**

```
static ex GiNaC::zeta2_evalf (
    const ex & x,
    const ex & s ) [static]
```

References `GiNaC::container< C >::begin()`, `evalf()`, `GiNaC::basic::hold()`, `GiNaC::ex::nops()`, `GiNaC::info_flags::posint`, `x`, and `zeta()`.

**5.1.3.370 zeta2\_eval()**

```
static ex GiNaC::zeta2_eval (
    const ex & m,
    const ex & s_ ) [static]
```

References `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `m`, `GiNaC::info_flags::positive`, and `zeta()`.



**5.1.3.371 zeta2\_deriv()**

```
static ex GiNaC::zeta2_deriv (
    const ex & m,
    const ex & s,
    unsigned deriv_param ) [static]
```

References `_ex0`, `_ex1`, `GINAC_ASSERT`, `GiNaC::ex::info()`, `m`, `GiNaC::ex::op()`, and `GiNaC::info_flags::positive`.

**5.1.3.372 zeta2\_print\_latex()**

```
static void GiNaC::zeta2_print_latex (
    const ex & m_,
    const ex & s_,
    const print_context & c ) [static]
```

References `GiNaC::container< C >::begin()`, `c`, and `m`.

**5.1.3.373 exp\_evalf()**

```
static ex GiNaC::exp_evalf (
    const ex & x ) [static]
```

References `exp()`, `GiNaC::basic::hold()`, and `x`.

**5.1.3.374 exp\_eval()**

```
static ex GiNaC::exp_eval (
    const ex & x ) [static]
```

References `_ex1`, `_ex2`, `_ex_1`, `_num0_p`, `_num1_p`, `_num2_p`, `_num3_p`, `_num4_p`, `GiNaC::info_flags::crational`, `exp()`, `GiNaC::basic::hold()`, `l`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `GiNaC::numeric::is_equal()`, `is_ex_↵` the\_function, `GiNaC::ex::is_zero()`, `log()`, `mod()`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `Pi`, and `x`.

**5.1.3.375 exp\_expand()**

```
static ex GiNaC::exp_expand (
    const ex & arg,
    unsigned options ) [static]
```

References `GiNaC::ex::begin()`, `GiNaC::ex::end()`, `exp()`, `GiNaC::ex::expand()`, `GiNaC::expand_options::expand_↵` \_function\_args, `GiNaC::expand_options::expand_transcendental`, `GiNaC::status_flags::expanded`, `GiNaC::basic_↵` ::hold(), `GiNaC::ex::nops()`, and `options`.

**5.1.3.376 exp\_deriv()**

```
static ex GiNaC::exp_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `exp()`, `GINAC_ASSERT`, and `x`.

**5.1.3.377 exp\_real\_part()**

```
static ex GiNaC::exp_real_part (
    const ex & x ) [static]
```

References `cos()`, `exp()`, `imag_part()`, `real_part()`, and `x`.

**5.1.3.378 exp\_imag\_part()**

```
static ex GiNaC::exp_imag_part (
    const ex & x ) [static]
```

References `exp()`, `imag_part()`, `real_part()`, `sin()`, and `x`.

**5.1.3.379 exp\_conjugate()**

```
static ex GiNaC::exp_conjugate (
    const ex & x ) [static]
```

References `GiNaC::ex::conjugate()`, `exp()`, and `x`.

**5.1.3.380 exp\_power()**

```
static ex GiNaC::exp_power (
    const ex & x,
    const ex & a ) [static]
```

References `_ex_1`, `exp()`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::nonnegative`, `GiNaC::info_flags::real`, and `x`.

## 5.1.3.381 REGISTER\_FUNCTION() [22/36]

```
GiNaC::REGISTER_FUNCTION (
    exp ,
    eval_func(exp_eval). evalf_func(exp_evalf). expand_func(exp_expand). derivative←
_func(exp_deriv). real_part_func(exp_real_part). imag_part_func(exp_imag_part). conjugate←
_func(exp_conjugate). power_func(exp_power). latex_name("\p") )
```

## 5.1.3.382 log\_evalf()

```
static ex GiNaC::log_evalf (
    const ex & x ) [static]
```

References `GiNaC::basic::hold()`, `log()`, and `x`.

## 5.1.3.383 log\_eval()

```
static ex GiNaC::log_eval (
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `_ex1_2`, `_ex_1_2`, `GiNaC::info_flags::crational`, `exp()`, `GiNaC::basic::hold()`, `I`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `is_ex_the_function`, `GiNaC::ex::is_zero()`, `log()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `Pi`, `GiNaC::info_flags::rational`, `GiNaC::info_flags::real`, and `x`.

## 5.1.3.384 log\_deriv()

```
static ex GiNaC::log_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex_1`, `GINAC_ASSERT`, and `x`.

## 5.1.3.385 log\_series()

```
static ex GiNaC::log_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex0`, `_ex1`, `_ex_1`, `GiNaC::pseries::add_series()`, `GiNaC::pseries::coeff()`, `coeff()`, `csgn()`, `GiNaC::ex::diff()`, `GINAC_ASSERT`, `I`, `GiNaC::ex::info()`, `GiNaC::pseries::is_terminating()`, `GiNaC::ex::is_zero()`, `GiNaC::pseries::ldegree()`, `GiNaC::relational::lhs()`, `log()`, `n`, `GiNaC::info_flags::negative`, `GiNaC::subs_options::no_pattern`, `GiNaC::pseries::nops()`, `options`, `order`, `Pi`, `GiNaC::info_flags::positive`, `pow()`, `GiNaC::relational::rhs()`, `GiNaC::ex::series()`, `series()`, `GiNaC::ex::subs()`, and `GiNaC::series_options::suppress_branchcut`.

**5.1.3.386 log\_real\_part()**

```
static ex GiNaC::log_real_part (
    const ex & x ) [static]
```

References `abs()`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `log()`, `GiNaC::info_flags::nonnegative`, and `x`.

**5.1.3.387 log\_imag\_part()**

```
static ex GiNaC::log_imag_part (
    const ex & x ) [static]
```

References `imag_part()`, `GiNaC::ex::info()`, `GiNaC::info_flags::nonnegative`, `real_part()`, and `x`.

**5.1.3.388 log\_expand()**

```
static ex GiNaC::log_expand (
    const ex & arg,
    unsigned options ) [static]
```

References `_ex1`, `_ex_1`, `GiNaC::ex::begin()`, `GiNaC::ex::end()`, `GiNaC::ex::expand()`, `expand()`, `GiNaC::expand_↵_options::expand_function_args`, `GiNaC::expand_options::expand_transcendental`, `GiNaC::basic::hold()`, `GiNaC::↵info_flags::indefinite`, `GiNaC::ex::info()`, `log()`, `GiNaC::info_flags::negative`, `GiNaC::ex::nops()`, `options`, `GiNaC::↵info_flags::positive`, and `GiNaC::status_flags::purely_indefinite`.

**5.1.3.389 log\_conjugate()**

```
static ex GiNaC::log_conjugate (
    const ex & x ) [static]
```

References `GiNaC::ex::conjugate()`, `GiNaC::basic::hold()`, `GiNaC::ex::imag_part()`, `GiNaC::ex::info()`, `GiNaC::ex::↵is_zero()`, `log()`, `GiNaC::info_flags::positive`, and `x`.

**5.1.3.390 REGISTER\_FUNCTION()** [23/36]

```
GiNaC::REGISTER_FUNCTION (
    log ,
    eval_func(log_eval). evalf_func(log_evalf). expand_func(log_expand). derivative↵_func(log_deriv). series_func(log_series). real_part_func(log_real_part). imag_part_↵func(log_imag_part). conjugate_func(log_conjugate). latex_name("\") )
```

**5.1.3.391 sin\_evalf()**

```
static ex GiNaC::sin_evalf (
    const ex & x ) [static]
```

References `GiNaC::basic::hold()`, `sin()`, and `x`.

**5.1.3.392 sin\_eval()**

```
static ex GiNaC::sin_eval (
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `_ex1_2`, `_ex1_3`, `_ex1_4`, `_ex2`, `_ex3`, `_ex5`, `_ex6`, `_ex60`, `_ex_1`, `_ex_1_2`, `_ex_1_3`, `_ex_1_4`, `_num0_p`, `_num10_p`, `_num120_p`, `_num15_p`, `_num18_p`, `_num20_p`, `_num25_p`, `_num30_p`, `_num5_p`, `_num60_p`, `_num6_p`, `acos()`, `asin()`, `atan()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `GiNaC::numeric::is_equal()`, `is_ex_the_function`, `mod()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `Pi`, `sin()`, `sqrt()`, and `x`.

**5.1.3.393 sin\_deriv()**

```
static ex GiNaC::sin_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `cos()`, `GINAC_ASSERT`, and `x`.

**5.1.3.394 sin\_real\_part()**

```
static ex GiNaC::sin_real_part (
    const ex & x ) [static]
```

References `cosh()`, `imag_part()`, `real_part()`, `sin()`, and `x`.

**5.1.3.395 sin\_imag\_part()**

```
static ex GiNaC::sin_imag_part (
    const ex & x ) [static]
```

References `cos()`, `imag_part()`, `real_part()`, `sinh()`, and `x`.

**5.1.3.396 sin\_conjugate()**

```
static ex GiNaC::sin_conjugate (
    const ex & x ) [static]
```

References GiNaC::ex::conjugate(), sin(), and x.

**5.1.3.397 REGISTER\_FUNCTION()** [24/36]

```
GiNaC::REGISTER_FUNCTION (
    sin ,
    eval_func(sin_eval). evalf_func(sin_evalf). derivative_func(sin_deriv). real↔
_part_func(sin_real_part). imag_part_func(sin_imag_part). conjugate_func(sin_conjugate).
    latex_name("\n" ) )
```

**5.1.3.398 cos\_evalf()**

```
static ex GiNaC::cos_evalf (
    const ex & x ) [static]
```

References cos(), GiNaC::basic::hold(), and x.

**5.1.3.399 cos\_eval()**

```
static ex GiNaC::cos_eval (
    const ex & x ) [static]
```

References \_ex0, \_ex1, \_ex1\_2, \_ex1\_3, \_ex1\_4, \_ex2, \_ex3, \_ex5, \_ex6, \_ex60, \_ex\_1, \_ex\_1\_2, \_ex\_1\_3, \_↔  
ex\_1\_4, \_num0\_p, \_num10\_p, \_num120\_p, \_num12\_p, \_num15\_p, \_num20\_p, \_num24\_p, \_num25\_p, \_num30↔  
\_p, \_num5\_p, \_num60\_p, acos(), asin(), atan(), cos(), GiNaC::info\_flags::crational, GiNaC::basic::hold(), GiNaC↔  
::ex::info(), GiNaC::info\_flags::integer, GiNaC::numeric::is\_equal(), is\_ex\_the\_function, mod(), GiNaC::info\_flags↔  
::negative, GiNaC::info\_flags::numeric, GiNaC::ex::op(), Pi, sqrt(), and x.

**5.1.3.400 cos\_deriv()**

```
static ex GiNaC::cos_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References GINAC\_ASSERT, sin(), and x.

**5.1.3.401 cos\_real\_part()**

```
static ex GiNaC::cos_real_part (
    const ex & x ) [static]
```

References `cos()`, `cosh()`, `imag_part()`, `real_part()`, and `x`.

**5.1.3.402 cos\_imag\_part()**

```
static ex GiNaC::cos_imag_part (
    const ex & x ) [static]
```

References `imag_part()`, `real_part()`, `sin()`, `sinh()`, and `x`.

**5.1.3.403 cos\_conjugate()**

```
static ex GiNaC::cos_conjugate (
    const ex & x ) [static]
```

References `GiNaC::ex::conjugate()`, `cos()`, and `x`.

**5.1.3.404 REGISTER\_FUNCTION()** [25/36]

```
GiNaC::REGISTER_FUNCTION (
    cos ,
    eval_func(cos_eval). evalf_func(cos_evalf). derivative_func(cos_deriv). real↔
_part_func(cos_real_part). imag_part_func(cos_imag_part). conjugate_func(cos_conjugate).
    latex_name("\s" ) )
```

**5.1.3.405 tan\_evalf()**

```
static ex GiNaC::tan_evalf (
    const ex & x ) [static]
```

References `GiNaC::basic::hold()`, `tan()`, and `x`.

**5.1.3.406 tan\_eval()**

```
static ex GiNaC::tan_eval (
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `_ex1_3`, `_ex2`, `_ex3`, `_ex60`, `_ex_1`, `_ex_1_2`, `_num0_p`, `_num10_p`, `_num15_p`, `_num20_p`, `_num25_p`, `_num30_p`, `_num5_p`, `_num60_p`, `acos()`, `asin()`, `atan()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `GiNaC::numeric::is_equal()`, `is_ex_the_function`, `mod()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `Pi`, `sqrt()`, `tan()`, and `x`.

**5.1.3.407 tan\_deriv()**

```
static ex GiNaC::tan_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex1`, `_ex2`, `GINAC_ASSERT`, `tan()`, and `x`.

**5.1.3.408 tan\_real\_part()**

```
static ex GiNaC::tan_real_part (
    const ex & x ) [static]
```

References `imag_part()`, `real_part()`, `tan()`, and `x`.

**5.1.3.409 tan\_imag\_part()**

```
static ex GiNaC::tan_imag_part (
    const ex & x ) [static]
```

References `imag_part()`, `real_part()`, `tan()`, `tanh()`, and `x`.

**5.1.3.410 tan\_series()**

```
static ex GiNaC::tan_series (
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `cos()`, `GINAC_ASSERT`, `GiNaC::relational::lhs()`, `GiNaC::subs_options::no_pattern`, `GiNaC::info_flags::odd`, `options`, `order`, `Pi`, `sin()`, `GiNaC::ex::subs()`, and `x`.



5.1.3.411 `tan_conjugate()`

```
static ex GiNaC::tan_conjugate (
    const ex & x ) [static]
```

References `GiNaC::ex::conjugate()`, `tan()`, and `x`.

5.1.3.412 `REGISTER_FUNCTION()` [26/36]

```
GiNaC::REGISTER_FUNCTION (
    tan ,
    eval_func(tan_eval). evalf_func(tan_evalf). derivative_func(tan_deriv). series↵
_func(tan_series). real_part_func(tan_real_part). imag_part_func(tan_imag_part). conjugate↵
_func(tan_conjugate). latex_name("\n") )
```

5.1.3.413 `asin_evalf()`

```
static ex GiNaC::asin_evalf (
    const ex & x ) [static]
```

References `asin()`, `GiNaC::basic::hold()`, and `x`.

5.1.3.414 `asin_eval()`

```
static ex GiNaC::asin_eval (
    const ex & x ) [static]
```

References `_ex1`, `_ex1_2`, `_ex_1`, `_ex_1_2`, `asin()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `GiNaC::ex↵`  
`::info()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, `Pi`,  
and `x`.

5.1.3.415 `asin_deriv()`

```
static ex GiNaC::asin_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex2`, `_ex_1_2`, `GINAC_ASSERT`, and `x`.

**5.1.3.416 asin\_conjugate()**

```
static ex GiNaC::asin_conjugate (
    const ex & x ) [static]
```

References `_num1_p`, `_num_1_p`, `asin()`, `GiNaC::ex::conjugate()`, `GiNaC::basic::hold()`, `GiNaC::ex::imag_part()`, `GiNaC::ex::is_zero()`, and `x`.

**5.1.3.417 REGISTER\_FUNCTION()** [27/36]

```
GiNaC::REGISTER_FUNCTION (
    asin ,
    eval_func(asin_eval) . evalf_func(asin_evalf) . derivative_func(asin_deriv) .
    conjugate_func(asin_conjugate) . latex_name("\csin" ) )
```

**5.1.3.418 acos\_evalf()**

```
static ex GiNaC::acos_evalf (
    const ex & x ) [static]
```

References `acos()`, `GiNaC::basic::hold()`, and `x`.

**5.1.3.419 acos\_eval()**

```
static ex GiNaC::acos_eval (
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `_ex1_2`, `_ex1_3`, `_ex_1`, `_ex_1_2`, `acos()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, `Pi`, and `x`.

**5.1.3.420 acos\_deriv()**

```
static ex GiNaC::acos_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex2`, `_ex_1_2`, `GINAC_ASSERT`, and `x`.

**5.1.3.421** `acos_conjugate()`

```
static ex GiNaC::acos_conjugate (
    const ex & x ) [static]
```

References `_num1_p`, `_num_1_p`, `acos()`, `GiNaC::ex::conjugate()`, `GiNaC::basic::hold()`, `GiNaC::ex::imag_part()`, `GiNaC::ex::is_zero()`, and `x`.

**5.1.3.422** `REGISTER_FUNCTION()` [28/36]

```
GiNaC::REGISTER_FUNCTION (
    acos ,
    eval_func(acos_eval) . evalf_func(acos_evalf) . derivative_func(acos_deriv) .
    conjugate_func(acos_conjugate) . latex_name("\ccos") )
```

**5.1.3.423** `atan_evalf()`

```
static ex GiNaC::atan_evalf (
    const ex & x ) [static]
```

References `atan()`, `GiNaC::basic::hold()`, and `x`.

**5.1.3.424** `atan_eval()`

```
static ex GiNaC::atan_eval (
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `_ex1_4`, `_ex_1`, `_ex_1_4`, `atan()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `I`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, `Pi`, and `x`.

**5.1.3.425** `atan_deriv()`

```
static ex GiNaC::atan_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex1`, `_ex2`, `_ex_1`, `GINAC_ASSERT`, and `x`.

## 5.1.3.426 atan\_series()

```
static ex GiNaC::atan_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex0`, `_ex1`, `_ex1_2`, `_ex_1`, `_ex_1_2`, `abs()`, `atan()`, `csgn()`, `GINAC_ASSERT`, `I`, `GiNaC::relational::lhs()`, `log()`, `GiNaC::subs_options::no_pattern`, `GiNaC::ex::op()`, `options`, `order`, `Pi`, `GiNaC::info_flags::real`, `GiNaC::relational::rhs()`, `series()`, `GiNaC::ex::subs()`, and `GiNaC::series_options::suppress_branchcut`.

## 5.1.3.427 atan\_conjugate()

```
static ex GiNaC::atan_conjugate (
    const ex & x ) [static]
```

References `_num1_p`, `_num_1_p`, `atan()`, `GiNaC::ex::conjugate()`, `GiNaC::basic::hold()`, `GiNaC::ex::imag_part()`, `GiNaC::ex::info()`, `GiNaC::numeric::is_zero()`, `GiNaC::info_flags::real`, `GiNaC::ex::real_part()`, and `x`.

## 5.1.3.428 REGISTER\_FUNCTION() [29/36]

```
GiNaC::REGISTER_FUNCTION (
    atan ,
    eval_func(atan_eval). evalf_func(atan_evalf). derivative_func(atan_deriv).
    series_func(atan_series). conjugate_func(atan_conjugate). latex_name("\ctan") )
```

## 5.1.3.429 atan2\_evalf()

```
static ex GiNaC::atan2_evalf (
    const ex & y,
    const ex & x ) [static]
```

References `atan()`, and `x`.

## 5.1.3.430 atan2\_eval()

```
static ex GiNaC::atan2_eval (
    const ex & y,
    const ex & x ) [static]
```

References `_ex0`, `_ex1_2`, `_ex1_4`, `_ex_1_2`, `_ex_1_4`, `atan()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::negative`, `Pi`, `GiNaC::info_flags::positive`, `GiNaC::info_flags::real`, and `x`.

**5.1.3.431 atan2\_deriv()**

```
static ex GiNaC::atan2_deriv (
    const ex & y,
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex2`, `_ex_1`, `GINAC_ASSERT`, and `x`.

**5.1.3.432 REGISTER\_FUNCTION()** [30/36]

```
GiNaC::REGISTER_FUNCTION (
    atan2 ,
    eval_func(atan2_eval). evalf_func(atan2_evalf). derivative_func(atan2_deriv) )
```

**5.1.3.433 sinh\_evalf()**

```
static ex GiNaC::sinh_evalf (
    const ex & x ) [static]
```

References `GiNaC::basic::hold()`, `sinh()`, and `x`.

**5.1.3.434 sinh\_eval()**

```
static ex GiNaC::sinh_eval (
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `_ex2`, `_ex_1_2`, `acosh()`, `asinh()`, `atanh()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `I`, `GiNaC::ex::info()`, `is_ex_the_function`, `GiNaC::numeric::is_zero()`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `Pi`, `real()`, `sin()`, `sinh()`, `sqrt()`, and `x`.

**5.1.3.435 sinh\_deriv()**

```
static ex GiNaC::sinh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `cosh()`, `GINAC_ASSERT`, and `x`.

**5.1.3.436 sinh\_real\_part()**

```
static ex GiNaC::sinh_real_part (
    const ex & x ) [static]
```

References `cos()`, `imag_part()`, `real_part()`, `sinh()`, and `x`.

**5.1.3.437 sinh\_imag\_part()**

```
static ex GiNaC::sinh_imag_part (
    const ex & x ) [static]
```

References `cosh()`, `imag_part()`, `real_part()`, `sin()`, and `x`.

**5.1.3.438 sinh\_conjugate()**

```
static ex GiNaC::sinh_conjugate (
    const ex & x ) [static]
```

References `GiNaC::ex::conjugate()`, `sinh()`, and `x`.

**5.1.3.439 REGISTER\_FUNCTION()** [31/36]

```
GiNaC::REGISTER_FUNCTION (
    sinh ,
    eval_func(sinh_eval) . evalf_func(sinh_evalf) . derivative_func(sinh_deriv) .
    real_part_func(sinh_real_part) . imag_part_func(sinh_imag_part) . conjugate_func(sinh_conjugate) .
    latex_name("\nh") )
```

**5.1.3.440 cosh\_evalf()**

```
static ex GiNaC::cosh_evalf (
    const ex & x ) [static]
```

References `cosh()`, `GiNaC::basic::hold()`, and `x`.

**5.1.3.441 cosh\_eval()**

```
static ex GiNaC::cosh_eval (
    const ex & x ) [static]
```

References `_ex1`, `_ex2`, `_ex_1_2`, `acosh()`, `asinh()`, `atanh()`, `cos()`, `cosh()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `I`, `GiNaC::ex::info()`, `is_ex_the_function`, `GiNaC::numeric::is_zero()`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `Pi`, `real()`, `sqrt()`, and `x`.

**5.1.3.442 cosh\_deriv()**

```
static ex GiNaC::cosh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `GINAC_ASSERT`, `sinh()`, and `x`.

**5.1.3.443 cosh\_real\_part()**

```
static ex GiNaC::cosh_real_part (
    const ex & x ) [static]
```

References `cos()`, `cosh()`, `imag_part()`, `real_part()`, and `x`.

**5.1.3.444 cosh\_imag\_part()**

```
static ex GiNaC::cosh_imag_part (
    const ex & x ) [static]
```

References `imag_part()`, `real_part()`, `sin()`, `sinh()`, and `x`.

**5.1.3.445 cosh\_conjugate()**

```
static ex GiNaC::cosh_conjugate (
    const ex & x ) [static]
```

References `GiNaC::ex::conjugate()`, `cosh()`, and `x`.

**5.1.3.446 REGISTER\_FUNCTION()** [32/36]

```
GiNaC::REGISTER_FUNCTION (
    cosh ,
    eval_func(cosh_eval). evalf_func(cosh_evalf). derivative_func(cosh_deriv).
    real_part_func(cosh_real_part). imag_part_func(cosh_imag_part). conjugate_func(cosh_conjugate).
    latex_name("\\sh") )
```

**5.1.3.447 tanh\_evalf()**

```
static ex GiNaC::tanh_evalf (
    const ex & x ) [static]
```

References GiNaC::basic::hold(), tanh(), and x.

**5.1.3.448 tanh\_eval()**

```
static ex GiNaC::tanh_eval (
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `_ex2`, `_ex_1`, `_ex_1_2`, `acosh()`, `asinh()`, `atanh()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `I`, `GiNaC::ex::info()`, `is_ex_the_function`, `GiNaC::numeric::is_zero()`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, `GiNaC::ex::op()`, `Pi`, `real()`, `sqrt()`, `tan()`, `tanh()`, and `x`.

**5.1.3.449 tanh\_deriv()**

```
static ex GiNaC::tanh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex1`, `_ex2`, `GINAC_ASSERT`, `tanh()`, and `x`.

**5.1.3.450 tanh\_series()**

```
static ex GiNaC::tanh_series (
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `cosh()`, `GINAC_ASSERT`, `I`, `GiNaC::relational::lhs()`, `GiNaC::subs_options::no_pattern`, `GiNaC::info_flags::odd`, `options`, `order`, `Pi`, `sinh()`, `GiNaC::ex::subs()`, and `x`.



**5.1.3.451 tanh\_real\_part()**

```
static ex GiNaC::tanh_real_part (
    const ex & x ) [static]
```

References `imag_part()`, `real_part()`, `tan()`, `tanh()`, and `x`.

**5.1.3.452 tanh\_imag\_part()**

```
static ex GiNaC::tanh_imag_part (
    const ex & x ) [static]
```

References `imag_part()`, `real_part()`, `tan()`, `tanh()`, and `x`.

**5.1.3.453 tanh\_conjugate()**

```
static ex GiNaC::tanh_conjugate (
    const ex & x ) [static]
```

References `GiNaC::ex::conjugate()`, `tanh()`, and `x`.

**5.1.3.454 REGISTER\_FUNCTION()** [33/36]

```
GiNaC::REGISTER_FUNCTION (
    tanh ,
    eval_func(tanh_eval). evalf_func(tanh_evalf). derivative_func(tanh_deriv).
    series_func(tanh_series). real_part_func(tanh_real_part). imag_part_func(tanh_imag_part).
    conjugate_func(tanh_conjugate). latex_name("\nh" )
```

**5.1.3.455 asinh\_evalf()**

```
static ex GiNaC::asinh_evalf (
    const ex & x ) [static]
```

References `asinh()`, `GiNaC::basic::hold()`, and `x`.

**5.1.3.456 asinh\_eval()**

```
static ex GiNaC::asinh_eval (
    const ex & x ) [static]
```

References `_ex0`, `asinh()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, and `x`.

**5.1.3.457 asinh\_deriv()**

```
static ex GiNaC::asinh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex1`, `_ex2`, `_ex_1_2`, `GINAC_ASSERT`, and `x`.

**5.1.3.458 asinh\_conjugate()**

```
static ex GiNaC::asinh_conjugate (
    const ex & x ) [static]
```

References `_num1_p`, `_num_1_p`, `asinh()`, `GiNaC::ex::conjugate()`, `GiNaC::basic::hold()`, `GiNaC::ex::imag_part()`, `GiNaC::ex::info()`, `GiNaC::numeric::is_zero()`, `GiNaC::info_flags::real`, `GiNaC::ex::real_part()`, and `x`.

**5.1.3.459 REGISTER\_FUNCTION()** [34/36]

```
GiNaC::REGISTER_FUNCTION (
    asinh ,
    eval_func(asinh_eval) . evalf_func(asinh_evalf) . derivative_func(asinh_deriv) .
    conjugate_func(asinh_conjugate) )
```

**5.1.3.460 acosh\_evalf()**

```
static ex GiNaC::acosh_evalf (
    const ex & x ) [static]
```

References `acosh()`, `GiNaC::basic::hold()`, and `x`.

**5.1.3.461    acosh\_eval()**

```
static ex GiNaC::acosh_eval (
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `_ex_1`, `acosh()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `I`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, `Pi`, and `x`.

**5.1.3.462    acosh\_deriv()**

```
static ex GiNaC::acosh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex1`, `_ex_1`, `_ex_1_2`, `GINAC_ASSERT`, and `x`.

**5.1.3.463    acosh\_conjugate()**

```
static ex GiNaC::acosh_conjugate (
    const ex & x ) [static]
```

References `_num1_p`, `acosh()`, `GiNaC::ex::conjugate()`, `GiNaC::basic::hold()`, `GiNaC::ex::imag_part()`, `GiNaC::ex::is_zero()`, and `x`.

**5.1.3.464    REGISTER\_FUNCTION()** [35/36]

```
GiNaC::REGISTER_FUNCTION (
    acosh ,
    eval_func(acosh_eval) . evalf_func(acosh_evalf) . derivative_func(acosh_deriv) .
    conjugate_func(acosh_conjugate) )
```

**5.1.3.465    atanh\_evalf()**

```
static ex GiNaC::atanh_evalf (
    const ex & x ) [static]
```

References `atanh()`, `GiNaC::basic::hold()`, and `x`.

**5.1.3.466 atanh\_eval()**

```
static ex GiNaC::atanh_eval (
    const ex & x ) [static]
```

References `_ex0`, `_ex1`, `_ex_1`, `atanh()`, `GiNaC::info_flags::crational`, `GiNaC::basic::hold()`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, and `x`.

**5.1.3.467 atanh\_deriv()**

```
static ex GiNaC::atanh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References `_ex1`, `_ex2`, `_ex_1`, `GINAC_ASSERT`, and `x`.

**5.1.3.468 atanh\_series()**

```
static ex GiNaC::atanh_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References `_ex0`, `_ex1`, `_ex1_2`, `_ex_1`, `_ex_1_2`, `abs()`, `atanh()`, `csgn()`, `GINAC_ASSERT`, `I`, `GiNaC::ex::is_equal()`, `GiNaC::relational::lhs()`, `log()`, `GiNaC::subs_options::no_pattern`, `GiNaC::ex::op()`, `options`, `order`, `Pi`, `GiNaC::info_flags::real`, `GiNaC::relational::rhs()`, `series()`, `GiNaC::ex::subs()`, and `GiNaC::series_options::suppress_branchcut`.

**5.1.3.469 atanh\_conjugate()**

```
static ex GiNaC::atanh_conjugate (
    const ex & x ) [static]
```

References `_num1_p`, `_num_1_p`, `atanh()`, `GiNaC::ex::conjugate()`, `GiNaC::basic::hold()`, `GiNaC::ex::imag_part()`, `GiNaC::ex::is_zero()`, and `x`.

**5.1.3.470 REGISTER\_FUNCTION()** [36/36]

```
GiNaC::REGISTER_FUNCTION (
    atanh ,
    eval_func(atanh_eval) . evalf_func(atanh_evalf) . derivative_func(atanh_deriv) .
    series_func(atanh_series) . conjugate_func(atanh_conjugate) )
```

## 5.1.3.471 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [10/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    integral ,
    basic ,
    print_func< print_dflt > &::do_print.  print_func< print_python > &::do_print.
    print_func< print_latex > &::do_print_latex )
```

## 5.1.3.472 subsvalue()

```
ex GiNaC::subsvalue (
    const ex & var,
    const ex & value,
    const ex & fun )
```

References `GiNaC::ex::evalf()`, `GiNaC::ex::subs()`, and `value`.

Referenced by `adaptivesimpson()`.

## 5.1.3.473 adaptivesimpson()

```
GiNaC::ex GiNaC::adaptivesimpson (
    const ex & x,
    const ex & a_in,
    const ex & b_in,
    const ex & f,
    const ex & error )
```

Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.

Parameters are integration variable, left boundary, right boundary, function to be integrated and the relative integration error. The function should evalf into a number after substituting the integration variable by a number. Another thing to note is that this implementation is no good at integrating functions with discontinuities.

References `abs()`, `GiNaC::ex::evalf()`, `GiNaC::integral::max_integration_level`, `GiNaC::ex::subs()`, `subsvalue()`, and `x`.

Referenced by `GiNaC::integral::evalf()`.

## 5.1.3.474 GINAC\_BIND\_UNARCHIVER() [21/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    integral )
```

**5.1.3.475 GINAC\_DECLARE\_UNARCHIVER()** [22/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    integral )
```

**5.1.3.476 ifactor()**

```
ex GiNaC::ifactor (
    const numeric & n )
```

Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $\text{lst}(\text{lst}(p_1, \dots, p_r), \text{lst}(a_1, \dots, a_r))$  such that  $n = p_1^{a_1} \dots$

$p_r^{a_r}$ .

References `GiNaC::container< C >::append()`, `irem()`,  $n$ , and `GiNaC::info_flags::prime`.

Referenced by `is_discriminant_of_quadratic_number_field()`, and `kronecker_symbol()`.

**5.1.3.477 is\_discriminant\_of\_quadratic\_number\_field()**

```
bool GiNaC::is_discriminant_of_quadratic_number_field (
    const numeric & n )
```

Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.

Returns false otherwise.

Ref.: Toshitsune Miyake, Modular Forms, Chapter 3.1

References `abs()`, `ifactor()`, `GiNaC::numeric::is_odd()`, `mod()`,  $n$ , `GiNaC::container< C >::nops()`, and `GiNaC::container< C >::op()`.

**5.1.3.478 kronecker\_symbol()**

```
numeric GiNaC::kronecker_symbol (
    const numeric & a,
    const numeric & n )
```

Returns the Kronecker symbol  $a: \text{integer } n: \text{integer}$ .

This routine defines `kronecker_symbol(1,0) = 1` `kronecker_symbol(-1,0) = 1` `kronecker_symbol(a,0) = 0`,  $a \neq 1, -1$

In particular `kronecker_symbol(-1,0) = 1` (in agreement with Sage)

Ref.: Toshitsune Miyake, Modular Forms, Chapter 3.1

References `ifactor()`, `GiNaC::numeric::is_even()`,  $n$ , `GiNaC::container< C >::op()`, and `pow()`.

Referenced by `primitive_dirichlet_character()`.

**5.1.3.479 primitive\_dirichlet\_character()**

```
numeric GiNaC::primitive_dirichlet_character (
    const numeric & n,
    const numeric & a )
```

Defines a primitive Dirichlet character through the Kronecker symbol.

n: integer a: discriminant of a quadratic field  $|a|$ : conductor

The character takes the values -1,0,1.

References kronecker\_symbol(), and n.

Referenced by dirichlet\_character(), and generalised\_Bernoulli\_number().

**5.1.3.480 dirichlet\_character()**

```
numeric GiNaC::dirichlet_character (
    const numeric & n,
    const numeric & a,
    const numeric & N )
```

Defines a Dirichlet character through the Kronecker symbol.

n: integer a: discriminant of a quadratic field  $|a|$ : conductor N: modulus, needs to be multiple of  $|a|$

The character takes the values -1,0,1.

References gcd(), n, and primitive\_dirichlet\_character().

**5.1.3.481 generalised\_Bernoulli\_number()**

```
numeric GiNaC::generalised_Bernoulli_number (
    const numeric & k,
    const numeric & b )
```

The generalised Bernoulli number.

k: index / modular weight

b: discriminant of a quadratic field, defines primitive character  $\psi$   $M=|b|$ : conductor of primitive character  $\psi$

The generalised Bernoulli number is computed from the series expansion of the generating function. The generating function is given in eq.(34), arXiv:1704.08895

References abs(), GiNaC::ex::coeff(), exp(), factorial(), k, primitive\_dirichlet\_character(), GiNaC::ex::series(), series\_to\_poly(), and x.

**5.1.3.482 Bernoulli\_polynomial()**

```
ex GiNaC::Bernoulli_polynomial (
    const numeric & k,
    const ex & x )
```

The Bernoulli polynomials.

References GiNaC::ex::coeff(), exp(), factorial(), k, GiNaC::ex::series(), series\_to\_poly(), and x.

Referenced by GiNaC::Eisenstein\_h\_kernel::coefficient\_a0().

**5.1.3.483 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [11/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    integration_kernel ,
    basic ,
    print_func< print_context > &::do_print )
```

**5.1.3.484 GINAC\_BIND\_UNARCHIVER()** [22/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    integration_kernel )
```

**5.1.3.485 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [12/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    basic_log_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.486 GINAC\_BIND\_UNARCHIVER()** [23/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    basic_log_kernel )
```



**5.1.3.487 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [13/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    multiple_polylog_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.488 GINAC\_BIND\_UNARCHIVER()** [24/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    multiple_polylog_kernel )
```

**5.1.3.489 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [14/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    ELi_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.490 GINAC\_BIND\_UNARCHIVER()** [25/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    ELi_kernel )
```

**5.1.3.491 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [15/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Ebar_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.492 GINAC\_BIND\_UNARCHIVER()** [26/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Ebar_kernel )
```

**5.1.3.493 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [16/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Kronecker_dtau_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.494 GINAC\_BIND\_UNARCHIVER()** [27/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Kronecker_dtau_kernel )
```

**5.1.3.495 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [17/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Kronecker_dz_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.496 GINAC\_BIND\_UNARCHIVER()** [28/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Kronecker_dz_kernel )
```

**5.1.3.497 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [18/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Eisenstein_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.498 GINAC\_BIND\_UNARCHIVER()** [29/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Eisenstein_kernel )
```

**5.1.3.499 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [19/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Eisenstein_h_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.500 GINAC\_BIND\_UNARCHIVER()** [30/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Eisenstein_h_kernel )
```

**5.1.3.501 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [20/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    modular_form_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.502 GINAC\_BIND\_UNARCHIVER()** [31/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    modular_form_kernel )
```

**5.1.3.503 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [21/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    user_defined_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.504 GINAC\_BIND\_UNARCHIVER()** [32/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    user_defined_kernel )
```

**5.1.3.505 GINAC\_DECLARE\_UNARCHIVER()** [23/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    integration_kernel )
```

**5.1.3.506 GINAC\_DECLARE\_UNARCHIVER()** [24/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    basic_log_kernel )
```

**5.1.3.507 GINAC\_DECLARE\_UNARCHIVER()** [25/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    multiple_polylog_kernel )
```

**5.1.3.508 GINAC\_DECLARE\_UNARCHIVER()** [26/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    ELi_kernel )
```

**5.1.3.509 GINAC\_DECLARE\_UNARCHIVER()** [27/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Ebar_kernel )
```

**5.1.3.510 GINAC\_DECLARE\_UNARCHIVER()** [28/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Kronecker_dtau_kernel )
```

**5.1.3.511 GINAC\_DECLARE\_UNARCHIVER()** [29/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Kronecker_dz_kernel )
```

**5.1.3.512 GINAC\_DECLARE\_UNARCHIVER()** [30/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Eisenstein_kernel )
```

**5.1.3.513 GINAC\_DECLARE\_UNARCHIVER()** [31/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Eisenstein_h_kernel )
```

**5.1.3.514 GINAC\_DECLARE\_UNARCHIVER()** [32/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    modular_form_kernel )
```

**5.1.3.515 GINAC\_DECLARE\_UNARCHIVER()** [33/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    user_defined_kernel )
```

**5.1.3.516 GINAC\_DECLARE\_UNARCHIVER()** [34/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    lst )
```

**5.1.3.517 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [22/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    matrix ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python_repr > &↵
::do_print_python_repr )
```

Default ctor.

Initializes to 1 x 1-dimensional zero-matrix.

References GiNaC::status\_flags::not\_shareable.

**5.1.3.518 GINAC\_BIND\_UNARCHIVER()** [33/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    matrix )
```

**5.1.3.519 lst\_to\_matrix()**

```
ex GiNaC::lst_to_matrix (
    const lst & l )
```

Convert list of lists to matrix.

References cols(), GiNaC::container< C >::nops(), and rows().

**5.1.3.520 diag\_matrix()** [1/2]

```
ex GiNaC::diag_matrix (
    const lst & l )
```

Convert list of diagonal elements to matrix.

References GiNaC::container< C >::nops().

**5.1.3.521 diag\_matrix()** [2/2]

```
ex GiNaC::diag_matrix (
    std::initializer_list< ex > l )
```

**5.1.3.522 unit\_matrix()** [1/2]

```
ex GiNaC::unit_matrix (
    unsigned r,
    unsigned c )
```

Create an r times c unit matrix.

References \_ex1, c, GiNaC::status\_flags::evaluated, r, and GiNaC::basic::setflag().

Referenced by unit\_matrix().

**5.1.3.523** `symbolic_matrix()` [1/2]

```
ex GiNaC::symbolic_matrix (
    unsigned r,
    unsigned c,
    const std::string & base_name,
    const std::string & tex_base_name )
```

Create an  $r$  times  $c$  matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.

The base name for LaTeX output is specified separately.

References `c`, `GiNaC::status_flags::evaluated`, `r`, and `GiNaC::basic::setflag()`.

Referenced by `symbolic_matrix()`.

**5.1.3.524** `reduced_matrix()`

```
ex GiNaC::reduced_matrix (
    const matrix & m,
    unsigned r,
    unsigned c )
```

Return the reduced matrix that is formed by deleting the  $r$ th row and  $c$ th column of matrix  $m$ .

The determinant of the result is the Minor  $r, c$ .

References `c`, `cols()`, `GiNaC::status_flags::evaluated`, `m`, `r`, `rows()`, and `GiNaC::basic::setflag()`.

**5.1.3.525** `sub_matrix()`

```
ex GiNaC::sub_matrix (
    const matrix & m,
    unsigned r,
    unsigned nr,
    unsigned c,
    unsigned nc )
```

Return the  $nr$  times  $nc$  submatrix starting at position  $r, c$  of matrix  $m$ .

References `c`, `GiNaC::status_flags::evaluated`, `m`, `r`, and `GiNaC::basic::setflag()`.

**5.1.3.526** `GINAC_DECLARE_UNARCHIVER()` [35/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    matrix )
```

**5.1.3.527 nops()** [2/2]

```
size_t GiNaC::nops (
    const matrix & m ) [inline]
```

References m.

**5.1.3.528 expand()** [2/2]

```
ex GiNaC::expand (
    const matrix & m,
    unsigned options = 0 ) [inline]
```

References m, and options.

**5.1.3.529 evalf()** [2/2]

```
ex GiNaC::evalf (
    const matrix & m ) [inline]
```

References m.

**5.1.3.530 rows()**

```
unsigned GiNaC::rows (
    const matrix & m ) [inline]
```

References m.

Referenced by `lst_to_matrix()`, and `reduced_matrix()`.

**5.1.3.531 cols()**

```
unsigned GiNaC::cols (
    const matrix & m ) [inline]
```

References m.

Referenced by `lst_to_matrix()`, and `reduced_matrix()`.



### 5.1.3.532 transpose()

```
matrix GiNaC::transpose (
    const matrix & m ) [inline]
```

References m.

Referenced by GiNaC::clifford::get\_metric().

### 5.1.3.533 determinant()

```
ex GiNaC::determinant (
    const matrix & m,
    unsigned options = determinant_algo::automatic ) [inline]
```

References m, and options.

### 5.1.3.534 trace()

```
ex GiNaC::trace (
    const matrix & m ) [inline]
```

References m.

### 5.1.3.535 charpoly()

```
ex GiNaC::charpoly (
    const matrix & m,
    const ex & lambda ) [inline]
```

References m.

### 5.1.3.536 inverse() [1/3]

```
matrix GiNaC::inverse (
    const matrix & m ) [inline]
```

References GiNaC::solve\_algo::automatic, and m.

Referenced by multiply\_lcm().

**5.1.3.537 inverse()** [2/3]

```

matrix GiNaC::inverse (
    const matrix & m,
    unsigned algo ) [inline]

```

References [m](#).

**5.1.3.538 rank()** [1/2]

```

unsigned GiNaC::rank (
    const matrix & m ) [inline]

```

References [m](#).

**5.1.3.539 rank()** [2/2]

```

unsigned GiNaC::rank (
    const matrix & m,
    unsigned solve_algo ) [inline]

```

References [m](#).

**5.1.3.540 unit\_matrix()** [2/2]

```

ex GiNaC::unit_matrix (
    unsigned x ) [inline]

```

Create a x times x unit matrix.

References [unit\\_matrix\(\)](#), and [x](#).

**5.1.3.541 symbolic\_matrix()** [2/2]

```

ex GiNaC::symbolic_matrix (
    unsigned r,
    unsigned c,
    const std::string & base_name ) [inline]

```

Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.

References [c](#), [r](#), and [symbolic\\_matrix\(\)](#).

**5.1.3.542 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [23/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    mul ,
    expairseq ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
print_tree. print_func< print_python_repr > &::do_print_python_repr )
```

**5.1.3.543 tryfactsubs()**

```
bool GiNaC::tryfactsubs (
    const ex & origfactor,
    const ex & patternfactor,
    int & nummatches,
    exmap & repls )
```

References `GiNaC::ex::info()`, `GiNaC::ex::match()`, `GiNaC::ex::op()`, and `to_int()`.

Referenced by `algebraic_match_mul_with_mul()`, `GiNaC::mul::algebraic_subs_mul()`, and `GiNaC::power::subs()`.

**5.1.3.544 algebraic\_match\_mul\_with\_mul()**

```
bool GiNaC::algebraic_match_mul_with_mul (
    const mul & e,
    const ex & pat,
    exmap & repls,
    int factor,
    int & nummatches,
    const std::vector< bool > & substed,
    std::vector< bool > & matched )
```

Checks whether `e` matches to the pattern `pat` and the (possibly to be updated) list of replacements `repls`.

This matching is in the sense of algebraic substitutions. Matching starts with `pat.op(factor)` of the pattern because the factors before this one have already been matched. The (possibly updated) number of matches is in `nummatches`. `substed[i]` is true for factors that already have been replaced by previous substitutions and `matched[i]` is true for factors that have been matched by the current match.

References `factor()`, `GINAC_ASSERT`, `GiNaC::expairseq::nops()`, `GiNaC::ex::nops()`, `GiNaC::expairseq::op()`, `GiNaC::ex::op()`, and `tryfactsubs()`.

Referenced by `GiNaC::mul::algebraic_subs_mul()`, and `GiNaC::mul::has()`.

**5.1.3.545 GINAC\_BIND\_UNARCHIVER()** [34/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    mul )
```

**5.1.3.546 GINAC\_DECLARE\_UNARCHIVER()** [36/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    mul )
```

**5.1.3.547 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [24/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    ncmul ,
    exprseq ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print↵
    _tree. print_func< print_csrc > &::do_print_csrc. print_func< print_python_repr > &::do_↵
    print_csrc )
```

**5.1.3.548 reeval\_ncmul()**

```
ex GiNaC::reeval_ncmul (
    const exvector & v )
```

**5.1.3.549 hold\_ncmul()**

```
ex GiNaC::hold_ncmul (
    const exvector & v )
```

References `_ex1`, and `GiNaC::status_flags::evaluated`.

Referenced by `GiNaC::color::eval_ncmul()`, `GiNaC::structure< T, ComparisonPolicy >::eval_ncmul()`, and `GiNaC↵::basic::eval_ncmul()`.

**5.1.3.550 GINAC\_BIND\_UNARCHIVER()** [35/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    ncmul )
```

**5.1.3.551 GINAC\_DECLARE\_UNARCHIVER()** [37/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    ncmul )
```

**5.1.3.552 get\_first\_symbol()**

```
static bool GiNaC::get_first_symbol (
    const ex & e,
    ex & x ) [static]
```

Return pointer to first symbol found in expression.

Due to [GiNaC](#)'s internal ordering of terms, it may not be obvious which symbol this function returns for a given expression.

## Parameters

<i>e</i>	expression to search
<i>x</i>	first symbol found (returned)

## Returns

"false" if no symbol was found, "true" otherwise

References `GiNaC::ex::nops()`, `GiNaC::ex::op()`, and `x`.

Referenced by `divide()`, `frac_cancel()`, and `GiNaC::ex::unit()`.

5.1.3.553 `add_symbol()`

```
static void GiNaC::add_symbol (
    const ex & s,
    sym_desc_vec & v ) [static]
```

Referenced by `collect_symbols()`.

5.1.3.554 `collect_symbols()`

```
static void GiNaC::collect_symbols (
    const ex & e,
    sym_desc_vec & v ) [static]
```

References `add_symbol()`, `GiNaC::ex::nops()`, and `GiNaC::ex::op()`.

Referenced by `get_symbol_stats()`.

5.1.3.555 `get_symbol_stats()`

```
static void GiNaC::get_symbol_stats (
    const ex & a,
    const ex & b,
    sym_desc_vec & v ) [static]
```

Collect statistical information about symbols in polynomials.

This function fills in a vector of "sym\_desc" structs which contain information about the highest and lowest degrees of all symbols that appear in two polynomials. The vector is then sorted by minimum degree (lowest to highest). The information gathered by this function is used by the GCD routines to identify trivial factors and to determine which variable to choose as the main variable for GCD computation.

## Parameters

<i>a</i>	first multivariate polynomial
<i>b</i>	second multivariate polynomial
<i>v</i>	vector of <a href="#">sym_desc</a> structs (filled in)

References [collect\\_symbols\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::ex::lcoeff\(\)](#), [GiNaC::ex::ldegree\(\)](#), and [GiNaC::ex::nops\(\)](#).

Referenced by [divide\\_in\\_z\(\)](#), [gcd\(\)](#), and [sqrfree\(\)](#).

5.1.3.556 [lcmcoeff\(\)](#)

```
static numeric GiNaC::lcmcoeff (
    const ex & e,
    const numeric & l ) [static]
```

References [\\_num1\\_p](#), [c](#), [GiNaC::ex::info\(\)](#), [lcm\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::rational](#).

Referenced by [heur\\_gcd\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), and [multiply\\_lcm\(\)](#).

5.1.3.557 [lcm\\_of\\_coefficients\\_denominators\(\)](#)

```
static numeric GiNaC::lcm_of_coefficients_denominators (
    const ex & e ) [static]
```

Compute LCM of denominators of coefficients of a polynomial.

Given a polynomial with rational coefficients, this function computes the LCM of the denominators of all coefficients. This can be used to bring a polynomial from  $\mathbb{Q}[X]$  to  $\mathbb{Z}[X]$ .

## Parameters

<i>e</i>	multivariate polynomial (need not be expanded)
----------	--

## Returns

LCM of denominators of coefficients

References [\\_num1\\_p](#), and [lcmcoeff\(\)](#).

Referenced by [frac\\_cancel\(\)](#), [heur\\_gcd\(\)](#), and [sqrfree\(\)](#).

## 5.1.3.558 multiply\_lcm()

```
static ex GiNaC::multiply_lcm (
    const ex & e,
    const numeric & lcm ) [static]
```

Bring polynomial from  $\mathbb{Q}[X]$  to  $\mathbb{Z}[X]$  by multiplying in the previously determined LCM of the coefficient's denominators.

## Parameters

<i>e</i>	multivariate polynomial (need not be expanded)
<i>lcm</i>	LCM to multiply in

References `_num1_p`, `inverse()`, `GiNaC::ex::is_equal()`, `GiNaC::numeric::is_rational()`, `lcm()`, `lcmcoeff()`, `GiNaC::ex::nops()`, `GiNaC::ex::op()`, and `pow()`.

Referenced by `frac_cancel()`, and `sqrfree()`.

## 5.1.3.559 quo()

```
ex GiNaC::quo (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args )
```

Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

It satisfies  $a(x)=b(x)*q(x)+r(x)$ .

## Parameters

<i>a</i>	first polynomial in x (dividend)
<i>b</i>	second polynomial in x (divisor)
<i>x</i>	a and b are polynomials in x
<i>check_args</i>	check whether a and b are polynomials with rational coefficients (defaults to "true")

## Returns

quotient of a and b in  $\mathbb{Q}[x]$

References `_ex1`, `GiNaC::ex::coeff()`, `GiNaC::ex::degree()`, `divide()`, `GiNaC::ex::expand()`, `expand()`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `pow()`, `r`, `GiNaC::info_flags::rational_polynomial`, and `x`.

Referenced by `decomp_rational()`, `GiNaC::ex::primpart()`, `sqrfree_parfrac()`, `sqrfree_yun()`, and `GiNaC::ex::unitcontprim()`.

5.1.3.560 `rem()`

```
ex GiNaC::rem (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args )
```

Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

It satisfies  $a(x)=b(x)*q(x)+r(x)$ .

## Parameters

$a$	first polynomial in $x$ (dividend)
$b$	second polynomial in $x$ (divisor)
$x$	$a$ and $b$ are polynomials in $x$
$check\_args$	check whether $a$ and $b$ are polynomials with rational coefficients (defaults to "true")

## Returns

remainder of  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$

References `_ex0`, `GiNaC::ex::coeff()`, `GiNaC::ex::degree()`, `divide()`, `GiNaC::ex::expand()`, `expand()`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `pow()`, `r`, `GiNaC::info_flags::rational_polynomial`, and `x`.

Referenced by `decomp_rational()`, `irem()`, and `sqrfree_parfrac()`.

5.1.3.561 `decomp_rational()`

```
ex GiNaC::decomp_rational (
    const ex & a,
    const ex & x )
```

Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .

## Parameters

$a$	rational function in $x$
$x$	$a$ is a function of $x$

## Returns

decomposed function.

References `denom()`, `numer()`, `numer_denom()`, `GiNaC::ex::op()`, `quo()`, `rem()`, and `x`.



## 5.1.3.562 prem()

```
ex GiNaC::prem (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args )
```

Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

## Parameters

<i>a</i>	first polynomial in $x$ (dividend)
<i>b</i>	second polynomial in $x$ (divisor)
<i>x</i>	$a$ and $b$ are polynomials in $x$
<i>check_args</i>	check whether $a$ and $b$ are polynomials with rational coefficients (defaults to "true")

## Returns

pseudo-remainder of  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$

References `_ex0`, `_ex1`, `GiNaC::ex::coeff()`, `GiNaC::ex::degree()`, `GiNaC::ex::expand()`, `expand()`, `GiNaC::ex::info()`, `GiNaC::ex::is_zero()`, `pow()`, `r`, `GiNaC::info_flags::rational_polynomial`, and `x`.

Referenced by `sr_gcd()`.

## 5.1.3.563 sprem()

```
ex GiNaC::sprem (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args )
```

Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

## Parameters

<i>a</i>	first polynomial in $x$ (dividend)
<i>b</i>	second polynomial in $x$ (divisor)
<i>x</i>	$a$ and $b$ are polynomials in $x$
<i>check_args</i>	check whether $a$ and $b$ are polynomials with rational coefficients (defaults to "true")

## Returns

sparse pseudo-remainder of  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$

References `_ex0`, `_ex1`, `GiNaC::ex::coeff()`, `GiNaC::ex::degree()`, `GiNaC::ex::expand()`, `expand()`, `GiNaC::ex::info()`, `GiNaC::ex::is_zero()`, `pow()`, `r`, `GiNaC::info_flags::rational_polynomial`, and `x`.

5.1.3.564 `divide()`

```
bool GiNaC::divide (
    const ex & a,
    const ex & b,
    ex & q,
    bool check_args )
```

Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Q[X]$ .

## Parameters

<i>a</i>	first multivariate polynomial (dividend)
<i>b</i>	second multivariate polynomial (divisor)
<i>q</i>	quotient (returned)
<i>check_args</i>	check whether a and b are polynomials with rational coefficients (defaults to "true")

## Returns

"true" when exact division succeeds (quotient returned in q), "false" otherwise (q left untouched)

References `_ex0`, `_ex1`, `GiNaC::ex::coeff()`, `GiNaC::ex::degree()`, `GiNaC::ex::expand()`, `expand()`, `get_first_↵symbol()`, `GiNaC::ex::info()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `GiNaC::ex::nops()`, `GiNaC::ex::op()`, `pow()`, `r`, `GiNaC::info_flags::rational_polynomial`, `to_int()`, and `x`.

Referenced by `find_common_factor()`, `GiNaC::matrix::fraction_free_elimination()`, `gcd()`, `quo()`, and `rem()`.

5.1.3.565 `divide_in_z()`

```
static bool GiNaC::divide_in_z (
    const ex & a,
    const ex & b,
    ex & q,
    sym_desc_vec::const_iterator var ) [static]
```

Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Z[X]$ .

This functions works like `divide()` but the input and output polynomials are in  $Z[X]$  instead of  $Q[X]$ . Unlike `divide()`, it doesn't check whether the input polynomials really are integer polynomials, so be careful of what you pass in. Also, you have to run `get_symbol_stats()` over the input polynomials before calling this function and pass an iterator to the first element of the `sym_desc` vector. This function is used internally by the `heur_gcd()`.

## Parameters

<i>a</i>	first multivariate polynomial (dividend)
<i>b</i>	second multivariate polynomial (divisor)
<i>q</i>	quotient (returned)
<i>var</i>	iterator to first element of vector of <code>sym_desc</code> structs

**Returns**

"true" when exact division succeeds (the quotient is returned in q), "false" otherwise.

**See also**

[get\\_symbol\\_stats](#), [heur\\_gcd](#)

References [\\_ex0](#), [\\_ex1](#), [\\_num0\\_p](#), [\\_num1\\_p](#), [GiNaC::ex::begin\(\)](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [GiNaC::ex::find\(\)](#), [get\\_symbol\\_stats\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::inverse\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [k](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), [qbar](#), [r](#), [GiNaC::ex::subs\(\)](#), [to\\_int\(\)](#), and [x](#).

Referenced by [heur\\_gcd\\_z\(\)](#), and [sr\\_gcd\(\)](#).

**5.1.3.566 sr\_gcd()**

```
static ex GiNaC::sr_gcd (
    const ex & a,
    const ex & b,
    sym_desc_vec::const_iterator var ) [static]
```

Compute GCD of multivariate polynomials using the subresultant PRS algorithm.

This function is used internally by [gcd\(\)](#).

**Parameters**

<i>a</i>	first multivariate polynomial
<i>b</i>	second multivariate polynomial
<i>var</i>	iterator to first element of vector of <a href="#">sym_desc</a> structs

**Returns**

the GCD as a new expression

**See also**

[gcd](#)

References [\\_ex0](#), [\\_ex1](#), [c](#), [GiNaC::ex::content\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\\_in\\_z\(\)](#), [gcd\(\)](#), [pow\(\)](#), [prem\(\)](#), [GiNaC::ex::primpart\(\)](#), [psi\(\)](#), [r](#), and [x](#).

Referenced by [gcd\(\)](#).

5.1.3.567 `interpolate()`

```
static ex GiNaC::interpolate (
    const ex & gamma,
    const numeric & xi,
    const ex & x,
    int degree_hint = 1 ) [static]
```

xi-adic polynomial interpolation

References `GiNaC::numeric::inverse()`, `GiNaC::ex::is_zero()`, `pow()`, `GiNaC::ex::smod()`, and `x`.

Referenced by `heur_gcd_z()`.

5.1.3.568 `heur_gcd_z()`

```
static bool GiNaC::heur_gcd_z (
    ex & res,
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb,
    sym_desc_vec::const_iterator var ) [static]
```

Compute GCD of multivariate polynomials using the heuristic GCD algorithm.

`get_symbol_stats()` must have been called previously with the input polynomials and an iterator to the first element of the `sym_desc` vector passed in. This function is used internally by `gcd()`.

## Parameters

<i>a</i>	first integer multivariate polynomial (expanded)
<i>b</i>	second integer multivariate polynomial (expanded)
<i>ca</i>	cofactor of polynomial a (returned), nullptr to suppress calculation of cofactor
<i>cb</i>	cofactor of polynomial b (returned), nullptr to suppress calculation of cofactor
<i>var</i>	iterator to first element of vector of <code>sym_desc</code> structs
<i>res</i>	the GCD (returned)

## Returns

true if GCD was computed, false otherwise.

## See also

[gcd](#)

## Exceptions

<code>gcdheu_failed()</code>	
------------------------------	--

References `GiNaC::ex::degree()`, `divide_in_z()`, `GiNaC::ex::expand()`, `gcd()`, `GiNaC::numeric::int_length()`, `GiNaC::ex::integer_content()`, `interpolate()`, `GiNaC::numeric::inverse()`, `iquo()`, `GiNaC::ex::is_zero()`, `isqrt()`, `GiNaC::ex::max_coefficient()`, `GiNaC::subs_options::no_pattern`, `GiNaC::ex::subs()`, and `x`.

Referenced by `heur_gcd()`.

### 5.1.3.569 `heur_gcd()`

```
static bool GiNaC::heur_gcd (
    ex & res,
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb,
    sym_desc_vec::const_iterator var ) [static]
```

Compute GCD of multivariate polynomials using the heuristic GCD algorithm.

`get_symbol_stats()` must have been called previously with the input polynomials and an iterator to the first element of the `sym_desc` vector passed in. This function is used internally by `gcd()`.

#### Parameters

<i>a</i>	first rational multivariate polynomial (expanded)
<i>b</i>	second rational multivariate polynomial (expanded)
<i>ca</i>	cofactor of polynomial a (returned), nullptr to suppress calculation of cofactor
<i>cb</i>	cofactor of polynomial b (returned), nullptr to suppress calculation of cofactor
<i>var</i>	iterator to first element of vector of <code>sym_desc</code> structs
<i>res</i>	the GCD (returned)

#### Returns

true if GCD was computed, false otherwise.

#### See also

[heur\\_gcd\\_z](#)  
[gcd](#)

References `heur_gcd_z()`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer_polynomial`, `lcm_of_coefficients_denominators()`, and `lcmcoeff()`.

Referenced by `gcd()`.

## 5.1.3.570 gcd\_pf\_pow()

```
static ex GiNaC::gcd_pf_pow (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb ) [static]
```

References `_ex1`, `expand()`, `gcd()`, `gcd_pf_pow_pow()`, `GINAC_ASSERT`, `GiNaC::ex::is_equal()`, `GiNaC::ex::ldegree()`, `GiNaC::ex::op()`, `pow()`, and `to_int()`.

Referenced by `gcd()`.

## 5.1.3.571 gcd\_pf\_mul()

```
static ex GiNaC::gcd_pf_mul (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb ) [static]
```

References `gcd()`, `GINAC_ASSERT`, `GiNaC::ex::nops()`, and `GiNaC::ex::op()`.

Referenced by `gcd()`.

## 5.1.3.572 gcd() [1/2]

```
ex GiNaC::gcd (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb,
    bool check_args,
    unsigned options )
```

Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $\mathbb{Z}[X]$ .

Optionally also compute the cofactors of  $a$  and  $b$ , defined by  $a = ca * \text{gcd}(a, b)$  and  $b = cb * \text{gcd}(a, b)$ .

## Parameters

<i>a</i>	first multivariate polynomial
<i>b</i>	second multivariate polynomial
<i>ca</i>	pointer to expression that will receive the cofactor of $a$ , or nullptr
<i>cb</i>	pointer to expression that will receive the cofactor of $b$ , or nullptr
<i>check_args</i>	check whether $a$ and $b$ are polynomials with rational coefficients (defaults to "true")

**Returns**

the GCD as a new expression

References `_ex0`, `_ex1`, `GiNaC::ex::begin()`, `divide()`, `GiNaC::ex::expand()`, `expand()`, `gcd_pf_mul()`, `gcd_pf_pow()`, `get_symbol_stats()`, `heur_gcd()`, `GiNaC::ex::info()`, `GiNaC::ex::integer_content()`, `GiNaC::ex::is_equal()`, `GiNaC::numeric::is_zero()`, `GiNaC::ex::is_zero()`, `n`, `GiNaC::gcd_options::no_heur_gcd`, `GiNaC::gcd_options::no_part_factored`, `GiNaC::subs_options::no_pattern`, `options`, `pow()`, `GiNaC::info_flags::rational_polynomial`, `sr_gcd()`, `GiNaC::ex::subs()`, `GiNaC::ex::unitcontprim()`, `GiNaC::gcd_options::use_sr_gcd`, and `x`.

Referenced by `GiNaC::ex::content()`, `dirichlet_character()`, `find_common_factor()`, `frac_cancel()`, `gcd_pf_mul()`, `gcd_pf_pow()`, `gcd_pf_pow_pow()`, `heur_gcd_z()`, `GiNaC::add::integer_content()`, `lcm()`, `GiNaC::add::normal()`, `sqrfree_yun()`, and `sr_gcd()`.

**5.1.3.573 gcd\_pf\_pow\_pow()**

```
static ex GiNaC::gcd_pf_pow_pow (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb ) [static]
```

References `_ex1`, `gcd()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::op()`, and `pow()`.

Referenced by `gcd_pf_pow()`.

**5.1.3.574 lcm()** [1/2]

```
ex GiNaC::lcm (
    const ex & a,
    const ex & b,
    bool check_args )
```

Compute LCM (Least Common Multiple) of multivariate polynomials in  $\mathbb{Z}[X]$ .

**Parameters**

<i>a</i>	first multivariate polynomial
<i>b</i>	second multivariate polynomial
<i>check_args</i>	check whether a and b are polynomials with rational coefficients (defaults to "true")

**Returns**

the LCM as a new expression

References `gcd()`, `GiNaC::ex::info()`, and `GiNaC::info_flags::rational_polynomial`.

Referenced by `GiNaC::add::integer_content()`, `lcmcoeff()`, `multiply_lcm()`, and `sqrfree()`.

5.1.3.575 `sqrfree_yun()`

```
static epvector GiNaC::sqrfree_yun (
    const ex & a,
    const symbol & x ) [static]
```

Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.

Used internally by `sqrfree()`.

## Parameters

$a$	multivariate polynomial over $\mathbb{Z}[X]$ , treated here as univariate polynomial in $x$ (needs not be expanded).
$x$	variable to factor in

## Returns

vector of expairs (factor, exponent), sorted by exponents

References `_ex1`, `GiNaC::ex::diff()`, `factors`, `gcd()`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `quo()`, and `x`.

Referenced by `sqrfree()`, and `sqrfree_parfrac()`.

5.1.3.576 `sqrfree()`

```
ex GiNaC::sqrfree (
    const ex & a,
    const lst & l )
```

Compute a square-free factorization of a multivariate polynomial in  $\mathbb{Q}[X]$ .

## Parameters

$a$	multivariate polynomial over $\mathbb{Q}[X]$ (needs not be expanded)
$l$	lst of variables to factor in, may be left empty for autodetection

## Returns

a square-free factorization of  $a$ .

## Note

A polynomial  $p(X) \in C[X]$  is said *square-free* if, whenever any two polynomials  $q(X)$  and  $r(X)$  are such that

$$p(X) = q(X)^2 r(X),$$

we have  $q(X) \in C$ . This means that  $p(X)$  has no repeated factors, apart eventually from constants. Given a polynomial  $p(X) \in C[X]$ , we say that the decomposition

$$p(X) = b \cdot p_1(X)^{a_1} \cdot p_2(X)^{a_2} \cdots p_r(X)^{a_r}$$

is a *square-free factorization* of  $p(X)$  if the following conditions hold:



1.  $b \in C$  and  $b \neq 0$ ;
2.  $a_i$  is a positive integer for  $i = 1, \dots, r$ ;
3. the degree of the polynomial  $p_i$  is strictly positive for  $i = 1, \dots, r$ ;
4. the polynomial  $\prod_{i=1}^r p_i(X)$  is square-free.

Square-free factorizations need not be unique. For example, if  $a_i$  is even, we could change the polynomial  $p_i(X)$  into  $-p_i(X)$ . Observe also that the factors  $p_i(X)$  need not be irreducible polynomials.

References `_ex0`, `GiNaC::container< C >::append()`, `factors`, `get_symbol_stats()`, `lcm()`, `lcm_of_coefficients_`  $\leftrightarrow$  `denominators()`, `multiply_lcm()`, `GiNaC::container< C >::nops()`, `GiNaC::container< C >::op()`, `GiNaC::container< C >::remove_first()`, `sqrfree_yun()`, and `x`.

#### 5.1.3.577 `sqrfree_parfrac()`

```
ex GiNaC::sqrfree_parfrac (
    const ex & a,
    const symbol & x )
```

Compute square-free partial fraction decomposition of rational function  $a(x)$ .

##### Parameters

$a$	rational function over $\mathbb{Z}[x]$ , treated as univariate polynomial in $x$
$x$	variable to factor in

##### Returns

decomposed rational function

References `_ex1`, `GiNaC::ex::coeff()`, `GiNaC::basic::coeff()`, `coeff()`, `GiNaC::ex::degree()`, `denom()`, `GiNaC::ex`  $\leftrightarrow$  `::expand()`, `factor()`, `k`, `numer()`, `numer_denom()`, `GiNaC::ex::op()`, `pow()`, `quo()`, `rem()`, `rhs()`, `GiNaC::matrix::solve()`, `sqrfree_yun()`, `to_int()`, and `x`.

#### 5.1.3.578 `replace_with_symbol()` [1/2]

```
static ex GiNaC::replace_with_symbol (
    const ex & e,
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) [static]
```

Create a symbol for replacing the expression "e" (or return a previously assigned symbol).

The symbol and expression are appended to `repl`, for a later application of `subs()`. An entry in the replacement table `repl` can be changed in some cases. If it was altered, we need to provide the modifier for the previously build expressions. The modifier is an (ordered) list, because those substitutions need to be done in the incremental order. As

an example let us consider a rationalisation of the expression  $e = \exp(2x) \cdot \cos(\exp(2x)+1) \cdot \exp(x)$ . The first factor [GiNaC](#) denotes by something like `symbol1` and will record: `e = symbol1*cos(symbol1 + 1)*exp(x)` `repl = {symbol1 : exp(2*x)}`. Similarly, the second factor would be denoted as `symbol2` and we will have `e = symbol1*symbol2*exp(x)` `repl = {symbol1 : exp(2*x), symbol2 : cos(symbol1 + 1)}`. Denoting the third term as `symbol3` [GiNaC](#) is willing to re-think `exp(2*x)` as `symbol3^2` rather than just `symbol1`. Here are two issues: 1) The replacement "`symbol1 -> symbol3^2`" in the previous part of the expression needs to be done outside of the present routine; 2) The pair "`symbol1 : exp(2*x)`" shall be deleted from the replacement table `repl`. However, this will create illegal substitution "`symbol2 : cos(symbol1 + 1)`" with undefined `symbol1`. These both problems are mitigated through the additions of the record "`symbol1==symbol3^2`" to the list modifier. Changed length of the modifier signals to the calling code that the previous portion of the expression needs to be altered (it solves 1). Thus [GiNaC](#) can record now `e = symbol3^2*symbol2*symbol3` `repl = {symbol2 : cos(symbol1 + 1), symbol3 : exp(x)}` `modifier = {symbol1==symbol3^2}`. Then, doing the backward substitutions the list modifier will be used to restore such iterative substitutions in the right way (this solves 2).

See also

[ex::normal](#)

References `_ex_1`, `GiNaC::container< C >::append()`, `degree()`, `denom()`, `exp()`, `GiNaC::ex::find()`, `GiNaC::ex::is_←  
_equal()`, `is_ex_the_function`, `is_integer()`, `is_rational()`, `GiNaC::subs_options::no_pattern`, `normal()`, `numer()`, `Gi←  
NaC::ex::op()`, `pow()`, and `GiNaC::ex::subs()`.

Referenced by `GiNaC::pseries::normal()`, `GiNaC::power::normal()`, `GiNaC::numeric::normal()`, `GiNaC::basic←  
::normal()`, `GiNaC::power::to_polynomial()`, `GiNaC::numeric::to_polynomial()`, `GiNaC::basic::to_polynomial()`, `Gi←  
NaC::power::to_rational()`, `GiNaC::numeric::to_rational()`, and `GiNaC::basic::to_rational()`.

### 5.1.3.579 `replace_with_symbol()` [2/2]

```
static ex GiNaC::replace_with_symbol (
    const ex & e,
    exmap & repl ) [static]
```

Create a symbol for replacing the expression "e" (or return a previously assigned symbol).

The symbol and expression are appended to `repl`, and the symbol is returned.

See also

[basic::to\\_rational](#)  
[basic::to\\_polynomial](#)

References `GiNaC::subs_options::no_pattern`, and `GiNaC::ex::subs()`.

### 5.1.3.580 `frac_cancel()`

```
static ex GiNaC::frac_cancel (
    const ex & n,
    const ex & d ) [static]
```

Fraction cancellation.

## Parameters

$n$	numerator
$d$	denominator

## Returns

cancelled fraction {n, d} as a list

References `_ex1`, `_ex_1`, `_num1_p`, `GiNaC::numeric::denom()`, `GiNaC::ex::expand()`, `gcd()`, `get_first_symbol()`, `GINAC_ASSERT`, `GiNaC::ex::is_equal()`, `is_negative()`, `GiNaC::ex::is_zero()`, `lcm_of_coefficients_denominators()`, `multiply_lcm()`, `n`, `GiNaC::numeric::number()`, `GiNaC::ex::unit()`, and `x`.

Referenced by `GiNaC::add::normal()`, and `GiNaC::mul::normal()`.

5.1.3.581 `find_common_factor()`

```
static ex GiNaC::find_common_factor (
    const ex & e,
    ex & factor,
    exmap & repl ) [static]
```

Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).

References `_ex0`, `_ex1`, `divide()`, `factor()`, `gcd()`, `GiNaC::info_flags::integer`, `GiNaC::ex::is_equal()`, `GiNaC::ex::is_zero()`, `k`, `GiNaC::ex::nops()`, `GiNaC::ex::op()`, `pow()`, `GiNaC::ex::to_polynomial()`, and `x`.

Referenced by `collect_common_factors()`.

5.1.3.582 `collect_common_factors()`

```
ex GiNaC::collect_common_factors (
    const ex & e )
```

Collect common factors in sums.

This converts expressions like 'a\*(b\*x+b\*y)' to 'a\*b\*(x+y)'.

References `factor()`, `find_common_factor()`, `GiNaC::subs_options::no_pattern`, `r`, and `GiNaC::ex::subs()`.

Referenced by `GiNaC::power::to_polynomial()`.

**5.1.3.583 resultant()**

```
ex GiNaC::resultant (
    const ex & e1,
    const ex & e2,
    const ex & s )
```

Resultant of two expressions e1,e2 with respect to symbol s.

Method: Compute determinant of Sylvester matrix of e1,e2,s.

References `GiNaC::ex::coeff()`, `GiNaC::ex::degree()`, `GiNaC::ex::expand()`, `GiNaC::ex::info()`, `k`, `GiNaC::ex::ldegree()`, `m`, and `GiNaC::info_flags::polynomial`.

**5.1.3.584 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [25/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    numeric ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_csrc_cl_N > &↵
::do_print_csrc_cl_N. print_func< print_tree > &::do_print_tree. print_func< print_python_repr
> &::do_print_python_repr )
```

default ctor.

Numerically it initializes to an integer zero.

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `GiNaC::basic::setflag()`, and `GiNaC::numeric::value`.

**5.1.3.585 make\_real\_float()**

```
static const cln::cl_F GiNaC::make_real_float (
    const cln::cl_idcoded_float & dec ) [static]
```

Construct a floating point number from sign, mantissa, and exponent.

References `x`.

Referenced by `read_real_float()`.

**5.1.3.586 read\_real\_float()**

```
static const cln::cl_F GiNaC::read_real_float (
    std::istream & s ) [static]
```

Read serialized floating point number.

References `make_real_float()`, and `x`.

Referenced by `GiNaC::numeric::read_archive()`.

**5.1.3.587 GINAC\_BIND\_UNARCHIVER()** [36/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    numeric )
```

**5.1.3.588 write\_real\_float()**

```
static void GiNaC::write_real_float (
    std::ostream & s,
    const cln::cl_R & n ) [static]
```

References `n`.

Referenced by `GiNaC::numeric::archive()`.

**5.1.3.589 print\_real\_number()**

```
static void GiNaC::print_real_number (
    const print_context & c,
    const cln::cl_R & x ) [static]
```

Helper function to print a real number in a nicer way than is CLN's default.

Instead of printing 42.0L0 this just prints 42.0 to ostream `os` and instead of 3.99168L7 it prints 3.99168E7. This is fine in [GiNaC](#) as long as it only uses `cl_LF` and no other floating point types that we might want to visibly distinguish from `cl_LF`.

See also

[numeric::print\(\)](#)

References `abs()`, `c`, and `x`.

Referenced by `GiNaC::numeric::print_numeric()`, and `print_real_cl_N()`.

#### 5.1.3.590 `print_integer_csrc()`

```
static void GiNaC::print_integer_csrc (
    const print\_context & c,
    const cln::cl\_I & x ) [static]
```

Helper function to print integer number in C++ source format.

See also

[numeric::print\(\)](#)

References `c`, and `x`.

Referenced by `print_real_csrc()`.

#### 5.1.3.591 `print_real_csrc()`

```
static void GiNaC::print_real_csrc (
    const print\_context & c,
    const cln::cl\_R & x ) [static]
```

Helper function to print real number in C++ source format.

See also

[numeric::print\(\)](#)

References `c`, `denom()`, `numer()`, `print_integer_csrc()`, and `x`.

Referenced by `GiNaC::numeric::do_print_csrc()`.

#### 5.1.3.592 `coerce()`

```
template<typename T1 , typename T2 >
static bool GiNaC::coerce (
    T1 & dst,
    const T2 & arg ) [inline], [static]
```

Referenced by `print_real_cl_N()`.

**5.1.3.593** `coerce< int, cln::cl_I >()`

```
template<>
bool GiNaC::coerce< int, cln::cl_I > (
    int & dst,
    const cln::cl_I & arg ) [inline]
```

Check if CLN integer can be converted into int.

See also

<https://www.ginac.de/pipermail/cln-list/2006-October/000248.html>

**5.1.3.594** `coerce< unsigned int, cln::cl_I >()`

```
template<>
bool GiNaC::coerce< unsigned int, cln::cl_I > (
    unsigned int & dst,
    const cln::cl_I & arg ) [inline]
```

**5.1.3.595** `print_real_cl_N()`

```
static void GiNaC::print_real_cl_N (
    const print_context & c,
    const cln::cl_R & x ) [static]
```

Helper function to print real number in C++ source format using cl\_N types.

See also

`numeric::print()`

References `c`, `coerce()`, `Digits`, `print_real_number()`, and `x`.

Referenced by `GiNaC::numeric::do_print_csrc_cl_N()`.

**5.1.3.596** `exp()`

```
const numeric GiNaC::exp (
    const numeric & x )
```

Exponential function.

Returns

arbitrary precision numerical `exp(x)`.

References `x`.

Referenced by `abs_eval()`, `abs_power()`, `Bernoulli_polynomial()`, `beta_eval()`, `GiNaC::Eisenstein_h_kernel::coefficient_an()`, `csgn_power()`, `GiNaC::power::do_print_csrc()`, `exp_conjugate()`, `exp_deriv()`, `exp_eval()`, `exp_evalf()`, `exp_expand()`, `exp_imag_part()`, `exp_power()`, `exp_real_part()`, `generalised_Bernoulli_number()`, `GiNaC::Kronecker_dtau_kernel::get_numerical_value()`, `GiNaC::power::imag_part()`, `log_eval()`, `print_sym_pow()`, `GiNaC::power::real_part()`, `replace_with_symbol()`, `GiNaC::power::series()`, `GiNaC::Kronecker_dtau_kernel::series_coeff_impl()`, `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`, and `tgamma()`.

5.1.3.597 `log()`

```
const numeric GiNaC::log (
    const numeric & x )
```

Natural logarithm.

## Parameters

<code>x</code>	complex number
----------------	----------------

## Returns

arbitrary precision numerical  $\log(x)$ .

## Exceptions

<code>pole_error("log()"</code>	<code>logarithmic pole",0)</code>
---------------------------------	-----------------------------------

References `GiNaC::ex::is_zero()`, and `x`.

Referenced by `atan()`, `atan_series()`, `atanh_series()`, `GiNaC::power::derivative()`, `GiNaC::integral::eval_integ()`, `exp_eval()`, `G2_eval()`, `G2_evalf()`, `G3_eval()`, `G3_evalf()`, `H_eval()`, `GiNaC::power::imag_part()`, `lgamma()`, `lgamma_eval()`, `lgamma_series()`, `Li2_()`, `Li2_deriv()`, `Li2_eval()`, `Li2_projection()`, `Li2_series()`, `Li_eval()`, `log_conjugate()`, `log_eval()`, `log_evalf()`, `log_expand()`, `log_real_part()`, `log_series()`, `psi1_eval()`, `psi2_eval()`, `GiNaC::power::real_part()`, `S_eval()`, and `GiNaC::power::series()`.

5.1.3.598 `sin()`

```
const numeric GiNaC::sin (
    const numeric & x )
```

Numeric sine (trigonometric function).

## Returns

arbitrary precision numerical  $\sin(x)$ .

References `x`.

Referenced by `GiNaC::Eisenstein_h_kernel::coefficient_a0()`, `cos_deriv()`, `cos_imag_part()`, `cosh_imag_part()`, `exp_imag_part()`, `GiNaC::power::imag_part()`, `lgamma()`, `sin_conjugate()`, `sin_eval()`, `sin_evalf()`, `sin_real_part()`, `sinh_eval()`, `sinh_imag_part()`, `tan_series()`, and `tgamma()`.



**5.1.3.599 cos()**

```
const numeric GiNaC::cos (
    const numeric & x )
```

Numeric cosine (trigonometric function).

**Returns**

arbitrary precision numerical  $\cos(x)$ .

References  $x$ .

Referenced by `GiNaC::Eisenstein_h_kernel::coefficient_a0()`, `cos_conjugate()`, `cos_eval()`, `cos_evalf()`, `cos_real_part()`, `cosh_eval()`, `cosh_real_part()`, `exp_real_part()`, `GiNaC::power::real_part()`, `sin_deriv()`, `sin_imag_part()`, `sinh_real_part()`, and `tan_series()`.

**5.1.3.600 tan()**

```
const numeric GiNaC::tan (
    const numeric & x )
```

Numeric tangent (trigonometric function).

**Returns**

arbitrary precision numerical  $\tan(x)$ .

References  $x$ .

Referenced by `tan_conjugate()`, `tan_deriv()`, `tan_eval()`, `tan_evalf()`, `tan_imag_part()`, `tan_real_part()`, `tanh_eval()`, `tanh_imag_part()`, and `tanh_real_part()`.

**5.1.3.601 asin()**

```
const numeric GiNaC::asin (
    const numeric & x )
```

Numeric inverse sine (trigonometric function).

**Returns**

arbitrary precision numerical  $\operatorname{asin}(x)$ .

References  $x$ .

Referenced by `asin_conjugate()`, `asin_eval()`, `asin_evalf()`, `cos_eval()`, `sin_eval()`, and `tan_eval()`.

**5.1.3.602** `acos()`

```
const numeric GiNaC::acos (
    const numeric & x )
```

Numeric inverse cosine (trigonometric function).

**Returns**

arbitrary precision numerical `acos(x)`.

References `x`.

Referenced by `acos_conjugate()`, `acos_eval()`, `acos_evalf()`, `cos_eval()`, `sin_eval()`, and `tan_eval()`.

**5.1.3.603** `atan()` [1/2]

```
const numeric GiNaC::atan (
    const numeric & x )
```

Numeric arcustangent.

**Parameters**

<code>x</code>	complex number
----------------	----------------

**Returns**

`atan(x)`

**Exceptions**

<code>pole_error("atan()"</code>	<code>logarithmic pole",0)</code> if <code>x==I</code> or <code>x==I</code> .
----------------------------------	---

References `_num1_p`, `abs()`, `atan()`, `GiNaC::ex::is_zero()`, and `x`.

Referenced by `atan2_eval()`, `atan2_evalf()`, `atan_conjugate()`, `atan_eval()`, `atan_evalf()`, `atan_series()`, `cos_eval()`, `sin_eval()`, and `tan_eval()`.

**5.1.3.604** `atan()` [2/2]

```
const numeric GiNaC::atan (
    const numeric & y,
    const numeric & x )
```

Numeric arcustangent of two arguments, analytically continued in a suitable way.

## Parameters

<i>y</i>	complex number
<i>x</i>	complex number

## Returns

$-i \log((x + i y) / \sqrt{x^2 + y^2})$ , which is equal to  $\operatorname{atan}(y/x)$  if  $y$  and  $x$  are both real.

## Exceptions

<i>pole_error("atan()"</i>	<i>logarithmic pole",0)</i> if $y/x == +i$ or $y/x == -i$ .
----------------------------	---

References `_num0_p`, `GiNaC::numeric::is_real()`, `GiNaC::numeric::is_zero()`, `GiNaC::ex::is_zero()`, `log()`, `sqrt()`, `GiNaC::numeric::to_cl_N()`, and `x`.

Referenced by `atan()`.

5.1.3.605 `sinh()`

```
const numeric GiNaC::sinh (
    const numeric & x )
```

Numeric hyperbolic sine (trigonometric function).

## Returns

arbitrary precision numerical  $\sinh(x)$ .

References `x`.

Referenced by `cos_imag_part()`, `cosh_deriv()`, `cosh_imag_part()`, `sin_imag_part()`, `sinh_conjugate()`, `sinh_eval()`, `sinh_evalf()`, `sinh_real_part()`, and `tanh_series()`.

5.1.3.606 `cosh()`

```
const numeric GiNaC::cosh (
    const numeric & x )
```

Numeric hyperbolic cosine (trigonometric function).

## Returns

arbitrary precision numerical  $\cosh(x)$ .

References `x`.

Referenced by `cos_real_part()`, `cosh_conjugate()`, `cosh_eval()`, `cosh_evalf()`, `cosh_real_part()`, `sin_real_part()`, `sinh_deriv()`, `sinh_imag_part()`, and `tanh_series()`.

### 5.1.3.607 tanh()

```
const numeric GiNaC::tanh (
    const numeric & x )
```

Numeric hyperbolic tangent (trigonometric function).

#### Returns

arbitrary precision numerical tanh(x).

References x.

Referenced by tan\_imag\_part(), tanh\_conjugate(), tanh\_deriv(), tanh\_eval(), tanh\_evalf(), tanh\_imag\_part(), and tanh\_real\_part().

### 5.1.3.608 asinh()

```
const numeric GiNaC::asinh (
    const numeric & x )
```

Numeric inverse hyperbolic sine (trigonometric function).

#### Returns

arbitrary precision numerical asinh(x).

References x.

Referenced by asinh\_conjugate(), asinh\_eval(), asinh\_evalf(), cosh\_eval(), sinh\_eval(), and tanh\_eval().

### 5.1.3.609 acosh()

```
const numeric GiNaC::acosh (
    const numeric & x )
```

Numeric inverse hyperbolic cosine (trigonometric function).

#### Returns

arbitrary precision numerical acosh(x).

References x.

Referenced by acosh\_conjugate(), acosh\_eval(), acosh\_evalf(), cosh\_eval(), sinh\_eval(), and tanh\_eval().

### 5.1.3.610 atanh()

```
const numeric GiNaC::atanh (
    const numeric & x )
```

Numeric inverse hyperbolic tangent (trigonometric function).

#### Returns

arbitrary precision numerical atanh(x).

References x.

Referenced by atanh\_conjugate(), atanh\_eval(), atanh\_evalf(), atanh\_series(), cosh\_eval(), sinh\_eval(), and tanh\_eval().

### 5.1.3.611 Li2\_series() [2/2]

```
static cln::cl_N GiNaC::Li2_series (
    const cln::cl_N & x,
    const cln::float_format_t & prec ) [static]
```

Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.

References x.

### 5.1.3.612 Li2\_projection()

```
static cln::cl_N GiNaC::Li2_projection (
    const cln::cl_N & x,
    const cln::float_format_t & prec ) [static]
```

Folds Li2's argument inside a small rectangle to enhance convergence.

References abs(), Li2\_series(), log(), x, and zeta().

Referenced by Li2\_().

**5.1.3.613 Li2\_()**

```
const cln::cl_N GiNaC::Li2_ (
    const cln::cl_N & value )
```

Numeric evaluation of Dilogarithm.

The domain is the entire complex plane, the branch cut lies along the positive real axis, starting at 1 and continuous with quadrant IV.

**Returns**

arbitrary precision numerical Li2(x).

References `abs()`, `Li2_projection()`, `log()`, `value`, and `zeta()`.

Referenced by `Li2()`.

**5.1.3.614 Li2()**

```
const numeric GiNaC::Li2 (
    const numeric & x )
```

References `_num0_p`, `Li2_()`, and `x`.

Referenced by `Li2_conjugate()`, `Li2_eval()`, `Li2_evalf()`, and `Li2_series()`.

**5.1.3.615 zeta()** [3/3]

```
const numeric GiNaC::zeta (
    const numeric & x )
```

Numeric evaluation of Riemann's Zeta function.

Currently works only for integer arguments.

References `x`.

Referenced by `Li2_()`, and `Li2_projection()`.

**5.1.3.616 guess\_precision()**

```
static cln::float_format_t GiNaC::guess_precision (
    const cln::cl_N & x ) [static]
```

References `x`.

Referenced by `lgamma()`, and `tgamma()`.

**5.1.3.617 lgamma()** [1/2]

```
const cln::cl_N GiNaC::lgamma (
    const cln::cl_N & x )
```

The Gamma function.

Use the Lanczos approximation. If the coefficients used here are not sufficiently many or sufficiently accurate, more can be calculated using the program `doc/examples/lanczos.cpp`. In that case, be sure to read the comments in that file.

References `GiNaC::lanczos_coeffs::calc_lanczos_A()`, `GiNaC::lanczos_coeffs::get_order()`, `guess_precision()`, `log()`, `sin()`, `GiNaC::lanczos_coeffs::sufficiently_accurate()`, and `x`.

Referenced by `beta_evalf()`, `lgamma()`, `lgamma_conjugate()`, `lgamma_eval()`, `lgamma_evalf()`, and `lgamma_series()`.

**5.1.3.618 lgamma()** [2/2]

```
const numeric GiNaC::lgamma (
    const numeric & x )
```

References `lgamma()`, and `x`.

**5.1.3.619 tgamma()** [1/2]

```
const cln::cl_N GiNaC::tgamma (
    const cln::cl_N & x )
```

References `GiNaC::lanczos_coeffs::calc_lanczos_A()`, `exp()`, `GiNaC::lanczos_coeffs::get_order()`, `guess_precision()`, `sin()`, `sqrt()`, `GiNaC::lanczos_coeffs::sufficiently_accurate()`, and `x`.

Referenced by `beta_eval()`, `beta_series()`, `psi2_eval()`, `tgamma()`, `tgamma_conjugate()`, `tgamma_deriv()`, `tgamma_eval()`, `tgamma_evalf()`, and `tgamma_series()`.

**5.1.3.620 tgamma()** [2/2]

```
const numeric GiNaC::tgamma (
    const numeric & x )
```

References `tgamma()`, and `x`.

**5.1.3.621** `psi()` [3/4]

```
const numeric GiNaC::psi (
    const numeric & x )
```

The psi function (aka polygamma function).

This is only a stub!

**5.1.3.622** `psi()` [4/4]

```
const numeric GiNaC::psi (
    const numeric & n,
    const numeric & x )
```

The psi functions (aka polygamma functions).

This is only a stub!

**5.1.3.623** `factorial()`

```
const numeric GiNaC::factorial (
    const numeric & n )
```

Factorial combinatorial function.

**Parameters**

$n$	integer argument $\geq 0$
-----	---------------------------

**Exceptions**

<code>range_error</code>	(argument must be integer $\geq 0$ )
--------------------------	--------------------------------------

References n.

Referenced by `Bernoulli_polynomial()`, `factorial_conjugate()`, `factorial_eval()`, `factorial_evalf()`, `factorial_real_evalf()`, `G2_eval()`, `G2_evalf()`, `G3_eval()`, `G3_evalf()`, `generalised_Bernoulli_number()`, `H_eval()`, `lgamma_eval()`, `multinomial_coefficient()`, `psi2_eval()`, `psi2_series()`, `S_eval()`, `GiNaC::Kronecker_dtau_kernel::series_coeff_impl()`, `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`, `symm()`, `tgamma_eval()`, and `zeta1_eval()`.

**5.1.3.624** `doublefactorial()`

```
const numeric GiNaC::doublefactorial (
    const numeric & n )
```

The double factorial combinatorial function.

(Scarcely used, but still useful in cases, like for exact results of `tgamma(n+1/2)` for instance.)



## Parameters

$n$	integer argument $\geq -1$
-----	----------------------------

## Returns

$n!! == n * (n-2) * (n-4) * \dots * (\{1|2\})$  with  $0!! == (-1)!! == 1$

## Exceptions

<i>range_error</i>	(argument must be integer $\geq -1$ )
--------------------	---------------------------------------

References `_num1_p`, `_num_1_p`, and `n`.

Referenced by `tgamma_eval()`.

5.1.3.625 `binomial()`

```
const numeric GiNaC::binomial (
    const numeric & n,
    const numeric & k )
```

The Binomial coefficients.

It computes the binomial coefficients. For integer  $n$  and  $k$  and positive  $n$  this is the number of ways of choosing  $k$  objects from  $n$  distinct objects. If  $n$  is negative, the formula  $\text{binomial}(n,k) == (-1)^k \text{binomial}(k-n-1,k)$  is used to compute the result.

References `_num0_p`, `_num1_p`, `_num_1_p`, `k`, `n`, and `GiNaC::numeric::power()`.

Referenced by `binomial_conjugate()`, `binomial_eval()`, `binomial_evalf()`, `binomial_real_part()`, `binomial_sym()`, `EllipticE_series()`, `EllipticK_series()`, `GiNaC::power::expand_add()`, `GiNaC::power::imag_part()`, and `GiNaC::power::real_part()`.

5.1.3.626 `bernoulli()`

```
const numeric GiNaC::bernoulli (
    const numeric & nn )
```

Bernoulli number.

The  $n$ th Bernoulli number is the coefficient of  $x^n/n!$  in the expansion of the function  $x/(e^x-1)$ .

## Returns

the  $n$ th Bernoulli number (a rational number).

## Exceptions

<code>range_error</code>	(argument must be integer $\geq 0$ )
--------------------------	--------------------------------------

References `_num1_p`, `c`, `GiNaC::numeric::is_integer()`, `GiNaC::numeric::is_negative()`, `k`, `n`, and `GiNaC::numeric::to_int()`.

Referenced by `GiNaC::Kronecker_dtau_kernel::series_coeff_impl()`, `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`, and `zeta1_eval()`.

5.1.3.627 `fibonacci()`

```
const numeric GiNaC::fibonacci (
    const numeric & n )
```

Fibonacci number.

The  $n$ th Fibonacci number  $F(n)$  is defined by the recurrence formula  $F(n) = F(n-1) + F(n-2)$  with  $F(0) = 0$  and  $F(1) = 1$ .

## Parameters

<code>n</code>	an integer
----------------	------------

## Returns

the  $n$ th Fibonacci number  $F(n)$  (an integer number)

## Exceptions

<code>range_error</code>	(argument must be an integer)
--------------------------	-------------------------------

References `_num0_p`, `m`, and `n`.

5.1.3.628 `abs()`

```
const numeric GiNaC::abs (
    const numeric & x )
```

Absolute value.

References `x`.

Referenced by `abs_conjugate()`, `abs_eval()`, `abs_evalf()`, `abs_expand()`, `abs_expl_derivative()`, `abs_power()`, `abs_real_part()`, `adaptivesimpson()`, `atan()`, `atan_series()`, `atanh_series()`, `GiNaC::power::eval()`, `generalised_Bernoulli_number()`, `H_evalf()`, `GiNaC::power::imag_part()`, `GiNaC::mul::integer_content()`, `GiNaC::numeric::integer_content()`, `is_discriminant_of_quadratic_number_field()`, `Li2_()`, `Li2_projection()`, `log_real_part()`, `GiNaC::add::max_coefficient()`, `GiNaC::mul::max_coefficient()`, `GiNaC::numeric::max_coefficient()`, `GiNaC::matrix::pivot()`, `print_real_number()`, `GiNaC::power::real_part()`, `tgamma_eval()`, `GiNaC::ex::unitcontprim()`, and `zeta1_eval()`.

## 5.1.3.629 mod()

```
const numeric GiNaC::mod (
    const numeric & a,
    const numeric & b )
```

Modulus (in positive representation).

In general,  $\text{mod}(a,b)$  has the sign of  $b$  or is zero, and  $\text{rem}(a,b)$  has the sign of  $a$  or is zero. This is different from Maple's  $\text{modp}$ , where the sign of  $b$  is ignored. It is in agreement with Mathematica's  $\text{Mod}$ .

## Returns

$a \bmod b$  in the range  $[0, \text{abs}(b)-1]$  with sign of  $b$  if both are integer, 0 otherwise.

References `_num0_p`, `GiNaC::numeric::is_integer()`, and `GiNaC::numeric::to_cl_N()`.

Referenced by `GiNaC::Eisenstein_h_kernel::coefficient_a0()`, `GiNaC::Eisenstein_h_kernel::coefficient_an()`, `cos_eval()`, `exp_eval()`, `is_discriminant_of_quadratic_number_field()`, `sin_eval()`, `smod()`, and `tan_eval()`.

## 5.1.3.630 smod()

```
const numeric GiNaC::smod (
    const numeric & a_,
    const numeric & b_ )
```

Modulus (in symmetric representation).

## Returns

$a \bmod b$  in the range  $[-\text{iquo}(\text{abs}(b),2), \text{iquo}(\text{abs}(b),2)]$ .

References `_num0_p`, `GiNaC::numeric::is_integer()`, `m`, `mod()`, and `GiNaC::numeric::to_cl_N()`.

Referenced by `GiNaC::add::smod()`, `GiNaC::mul::smod()`, and `GiNaC::numeric::smod()`.

## 5.1.3.631 irem() [1/2]

```
const numeric GiNaC::irem (
    const numeric & a,
    const numeric & b )
```

Numeric integer remainder.

Equivalent to Maple's  $\text{irem}(a,b)$  as far as sign conventions are concerned. In general,  $\text{mod}(a,b)$  has the sign of  $b$  or is zero, and  $\text{irem}(a,b)$  has the sign of  $a$  or is zero.

## Returns

remainder of  $a/b$  if both are integer, 0 otherwise.

**Exceptions**

<code>overflow_error</code>	(division by zero) if b is zero.
-----------------------------	----------------------------------

References `_num0_p`, `GiNaC::numeric::is_integer()`, `GiNaC::numeric::is_zero()`, `rem()`, and `GiNaC::numeric::to_cl_N()`.

Referenced by `GiNaC::Eisenstein_h_kernel::coefficient_a0()`, `GiNaC::Eisenstein_h_kernel::coefficient_an()`, and `ifactor()`.

**5.1.3.632 `irem()`** [2/2]

```
const numeric GiNaC::irem (
    const numeric & a,
    const numeric & b,
    numeric & q )
```

Numeric integer remainder.

Equivalent to Maple's `irem(a,b,'q')` it obeys the relation `irem(a,b,q) == a - q*b`. In general, `mod(a,b)` has the sign of b or is zero, and `irem(a,b)` has the sign of a or is zero.

**Returns**

remainder of a/b and quotient stored in q if both are integer, 0 otherwise.

**Exceptions**

<code>overflow_error</code>	(division by zero) if b is zero.
-----------------------------	----------------------------------

References `_num0_p`, `GiNaC::numeric::is_integer()`, `GiNaC::numeric::is_zero()`, and `GiNaC::numeric::to_cl_N()`.

**5.1.3.633 `iquo()`** [1/2]

```
const numeric GiNaC::iquo (
    const numeric & a,
    const numeric & b )
```

Numeric integer quotient.

Equivalent to Maple's `iquo` as far as sign conventions are concerned.

**Returns**

truncated quotient of a/b if both are integer, 0 otherwise.

## Exceptions

<code>overflow_error</code>	(division by zero) if b is zero.
-----------------------------	----------------------------------

References `_num0_p`, `GiNaC::numeric::is_integer()`, `GiNaC::numeric::is_zero()`, and `GiNaC::numeric::to_cl_N()`.

Referenced by `GiNaC::power::eval()`, and `heur_gcd_z()`.

5.1.3.634 `iquo()` [2/2]

```
const numeric GiNaC::iquo (
    const numeric & a,
    const numeric & b,
    numeric & r )
```

Numeric integer quotient.

Equivalent to Maple's `iquo(a,b,'r')` it obeys the relation `r == a - iquo(a,b,r)*b`.

## Returns

truncated quotient of `a/b` and remainder stored in `r` if both are integer, 0 otherwise.

## Exceptions

<code>overflow_error</code>	(division by zero) if b is zero.
-----------------------------	----------------------------------

References `_num0_p`, `GiNaC::numeric::is_integer()`, `GiNaC::numeric::is_zero()`, `r`, and `GiNaC::numeric::to_cl_N()`.

5.1.3.635 `gcd()` [2/2]

```
const numeric GiNaC::gcd (
    const numeric & a,
    const numeric & b )
```

Greatest Common Divisor.

## Returns

The GCD of two numbers if both are integer, a numerical 1 if they are not.

References `_num1_p`, `GiNaC::numeric::is_integer()`, and `GiNaC::numeric::to_cl_N()`.

**5.1.3.636** `lcm()` [2/2]

```
const numeric GiNaC::lcm (
    const numeric & a,
    const numeric & b )
```

Least Common Multiple.

**Returns**

The LCM of two numbers if both are integer, the product of those two numbers if they are not.

References `GiNaC::numeric::is_integer()`, `GiNaC::numeric::mul()`, and `GiNaC::numeric::to_cl_N()`.

Referenced by `GiNaC::numeric::denom()`, and `GiNaC::numeric::num()`.

**5.1.3.637** `sqrt()` [1/2]

```
const numeric GiNaC::sqrt (
    const numeric & x )
```

Numeric square root.

If possible, `sqrt(x)` should respect squares of exact numbers, i.e. `sqrt(4)` should return integer 2.

**Parameters**

$x$	numeric argument
-----	------------------

**Returns**

square root of  $x$ . Branch cut along negative real axis, the negative real axis itself where  $\text{imag}(x)=0$  and  $\text{real}(x)<0$  belongs to the upper part where  $\text{imag}(x)>0$ .

References  $x$ .

Referenced by `atan()`, `cos_eval()`, `cosh_eval()`, `EllipticE_evalf()`, `EllipticK_evalf()`, `GiNaC::su3f::eval_indexed()`, `GiNaC::su3d::eval_indexed()`, `sin_eval()`, `sinh_eval()`, `tan_eval()`, `tanh_eval()`, `tgamma()`, and `tgamma_eval()`.

**5.1.3.638** `isqrt()`

```
const numeric GiNaC::isqrt (
    const numeric & x )
```

Integer numeric square root.

References `_num0_p`, and  $x$ .

Referenced by `heur_gcd_z()`.

**5.1.3.639 PiEvalf()**

```
ex GiNaC::PiEvalf ( )
```

Floating point evaluation of Archimedes' constant Pi.

**5.1.3.640 EulerEvalf()**

```
ex GiNaC::EulerEvalf ( )
```

Floating point evaluation of Euler's constant gamma.

**5.1.3.641 CatalanEvalf()**

```
ex GiNaC::CatalanEvalf ( )
```

Floating point evaluation of Catalan's constant.

**5.1.3.642 operator<<()** [6/16]

```
std::ostream& GiNaC::operator<< (
    std::ostream & os,
    const _numeric_digits & e )
```

References GiNaC::\_numeric\_digits::print().

**5.1.3.643 GINAC\_DECLARE\_UNARCHIVER()** [38/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    numeric )
```

**5.1.3.644 pow()** [1/3]

```
const numeric GiNaC::pow (
    const numeric & x,
    const numeric & y ) [inline]
```

References x.

Referenced by `abs_eval()`, `abs_power()`, `GiNaC::mul::algebraic_subs_mul()`, `beta_eval()`, `GiNaC::Eisenstein_h_kernel::coefficient_an()`, `GiNaC::basic::collect()`, `GiNaC::mul::combine_ex_with_coeff_to_pair()`, `GiNaC::mul::combine_pair_with_coeff_to_pair()`, `GiNaC::pseries::convert_to_poly()`, `GiNaC::mul::derivative()`, `GiNaC::power::derivative()`, `divide()`, `divide_in_z()`, `EllipticE_series()`, `EllipticK_series()`, `GiNaC::power::eval()`, `GiNaC::power::evalm()`, `GiNaC::power::expand()`, `GiNaC::power::expand_add()`, `factor()`, `find_common_factor()`, `G2_eval()`, `G2_evalf()`, `G3_eval()`, `G3_evalf()`, `gcd()`, `gcd_pf_pow()`, `gcd_pf_pow_pow()`, `GiNaC::Kronecker_dtau_kernel::get_numerical_value()`, `GiNaC::Kronecker_dz_kernel::get_numerical_value()`, `H_eval()`, `GiNaC::power::imag_part()`, `interpolate()`, `kronecker_symbol()`, `GiNaC::integration_kernel::Laurent_series()`, `lcmcoeff()`, `Li2_series()`, `Li_eval()`, `Li_series()`, `log_series()`, `multiply_lcm()`, `GiNaC::power::normal()`, `GiNaC::pseries::op()`, `GiNaC::pseries::power_const()`, `prem()`, `GiNaC::pseries::print_series()`, `psi1_eval()`, `psi2_eval()`, `GiNaC::Eisenstein_h_kernel::q_expansion_modular_form()`, `quo()`, `GiNaC::power::real_part()`, `rem()`, `replace_with_symbol()`, `S_eval()`, `S_series()`, `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`, `sprem()`, `sqrfree_parfrac()`, `sr_gcd()`, `GiNaC::power::subs()`, `tgamma_eval()`, `GiNaC::power::to_polynomial()`, `GiNaC::power::to_rational()`, and `zeta1_eval()`.

**5.1.3.645 inverse()** [3/3]

```
const numeric GiNaC::inverse (
    const numeric & x ) [inline]
```

References x.

**5.1.3.646 step()**

```
numeric GiNaC::step (
    const numeric & x ) [inline]
```

References x.

Referenced by `abs_eval()`, `H_eval()`, `step_conjugate()`, `step_eval()`, `step_evalf()`, `step_real_part()`, and `step_series()`.

**5.1.3.647 csgn()**

```
int GiNaC::csgn (
    const numeric & x ) [inline]
```

References x.

Referenced by `atan_series()`, `atanh_series()`, `csgn_conjugate()`, `csgn_eval()`, `csgn_evalf()`, `csgn_power()`, `csgn_real_part()`, `csgn_series()`, `eta_eval()`, `eta_evalf()`, and `log_series()`.



**5.1.3.648 is\_zero()** [2/2]

```
bool GiNaC::is_zero (
    const numeric & x ) [inline]
```

References GiNaC::ex::is\_zero(), and x.

**5.1.3.649 is\_positive()**

```
bool GiNaC::is_positive (
    const numeric & x ) [inline]
```

References x.

Referenced by beta\_eval(), and GiNaC::mul::info().

**5.1.3.650 is\_negative()**

```
bool GiNaC::is_negative (
    const numeric & x ) [inline]
```

References x.

Referenced by GiNaC::power::do\_print\_latex(), GiNaC::power::eval(), frac\_cancel(), and GiNaC::mul::info().

**5.1.3.651 is\_integer()**

```
bool GiNaC::is_integer (
    const numeric & x ) [inline]
```

References x.

Referenced by beta\_eval(), binomial\_eval(), GiNaC::power::coeff(), csgn\_power(), GiNaC::power::degree(), GiNaC::power::expand(), GiNaC::power::ldegree(), GiNaC::pseries::power\_const(), psi1\_eval(), psi2\_eval(), replace\_with\_symbol(), and GiNaC::power::series().

**5.1.3.652 is\_pos\_integer()**

```
bool GiNaC::is_pos_integer (
    const numeric & x ) [inline]
```

References x.

Referenced by GiNaC::power::expand\_add(), and GiNaC::power::expand\_add\_2().

#### 5.1.3.653 `is_nonneg_integer()`

```
bool GiNaC::is_nonneg_integer (
    const numeric & x ) [inline]
```

References x.

#### 5.1.3.654 `is_even()`

```
bool GiNaC::is_even (
    const numeric & x ) [inline]
```

References x.

Referenced by `abs_power()`, and `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`.

#### 5.1.3.655 `is_odd()`

```
bool GiNaC::is_odd (
    const numeric & x ) [inline]
```

References x.

#### 5.1.3.656 `is_prime()`

```
bool GiNaC::is_prime (
    const numeric & x ) [inline]
```

References x.

#### 5.1.3.657 `is_rational()`

```
bool GiNaC::is_rational (
    const numeric & x ) [inline]
```

References x.

Referenced by `beta_eval()`, and `replace_with_symbol()`.

**5.1.3.658 is\_real()**

```
bool GiNaC::is_real (
    const numeric & x ) [inline]
```

References x.

Referenced by beta\_eval(), fsolve(), G2\_evalf(), and Li2\_series().

**5.1.3.659 is\_cinteger()**

```
bool GiNaC::is_cinteger (
    const numeric & x ) [inline]
```

References x.

**5.1.3.660 is\_crational()**

```
bool GiNaC::is_crational (
    const numeric & x ) [inline]
```

References x.

**5.1.3.661 to\_int()**

```
int GiNaC::to_int (
    const numeric & x ) [inline]
```

References x.

Referenced by GiNaC::pseries::add\_series(), GiNaC::power::coeff(), GiNaC::power::degree(), divide(), divide\_↵  
in\_z(), GiNaC::power::do\_print\_csrc(), GiNaC::matrix::eval\_indexed(), GiNaC::tensdelta::eval\_indexed(), GiNaC::↵  
C::minkmetric::eval\_indexed(), GiNaC::su3f::eval\_indexed(), GiNaC::su3d::eval\_indexed(), GiNaC::spinmetric↵  
::eval\_indexed(), GiNaC::tensepsilon::eval\_indexed(), expand\_dummy\_sum(), gcd\_pf\_pow(), H\_evalf(), iterated\_↵  
integral\_evalf\_impl(), GiNaC::power::ldegree(), Li\_eval(), Li\_evalf(), GiNaC::pseries::mul\_series(), GiNaC::basic↵  
::operator[](), GiNaC::pseries::power\_const(), S\_eval(), S\_evalf(), GiNaC::pseries::series(), GiNaC::mul::series(),  
GiNaC::integral::series(), GiNaC::power::series(), GiNaC::ELi\_kernel::series\_coeff\_impl(), GiNaC::Ebar\_kernel↵  
::series\_coeff\_impl(), GiNaC::Kronecker\_dtau\_kernel::series\_coeff\_impl(), sqrfree\_parfrac(), and tryfactsubs().

#### 5.1.3.662 to\_long()

```
long GiNaC::to_long (
    const numeric & x ) [inline]
```

References x.

Referenced by GiNaC::power::imag\_part(), and GiNaC::power::real\_part().

#### 5.1.3.663 to\_double()

```
double GiNaC::to_double (
    const numeric & x ) [inline]
```

References x.

#### 5.1.3.664 real()

```
const numeric GiNaC::real (
    const numeric & x ) [inline]
```

References x.

Referenced by cosh\_eval(), csgn\_series(), GiNaC::mul::info(), real\_part\_evalf(), sinh\_eval(), step\_series(), and tanh\_eval().

#### 5.1.3.665 imag()

```
const numeric GiNaC::imag (
    const numeric & x ) [inline]
```

References x.

Referenced by eta\_eval(), eta\_evalf(), G2\_eval(), G2\_evalf(), and imag\_part\_evalf().

#### 5.1.3.666 numer() [2/2]

```
const numeric GiNaC::numer (
    const numeric & x ) [inline]
```

References GiNaC::ex::numer(), and x.

**5.1.3.667** `denom()` [2/2]

```
const numeric GiNaC::denom (  
    const numeric & x ) [inline]
```

References `GiNaC::ex::denom()`, and `x`.

**5.1.3.668** `exadd()`

```
static const ex GiNaC::exadd (  
    const ex & lh,  
    const ex & rh ) [inline], [static]
```

Used internally by `operator+()` to add two `ex` objects.

Referenced by `operator+()`, `operator++()`, `operator+=()`, `operator-()`, `operator--()`, and `operator-=()`.

**5.1.3.669** `exmul()`

```
static const ex GiNaC::exmul (  
    const ex & lh,  
    const ex & rh ) [inline], [static]
```

Used internally by `operator*()` to multiply two `ex` objects.

References `GiNaC::return_types::commutative`, and `GiNaC::ex::return_type()`.

Referenced by `operator*()`, `operator*=(())`, `operator/()`, and `operator/=()`.

**5.1.3.670** `exminus()`

```
static const ex GiNaC::exminus (  
    const ex & lh ) [inline], [static]
```

Used internally by `operator-()` and friends to change the sign of an argument.

References `_ex_1`.

Referenced by `operator-()`, and `operator-=()`.

**5.1.3.671 operator+()** [1/4]

```
const ex GiNaC::operator+ (
    const ex & lh,
    const ex & rh )
```

References `exadd()`.

**5.1.3.672 operator-()** [1/4]

```
const ex GiNaC::operator- (
    const ex & lh,
    const ex & rh )
```

References `exadd()`, and `exminus()`.

**5.1.3.673 operator\*()** [1/2]

```
const ex GiNaC::operator* (
    const ex & lh,
    const ex & rh )
```

References `exmul()`.

**5.1.3.674 operator/()** [1/2]

```
const ex GiNaC::operator/ (
    const ex & lh,
    const ex & rh )
```

References `_ex_1`, and `exmul()`.

**5.1.3.675 operator+()** [2/4]

```
const numeric GiNaC::operator+ (
    const numeric & lh,
    const numeric & rh )
```

References `GiNaC::numeric::add()`.

**5.1.3.676 operator-()** [2/4]

```
const numeric GiNaC::operator- (
    const numeric & lh,
    const numeric & rh )
```

References `GiNaC::numeric::sub()`.

**5.1.3.677 operator\*()** [2/2]

```
const numeric GiNaC::operator* (
    const numeric & lh,
    const numeric & rh )
```

References `GiNaC::numeric::mul()`.

**5.1.3.678 operator/()** [2/2]

```
const numeric GiNaC::operator/ (
    const numeric & lh,
    const numeric & rh )
```

References `GiNaC::numeric::div()`.

**5.1.3.679 operator+=()** [1/2]

```
ex & GiNaC::operator+= (
    ex & lh,
    const ex & rh )
```

References `exadd()`.

**5.1.3.680 operator-=()** [1/2]

```
ex & GiNaC::operator-= (
    ex & lh,
    const ex & rh )
```

References `exadd()`, and `exminus()`.

**5.1.3.681 operator\*=( )** [1/2]

```
ex & GiNaC::operator*= (
    ex & lh,
    const ex & rh )
```

References `exmul()`.

**5.1.3.682 operator/=( )** [1/2]

```
ex & GiNaC::operator/= (
    ex & lh,
    const ex & rh )
```

References `_ex_1`, and `exmul()`.

**5.1.3.683 operator+=( )** [2/2]

```
numeric & GiNaC::operator+= (
    numeric & lh,
    const numeric & rh )
```

References `GiNaC::numeric::add()`.

**5.1.3.684 operator-=( )** [2/2]

```
numeric & GiNaC::operator-= (
    numeric & lh,
    const numeric & rh )
```

References `GiNaC::numeric::sub()`.

**5.1.3.685 operator\*=( )** [2/2]

```
numeric & GiNaC::operator*= (
    numeric & lh,
    const numeric & rh )
```

References `GiNaC::numeric::mul()`.



**5.1.3.686 operator/=( )** [2/2]

```
numeric & GiNaC::operator/= (
    numeric & lh,
    const numeric & rh )
```

References `GiNaC::numeric::div()`.

**5.1.3.687 operator+( )** [3/4]

```
const ex GiNaC::operator+ (
    const ex & lh )
```

**5.1.3.688 operator-( )** [3/4]

```
const ex GiNaC::operator- (
    const ex & lh )
```

References `exminus()`.

**5.1.3.689 operator+( )** [4/4]

```
const numeric GiNaC::operator+ (
    const numeric & lh )
```

**5.1.3.690 operator-( )** [4/4]

```
const numeric GiNaC::operator- (
    const numeric & lh )
```

References `_num_1_p`, and `GiNaC::numeric::mul()`.

**5.1.3.691 operator++( )** [1/4]

```
ex & GiNaC::operator++ (
    ex & rh )
```

Expression prefix increment.

Adds 1 and returns incremented `ex`.

References `_ex1`, and `exadd()`.

**5.1.3.692 operator--()** [1/4]

```
ex & GiNaC::operator-- (
    ex & rh )
```

Expression prefix decrement.

Subtracts 1 and returns decremented ex.

References `_ex_1`, and `exadd()`.

**5.1.3.693 operator++()** [2/4]

```
const ex GiNaC::operator++ (
    ex & lh,
    int )
```

Expression postfix increment.

Returns the ex and leaves the original incremented by 1.

References `_ex1`, and `exadd()`.

**5.1.3.694 operator--()** [2/4]

```
const ex GiNaC::operator-- (
    ex & lh,
    int )
```

Expression postfix decrement.

Returns the ex and leaves the original decremented by 1.

References `_ex_1`, and `exadd()`.

**5.1.3.695 operator++()** [3/4]

```
numeric & GiNaC::operator++ (
    numeric & rh )
```

Numeric prefix increment.

Adds 1 and returns incremented number.

References `_num1_p`, and `GiNaC::numeric::add()`.

**5.1.3.696 operator--()** [3/4]

```
numeric & GiNaC::operator-- (
    numeric & rh )
```

Numeric prefix decrement.

Subtracts 1 and returns decremented number.

References `_num_1_p`, and `GiNaC::numeric::add()`.

**5.1.3.697 operator++()** [4/4]

```
const numeric GiNaC::operator++ (
    numeric & lh,
    int )
```

Numeric postfix increment.

Returns the number and leaves the original incremented by 1.

References `_num1_p`, and `GiNaC::numeric::add()`.

**5.1.3.698 operator--()** [4/4]

```
const numeric GiNaC::operator-- (
    numeric & lh,
    int )
```

Numeric postfix decrement.

Returns the number and leaves the original decremented by 1.

References `_num_1_p`, and `GiNaC::numeric::add()`.

**5.1.3.699 operator==()**

```
const relational GiNaC::operator== (
    const ex & lh,
    const ex & rh )
```

References `GiNaC::relational::equal`.

#### 5.1.3.700 operator!=(())

```
const relational GiNaC::operator!= (
    const ex & lh,
    const ex & rh )
```

References GiNaC::relational::not\_equal.

#### 5.1.3.701 operator<()

```
const relational GiNaC::operator< (
    const ex & lh,
    const ex & rh )
```

References GiNaC::relational::less.

#### 5.1.3.702 operator<=()

```
const relational GiNaC::operator<= (
    const ex & lh,
    const ex & rh )
```

References GiNaC::relational::less\_or\_equal.

#### 5.1.3.703 operator>()

```
const relational GiNaC::operator> (
    const ex & lh,
    const ex & rh )
```

References GiNaC::relational::greater.

#### 5.1.3.704 operator>=()

```
const relational GiNaC::operator>= (
    const ex & lh,
    const ex & rh )
```

References GiNaC::relational::greater\_or\_equal.

**5.1.3.705 my\_ios\_index()**

```
static int GiNaC::my_ios_index ( ) [static]
```

Referenced by `get_print_context()`, and `set_print_context()`.

**5.1.3.706 my\_ios\_callback()**

```
static void GiNaC::my_ios_callback (
    std::ios_base::event ev,
    std::ios_base & s,
    int i ) [static]
```

Referenced by `set_print_context()`.

**5.1.3.707 get\_print\_context()**

```
static print_context* GiNaC::get_print_context (
    std::ios_base & s ) [inline], [static]
```

References `my_ios_index()`.

Referenced by `get_print_options()`, `operator<<()`, and `set_print_options()`.

**5.1.3.708 set\_print\_context()**

```
static void GiNaC::set_print_context (
    std::ios_base & s,
    const print_context & c ) [static]
```

References `c`, `callback_registered`, `my_ios_callback()`, `my_ios_index()`, `GiNaC::print_context::options`, and `options`.

Referenced by `csrc()`, `csrc_cl_N()`, `csrc_double()`, `csrc_float()`, `dflt()`, `latex()`, `python()`, `python_repr()`, `set_print_↵ options()`, and `tree()`.

**5.1.3.709 get\_print\_options()**

```
static unsigned GiNaC::get_print_options (
    std::ios_base & s ) [inline], [static]
```

References `get_print_context()`, and `GiNaC::print_context::options`.

Referenced by `index_dimensions()`, and `no_index_dimensions()`.

#### 5.1.3.710 `set_print_options()`

```
static void GiNaC::set_print_options (
    std::ostream & s,
    unsigned options ) [static]
```

References `get_print_context()`, `GiNaC::print_context::options`, `options`, and `set_print_context()`.

Referenced by `dflt()`, `index_dimensions()`, and `no_index_dimensions()`.

#### 5.1.3.711 `operator<<()` [7/16]

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const ex & e )
```

References `get_print_context()`, and `GiNaC::ex::print()`.

#### 5.1.3.712 `operator>>()` [3/3]

```
std::istream & GiNaC::operator>> (
    std::istream & is,
    ex & e )
```

#### 5.1.3.713 `dflt()`

```
std::ostream & GiNaC::dflt (
    std::ostream & os )
```

References `set_print_context()`, and `set_print_options()`.

#### 5.1.3.714 `latex()`

```
std::ostream & GiNaC::latex (
    std::ostream & os )
```

References `set_print_context()`.

#### 5.1.3.715 python()

```
std::ostream & GiNaC::python (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

#### 5.1.3.716 python\_repr()

```
std::ostream & GiNaC::python_repr (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

#### 5.1.3.717 tree()

```
std::ostream & GiNaC::tree (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

Referenced by [GiNaC::class\\_info< OPT >::dump\\_hierarchy\(\)](#).

#### 5.1.3.718 csrc()

```
std::ostream & GiNaC::csrc (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

#### 5.1.3.719 csrc\_float()

```
std::ostream & GiNaC::csrc_float (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

**5.1.3.720 csrc\_double()**

```
std::ostream & GiNaC::csrc_double (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

**5.1.3.721 csrc\_cl\_N()**

```
std::ostream & GiNaC::csrc_cl_N (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

**5.1.3.722 index\_dimensions()**

```
std::ostream & GiNaC::index_dimensions (
    std::ostream & os )
```

References [get\\_print\\_options\(\)](#), [GiNaC::print\\_options::print\\_index\\_dimensions](#), and [set\\_print\\_options\(\)](#).

**5.1.3.723 no\_index\_dimensions()**

```
std::ostream & GiNaC::no_index_dimensions (
    std::ostream & os )
```

References [get\\_print\\_options\(\)](#), [GiNaC::print\\_options::print\\_index\\_dimensions](#), and [set\\_print\\_options\(\)](#).

**5.1.3.724 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [26/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    power ,
    basic ,
    print_func< print_dflt > &::do_print_dflt. print_func< print_latex > &::do←
_print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_python > &←
::do_print_python. print_func< print_python_repr > &::do_print_python_repr. print_func<
print_csrc_cl_N > &::do_print_csrc_cl_N )
```



## 5.1.3.725 print\_sym\_pow()

```
static void GiNaC::print_sym_pow (
    const print_context & c,
    const symbol & x,
    int exp ) [static]
```

References `c`, `exp()`, `GiNaC::ex::print()`, and `x`.

Referenced by `GiNaC::power::do_print_csrc()`.

## 5.1.3.726 GINAC\_BIND\_UNARCHIVER() [37/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    power )
```

## 5.1.3.727 GINAC\_DECLARE\_UNARCHIVER() [39/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    power )
```

## 5.1.3.728 pow() [2/3]

```
ex GiNaC::pow (
    const ex & b,
    const ex & e ) [inline]
```

Symbolic exponentiation.

Returns a power-object as a new expression.

## Parameters

<i>b</i>	the basis expression
<i>e</i>	the exponent expression

## 5.1.3.729 pow() [3/3]

```
template<typename T1 , typename T2 >
ex GiNaC::pow (
```

```
const T1 & b,
const T2 & e ) [inline]
```

#### 5.1.3.730 `sqrt()` [2/2]

```
ex GiNaC::sqrt (
    const ex & a ) [inline]
```

Square root expression.

Returns a power-object with exponent 1/2.

References `_ex1_2`.

#### 5.1.3.731 `is_a()` [3/3]

```
template<class T >
bool GiNaC::is_a (
    const print_context & obj ) [inline]
```

Check if obj is a T, including base classes.

#### 5.1.3.732 `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [27/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    pseries ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python > &::do_↵
print_python. print_func< print_python_repr > &::do_print_python_repr )
```

#### 5.1.3.733 `GINAC_BIND_UNARCHIVER()` [38/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    pseries )
```

#### 5.1.3.734 `GINAC_DECLARE_UNARCHIVER()` [40/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    pseries )
```

#### 5.1.3.735 `series_to_poly()`

```
ex GiNaC::series_to_poly (
    const ex & e ) [inline]
```

Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.

The result is undefined if the expression does not contain a pseries object at its top level.

## Parameters

<i>e</i>	expression
----------	------------

## Returns

polynomial expression

## See also

[is\\_a<>](#)  
[pseries::convert\\_to\\_poly](#)

Referenced by [Bernoulli\\_polynomial\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), and [GiNaC::modular\\_form\\_kernel::Laurent\\_series\(\)](#).

5.1.3.736 [is\\_terminating\(\)](#)

```
bool GiNaC::is_terminating (
    const pseries & s ) [inline]
```

References [GiNaC::pseries::is\\_terminating\(\)](#).

5.1.3.737 [make\\_return\\_type\\_t\(\)](#)

```
template<typename T >
return\_type\_t GiNaC::make_return_type_t (
    const unsigned rl = 0 ) [inline]
```

References [GiNaC::return\\_type\\_t::rl](#), and [GiNaC::return\\_type\\_t::info](#).

5.1.3.738 [set\\_print\\_func\(\)](#) [1/2]

```
template<class Alg , class Ctx , class T , class C >
void GiNaC::set_print_func (
    void fconst T &, const C &c, unsigned )
```

Add or replace a print method.

References options.

**5.1.3.739** `set_print_func()` [2/2]

```
template<class Alg , class Ctx , class T , class C >
void GiNaC::set_print_func (
    void(T::*)(const C &, unsigned) f )
```

Add or replace a print method.

References options.

**5.1.3.740** `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [28/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    relational ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print↵
    _tree. print_func< print_python_repr > &::do_print_python_repr )
```

**5.1.3.741** `GINAC_BIND_UNARCHIVER()` [39/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    relational )
```

**5.1.3.742** `print_operator()`

```
static void GiNaC::print_operator (
    const print_context & c,
    relational::operators o ) [static]
```

References `c`, `GiNaC::relational::equal`, `GiNaC::relational::greater`, `GiNaC::relational::greater_or_equal`, `GiNaC::relational::less`, `GiNaC::relational::less_or_equal`, and `GiNaC::relational::not_equal`.

Referenced by `GiNaC::relational::do_print()`, and `GiNaC::relational::do_print_python_repr()`.

**5.1.3.743** `GINAC_DECLARE_UNARCHIVER()` [41/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    relational )
```

**5.1.3.744** GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [29/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    symbol ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python_repr > &↵
::do_print_python_repr )
```

References GiNaC::status\_flags::evaluated, and GiNaC::status\_flags::expanded.

**5.1.3.745** get\_default\_TeX\_name()

```
static const std::string & GiNaC::get_default_TeX_name (
    const std::string & name ) [static]
```

Return default TeX name for symbol.

This recognizes some greek letters.

Referenced by GiNaC::symbol::do\_print\_latex(), and GiNaC::symbol::get\_TeX\_name().

**5.1.3.746** GINAC\_BIND\_UNARCHIVER() [40/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    symbol )
```

**5.1.3.747** GINAC\_BIND\_UNARCHIVER() [41/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    realsymbol )
```

**5.1.3.748** GINAC\_BIND\_UNARCHIVER() [42/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    possymbol )
```

**5.1.3.749 GINAC\_DECLARE\_UNARCHIVER()** [42/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    symbol )
```

**5.1.3.750 GINAC\_DECLARE\_UNARCHIVER()** [43/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    realsymbol )
```

**5.1.3.751 GINAC\_DECLARE\_UNARCHIVER()** [44/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    possymbol )
```

**5.1.3.752 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [30/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    symmetry ,
    basic ,
    print_func< print\_context > &::do_print.  print_func< print\_tree > &::do_print↵
    _tree )
```

References `GiNaC::status_flags::evaluated`, and `GiNaC::status_flags::expanded`.

**5.1.3.753 GINAC\_BIND\_UNARCHIVER()** [43/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    symmetry )
```

**5.1.3.754 index0()**

```
static const symmetry& GiNaC::index0 ( ) [static]
```

Referenced by `antisymmetric2()`, `antisymmetric3()`, `antisymmetric4()`, `symmetric2()`, `symmetric3()`, and `symmetric4()`.

**5.1.3.755 index1()**

```
static const symmetry& GiNaC::index1 ( ) [static]
```

Referenced by antisymmetric2(), antisymmetric3(), antisymmetric4(), symmetric2(), symmetric3(), and symmetric4().

**5.1.3.756 index2()**

```
static const symmetry& GiNaC::index2 ( ) [static]
```

Referenced by antisymmetric3(), antisymmetric4(), symmetric3(), and symmetric4().

**5.1.3.757 index3()**

```
static const symmetry& GiNaC::index3 ( ) [static]
```

Referenced by antisymmetric4(), and symmetric4().

**5.1.3.758 not\_symmetric()**

```
const symmetry & GiNaC::not_symmetric ( )
```

Referenced by GiNaC::indexed::read\_archive().

**5.1.3.759 symmetric2()**

```
const symmetry & GiNaC::symmetric2 ( )
```

References index0(), index1(), and GiNaC::symmetry::symmetric.

Referenced by delta\_tensor(), GiNaC::clifford::get\_metric(), lorentz\_g(), and metric\_tensor().

**5.1.3.760 symmetric3()**

```
const symmetry & GiNaC::symmetric3 ( )
```

References index0(), index1(), index2(), and GiNaC::symmetry::symmetric.

Referenced by color\_d().

**5.1.3.761 symmetric4()**

```
const symmetry & GiNaC::symmetric4 ( )
```

References `index0()`, `index1()`, `index2()`, `index3()`, and `GiNaC::symmetry::symmetric`.

**5.1.3.762 antisymmetric2()**

```
const symmetry & GiNaC::antisymmetric2 ( )
```

References `GiNaC::symmetry::antisymmetric`, `index0()`, and `index1()`.

Referenced by `epsilon_tensor()`, and `spinor_metric()`.

**5.1.3.763 antisymmetric3()**

```
const symmetry & GiNaC::antisymmetric3 ( )
```

References `GiNaC::symmetry::antisymmetric`, `index0()`, `index1()`, and `index2()`.

Referenced by `color_f()`, and `epsilon_tensor()`.

**5.1.3.764 antisymmetric4()**

```
const symmetry & GiNaC::antisymmetric4 ( )
```

References `GiNaC::symmetry::antisymmetric`, `index0()`, `index1()`, `index2()`, and `index3()`.

Referenced by `lorentz_eps()`.

**5.1.3.765 canonicalize()**

```
int GiNaC::canonicalize (
    exvector::iterator v,
    const symmetry & symm )
```

Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.

**Parameters**

<i>v</i>	Start of expression vector
<i>symm</i>	Root node of symmetry tree



**Returns**

the overall sign introduced by the reordering (+1, -1 or 0) or `numeric_limits<int>::max()` if nothing changed

References `GiNaC::symmetry::antisymmetric`, `GiNaC::ex::begin()`, `GiNaC::symmetry::cyclic`, `cyclic_permutation()`, `GiNaC::ex::end()`, `GINAC_ASSERT`, `last`, `permutation_sign()`, `shaker_sort()`, `symm()`, and `GiNaC::symmetry::symmetric`.

Referenced by `GiNaC::indexed::eval()`, and `GiNaC::function::eval()`.

**5.1.3.766 `symm()`**

```
static ex GiNaC::symm (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last,
    bool asymmetric ) [static]
```

References `GiNaC::container< C >::append()`, `factorial()`, `last`, `GiNaC::subs_options::no_index_renaming`, `GiNaC::subs_options::no_pattern`, `GiNaC::container< C >::op()`, `permutation_sign()`, and `GiNaC::ex::subs()`.

Referenced by `GiNaC::ex::antisymmetrize()`, `antisymmetrize()`, `canonicalize()`, `GiNaC::indexed::read_archive()`, `GiNaC::ex::symmetrize()`, and `symmetrize()`.

**5.1.3.767 `symmetrize()` [3/4]**

```
ex GiNaC::symmetrize (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last )
```

Symmetrize expression over a set of objects (symbols, indices).

References `last`, and `symm()`.

**5.1.3.768 `antisymmetrize()` [3/4]**

```
ex GiNaC::antisymmetrize (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last )
```

Antisymmetrize expression over a set of objects (symbols, indices).

References `last`, and `symm()`.

5.1.3.769 `symmetrize_cyclic()` [3/4]

```
ex GiNaC::symmetrize_cyclic (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last )
```

Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).

References `GiNaC::container< C >::append()`, `last`, `GiNaC::subs_options::no_index_renaming`, `GiNaC::subs_options::no_pattern`, `GiNaC::container< C >::op()`, `GiNaC::container< C >::remove_first()`, and `GiNaC::ex::subs()`.

5.1.3.770 `GINAC_DECLARE_UNARCHIVER()` [45/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    symmetry )
```

5.1.3.771 `sy_none()` [1/4]

```
symmetry GiNaC::sy_none ( ) [inline]
```

5.1.3.772 `sy_none()` [2/4]

```
symmetry GiNaC::sy_none (
    const symmetry & c1,
    const symmetry & c2 ) [inline]
```

References `GiNaC::symmetry::none`.

5.1.3.773 `sy_none()` [3/4]

```
symmetry GiNaC::sy_none (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3 ) [inline]
```

References `GiNaC::symmetry::add()`, and `GiNaC::symmetry::none`.

**5.1.3.774** `sy_none()` [4/4]

```
symmetry GiNaC::sy_none (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4 ) [inline]
```

References `GiNaC::symmetry::add()`, and `GiNaC::symmetry::none`.

**5.1.3.775** `sy_symm()` [1/4]

```
symmetry GiNaC::sy_symm ( ) [inline]
```

References `GiNaC::symmetry::set_type()`, and `GiNaC::symmetry::symmetric`.

Referenced by `GiNaC::indexed::read_archive()`.

**5.1.3.776** `sy_symm()` [2/4]

```
symmetry GiNaC::sy_symm (
    const symmetry & c1,
    const symmetry & c2 ) [inline]
```

References `GiNaC::symmetry::symmetric`.

**5.1.3.777** `sy_symm()` [3/4]

```
symmetry GiNaC::sy_symm (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3 ) [inline]
```

References `GiNaC::symmetry::add()`, and `GiNaC::symmetry::symmetric`.

**5.1.3.778** `sy_symm()` [4/4]

```
symmetry GiNaC::sy_symm (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4 ) [inline]
```

References `GiNaC::symmetry::add()`, and `GiNaC::symmetry::symmetric`.

**5.1.3.779 sy\_anti()** [1/4]

```
symmetry GiNaC::sy_anti ( ) [inline]
```

References GiNaC::symmetry::antisymmetric, and GiNaC::symmetry::set\_type().

Referenced by GiNaC::indexed::read\_archive().

**5.1.3.780 sy\_anti()** [2/4]

```
symmetry GiNaC::sy_anti (
    const symmetry & c1,
    const symmetry & c2 ) [inline]
```

References GiNaC::symmetry::antisymmetric.

**5.1.3.781 sy\_anti()** [3/4]

```
symmetry GiNaC::sy_anti (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3 ) [inline]
```

References GiNaC::symmetry::add(), and GiNaC::symmetry::antisymmetric.

**5.1.3.782 sy\_anti()** [4/4]

```
symmetry GiNaC::sy_anti (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4 ) [inline]
```

References GiNaC::symmetry::add(), and GiNaC::symmetry::antisymmetric.

**5.1.3.783 sy\_cycl()** [1/4]

```
symmetry GiNaC::sy_cycl ( ) [inline]
```

References GiNaC::symmetry::cyclic, and GiNaC::symmetry::set\_type().

**5.1.3.784** `sy_cycl()` [2/4]

```
symmetry GiNaC::sy_cycl (
    const symmetry & c1,
    const symmetry & c2 ) [inline]
```

References `GiNaC::symmetry::cyclic`.

**5.1.3.785** `sy_cycl()` [3/4]

```
symmetry GiNaC::sy_cycl (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3 ) [inline]
```

References `GiNaC::symmetry::add()`, and `GiNaC::symmetry::cyclic`.

**5.1.3.786** `sy_cycl()` [4/4]

```
symmetry GiNaC::sy_cycl (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4 ) [inline]
```

References `GiNaC::symmetry::add()`, and `GiNaC::symmetry::cyclic`.

**5.1.3.787** `symmetrize()` [4/4]

```
ex GiNaC::symmetrize (
    const ex & e,
    const exvector & v ) [inline]
```

Symmetrize expression over a set of objects (symbols, indices).

References `GiNaC::ex::begin()`, and `symmetrize()`.

**5.1.3.788** `antisymmetrize()` [4/4]

```
ex GiNaC::antisymmetrize (
    const ex & e,
    const exvector & v ) [inline]
```

Antisymmetrize expression over a set of objects (symbols, indices).

References `antisymmetrize()`, and `GiNaC::ex::begin()`.

**5.1.3.789 symmetrize\_cyclic()** [4/4]

```
ex GiNaC::symmetrize_cyclic (
    const ex & e,
    const exvector & v ) [inline]
```

Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).

References GiNaC::ex::begin(), and symmetrize().

**5.1.3.790 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [31/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    tensdelta ,
    tensor ,
    print_func< print_dflt > &::do_print. print_func< print_latex > &::do_print_↵
    latex )
```

**5.1.3.791 print\_func< print\_dflt >()** [3/3]

```
GiNaC::print_func< print_dflt > (
    &tensmetric::do_print ) &
```

References GiNaC::status\_flags::evaluated, GiNaC::status\_flags::expanded, and GiNaC::basic::setflag().

**5.1.3.792 GINAC\_BIND\_UNARCHIVER()** [44/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    minkmetric )
```

**5.1.3.793 GINAC\_BIND\_UNARCHIVER()** [45/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    tensepsilon )
```

**5.1.3.794 GINAC\_BIND\_UNARCHIVER()** [46/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    tensdelta )
```

**5.1.3.795** GINAC\_BIND\_UNARCHIVER() [47/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    tensmetric )
```

**5.1.3.796** GINAC\_BIND\_UNARCHIVER() [48/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    spinmetric )
```

**5.1.3.797** delta\_tensor()

```
ex GiNaC::delta_tensor (
    const ex & i1,
    const ex & i2 )
```

Create a delta tensor with specified indices.

The indices must be of class `idx` or a subclass. The delta tensor is always symmetric and its trace is the dimension of the index space.

**Parameters**

<i>i1</i>	First index
<i>i2</i>	Second index

**Returns**

newly constructed delta tensor

References `symmetric2()`.

Referenced by `color_trace()`, `GiNaC::su3f::contract_with()`, `GiNaC::su3d::contract_with()`, `GiNaC::spinmetric::contract_with()`, `GiNaC::tensepsilon::contract_with()`, and `GiNaC::tensmetric::eval_indexed()`.

**5.1.3.798** metric\_tensor()

```
ex GiNaC::metric_tensor (
    const ex & i1,
    const ex & i2 )
```

Create a symmetric metric tensor with specified indices.

The indices must be of class `varidx` or a subclass. A metric tensor with one covariant and one contravariant index is equivalent to the delta tensor.

**Parameters**

<i>i1</i>	First index
<i>i2</i>	Second index

**Returns**

newly constructed metric tensor

References `symmetric2()`.

Referenced by `GiNaC::tensepsilon::contract_with()`.

**5.1.3.799 `lorentz_g()`**

```
ex GiNaC::lorentz_g (
    const ex & i1,
    const ex & i2,
    bool pos_sig = false )
```

Create a Minkowski metric tensor with specified indices.

The indices must be of class `varidx` or a subclass. The Lorentz metric is a symmetric tensor with a matrix representation of `diag(1,-1,-1,...)` (negative signature, the default) or `diag(-1,1,1,...)` (positive signature).

**Parameters**

<i>i1</i>	First index
<i>i2</i>	Second index
<i>pos_sig</i>	Whether the signature is positive

**Returns**

newly constructed Lorentz metric tensor

References `symmetric2()`.

Referenced by `GiNaC::tensepsilon::contract_with()`.

**5.1.3.800 `spinor_metric()`**

```
ex GiNaC::spinor_metric (
    const ex & i1,
    const ex & i2 )
```

Create a spinor metric tensor with specified indices.

The indices must be of class `spinidx` or a subclass and have a dimension of 2. The spinor metric is an antisymmetric tensor with a matrix representation of `[[ [ 0, 1 ], [ -1, 0 ] ]]`.



## Parameters

<i>i1</i>	First index
<i>i2</i>	Second index

## Returns

newly constructed spinor metric tensor

References `antisymmetric2()`.

5.1.3.801 `epsilon_tensor()` [1/2]

```
ex GiNaC::epsilon_tensor (
    const ex & i1,
    const ex & i2 )
```

Create an epsilon tensor in a Euclidean space with two indices.

The indices must be of class `idx` or a subclass, and have a dimension of 2.

## Parameters

<i>i1</i>	First index
<i>i2</i>	Second index

## Returns

newly constructed epsilon tensor

References `_ex2`, `antisymmetric2()`, `GiNaC::basic::hold()`, `GiNaC::ex::is_equal()`, and `GiNaC::ex::op()`.

5.1.3.802 `epsilon_tensor()` [2/2]

```
ex GiNaC::epsilon_tensor (
    const ex & i1,
    const ex & i2,
    const ex & i3 )
```

Create an epsilon tensor in a Euclidean space with three indices.

The indices must be of class `idx` or a subclass, and have a dimension of 3.

## Parameters

<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index

**Returns**

newly constructed epsilon tensor

References `_ex3`, `antisymmetric3()`, `GiNaC::basic::hold()`, `GiNaC::ex::is_equal()`, and `GiNaC::ex::op()`.

**5.1.3.803 `lorentz_eps()`**

```
ex GiNaC::lorentz_eps (
    const ex & i1,
    const ex & i2,
    const ex & i3,
    const ex & i4,
    bool pos_sig = false )
```

Create an epsilon tensor in a Minkowski space with four indices.

The indices must be of class `varidx` or a subclass, and have a dimension of 4.

**Parameters**

<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index
<i>i4</i>	Fourth index
<i>pos_sig</i>	Whether the signature of the metric is positive

**Returns**

newly constructed epsilon tensor

References `_ex4`, `antisymmetric4()`, `GiNaC::basic::hold()`, `GiNaC::ex::is_equal()`, and `GiNaC::ex::op()`.

**5.1.3.804 `GINAC_DECLARE_UNARCHIVER()`** [46/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    tensdelta )
```

**5.1.3.805 `GINAC_DECLARE_UNARCHIVER()`** [47/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    tensmetric )
```

**5.1.3.806** GINAC\_DECLARE\_UNARCHIVER() [48/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    minkmetric )
```

**5.1.3.807** GINAC\_DECLARE\_UNARCHIVER() [49/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    spinmetric )
```

**5.1.3.808** GINAC\_DECLARE\_UNARCHIVER() [50/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    tensepsilon )
```

**5.1.3.809** log2()

```
unsigned GiNaC::log2 (
    unsigned n )
```

Integer binary logarithm.

References *k*, and *n*.

Referenced by GiNaC::remember\_table::remember\_table().

**5.1.3.810** multinomial\_coefficient()

```
const numeric GiNaC::multinomial_coefficient (
    const std::vector< unsigned > & p )
```

Compute the multinomial coefficient  $n!/(p_1! * p_2! * \dots * p_k!)$  where  $n = p_1 + p_2 + \dots + p_k$ , i.e.

*p* is a partition of *n*.

References GiNaC::numeric::div(), factorial(), and *n*.

Referenced by GiNaC::power::expand\_add().

**5.1.3.811 rotate\_left()**

```
unsigned GiNaC::rotate_left (
    unsigned n ) [inline]
```

Rotate bits of unsigned value by one bit to the left.

This can be necessary if the user wants to define its own hashes.

References n.

Referenced by `GiNaC::idx::calchash()`, `GiNaC::symmetry::calchash()`, `GiNaC::relational::calchash()`, `GiNaC::basic::calchash()`, and `GiNaC::function::calchash()`.

**5.1.3.812 compare\_pointers()**

```
template<class T >
int GiNaC::compare_pointers (
    const T * a,
    const T * b ) [inline]
```

Compare two pointers (just to establish some sort of canonical order).

**Returns**

-1, 0, or 1

Referenced by `GiNaC::basic::compare_same_type()`.

**5.1.3.813 golden\_ratio\_hash()**

```
unsigned GiNaC::golden_ratio_hash (
    uintptr_t n ) [inline]
```

Truncated multiplication with golden ratio, for computing hash values.

References n.

Referenced by `GiNaC::wildcard::calchash()`, `GiNaC::constant::calchash()`, `GiNaC::symbol::calchash()`, `GiNaC::numeric::calchash()`, `GiNaC::function::calchash()`, and `make_hash_seed()`.

**5.1.3.814 permutation\_sign()** [1/2]

```
template<class It >
int GiNaC::permutation_sign (
    It first,
    It last )
```

References `last`, `swap()`, and `std::swap()`.

Referenced by `canonicalize()`, `GiNaC::matrix::determinant()`, `GiNaC::tensepsilon::eval_indexed()`, and `symm()`.

**5.1.3.815 permutation\_sign()** [2/2]

```
template<class It , class Cmp , class Swap >
int GiNaC::permutation_sign (
    It first,
    It last,
    Cmp comp,
    Swap swapit )
```

References `last`.

**5.1.3.816 shaker\_sort()**

```
template<class It , class Cmp , class Swap >
void GiNaC::shaker_sort (
    It first,
    It last,
    Cmp comp,
    Swap swapit )
```

References `last`.

Referenced by `canonicalize()`, `find_free_and_dummy()`, and `rename_dummy_indices()`.

**5.1.3.817 cyclic\_permutation()**

```
template<class It , class Swap >
void GiNaC::cyclic_permutation (
    It first,
    It last,
    It new_first,
    Swap swapit )
```

References `last`.

Referenced by `canonicalize()`.

**5.1.3.818** `format_index_value()` [1/2]

```
template<typename T >
std::enable_if<has\_distance<T>::value, typename std::iterator_traits<T>::difference_type>↵
::type GiNaC::format_index_value (
    const T & a,
    const T & b )
```

For printing a multi-index: If the templates are used, where T is an iterator, printing the address where the iterator points to is not meaningful.

However, we may print the difference to the starting point.

Referenced by operator<<().

**5.1.3.819** `format_index_value()` [2/2]

```
template<typename T >
std::enable_if<!has\_distance<T>::value, T>::type GiNaC::format_index_value (
    const T & a,
    const T & b )
```

For all other cases we simply print the value.

**5.1.3.820** `operator<<()` [8/16]

```
template<class T >
std::ostream& GiNaC::operator<< (
    std::ostream & os,
    const basic\_multi\_iterator< T > & v ) [inline]
```

Output operator.

A multi\_iterator prints out as [basic\\_multi\\_iterator](#)(  $n_0, n_1, \dots$  ).

References `format_index_value()`.

**5.1.3.821** `operator<<()` [9/16]

```
template<class T >
std::ostream& GiNaC::operator<< (
    std::ostream & os,
    const multi\_iterator\_ordered< T > & v ) [inline]
```

Output operator.

A [multi\\_iterator\\_ordered](#) prints out as [multi\\_iterator\\_ordered](#)(  $n_0, n_1, \dots$  ).

References `format_index_value()`.

**5.1.3.822 operator<<()** [10/16]

```
template<class T >
std::ostream& GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq< T > & v ) [inline]
```

Output operator.

A [multi\\_iterator\\_ordered\\_eq](#) prints out as [multi\\_iterator\\_ordered\\_eq](#)( $n_0, n_1, \dots$ ).

References [format\\_index\\_value\(\)](#).

**5.1.3.823 operator<<()** [11/16]

```
template<class T >
std::ostream& GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq_indv< T > & v ) [inline]
```

Output operator.

A [multi\\_iterator\\_ordered\\_eq\\_indv](#) prints out as [multi\\_iterator\\_ordered\\_eq\\_indv](#)( $n_0, n_1, \dots$ ).

References [format\\_index\\_value\(\)](#).

**5.1.3.824 operator<<()** [12/16]

```
template<class T >
std::ostream& GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_counter< T > & v ) [inline]
```

Output operator.

A [multi\\_iterator\\_counter](#) prints out as [multi\\_iterator\\_counter](#)( $n_0, n_1, \dots$ ).

References [format\\_index\\_value\(\)](#).

**5.1.3.825 operator<<()** [13/16]

```
template<class T >
std::ostream& GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_counter_indv< T > & v ) [inline]
```

Output operator.

A [multi\\_iterator\\_counter\\_indv](#) prints out as [multi\\_iterator\\_counter\\_indv](#)( $n_0, n_1, \dots$ ).

References [format\\_index\\_value\(\)](#).

**5.1.3.826 operator<<()** [14/16]

```
template<class T >
std::ostream& GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_permutation< T > & v ) [inline]
```

Output operator.

A [multi\\_iterator\\_permutation](#) prints out as [multi\\_iterator\\_permutation](#)(  $n_0, n_1, \dots$  ).

References [format\\_index\\_value\(\)](#).

**5.1.3.827 operator<<()** [15/16]

```
template<class T >
std::ostream& GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_shuffle< T > & v ) [inline]
```

Output operator.

A [multi\\_iterator\\_shuffle](#) prints out as [multi\\_iterator\\_shuffle](#)(  $n_0, n_1, \dots$  ).

References [format\\_index\\_value\(\)](#).

**5.1.3.828 operator<<()** [16/16]

```
template<class T >
std::ostream& GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_shuffle_prime< T > & v ) [inline]
```

Output operator.

A [multi\\_iterator\\_shuffle\\_prime](#) prints out as [multi\\_iterator\\_shuffle\\_prime](#)(  $n_0, n_1, \dots$  ).

References [format\\_index\\_value\(\)](#).

**5.1.3.829 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [32/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    wildcard ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print↔
_tree. print_func< print_python_repr > &::do_print_python_repr )
```

References [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).



**5.1.3.830 GINAC\_BIND\_UNARCHIVER()** [49/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    wildcard )
```

**5.1.3.831 haswild()**

```
bool GiNaC::haswild (
    const ex & x )
```

Check whether x has a wildcard anywhere as a subexpression.

References GiNaC::ex::nops(), GiNaC::ex::op(), and x.

Referenced by GiNaC::integral::eval().

**5.1.3.832 GINAC\_DECLARE\_UNARCHIVER()** [51/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    wildcard )
```

**5.1.3.833 wild()**

```
ex GiNaC::wild (
    unsigned label = 0 ) [inline]
```

Create a wildcard object with the specified label.

**5.1.4 Variable Documentation****5.1.4.1 unarch\_table\_instance**

```
unarchive_table_t GiNaC::unarch_table_instance [static]
```

#### 5.1.4.2 map\_evalm

`GiNaC::evalm_map_function` `GiNaC::map_evalm`

Referenced by `GiNaC::basic::evalm()`.

#### 5.1.4.3 map\_eval\_integ

`GiNaC::eval_integ_map_function` `GiNaC::map_eval_integ`

Referenced by `GiNaC::basic::eval_integ()`.

#### 5.1.4.4 tensor

`GiNaC::tensor`

#### 5.1.4.5 Pi

`const constant` `GiNaC::Pi`

Pi.

(3.14159...) Diverts straight into CLN for `evalf()`.

Referenced by `acos_eval()`, `acosh_eval()`, `asin_eval()`, `atan2_eval()`, `atan_eval()`, `atan_series()`, `atanh_series()`, `GiNaC::Eisenstein_h_kernel::coefficient_a0()`, `GiNaC::Eisenstein_h_kernel::coefficient_an()`, `cos_eval()`, `cosh_eval()`, `EllipticE_eval()`, `EllipticE_evalf()`, `EllipticE_series()`, `EllipticK_eval()`, `EllipticK_evalf()`, `EllipticK_series()`, `eta_eval()`, `eta_evalf()`, `exp_eval()`, `GiNaC::Kronecker_dtau_kernel::get_numerical_value()`, `GiNaC::Kronecker_dz_kernel::get_numerical_value()`, `H_evalf()`, `Li2_eval()`, `Li2_series()`, `Li_eval()`, `log_eval()`, `log_series()`, `GiNaC::constant::read_archive()`, `GiNaC::Kronecker_dtau_kernel::series_coeff_impl()`, `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`, `sin_eval()`, `sinh_eval()`, `tan_eval()`, `tan_series()`, `tanh_eval()`, `tanh_series()`, `tgamma_eval()`, and `zeta1_eval()`.

#### 5.1.4.6 Euler

`const constant` `GiNaC::Euler`

Euler's constant.

(0.57721...) Sometimes called Euler-Mascheroni constant. Diverts straight into CLN for `evalf()`.

Referenced by `psi1_eval()`, and `GiNaC::constant::read_archive()`.

#### 5.1.4.7 Catalan

```
const constant GiNaC::Catalan
```

Catalan's constant.

(0.91597...) Diverts straight into CLN for [evalf\(\)](#).

Referenced by [Li2\\_eval\(\)](#), [Li\\_eval\(\)](#), and [GiNaC::constant::read\\_archive\(\)](#).

#### 5.1.4.8 crctab

```
unsigned const GiNaC::crctab[256] [static]
```

Referenced by [crc32\(\)](#).

#### 5.1.4.9 library\_initializer

```
library\_init GiNaC::library_initializer [static]
```

For construction of flyweights, etc.

#### 5.1.4.10 \_num0\_bp

```
const basic * GiNaC::_num0_bp
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.11 idx

```
GiNaC::idx
```

Referenced by [expand\\_dummy\\_sum\(\)](#).

#### 5.1.4.12 force\_include\_tgamma

```
unsigned GiNaC::force_include_tgamma = tgamma_SERIAL::serial
```

#### 5.1.4.13 force\_include\_zeta1

```
unsigned GiNaC::force_include_zeta1 = zetal_SERIAL::serial
```

#### 5.1.4.14 I

```
const numeric GiNaC::I = numeric(cln::complex(cln::cl_I(0),cln::cl_I(1)))
```

Imaginary unit.

This is not a constant but a numeric since we are natively handing complex numbers anyways, so in each expression containing an I it is automatically eval'ed away anyhow.

Referenced by `acosh_eval()`, `atan_eval()`, `atan_series()`, `atanh_series()`, `GiNaC::Eisenstein_h_kernel::coefficient_a0()`, `GiNaC::Eisenstein_h_kernel::coefficient_an()`, `color_h()`, `GiNaC::su3f::contract_with()`, `cosh_eval()`, `csgn_eval()`, `eta_eval()`, `eta_evalf()`, `eta_imag_part()`, `exp_eval()`, `GiNaC::Kronecker_dtau_kernel::get_numerical_value()`, `GiNaC::Kronecker_dz_kernel::get_numerical_value()`, `H_evalf()`, `GiNaC::numeric::has()`, `Li2_eval()`, `Li2_series()`, `Li_eval()`, `log_eval()`, `log_series()`, `GiNaC::numeric::normal()`, `GiNaC::Kronecker_dtau_kernel::series_coeff_impl()`, `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`, `sinh_eval()`, `step_eval()`, `tanh_eval()`, `tanh_series()`, `GiNaC::numeric::to_polynomial()`, and `GiNaC::numeric::to_rational()`.

#### 5.1.4.15 Digits

```
_numeric_digits GiNaC::Digits
```

Accuracy in decimal digits.

Only object of this type! Can be set using assignment from C++ unsigned ints and evaluated like any built-in type.

Referenced by `GiNaC::integration_kernel::get_numerical_value_impl()`, `iterated_integral_evalf_impl()`, `GiNaC::numeric::numeric()`, `print_real_cl_N()`, and `zeta1_evalf()`.

#### 5.1.4.16 next\_print\_context\_id

```
unsigned GiNaC::next_print_context_id = 0
```

Next free ID for `print_context` types.

#### 5.1.4.17 version\_major

```
const int GiNaC::version_major = GINACLIB_MAJOR_VERSION
```

#### 5.1.4.18 version\_minor

```
const int GiNaC::version_minor = GINACLIB_MINOR_VERSION
```

#### 5.1.4.19 version\_micro

```
const int GiNaC::version_micro = GINACLIB_MICRO_VERSION
```

#### 5.1.4.20 \_num\_120\_p

```
const numeric * GiNaC::_num_120_p
```

Referenced by `GiNaC::library_init::library_init()`.

#### 5.1.4.21 \_ex\_120

```
const ex GiNaC::_ex_120 = _ex_120
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.22 \_num\_60\_p

```
const numeric * GiNaC::_num_60_p
```

Referenced by `GiNaC::library_init::library_init()`.

#### 5.1.4.23 \_ex\_60

```
const ex GiNaC::_ex_60 = _ex_60
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.24 `_num_48_p`

```
const numeric * GiNaC::_num_48_p
```

Referenced by `GiNaC::library_init::library_init()`.

#### 5.1.4.25 `_ex_48`

```
const ex GiNaC::_ex_48 = _ex_48
```

Referenced by `Li2_eval()`, `Li_eval()`, `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.26 `_num_30_p`

```
const numeric * GiNaC::_num_30_p
```

Referenced by `GiNaC::library_init::library_init()`.

#### 5.1.4.27 `_ex_30`

```
const ex GiNaC::_ex_30 = _ex_30
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.28 `_num_25_p`

```
const numeric * GiNaC::_num_25_p
```

Referenced by `GiNaC::library_init::library_init()`.

#### 5.1.4.29 `_ex_25`

```
const ex GiNaC::_ex_25 = _ex_25
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.30 `_num_24_p`

```
const numeric * GiNaC::_num_24_p
```

Referenced by `GiNaC::library_init::library_init()`.

#### 5.1.4.31 `_ex_24`

```
const ex GiNaC::_ex_24 = _ex_24
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.32 `_num_20_p`

```
const numeric * GiNaC::_num_20_p
```

Referenced by `GiNaC::library_init::library_init()`.

#### 5.1.4.33 `_ex_20`

```
const ex GiNaC::_ex_20 = _ex_20
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.34 `_num_18_p`

```
const numeric * GiNaC::_num_18_p
```

Referenced by `GiNaC::library_init::library_init()`.

#### 5.1.4.35 `_ex_18`

```
const ex GiNaC::_ex_18 = _ex_18
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.36 `_num_15_p`

```
const numeric * GiNaC::_num_15_p
```

Referenced by `GiNaC::library_init::library_init()`.

#### 5.1.4.37 `_ex_15`

```
const ex GiNaC::_ex_15 = _ex_15
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.38 `_num_12_p`

```
const numeric * GiNaC::_num_12_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, and `GiNaC::library_init::library_init()`.

#### 5.1.4.39 `_ex_12`

```
const ex GiNaC::_ex_12 = _ex_12
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.40 `_num_11_p`

```
const numeric * GiNaC::_num_11_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, and `GiNaC::library_init::library_init()`.

#### 5.1.4.41 `_ex_11`

```
const ex GiNaC::_ex_11 = _ex_11
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.



#### 5.1.4.42 `_num_10_p`

```
const numeric * GiNaC::_num_10_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, and `GiNaC::library_init↵::library_init()`.

#### 5.1.4.43 `_ex_10`

```
const ex GiNaC::_ex_10 = _ex_10
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.44 `_num_9_p`

```
const numeric * GiNaC::_num_9_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, and `GiNaC::library_init↵::library_init()`.

#### 5.1.4.45 `_ex_9`

```
const ex GiNaC::_ex_9 = _ex_9
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.46 `_num_8_p`

```
const numeric * GiNaC::_num_8_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, and `GiNaC::library_init↵::library_init()`.

#### 5.1.4.47 `_ex_8`

```
const ex GiNaC::_ex_8 = _ex_8
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.48 `_num_7_p`

```
const numeric * GiNaC::_num_7_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, and `GiNaC::library_init↵::library_init()`.

#### 5.1.4.49 `_ex_7`

```
const ex GiNaC::_ex_7 = _ex_7
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.50 `_num_6_p`

```
const numeric * GiNaC::_num_6_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, and `GiNaC::library_init↵::library_init()`.

#### 5.1.4.51 `_ex_6`

```
const ex GiNaC::_ex_6 = _ex_6
```

Referenced by `GiNaC::su3d::eval_indexed()`, `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library↵init()`.

#### 5.1.4.52 `_num_5_p`

```
const numeric * GiNaC::_num_5_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, and `GiNaC::library_init↵::library_init()`.

#### 5.1.4.53 `_ex_5`

```
const ex GiNaC::_ex_5 = _ex_5
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.54 `_num_4_p`

```
const numeric * GiNaC::_num_4_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, and `GiNaC::library_init::library_init()`.

#### 5.1.4.55 `_ex_4`

```
const ex GiNaC::_ex_4 = _ex_4
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.56 `_num_3_p`

```
const numeric * GiNaC::_num_3_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, and `GiNaC::library_init::library_init()`.

#### 5.1.4.57 `_ex_3`

```
const ex GiNaC::_ex_3 = _ex_3
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.58 `_num_2_p`

```
const numeric * GiNaC::_num_2_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::library_init::library_init()`, and `tgamma_eval()`.

#### 5.1.4.59 `_ex_2`

```
const ex GiNaC::_ex_2 = _ex_2
```

Referenced by `GiNaC::spinmetric::contract_with()`, `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

5.1.4.60 `_num_1_p`

```
const numeric * GiNaC::_num_1_p
```

Referenced by `acos_conjugate()`, `asin_conjugate()`, `asinh_conjugate()`, `atan_conjugate()`, `atanh_conjugate()`, `beta_eval()`, `binomial()`, `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::add::do_print_csrc()`, `doublefactorial()`, `GiNaC::mul::eval()`, `GiNaC::power::eval()`, `GiNaC::library_init::library_init()`, `operator--()`, `GiNaC::add::print_add()`, `GiNaC::mul::print_overall_coeff()`, `psi1_eval()`, and `psi2_eval()`.

5.1.4.61 `_ex_1`

```
const ex GiNaC::_ex_1 = _ex_1
```

Referenced by `acos_eval()`, `acosh_deriv()`, `acosh_eval()`, `asin_eval()`, `atan2_deriv()`, `atan_deriv()`, `atan_eval()`, `atan_series()`, `atanh_deriv()`, `atanh_eval()`, `atanh_series()`, `cos_eval()`, `GiNaC::power::derivative()`, `GiNaC::add::do_print_csrc()`, `GiNaC::power::do_print_csrc()`, `GiNaC::mul::do_print_csrc()`, `GiNaC::power::do_print_csrc_cl_N()`, `EllipticE_eval()`, `EllipticE_series()`, `EllipticK_series()`, `GiNaC::power::eval()`, `GiNaC::minkmetric::eval_indexed()`, `GiNaC::spinmetric::eval_indexed()`, `exminus()`, `exp_eval()`, `exp_power()`, `GiNaC::power::expand()`, `frac_cancel()`, `H_deriv()`, `H_eval()`, `lgamma_eval()`, `Li2_eval()`, `Li_eval()`, `GiNaC::library_init::library_init()`, `log_deriv()`, `log_expand()`, `log_series()`, `operator--()`, `operator/()`, `operator/=(())`, `psi1_series()`, `psi2_eval()`, `psi2_series()`, `replace_with_symbol()`, `sin_eval()`, `tan_eval()`, `tanh_eval()`, `GiNaC::power::to_polynomial()`, `GiNaC::ex::unit()`, `GiNaC::ex::unitcontprim()`, and `GiNaC::library_init::~~library_init()`.

5.1.4.62 `_num_1_2_p`

```
const numeric * GiNaC::_num_1_2_p
```

Referenced by `GiNaC::library_init::library_init()`.

5.1.4.63 `_ex_1_2`

```
const ex GiNaC::_ex_1_2 = _ex_1_2
```

Referenced by `acos_deriv()`, `acos_eval()`, `acosh_deriv()`, `asin_deriv()`, `asin_eval()`, `asinh_deriv()`, `atan2_eval()`, `atan_series()`, `atanh_series()`, `cos_eval()`, `cosh_eval()`, `GiNaC::su3f::eval_indexed()`, `GiNaC::su3d::eval_indexed()`, `Li2_eval()`, `GiNaC::library_init::library_init()`, `log_eval()`, `sin_eval()`, `sinh_eval()`, `tan_eval()`, `tanh_eval()`, `zeta1_eval()`, and `GiNaC::library_init::~~library_init()`.

5.1.4.64 `_num_1_3_p`

```
const numeric * GiNaC::_num_1_3_p
```

Referenced by `GiNaC::library_init::library_init()`.

5.1.4.65 `_ex_1_3`

```
const ex GiNaC::_ex_1_3 = _ex_1_3
```

Referenced by `cos_eval()`, `GiNaC::su3d::eval_indexed()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `GiNaC::library_init::~library_init()`.

5.1.4.66 `_num_1_4_p`

```
const numeric * GiNaC::_num_1_4_p
```

Referenced by `GiNaC::library_init::library_init()`.

5.1.4.67 `_ex_1_4`

```
const ex GiNaC::_ex_1_4 = _ex_1_4
```

Referenced by `atan2_eval()`, `atan_eval()`, `cos_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `GiNaC::library_init::~library_init()`.

5.1.4.68 `_num0_p`

```
const numeric * GiNaC::_num0_p
```

Referenced by `GiNaC::numeric::add_dyn()`, `atan()`, `binomial()`, `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, `cos_eval()`, `divide_in_z()`, `GiNaC::power::eval()`, `exp_eval()`, `GiNaC::mul::expand()`, `fibonacci()`, `GiNaC::numeric::has()`, `GiNaC::add::integer_content()`, `iquo()`, `irem()`, `isqrt()`, `Li2()`, `GiNaC::library_init::library_init()`, `mod()`, `GiNaC::relational::operator safe_bool()`, `GiNaC::numeric::power()`, `GiNaC::numeric::power_dyn()`, `sin_eval()`, `smod()`, `GiNaC::numeric::sub_dyn()`, and `tan_eval()`.

5.1.4.69 `_ex0`

```
const ex GiNaC::_ex0 = _ex0
```

Referenced by `acos_eval()`, `acosh_eval()`, `GiNaC::add::add()`, `GiNaC::pseries::add_series()`, `asinh_eval()`, `atan2_eval()`, `atan_eval()`, `atan_series()`, `atanh_eval()`, `atanh_series()`, `beta_eval()`, `binomial_sym()`, `GiNaC::pseries::coeff()`, `GiNaC::add::coeff()`, `GiNaC::mul::coeff()`, `GiNaC::power::coeff()`, `GiNaC::ncmul::coeff()`, `GiNaC::numeric::coeff()`, `GiNaC::basic::coeff()`, `GiNaC::basic::collect()`, `color_trace()`, `GiNaC::ex::content()`, `cos_eval()`, `csgn_series()`, `GiNaC::constant::derivative()`, `GiNaC::idx::derivative()`, `GiNaC::symbol::derivative()`, `GiNaC::indexed::derivative()`, `GiNaC::basic::derivative()`, `GiNaC::matrix::determinant()`, `GiNaC::matrix::determinant_minor()`, `divide()`, `divide_in_z()`, `GiNaC::matrix::division_free_elimination()`, `EllipticE_eval()`, `EllipticE_series()`, `EllipticK_eval()`, `EllipticK_series()`, `eta_eval()`, `eta_evalf()`, `eta_series()`, `GiNaC::integral::eval()`, `GiNaC::mul::eval()`, `GiNaC::power::eval()`, `GiNaC::indexed::eval()`, `GiNaC::function::eval()`, `GiNaC::tensdelta::eval_indexed()`, `GiNaC::minkmetric::eval_indexed()`, `GiNaC::su3f::eval_indexed()`, `GiNaC::su3d::eval_indexed()`, `GiNaC::spinmetric::eval_indexed()`, `GiNaC::tensepsilon::eval_indexed()`, `GiNaC::indexed::expand()`, `find_common_factor()`, `GiNaC::matrix::fraction_free_elimination()`, `G2_eval()`, `G2_evalf()`, `G3_eval()`, `G3_evalf()`, `GiNaC::matrix::gauss_elimination()`, `gcd()`, `H_deriv()`, `H_eval()`, `GiNaC::ex::is_zero()`, `Li2_eval()`, `Li2_series()`, `Li_deriv()`, `Li_eval()`, `Li_evalf()`, `GiNaC::library_init::library_init()`, `log_eval()`, `log_series()`, `GiNaC::matrix::markowitz_elimination()`, `GiNaC::pseries::mul_series()`, `Order_eval()`, `GiNaC::pseries::power_const()`, `prem()`, `GiNaC::ex::primpart()`, `rem()`, `S_deriv()`, `S_eval()`, `GiNaC::symbol::series()`, `GiNaC::ex::series()`, `GiNaC::basic::series()`, `simplify_indexed_product()`, `sin_eval()`, `sinh_eval()`, `sprem()`, `sqrfree()`, `sr_gcd()`, `step_series()`, `tan_eval()`, `tanh_eval()`, `GiNaC::ex::unitcontprim()`, `zeta1_deriv()`, `zeta1_eval()`, `zeta2_deriv()`, and `GiNaC::library_init::~library_init()`.

**5.1.4.70** `_num1_4_p`

```
const numeric * GiNaC::_num1_4_p
```

Referenced by `GiNaC::library_init::library_init()`.

**5.1.4.71** `_ex1_4`

```
const ex GiNaC::_ex1_4 = _ex1_4
```

Referenced by `atan2_eval()`, `atan_eval()`, `cos_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `GiNaC::library_init::~~library_init()`.

**5.1.4.72** `_num1_3_p`

```
const numeric * GiNaC::_num1_3_p
```

Referenced by `GiNaC::library_init::library_init()`.

**5.1.4.73** `_ex1_3`

```
const ex GiNaC::_ex1_3 = _ex1_3
```

Referenced by `acos_eval()`, `cos_eval()`, `GiNaC::su3d::eval_indexed()`, `GiNaC::library_init::library_init()`, `sin_eval()`, `tan_eval()`, and `GiNaC::library_init::~~library_init()`.

**5.1.4.74** `_num1_2_p`

```
const numeric * GiNaC::_num1_2_p
```

Referenced by `GiNaC::library_init::library_init()`, `psi2_eval()`, and `tgamma_eval()`.

**5.1.4.75** `_ex1_2`

```
const ex GiNaC::_ex1_2 = _ex1_2
```

Referenced by `acos_eval()`, `asin_eval()`, `atan2_eval()`, `atan_series()`, `atanh_series()`, `cos_eval()`, `GiNaC::power::do_print_dflt()`, `GiNaC::power::do_print_latex()`, `GiNaC::su3f::eval_indexed()`, `GiNaC::su3d::eval_indexed()`, `GiNaC::clifford::get_metric()`, `Li2_eval()`, `GiNaC::library_init::library_init()`, `log_eval()`, `psi1_eval()`, `psi2_eval()`, `sin_eval()`, `sqrt()`, and `GiNaC::library_init::~~library_init()`.

5.1.4.76 `_num1_p`

```
const numeric * GiNaC::_num1_p
```

Referenced by `acos_conjugate()`, `acosh_conjugate()`, `asin_conjugate()`, `asinh_conjugate()`, `atan()`, `atan_conjugate()`, `atanh_conjugate()`, `bernoulli()`, `binomial()`, `GiNaC::add::combine_pair_with_coeff_to_pair()`, `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, `GiNaC::numeric::denom()`, `GiNaC::numeric::div_dyn()`, `divide_in_z()`, `GiNaC::add::do_print_csrc()`, `doublefactorial()`, `GiNaC::mul::eval()`, `GiNaC::power::eval()`, `exp_eval()`, `GiNaC::power::expand_add()`, `frac_cancel()`, `gcd()`, `GiNaC::mul::info()`, `GiNaC::add::integer_content()`, `GiNaC::basic::integer_content()`, `lcm_of_coefficients_denominators()`, `lcmcoeff()`, `Li2_conjugate()`, `GiNaC::library_init::library_init()`, `GiNaC::basic::max_coefficient()`, `GiNaC::numeric::mul_dyn()`, `multiply_lcm()`, `operator++()`, `GiNaC::matrix::pow()`, `GiNaC::numeric::power()`, `GiNaC::numeric::power_dyn()`, `GiNaC::add::print_add()`, `GiNaC::mul::print_overall_coeff()`, `psi2_eval()`, `GiNaC::add::recombine_pair_to_ex()`, `tgamma_eval()`, and `zeta1_eval()`.

5.1.4.77 `_ex1`

```
const ex GiNaC::_ex1 = _ex1
```

Referenced by `acos_eval()`, `acosh_deriv()`, `acosh_eval()`, `GiNaC::pseries::add_series()`, `asin_eval()`, `asinh_deriv()`, `atan_deriv()`, `atan_eval()`, `atan_series()`, `atanh_deriv()`, `atanh_eval()`, `atanh_series()`, `beta_eval()`, `binomial_sym()`, `GiNaC::power::coeff()`, `GiNaC::ncmul::coeff()`, `GiNaC::basic::coeff()`, `GiNaC::basic::collect()`, `color_trace()`, `GiNaC::add::combine_ex_with_coeff_to_pair()`, `GiNaC::mul::combine_ex_with_coeff_to_pair()`, `GiNaC::add::combine_pair_with_coeff_to_pair()`, `GiNaC::mul::combine_pair_with_coeff_to_pair()`, `GiNaC::matrix::contract_with()`, `GiNaC::su3t::contract_with()`, `GiNaC::su3f::contract_with()`, `GiNaC::su3d::contract_with()`, `GiNaC::spinmetric::contract_with()`, `GiNaC::tensepsilon::contract_with()`, `cos_eval()`, `cosh_eval()`, `GiNaC::mul::default_overall_coeff()`, `GiNaC::symbol::derivative()`, `GiNaC::mul::derivative()`, `GiNaC::power::derivative()`, `GiNaC::matrix::determinant_minor()`, `divide()`, `divide_in_z()`, `GiNaC::add::do_print_csrc()`, `GiNaC::mul::do_print_csrc()`, `EllipticE_eval()`, `EllipticE_series()`, `EllipticK_series()`, `GiNaC::mul::eval()`, `GiNaC::power::eval()`, `GiNaC::ncmul::eval()`, `GiNaC::tensdelta::eval_indexed()`, `GiNaC::minkmetric::eval_indexed()`, `GiNaC::spinmetric::eval_indexed()`, `GiNaC::mul::evalm()`, `exp_eval()`, `GiNaC::mul::expair_needs_further_processing()`, `GiNaC::ncmul::expand()`, `GiNaC::power::expand()`, `GiNaC::mul::expand()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_add_2()`, `GiNaC::power::expand_mul()`, `find_common_factor()`, `frac_cancel()`, `GiNaC::matrix::fraction_free_elimination()`, `G2_eval()`, `G2_evalf()`, `G3_eval()`, `G3_evalf()`, `gcd()`, `gcd_pf_pow()`, `gcd_pf_pow_pow()`, `H_deriv()`, `H_eval()`, `hold_ncmul()`, `GiNaC::power::imag_part()`, `GiNaC::matrix::inverse()`, `lgamma_series()`, `Li2_deriv()`, `Li2_eval()`, `Li2_series()`, `Li_eval()`, `Li_evalf()`, `Li_series()`, `GiNaC::library_init::library_init()`, `log_eval()`, `log_expand()`, `log_series()`, `GiNaC::mul::mul()`, `GiNaC::pseries::mul_series()`, `GiNaC::symbol::normal()`, `GiNaC::pseries::normal()`, `GiNaC::power::normal()`, `GiNaC::basic::normal()`, `operator++()`, `Order_eval()`, `Order_series()`, `GiNaC::matrix::pow()`, `GiNaC::pseries::power_const()`, `prem()`, `GiNaC::ex::primpart()`, `GiNaC::pseries::print_series()`, `psi1_deriv()`, `psi1_series()`, `psi2_deriv()`, `psi2_eval()`, `psi2_series()`, `quo()`, `GiNaC::power::real_part()`, `GiNaC::mul::recombine_pair_to_ex()`, `GiNaC::tensor::replace_contr_index()`, `S_series()`, `GiNaC::symbol::series()`, `GiNaC::pseries::series()`, `GiNaC::add::series()`, `GiNaC::mul::series()`, `GiNaC::integral::series()`, `GiNaC::power::series()`, `GiNaC::basic::series()`, `simplify_indexed_product()`, `sin_eval()`, `sinh_eval()`, `GiNaC::add::split_ex_to_pair()`, `GiNaC::mul::split_ex_to_pair()`, `sprem()`, `sqrfree_parfrac()`, `sqrfree_yun()`, `sr_gcd()`, `tan_deriv()`, `tan_eval()`, `tanh_deriv()`, `tanh_eval()`, `tgamma_series()`, `GiNaC::expairseq::to_polynomial()`, `GiNaC::expairseq::to_rational()`, `GiNaC::ex::unit()`, `unit_matrix()`, `GiNaC::ex::unitcontprim()`, `zeta1_deriv()`, `zeta2_deriv()`, and `GiNaC::library_init::~~library_init()`.

5.1.4.78 `_num2_p`

```
const numeric * GiNaC::_num2_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, `exp_eval()`, `GiNaC::power::expand_add_2()`, `Li2_series()`, `GiNaC::library_init::library_init()`, `GiNaC::matrix::pow()`, `psi1_eval()`, `psi2_eval()`, `tgamma_eval()`, and `zeta1_eval()`.

5.1.4.79 `_ex2`

```
const ex GiNaC::_ex2 = _ex2
```

Referenced by `acos_deriv()`, `asin_deriv()`, `asinh_deriv()`, `atan2_deriv()`, `atan_deriv()`, `atanh_deriv()`, `GiNaC::spinmetric::contract_with()`, `cos_eval()`, `cosh_eval()`, `csgn_power()`, `epsilon_tensor()`, `exp_eval()`, `GiNaC::power::expand_add_2()`, `Li2_eval()`, `Li2_series()`, `Li_eval()`, `GiNaC::library_init::library_init()`, `product_to_exvector()`, `psi1_eval()`, `simplify_indexed()`, `sin_eval()`, `sinh_eval()`, `tan_deriv()`, `tan_eval()`, `tanh_deriv()`, `tanh_eval()`, and `GiNaC::library_init::~~library_init()`.

5.1.4.80 `_num3_p`

```
const numeric * GiNaC::_num3_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, `exp_eval()`, and `GiNaC::library_init::library_init()`.

5.1.4.81 `_ex3`

```
const ex GiNaC::_ex3 = _ex3
```

Referenced by `color_trace()`, `cos_eval()`, `epsilon_tensor()`, `GiNaC::su3f::eval_indexed()`, `GiNaC::su3d::eval_indexed()`, `GiNaC::library_init::library_init()`, `sin_eval()`, `tan_eval()`, and `GiNaC::library_init::~~library_init()`.

5.1.4.82 `_num4_p`

```
const numeric * GiNaC::_num4_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, `exp_eval()`, and `GiNaC::library_init::library_init()`.

5.1.4.83 `_ex4`

```
const ex GiNaC::_ex4 = _ex4
```

Referenced by `GiNaC::library_init::library_init()`, `lorentz_eps()`, and `GiNaC::library_init::~~library_init()`.



#### 5.1.4.84 \_num5\_p

```
const numeric * GiNaC::_num5_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, `cos_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `tan_eval()`.

#### 5.1.4.85 \_ex5

```
const ex GiNaC::_ex5 = _ex5
```

Referenced by `cos_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.86 \_num6\_p

```
const numeric * GiNaC::_num6_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, `GiNaC::library_init::library_init()`, and `sin_eval()`.

#### 5.1.4.87 \_ex6

```
const ex GiNaC::_ex6 = _ex6
```

Referenced by `cos_eval()`, `Li2_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.88 \_num7\_p

```
const numeric * GiNaC::_num7_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, and `GiNaC::library_init::library_init()`.

#### 5.1.4.89 \_ex7

```
const ex GiNaC::_ex7 = _ex7
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.90 `_num8_p`

```
const numeric * GiNaC::_num8_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, and `GiNaC::library_init::library_init()`.

#### 5.1.4.91 `_ex8`

```
const ex GiNaC::_ex8 = _ex8
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.92 `_num9_p`

```
const numeric * GiNaC::_num9_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, and `GiNaC::library_init::library_init()`.

#### 5.1.4.93 `_ex9`

```
const ex GiNaC::_ex9 = _ex9
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.94 `_num10_p`

```
const numeric * GiNaC::_num10_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, `cos_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `tan_eval()`.

#### 5.1.4.95 `_ex10`

```
const ex GiNaC::_ex10 = _ex10
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.96 \_num11\_p

```
const numeric * GiNaC::_num11_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, and `GiNaC::library_init::library_init()`.

#### 5.1.4.97 \_ex11

```
const ex GiNaC::_ex11 = _ex11
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.98 \_num12\_p

```
const numeric * GiNaC::_num12_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, `cos_eval()`, and `GiNaC::library_init::library_init()`.

#### 5.1.4.99 \_ex12

```
const ex GiNaC::_ex12 = _ex12
```

Referenced by `Li2_eval()`, `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.100 \_num15\_p

```
const numeric * GiNaC::_num15_p
```

Referenced by `cos_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `tan_eval()`.

#### 5.1.4.101 \_ex15

```
const ex GiNaC::_ex15 = _ex15
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.102 \_num18\_p

```
const numeric * GiNaC::_num18_p
```

Referenced by `GiNaC::library_init::library_init()`, and `sin_eval()`.

#### 5.1.4.103 \_ex18

```
const ex GiNaC::_ex18 = _ex18
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.104 \_num20\_p

```
const numeric * GiNaC::_num20_p
```

Referenced by `cos_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `tan_eval()`.

#### 5.1.4.105 \_ex20

```
const ex GiNaC::_ex20 = _ex20
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.106 \_num24\_p

```
const numeric * GiNaC::_num24_p
```

Referenced by `cos_eval()`, and `GiNaC::library_init::library_init()`.

#### 5.1.4.107 \_ex24

```
const ex GiNaC::_ex24 = _ex24
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.108 \_num25\_p

```
const numeric * GiNaC::_num25_p
```

Referenced by `cos_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `tan_eval()`.

#### 5.1.4.109 \_ex25

```
const ex GiNaC::_ex25 = _ex25
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.110 \_num30\_p

```
const numeric * GiNaC::_num30_p
```

Referenced by `cos_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `tan_eval()`.

#### 5.1.4.111 \_ex30

```
const ex GiNaC::_ex30 = _ex30
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

#### 5.1.4.112 \_num48\_p

```
const numeric * GiNaC::_num48_p
```

Referenced by `GiNaC::library_init::library_init()`.

#### 5.1.4.113 \_ex48

```
const ex GiNaC::_ex48 = _ex48
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

5.1.4.114 `_num60_p`

```
const numeric * GiNaC::_num60_p
```

Referenced by `cos_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, and `tan_eval()`.

5.1.4.115 `_ex60`

```
const ex GiNaC::_ex60 = _ex60
```

Referenced by `cos_eval()`, `GiNaC::library_init::library_init()`, `sin_eval()`, `tan_eval()`, and `GiNaC::library_init::~library_init()`.

5.1.4.116 `_num120_p`

```
const numeric * GiNaC::_num120_p
```

Referenced by `cos_eval()`, `GiNaC::library_init::library_init()`, and `sin_eval()`.

5.1.4.117 `_ex120`

```
const ex GiNaC::_ex120 = _ex120
```

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

## 5.2 GiNaC::internal Namespace Reference

### Classes

- struct `_iter_rep`

## 5.3 std Namespace Reference

### Classes

- struct `equal_to< GiNaC::ex >`  
*Specialization of `std::equal_to()` for `ex` objects.*
- struct `hash< GiNaC::ex >`  
*Specialization of `std::hash()` for `ex` objects.*
- struct `less< GiNaC::ptr< T > >`  
*Specialization of `std::less` for `ptr< T>` to enable ordering of `ptr< T>` objects (e.g.*

## Functions

- `template<>`  
`void swap (GiNaC::ex &a, GiNaC::ex &b)`  
*Specialization of `std::swap()` for ex objects.*

### 5.3.1 Function Documentation

#### 5.3.1.1 swap()

```
template<>
void std::swap (
    GiNaC::ex & a,
    GiNaC::ex & b ) [inline]
```

Specialization of `std::swap()` for ex objects.

References `GiNaC::ex::swap()`.

Referenced by `GiNaC::su3f::eval_indexed()`, `GiNaC::su3d::eval_indexed()`, `GiNaC::matrix::markowitz_elimination()`, and `GiNaC::permutation_sign()`.





## Chapter 6

# Class Documentation

### 6.1 GiNaC::internal::\_iter\_rep Struct Reference

```
#include <ex.h>
```

#### Public Member Functions

- [\\_iter\\_rep](#) (const [ex](#) &[e\\_](#), [size\\_t](#) [i\\_](#), [size\\_t](#) [i\\_end\\_](#))
- bool [operator==](#) (const [\\_iter\\_rep](#) &[other](#)) const noexcept
- bool [operator!=](#) (const [\\_iter\\_rep](#) &[other](#)) const noexcept

#### Public Attributes

- [ex](#) [e](#)
- [size\\_t](#) [i](#)
- [size\\_t](#) [i\\_end](#)

#### 6.1.1 Constructor & Destructor Documentation

##### 6.1.1.1 \_iter\_rep()

```
GiNaC::internal::_iter_rep::_iter_rep (
    const ex & e\_,
    size\_t i\_,
    size\_t i\_end\_ ) [inline]
```

#### 6.1.2 Member Function Documentation

#### 6.1.2.1 operator==()

```
bool GiNaC::internal::_iter_rep::operator== (
    const \_iter\_rep & other ) const [inline], [noexcept]
```

References `GiNaC::are_ex_trivially_equal()`, `e`, and `i`.

#### 6.1.2.2 operator!=()

```
bool GiNaC::internal::_iter_rep::operator!= (
    const \_iter\_rep & other ) const [inline], [noexcept]
```

### 6.1.3 Member Data Documentation

#### 6.1.3.1 e

```
ex GiNaC::internal::_iter_rep::e
```

Referenced by `GiNaC::const_postorder_iterator::descend()`, `GiNaC::const_preorder_iterator::increment()`, and `operator==()`.

#### 6.1.3.2 i

```
size_t GiNaC::internal::_iter_rep::i
```

Referenced by `GiNaC::const_postorder_iterator::descend()`, `GiNaC::const_preorder_iterator::increment()`, and `operator==()`.

#### 6.1.3.3 i\_end

```
size_t GiNaC::internal::_iter_rep::i_end
```

Referenced by `GiNaC::const_preorder_iterator::increment()`.

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.2 GiNaC::\_numeric\_digits Class Reference

This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.

```
#include <numeric.h>
```

### Public Member Functions

- [\\_numeric\\_digits](#) ()  
*\_numeric\_digits* default ctor, checking for singleton invariance.
- [\\_numeric\\_digits & operator=](#) (long prec)  
*Assign a native long to global Digits object.*
- [operator long](#) ()  
*Convert global Digits object to native type long.*
- void [print](#) (std::ostream &os) const  
*Append global Digits object to ostream.*
- void [add\\_callback](#) ([digits\\_changed\\_callback](#) callback)  
*Add a new callback function.*

### Private Attributes

- long [digits](#)  
*Number of decimal digits.*
- std::vector< [digits\\_changed\\_callback](#) > [callbacklist](#)

### Static Private Attributes

- static bool [too\\_late](#) = false  
*Already one object present.*

#### 6.2.1 Detailed Description

This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.

We need an object rather than a dumb basic type since as a side-effect we let it change `cl_default_float_format` when it gets changed. The only other meaningful thing to do with it is converting it to an unsigned, for temporarily storing its value e.g. The user must not create an own working object of this class! Since C++ forces us to make the class definition visible in order to use an object we put in a flag which prevents other objects of that class to be created.

#### 6.2.2 Constructor & Destructor Documentation

### 6.2.2.1 `_numeric_digits()`

```
GiNaC::_numeric_digits::_numeric_digits ( )
```

`_numeric_digits` default ctor, checking for singleton invariance.

References `too_late`.

## 6.2.3 Member Function Documentation

### 6.2.3.1 `operator=()`

```
_numeric_digits & GiNaC::_numeric_digits::operator= (
    long prec )
```

Assign a native long to global Digits object.

References `callbacklist`, and `digits`.

### 6.2.3.2 `operator long()`

```
GiNaC::_numeric_digits::operator long ( )
```

Convert global Digits object to native type long.

### 6.2.3.3 `print()`

```
void GiNaC::_numeric_digits::print (
    std::ostream & os ) const
```

Append global Digits object to ostream.

References `digits`.

Referenced by `GiNaC::operator<<()`.

### 6.2.3.4 `add_callback()`

```
void GiNaC::_numeric_digits::add_callback (
    digits_changed_callback callback )
```

Add a new callback function.

References `callbacklist`.

## 6.2.4 Member Data Documentation

### 6.2.4.1 digits

```
long GiNaC::_numeric_digits::digits [private]
```

Number of decimal digits.

Referenced by operator=(), and print().

### 6.2.4.2 too\_late

```
bool GiNaC::_numeric_digits::too_late = false [static], [private]
```

Already one object present.

Referenced by \_numeric\_digits().

### 6.2.4.3 callbacklist

```
std::vector<digits_changed_callback> GiNaC::_numeric_digits::callbacklist [private]
```

Referenced by add\_callback(), and operator=().

The documentation for this class was generated from the following files:

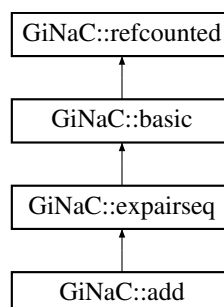
- [numeric.h](#)
- [numeric.cpp](#)

## 6.3 GiNaC::add Class Reference

Sum of expressions.

```
#include <add.h>
```

Inheritance diagram for GiNaC::add:



## Public Member Functions

- `add` (const `ex` &lh, const `ex` &rh)
- `add` (const `exvector` &v)
- `add` (const `epvector` &v)
- `add` (const `epvector` &v, const `ex` &oc)
- `add` (`epvector` &&v)
- `add` (`epvector` &&v, const `ex` &oc)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*
- int `ldegree` (const `ex` &s) const override  
*Return degree of lowest power in object s.*
- `ex coeff` (const `ex` &s, int n=1) const override  
*Return coefficient of degree n in object s.*
- `ex eval` () const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalm` () const override  
*Evaluate sums, products and integer powers of matrices.*
- `ex series` (const `relational` &r, int order, unsigned options=0) const override  
*Implementation of `ex::series()` for sums.*
- `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const override  
*Implementation of `ex::normal()` for a sum.*
- `numeric integer_content` () const override
- `ex smod` (const `numeric` &xi) const override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient` () const override  
*Implementation `ex::max_coefficient()`.*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- `ex eval_ncmul` (const `exvector` &v) const override

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for a sum.*
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex thisexpairseq` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false) const override
- `ex thisexpairseq` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false) const override
- `expair split_ex_to_pair` (const `ex` &e) const override
- `expair combine_ex_with_coeff_to_pair` (const `ex` &e, const `ex` &c) const override
- `expair combine_pair_with_coeff_to_pair` (const `expair` &p, const `ex` &c) const override
- `ex recombine_pair_to_ex` (const `expair` &p) const override

- `ex expand` (unsigned `options=0`) const override  
*Expand expression, i.e.*
- void `print_add` (const `print_context` &`c`, const char \*`openbrace`, const char \*`closebrace`, const char \*`mul_sym`, unsigned level) const
- void `do_print` (const `print_context` &`c`, unsigned level) const
- void `do_print_latex` (const `print_latex` &`c`, unsigned level) const
- void `do_print_csrc` (const `print_csrc` &`c`, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &`c`, unsigned level) const

## Friends

- class `mul`
- class `power`

## Additional Inherited Members

### 6.3.1 Detailed Description

Sum of expressions.

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 `add()` [1/6]

```
GiNaC::add::add (
    const ex & lh,
    const ex & rh )
```

References `GiNaC::_ex0`, `GiNaC::expairseq::construct_from_2_ex()`, `GINAC_ASSERT`, `GiNaC::expairseq::is_↔canonical()`, and `GiNaC::expairseq::overall_coeff`.

Referenced by `conjugate()`.

#### 6.3.2.2 `add()` [2/6]

```
GiNaC::add::add (
    const exvector & v )
```

References `GiNaC::_ex0`, `GiNaC::expairseq::construct_from_exvector()`, `GINAC_ASSERT`, `GiNaC::expairseq::is_↔_canonical()`, and `GiNaC::expairseq::overall_coeff`.

**6.3.2.3 add()** [3/6]

```
GiNaC::add::add (
    const epvector & v )
```

References `GiNaC::_ex0`, `GiNaC::expairseq::construct_from_epvector()`, `GINAC_ASSERT`, `GiNaC::expairseq::is_↵_canonical()`, and `GiNaC::expairseq::overall_coeff`.

**6.3.2.4 add()** [4/6]

```
GiNaC::add::add (
    const epvector & v,
    const ex & oc )
```

References `GiNaC::expairseq::construct_from_epvector()`, `GINAC_ASSERT`, `GiNaC::expairseq::is_canonical()`, and `GiNaC::expairseq::overall_coeff`.

**6.3.2.5 add()** [5/6]

```
GiNaC::add::add (
    epvector && v )
```

References `GiNaC::_ex0`, `GiNaC::expairseq::construct_from_epvector()`, `GINAC_ASSERT`, `GiNaC::expairseq::is_↵_canonical()`, and `GiNaC::expairseq::overall_coeff`.

**6.3.2.6 add()** [6/6]

```
GiNaC::add::add (
    epvector && v,
    const ex & oc )
```

References `GiNaC::expairseq::construct_from_epvector()`, `GINAC_ASSERT`, `GiNaC::expairseq::is_canonical()`, and `GiNaC::expairseq::overall_coeff`.

**6.3.3 Member Function Documentation**



## 6.3.3.1 precedence()

```
unsigned GiNaC::add::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by `do_print_csrc()`, and `print_add()`.

## 6.3.3.2 info()

```
bool GiNaC::add::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References `GiNaC::info_flags::cinteger`, `GiNaC::info_flags::cinteger_polynomial`, `GiNaC::info_flags::crational`, `GiNaC::info_flags::crational_polynomial`, `GiNaC::info_flags::even`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `GiNaC::info_flags::integer_polynomial`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::nonnegative`, `GiNaC::info_flags::nonnegint`, `GiNaC::expairseq::overall_coeff`, `GiNaC::info_flags::polynomial`, `GiNaC::info_flags::posint`, `GiNaC::info_flags::positive`, `GiNaC::info_flags::rational`, `GiNaC::info_flags::rational_function`, `GiNaC::info_flags::rational_polynomial`, `GiNaC::info_flags::real`, `recombine_pair_to_ex()`, and `GiNaC::expairseq::seq`.

## 6.3.3.3 is\_polynomial()

```
bool GiNaC::add::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References `GiNaC::expairseq::seq`.

#### 6.3.3.4 degree()

```
int GiNaC::add::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

#### 6.3.3.5 ldegree()

```
int GiNaC::add::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

#### 6.3.3.6 coeff()

```
ex GiNaC::add::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::clifford\\_max\\_label\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::dirac\\_↔ONE\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), n, [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

Referenced by [print\\_add\(\)](#), and [smod\(\)](#).

#### 6.3.3.7 eval()

```
ex GiNaC::add::eval ( ) const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following x stands for a symbolic variables of type [ex](#) and c stands for such an expression that contain a plain number.

- $+(;c) \rightarrow c$
- $+(x;0) \rightarrow x$

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [unlikely](#).

### 6.3.3.8 evalm()

```
ex GiNaC::add::evalm ( ) const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::matrix::add\(\)](#), [GiNaC::ex::evalm\(\)](#), [m](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 6.3.3.9 series()

```
ex GiNaC::add::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for sums.

This performs series addition when adding pseries objects.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::op\(\)](#), [options](#), [order](#), [GiNaC::expairseq::overall\\_coeff](#), [r](#), [GiNaC::expairseq::seq](#), and [GiNaC::ex::series\(\)](#).

### 6.3.3.10 normal()

```
ex GiNaC::add::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for a sum.

It expands terms and performs fractional addition.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [expand\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::gcd\(\)](#), [GINAC\\_ASSERT](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.3.3.11 integer\_content()

```
numeric GiNaC::add::integer_content ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), [c](#), [GiNaC::denom\(\)](#), [GiNaC::gcd\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::lcm\(\)](#), [GiNaC::numer\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

### 6.3.3.12 smod()

```
ex GiNaC::add::smod (
    const numeric & xi ) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

#### Parameters

$xi$	modulus
------	---------

#### Returns

mapped polynomial

#### See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::expairseq::seq](#), and [GiNaC::smod\(\)](#).

### 6.3.3.13 max\_coefficient()

```
numeric GiNaC::add::max_coefficient ( ) const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

#### See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

**6.3.3.14 conjugate()**

```
ex GiNaC::add::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [add\(\)](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

**6.3.3.15 real\_part()**

```
ex GiNaC::add::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

**6.3.3.16 imag\_part()**

```
ex GiNaC::add::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::real](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

**6.3.3.17 get\_free\_indices()**

```
exvector GiNaC::add::get_free_indices ( ) const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::indices\\_consistent\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

**6.3.3.18 eval\_ncmul()**

```
ex GiNaC::add::eval_ncmul (
    const exvector & v ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

#### 6.3.3.19 derivative()

```
ex GiNaC::add::derivative (
    const symbol & y ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a sum.

It differentiates each term.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

#### 6.3.3.20 return\_type()

```
unsigned GiNaC::add::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), and [GiNaC::expairseq::seq](#).

#### 6.3.3.21 return\_type\_tinfo()

```
return_type_t GiNaC::add::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

#### 6.3.3.22 thisexpairseq() [1/2]

```
ex GiNaC::add::thisexpairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

**6.3.3.23 thisexpairseq()** [2/2]

```
ex GiNaC::add::thisexpairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

**6.3.3.24 split\_ex\_to\_pair()**

```
expair GiNaC::add::split_ex_to_pair (
    const ex & e ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

Referenced by [evalm\(\)](#), [imag\\_part\(\)](#), and [real\\_part\(\)](#).

**6.3.3.25 combine\_ex\_with\_coeff\_to\_pair()**

```
expair GiNaC::add::combine_ex_with_coeff_to_pair (
    const ex & e,
    const ex & c ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [c](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::ex::is\\_equal\(\)](#), [likely](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.3.26 combine\_pair\_with\_coeff\_to\_pair()**

```
expair GiNaC::add::combine_pair_with_coeff_to_pair (
    const expair & p,
    const ex & c ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [c](#), [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), and [GiNaC::expair::rest](#).

Referenced by [GiNaC::mul::eval\(\)](#), and [GiNaC::power::expand\\_add\\_2\(\)](#).

### 6.3.3.27 recombine\_pair\_to\_ex()

```
ex GiNaC::add::recombine_pair_to_ex (
    const expair & p ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_num1\\_p](#), [GiNaC::expair::coeff](#), and [GiNaC::expair::rest](#).

Referenced by [eval\(\)](#), [evalm\(\)](#), [GiNaC::power::expand\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [normal\(\)](#), and [real\\_part\(\)](#).

### 6.3.3.28 expand()

```
ex GiNaC::add::expand (
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [normal\(\)](#).

### 6.3.3.29 print\_add()

```
void GiNaC::add::print_add (
    const print\_context & c,
    const char * openbrace,
    const char * closebrace,
    const char * mul_sym,
    unsigned level ) const [protected]
```

References [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [c](#), [coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).

### 6.3.3.30 do\_print()

```
void GiNaC::add::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_add\(\)](#).



**6.3.3.31 do\_print\_latex()**

```
void GiNaC::add::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_add\(\)](#).

**6.3.3.32 do\_print\_csrc()**

```
void GiNaC::add::do_print_csrc (
    const print\_csrc & c,
    unsigned level ) const [protected]
```

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [c](#), [GiNaC::denom\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::numer\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), [GiNaC::info\\_flags::real](#), and [GiNaC::expairseq::seq](#).

**6.3.3.33 do\_print\_python\_repr()**

```
void GiNaC::add::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::op\(\)](#), and [GiNaC::ex::print\(\)](#).

**6.3.4 Friends And Related Function Documentation****6.3.4.1 mul**

```
friend class mul [friend]
```

**6.3.4.2 power**

```
friend class power [friend]
```

The documentation for this class was generated from the following files:

- [add.h](#)
- [add.cpp](#)
- [indexed.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.4 GiNaC::archive Class Reference

This class holds archived versions of [GiNaC](#) expressions (class [ex](#)).

```
#include <archive.h>
```

### Classes

- struct [archived\\_ex](#)  
*Archived expression descriptor.*

### Public Member Functions

- [archive](#) ()  
Construct archive from expression using the default name "ex".
- [~archive](#) ()
- [archive](#) (const [ex](#) &e)  
Construct archive from expression using the specified name.
- [archive\\_ex](#) (const [ex](#) &e, const char \*n)  
Archive an expression.
- [ex unarchive\\_ex](#) (const [lst](#) &sym\_lst, const char \*name) const  
Retrieve expression from archive by name.
- [ex unarchive\\_ex](#) (const [lst](#) &sym\_lst, unsigned index=0) const  
Retrieve expression from archive by index.
- [ex unarchive\\_ex](#) (const [lst](#) &sym\_lst, std::string &name, unsigned index=0) const  
Retrieve expression and its name from archive by index.
- unsigned [num\\_expressions](#) () const  
Return number of archived expressions.
- const [archive\\_node](#) & [get\\_top\\_node](#) (unsigned index=0) const  
Return reference to top node of an expression specified by index.
- void [clear](#) ()  
Clear all archived expressions.
- [archive\\_node\\_id add\\_node](#) (const [archive\\_node](#) &n)  
Add [archive\\_node](#) to archive if the corresponding expression is not already archived.
- [archive\\_node](#) & [get\\_node](#) ([archive\\_node\\_id](#) id)  
Retrieve [archive\\_node](#) by ID.
- void [forget](#) ()  
Delete cached unarchived expressions in all [archive\\_nodes](#) (mainly for debugging).
- void [prinraw](#) (std::ostream &os) const  
Print archive to stream in ugly raw format (for debugging).
- [archive\\_atom atomize](#) (const std::string &s) const  
Atomize a string (i.e.
- const std::string & [unatomize](#) ([archive\\_atom](#) id) const  
Unatomize a string (i.e.

### Private Types

- typedef std::map< std::string, [archive\\_atom](#) >::const\_iterator [inv\\_at\\_cit](#)  
The map of from strings to indices of the atoms vectors allows for faster archiving.

## Private Attributes

- `std::vector< archive_node > nodes`  
*Vector of archived nodes.*
- `std::vector< archived_ex > exprs`  
*Vector of archived expression descriptors.*
- `std::vector< std::string > atoms`  
*Vector of atomized strings (using a vector allows faster unarchiving).*
- `std::map< std::string, archive_atom > inverse_atoms`
- `std::map< ex, archive_node_id, ex_is_less > exprtable`  
*Map of stored expressions to nodes for faster archiving.*

## Friends

- `std::ostream & operator<< (std::ostream &os, const archive &ar)`  
*Write archive to binary data stream.*
- `std::istream & operator>> (std::istream &is, archive &ar)`  
*Read archive from binary data stream.*

### 6.4.1 Detailed Description

This class holds archived versions of [GiNaC](#) expressions (class `ex`).

An archive can be constructed from an expression and then written to a stream; or it can be read from a stream and then unarchived, yielding back the expression. Archives can hold multiple expressions which can be referred to by name or index number. The main component of the archive class is a vector of `archive_nodes` which each store one object of class `basic` (or a derived class).

### 6.4.2 Member Typedef Documentation

#### 6.4.2.1 `inv_at_cit`

```
typedef std::map<std::string, archive_atom>::const_iterator GiNaC::archive::inv_at_cit [private]
```

The map of from strings to indices of the atoms vectors allows for faster archiving.

### 6.4.3 Constructor & Destructor Documentation

#### 6.4.3.1 `archive()` [1/3]

```
GiNaC::archive::archive ( ) [inline]
```

#### 6.4.3.2 `~archive()`

```
GiNaC::archive::~~archive ( ) [inline]
```

#### 6.4.3.3 `archive()` [2/3]

```
GiNaC::archive::archive (
    const ex & e ) [inline]
```

Construct archive from expression using the default name "ex".

References `archive_ex()`.

#### 6.4.3.4 `archive()` [3/3]

```
GiNaC::archive::archive (
    const ex & e,
    const char * n ) [inline]
```

Construct archive from expression using the specified name.

References `archive_ex()`, and `n`.

### 6.4.4 Member Function Documentation

#### 6.4.4.1 `archive_ex()`

```
void GiNaC::archive::archive_ex (
    const ex & e,
    const char * name )
```

Archive an expression.

##### Parameters

<i>e</i>	the expression to be archived
<i>name</i>	name under which the expression is stored

References `add_node()`, `atomize()`, and `exprs`.

Referenced by `archive()`.

6.4.4.2 `unarchive_ex()` [1/3]

```
ex GiNaC::archive::unarchive_ex (
    const lst & sym_lst,
    const char * name ) const
```

Retrieve expression from archive by name.

## Parameters

<i>sym_lst</i>	list of pre-defined symbols
<i>name</i>	name of expression

References `atomize()`, `exprs`, and `nodes`.

6.4.4.3 `unarchive_ex()` [2/3]

```
ex GiNaC::archive::unarchive_ex (
    const lst & sym_lst,
    unsigned index = 0 ) const
```

Retrieve expression from archive by index.

## Parameters

<i>sym_lst</i>	list of pre-defined symbols
<i>index</i>	index of expression

## See also

`count_expressions`

References `exprs`, and `nodes`.

6.4.4.4 `unarchive_ex()` [3/3]

```
ex GiNaC::archive::unarchive_ex (
    const lst & sym_lst,
    std::string & name,
    unsigned index = 0 ) const
```

Retrieve expression and its name from archive by index.

## Parameters

<i>sym_lst</i>	list of pre-defined symbols
<i>name</i>	receives the name of the expression
<i>index</i>	index of expression

See also

`count_expressions`

References `exprs`, `nodes`, and `unatomize()`.

#### 6.4.4.5 `num_expressions()`

```
unsigned GiNaC::archive::num_expressions ( ) const
```

Return number of archived expressions.

References `exprs`.

#### 6.4.4.6 `get_top_node()`

```
const archive_node & GiNaC::archive::get_top_node (
    unsigned index = 0 ) const
```

Return reference to top node of an expression specified by index.

References `exprs`, and `nodes`.

#### 6.4.4.7 `clear()`

```
void GiNaC::archive::clear ( )
```

Clear all archived expressions.

References `atoms`, `exprs`, `exprtable`, `inverse_atoms`, and `nodes`.

#### 6.4.4.8 `add_node()`

```
archive_node_id GiNaC::archive::add_node (
    const archive_node & n )
```

Add `archive_node` to archive if the corresponding expression is not already archived.

Returns

ID of archived node

References `exprtable`, `n`, and `nodes`.

Referenced by `GiNaC::archive_node::add_ex()`, and `archive_ex()`.

6.4.4.9 `get_node()`

```
archive_node & GiNaC::archive::get_node (
    archive_node_id id )
```

Retrieve `archive_node` by ID.

References nodes.

Referenced by `GiNaC::archive_node::find_ex()`, `GiNaC::archive_node::find_ex_by_loc()`, and `GiNaC::archive_node::find_ex_node()`.

6.4.4.10 `forget()`

```
void GiNaC::archive::forget ( )
```

Delete cached unarchived expressions in all `archive_nodes` (mainly for debugging).

References `GiNaC::archive_node::forget()`, and nodes.

6.4.4.11 `printraw()`

```
void GiNaC::archive::printraw (
    std::ostream & os ) const
```

Print archive to stream in ugly raw format (for debugging).

References atoms, exprs, nodes, and `unatomize()`.

6.4.4.12 `atomize()`

```
archive_atom GiNaC::archive::atomize (
    const std::string & s ) const
```

Atomize a string (i.e.

convert it into an ID number that uniquely represents the string).

References atoms, and `inverse_atoms`.

Referenced by `GiNaC::archive_node::add_bool()`, `GiNaC::archive_node::add_ex()`, `GiNaC::archive_node::add_string()`, `GiNaC::archive_node::add_unsigned()`, `archive_ex()`, `GiNaC::archive_node::find_bool()`, `GiNaC::archive_node::find_ex()`, `GiNaC::archive_node::find_ex_node()`, `GiNaC::archive_node::find_first()`, `GiNaC::archive_node::find_last()`, `GiNaC::archive_node::find_property_range()`, `GiNaC::archive_node::find_string()`, `GiNaC::archive_node::find_unsigned()`, and `unarchive_ex()`.

#### 6.4.4.13 unatomize()

```
const std::string & GiNaC::archive::unatomize (
    archive_atom id ) const
```

Unatomize a string (i.e.

convert the ID number back to the string).

References atoms.

Referenced by `GiNaC::archive_node::find_string()`, `GiNaC::archive_node::get_properties()`, `GiNaC::archive_node::printraw()`, `printraw()`, and `unarchive_ex()`.

### 6.4.5 Friends And Related Function Documentation

#### 6.4.5.1 operator<<

```
std::ostream& operator<< (
    std::ostream & os,
    const archive & ar ) [friend]
```

Write archive to binary data stream.

#### 6.4.5.2 operator>>

```
std::istream& operator>> (
    std::istream & is,
    archive & ar ) [friend]
```

Read archive from binary data stream.

### 6.4.6 Member Data Documentation

#### 6.4.6.1 nodes

```
std::vector<archive_node> GiNaC::archive::nodes [private]
```

Vector of archived nodes.

Referenced by `add_node()`, `clear()`, `forget()`, `get_node()`, `get_top_node()`, `GiNaC::operator<<()`, `GiNaC::operator>>()`, `printraw()`, and `unarchive_ex()`.



#### 6.4.6.2 exprs

```
std::vector<archived_ex> GiNaC::archive::exprs [private]
```

Vector of archived expression descriptors.

Referenced by `archive_ex()`, `clear()`, `get_top_node()`, `num_expressions()`, `GiNaC::operator<<()`, `GiNaC::operator>>()`, `printraw()`, and `unarchive_ex()`.

#### 6.4.6.3 atoms

```
std::vector<std::string> GiNaC::archive::atoms [mutable], [private]
```

Vector of atomized strings (using a vector allows faster unarchiving).

Referenced by `atomize()`, `clear()`, `GiNaC::operator<<()`, `GiNaC::operator>>()`, `printraw()`, and `unatomize()`.

#### 6.4.6.4 inverse\_atoms

```
std::map<std::string, archived_atom> GiNaC::archive::inverse_atoms [mutable], [private]
```

Referenced by `atomize()`, `clear()`, and `GiNaC::operator>>()`.

#### 6.4.6.5 exprtable

```
std::map<ex, archive_node_id, ex_is_less> GiNaC::archive::exprtable [mutable], [private]
```

Map of stored expressions to nodes for faster archiving.

Referenced by `add_node()`, and `clear()`.

The documentation for this class was generated from the following files:

- [archive.h](#)
- [archive.cpp](#)

## 6.5 GiNaC::archive\_node Class Reference

This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class).

```
#include <archive.h>
```

## Classes

- struct [archive\\_node\\_cit\\_range](#)
- struct [property](#)  
*Archived property (data type, name and associated data)*
- struct [property\\_info](#)  
*Information about a stored property.*

## Public Types

- enum [property\\_type](#) { [PTYPE\\_BOOL](#), [PTYPE\\_UNSIGNED](#), [PTYPE\\_STRING](#), [PTYPE\\_NODE](#) }  
*Property data types.*
- typedef std::vector< [property\\_info](#) > [propinfovector](#)
- typedef std::vector< [property](#) >::const\_iterator [archive\\_node\\_cit](#)

## Public Member Functions

- [archive\\_node](#) ([archive](#) &ar)
- [archive\\_node](#) ([archive](#) &ar, const [ex](#) &expr)  
*Recursively construct archive node from expression.*
- const [archive\\_node](#) & [operator=](#) (const [archive\\_node](#) &other)  
*Assignment operator of [archive\\_node](#).*
- void [add\\_bool](#) (const std::string &name, bool [value](#))  
*Add property of type "bool" to node.*
- void [add\\_unsigned](#) (const std::string &name, unsigned [value](#))  
*Add property of type "unsigned int" to node.*
- void [add\\_string](#) (const std::string &name, const std::string &[value](#))  
*Add property of type "string" to node.*
- void [add\\_ex](#) (const std::string &name, const [ex](#) &[value](#))  
*Add property of type "ex" to node.*
- bool [find\\_bool](#) (const std::string &name, bool &ret, unsigned index=0) const  
*Retrieve property of type "bool" from node.*
- bool [find\\_unsigned](#) (const std::string &name, unsigned &ret, unsigned index=0) const  
*Retrieve property of type "unsigned" from node.*
- bool [find\\_string](#) (const std::string &name, std::string &ret, unsigned index=0) const  
*Retrieve property of type "string" from node.*
- [archive\\_node\\_cit](#) [find\\_first](#) (const std::string &name) const  
*Find the location in the vector of properties of the first/last property with a given name.*
- [archive\\_node\\_cit](#) [find\\_last](#) (const std::string &name) const
- [archive\\_node\\_cit\\_range](#) [find\\_property\\_range](#) (const std::string &name1, const std::string &name2) const  
*Find a range of locations in the vector of properties.*
- bool [find\\_ex](#) (const std::string &name, [ex](#) &ret, [lst](#) &sym\_lst, unsigned index=0) const  
*Retrieve property of type "ex" from node.*
- void [find\\_ex\\_by\\_loc](#) ([archive\\_node\\_cit](#) loc, [ex](#) &ret, [lst](#) &sym\_lst) const  
*Retrieve property of type "ex" from the node if it is known that this node in fact contains such a property at the given location.*
- const [archive\\_node](#) & [find\\_ex\\_node](#) (const std::string &name, unsigned index=0) const  
*Retrieve property of type "ex" from node, returning the node of the sub-expression.*
- void [get\\_properties](#) ([propinfovector](#) &v) const  
*Return vector of properties stored in node.*

- `ex unarchive (lst &sym_lst) const`  
*Convert archive node to [GiNaC](#) expression.*
- `bool has_same_ex_as (const archive\_node &other) const`  
*Check if the [archive\\_node](#) stores the same expression as another [archive\\_node](#).*
- `bool has_ex () const`
- `ex get_ex () const`
- `void forget ()`  
*Delete cached unarchived expressions from node (for debugging).*
- `void printraw (std::ostream &os) const`  
*Output [archive\\_node](#) to stream in ugly raw format (for debugging).*

### Private Attributes

- `archive & a`  
*Reference to the archive to which this node belongs.*
- `std::vector< property > props`  
*Vector of stored properties.*
- `bool has_expression`  
*Flag indicating whether a cached unarchived representation of this node exists.*
- `ex e`  
*The cached unarchived representation of this node (if any).*

### Friends

- `std::ostream & operator<< (std::ostream &os, const archive\_node &ar)`  
*Write [archive\\_node](#) to binary data stream.*
- `std::istream & operator>> (std::istream &is, archive\_node &ar)`  
*Read [archive\\_node](#) from binary data stream.*

## 6.5.1 Detailed Description

This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class).

Each property is addressed by its name and data type.

## 6.5.2 Member Typedef Documentation

### 6.5.2.1 propinfovector

```
typedef std::vector<property\_info> GiNaC::archive\_node::propinfovector
```

### 6.5.2.2 archive\_node\_cit

```
typedef std::vector<property>::const_iterator GiNaC::archive_node::archive_node_cit
```

## 6.5.3 Member Enumeration Documentation

### 6.5.3.1 property\_type

```
enum GiNaC::archive_node::property_type
```

Property data types.

Enumerator

PTYPE_BOOL	
PTYPE_UNSIGNED	
PTYPE_STRING	
PTYPE_NODE	

## 6.5.4 Constructor & Destructor Documentation

### 6.5.4.1 archive\_node() [1/2]

```
GiNaC::archive_node::archive_node (
    archive & ar ) [inline]
```

Referenced by add\_ex().

### 6.5.4.2 archive\_node() [2/2]

```
GiNaC::archive_node::archive_node (
    archive & ar,
    const ex & expr )
```

Recursively construct archive node from expression.

References GiNaC::ex::bp.

## 6.5.5 Member Function Documentation

### 6.5.5.1 operator=()

```
const archive_node & GiNaC::archive_node::operator= (
    const archive_node & other )
```

Assignment operator of [archive\\_node](#).

References [e](#), [has\\_expression](#), and [props](#).

### 6.5.5.2 add\_bool()

```
void GiNaC::archive_node::add_bool (
    const std::string & name,
    bool value )
```

Add property of type "bool" to node.

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_BOOL](#), and [value](#).

### 6.5.5.3 add\_unsigned()

```
void GiNaC::archive_node::add_unsigned (
    const std::string & name,
    unsigned value )
```

Add property of type "unsigned int" to node.

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_UNSIGNED](#), and [value](#).

### 6.5.5.4 add\_string()

```
void GiNaC::archive_node::add_string (
    const std::string & name,
    const std::string & value )
```

Add property of type "string" to node.

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_STRING](#), and [value](#).

#### 6.5.5.5 add\_ex()

```
void GiNaC::archive_node::add_ex (
    const std::string & name,
    const ex & value )
```

Add property of type "ex" to node.

References a, GiNaC::archive::add\_node(), archive\_node(), GiNaC::archive::atomize(), props, PTYPE\_NODE, and value.

Referenced by GiNaC::container< C >::archive().

#### 6.5.5.6 find\_bool()

```
bool GiNaC::archive_node::find_bool (
    const std::string & name,
    bool & ret,
    unsigned index = 0 ) const
```

Retrieve property of type "bool" from node.

##### Returns

"true" if property was found, "false" otherwise

References a, GiNaC::archive::atomize(), props, and PTYPE\_BOOL.

#### 6.5.5.7 find\_unsigned()

```
bool GiNaC::archive_node::find_unsigned (
    const std::string & name,
    unsigned & ret,
    unsigned index = 0 ) const
```

Retrieve property of type "unsigned" from node.

##### Returns

"true" if property was found, "false" otherwise

References a, GiNaC::archive::atomize(), props, and PTYPE\_UNSIGNED.

## 6.5.5.8 find\_string()

```
bool GiNaC::archive_node::find_string (
    const std::string & name,
    std::string & ret,
    unsigned index = 0 ) const
```

Retrieve property of type "string" from node.

## Returns

"true" if property was found, "false" otherwise

References a, GiNaC::archive::atomize(), props, PTYPE\_STRING, and GiNaC::archive::unatomize().

Referenced by unarchive().

## 6.5.5.9 find\_first()

```
archive_node::archive_node_cit GiNaC::archive_node::find_first (
    const std::string & name ) const
```

Find the location in the vector of properties of the first/last property with a given name.

References a, GiNaC::archive::atomize(), and props.

## 6.5.5.10 find\_last()

```
archive_node::archive_node_cit GiNaC::archive_node::find_last (
    const std::string & name ) const
```

References a, GiNaC::archive::atomize(), and props.

## 6.5.5.11 find\_property\_range()

```
archive_node::archive_node_cit_range GiNaC::archive_node::find_property_range (
    const std::string & name1,
    const std::string & name2 ) const
```

Find a range of locations in the vector of properties.

The result begins at the first property with name1 and ends one past the last property with name2.

References a, GiNaC::archive::atomize(), GiNaC::archive\_node::archive\_node\_cit\_range::begin, GiNaC::archive\_node::archive\_node\_cit\_range::end, and props.

#### 6.5.5.12 find\_ex()

```
bool GiNaC::archive_node::find_ex (
    const std::string & name,
    ex & ret,
    lst & sym_lst,
    unsigned index = 0 ) const
```

Retrieve property of type "ex" from node.

##### Returns

"true" if property was found, "false" otherwise

References a, GiNaC::archive::atomize(), GiNaC::archive::get\_node(), props, PTYPE\_NODE, and unarchive().

#### 6.5.5.13 find\_ex\_by\_loc()

```
void GiNaC::archive_node::find_ex_by_loc (
    archive_node_cit loc,
    ex & ret,
    lst & sym_lst ) const
```

Retrieve property of type "ex" from the node if it is known that this node in fact contains such a property at the given location.

This is much more efficient than the preceding function.

References a, GiNaC::archive::get\_node(), and unarchive().

#### 6.5.5.14 find\_ex\_node()

```
const archive_node & GiNaC::archive_node::find_ex_node (
    const std::string & name,
    unsigned index = 0 ) const
```

Retrieve property of type "ex" from node, returning the node of the sub-expression.

References a, GiNaC::archive::atomize(), GiNaC::archive::get\_node(), props, and PTYPE\_NODE.

#### 6.5.5.15 get\_properties()

```
void GiNaC::archive_node::get_properties (
    propinfovector & v ) const
```

Return vector of properties stored in node.

References a, props, and GiNaC::archive::unatomize().



#### 6.5.5.16 unarchive()

```
ex GiNaC::archive_node::unarchive (
    lst & sym_lst ) const
```

Convert archive node to [GiNaC](#) expression.

References [GiNaC::status\\_flags::dynallocated](#), [e](#), [GiNaC::find\\_factory\\_fcn\(\)](#), [find\\_string\(\)](#), and [has\\_expression](#).

Referenced by [find\\_ex\(\)](#), and [find\\_ex\\_by\\_loc\(\)](#).

#### 6.5.5.17 has\_same\_ex\_as()

```
bool GiNaC::archive_node::has_same_ex_as (
    const archive_node & other ) const
```

Check if the [archive\\_node](#) stores the same expression as another [archive\\_node](#).

##### Returns

"true" if expressions are the same

References [GiNaC::ex::bp](#), [e](#), and [has\\_expression](#).

#### 6.5.5.18 has\_ex()

```
bool GiNaC::archive_node::has_ex ( ) const [inline]
```

References [has\\_expression](#).

#### 6.5.5.19 get\_ex()

```
ex GiNaC::archive_node::get_ex ( ) const [inline]
```

References [e](#).

#### 6.5.5.20 forget()

```
void GiNaC::archive_node::forget ( )
```

Delete cached unarchived expressions from node (for debugging).

References [e](#), and [has\\_expression](#).

Referenced by [GiNaC::archive::forget\(\)](#).

### 6.5.5.21 printraw()

```
void GiNaC::archive_node::printraw (
    std::ostream & os ) const
```

Output [archive\\_node](#) to stream in ugly raw format (for debugging).

References [a](#), [GiNaC::ex::bp](#), [e](#), [has\\_expression](#), [props](#), [PTYPE\\_BOOL](#), [PTYPE\\_NODE](#), [PTYPE\\_STRING](#), [PTYPE\\_UNSIGNED](#), and [GiNaC::archive::unatomize\(\)](#).

## 6.5.6 Friends And Related Function Documentation

### 6.5.6.1 operator<<

```
std::ostream& operator<< (
    std::ostream & os,
    const archive_node & ar ) [friend]
```

Write [archive\\_node](#) to binary data stream.

### 6.5.6.2 operator>>

```
std::istream& operator>> (
    std::istream & is,
    archive_node & ar ) [friend]
```

Read [archive\\_node](#) from binary data stream.

## 6.5.7 Member Data Documentation

### 6.5.7.1 a

```
archive& GiNaC::archive_node::a [private]
```

Reference to the archive to which this node belongs.

Referenced by [add\\_bool\(\)](#), [add\\_ex\(\)](#), [add\\_string\(\)](#), [add\\_unsigned\(\)](#), [find\\_bool\(\)](#), [find\\_ex\(\)](#), [find\\_ex\\_by\\_loc\(\)](#), [find\\_ex\\_node\(\)](#), [find\\_first\(\)](#), [find\\_last\(\)](#), [find\\_property\\_range\(\)](#), [find\\_string\(\)](#), [find\\_unsigned\(\)](#), [get\\_properties\(\)](#), and [printraw\(\)](#).

```
std::vector<property> GiNaC::archive_node::props [private]
```

Referenced by `add_bool()`, `add_ex()`, `add_string()`, `add_unsigned()`, `find_bool()`, `find_ex()`, `find_ex_node()`, `find_←`  
`first()`, `find_last()`, `find_property_range()`, `find_string()`, `find_unsigned()`, `get_properties()`, `operator=()`, and `printraw()`.

```
bool GiNaC::archive_node::has_expression [mutable], [private]
```

Referenced by forget(), has\_ex(), has\_same\_ex\_as(), operator=(), printraw(), and unarchive().

```
ex GiNaC::archive_node::e [mutable], [private]
```

The documentation for this class was generated from the following files:

- archive.h
- archive.cpp

## 6.6 GiNaC::archive\_node::archive\_node\_cit\_range Struct Reference

```
#include <archive.h>
```

- archive\_node\_cit begin
- archive node cit end

### 6.6.1 Member Data Documentation

#### 6.6.1.1 begin

`archive_node_cit` `GiNaC::archive_node::archive_node_cit_range::begin`

Referenced by `GiNaC::archive_node::find_property_range()`.

#### 6.6.1.2 end

`archive_node_cit` `GiNaC::archive_node::archive_node_cit_range::end`

Referenced by `GiNaC::archive_node::find_property_range()`.

The documentation for this struct was generated from the following file:

- [archive.h](#)

## 6.7 GiNaC::archive::archived\_ex Struct Reference

Archived expression descriptor.

### Public Member Functions

- [archived\\_ex](#) ()
- [archived\\_ex](#) ([archive\\_atom](#) n, [archive\\_node\\_id](#) node)

### Public Attributes

- [archive\\_atom](#) name  
*Name of expression.*
- [archive\\_node\\_id](#) root  
*ID of root node.*

#### 6.7.1 Detailed Description

Archived expression descriptor.

#### 6.7.2 Constructor & Destructor Documentation

### 6.7.2.1 archived\_ex() [1/2]

```
GiNaC::archive::archived_ex::archived_ex ( ) [inline]
```

### 6.7.2.2 archived\_ex() [2/2]

```
GiNaC::archive::archived_ex::archived_ex (
    archive_atom n,
    archive_node_id node ) [inline]
```

## 6.7.3 Member Data Documentation

### 6.7.3.1 name

`archive_atom` GiNaC::archive::archived\_ex::name

Name of expression.

### 6.7.3.2 root

`archive_node_id` GiNaC::archive::archived\_ex::root

ID of root node.

The documentation for this struct was generated from the following file:

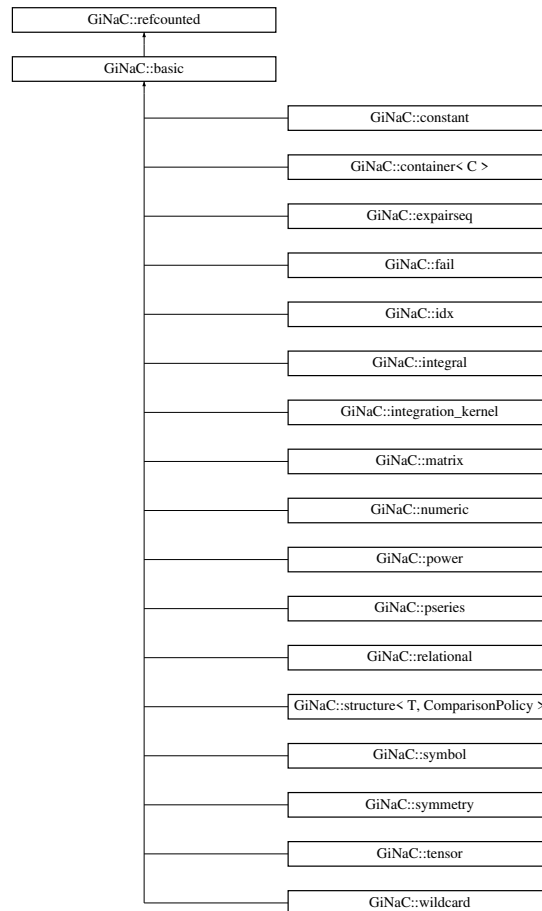
- [archive.h](#)

## 6.8 GiNaC::basic Class Reference

This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.

```
#include <basic.h>
```

Inheritance diagram for GiNaC::basic:



### Public Member Functions

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const

- Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const
  - Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const
  - Output to stream.*
- virtual void `dbgprint` () const
  - Little wrapper around print to be called within a debugger.*
- virtual void `dbgprnttree` () const
  - Little wrapper around prnttree to be called within a debugger.*
- virtual unsigned `precedence` () const
  - Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const
  - Information about the object.*
- virtual size\_t `nops` () const
  - Number of operands/members.*
- virtual `ex op` (size\_t i) const
  - Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)
  - Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const
  - Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
  - Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const
  - Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const
  - Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const
  - Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const
  - Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const
  - Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const
  - Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const
  - Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const
  - Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
  - Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
  - Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const
  - Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const

- virtual `numeric integer_content ()` const
- virtual `ex smod (const numeric &xi)` const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient ()` const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `unsigned return_type ()` const
- virtual `return_type_t return_type_tinfo ()` const
- virtual `ex conjugate ()` const
- virtual `ex real_part ()` const
- virtual `ex imag_part ()` const
- template<class T >  
void `print_dispatch (const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive (archive_node &n)` const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive (const archive_node &n, lst &syms)`  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level (const exmap &m, unsigned options)` const  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1)` const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare (const basic &other)` const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal (const basic &other)` const  
*Test for syntactic equality.*
- const `basic & hold ()` const  
*Stop further evaluation.*
- unsigned `gethash ()` const
- const `basic & setflag (unsigned f)` const  
*Set some `status_flags`.*
- const `basic & clearflag (unsigned f)` const  
*Clear some `status_flags`.*

## Protected Member Functions

- `basic ()`
- virtual `ex eval_ncmul (const exvector &v)` const
- virtual `bool match_same_type (const basic &other)` const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative (const symbol &s)` const  
*Default implementation of `ex::diff()`.*



- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### Friends

- class `ex`

## 6.8.1 Detailed Description

This class is the ABC (abstract base class) of GiNaC's class hierarchy.

## 6.8.2 Constructor & Destructor Documentation

### 6.8.2.1 `basic()` [1/2]

```
GiNaC::basic::basic ( ) [inline], [protected]
```

Referenced by `duplicate()`.

### 6.8.2.2 `~basic()`

```
virtual GiNaC::basic::~~basic ( ) [inline], [virtual]
```

basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.

References `GiNaC::status_flags::dynallocated`, `flags`, `GiNaC::refcounted::get_refcount()`, and `GINAC_ASSERT`.

### 6.8.2.3 `basic()` [2/2]

```
GiNaC::basic::basic (
    const basic & other )
```

## 6.8.3 Member Function Documentation

### 6.8.3.1 `operator=()`

```
const basic & GiNaC::basic::operator= (
    const basic & other )
```

basic assignment operator: the other object might be of a derived class.

References `flags`, and `hashvalue`.

### 6.8.3.2 `duplicate()`

```
virtual basic* GiNaC::basic::duplicate ( ) const [inline], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the `refcounted` construction of an `ex` from a `basic`.

Reimplemented in [GiNaC::possymbol](#), and [GiNaC::realsymbol](#).

References `basic()`, `GiNaC::status_flags::dynallocated`, and `setflag()`.

Referenced by `GiNaC::ex::construct_from_basic()`, `GiNaC::idx::map()`, `GiNaC::idx::replace_dim()`, `GiNaC::idx::subs()`, `GiNaC::spinidx::toggle_dot()`, `GiNaC::varidx::toggle_variance()`, and `GiNaC::spinidx::toggle_variance_dot()`.

### 6.8.3.3 eval()

```
ex GiNaC::basic::eval ( ) const [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::numeric](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::mul](#), [GiNaC::add](#), [GiNaC::pseries](#), [GiNaC::symbol](#), and [GiNaC::integral](#).

Referenced by [GiNaC::ex::construct\\_from\\_basic\(\)](#).

### 6.8.3.4 evalf()

```
ex GiNaC::basic::evalf ( ) const [virtual]
```

Evaluate object numerically.

Reimplemented in [GiNaC::function](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::mul](#), [GiNaC::idx](#), [GiNaC::pseries](#), [GiNaC::constant](#), [GiNaC::symbol](#), and [GiNaC::integral](#).

References [GiNaC::nops\(\)](#).

Referenced by [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

### 6.8.3.5 evalm()

```
ex GiNaC::basic::evalm ( ) const [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::mul](#), [GiNaC::add](#), and [GiNaC::matrix](#).

References [GiNaC::map\\_evalm](#), and [GiNaC::nops\(\)](#).

### 6.8.3.6 eval\_integ()

```
ex GiNaC::basic::eval_integ ( ) const [virtual]
```

Evaluate integrals, if result is known.

Reimplemented in [GiNaC::pseries](#), and [GiNaC::integral](#).

References [GiNaC::map\\_eval\\_integ](#), and [GiNaC::nops\(\)](#).

#### 6.8.3.7 eval\_ncmul()

```
ex GiNaC::basic::eval_ncmul (
    const exvector & v ) const [protected], [virtual]
```

Reimplemented in [GiNaC::function](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::power](#), [GiNaC::mul](#), [GiNaC::relational](#), [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::color](#), and [GiNaC::integral](#).

References [GiNaC::hold\\_ncmul\(\)](#).

#### 6.8.3.8 eval\_indexed()

```
ex GiNaC::basic::eval_indexed (
    const basic & i ) const [virtual]
```

Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.

Reimplemented in [GiNaC::tensepsilon](#), [GiNaC::spinmetric](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::su3d](#), [GiNaC::su3f](#), [GiNaC::minkmetric](#), [GiNaC::tensmetric](#), [GiNaC::tensdelta](#), and [GiNaC::matrix](#).

Referenced by [GiNaC::indexed::eval\(\)](#).

#### 6.8.3.9 print()

```
void GiNaC::basic::print (
    const print\_context & c,
    unsigned level = 0 ) const [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

##### Parameters

<i>c</i>	print context object that describes the output formatting
<i>level</i>	value that is used to identify the precedence or indentation level for placing parentheses and formatting

Reimplemented in [GiNaC::function](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::fderivative](#).

References [c](#).

Referenced by [GiNaC::fderivative::print\(\)](#), and [GiNaC::pseries::print\\_series\(\)](#).

**6.8.3.10 dbgprint()**

```
void GiNaC::basic::dbgprint ( ) const [virtual]
```

Little wrapper around print to be called within a debugger.

This is needed because you cannot call `foo.print(cout)` from within the debugger because it might not know what `cout` is. This method can be invoked with no argument and it will simply print to `stdout`.

See also

[basic::print](#)  
[basic::dbgprinttree](#)

**6.8.3.11 dbgprinttree()**

```
void GiNaC::basic::dbgprinttree ( ) const [virtual]
```

Little wrapper around printtree to be called within a debugger.

See also

[basic::dbgprint](#)

**6.8.3.12 precedence()**

```
unsigned GiNaC::basic::precedence ( ) const [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::container< C >](#), [GiNaC::numeric](#), [GiNaC::expairseq](#), [GiNaC::ncmul](#), [GiNaC::relational](#), [GiNaC::clifford](#), [GiNaC::power](#), [GiNaC::mul](#), [GiNaC::add](#), [GiNaC::pseries](#), and [GiNaC::integral](#).

**6.8.3.13 info()**

```
bool GiNaC::basic::info (
    unsigned inf ) const [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented in [GiNaC::function](#), [GiNaC::tensepsilon](#), [GiNaC::indexed](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::spinmetric](#), [GiNaC::container< C >](#), [GiNaC::minkmetric](#), [GiNaC::numeric](#), [GiNaC::tensmetric](#), [GiNaC::expairseq](#), [GiNaC::tensdelta](#), [GiNaC::ncmul](#), [GiNaC::relational](#), [GiNaC::power](#), [GiNaC::mul](#), [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::idx](#), and [GiNaC::symbol](#).

Referenced by [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::function::info\(\)](#), and [GiNaC::matrix::solve\(\)](#).

#### 6.8.3.14 nops()

```
size_t GiNaC::basic::nops ( ) const [virtual]
```

Number of operands/members.

Reimplemented in [GiNaC::user\\_defined\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::container< C >](#), [GiNaC::clifford](#), [GiNaC::expairseq](#), [GiNaC::relational](#), [GiNaC::power](#), [GiNaC::matrix](#), [GiNaC::idx](#), [GiNaC::integral](#), and [GiNaC::pseries](#).

Referenced by [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), and [normal\(\)](#).

#### 6.8.3.15 op()

```
ex GiNaC::basic::op (
    size_t i ) const [virtual]
```

Return operand/member at position i.

Reimplemented in [GiNaC::user\\_defined\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::container< C >](#), [GiNaC::clifford](#), [GiNaC::expairseq](#), [GiNaC::relational](#), [GiNaC::power](#), [GiNaC::matrix](#), [GiNaC::idx](#), [GiNaC::integral](#), and [GiNaC::pseries](#).

Referenced by [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), and [GiNaC::tensepsilon::eval\\_indexed\(\)](#).

#### 6.8.3.16 operator[]() [1/4]

```
ex GiNaC::basic::operator[] (
    const ex & index ) const [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::op\(\)](#), and [GiNaC::to\\_int\(\)](#).

#### 6.8.3.17 operator[]() [2/4]

```
ex GiNaC::basic::operator[] (
    size_t i ) const [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::op\(\)](#).

**6.8.3.18** `let_op()`

```
ex & GiNaC::basic::let_op (
    size_t i ) [virtual]
```

Return modifiable operand/member at position i.

Reimplemented in [GiNaC::user\\_defined\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::container< C >](#), [GiNaC::clifford](#), [GiNaC::matrix](#), and [GiNaC::integral](#).

Referenced by `map()`, and `subs()`.

**6.8.3.19** `operator[]()` [3/4]

```
ex & GiNaC::basic::operator[] (
    const ex & index ) [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::to\\_int\(\)](#).

**6.8.3.20** `operator[]()` [4/4]

```
ex & GiNaC::basic::operator[] (
    size_t i ) [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

**6.8.3.21** `has()`

```
bool GiNaC::basic::has (
    const ex & pattern,
    unsigned options = 0 ) const [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::numeric](#), [GiNaC::power](#), and [GiNaC::mul](#).

References [GiNaC::ex::has\(\)](#), [GiNaC::match\(\)](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), and `options`.

Referenced by [GiNaC::power::has\(\)](#), and `series()`.

### 6.8.3.22 match()

```
bool GiNaC::basic::match (
    const ex & pattern,
    exmap & repl_lst ) const [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to `repl_lst`.

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::expairseq](#), and [GiNaC::wildcard](#).

References [GiNaC::ex::match\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::op\(\)](#).

### 6.8.3.23 match\_same\_type()

```
bool GiNaC::basic::match_same_type (
    const basic & other ) const [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented in [GiNaC::function](#), [GiNaC::spinidx](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::varidx](#), [GiNaC::fderivative](#), [GiNaC::matrix](#), [GiNaC::relational](#), [GiNaC::idx](#), [GiNaC::clifford](#), and [GiNaC::color](#).

### 6.8.3.24 subs()

```
ex GiNaC::basic::subs (
    const exmap & m,
    unsigned options = 0 ) const [virtual]
```

Substitute a set of objects by arbitrary expressions.

The `ex` returned will already be evaluated.

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::container< C >](#), [GiNaC::numeric](#), [GiNaC::clifford](#), [GiNaC::expairseq](#), [GiNaC::power](#), [GiNaC::relational](#), [GiNaC::matrix](#), [GiNaC::pseries](#), [GiNaC::idx](#), and [GiNaC::symbol](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [clearflag\(\)](#), [let\\_op\(\)](#), *m*, [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), *options*, [GiNaC::ex::subs\(\)](#), and [subs\\_one\\_level\(\)](#).

Referenced by [GiNaC::tensmetric::eval\\_indexed\(\)](#), and [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#).



**6.8.3.25 map()**

```
ex GiNaC::basic::map (
    map_function & f ) const [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::expairseq](#), [GiNaC::relational](#), [GiNaC::power](#), and [GiNaC::idx](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [clearflag\(\)](#), [let\\_op\(\)](#), [n](#), [GiNaC::nops\(\)](#), and [GiNaC::op\(\)](#).

Referenced by [normal\(\)](#).

**6.8.3.26 accept()**

```
virtual void GiNaC::basic::accept (
    GiNaC::visitor & v ) const [inline], [virtual]
```

**6.8.3.27 is\_polynomial()**

```
bool GiNaC::basic::is_polynomial (
    const ex & var ) const [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented in [GiNaC::numeric](#), [GiNaC::symbol](#), [GiNaC::power](#), [GiNaC::mul](#), [GiNaC::constant](#), and [GiNaC::add](#).

References [GiNaC::has\(\)](#).

**6.8.3.28 degree()**

```
int GiNaC::basic::degree (
    const ex & s ) const [virtual]
```

Return degree of highest power in object s.

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::numeric](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::mul](#), [GiNaC::add](#), [GiNaC::pseries](#), and [GiNaC::integral](#).

**6.8.3.29 ldegree()**

```
int GiNaC::basic::ldegree (
    const ex & s ) const [virtual]
```

Return degree of lowest power in object s.

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::numeric](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::mul](#), [GiNaC::add](#), [GiNaC::pseries](#), and [GiNaC::integral](#).

**6.8.3.30 coeff()**

```
ex GiNaC::basic::coeff (
    const ex & s,
    int n = 1 ) const [virtual]
```

Return coefficient of degree n in object s.

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::numeric](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::mul](#), [GiNaC::add](#), and [GiNaC::pseries](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), and n.

Referenced by [series\(\)](#), and [GiNaC::sqrfree\\_parfrac\(\)](#).

**6.8.3.31 expand()**

```
ex GiNaC::basic::expand (
    unsigned options = 0 ) const [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented in [GiNaC::function](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::indexed](#), [GiNaC::mul](#), [GiNaC::power](#), [GiNaC::expairseq](#), [GiNaC::add](#), [GiNaC::ncmul](#), [GiNaC::pseries](#), and [GiNaC::integral](#).

References [GiNaC::nops\(\)](#), and options.

Referenced by [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), and [GiNaC::matrix::pivot\(\)](#).

**6.8.3.32 collect()**

```
ex GiNaC::basic::collect (
    const ex & s,
    bool distributed = false ) const [virtual]
```

Sort expanded expression in terms of powers of some object(s).

## Parameters

<i>s</i>	object(s) to sort in
<i>distributed</i>	recursive or distributed form (only used when <i>s</i> is a list)

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::pseries](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::coeff\(\)](#), [GiNaC::collect\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::degree\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::ldegree\(\)](#), [n](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::pow\(\)](#), and [x](#).

**6.8.3.33 derivative()**

```
ex GiNaC::basic::derivative (
    const symbol & s ) const [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

See also

[ex::diff](#)

Reimplemented in [GiNaC::function](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::indexed](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::ncmul](#), [GiNaC::mul](#), [GiNaC::fderivative](#), [GiNaC::pseries](#), [GiNaC::add](#), [GiNaC::symbol](#), [GiNaC::integral](#), [GiNaC::constant](#), and [GiNaC::idx](#).

References [GiNaC::\\_ex0](#), and [GiNaC::nops\(\)](#).

**6.8.3.34 series()**

```
ex GiNaC::basic::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented in [GiNaC::function](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::power](#), [GiNaC::fderivative](#), [GiNaC::integral](#), [GiNaC::integration\\_kernel](#), [GiNaC::mul](#), [GiNaC::add](#), [GiNaC::pseries](#), and [GiNaC::symbol](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [coeff\(\)](#), [GiNaC::ex::diff\(\)](#), [GiNaC::numeric::div\(\)](#), [GiNaC::ex::expand\(\)](#), [has\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [order](#), [r](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::fderivative::series\(\)](#), [GiNaC::power::series\(\)](#), and [GiNaC::function::series\(\)](#).

### 6.8.3.35 normal()

```
ex GiNaC::basic::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [virtual]
```

Default implementation of [ex::normal\(\)](#).

It normalizes the children and replaces the object with a temporary symbol.

See also

[ex::normal](#)

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::mul](#), [GiNaC::add](#), [GiNaC::pseries](#), and [GiNaC::symbol](#).

References [GiNaC::\\_ex1](#), [map\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::container< C >::subs\(\)](#).

Referenced by [GiNaC::matrix::markowitz\\_elimination\(\)](#), and [GiNaC::matrix::solve\(\)](#).

### 6.8.3.36 to\_rational()

```
ex GiNaC::basic::to_rational (
    exmap & repl ) const [virtual]
```

Default implementation of [ex::to\\_rational\(\)](#).

This replaces the object with a temporary symbol.

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::expairseq](#), and [GiNaC::symbol](#).

References [GiNaC::replace\\_with\\_symbol\(\)](#).

### 6.8.3.37 to\_polynomial()

```
ex GiNaC::basic::to_polynomial (
    exmap & repl ) const [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::expairseq](#), and [GiNaC::symbol](#).

References [GiNaC::replace\\_with\\_symbol\(\)](#).

**6.8.3.38 integer\_content()**

```
numeric GiNaC::basic::integer_content ( ) const [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::numeric](#), [GiNaC::mul](#), and [GiNaC::add](#).

References [GiNaC::\\_num1\\_p](#).

**6.8.3.39 smod()**

```
ex GiNaC::basic::smod (
    const numeric & xi ) const [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

**Parameters**

$xi$	modulus
------	---------

**Returns**

mapped polynomial

**See also**

[heur\\_gcd](#)

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::numeric](#), [GiNaC::mul](#), and [GiNaC::add](#).

**6.8.3.40 max\_coefficient()**

```
numeric GiNaC::basic::max_coefficient ( ) const [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

**See also**

[heur\\_gcd](#)

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::numeric](#), [GiNaC::mul](#), and [GiNaC::add](#).

References [GiNaC::\\_num1\\_p](#).

#### 6.8.3.41 `get_free_indices()`

```
exvector GiNaC::basic::get_free_indices ( ) const [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::indexed](#), [GiNaC::mul](#), [GiNaC::add](#), [GiNaC::ncmul](#), and [GiNaC::integral](#).

#### 6.8.3.42 `add_indexed()`

```
ex GiNaC::basic::add_indexed (
    const ex & self,
    const ex & other ) const [virtual]
```

Add two indexed expressions.

They are guaranteed to be of class `indexed` (or a subclass) and their indices are compatible. This function is used internally by [simplify\\_indexed\(\)](#).

##### Parameters

<i>self</i>	First indexed expression; its base object is <code>*this</code>
<i>other</i>	Second indexed expression

##### Returns

sum of `self` and `other`

##### See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::matrix](#).

#### 6.8.3.43 `scalar_mul_indexed()`

```
ex GiNaC::basic::scalar_mul_indexed (
    const ex & self,
    const numeric & other ) const [virtual]
```

Multiply an indexed expression with a scalar.

This function is used internally by [simplify\\_indexed\(\)](#).

## Parameters

<i>self</i>	Indexed expression; its base object is <i>*this</i>
<i>other</i>	Numeric value

## Returns

product of *self* and *other*

## See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::matrix](#).

## 6.8.3.44 contract\_with()

```
bool GiNaC::basic::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [virtual]
```

Try to contract two indexed expressions that appear in the same product.

If a contraction exists, the function overwrites one or both of the expressions and returns true. Otherwise it returns false. It is guaranteed that both expressions are of class [indexed](#) (or a subclass) and that at least one dummy index has been found. This functions is used internally by [simplify\\_indexed\(\)](#).

## Parameters

<i>self</i>	Pointer to first indexed expression; its base object is <i>*this</i>
<i>other</i>	Pointer to second indexed expression
<i>v</i>	The complete vector of factors

## Returns

true if the contraction was successful, false otherwise

## See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::tensepsilon](#), [GiNaC::spinmetric](#), [GiNaC::su3d](#), [GiNaC::diracgamma](#), [GiNaC::su3f](#), [GiNaC::cliffordunit](#), [GiNaC::su3t](#), [GiNaC::tensmetric](#), [GiNaC::tensdelta](#), and [GiNaC::matrix](#).

**6.8.3.45** `return_type()`

```
unsigned GiNaC::basic::return_type ( ) const [virtual]
```

Reimplemented in [GiNaC::function](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::tensepsilon](#), [GiNaC::indexed](#), [GiNaC::su3d](#), [GiNaC::minkmetric](#), [GiNaC::su3f](#), [GiNaC::tensmetric](#), [GiNaC::power](#), [GiNaC::expairseq](#), [GiNaC::ncmul](#), [GiNaC::mul](#), [GiNaC::matrix](#), [GiNaC::add](#), [GiNaC::relational](#), [GiNaC::tensdelta](#), [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::integral](#), [GiNaC::tensor](#), and [GiNaC::fail](#).

**6.8.3.46** `return_type_tinfo()`

```
return_type_t GiNaC::basic::return_type_tinfo ( ) const [virtual]
```

Reimplemented in [GiNaC::function](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::indexed](#), [GiNaC::power](#), [GiNaC::ncmul](#), [GiNaC::mul](#), [GiNaC::add](#), [GiNaC::relational](#), [GiNaC::clifford](#), [GiNaC::color](#), and [GiNaC::integral](#).

References [GiNaC::return\\_type\\_t::rl](#), and [GiNaC::return\\_type\\_t::tinfo](#).

**6.8.3.47** `conjugate()`

```
ex GiNaC::basic::conjugate ( ) const [virtual]
```

Reimplemented in [GiNaC::function](#), [GiNaC::spinidx](#), [GiNaC::diracgammaR](#), [GiNaC::diracgammaL](#), [GiNaC::container< C >](#), [GiNaC::diracgamma5](#), [GiNaC::numeric](#), [GiNaC::realsymbol](#), [GiNaC::expairseq](#), [GiNaC::power](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::add](#), [GiNaC::matrix](#), [GiNaC::pseries](#), [GiNaC::integral](#), [GiNaC::symbol](#), and [GiNaC::constant](#).

**6.8.3.48** `real_part()`

```
ex GiNaC::basic::real_part ( ) const [virtual]
```

Reimplemented in [GiNaC::function](#), [GiNaC::container< C >](#), [GiNaC::indexed](#), [GiNaC::numeric](#), [GiNaC::realsymbol](#), [GiNaC::power](#), [GiNaC::ncmul](#), [GiNaC::add](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::pseries](#), [GiNaC::symbol](#), and [GiNaC::constant](#).

Referenced by [GiNaC::ncmul::real\\_part\(\)](#), and [GiNaC::function::real\\_part\(\)](#).

**6.8.3.49** `imag_part()`

```
ex GiNaC::basic::imag_part ( ) const [virtual]
```

Reimplemented in [GiNaC::function](#), [GiNaC::container< C >](#), [GiNaC::indexed](#), [GiNaC::numeric](#), [GiNaC::realsymbol](#), [GiNaC::power](#), [GiNaC::ncmul](#), [GiNaC::add](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::pseries](#), [GiNaC::symbol](#), and [GiNaC::constant](#).

Referenced by [GiNaC::ncmul::imag\\_part\(\)](#), and [GiNaC::function::imag\\_part\(\)](#).



6.8.3.50 `compare_same_type()`

```
int GiNaC::basic::compare_same_type (
    const basic & other ) const [protected], [virtual]
```

Returns order relation between two objects of same type.

This needs to be implemented by each class. It may never return anything else than 0, signalling equality, or +1 and -1 signalling inequality and determining the canonical ordering. (Perl hackers will wonder why C++ doesn't feature the spaceship operator `<=>` for denoting just this.)

References `GiNaC::compare_pointers()`.

Referenced by `GiNaC::symbol::derivative()`.

6.8.3.51 `is_equal_same_type()`

```
bool GiNaC::basic::is_equal_same_type (
    const basic & other ) const [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls `compare_same_type()`. The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented in `GiNaC::function`, `GiNaC::structure< T, ComparisonPolicy >`, `GiNaC::container< C >`, `GiNaC::numeric`, `GiNaC::expairseq`, `GiNaC::fderivative`, `GiNaC::symbol`, and `GiNaC::constant`.

6.8.3.52 `calchash()`

```
unsigned GiNaC::basic::calchash ( ) const [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.

The method inherited from class `basic` computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented in `GiNaC::function`, `GiNaC::structure< T, ComparisonPolicy >`, `GiNaC::numeric`, `GiNaC::expairseq`, `GiNaC::relational`, `GiNaC::symmetry`, `GiNaC::symbol`, `GiNaC::constant`, `GiNaC::idx`, and `GiNaC::wildcard`.

References `GiNaC::ex::gethash()`, `GiNaC::make_hash_seed()`, `GiNaC::nops()`, `GiNaC::op()`, and `GiNaC::rotate_↔left()`.

Referenced by `gethash()`.

**6.8.3.53 print\_dispatch()** [1/2]

```
template<class T >
void GiNaC::basic::print_dispatch (
    const print\_context & c,
    unsigned level ) const [inline]
```

Like [print\(\)](#), but dispatch to the specified class.

Can be used by implementations of print methods to dispatch to the method of the superclass.

See also

[basic::print](#)

References c.

**6.8.3.54 print\_dispatch()** [2/2]

```
void GiNaC::basic::print_dispatch (
    const registered\_class\_info & ri,
    const print\_context & c,
    unsigned level ) const
```

Like [print\(\)](#), but dispatch to the specified class.

Can be used by implementations of print methods to dispatch to the method of the superclass.

See also

[basic::print](#)

References c, [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#), and [GiNaC::class\\_info< OPT >::options](#).

**6.8.3.55 archive()**

```
void GiNaC::basic::archive (
    archive\_node & n ) const [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented in [GiNaC::function](#), [GiNaC::spinidx](#), [GiNaC::tensepsilon](#), [GiNaC::indexed](#), [GiNaC::container< C >](#), [GiNaC::varidx](#), [GiNaC::numeric](#), [GiNaC::minkmetric](#), [GiNaC::power](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::symmetry](#), [GiNaC::matrix](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::symbol](#), [GiNaC::integral](#), [GiNaC::constant](#), [GiNaC::idx](#), [GiNaC::clifford](#), [GiNaC::color](#), and [GiNaC::wildcard](#).

References n.

## 6.8.3.56 read\_archive()

```
void GiNaC::basic::read_archive (
    const archive_node & n,
    lst & syms ) [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

## Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented in [GiNaC::function](#), [GiNaC::spinidx](#), [GiNaC::tensepsilon](#), [GiNaC::indexed](#), [GiNaC::varidx](#), [GiNaC::container< C >](#), [GiNaC::numeric](#), [GiNaC::minkmetric](#), [GiNaC::power](#), [GiNaC::expairseq](#), [GiNaC::symmetry](#), [GiNaC::fderivative](#), [GiNaC::matrix](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::symbol](#), [GiNaC::integral](#), [GiNaC::constant](#), [GiNaC::idx](#), [GiNaC::clifford](#), [GiNaC::color](#), and [GiNaC::wildcard](#).

## 6.8.3.57 subs\_one\_level()

```
ex GiNaC::basic::subs_one_level (
    const exmap & m,
    unsigned options ) const
```

Helper function for [subs\(\)](#).

Does not recurse into subexpressions.

References m, [GiNaC::match\(\)](#), options, and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::symbol::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::numeric::subs\(\)](#), and [subs\(\)](#).

## 6.8.3.58 diff()

```
ex GiNaC::basic::diff (
    const symbol & s,
    unsigned nth = 1 ) const
```

Default interface of nth derivative [ex::diff\(s, n\)](#).

It should be called instead of [::derivative\(s\)](#) for first derivatives and for nth derivatives it just recurses down.

## Parameters

s	symbol to differentiate in
nth	order of differentiation

See also

[ex::diff](#)

References `GiNaC::ex::diff()`, and `GiNaC::ex::is_zero()`.

Referenced by `GiNaC::mul::derivative()`.

#### 6.8.3.59 `compare()`

```
int GiNaC::basic::compare (
    const basic & other ) const
```

Compare objects syntactically to establish canonical ordering.

All compare functions return: -1 for \*this less than other, 0 equal, 1 greater.

References `gethash()`.

Referenced by `GiNaC::pseries::coeff()`.

#### 6.8.3.60 `is_equal()`

```
bool GiNaC::basic::is_equal (
    const basic & other ) const
```

Test for syntactic equality.

This is only a quick test, meaning objects should be in the same domain. You might have to [.expand\(\)](#), [.normal\(\)](#) objects first, depending on the domain of your computation, to get a more reliable answer.

See also

[is\\_equal\\_same\\_type](#)

References `gethash()`.

Referenced by `GiNaC::power::coeff()`, `GiNaC::ncmul::coeff()`, `GiNaC::power::degree()`, `GiNaC::ncmul::degree()`, `GiNaC::ncmul::ldegree()`, `GiNaC::power::ldegree()`, and `GiNaC::wildcard::match()`.

## 6.8.3.61 hold()

```
const basic & GiNaC::basic::hold ( ) const
```

Stop further evaluation.

See also

[basic::eval](#)

Referenced by `GiNaC::abs_conjugate()`, `GiNaC::abs_eval()`, `GiNaC::abs_evalf()`, `GiNaC::abs_expand()`, `GiNaC::abs_power()`, `GiNaC::abs_real_part()`, `GiNaC::acos_conjugate()`, `GiNaC::acos_eval()`, `GiNaC::acos_evalf()`, `GiNaC::acosh_conjugate()`, `GiNaC::acosh_eval()`, `GiNaC::acosh_evalf()`, `GiNaC::asin_conjugate()`, `GiNaC::asin_eval()`, `GiNaC::asin_evalf()`, `GiNaC::asinh_conjugate()`, `GiNaC::asinh_eval()`, `GiNaC::asinh_evalf()`, `GiNaC::atan2_eval()`, `GiNaC::atan_conjugate()`, `GiNaC::atan_eval()`, `GiNaC::atan_evalf()`, `GiNaC::atanh_conjugate()`, `GiNaC::atanh_eval()`, `GiNaC::atanh_evalf()`, `GiNaC::binomial_conjugate()`, `GiNaC::binomial_eval()`, `GiNaC::binomial_evalf()`, `GiNaC::binomial_real_part()`, `GiNaC::binomial_sym()`, `GiNaC::conjugate_expl_derivative()`, `GiNaC::cos_eval()`, `GiNaC::cos_evalf()`, `GiNaC::cosh_eval()`, `GiNaC::cosh_evalf()`, `GiNaC::csgn_power()`, `GiNaC::epsilon_tensor()`, `GiNaC::eta_imag_part()`, `GiNaC::integral::eval()`, `GiNaC::pseries::eval()`, `GiNaC::add::eval()`, `GiNaC::power::eval()`, `GiNaC::fderivative::eval()`, `GiNaC::numeric::eval()`, `GiNaC::structure< T, ComparisonPolicy >::eval()`, `GiNaC::function::eval()`, `GiNaC::matrix::eval_indexed()`, `GiNaC::tensmetric::eval_indexed()`, `GiNaC::su3f::eval_indexed()`, `GiNaC::su3d::eval_indexed()`, `GiNaC::structure< T, ComparisonPolicy >::eval_indexed()`, `GiNaC::spinmetric::eval_indexed()`, `GiNaC::tensepsilon::eval_indexed()`, `GiNaC::function::evalf()`, `GiNaC::exp_eval()`, `GiNaC::exp_evalf()`, `GiNaC::exp_expand()`, `GiNaC::exp_power()`, `GiNaC::power::expand()`, `GiNaC::factorial_conjugate()`, `GiNaC::factorial_eval()`, `GiNaC::factorial_evalf()`, `GiNaC::factorial_real_part()`, `GiNaC::G2_eval()`, `GiNaC::G2_evalf()`, `GiNaC::G3_eval()`, `GiNaC::G3_evalf()`, `GiNaC::imag_part_expl_derivative()`, `GiNaC::iterated_integral2_eval()`, `GiNaC::iterated_integral3_eval()`, `GiNaC::iterated_integral_evalf_impl()`, `GiNaC::Li2_conjugate()`, `GiNaC::Li2_eval()`, `GiNaC::Li2_evalf()`, `GiNaC::log_conjugate()`, `GiNaC::log_eval()`, `GiNaC::log_evalf()`, `GiNaC::log_expand()`, `GiNaC::log_real_part()`, `GiNaC::lorentz_eps()`, `GiNaC::psi1_eval()`, `GiNaC::psi1_evalf()`, `GiNaC::psi2_eval()`, `GiNaC::psi2_evalf()`, `GiNaC::real_part_expl_derivative()`, `GiNaC::sin_eval()`, `GiNaC::sin_evalf()`, `GiNaC::sinh_eval()`, `GiNaC::sinh_evalf()`, `GiNaC::step_conjugate()`, `GiNaC::step_eval()`, `GiNaC::step_evalf()`, `GiNaC::step_real_part()`, `GiNaC::tan_eval()`, `GiNaC::tan_evalf()`, `GiNaC::tanh_eval()`, `GiNaC::tanh_evalf()`, `GiNaC::zeta1_eval()`, `GiNaC::zeta1_evalf()`, `GiNaC::zeta2_eval()`, `GiNaC::zeta2_evalf()`, and `GiNaC::zetaderiv_eval()`.

## 6.8.3.62 gethash()

```
unsigned GiNaC::basic::gethash ( ) const [inline]
```

References `calchash()`, `flags`, `GiNaC::status_flags::hash_calculated`, and `hashvalue`.

Referenced by `compare()`, and `is_equal()`.

### 6.8.3.63 setflag()

```
const basic& GiNaC::basic::setflag (
    unsigned f ) const [inline]
```

Set some [status\\_flags](#).

References flags.

Referenced by `GiNaC::wildcard::calchash()`, `GiNaC::constant::calchash()`, `GiNaC::idx::calchash()`, `GiNaC::symbol::calchash()`, `GiNaC::symmetry::calchash()`, `GiNaC::relational::calchash()`, `GiNaC::numeric::calchash()`, `GiNaC::function::calchash()`, `GiNaC::constant::constant()`, `GiNaC::container< C >::container()`, `GiNaC::realsymbol::duplicate()`, `GiNaC::possymbol::duplicate()`, `duplicate()`, `GiNaC::power::eval()`, `GiNaC::ncmul::eval()`, `GiNaC::pseries::evalf()`, `GiNaC::integral::expand()`, `GiNaC::pseries::expand()`, `GiNaC::ncmul::expand()`, `GiNaC::add::expand()`, `GiNaC::power::expand()`, `GiNaC::function::expand()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_add_2()`, `GiNaC::power::expand_mul()`, `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()`, `GiNaC::power::info()`, `GiNaC::matrix::matrix()`, `GiNaC::numeric::numeric()`, `GiNaC::print_func< print_dflt >()`, `GiNaC::wildcard::read_archive()`, `GiNaC::symbol::read_archive()`, `GiNaC::container< C >::read_archive()`, `GiNaC::numeric::read_archive()`, `GiNaC::reduced_matrix()`, `GiNaC::sub_matrix()`, `GiNaC::symbol::symbol()`, `GiNaC::symbolic_matrix()`, `GiNaC::symmetry::symmetry()`, `GiNaC::unit_matrix()`, and `GiNaC::wildcard::wildcard()`.

### 6.8.3.64 clearflag()

```
const basic& GiNaC::basic::clearflag (
    unsigned f ) const [inline]
```

Clear some [status\\_flags](#).

References flags.

Referenced by `GiNaC::add::combine_ex_with_coeff_to_pair()`, `GiNaC::mul::eval()`, `GiNaC::power::eval()`, `GiNaC::function::function()`, `GiNaC::power::info()`, `GiNaC::idx::map()`, `map()`, `GiNaC::idx::replace_dim()`, `GiNaC::mul::smod()`, `GiNaC::add::split_ex_to_pair()`, `GiNaC::idx::subs()`, `subs()`, `GiNaC::spinidx::toggle_dot()`, `GiNaC::varidx::toggle_variance()`, and `GiNaC::spinidx::toggle_variance_dot()`.

### 6.8.3.65 ensure\_if\_modifiable()

```
void GiNaC::basic::ensure_if_modifiable ( ) const [protected]
```

Ensure the object may be modified without hurting others, throws if this is not the case.

Referenced by `GiNaC::matrix::division_free_elimination()`, `GiNaC::matrix::fraction_free_elimination()`, `GiNaC::matrix::gauss_elimination()`, `GiNaC::integral::let_op()`, `GiNaC::matrix::let_op()`, `GiNaC::clifford::let_op()`, `GiNaC::multiple_polylog_kernel::let_op()`, `GiNaC::ELi_kernel::let_op()`, `GiNaC::Ebar_kernel::let_op()`, `GiNaC::Kronecker_dtau_kernel::let_op()`, `GiNaC::Kronecker_dz_kernel::let_op()`, `GiNaC::Eisenstein_kernel::let_op()`, `GiNaC::Eisenstein_h_kernel::let_op()`, `GiNaC::modular_form_kernel::let_op()`, `GiNaC::user_defined_kernel::let_op()`, `GiNaC::matrix::operator()()`, and `GiNaC::matrix::pivot()`.

**6.8.3.66 do\_print()**

```
void GiNaC::basic::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

Default output to stream.

References `c`.

**6.8.3.67 do\_print\_tree()**

```
void GiNaC::basic::do_print_tree (
    const print_tree & c,
    unsigned level ) const [protected]
```

Tree output to stream.

References `c`, `GiNaC::nops()`, `GiNaC::op()`, and `GiNaC::ex::print()`.

**6.8.3.68 do\_print\_python\_repr()**

```
void GiNaC::basic::do_print_python_repr (
    const print_python_repr & c,
    unsigned level ) const [protected]
```

Python parsable output to stream.

References `c`.

**6.8.4 Friends And Related Function Documentation****6.8.4.1 ex**

```
friend class ex [friend]
```

Referenced by `GiNaC::matrix::charpoly()`, `GiNaC::indexed::eval()`, `GiNaC::function::eval()`, `GiNaC::integral::expand()`, `GiNaC::power::expand_mul()`, and `GiNaC::structure< T, ComparisonPolicy >::scalar_mul_indexed()`.

**6.8.5 Member Data Documentation**

### 6.8.5.1 flags

unsigned GiNaC::basic::flags [mutable], [protected]

of type [status\\_flags](#)

Referenced by [GiNaC::idx::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), [GiNaC::symmetry::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [clearflag\(\)](#), [GiNaC::ex::construct\\_from\\_basic\(\)](#), [GiNaC::wildcard::do\\_print\\_tree\(\)](#), [GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::integral::expand\(\)](#), [gethash\(\)](#), [GiNaC::power::info\(\)](#), [operator=\(\)](#), [GiNaC::function::print\(\)](#), [setflag\(\)](#), and [~basic\(\)](#).

### 6.8.5.2 hashvalue

unsigned GiNaC::basic::hashvalue [mutable], [protected]

hash value

Referenced by [GiNaC::wildcard::calchash\(\)](#), [GiNaC::constant::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::symbol::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), [GiNaC::symmetry::calchash\(\)](#), [GiNaC::numeric::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::wildcard::do\\_print\\_tree\(\)](#), [GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [gethash\(\)](#), [operator=\(\)](#), and [GiNaC::function::print\(\)](#).

The documentation for this class was generated from the following files:

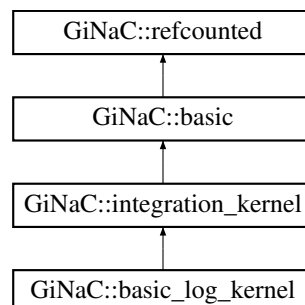
- [basic.h](#)
- [basic.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.9 GiNaC::basic\_log\_kernel Class Reference

The basic integration kernel with a logarithmic singularity at the origin.

```
#include <integration_kernel.h>
```

Inheritance diagram for [GiNaC::basic\\_log\\_kernel](#):





## Protected Member Functions

- `cln::cl_N series_coeff_impl (int i) const` override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `void do_print (const print_context &c, unsigned level) const`

## Additional Inherited Members

### 6.9.1 Detailed Description

The basic integration kernel with a logarithmic singularity at the origin.

This class represents the differential one-form

$$L_0 = \frac{d\lambda}{\lambda}$$

### 6.9.2 Member Function Documentation

#### 6.9.2.1 series\_coeff\_impl()

```
cln::cl_N GiNaC::basic_log_kernel::series_coeff_impl (
    int i ) const    [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

#### 6.9.2.2 do\_print()

```
void GiNaC::basic_log_kernel::do_print (
    const print_context & c,
    unsigned level ) const    [protected]
```

References `c`.

The documentation for this class was generated from the following files:

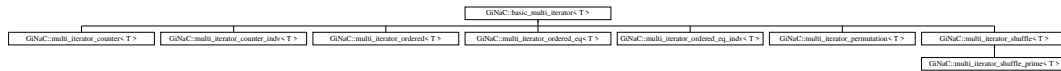
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.10 GiNaC::basic\_multi\_iterator< T > Class Template Reference

[basic\\_multi\\_iterator](#) is a base class.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::basic\_multi\_iterator< T >:



### Public Member Functions

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k.*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*
- virtual [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to*  

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$
- virtual [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*No effect for [basic\\_multi\\_iterator](#).*

### Protected Attributes

- T N
- T B
- std::vector< T > v
- bool [flag\\_overflow](#)

## Friends

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const basic_multi_iterator< TT > &v)`

### 6.10.1 Detailed Description

`template<class T>`  
`class GiNaC::basic_multi_iterator< T >`

`basic_multi_iterator` is a base class.

The base class itself does not do anything useful. A typical use of a class derived from `basic_multi_iterator` is

```
multi_iterator_ordered<int> k(0,4,2);
```

```
for( k.init(); !k.overflow(); k++) { std::cout << k << std::endl; }
```

which prints out

```
multi_iterator_ordered(0,1) multi_iterator_ordered(0,2) multi_iterator_ordered(0,3) multi_iterator_ordered(1,2)
multi_iterator_ordered(1,3) multi_iterator_ordered(2,3)
```

Individual components of `k` can be accessed with `k[i]` or `k(i)`.

All classes derived from `basic_multi_iterator` follow the same syntax.

### 6.10.2 Constructor & Destructor Documentation

#### 6.10.2.1 basic\_multi\_iterator() [1/3]

```
template<class T >
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
    void ) [inline]
```

Default constructor.

#### 6.10.2.2 basic\_multi\_iterator() [2/3]

```
template<class T >
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
    T B,
    T N,
    size_t k ) [inline], [explicit]
```

Construct a `multi_iterator` with upper limit `N`, lower limit `B` and size `k`.

**6.10.2.3 basic\_multi\_iterator()** [3/3]

```
template<class T >
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
    T B,
    T N,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

**6.10.2.4 ~basic\_multi\_iterator()**

```
template<class T >
GiNaC::basic_multi_iterator< T >::~~basic_multi_iterator ( ) [inline], [virtual]
```

Destructor.

**6.10.3 Member Function Documentation****6.10.3.1 size()**

```
template<class T >
size_t GiNaC::basic_multi_iterator< T >::size (
    void ) const [inline]
```

Returns the size of a multi\_iterator.

**6.10.3.2 overflow()**

```
template<class T >
bool GiNaC::basic_multi_iterator< T >::overflow (
    void ) const [inline]
```

Return the overflow flag.

**6.10.3.3 get\_vector()**

```
template<class T >
const std::vector< T > & GiNaC::basic_multi_iterator< T >::get_vector (
    void ) const [inline]
```

Returns a reference to the vector v.

**6.10.3.4 operator[]()** [1/2]

```
template<class T >
T GiNaC::basic_multi_iterator< T >::operator[] (
    size_t i ) const [inline]
```

Subscription via [].

**6.10.3.5 operator[]()** [2/2]

```
template<class T >
T & GiNaC::basic_multi_iterator< T >::operator[] (
    size_t i ) [inline]
```

Subscription via [].

**6.10.3.6 operator()()** [1/2]

```
template<class T >
T GiNaC::basic_multi_iterator< T >::operator() (
    size_t i ) const [inline]
```

Subscription via ().

**6.10.3.7 operator()()** [2/2]

```
template<class T >
T & GiNaC::basic_multi_iterator< T >::operator() (
    size_t i ) [inline]
```

Subscription via ().

**6.10.3.8 init()**

```
template<class T >
basic_multi_iterator< T > & GiNaC::basic_multi_iterator< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

.

Reimplemented in [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#), [GiNaC::multi\\_iterator\\_shuffle< T >](#), [GiNaC::multi\\_iterator\\_permutation< T >](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#), [GiNaC::multi\\_iterator\\_counter< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), and [GiNaC::multi\\_iterator\\_ordered< T >](#).

### 6.10.3.9 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::basic_multi_iterator< T >::operator++ (
    int ) [inline], [virtual]
```

No effect for [basic\\_multi\\_iterator](#).

Reimplemented in [GiNaC::multi\\_iterator\\_shuffle< T >](#), [GiNaC::multi\\_iterator\\_permutation< T >](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#), [GiNaC::multi\\_iterator\\_counter< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), and [GiNaC::multi\\_iterator\\_ordered< T >](#).

## 6.10.4 Friends And Related Function Documentation

### 6.10.4.1 operator<<

```
template<class T>
template<class TT >
std::ostream& operator<< (
    std::ostream & os,
    const basic_multi_iterator< TT > & v ) [friend]
```

## 6.10.5 Member Data Documentation

### 6.10.5.1 N

```
template<class T>
T GiNaC::basic_multi_iterator< T >::N [protected]
```

### 6.10.5.2 B

```
template<class T>
T GiNaC::basic_multi_iterator< T >::B [protected]
```

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#).

## 6.10.5.3 v

```
template<class T>
std::vector<T> GiNaC::basic_multi_iterator< T >::v [protected]
```

Referenced by `GiNaC::multi_iterator_shuffle< T >::multi_iterator_shuffle()`.

## 6.10.5.4 flag\_overflow

```
template<class T>
bool GiNaC::basic_multi_iterator< T >::flag_overflow [protected]
```

The documentation for this class was generated from the following file:

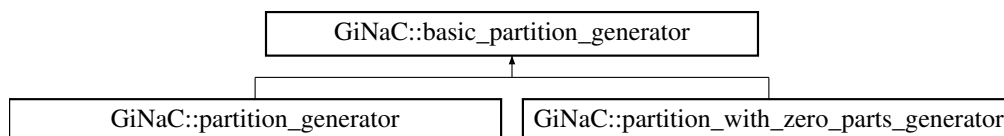
- [utils\\_multi\\_iterator.h](#)

## 6.11 GiNaC::basic\_partition\_generator Class Reference

Base class for generating all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts in non-decreasing order.

```
#include <utils.h>
```

Inheritance diagram for `GiNaC::basic_partition_generator`:



## Classes

- struct [mpartition2](#)

## Protected Member Functions

- [basic\\_partition\\_generator](#) (unsigned  $n$ \_, unsigned  $m$ \_)

## Protected Attributes

- [mpartition2](#) `mpgen`

### 6.11.1 Detailed Description

Base class for generating all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts in non-decreasing order.

### 6.11.2 Constructor & Destructor Documentation

#### 6.11.2.1 `basic_partition_generator()`

```
GiNaC::basic_partition_generator::basic_partition_generator (
    unsigned n_,
    unsigned m_ ) [inline], [protected]
```

### 6.11.3 Member Data Documentation

#### 6.11.3.1 `mpgen`

```
mpartition2 GiNaC::basic_partition_generator::mpgen [protected]
```

Referenced by `GiNaC::partition_with_zero_parts_generator::get()`, `GiNaC::partition_generator::get()`, `GiNaC::partition_with_zero_parts_generator::next()`, and `GiNaC::partition_generator::next()`.

The documentation for this class was generated from the following file:

- [utils.h](#)

## 6.12 `GiNaC::class_info< OPT >` Class Template Reference

```
#include <class_info.h>
```

### Classes

- struct [tree\\_node](#)

### Public Member Functions

- [class\\_info](#) (const OPT &o)
- [class\\_info](#) \* [get\\_parent](#) () const  
*Get pointer to [class\\_info](#) of parent class (or nullptr).*



## Static Public Member Functions

- static const [class\\_info](#) \* [find](#) (const std::string &class\_name)  
*Find [class\\_info](#) by name.*
- static void [dump\\_hierarchy](#) (bool verbose=false)  
*Dump class hierarchy to std::cout.*

## Public Attributes

- OPT [options](#)

## Static Private Member Functions

- static void [dump\\_tree](#) ([tree\\_node](#) \*n, const std::string &prefix, bool verbose)
- static void [identify\\_parents](#) ()

## Private Attributes

- [class\\_info](#) \* [next](#)
- [class\\_info](#) \* [parent](#)

## Static Private Attributes

- static [class\\_info](#) \* [first](#) = nullptr
- static bool [parents\\_identified](#) = false

## 6.12.1 Constructor & Destructor Documentation

### 6.12.1.1 class\_info()

```
template<class OPT>
GiNaC::class_info< OPT >::class_info (
    const OPT & o ) [inline]
```

References [GiNaC::class\\_info< OPT >::first](#), and [GiNaC::class\\_info< OPT >::parents\\_identified](#).

## 6.12.2 Member Function Documentation

#### 6.12.2.1 get\_parent()

```
template<class OPT>
class_info* GiNaC::class_info< OPT >::get_parent ( ) const [inline]
```

Get pointer to [class\\_info](#) of parent class (or nullptr).

References [GiNaC::class\\_info< OPT >::identify\\_parents\(\)](#), and [GiNaC::class\\_info< OPT >::parent](#).

Referenced by [GiNaC::class\\_info< OPT >::dump\\_hierarchy\(\)](#), [GiNaC::function::print\(\)](#), and [GiNaC::basic::print\\_↵  
dispatch\(\)](#).

#### 6.12.2.2 find()

```
template<class OPT >
const class_info< OPT > * GiNaC::class_info< OPT >::find (
    const std::string & class_name ) [static]
```

Find [class\\_info](#) by name.

References [GiNaC::class\\_info< OPT >::find\(\)](#), [GiNaC::class\\_info< OPT >::next](#), and [GiNaC::class\\_info< OPT >::options](#).

Referenced by [GiNaC::class\\_info< OPT >::find\(\)](#).

#### 6.12.2.3 dump\_hierarchy()

```
template<class OPT >
void GiNaC::class_info< OPT >::dump_hierarchy (
    bool verbose = false ) [static]
```

Dump class hierarchy to `std::cout`.

References [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#), [GiNaC::class\\_info< OPT >::next](#), and [GiNaC::tree\(\)](#).

#### 6.12.2.4 dump\_tree()

```
template<class OPT >
void GiNaC::class_info< OPT >::dump_tree (
    tree_node * n,
    const std::string & prefix,
    bool verbose ) [static], [private]
```

References `n`.

#### 6.12.2.5 identify\_parents()

```
template<class OPT >
void GiNaC::class_info< OPT >::identify_parents ( ) [static], [private]
```

References GiNaC::class\_info< OPT >::next.

Referenced by GiNaC::class\_info< OPT >::get\_parent().

### 6.12.3 Member Data Documentation

#### 6.12.3.1 options

```
template<class OPT>
OPT GiNaC::class_info< OPT >::options
```

Referenced by GiNaC::class\_info< OPT >::find(), GiNaC::function::print(), and GiNaC::basic::print\_dispatch().

#### 6.12.3.2 first

```
template<class OPT>
class_info< OPT > * GiNaC::class_info< OPT >::first = nullptr [static], [private]
```

Referenced by GiNaC::class\_info< OPT >::class\_info().

#### 6.12.3.3 next

```
template<class OPT>
class_info* GiNaC::class_info< OPT >::next [private]
```

Referenced by GiNaC::class\_info< OPT >::dump\_hierarchy(), GiNaC::class\_info< OPT >::find(), and GiNaC::class\_info< OPT >::identify\_parents().

#### 6.12.3.4 parent

```
template<class OPT>
class_info* GiNaC::class_info< OPT >::parent [mutable], [private]
```

Referenced by GiNaC::class\_info< OPT >::get\_parent().

### 6.12.3.5 parents\_identified

```
template<class OPT>
bool GiNaC::class_info< OPT >::parents_identified = false [static], [private]
```

Referenced by `GiNaC::class_info< OPT >::class_info()`.

The documentation for this class was generated from the following file:

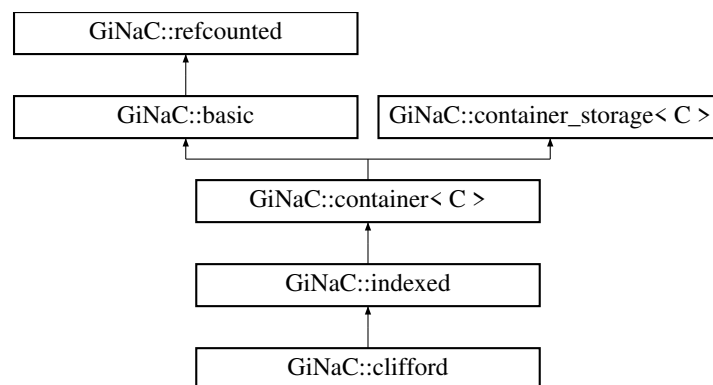
- [class\\_info.h](#)

## 6.13 GiNaC::clifford Class Reference

This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).

```
#include <clifford.h>
```

Inheritance diagram for `GiNaC::clifford`:



### Public Member Functions

- `clifford` (const `ex` &b, unsigned char rl=0)  
*Construct object without any indices.*
- `clifford` (const `ex` &b, const `ex` &mu, const `ex` &metr, unsigned char rl=0, int comm\_sign=-1)  
*Construct object with one Lorentz index.*
- `clifford` (unsigned char rl, const `ex` &metr, int comm\_sign, const `exvector` &v)
- `clifford` (unsigned char rl, const `ex` &metr, int comm\_sign, `exvector` &&v)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthesizing output).*
- void `archive` (`archive_node` &n) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &n, `lst` &sym\_lst) override  
*Load (deserialize) the object from an archive node.*
- unsigned char `get_representation_label` () const
- `ex` `get_metric` () const
- virtual `ex` `get_metric` (const `ex` &i, const `ex` &j, bool symmetrised=false) const
- bool `same_metric` (const `ex` &other) const

- `int get_commutator_sign ()` const
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position i.*
- `ex & let_op (size_t i)` override  
*Return modifiable operand/member at position i.*
- `ex subs (const exmap &m, unsigned options=0)` const override  
*Substitute a set of objects by arbitrary expressions.*

## Protected Member Functions

- `ex eval_ncmul (const exvector &v)` const override  
*Perform automatic simplification on noncommutative product of clifford objects.*
- `bool match_same_type (const basic &other)` const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `ex thiscontainer (const exvector &v)` const override
- `ex thiscontainer (exvector &&v)` const override
- `unsigned return_type ()` const override
- `return_type_t return_type_tinfo ()` const override
- `void do_print_dflt (const print_dflt &c, unsigned level)` const
- `void do_print_latex (const print_latex &c, unsigned level)` const
- `void do_print_tree (const print_tree &c, unsigned level)` const

## Protected Attributes

- `unsigned char representation_label`  
*Representation label to distinguish independent spin lines.*
- `ex metric`  
*Metric of the space, all constructors make it an indexed object.*
- `int commutator_sign`  
*It is the sign in the definition  $e^{\sim i} e^{\sim j} +/ - e^{\sim j} e^{\sim i} = B(i, j) + B(j, i)$*

## Additional Inherited Members

### 6.13.1 Detailed Description

This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).

These objects only carry Lorentz indices. Spinor indices are hidden. A representation label (an unsigned 8-bit integer) is used to distinguish elements from different Clifford algebras (objects with different labels commute).

### 6.13.2 Constructor & Destructor Documentation

### 6.13.2.1 `clifford()` [1/4]

```
GiNaC::clifford::clifford (
    const ex & b,
    unsigned char rl = 0 )
```

Construct object without any indices.

This constructor is for internal use only. Use the [dirac\\_ONE\(\)](#) function instead.

See also

[dirac\\_ONE](#)

### 6.13.2.2 `clifford()` [2/4]

```
GiNaC::clifford::clifford (
    const ex & b,
    const ex & mu,
    const ex & metr,
    unsigned char rl = 0,
    int comm_sign = -1 )
```

Construct object with one Lorentz index.

This constructor is for internal use only. Use the [clifford\\_unit\(\)](#) or [dirac\\_gamma\(\)](#) functions instead.

See also

[clifford\\_unit](#)  
[dirac\\_gamma](#)

References GINAC\_ASSERT.

### 6.13.2.3 `clifford()` [3/4]

```
GiNaC::clifford::clifford (
    unsigned char rl,
    const ex & metr,
    int comm_sign,
    const exvector & v )
```

#### 6.13.2.4 clifford() [4/4]

```
GiNaC::clifford::clifford (
    unsigned char rl,
    const ex & metr,
    int comm_sign,
    exvector && v )
```

### 6.13.3 Member Function Documentation

#### 6.13.3.1 precedence()

```
unsigned GiNaC::clifford::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by `do_print_dflt()`, and `do_print_latex()`.

#### 6.13.3.2 archive()

```
void GiNaC::clifford::archive (
    archive_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References `commutator_sign`, `metric`, `n`, and `representation_label`.

### 6.13.3.3 read\_archive()

```
void GiNaC::clifford::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References `commutator_sign`, `metric`, `n`, and `representation_label`.

### 6.13.3.4 eval\_ncmul()

```
ex GiNaC::clifford::eval_ncmul (
    const exvector & v ) const [override], [protected], [virtual]
```

Perform automatic simplification on noncommutative product of clifford objects.

This removes superfluous ONEs, permutes gamma5/L/R's to the front and removes squares of gamma objects.

Reimplemented from [GiNaC::basic](#).

### 6.13.3.5 match\_same\_type()

```
bool GiNaC::clifford::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

#### See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References `commutator_sign`, `get_commutator_sign()`, `GINAC_ASSERT`, `representation_label`, and `same_metric()`.



#### 6.13.3.6 thiscontainer() [1/2]

```
ex GiNaC::clifford::thiscontainer (
    const exvector & v ) const [override], [protected]
```

#### 6.13.3.7 thiscontainer() [2/2]

```
ex GiNaC::clifford::thiscontainer (
    exvector && v ) const [override], [protected]
```

#### 6.13.3.8 return\_type()

```
unsigned GiNaC::clifford::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#).

#### 6.13.3.9 return\_type\_tinfo()

```
return\_type\_t GiNaC::clifford::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [representation\\_label](#).

#### 6.13.3.10 get\_representation\_label()

```
unsigned char GiNaC::clifford::get_representation_label ( ) const [inline]
```

References [representation\\_label](#).

#### 6.13.3.11 get\_metric() [1/2]

```
ex GiNaC::clifford::get_metric ( ) const [inline]
```

References [metric](#).

Referenced by [same\\_metric\(\)](#).

#### 6.13.3.12 `get_metric()` [2/2]

```
ex GiNaC::clifford::get_metric (
    const ex & i,
    const ex & j,
    bool symmetrised = false ) const [virtual]
```

References `GiNaC::_ex1_2`, `GiNaC::ex::get_free_indices()`, `GiNaC::indexed::get_symmetry()`, `GiNaC::indexed::indexed()`, `metric`, `GiNaC::subs_options::no_pattern`, `GiNaC::ex::op()`, `GiNaC::indexed::simplify_indexed`, `GiNaC::ex::subs()`, `GiNaC::symmetric2()`, and `GiNaC::transpose()`.

#### 6.13.3.13 `same_metric()`

```
bool GiNaC::clifford::same_metric (
    const ex & other ) const
```

References `GiNaC::ex::get_free_indices()`, `get_metric()`, `GiNaC::ex::is_equal()`, `GiNaC::is_zero()`, `op()`, `GiNaC::ex::op()`, and `GiNaC::indexed::simplify_indexed`.

Referenced by `match_same_type()`.

#### 6.13.3.14 `get_commutator_sign()`

```
int GiNaC::clifford::get_commutator_sign ( ) const [inline]
```

References `commutator_sign`.

Referenced by `match_same_type()`.

#### 6.13.3.15 `nops()`

```
size_t GiNaC::clifford::nops ( ) const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

References `GiNaC::nops()`.

Referenced by `let_op()`, and `op()`.

## 6.13.3.16 op()

```
ex GiNaC::clifford::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References GINAC\_ASSERT, nops(), GiNaC::op(), and representation\_label.

Referenced by same\_metric().

## 6.13.3.17 let\_op()

```
ex & GiNaC::clifford::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References GiNaC::basic::ensure\_if\_modifiable(), GINAC\_ASSERT, nops(), and representation\_label.

## 6.13.3.18 subs()

```
ex GiNaC::clifford::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References GiNaC::are\_ex\_trivially\_equal(), c, m, metric, options, GiNaC::ex::subs(), and GiNaC::subs().

## 6.13.3.19 do\_print\_dflt()

```
void GiNaC::clifford::do_print_dflt (
    const print_dflt & c,
    unsigned level ) const [protected]
```

References c, GiNaC::is\_dirac\_slash(), precedence(), GiNaC::indexed::printindices(), representation\_label, and GiNaC::container\_storage< C >::seq.

#### 6.13.3.20 do\_print\_latex()

```
void GiNaC::clifford::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::is\\_dirac\\_slash\(\)](#), [precedence\(\)](#), [representation\\_label](#), and [GiNaC::container\\_storage< C >::seq](#).

#### 6.13.3.21 do\_print\_tree()

```
void GiNaC::clifford::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [metric](#), [GiNaC::ex::print\(\)](#), [GiNaC::indexed< T >::printindices\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::indexed::symtree](#).

### 6.13.4 Member Data Documentation

#### 6.13.4.1 representation\_label

```
unsigned char GiNaC::clifford::representation_label [protected]
```

Representation label to distinguish independent spin lines.

Referenced by [archive\(\)](#), [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), [get\\_representation\\_label\(\)](#), [let\\_op\(\)](#), [match\\_same\\_type\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [return\\_type\\_tinfo\(\)](#).

#### 6.13.4.2 metric

```
ex GiNaC::clifford::metric [protected]
```

Metric of the space, all constructors make it an indexed object.

Referenced by [archive\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_metric\(\)](#), [read\\_archive\(\)](#), and [subs\(\)](#).

## 6.13.4.3 commutator\_sign

```
int GiNaC::clifford::commutator_sign [protected]
```

It is the sign in the definition  $e_i e_j \pm e_j e_i = B(i, j) + B(j, i)$

Referenced by `archive()`, `get_commutator_sign()`, `match_same_type()`, and `read_archive()`.

The documentation for this class was generated from the following files:

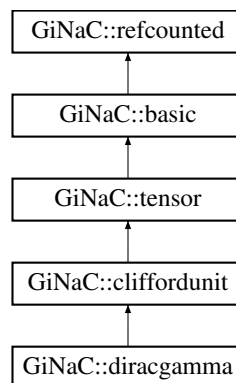
- [clifford.h](#)
- [clifford.cpp](#)

## 6.14 GiNaC::cliffordunit Class Reference

This class represents the Clifford algebra generators (units).

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::cliffordunit:



## Public Member Functions

- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of a Clifford unit with something else.*

## Protected Member Functions

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

## Additional Inherited Members

## 6.14.1 Detailed Description

This class represents the Clifford algebra generators (units).

## 6.14.2 Member Function Documentation

### 6.14.2.1 `contract_with()`

```
bool GiNaC::cliffordunit::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of a Clifford unit with something else.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::diracgamma](#).

### 6.14.2.2 `do_print()`

```
void GiNaC::cliffordunit::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

### 6.14.2.3 `do_print_latex()`

```
void GiNaC::cliffordunit::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

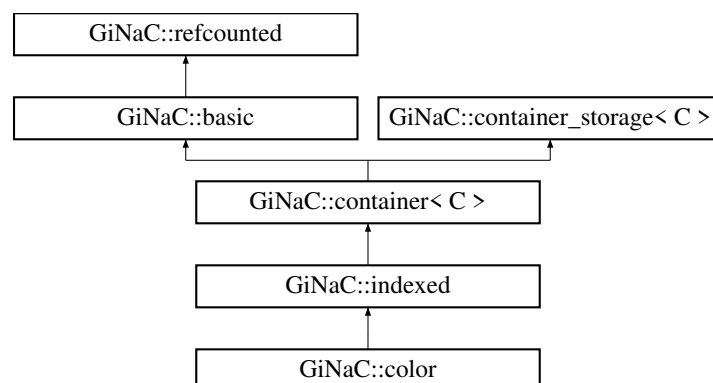
- [clifford.h](#)
- [clifford.cpp](#)

## 6.15 GiNaC::color Class Reference

This class holds a generator  $T_a$  or the unity element of the Lie algebra of  $SU(3)$ , as used for calculations in quantum chromodynamics.

```
#include <color.h>
```

Inheritance diagram for `GiNaC::color`:



## Public Member Functions

- `color` (const `ex` &b, unsigned char rl=0)  
*Construct object without any color index.*
- `color` (const `ex` &b, const `ex` &i1, unsigned char rl=0)  
*Construct object with one color index.*
- `color` (unsigned char rl, const `exvector` &v)
- `color` (unsigned char rl, `exvector` &&v)
- void `archive` (`archive_node` &n) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &n, `lst` &sym\_lst) override  
*Load (deserialize) the object from an archive node.*
- unsigned char `get_representation_label` () const

## Protected Member Functions

- `ex eval_ncmul` (const `exvector` &v) const override  
*Perform automatic simplification on noncommutative product of color objects.*
- bool `match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `ex thiscontainer` (const `exvector` &v) const override
- `ex thiscontainer` (`exvector` &&v) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override

## Private Attributes

- unsigned char `representation_label`  
*Representation label to distinguish independent color matrices coming from separated fermion lines.*

## Additional Inherited Members

### 6.15.1 Detailed Description

This class holds a generator `T_a` or the unity element of the Lie algebra of  $SU(3)$ , as used for calculations in quantum chromodynamics.

A representation label (an unsigned 8-bit integer) is used to distinguish elements from different Lie algebras (objects with different labels commute). These objects implement an abstract representation of the group, not a specific matrix representation. The indices used for color objects should not have a variance.

### 6.15.2 Constructor & Destructor Documentation

**6.15.2.1 color()** [1/4]

```
GiNaC::color::color (
    const ex & b,
    unsigned char rl = 0 )
```

Construct object without any color index.

This constructor is for internal use only. Use the [color\\_ONE\(\)](#) function instead.

See also

[color\\_ONE](#)

Referenced by [thiscontainer\(\)](#).

**6.15.2.2 color()** [2/4]

```
GiNaC::color::color (
    const ex & b,
    const ex & il,
    unsigned char rl = 0 )
```

Construct object with one color index.

This constructor is for internal use only. Use the [color\\_T\(\)](#) function instead.

See also

[color\\_T](#)

**6.15.2.3 color()** [3/4]

```
GiNaC::color::color (
    unsigned char rl,
    const exvector & v )
```

**6.15.2.4 color()** [4/4]

```
GiNaC::color::color (
    unsigned char rl,
    exvector && v )
```



### 6.15.3 Member Function Documentation

#### 6.15.3.1 archive()

```
void GiNaC::color::archive (
    archive_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References `n`, and `representation_label`.

#### 6.15.3.2 read\_archive()

```
void GiNaC::color::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References `n`, and `representation_label`.

#### 6.15.3.3 eval\_ncmul()

```
ex GiNaC::color::eval_ncmul (
    const exvector & v ) const [override], [protected], [virtual]
```

Perform automatic simplification on noncommutative product of color objects.

This removes superfluous ONES.

Reimplemented from [GiNaC::basic](#).

References `GiNaC::hold_ncmul()`, and `GiNaC::container_storage< C >::reserve()`.

#### 6.15.3.4 `match_same_type()`

```
bool GiNaC::color::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References `GINAC_ASSERT`, and `representation_label`.

#### 6.15.3.5 `thiscontainer()` [1/2]

```
ex GiNaC::color::thiscontainer (
    const exvector & v ) const [override], [protected]
```

References `color()`, and `representation_label`.

#### 6.15.3.6 `thiscontainer()` [2/2]

```
ex GiNaC::color::thiscontainer (
    exvector && v ) const [override], [protected]
```

References `color()`, and `representation_label`.

#### 6.15.3.7 `return_type()`

```
unsigned GiNaC::color::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References `GiNaC::return_types::noncommutative`.

## 6.15.3.8 return\_type\_tinfo()

```
return_type_t GiNaC::color::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [representation\\_label](#).

## 6.15.3.9 get\_representation\_label()

```
unsigned char GiNaC::color::get_representation_label ( ) const [inline]
```

References [representation\\_label](#).

## 6.15.4 Member Data Documentation

## 6.15.4.1 representation\_label

```
unsigned char GiNaC::color::representation_label [private]
```

Representation label to distinguish independent color matrices coming from separated fermion lines.

Referenced by [archive\(\)](#), [get\\_representation\\_label\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [return\\_type\\_tinfo\(\)](#), and [thiscontainer\(\)](#).

The documentation for this class was generated from the following files:

- [color.h](#)
- [color.cpp](#)

## 6.16 GiNaC::compare\_all\_equal&lt; T &gt; Class Template Reference

Comparison policy: all structures of one type are equal.

```
#include <structure.h>
```

## Protected Member Functions

- [~compare\\_all\\_equal\(\)](#)

## Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

### 6.16.1 Detailed Description

```
template<class T>
class GiNaC::compare_all_equal< T >
```

Comparison policy: all structures of one type are equal.

### 6.16.2 Constructor & Destructor Documentation

#### 6.16.2.1 ~compare\_all\_equal()

```
template<class T >
GiNaC::compare_all_equal< T >::~~compare_all_equal ( ) [inline], [protected]
```

### 6.16.3 Member Function Documentation

#### 6.16.3.1 struct\_is\_equal()

```
template<class T >
static bool GiNaC::compare_all_equal< T >::struct_is_equal (
    const T * t1,
    const T * t2 ) [inline], [static], [protected]
```

#### 6.16.3.2 struct\_compare()

```
template<class T >
static int GiNaC::compare_all_equal< T >::struct_compare (
    const T * t1,
    const T * t2 ) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

- [structure.h](#)

## 6.17 GiNaC::compare\_bitwise< T > Class Template Reference

Comparison policy: use bit-wise comparison to compare structures.

```
#include <structure.h>
```

### Protected Member Functions

- [~compare\\_bitwise\(\)](#)

### Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

#### 6.17.1 Detailed Description

```
template<class T>
class GiNaC::compare_bitwise< T >
```

Comparison policy: use bit-wise comparison to compare structures.

#### 6.17.2 Constructor & Destructor Documentation

##### 6.17.2.1 ~compare\_bitwise()

```
template<class T >
GiNaC::compare_bitwise< T >::~~compare_bitwise ( ) [inline], [protected]
```

#### 6.17.3 Member Function Documentation

##### 6.17.3.1 struct\_is\_equal()

```
template<class T >
static bool GiNaC::compare_bitwise< T >::struct_is_equal (
    const T * t1,
    const T * t2 ) [inline], [static], [protected]
```

### 6.17.3.2 struct\_compare()

```
template<class T >
static int GiNaC::compare_bitwise< T >::struct_compare (
    const T * t1,
    const T * t2 ) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

- [structure.h](#)

## 6.18 GiNaC::compare\_std\_less< T > Class Template Reference

Comparison policy: use std::equal\_to/std::less (defaults to operators == and <) to compare structures.

```
#include <structure.h>
```

### Protected Member Functions

- [~compare\\_std\\_less\(\)](#)

### Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

### 6.18.1 Detailed Description

```
template<class T>
class GiNaC::compare_std_less< T >
```

Comparison policy: use std::equal\_to/std::less (defaults to operators == and <) to compare structures.

### 6.18.2 Constructor & Destructor Documentation

#### 6.18.2.1 ~compare\_std\_less()

```
template<class T >
GiNaC::compare_std_less< T >::~~compare_std_less ( ) [inline], [protected]
```

### 6.18.3 Member Function Documentation

## 6.18.3.1 struct\_is\_equal()

```
template<class T >
static bool GiNaC::compare_std_less< T >::struct_is_equal (
    const T * t1,
    const T * t2 ) [inline], [static], [protected]
```

## 6.18.3.2 struct\_compare()

```
template<class T >
static int GiNaC::compare_std_less< T >::struct_compare (
    const T * t1,
    const T * t2 ) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

- [structure.h](#)

## 6.19 GiNaC::composition\_generator Class Reference

Generate all compositions of a partition of an integer n, starting with the compositions which has non-decreasing order.

```
#include <utils.h>
```

## Classes

- struct [coolmulti](#)

## Public Member Functions

- [composition\\_generator](#) (const std::vector< unsigned > &partition)
- const std::vector< unsigned > & [get](#) () const
- bool [next](#) ()

## Private Attributes

- struct [GiNaC::composition\\_generator::coolmulti](#) cmgen
- bool [atend](#)
- bool [trivial](#)
- std::vector< unsigned > [composition](#)
- bool [current\\_updated](#)

### 6.19.1 Detailed Description

Generate all compositions of a partition of an integer  $n$ , starting with the compositions which has non-decreasing order.

### 6.19.2 Constructor & Destructor Documentation

#### 6.19.2.1 composition\_generator()

```
GiNaC::composition_generator::composition_generator (
    const std::vector< unsigned > & partition ) [inline], [explicit]
```

References trivial.

### 6.19.3 Member Function Documentation

#### 6.19.3.1 get()

```
const std::vector<unsigned>& GiNaC::composition_generator::get ( ) const [inline]
```

References `cmgen`, `composition`, `current_updated`, `GiNaC::composition_generator::coolmulti::head`, `GiNaC::composition_generator::coolmulti::element::next`, and `GiNaC::composition_generator::coolmulti::element::value`.

Referenced by `GiNaC::power::expand_add()`.

#### 6.19.3.2 next()

```
bool GiNaC::composition_generator::next ( ) [inline]
```

References `atend`, `cmgen`, `current_updated`, `GiNaC::composition_generator::coolmulti::finished()`, `GiNaC::composition_generator::coolmulti::next_permutation()`, and `trivial`.

Referenced by `GiNaC::power::expand_add()`.

### 6.19.4 Member Data Documentation



#### 6.19.4.1 cmgen

```
struct GiNaC::composition_generator::coolmulti GiNaC::composition_generator::cmgen [private]
```

Referenced by `get()`, and `next()`.

#### 6.19.4.2 atend

```
bool GiNaC::composition_generator::atend [private]
```

Referenced by `next()`.

#### 6.19.4.3 trivial

```
bool GiNaC::composition_generator::trivial [private]
```

Referenced by `composition_generator()`, and `next()`.

#### 6.19.4.4 composition

```
std::vector<unsigned> GiNaC::composition_generator::composition [mutable], [private]
```

Referenced by `get()`.

#### 6.19.4.5 current\_updated

```
bool GiNaC::composition_generator::current_updated [mutable], [private]
```

Referenced by `get()`, and `next()`.

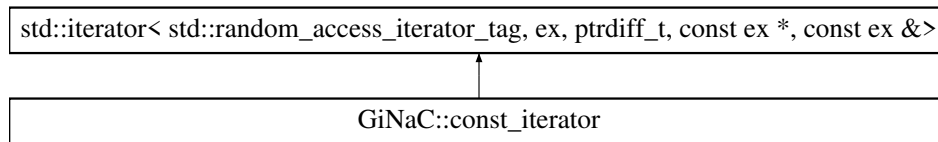
The documentation for this class was generated from the following file:

- [utils.h](#)

## 6.20 GiNaC::const\_iterator Class Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::const\_iterator:



### Public Member Functions

- `const_iterator ()` noexcept
- `ex operator*` () const
- `std::unique_ptr< ex > operator->` () const
- `ex operator[]` (difference\_type `n`) const
- `const_iterator & operator++` () noexcept
- `const_iterator operator++` (int) noexcept
- `const_iterator & operator+=` (difference\_type `n`) noexcept
- `const_iterator operator+` (difference\_type `n`) const noexcept
- `const_iterator & operator--` () noexcept
- `const_iterator operator--` (int) noexcept
- `const_iterator & operator-=` (difference\_type `n`) noexcept
- `const_iterator operator-` (difference\_type `n`) const noexcept
- `bool operator==` (const `const_iterator` &other) const noexcept
- `bool operator!=` (const `const_iterator` &other) const noexcept
- `bool operator<` (const `const_iterator` &other) const noexcept
- `bool operator>` (const `const_iterator` &other) const noexcept
- `bool operator<=` (const `const_iterator` &other) const noexcept
- `bool operator>=` (const `const_iterator` &other) const noexcept

### Protected Attributes

- `ex e`
- `size_t i`

### Private Member Functions

- `const_iterator` (const `ex` &`e_`, `size_t i_`) noexcept

### Friends

- class `ex`
- class `const_preorder_iterator`
- class `const_postorder_iterator`
- `const_iterator operator+` (difference\_type `n`, const `const_iterator` &`it`) noexcept
- difference\_type `operator-` (const `const_iterator` &`lhs`, const `const_iterator` &`rhs`) noexcept

## 6.20.1 Constructor & Destructor Documentation

### 6.20.1.1 const\_iterator() [1/2]

```
GiNaC::const_iterator::const_iterator ( ) [inline], [noexcept]
```

Referenced by operator+(), and operator-().

### 6.20.1.2 const\_iterator() [2/2]

```
GiNaC::const_iterator::const_iterator (
    const ex & e_,
    size_t i_ ) [inline], [private], [noexcept]
```

## 6.20.2 Member Function Documentation

### 6.20.2.1 operator\*()

```
ex GiNaC::const_iterator::operator* ( ) const [inline]
```

References [e](#), [i](#), and [GiNaC::ex::op\(\)](#).

### 6.20.2.2 operator->()

```
std::unique_ptr<ex> GiNaC::const_iterator::operator-> ( ) const [inline]
```

References [ex](#).

### 6.20.2.3 operator[]()

```
ex GiNaC::const_iterator::operator[] (
    difference_type n ) const [inline]
```

References [e](#), [i](#), [n](#), and [GiNaC::ex::op\(\)](#).

#### 6.20.2.4 operator++() [1/2]

```
const_iterator& GiNaC::const_iterator::operator++ ( ) [inline], [noexcept]
```

References i.

#### 6.20.2.5 operator++() [2/2]

```
const_iterator GiNaC::const_iterator::operator++ (
    int ) [inline], [noexcept]
```

References i.

#### 6.20.2.6 operator+=( )

```
const_iterator& GiNaC::const_iterator::operator+= (
    difference_type n ) [inline], [noexcept]
```

References i, and n.

#### 6.20.2.7 operator+( )

```
const_iterator GiNaC::const_iterator::operator+ (
    difference_type n ) const [inline], [noexcept]
```

References const\_iterator(), e, i, and n.

#### 6.20.2.8 operator--() [1/2]

```
const_iterator& GiNaC::const_iterator::operator-- ( ) [inline], [noexcept]
```

References i.

#### 6.20.2.9 operator--() [2/2]

```
const_iterator GiNaC::const_iterator::operator-- (
    int ) [inline], [noexcept]
```

References i.

#### 6.20.2.10 operator-=( )

```
const_iterator& GiNaC::const_iterator::operator-= (
    difference_type n ) [inline], [noexcept]
```

References `i`, and `n`.

#### 6.20.2.11 operator-( )

```
const_iterator GiNaC::const_iterator::operator- (
    difference_type n ) const [inline], [noexcept]
```

References `const_iterator()`, `e`, `i`, and `n`.

#### 6.20.2.12 operator==( )

```
bool GiNaC::const_iterator::operator== (
    const const_iterator & other ) const [inline], [noexcept]
```

References `GiNaC::are_ex_trivially_equal()`, `e`, and `i`.

#### 6.20.2.13 operator!=( )

```
bool GiNaC::const_iterator::operator!= (
    const const_iterator & other ) const [inline], [noexcept]
```

#### 6.20.2.14 operator<( )

```
bool GiNaC::const_iterator::operator< (
    const const_iterator & other ) const [inline], [noexcept]
```

References `i`.

#### 6.20.2.15 operator>( )

```
bool GiNaC::const_iterator::operator> (
    const const_iterator & other ) const [inline], [noexcept]
```

#### 6.20.2.16 operator<=()

```
bool GiNaC::const_iterator::operator<= (
    const const\_iterator & other ) const [inline], [noexcept]
```

#### 6.20.2.17 operator>=()

```
bool GiNaC::const_iterator::operator>= (
    const const\_iterator & other ) const [inline], [noexcept]
```

### 6.20.3 Friends And Related Function Documentation

#### 6.20.3.1 ex

```
friend class ex [friend]
```

Referenced by operator->().

#### 6.20.3.2 const\_preorder\_iterator

```
friend class const\_preorder\_iterator [friend]
```

#### 6.20.3.3 const\_postorder\_iterator

```
friend class const\_postorder\_iterator [friend]
```

#### 6.20.3.4 operator+

```
const\_iterator operator+ (
    difference_type n,
    const const\_iterator & it ) [friend]
```

## 6.20.3.5 operator-

```
difference_type operator- (
    const const\_iterator & lhs,
    const const\_iterator & rhs ) [friend]
```

## 6.20.4 Member Data Documentation

## 6.20.4.1 e

```
ex GiNaC::const_iterator::e [protected]
```

Referenced by [operator\\*\(\)](#), [operator+\(\)](#), [operator-\(\)](#), [operator==\(\)](#), and [operator\[\]\(\)](#).

## 6.20.4.2 i

```
size_t GiNaC::const_iterator::i [protected]
```

Referenced by [operator\\*\(\)](#), [operator+\(\)](#), [operator++\(\)](#), [operator+=\(\)](#), [operator-\(\)](#), [operator--\(\)](#), [operator-=\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [operator\[\]\(\)](#).

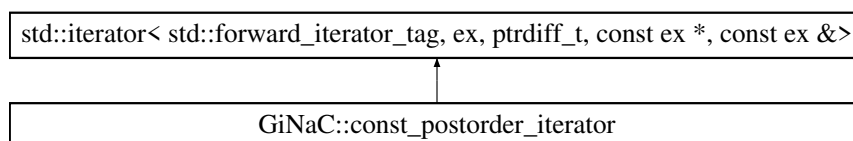
The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.21 GiNaC::const\_postorder\_iterator Class Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::const\_postorder\_iterator:



## Public Member Functions

- [const\\_postorder\\_iterator](#) () noexcept
- [const\\_postorder\\_iterator](#) (const [ex](#) &e, size\_t n)
- reference [operator\\*](#) () const
- pointer [operator->](#) () const
- [const\\_postorder\\_iterator](#) & [operator++](#) ()
- [const\\_postorder\\_iterator](#) [operator++](#) (int)
- bool [operator==](#) (const [const\\_postorder\\_iterator](#) &other) const noexcept
- bool [operator!=](#) (const [const\\_postorder\\_iterator](#) &other) const noexcept

## Private Member Functions

- void [descend](#) ()
- void [increment](#) ()

## Private Attributes

- `std::stack< internal::\_iter\_rep, std::vector< internal::\_iter\_rep > > s`

## 6.21.1 Constructor & Destructor Documentation

### 6.21.1.1 `const_postorder_iterator()` [1/2]

`GiNaC::const_postorder_iterator::const_postorder_iterator ( ) [inline], [noexcept]`

### 6.21.1.2 `const_postorder_iterator()` [2/2]

`GiNaC::const_postorder_iterator::const_postorder_iterator (`  
     `const ex & e,`  
     `size_t n ) [inline]`

References `descend()`, `n`, and `s`.

## 6.21.2 Member Function Documentation

### 6.21.2.1 `operator*()`

`reference GiNaC::const_postorder_iterator::operator* ( ) const [inline]`

References `s`.

### 6.21.2.2 `operator->()`

`pointer GiNaC::const_postorder_iterator::operator-> ( ) const [inline]`

References `s`.



**6.21.2.3 operator++()** [1/2]

```
const_postorder_iterator& GiNaC::const_postorder_iterator::operator++ ( ) [inline]
```

References increment().

**6.21.2.4 operator++()** [2/2]

```
const_postorder_iterator GiNaC::const_postorder_iterator::operator++ (
    int ) [inline]
```

References increment().

**6.21.2.5 operator==( )**

```
bool GiNaC::const_postorder_iterator::operator== (
    const const_postorder_iterator & other ) const [inline], [noexcept]
```

References s.

**6.21.2.6 operator!=( )**

```
bool GiNaC::const_postorder_iterator::operator!= (
    const const_postorder_iterator & other ) const [inline], [noexcept]
```

**6.21.2.7 descend()**

```
void GiNaC::const_postorder_iterator::descend ( ) [inline], [private]
```

References GiNaC::internal::\_iter\_rep::e, GiNaC::internal::\_iter\_rep::i, GiNaC::ex::nops(), GiNaC::ex::op(), and s.

Referenced by const\_postorder\_iterator(), and increment().

**6.21.2.8 increment()**

```
void GiNaC::const_postorder_iterator::increment ( ) [inline], [private]
```

References descend(), and s.

Referenced by operator++().

### 6.21.3 Member Data Documentation

#### 6.21.3.1 s

```
std::stack<internal::_iter_rep, std::vector<internal::_iter_rep> > GiNaC::const_postorder_iterator::s [private]
```

Referenced by `const_postorder_iterator()`, `descend()`, `increment()`, `operator*()`, `operator->()`, and `operator==()`.

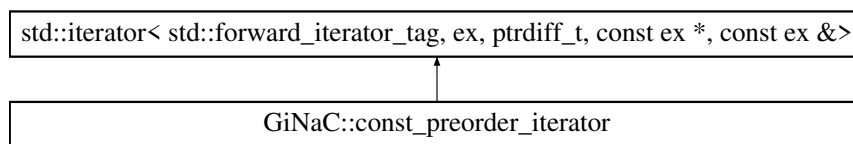
The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.22 GiNaC::const\_preorder\_iterator Class Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::const_preorder_iterator`:



### Public Member Functions

- [const\\_preorder\\_iterator](#) () noexcept
- [const\\_preorder\\_iterator](#) (const [ex](#) &*e*, [size\\_t](#) *n*)
- reference [operator\\*](#) () const
- pointer [operator->](#) () const
- [const\\_preorder\\_iterator](#) & [operator++](#) ()
- [const\\_preorder\\_iterator](#) [operator++](#) (int)
- bool [operator==](#) (const [const\\_preorder\\_iterator](#) &*other*) const noexcept
- bool [operator!=](#) (const [const\\_preorder\\_iterator](#) &*other*) const noexcept

### Private Member Functions

- void [increment](#) ()

### Private Attributes

- `std::stack< internal::\_iter\_rep, std::vector< internal::\_iter\_rep > > s`

## 6.22.1 Constructor & Destructor Documentation

### 6.22.1.1 const\_preorder\_iterator() [1/2]

```
GiNaC::const_preorder_iterator::const_preorder_iterator ( ) [inline], [noexcept]
```

### 6.22.1.2 const\_preorder\_iterator() [2/2]

```
GiNaC::const_preorder_iterator::const_preorder_iterator (
    const ex & e,
    size_t n ) [inline]
```

References [n](#), and [s](#).

## 6.22.2 Member Function Documentation

### 6.22.2.1 operator\*()

```
reference GiNaC::const_preorder_iterator::operator* ( ) const [inline]
```

References [s](#).

### 6.22.2.2 operator->()

```
pointer GiNaC::const_preorder_iterator::operator-> ( ) const [inline]
```

References [s](#).

### 6.22.2.3 operator++() [1/2]

```
const\_preorder\_iterator& GiNaC::const_preorder_iterator::operator++ ( ) [inline]
```

References [increment\(\)](#).

#### 6.22.2.4 `operator++()` [2/2]

```
const_preorder_iterator GiNaC::const_preorder_iterator::operator++ (
    int ) [inline]
```

References `increment()`.

#### 6.22.2.5 `operator==()`

```
bool GiNaC::const_preorder_iterator::operator== (
    const const_preorder_iterator & other ) const [inline], [noexcept]
```

References `s`.

#### 6.22.2.6 `operator!=()`

```
bool GiNaC::const_preorder_iterator::operator!= (
    const const_preorder_iterator & other ) const [inline], [noexcept]
```

#### 6.22.2.7 `increment()`

```
void GiNaC::const_preorder_iterator::increment ( ) [inline], [private]
```

References `GiNaC::internal::_iter_rep::e`, `GiNaC::internal::_iter_rep::i`, `GiNaC::internal::_iter_rep::i_end`, `GiNaC::ex::nops()`, `GiNaC::ex::op()`, and `s`.

Referenced by `operator++()`.

### 6.22.3 Member Data Documentation

#### 6.22.3.1 `s`

```
std::stack<internal::_iter_rep, std::vector<internal::_iter_rep> > GiNaC::const_preorder_iterator::s [private]
```

Referenced by `const_preorder_iterator()`, `increment()`, `operator*()`, `operator->()`, and `operator==()`.

The documentation for this class was generated from the following file:

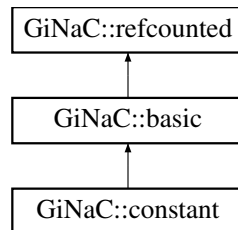
- [ex.h](#)

## 6.23 GiNaC::constant Class Reference

This class holds constants, symbols with specific numerical value.

```
#include <constant.h>
```

Inheritance diagram for GiNaC::constant:



### Public Member Functions

- `constant` (const std::string &initname, `evalffunctype` efun=nullptr, const std::string &texname=std::string(), unsigned `domain=domain::complex`)
- `constant` (const std::string &initname, const `numeric` &initnumber, const std::string &texname=std::string(), unsigned `domain=domain::complex`)
- bool `info` (unsigned inf) const override  
*Information about the object.*
- `ex evalf` () const override  
*Evaluate object numerically.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- void `archive` (`archive_node` &n) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Load (deserialize) the object from an archive node.*

### Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for a constant always returns 0.*
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const

## Private Attributes

- `std::string name`  
*printname of this constant*
- `std::string TeX_name`  
*LaTeX name.*
- `evalffunctype ef`
- `ex number`  
*numerical value this constant `evalf()`s to*
- `unsigned serial`  
*unique serial number for comparison*
- `unsigned domain`  
*numerical value this constant `evalf()`s to*

## Static Private Attributes

- `static unsigned next_serial = 0`

## Additional Inherited Members

### 6.23.1 Detailed Description

This class holds constants, symbols with specific numerical value.

Each object of this class must either provide their own function to evaluate it to class numeric or provide the constant as a numeric (if it's an exact number).

### 6.23.2 Constructor & Destructor Documentation

#### 6.23.2.1 `constant()` [1/2]

```
GiNaC::constant::constant (
    const std::string & initname,
    evalffunctype efun = nullptr,
    const std::string & texname = std::string(),
    unsigned domain = domain::complex )
```

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `name`, `GiNaC::basic::setflag()`, and `TeX_name`.

6.23.2.2 `constant()` [2/2]

```
GiNaC::constant::constant (
    const std::string & initname,
    const numeric & initnumber,
    const std::string & texname = std::string(),
    unsigned domain = domain::complex )
```

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `name`, `GiNaC::basic::setflag()`, and `TeX_name`.

## 6.23.3 Member Function Documentation

6.23.3.1 `info()`

```
bool GiNaC::constant::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References `GiNaC::info_flags::nonnegative`, `GiNaC::info_flags::polynomial`, `GiNaC::domain::positive`, `GiNaC::info_flags::positive`, `GiNaC::domain::real`, and `GiNaC::info_flags::real`.

6.23.3.2 `evalf()`

```
ex GiNaC::constant::evalf ( ) const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References `ef`, `GiNaC::ex::evalf()`, and `number`.

Referenced by `GiNaC::EllipticE_eval()`, and `GiNaC::EllipticK_eval()`.

#### 6.23.3.3 is\_polynomial()

```
bool GiNaC::constant::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

#### 6.23.3.4 conjugate()

```
ex GiNaC::constant::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

#### 6.23.3.5 real\_part()

```
ex GiNaC::constant::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

#### 6.23.3.6 imag\_part()

```
ex GiNaC::constant::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

#### 6.23.3.7 archive()

```
void GiNaC::constant::archive (
    archive\_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [n](#), and [name](#).



#### 6.23.3.8 read\_archive()

```
void GiNaC::constant::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

##### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::Catalan](#), [GiNaC::Euler](#), [n](#), [name](#), and [GiNaC::Pi](#).

#### 6.23.3.9 derivative()

```
ex GiNaC::constant::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a constant always returns 0.

##### See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#).

#### 6.23.3.10 is\_equal\_same\_type()

```
bool GiNaC::constant::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [serial](#).

#### 6.23.3.11 calchash()

```
unsigned GiNaC::constant::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class `basic` computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

#### 6.23.3.12 do\_print()

```
void GiNaC::constant::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [name](#).

#### 6.23.3.13 do\_print\_tree()

```
void GiNaC::constant::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), and [name](#).

#### 6.23.3.14 do\_print\_latex()

```
void GiNaC::constant::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), and [TeX\\_name](#).

#### 6.23.3.15 do\_print\_python\_repr()

```
void GiNaC::constant::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), [name](#), and [TeX\\_name](#).

## 6.23.4 Member Data Documentation

### 6.23.4.1 name

```
std::string GiNaC::constant::name [private]
```

printname of this constant

Referenced by `archive()`, `constant()`, `do_print()`, `do_print_python_repr()`, `do_print_tree()`, and `read_archive()`.

### 6.23.4.2 TeX\_name

```
std::string GiNaC::constant::TeX_name [private]
```

LaTeX name.

Referenced by `constant()`, `do_print_latex()`, and `do_print_python_repr()`.

### 6.23.4.3 ef

```
evalffunctype GiNaC::constant::ef [private]
```

Referenced by `evalf()`.

### 6.23.4.4 number

```
ex GiNaC::constant::number [private]
```

numerical value this constant `evalf()`s to

Referenced by `evalf()`.

### 6.23.4.5 serial

```
unsigned GiNaC::constant::serial [private]
```

unique serial number for comparison

Referenced by `calchash()`, and `is_equal_same_type()`.

#### 6.23.4.6 next\_serial

```
unsigned GiNaC::constant::next_serial = 0 [static], [private]
```

#### 6.23.4.7 domain

```
unsigned GiNaC::constant::domain [private]
```

numerical value this constant [evalf\(\)](#)s to

The documentation for this class was generated from the following files:

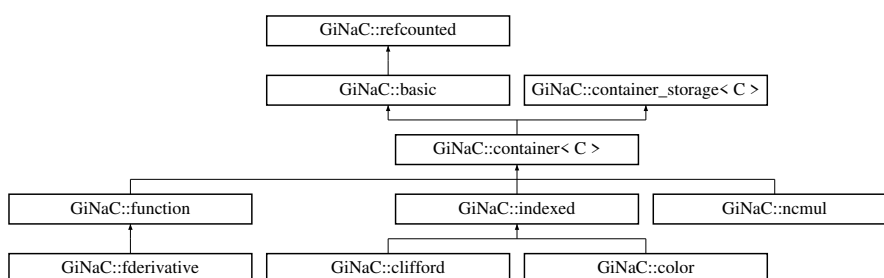
- [constant.h](#)
- [constant.cpp](#)

## 6.24 GiNaC::container< C > Class Template Reference

Wrapper template for making [GiNaC](#) classes out of STL containers.

```
#include <registrar.h>
```

Inheritance diagram for GiNaC::container< C >:



### Public Types

- typedef `STLT::const_iterator` [const\\_iterator](#)
- typedef `STLT::const_reverse_iterator` [const\\_reverse\\_iterator](#)

## Public Member Functions

- [container](#) (STLT const &s)
- [container](#) (STLT &&v)
- [container](#) (exvector::const\_iterator b, exvector::const\_iterator e)
- [container](#) (std::initializer\_list< [ex](#) > il)
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- size\_t [nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex & let\\_op](#) (size\_t i) override  
*Return modifiable operand/member at position i.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &sym\_lst) override  
*Load (deserialize) the object from an archive node.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Archive the object.*
- [container & prepend](#) (const [ex](#) &b)  
*Add element at front.*
- [container & append](#) (const [ex](#) &b)  
*Add element at back.*
- [container & remove\\_first](#) ()  
*Remove first element.*
- [container & remove\\_last](#) ()  
*Remove last element.*
- [container & remove\\_all](#) ()  
*Remove all elements.*
- [container & sort](#) ()  
*Sort elements.*
- [container & unique](#) ()  
*Remove adjacent duplicate elements.*
- [const\\_iterator begin](#) () const
- [const\\_iterator end](#) () const
- [const\\_reverse\\_iterator rbegin](#) () const
- [const\\_reverse\\_iterator rend](#) () const

## Protected Types

- typedef [container\\_storage](#)< C >::STLT STLT

## Protected Member Functions

- `ex conjugate ()` const override
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `bool is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- virtual `ex thiscontainer` (const `STLT` &v) const  
*Similar to `duplicate()`, but with a preset sequence.*
- virtual `ex thiscontainer` (`STLT` &&v) const  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- virtual void `printseq` (const `print_context` &c, char openbracket, char delim, char closebracket, unsigned this←\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `do_print_python` (const `print_python` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const
- `STLT` `subchildren` (const `exmap` &m, unsigned `options`=0) const

## Static Protected Member Functions

- static unsigned `get_default_flags ()`  
*Specialization of `container::get_default_flags()` for `lst`.*
- static char `get_open_delim ()`  
*Specialization of `container::get_open_delim()` for `lst`.*
- static char `get_close_delim ()`  
*Specialization of `container::get_close_delim()` for `lst`.*

## Private Member Functions

- void `sort_` (std::random\_access\_iterator\_tag)
- void `sort_` (std::input\_iterator\_tag)
- void `unique_` ()
- template<>  
void `unique_` ()  
*Specialization of `container::unique_()` for `std::list`.*

## Additional Inherited Members

### 6.24.1 Detailed Description

```
template<template< class T, class=std::allocator< T >> class C>
class GiNaC::container< C >
```

Wrapper template for making `GiNaC` classes out of STL containers.

## 6.24.2 Member Typedef Documentation

### 6.24.2.1 STL

```
template<template< class T, class=std::allocator< T >> class C>
typedef container\_storage<C>::STLT GiNaC::container< C >::STLT [protected]
```

### 6.24.2.2 const\_iterator

```
template<template< class T, class=std::allocator< T >> class C>
typedef STL::const_iterator GiNaC::container< C >::const_iterator
```

### 6.24.2.3 const\_reverse\_iterator

```
template<template< class T, class=std::allocator< T >> class C>
typedef STL::const_reverse_iterator GiNaC::container< C >::const_reverse_iterator
```

## 6.24.3 Constructor & Destructor Documentation

### 6.24.3.1 container() [1/4]

```
template<template< class T, class=std::allocator< T >> class C>
GiNaC::container< C >::container (
    STL const & s ) [inline]
```

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [GiNaC::container< C >::thiscontainer\(\)](#).

### 6.24.3.2 container() [2/4]

```
template<template< class T, class=std::allocator< T >> class C>
GiNaC::container< C >::container (
    STL && v ) [inline], [explicit]
```

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

**6.24.3.3 container()** [3/4]

```
template<template< class T, class=std::allocator< T >> class C>
GiNaC::container< C >::container (
    exvector::const_iterator b,
    exvector::const_iterator e ) [inline]
```

References `GiNaC::container< C >::get_default_flags()`, and `GiNaC::basic::setflag()`.

**6.24.3.4 container()** [4/4]

```
template<template< class T, class=std::allocator< T >> class C>
GiNaC::container< C >::container (
    std::initializer_list< ex > il ) [inline]
```

References `GiNaC::container< C >::get_default_flags()`, and `GiNaC::basic::setflag()`.

**6.24.4 Member Function Documentation****6.24.4.1 get\_default\_flags()**

```
unsigned GiNaC::lst::get_default_flags ( ) [inline], [static], [protected]
```

Specialization of `container::get_default_flags()` for `lst`.

Referenced by `GiNaC::container< C >::container()`, and `GiNaC::container< C >::read_archive()`.

**6.24.4.2 get\_open\_delim()**

```
char GiNaC::lst::get_open_delim ( ) [inline], [static], [protected]
```

Specialization of `container::get_open_delim()` for `lst`.

**6.24.4.3 get\_close\_delim()**

```
char GiNaC::lst::get_close_delim ( ) [inline], [static], [protected]
```

Specialization of `container::get_close_delim()` for `lst`.



## 6.24.4.4 info()

```
bool GiNaC::lst::info (
    unsigned inf ) const [inline], [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

## 6.24.4.5 precedence()

```
template<template< class T, class=std::allocator< T >> class C>
unsigned GiNaC::container< C >::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

Referenced by [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), and [GiNaC::function::print\(\)](#).

## 6.24.4.6 nops()

```
template<template< class T, class=std::allocator< T >> class C>
size_t GiNaC::container< C >::nops ( ) const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::function::calchash\(\)](#), [GiNaC::ncmul::conjugate\(\)](#), [GiNaC::ex::denom\(\)](#), [GiNaC::diag\\_matrix\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::ncmul::get\\_free\\_indices\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::lst\\_to\\_matrix\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::ex::numer\(\)](#), [GiNaC::ex::numer\\_denom\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::container< C >::real\\_part\(\)](#), [GiNaC::sqrfree\(\)](#), and [GiNaC::ex::subs\(\)](#).

#### 6.24.4.7 op()

```
template<template< class T, class=std::allocator< T >> class C>
ex GiNaC::container< C >::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References GINAC\_ASSERT, and GiNaC::nops().

Referenced by GiNaC::function::calchash(), GiNaC::ex::denom(), GiNaC::ncmul::expand(), GiNaC::mul::expand(), GiNaC::ncmul::get\_free\_indices(), GiNaC::H\_evalf(), GiNaC::indexed::imag\_part(), GiNaC::is\_discriminant\_of←\_quadratic\_number\_field(), GiNaC::kronecker\_symbol(), GiNaC::add::normal(), GiNaC::mul::normal(), GiNaC←::power::normal(), GiNaC::ex::normal(), GiNaC::basic::normal(), GiNaC::ex::numer(), GiNaC::ex::numer\_denom(), GiNaC::indexed::real\_part(), GiNaC::rename\_dummy\_indices\_uniquely(), GiNaC::indexed::return\_type(), GiNaC←::indexed::return\_type\_tinfo(), GiNaC::sqrfree(), GiNaC::symm(), and GiNaC::symmetrize\_cyclic().

#### 6.24.4.8 let\_op()

```
template<template< class T, class=std::allocator< T >> class C>
ex & GiNaC::container< C >::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References GINAC\_ASSERT, and GiNaC::nops().

#### 6.24.4.9 subs()

```
template<template< class T, class=std::allocator< T >> class C>
ex GiNaC::container< C >::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References GiNaC::is\_a(), m, and options.

Referenced by GiNaC::basic::normal().

## 6.24.4.10 read\_archive()

```
template<template< class T, class=std::allocator< T >> class C>
void GiNaC::container< C >::read_archive (
    const archive_node & n,
    lst & syms ) [inline], [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

## Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::fderivative](#).

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), [n](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

## 6.24.4.11 archive()

```
template<template< class T, class=std::allocator< T >> class C>
void GiNaC::container< C >::archive (
    archive_node & n ) const [inline], [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::fderivative](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [n](#), and [GiNaC::container\\_storage< C >::seq](#).

## 6.24.4.12 conjugate()

```
template<template< class T, class=std::allocator< T >> class C>
ex GiNaC::container< C >::conjugate ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), and [GiNaC::ncmul](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [GiNaC::container< C >::thiscontainer\(\)](#), and [x](#).

Referenced by [GiNaC::ncmul::conjugate\(\)](#).

#### 6.24.4.13 `real_part()`

```
template<template< class T, class=std::allocator< T >> class C>
ex GiNaC::container< C >::real_part ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), and [GiNaC::container< C >::thiscontainer\(\)](#).

#### 6.24.4.14 `imag_part()`

```
template<template< class T, class=std::allocator< T >> class C>
ex GiNaC::container< C >::imag_part ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), and [GiNaC::container< C >::thiscontainer\(\)](#).

#### 6.24.4.15 `is_equal_same_type()`

```
template<template< class T, class=std::allocator< T >> class C>
bool GiNaC::container< C >::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), and [GiNaC::fderivative](#).

References [GINAC\\_ASSERT](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::function::is\\_equal\\_same\\_type\(\)](#).

## 6.24.4.16 thiscontainer() [1/2]

```
template<template< class T, class=std::allocator< T >> class C>
virtual ex GiNaC::container< C >::thiscontainer (
    const STLT & v ) const [inline], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence.

Must be overridden by derived classes.

References GiNaC::container< C >::container().

Referenced by GiNaC::container< C >::conjugate(), GiNaC::container< C >::imag\_part(), and GiNaC::container< C >::real\_part().

## 6.24.4.17 thiscontainer() [2/2]

```
template<template< class T, class=std::allocator< T >> class C>
virtual ex GiNaC::container< C >::thiscontainer (
    STLT && v ) const [inline], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).

Must be overridden by derived classes.

References GiNaC::container< C >::container().

## 6.24.4.18 printseq()

```
template<template< class T, class=std::allocator< T >> class C>
void GiNaC::container< C >::printseq (
    const print_context & c,
    char openbracket,
    char delim,
    char closebracket,
    unsigned this_precedence,
    unsigned upper_precedence = 0 ) const [protected], [virtual]
```

Print sequence of contained elements.

References c.

Referenced by GiNaC::fderivative::do\_print(), GiNaC::ncmul::do\_print(), GiNaC::fderivative::do\_print\_csrc(), GiNaC::ncmul::do\_print\_csrc(), GiNaC::fderivative::do\_print\_latex(), and GiNaC::function::print().

**6.24.4.19** `sort_()` [1/2]

```
template<template< class T, class=std::allocator< T >> class C>
void GiNaC::container< C >::sort_ (
    std::random_access_iterator_tag ) [inline], [private]
```

References `GiNaC::container_storage< C >::seq`.

**6.24.4.20** `sort_()` [2/2]

```
template<template< class T, class=std::allocator< T >> class C>
void GiNaC::container< C >::sort_ (
    std::input_iterator_tag ) [inline], [private]
```

References `GiNaC::container_storage< C >::seq`.

**6.24.4.21** `unique_()` [1/2]

```
template<template< class T, class=std::allocator< T >> class C>
void GiNaC::container< C >::unique_ ( ) [inline], [private]
```

References `GiNaC::container_storage< C >::seq`.

**6.24.4.22** `prepend()`

```
template<template< class T, class=std::allocator< T >> class C>
container< C > & GiNaC::container< C >::prepend (
    const ex & b )
```

Add element at front.

**6.24.4.23** `append()`

```
template<template< class T, class=std::allocator< T >> class C>
container< C > & GiNaC::container< C >::append (
    const ex & b )
```

Add element at back.

Referenced by `GiNaC::ifactor()`, `GiNaC::Li_eval()`, `GiNaC::lsolve()`, `GiNaC::replace_with_symbol()`, `GiNaC::←  
::sqrfree()`, `GiNaC::symm()`, and `GiNaC::symmetrize_cyclic()`.

#### 6.24.4.24 remove\_first()

```
template<template< class T, class=std::allocator< T >> class C>
container< C > & GiNaC::container< C >::remove_first ( )
```

Remove first element.

Referenced by GiNaC::sqrfree(), and GiNaC::symmetrize\_cyclic().

#### 6.24.4.25 remove\_last()

```
template<template< class T, class=std::allocator< T >> class C>
container< C > & GiNaC::container< C >::remove_last ( )
```

Remove last element.

#### 6.24.4.26 remove\_all()

```
template<template< class T, class=std::allocator< T >> class C>
container< C > & GiNaC::container< C >::remove_all ( )
```

Remove all elements.

#### 6.24.4.27 sort()

```
template<template< class T, class=std::allocator< T >> class C>
container< C > & GiNaC::container< C >::sort ( )
```

Sort elements.

#### 6.24.4.28 unique()

```
template<template< class T, class=std::allocator< T >> class C>
container< C > & GiNaC::container< C >::unique ( )
```

Remove adjacent duplicate elements.

**6.24.4.29 begin()**

```
template<template< class T, class=std::allocator< T >> class C>
const_iterator GiNaC::container< C >::begin ( ) const [inline]
```

References GiNaC::container\_storage< C >::seq.

Referenced by GiNaC::ex::antisymmetrize(), GiNaC::ncmul::conjugate(), GiNaC::G3\_eval(), GiNaC::G3\_evalf(), GiNaC::H\_evalf(), GiNaC::container< C >::imag\_part(), GiNaC::container< C >::real\_part(), GiNaC::ex::subs(), GiNaC::ex::symmetrize(), GiNaC::ex::symmetrize\_cyclic(), GiNaC::zeta1\_evalf(), GiNaC::zeta2\_evalf(), and GiNaC::zeta2\_print\_latex().

**6.24.4.30 end()**

```
template<template< class T, class=std::allocator< T >> class C>
const_iterator GiNaC::container< C >::end ( ) const [inline]
```

References GiNaC::container\_storage< C >::seq.

Referenced by GiNaC::ex::antisymmetrize(), GiNaC::fderivative::archive(), GiNaC::ncmul::conjugate(), GiNaC::fderivative::do\_print(), GiNaC::fderivative::do\_print\_csrc(), GiNaC::fderivative::do\_print\_latex(), GiNaC::fderivative::do\_print\_tree(), GiNaC::ncmul::expandchildren(), GiNaC::H\_evalf(), GiNaC::container< C >::imag\_part(), GiNaC::container< C >::real\_part(), GiNaC::ncmul::return\_type(), GiNaC::ex::subs(), GiNaC::ex::symmetrize(), and GiNaC::ex::symmetrize\_cyclic().

**6.24.4.31 rbegin()**

```
template<template< class T, class=std::allocator< T >> class C>
const_reverse_iterator GiNaC::container< C >::rbegin ( ) const [inline]
```

References GiNaC::container\_storage< C >::seq.

**6.24.4.32 rend()**

```
template<template< class T, class=std::allocator< T >> class C>
const_reverse_iterator GiNaC::container< C >::rend ( ) const [inline]
```

References GiNaC::container\_storage< C >::seq.



**6.24.4.33 do\_print()**

```
template<template< class T, class=std::allocator< T >> class C>
void GiNaC::container< C >::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#).

**6.24.4.34 do\_print\_tree()**

```
template<template< class T, class=std::allocator< T >> class C>
void GiNaC::container< C >::do_print_tree (
    const print_tree & c,
    unsigned level ) const [protected]
```

References [c](#), and [GiNaC::nops\(\)](#).

**6.24.4.35 do\_print\_python()**

```
template<template< class T, class=std::allocator< T >> class C>
void GiNaC::container< C >::do_print_python (
    const print_python & c,
    unsigned level ) const [protected]
```

References [c](#).

**6.24.4.36 do\_print\_python\_repr()**

```
template<template< class T, class=std::allocator< T >> class C>
void GiNaC::container< C >::do_print_python_repr (
    const print_python_repr & c,
    unsigned level ) const [protected]
```

References [c](#).

**6.24.4.37 subschildren()**

```
template<template< class T, class=std::allocator< T >> class C>
container< C >::STLT GiNaC::container< C >::subschildren (
    const exmap & m,
    unsigned options = 0 ) const [protected]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [m](#), [options](#), and [GiNaC::ex::subs\(\)](#).

#### 6.24.4.38 `unique_()` [2/2]

```
template<>
void GiNaC::container< std::list >::unique_ ( ) [inline], [private]
```

Specialization of `container::unique_()` for `std::list`.

The documentation for this class was generated from the following files:

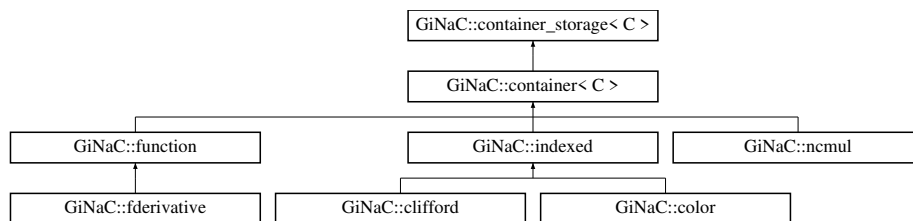
- [container.h](#)
- [exprseq.h](#)
- [lst.h](#)

## 6.25 `GiNaC::container_storage< C >` Class Template Reference

Helper template for encapsulating the `reserve()` mechanics of STL containers.

```
#include <container.h>
```

Inheritance diagram for `GiNaC::container_storage< C >`:



### Protected Types

- typedef `C< ex >` `STLT`

### Protected Member Functions

- `container_storage ()`
- `container_storage (size_t n, const ex &e)`
- `container_storage (std::initializer_list< ex > il)`
- `template<class In >`  
`container_storage (In b, In e)`
- `void reserve (size_t)`
- `~container_storage ()`
- `template<>`  
`void reserve (size_t n)`
- `template<>`  
`void reserve (std::vector< ex > &v, size_t n)`

### Static Protected Member Functions

- static void `reserve (STLT &, size_t)`

## Protected Attributes

- [STLT seq](#)

### 6.25.1 Detailed Description

```
template<template< class T, class=std::allocator< T >> class C>
class GiNaC::container_storage< C >
```

Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.

### 6.25.2 Member Typedef Documentation

#### 6.25.2.1 STLT

```
template<template< class T, class=std::allocator< T >> class C>
typedef C<ex> GiNaC::container_storage< C >::STLT [protected]
```

### 6.25.3 Constructor & Destructor Documentation

#### 6.25.3.1 container\_storage() [1/4]

```
template<template< class T, class=std::allocator< T >> class C>
GiNaC::container_storage< C >::container_storage ( ) [inline], [protected]
```

#### 6.25.3.2 container\_storage() [2/4]

```
template<template< class T, class=std::allocator< T >> class C>
GiNaC::container_storage< C >::container_storage (
    size_t n,
    const ex & e ) [inline], [protected]
```

#### 6.25.3.3 container\_storage() [3/4]

```
template<template< class T, class=std::allocator< T >> class C>
GiNaC::container_storage< C >::container_storage (
    std::initializer_list< ex > il ) [inline], [protected]
```

**6.25.3.4 container\_storage()** [4/4]

```
template<template< class T, class=std::allocator< T >> class C>
template<class In >
GiNaC::container_storage< C >::container_storage (
    In b,
    In e ) [inline], [protected]
```

**6.25.3.5 ~container\_storage()**

```
template<template< class T, class=std::allocator< T >> class C>
GiNaC::container_storage< C >::~~container_storage ( ) [inline], [protected]
```

**6.25.4 Member Function Documentation****6.25.4.1 reserve()** [1/4]

```
template<template< class T, class=std::allocator< T >> class C>
void GiNaC::container_storage< C >::reserve (
    size_t ) [inline], [protected]
```

Referenced by `GiNaC::container< C >::conjugate()`, `GiNaC::ncmul::eval()`, `GiNaC::color::eval_ncmul()`, `GiNaC::container< C >::imag_part()`, `GiNaC::container< C >::read_archive()`, `GiNaC::container< C >::real_part()`, and `GiNaC::rename_dummy_indices_uniquely()`.

**6.25.4.2 reserve()** [2/4]

```
template<template< class T, class=std::allocator< T >> class C>
static void GiNaC::container_storage< C >::reserve (
    STLT & ,
    size_t ) [inline], [static], [protected]
```

**6.25.4.3 reserve()** [3/4]

```
template<>
void GiNaC::container_storage< std::vector >::reserve (
    size_t n ) [inline], [protected]
```

References n.

6.25.4.4 `reserve()` [4/4]

```
template<>
void GiNaC::container_storage< std::vector >::reserve (
    std::vector< ex > & v,
    size_t n ) [inline], [protected]
```

References n.

## 6.25.5 Member Data Documentation

6.25.5.1 `seq`

```
template<template< class T, class=std::allocator< T >> class C>
STLT GiNaC::container_storage< C >::seq [protected]
```

Referenced by `GiNaC::indexed::all_index_values_are()`, `GiNaC::container< C >::archive()`, `GiNaC::container< C >::begin()`, `GiNaC::ncmul::coeff()`, `GiNaC::container< C >::conjugate()`, `GiNaC::function::conjugate()`, `GiNaC::container< C >::container()`, `GiNaC::ncmul::degree()`, `GiNaC::fderivative::derivative()`, `GiNaC::ncmul::derivative()`, `GiNaC::function::derivative()`, `GiNaC::clifford::do_print_dflt()`, `GiNaC::clifford::do_print_latex()`, `GiNaC::clifford::do_print_tree()`, `GiNaC::fderivative::do_print_tree()`, `GiNaC::indexed::do_print_tree()`, `GiNaC::container< C >::end()`, `GiNaC::fderivative::eval()`, `GiNaC::ncmul::eval()`, `GiNaC::indexed::eval()`, `GiNaC::function::eval()`, `GiNaC::function::eval_ncmul()`, `GiNaC::function::evalf()`, `GiNaC::ncmul::evalm()`, `GiNaC::ncmul::expand()`, `GiNaC::indexed::expand()`, `GiNaC::function::expand()`, `GiNaC::ncmul::expandchildren()`, `GiNaC::indexed::get_dummy_indices()`, `GiNaC::ncmul::get_factors()`, `GiNaC::indexed::get_free_indices()`, `GiNaC::indexed::get_indices()`, `GiNaC::indexed::has_dummy_index_for()`, `GiNaC::function::imag_part()`, `GiNaC::indexed::info()`, `GiNaC::function::info()`, `GiNaC::container< C >::is_equal_same_type()`, `GiNaC::ncmul::ldegree()`, `GiNaC::container< C >::nops()`, `GiNaC::function::print()`, `GiNaC::indexed::print_indexed()`, `GiNaC::indexed::printindices()`, `GiNaC::container< C >::rbegin()`, `GiNaC::container< C >::read_archive()`, `GiNaC::indexed::read_archive()`, `GiNaC::function::read_archive()`, `GiNaC::function::real_part()`, `GiNaC::container< C >::rend()`, `GiNaC::ncmul::return_type()`, `GiNaC::function::return_type()`, `GiNaC::ncmul::return_type_tinfo()`, `GiNaC::function::return_type_tinfo()`, `GiNaC::function::series()`, `GiNaC::simplify_indexed()`, `GiNaC::container< C >::sort_()`, `GiNaC::container< C >::unique_()`, and `GiNaC::indexed::validate()`.

The documentation for this class was generated from the following file:

- [container.h](#)

## 6.26 GiNaC::composition\_generator::coolmulti Struct Reference

## Classes

- struct [element](#)

## Public Member Functions

- [coolmulti](#) (const std::vector< unsigned > &partition)
- [~coolmulti](#) ()
- void [next\\_permutation](#) ()
- bool [finished](#) () const

## Public Attributes

- [element](#) \* [head](#)
- [element](#) \* [i](#)
- [element](#) \* [after\\_i](#)

## 6.26.1 Constructor & Destructor Documentation

### 6.26.1.1 coolmulti()

```
GiNaC::composition_generator::coolmulti::coolmulti (
    const std::vector< unsigned > & partition ) [inline], [explicit]
```

References [after\\_i](#), [head](#), [i](#), [n](#), and [GiNaC::composition\\_generator::coolmulti::element::next](#).

### 6.26.1.2 ~coolmulti()

```
GiNaC::composition_generator::coolmulti::~~coolmulti ( ) [inline]
```

References [head](#).

## 6.26.2 Member Function Documentation

### 6.26.2.1 next\_permutation()

```
void GiNaC::composition_generator::coolmulti::next_permutation ( ) [inline]
```

References [after\\_i](#), [head](#), [i](#), [k](#), [GiNaC::composition\\_generator::coolmulti::element::next](#), and [GiNaC::composition\\_generator::coolmulti::element::value](#).

Referenced by [GiNaC::composition\\_generator::next\(\)](#).

### 6.26.2.2 finished()

```
bool GiNaC::composition_generator::coolmulti::finished ( ) const [inline]
```

References [after\\_i](#), [head](#), [GiNaC::composition\\_generator::coolmulti::element::next](#), and [GiNaC::composition\\_generator::coolmulti::element::value](#).

Referenced by [GiNaC::composition\\_generator::next\(\)](#).

### 6.26.3 Member Data Documentation

#### 6.26.3.1 head

`element*` GiNaC::composition\_generator::coolmulti::head

Referenced by coolmulti(), finished(), GiNaC::composition\_generator::get(), next\_permutation(), and ~coolmulti().

#### 6.26.3.2 i

`element *` GiNaC::composition\_generator::coolmulti::i

Referenced by coolmulti(), and next\_permutation().

#### 6.26.3.3 after\_i

`element *` GiNaC::composition\_generator::coolmulti::after\_i

Referenced by coolmulti(), finished(), and next\_permutation().

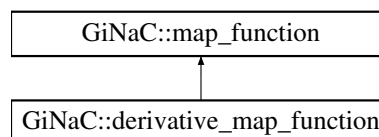
The documentation for this struct was generated from the following file:

- [utils.h](#)

## 6.27 GiNaC::derivative\_map\_function Struct Reference

Function object to be applied by [basic::derivative\(\)](#).

Inheritance diagram for GiNaC::derivative\_map\_function:



### Public Member Functions

- [derivative\\_map\\_function](#) (const [symbol](#) &sym)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Attributes

- const [symbol](#) & [s](#)

## Additional Inherited Members

### 6.27.1 Detailed Description

Function object to be applied by [basic::derivative\(\)](#).

### 6.27.2 Constructor & Destructor Documentation

#### 6.27.2.1 [derivative\\_map\\_function\(\)](#)

```
GiNaC::derivative_map_function::derivative_map_function (  
    const symbol & sym ) [inline]
```

### 6.27.3 Member Function Documentation

#### 6.27.3.1 [operator\(\)\(\)](#)

```
ex GiNaC::derivative_map_function::operator() (  
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::diff\(\)](#).

### 6.27.4 Member Data Documentation

#### 6.27.4.1 [s](#)

```
const symbol& GiNaC::derivative_map_function::s
```

The documentation for this struct was generated from the following file:

- [basic.cpp](#)



## 6.28 GiNaC::determinant\_algo Class Reference

Switch to control algorithm for determinant computation.

```
#include <flags.h>
```

### Public Types

- enum {  
[automatic](#), [gauss](#), [divfree](#), [laplace](#),  
[bareiss](#) }

### 6.28.1 Detailed Description

Switch to control algorithm for determinant computation.

### 6.28.2 Member Enumeration Documentation

#### 6.28.2.1 anonymous enum

anonymous enum

#### Enumerator

automatic	Let the system choose. A heuristics is applied for automatic determination of a suitable algorithm.
gauss	<p>Gauss elimination. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)} / m_{k,k}^{(k)}$ <p>The determinant is then just the product of diagonal elements. Choose this algorithm only for purely numerical matrices.</p>
divfree	<p>Division-free elimination. This is a modification of Gauss elimination where the division by the pivot element is not carried out. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}$ <p>The determinant can later be computed by inspecting the diagonal elements only. This algorithm is only there for the purpose of cross-checks. It is never fast.</p>
laplace	Laplace elimination. This is plain recursive elimination along minors although multiple minors are avoided by the algorithm. Although the algorithm is exponential in complexity it is frequently the fastest one when the matrix is populated by complicated symbolic expressions.
bareiss	<p>Bareiss fraction-free elimination. This is a modification of Gauss elimination where the division by the pivot element is <i>delayed</i> until it can be carried out without computing GCDs. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = (m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}) / m_{k-1,k-1}^{(k-1)}$
Generated by Doxygen	<p>(We have set <math>m_{-1,-1}^{(-1)} = 1</math> in order to avoid a case distinction in above formula.) It can be shown that nothing more than polynomial long division is needed for carrying out the division. The determinant can then be read of from the lower right entry. This algorithm is rarely fast for computing determinants.</p>

The documentation for this class was generated from the following file:

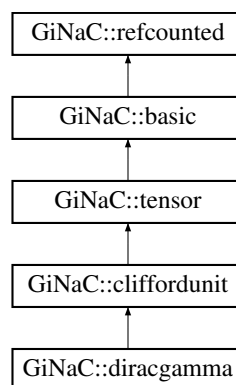
- [flags.h](#)

## 6.29 GiNaC::diracgamma Class Reference

This class represents the Dirac gamma Lorentz vector.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgamma:



### Public Member Functions

- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of a gamma matrix with something else.*

### Protected Member Functions

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

### Additional Inherited Members

#### 6.29.1 Detailed Description

This class represents the Dirac gamma Lorentz vector.

#### 6.29.2 Member Function Documentation

## 6.29.2.1 contract\_with()

```
bool GiNaC::diracgamma::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of a gamma matrix with something else.

Reimplemented from [GiNaC::cliffordunit](#).

## 6.29.2.2 do\_print()

```
void GiNaC::diracgamma::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

## 6.29.2.3 do\_print\_latex()

```
void GiNaC::diracgamma::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

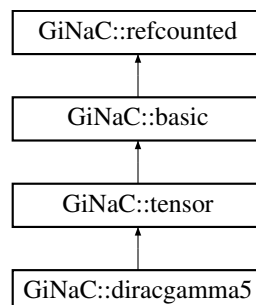
- [clifford.h](#)
- [clifford.cpp](#)

## 6.30 GiNaC::diracgamma5 Class Reference

This class represents the Dirac gamma5 object which anticommutes with all other gammas.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgamma5:



## Protected Member Functions

- void `do_print` (const `print_context` &`c`, unsigned level) const
- void `do_print_latex` (const `print_latex` &`c`, unsigned level) const

## Private Member Functions

- `ex conjugate` () const override

## Additional Inherited Members

### 6.30.1 Detailed Description

This class represents the Dirac gamma5 object which anticommutes with all other gammas.

### 6.30.2 Member Function Documentation

#### 6.30.2.1 `conjugate()`

```
ex GiNaC::diracgamma5::conjugate ( ) const [override], [private], [virtual]
```

Reimplemented from `GiNaC::basic`.

#### 6.30.2.2 `do_print()`

```
void GiNaC::diracgamma5::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

#### 6.30.2.3 `do_print_latex()`

```
void GiNaC::diracgamma5::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

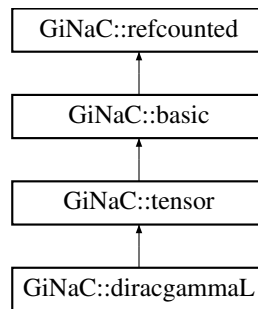
- `clifford.h`
- `clifford.cpp`

## 6.31 GiNaC::diracgammaL Class Reference

This class represents the Dirac gammaL object which behaves like  $1/2 (1-\gamma_5)$ .

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgammaL:



### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Private Member Functions

- `ex conjugate` () const override

### Additional Inherited Members

#### 6.31.1 Detailed Description

This class represents the Dirac gammaL object which behaves like  $1/2 (1-\gamma_5)$ .

#### 6.31.2 Member Function Documentation

##### 6.31.2.1 `conjugate()`

```
ex GiNaC::diracgammaL::conjugate ( ) const [override], [private], [virtual]
```

Reimplemented from `GiNaC::basic`.

### 6.31.2.2 do\_print()

```
void GiNaC::diracgammaL::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

### 6.31.2.3 do\_print\_latex()

```
void GiNaC::diracgammaL::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

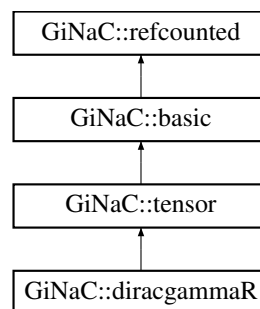
- [clifford.h](#)
- [clifford.cpp](#)

## 6.32 GiNaC::diracgammaR Class Reference

This class represents the Dirac gammaL object which behaves like  $1/2 (1+\gamma_5)$ .

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgammaR:



### Protected Member Functions

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

### Private Member Functions

- [ex conjugate](#) () const override

## Additional Inherited Members

### 6.32.1 Detailed Description

This class represents the Dirac gammaL object which behaves like  $\frac{1}{2} (1 + \gamma_5)$ .

### 6.32.2 Member Function Documentation

#### 6.32.2.1 conjugate()

**ex** `GiNaC::diracgammaR::conjugate ( ) const [override], [private], [virtual]`

Reimplemented from [GiNaC::basic](#).

#### 6.32.2.2 do\_print()

```
void GiNaC::diracgammaR::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

#### 6.32.2.3 do\_print\_latex()

```
void GiNaC::diracgammaR::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

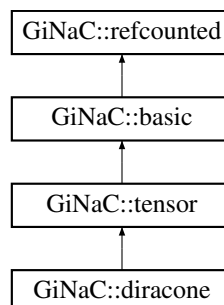
- [clifford.h](#)
- [clifford.cpp](#)

## 6.33 GiNaC::diracone Class Reference

This class represents the Clifford algebra unity element.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracone:



## Protected Member Functions

- void `do_print` (const `print_context` &`c`, unsigned level) const
- void `do_print_latex` (const `print_latex` &`c`, unsigned level) const

## Additional Inherited Members

### 6.33.1 Detailed Description

This class represents the Clifford algebra unity element.

### 6.33.2 Member Function Documentation

#### 6.33.2.1 `do_print()`

```
void GiNaC::diracone::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

#### 6.33.2.2 `do_print_latex()`

```
void GiNaC::diracone::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following file:

- [clifford.h](#)

## 6.34 GiNaC::do\_taylor Class Reference

Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.

```
#include <function.h>
```

### 6.34.1 Detailed Description

Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.

The documentation for this class was generated from the following file:

- [function.h](#)



## 6.35 GiNaC::domain Class Reference

Domain of an object.

```
#include <flags.h>
```

### Public Types

- enum { [complex](#), [real](#), [positive](#) }

#### 6.35.1 Detailed Description

Domain of an object.

#### 6.35.2 Member Enumeration Documentation

##### 6.35.2.1 anonymous enum

anonymous enum

##### Enumerator

complex	
real	
positive	

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.36 GiNaC::dunno Class Reference

Exception class thrown by functions to signal unimplemented functionality so the expression may just be .hold()

```
#include <utils.h>
```

#### 6.36.1 Detailed Description

Exception class thrown by functions to signal unimplemented functionality so the expression may just be .hold()

The documentation for this class was generated from the following file:

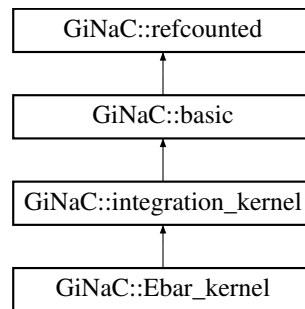
- [utils.h](#)

## 6.37 GiNaC::Ebar\_kernel Class Reference

The Ebar-kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Ebar\_kernel:



### Public Member Functions

- [Ebar\\_kernel](#) (const [ex](#) &n, const [ex](#) &m, const [ex](#) &x, const [ex](#) &y)
- [size\\_t nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex & let\\_op](#) (size\_t i) override  
*Return modifiable operand/member at position i.*
- [bool is\\_numeric](#) (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- [ex get\\_numerical\\_value](#) (const [ex](#) &qbar, int N\_trunc=0) const override  
*Returns the value of  $Ebar_{\{n,m\}}(x,y,qbar)$*

### Protected Member Functions

- [cln::cl\\_N series\\_coeff\\_impl](#) (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- [void do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

### Protected Attributes

- [ex n](#)
- [ex m](#)
- [ex x](#)
- [ex y](#)

### 6.37.1 Detailed Description

The Ebar-kernel.

This class represents the differential one-form

$$\omega_{n;m}^{\overline{E}}(x;y) = \overline{E}_{n;m}(x;y;\overline{q}) \frac{d\overline{q}}{\overline{q}}$$

### 6.37.2 Constructor & Destructor Documentation

#### 6.37.2.1 Ebar\_kernel()

```
GiNaC::Ebar_kernel::Ebar_kernel (
    const ex & n,
    const ex & m,
    const ex & x,
    const ex & y )
```

### 6.37.3 Member Function Documentation

#### 6.37.3.1 nops()

```
size_t GiNaC::Ebar_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 6.37.3.2 op()

```
ex GiNaC::Ebar_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References m, n, x, and y.

### 6.37.3.3 let\_op()

```
ex & GiNaC::Ebar_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), m, n, x, and y.

### 6.37.3.4 is\_numeric()

```
bool GiNaC::Ebar_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), m, n, [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), x, and y.

### 6.37.3.5 get\_numerical\_value()

```
ex GiNaC::Ebar_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of  $Ebar_{\{n,m\}}(x,y,qbar)$

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and qbar.

Referenced by [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

### 6.37.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::Ebar_kernel::series_coeff_impl (
    int i ) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The i-th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), k, m, n, [GiNaC::to\\_int\(\)](#), x, and y.

### 6.37.3.7 do\_print()

```
void GiNaC::Ebar_kernel::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References `c`, `m`, `n`, `GiNaC::ex::print()`, `x`, and `y`.

## 6.37.4 Member Data Documentation

### 6.37.4.1 n

```
ex GiNaC::Ebar_kernel::n [protected]
```

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

### 6.37.4.2 m

```
ex GiNaC::Ebar_kernel::m [protected]
```

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

### 6.37.4.3 x

```
ex GiNaC::Ebar_kernel::x [protected]
```

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

### 6.37.4.4 y

```
ex GiNaC::Ebar_kernel::y [protected]
```

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

The documentation for this class was generated from the following files:

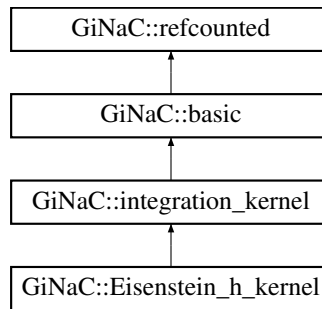
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.38 GiNaC::Eisenstein\_h\_kernel Class Reference

The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Eisenstein\_h\_kernel:



### Public Member Functions

- [Eisenstein\\_h\\_kernel](#) (const [ex](#) &k, const [ex](#) &N, const [ex](#) &r, const [ex](#) &s, const [ex](#) &C\_norm=numeric(1))
- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.*
- [size\\_t nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex &let\\_op](#) (size\_t i) override  
*Return modifiable operand/member at position i.*
- [bool is\\_numeric](#) (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- [ex Laurent\\_series](#) (const [ex](#) &x, int [order](#)) const override  
*Returns the Laurent series, starting possibly with the pole term.*
- [ex get\\_numerical\\_value](#) (const [ex](#) &qbar, int N\_trunc=0) const override  
*Returns the value of the modular form.*
- [ex coefficient\\_a0](#) (const [numeric](#) &k, const [numeric](#) &r, const [numeric](#) &s, const [numeric](#) &N) const  
*The constant coefficient in the Fourier expansion.*
- [ex coefficient\\_an](#) (const [numeric](#) &n, const [numeric](#) &k, const [numeric](#) &r, const [numeric](#) &s, const [numeric](#) &N) const  
*The higher coefficients in the Fourier expansion.*
- [ex q\\_expansion\\_modular\\_form](#) (const [ex](#) &q, int [order](#)) const

### Protected Member Functions

- [bool uses\\_Laurent\\_series](#) () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method Laurent\_series needs to be implemented).*
- [void do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Attributes

- [ex k](#)
- [ex N](#)
- [ex r](#)
- [ex s](#)
- [ex C\\_norm](#)

### 6.38.1 Detailed Description

The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .

This class represents the differential one-form

$$\omega_{k,N,r,s}^{\text{Eisenstein,h}} = C_k h_{k,N,r,s}(\tau) \frac{d\bar{q}_N}{\bar{q}_N}$$

### 6.38.2 Constructor & Destructor Documentation

#### 6.38.2.1 Eisenstein\_h\_kernel()

```
GiNaC::Eisenstein_h_kernel::Eisenstein_h_kernel (
    const ex & k,
    const ex & N,
    const ex & r,
    const ex & s,
    const ex & C_norm = numeric(1) )
```

### 6.38.3 Member Function Documentation

#### 6.38.3.1 series()

```
ex GiNaC::Eisenstein_h_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.

This allows for easy use in the class [modular\\_form\\_kernel](#).

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::lhs\(\)](#), [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [qbar](#), [r](#), [GiNaC::ex::rhs\(\)](#), and [GiNaC::ex::series\(\)](#).

### 6.38.3.2 nops()

```
size_t GiNaC::Eisenstein_h_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.38.3.3 op()

```
ex GiNaC::Eisenstein_h_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References C\_norm, k, N, r, and s.

### 6.38.3.4 let\_op()

```
ex & GiNaC::Eisenstein_h_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References C\_norm, [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), k, N, r, and s.

### 6.38.3.5 is\_numeric()

```
bool GiNaC::Eisenstein_h_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References C\_norm, [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), k, N, [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), r, and s.



## 6.38.3.6 Laurent\_series()

```
ex GiNaC::Eisenstein_h_kernel::Laurent_series (
    const ex & x,
    int order ) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References C\_norm, order, q\_expansion\_modular\_form(), GiNaC::ex::series(), and x.

## 6.38.3.7 get\_numerical\_value()

```
ex GiNaC::Eisenstein_h_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References C\_norm, GiNaC::integration\_kernel::get\_numerical\_value\_impl(), and qbar.

## 6.38.3.8 uses\_Laurent\_series()

```
bool GiNaC::Eisenstein_h_kernel::uses_Laurent_series ( ) const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method Laurent\_series needs to be implemented).

Returns false if this is not the case (and the class has an implementation of series\_coeff\_impl).

Reimplemented from [GiNaC::integration\\_kernel](#).

## 6.38.3.9 coefficient\_a0()

```
ex GiNaC::Eisenstein_h_kernel::coefficient_a0 (
    const numeric & k,
    const numeric & r,
    const numeric & s,
    const numeric & N ) const
```

The constant coefficient in the Fourier expansion.

References GiNaC::Bernoulli\_polynomial(), GiNaC::cos(), GiNaC::I, GiNaC::irem(), k, GiNaC::mod(), N, GiNaC::Pi, r, s, and GiNaC::sin().

Referenced by q\_expansion\_modular\_form().

#### 6.38.3.10 coefficient\_an()

```
ex GiNaC::Eisenstein_h_kernel::coefficient_an (
    const numeric & n,
    const numeric & k,
    const numeric & r,
    const numeric & s,
    const numeric & N ) const
```

The higher coefficients in the Fourier expansion.

References GiNaC::exp(), GiNaC::l, GiNaC::irem(), k, m, GiNaC::mod(), N, n, GiNaC::Pi, GiNaC::pow(), r, and s.

Referenced by q\_expansion\_modular\_form().

#### 6.38.3.11 q\_expansion\_modular\_form()

```
ex GiNaC::Eisenstein_h_kernel::q_expansion_modular_form (
    const ex & q,
    int order ) const
```

References coefficient\_a0(), coefficient\_an(), k, N, GiNaC::pow(), r, and s.

Referenced by Laurent\_series(), and series().

#### 6.38.3.12 do\_print()

```
void GiNaC::Eisenstein_h_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References c, C\_norm, k, N, GiNaC::ex::print(), r, and s.

### 6.38.4 Member Data Documentation

#### 6.38.4.1 k

```
ex GiNaC::Eisenstein_h_kernel::k [protected]
```

Referenced by coefficient\_a0(), coefficient\_an(), do\_print(), is\_numeric(), let\_op(), op(), and q\_expansion\_modular\_form().

## 6.38.4.2 N

`ex GiNaC::Eisenstein_h_kernel::N [protected]`

Referenced by `coefficient_a0()`, `coefficient_an()`, `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `q_expansion_modular_form()`.

## 6.38.4.3 r

`ex GiNaC::Eisenstein_h_kernel::r [protected]`

Referenced by `coefficient_a0()`, `coefficient_an()`, `do_print()`, `is_numeric()`, `let_op()`, `op()`, `q_expansion_modular_form()`, and `series()`.

## 6.38.4.4 s

`ex GiNaC::Eisenstein_h_kernel::s [protected]`

Referenced by `coefficient_a0()`, `coefficient_an()`, `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `q_expansion_modular_form()`.

## 6.38.4.5 C\_norm

`ex GiNaC::Eisenstein_h_kernel::C_norm [protected]`

Referenced by `do_print()`, `get_numerical_value()`, `is_numeric()`, `Laurent_series()`, `let_op()`, and `op()`.

The documentation for this class was generated from the following files:

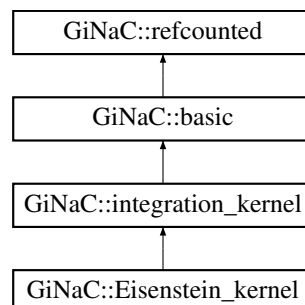
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.39 GiNaC::Eisenstein\_kernel Class Reference

The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Eisenstein\_kernel:



## Public Member Functions

- `Eisenstein_kernel` (const `ex` &k, const `ex` &N, const `ex` &a, const `ex` &b, const `ex` &K, const `ex` &C\_norm=numeric(1))
- `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const override  
*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex Laurent_series` (const `ex` &x, int `order`) const override  
*Returns the Laurent series, starting possibly with the pole term.*
- `ex get_numerical_value` (const `ex` &qbar, int N\_trunc=0) const override  
*Returns the value of the modular form.*
- `ex q_expansion_modular_form` (const `ex` &q, int `order`) const

## Protected Member Functions

- `bool uses_Laurent_series` () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method Laurent\_series needs to be implemented).*
- `void do_print` (const `print_context` &c, unsigned level) const

## Protected Attributes

- `ex k`
- `ex N`
- `ex a`
- `ex b`
- `ex K`
- `ex C_norm`

### 6.39.1 Detailed Description

The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .

This class represents the differential one-form

$$\omega_{k,N,a,b,K}^{\text{Eisenstein}} = C_k E_{k,N,a,b,K}(\tau) \frac{d\bar{q}_N}{\bar{q}_N}$$

The integers a and b are either one or the discriminant of a quadratic number field. This class represents Eisenstein series, which can be defined by primitive Dirichlet characters from the Kronecker symbol. This implies that the characters take the values -1,0,1, i.e. no higher roots of unity occur. The

$$\bar{q}$$

-expansion has then rational coefficients.

Ref.: W. Stein, Modular Forms: A Computational Approach, Chapter 5

## 6.39.2 Constructor & Destructor Documentation

### 6.39.2.1 Eisenstein\_kernel()

```
GiNaC::Eisenstein_kernel::Eisenstein_kernel (
    const ex & k,
    const ex & N,
    const ex & a,
    const ex & b,
    const ex & K,
    const ex & C_norm = numeric(1) )
```

## 6.39.3 Member Function Documentation

### 6.39.3.1 series()

```
ex GiNaC::Eisenstein_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.

This allows for easy use in the class [modular\\_form\\_kernel](#).

Reimplemented from [GiNaC::integration\\_kernel](#).

References [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [qbar](#), [r](#), and [GiNaC::ex::series\(\)](#).

### 6.39.3.2 nops()

```
size_t GiNaC::Eisenstein_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

**6.39.3.3 op()**

```
ex GiNaC::Eisenstein_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References a, b, C\_norm, k, K, and N.

**6.39.3.4 let\_op()**

```
ex & GiNaC::Eisenstein_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References a, b, C\_norm, [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), k, K, and N.

**6.39.3.5 is\_numeric()**

```
bool GiNaC::Eisenstein_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References a, b, C\_norm, [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), k, K, N, [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), and [GiNaC::info\\_flags::posint](#).

**6.39.3.6 Laurent\_series()**

```
ex GiNaC::Eisenstein_kernel::Laurent_series (
    const ex & x,
    int order ) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References C\_norm, order, [q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::ex::series\(\)](#), and x.

#### 6.39.3.7 get\_numerical\_value()

```
ex GiNaC::Eisenstein_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References C\_norm, [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and qbar.

#### 6.39.3.8 uses\_Laurent\_series()

```
bool GiNaC::Eisenstein_kernel::uses_Laurent_series ( ) const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

#### 6.39.3.9 q\_expansion\_modular\_form()

```
ex GiNaC::Eisenstein_kernel::q_expansion_modular_form (
    const ex & q,
    int order ) const
```

References a, b, k, K, N, and order.

Referenced by `Laurent_series()`, and `series()`.

#### 6.39.3.10 do\_print()

```
void GiNaC::Eisenstein_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References a, b, c, C\_norm, k, K, N, and [GiNaC::ex::print\(\)](#).

### 6.39.4 Member Data Documentation

#### 6.39.4.1 k

`ex GiNaC::Eisenstein_kernel::k [protected]`

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `q_expansion_modular_form()`.

#### 6.39.4.2 N

`ex GiNaC::Eisenstein_kernel::N [protected]`

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `q_expansion_modular_form()`.

#### 6.39.4.3 a

`ex GiNaC::Eisenstein_kernel::a [protected]`

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `q_expansion_modular_form()`.

#### 6.39.4.4 b

`ex GiNaC::Eisenstein_kernel::b [protected]`

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `q_expansion_modular_form()`.

#### 6.39.4.5 K

`ex GiNaC::Eisenstein_kernel::K [protected]`

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `q_expansion_modular_form()`.

#### 6.39.4.6 C\_norm

`ex GiNaC::Eisenstein_kernel::C_norm [protected]`

Referenced by `do_print()`, `get_numerical_value()`, `is_numeric()`, `Laurent_series()`, `let_op()`, and `op()`.

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)



## 6.40 GiNaC::composition\_generator::coolmulti::element Struct Reference

```
#include <utils.h>
```

### Public Member Functions

- [element](#) (unsigned val, [element](#) \*n)
- [~element](#) ()

### Public Attributes

- unsigned [value](#)
- [element](#) \* [next](#)

### 6.40.1 Constructor & Destructor Documentation

#### 6.40.1.1 element()

```
GiNaC::composition_generator::coolmulti::element::element (
    unsigned val,
    element * n ) [inline]
```

#### 6.40.1.2 ~element()

```
GiNaC::composition_generator::coolmulti::element::~~element ( ) [inline]
```

References next.

### 6.40.2 Member Data Documentation

#### 6.40.2.1 value

```
unsigned GiNaC::composition_generator::coolmulti::element::value
```

Referenced by [GiNaC::composition\\_generator::coolmulti::finished\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), and [GiNaC::composition\\_generator::coolmulti::next\\_permutation\(\)](#).

### 6.40.2.2 next

```
element* GiNaC::composition_generator::coolmulti::element::next
```

Referenced by `GiNaC::composition_generator::coolmulti::coolmulti()`, `GiNaC::composition_generator::coolmulti::finished()`, `GiNaC::composition_generator::get()`, `GiNaC::composition_generator::coolmulti::next_permutation()`, and `~element()`.

The documentation for this struct was generated from the following file:

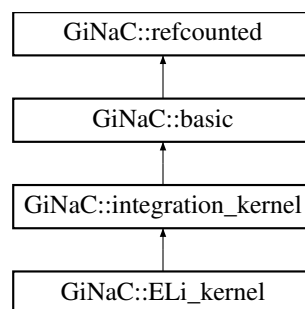
- [utils.h](#)

## 6.41 GiNaC::ELi\_kernel Class Reference

The ELi-kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for `GiNaC::ELi_kernel`:



### Public Member Functions

- [ELi\\_kernel](#) (const [ex](#) &n, const [ex](#) &m, const [ex](#) &x, const [ex](#) &y)
- [size\\_t nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex & let\\_op](#) (size\_t i) override  
*Return modifiable operand/member at position i.*
- [bool is\\_numeric](#) (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- [ex get\\_numerical\\_value](#) (const [ex](#) &qbar, int N\_trunc=0) const override  
*Returns the value of  $ELi_{\{n,m\}}(x,y,qbar)$*

### Protected Member Functions

- [cl::cl\\_N series\\_coeff\\_impl](#) (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- [void do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Attributes

- [ex n](#)
- [ex m](#)
- [ex x](#)
- [ex y](#)

### 6.41.1 Detailed Description

The ELi-kernel.

This class represents the differential one-form

$$\omega_{n;m}^{\text{ELi}}(x; y) = \text{ELi}_{n;m}(x; y; \bar{q}) \frac{d\bar{q}}{\bar{q}}$$

### 6.41.2 Constructor & Destructor Documentation

#### 6.41.2.1 ELi\_kernel()

```
GiNaC::ELi_kernel::ELi_kernel (
    const ex & n,
    const ex & m,
    const ex & x,
    const ex & y )
```

### 6.41.3 Member Function Documentation

#### 6.41.3.1 nops()

```
size_t GiNaC::ELi_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 6.41.3.2 op()

```
ex GiNaC::ELi_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References m, n, x, and y.

#### 6.41.3.3 let\_op()

```
ex & GiNaC::ELi_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), m, n, x, and y.

#### 6.41.3.4 is\_numeric()

```
bool GiNaC::ELi_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), m, n, [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), x, and y.

#### 6.41.3.5 get\_numerical\_value()

```
ex GiNaC::ELi_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of  $ELi_{\{n,m\}}(x,y,qbar)$

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and qbar.

#### 6.41.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::ELi_kernel::series_coeff_impl (
    int i ) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The i-th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), k, m, n, [GiNaC::to\\_int\(\)](#), x, and y.

#### 6.41.3.7 do\_print()

```
void GiNaC::ELi_kernel::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References `c`, `m`, `n`, `GiNaC::ex::print()`, `x`, and `y`.

### 6.41.4 Member Data Documentation

#### 6.41.4.1 n

```
ex GiNaC::ELi_kernel::n [protected]
```

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

#### 6.41.4.2 m

```
ex GiNaC::ELi_kernel::m [protected]
```

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

#### 6.41.4.3 x

```
ex GiNaC::ELi_kernel::x [protected]
```

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

#### 6.41.4.4 y

```
ex GiNaC::ELi_kernel::y [protected]
```

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.42 std::equal\_to< GiNaC::ex > Struct Template Reference

Specialization of std::equal\_to() for ex objects.

```
#include <ex.h>
```

### Public Member Functions

- bool [operator\(\)](#) (const [GiNaC::ex](#) &e1, const [GiNaC::ex](#) &e2) const noexcept

### 6.42.1 Detailed Description

```
template<>
struct std::equal_to< GiNaC::ex >
```

Specialization of std::equal\_to() for ex objects.

### 6.42.2 Member Function Documentation

#### 6.42.2.1 operator>()

```
bool std::equal_to< GiNaC::ex >::operator() (
    const GiNaC::ex & e1,
    const GiNaC::ex & e2 ) const    [inline], [noexcept]
```

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.43 GiNaC::error\_and\_integral Struct Reference

### Public Member Functions

- [error\\_and\\_integral](#) (const [ex](#) &err, const [ex](#) &integ)

### Public Attributes

- [ex error](#)
- [ex integral](#)

### 6.43.1 Constructor & Destructor Documentation

#### 6.43.1.1 error\_and\_integral()

```
GiNaC::error_and_integral::error_and_integral (
    const ex & err,
    const ex & integ ) [inline]
```

### 6.43.2 Member Data Documentation

#### 6.43.2.1 error

[ex](#) GiNaC::error\_and\_integral::error

Referenced by `GiNaC::error_and_integral_is_less::operator()()`.

#### 6.43.2.2 integral

[ex](#) GiNaC::error\_and\_integral::integral

Referenced by `GiNaC::error_and_integral_is_less::operator()()`.

The documentation for this struct was generated from the following file:

- [integral.cpp](#)

## 6.44 GiNaC::error\_and\_integral\_is\_less Struct Reference

### Public Member Functions

- bool [operator\(\)](#) (const [error\\_and\\_integral](#) &*e1*, const [error\\_and\\_integral](#) &*e2*) const

### 6.44.1 Member Function Documentation

#### 6.44.1.1 operator()

```
bool GiNaC::error_and_integral_is_less::operator() (
    const error\_and\_integral & e1,
    const error\_and\_integral & e2 ) const [inline]
```

References [c](#), [GiNaC::ex::compare\(\)](#), [GiNaC::error\\_and\\_integral::error](#), and [GiNaC::error\\_and\\_integral::integral](#).

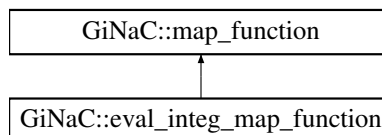
The documentation for this struct was generated from the following file:

- [integral.cpp](#)

## 6.45 GiNaC::eval\_integ\_map\_function Struct Reference

Function object to be applied by [basic::eval\\_integ\(\)](#).

Inheritance diagram for [GiNaC::eval\\_integ\\_map\\_function](#):



### Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &e) override

### Additional Inherited Members

#### 6.45.1 Detailed Description

Function object to be applied by [basic::eval\\_integ\(\)](#).

#### 6.45.2 Member Function Documentation

##### 6.45.2.1 operator()

```
ex GiNaC::eval_integ_map_function::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::eval\\_integ\(\)](#).

The documentation for this struct was generated from the following file:

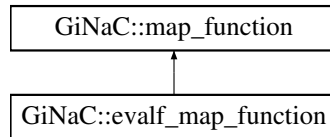
- [basic.cpp](#)



## 6.46 GiNaC::evalf\_map\_function Struct Reference

Function object to be applied by [basic::evalf\(\)](#).

Inheritance diagram for GiNaC::evalf\_map\_function:



### Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &e) override

### Additional Inherited Members

#### 6.46.1 Detailed Description

Function object to be applied by [basic::evalf\(\)](#).

#### 6.46.2 Member Function Documentation

##### 6.46.2.1 operator()

```
ex GiNaC::evalf_map_function::operator() (  
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::evalf\(\)](#).

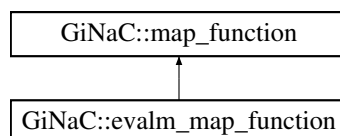
The documentation for this struct was generated from the following file:

- [basic.cpp](#)

## 6.47 GiNaC::evalm\_map\_function Struct Reference

Function object to be applied by [basic::evalm\(\)](#).

Inheritance diagram for GiNaC::evalm\_map\_function:



## Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &*e*) override

## Additional Inherited Members

### 6.47.1 Detailed Description

Function object to be applied by [basic::evalm\(\)](#).

### 6.47.2 Member Function Documentation

#### 6.47.2.1 operator()

```
ex GiNaC::evalm_map_function::operator() (  
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::evalm\(\)](#).

The documentation for this struct was generated from the following file:

- [basic.cpp](#)

## 6.48 GiNaC::ex Class Reference

Lightweight wrapper for [GiNaC](#)'s symbolic objects.

```
#include <ex.h>
```

## Public Member Functions

- `ex ()` noexcept
- `ex (const basic &other)`
- `ex (int i)`
- `ex (unsigned int i)`
- `ex (long i)`
- `ex (unsigned long i)`
- `ex (long long i)`
- `ex (unsigned long long i)`
- `ex (double const d)`
- `ex (const std::string &s, const ex &l)`  
*Construct ex from string and a list of symbols.*
- `void swap (ex &other)` noexcept  
*Efficiently swap the contents of two expressions.*
- `const_iterator begin ()` const noexcept
- `const_iterator end ()` const noexcept
- `const_preorder_iterator preorder_begin ()` const
- `const_preorder_iterator preorder_end ()` const noexcept
- `const_postorder_iterator postorder_begin ()` const
- `const_postorder_iterator postorder_end ()` const noexcept
- `ex eval ()` const
- `ex evalf ()` const
- `ex evalm ()` const
- `ex eval_ncmul (const exvector &v)` const
- `ex eval_integ ()` const
- `void print (const print_context &c, unsigned level=0)` const  
*Print expression to stream.*
- `void dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- `void dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- `bool info (unsigned inf)` const
- `size_t nops ()` const
- `ex op (size_t i)` const
- `ex operator[] (const ex &index)` const
- `ex operator[] (size_t i)` const
- `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- `ex & operator[] (const ex &index)`
- `ex & operator[] (size_t i)`
- `ex lhs ()` const  
*Left hand side of relational expression.*
- `ex rhs ()` const  
*Right hand side of relational expression.*
- `ex conjugate ()` const
- `ex real_part ()` const
- `ex imag_part ()` const
- `bool has (const ex &pattern, unsigned options=0)` const
- `bool find (const ex &pattern, exset &found)` const  
*Find all occurrences of a pattern.*
- `bool match (const ex &pattern)` const  
*Check whether expression matches a specified pattern.*

- `bool match` (const `ex` &pattern, `exmap` &repls) const
- `ex subs` (const `exmap` &m, unsigned `options`=0) const
- `ex subs` (const `lst` &ls, const `lst` &lr, unsigned `options`=0) const
 

*Substitute objects in an expression (syntactic substitution) and return the result as a new expression.*
- `ex subs` (const `ex` &e, unsigned `options`=0) const
 

*Substitute objects in an expression (syntactic substitution) and return the result as a new expression.*
- `ex map` (`map_function` &f) const
- `ex map` (`ex`(\*f)(const `ex` &e)) const
- `void accept` (`visitor` &v) const
- `void traverse_preorder` (`visitor` &v) const
 

*Traverse expression tree with given visitor, preorder traversal.*
- `void traverse_postorder` (`visitor` &v) const
 

*Traverse expression tree with given visitor, postorder traversal.*
- `void traverse` (`visitor` &v) const
- `bool is_polynomial` (const `ex` &vars) const
 

*Check whether expression is a polynomial.*
- `int degree` (const `ex` &s) const
- `int ldegree` (const `ex` &s) const
- `ex coeff` (const `ex` &s, int `n`=1) const
- `ex lcoeff` (const `ex` &s) const
- `ex tcoeff` (const `ex` &s) const
- `ex expand` (unsigned `options`=0) const
- `ex collect` (const `ex` &s, bool `distributed`=false) const
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const
 

*Compute partial derivative of an expression.*
- `ex series` (const `ex` &r, int `order`, unsigned `options`=0) const
 

*Compute the truncated series expansion of an expression.*
- `ex normal` () const
 

*Normalization of rational functions.*
- `ex to_rational` (`exmap` &repl) const
 

*Rationalization of non-rational functions.*
- `ex to_polynomial` (`exmap` &repl) const
- `ex numer` () const
 

*Get numerator of an expression.*
- `ex denom` () const
 

*Get denominator of an expression.*
- `ex numer_denom` () const
 

*Get numerator and denominator of an expression.*
- `ex unit` (const `ex` &x) const
 

*Compute unit part (= sign of leading coefficient) of a multivariate polynomial in  $Q[x]$ .*
- `ex content` (const `ex` &x) const
 

*Compute content part (= unit normal GCD of all coefficients) of a multivariate polynomial in  $Q[x]$ .*
- `numeric integer_content` () const
 

*Compute the integer content (= GCD of all numeric coefficients) of an expanded polynomial.*
- `ex primpart` (const `ex` &x) const
 

*Compute primitive part of a multivariate polynomial in  $Q[x]$ .*
- `ex primpart` (const `ex` &x, const `ex` &cont) const
 

*Compute primitive part of a multivariate polynomial in  $Q[x]$  when the content part is already known.*
- `void unitcontprim` (const `ex` &x, `ex` &u, `ex` &c, `ex` &p) const
 

*Compute unit part, content part, and primitive part of a multivariate polynomial in  $Q[x]$ .*
- `ex smod` (const `numeric` &xi) const
- `numeric max_coefficient` () const

*Return maximum (absolute value) coefficient of a polynomial.*

- `exvector get_free_indices ()` const
- `ex simplify_indexed` (unsigned `options=0`) const

*Simplify/canonicalize expression containing indexed objects.*

- `ex simplify_indexed` (const `scalar_products` &sp, unsigned `options=0`) const

*Simplify/canonicalize expression containing indexed objects.*

- int `compare` (const `ex` &other) const
- bool `is_equal` (const `ex` &other) const
- bool `is_zero` () const
- bool `is_zero_matrix` () const

*Check whether expression is zero or zero matrix.*

- `ex symmetrize` () const

*Symmetrize expression over its free indices.*

- `ex symmetrize` (const `lst` &l) const

*Symmetrize expression over a list of objects (symbols, indices).*

- `ex antisymmetrize` () const

*Antisymmetrize expression over its free indices.*

- `ex antisymmetrize` (const `lst` &l) const

*Antisymmetrize expression over a list of objects (symbols, indices).*

- `ex symmetrize_cyclic` () const

*Symmetrize expression by cyclic permutation over its free indices.*

- `ex symmetrize_cyclic` (const `lst` &l) const

*Symmetrize expression by cyclic permutation over a list of objects (symbols, indices).*

- unsigned `return_type` () const
- `return_type_t return_type_tinfo` () const
- unsigned `gethash` () const

## Private Member Functions

- void `makewritable` ()

*Make this ex writable (if more than one ex handle the same basic) by unlinking the object and creating an unshared copy of it.*

- void `share` (const `ex` &other) const

*Share equal objects between expressions.*

## Static Private Member Functions

- static `ptr< basic > construct_from_basic` (const `basic` &other)

*Helper function for the ex-from-basic constructor.*

- static `basic & construct_from_int` (int i)
- static `basic & construct_from_uint` (unsigned int i)
- static `basic & construct_from_long` (long i)
- static `basic & construct_from_ulong` (unsigned long i)
- static `basic & construct_from_longlong` (long long i)
- static `basic & construct_from_ulonglong` (unsigned long long i)
- static `basic & construct_from_double` (double d)
- static `ptr< basic > construct_from_string_and_lst` (const std::string &s, const `ex` &l)

## Private Attributes

- `ptr< basic > bp`  
*pointer to basic object managed by this*

## Friends

- class `archive_node`
- bool `are_ex_trivially_equal` (const `ex` &, const `ex` &)  
*Compare two objects of class quickly without doing a deep tree traversal.*
- template<class T >  
const T & `ex_to` (const `ex` &)  
*Return a reference to the basic-derived class T object embedded in an expression.*
- template<class T >  
bool `is_a` (const `ex` &)  
*Check if ex is a handle to a T, including base classes.*
- template<class T >  
bool `is_exactly_a` (const `ex` &)  
*Check if ex is a handle to a T, not including base classes.*

### 6.48.1 Detailed Description

Lightweight wrapper for `GiNaC`'s symbolic objects.

It holds a pointer to the other object in order to do garbage collection by the method of reference counting. I.e., it is a smart pointer. Also, the constructor `ex::ex(const basic & other)` calls the methods that do automatic evaluation. E.g., `x-x` turns automatically into 0.

### 6.48.2 Constructor & Destructor Documentation

#### 6.48.2.1 `ex()` [1/10]

```
GiNaC::ex::ex ( ) [inline], [noexcept]
```

References `bp`, `GiNaC::status_flags::dynallocated`, and `GINAC_ASSERT`.

#### 6.48.2.2 `ex()` [2/10]

```
GiNaC::ex::ex (
    const basic & other ) [inline]
```

References `bp`, `GiNaC::status_flags::dynallocated`, and `GINAC_ASSERT`.

**6.48.2.3 ex()** [3/10]

```
GiNaC::ex::ex (
    int i ) [inline]
```

References bp, GiNaC::status\_flags::dynallocated, and GINAC\_ASSERT.

**6.48.2.4 ex()** [4/10]

```
GiNaC::ex::ex (
    unsigned int i ) [inline]
```

References bp, GiNaC::status\_flags::dynallocated, and GINAC\_ASSERT.

**6.48.2.5 ex()** [5/10]

```
GiNaC::ex::ex (
    long i ) [inline]
```

References bp, GiNaC::status\_flags::dynallocated, and GINAC\_ASSERT.

**6.48.2.6 ex()** [6/10]

```
GiNaC::ex::ex (
    unsigned long i ) [inline]
```

References bp, GiNaC::status\_flags::dynallocated, and GINAC\_ASSERT.

**6.48.2.7 ex()** [7/10]

```
GiNaC::ex::ex (
    long long i ) [inline]
```

References bp, GiNaC::status\_flags::dynallocated, and GINAC\_ASSERT.

**6.48.2.8 ex()** [8/10]

```
GiNaC::ex::ex (
    unsigned long long i ) [inline]
```

References bp, GiNaC::status\_flags::dynallocated, and GINAC\_ASSERT.

### 6.48.2.9 `ex()` [9/10]

```
GiNaC::ex::ex (
    double const d ) [inline]
```

References `bp`, `GiNaC::status_flags::dynallocated`, and `GINAC_ASSERT`.

### 6.48.2.10 `ex()` [10/10]

```
GiNaC::ex::ex (
    const std::string & s,
    const ex & l ) [inline]
```

Construct `ex` from string and a list of symbols.

The input grammar is similar to the [GiNaC](#) output format. All symbols and indices to be used in the expression must be specified in a list in the second argument. Undefined symbols and other parser errors will throw an exception.

References `bp`, `GiNaC::status_flags::dynallocated`, and `GINAC_ASSERT`.

## 6.48.3 Member Function Documentation

### 6.48.3.1 `swap()`

```
void GiNaC::ex::swap (
    ex & other ) [inline], [noexcept]
```

Efficiently swap the contents of two expressions.

References `bp`, `GiNaC::status_flags::dynallocated`, and `GINAC_ASSERT`.

Referenced by `GiNaC::ncmul::derivative()`, `GiNaC::ex_swap::operator()()`, `GiNaC::expair::swap()`, `GiNaC::swap()`, and `std::swap()`.

### 6.48.3.2 `begin()`

```
const_iterator GiNaC::ex::begin ( ) const [inline], [noexcept]
```

Referenced by `GiNaC::abs_expand()`, `GiNaC::antisymmetrize()`, `GiNaC::canonicalize()`, `GiNaC::mul::derivative()`, `GiNaC::divide_in_z()`, `GiNaC::ncmul::evalm()`, `GiNaC::exp_expand()`, `GiNaC::mul::expandchildren()`, `GiNaC::G3↵_eval()`, `GiNaC::G3↵_evalf()`, `GiNaC::gcd()`, `GiNaC::Li_eval()`, `GiNaC::Li_evalf()`, `GiNaC::Li_print_latex()`, `GiNaC↵::log_expand()`, `GiNaC::rename_dummy_indices()`, `GiNaC::mul::series()`, `GiNaC::symmetrize()`, and `GiNaC↵::symmetrize_cyclic()`.



#### 6.48.3.3 end()

```
const_iterator GiNaC::ex::end ( ) const [inline], [noexcept]
```

References nops().

Referenced by GiNaC::abs\_expand(), GiNaC::canonicalize(), GiNaC::exp\_expand(), GiNaC::G3\_eval(), GiNaC::↔  
G3\_evalf(), GiNaC::Li\_print\_latex(), and GiNaC::log\_expand().

#### 6.48.3.4 preorder\_begin()

```
const_preorder_iterator GiNaC::ex::preorder_begin ( ) const [inline]
```

References nops().

#### 6.48.3.5 preorder\_end()

```
const_preorder_iterator GiNaC::ex::preorder_end ( ) const [inline], [noexcept]
```

#### 6.48.3.6 postorder\_begin()

```
const_postorder_iterator GiNaC::ex::postorder_begin ( ) const [inline]
```

References nops().

#### 6.48.3.7 postorder\_end()

```
const_postorder_iterator GiNaC::ex::postorder_end ( ) const [inline], [noexcept]
```

#### 6.48.3.8 eval()

```
ex GiNaC::ex::eval ( ) const [inline]
```

References bp.

Referenced by GiNaC::eval().

### 6.48.3.9 evalf()

```
ex GiNaC::ex::evalf ( ) const [inline]
```

References bp.

Referenced by `GiNaC::adaptivesimpson()`, `GiNaC::EllipticE_evalf()`, `GiNaC::EllipticK_evalf()`, `GiNaC::integral::evalf()`, `GiNaC::constant::evalf()`, `GiNaC::power::evalf()`, `GiNaC::evalf()`, `GiNaC::fsolve()`, `GiNaC::Kronecker_dtau_kernel::get_numerical_value()`, `GiNaC::integration_kernel::get_numerical_value_impl()`, `GiNaC::H_eval()`, `GiNaC::H_evalf()`, `GiNaC::multiple_polylog_kernel::is_numeric()`, `GiNaC::ELi_kernel::is_numeric()`, `GiNaC::Ebar_kernel::is_numeric()`, `GiNaC::Kronecker_dtau_kernel::is_numeric()`, `GiNaC::Kronecker_dz_kernel::is_numeric()`, `GiNaC::Eisenstein_kernel::is_numeric()`, `GiNaC::Eisenstein_h_kernel::is_numeric()`, `GiNaC::modular_form_kernel::is_numeric()`, `GiNaC::user_defined_kernel::is_numeric()`, `GiNaC::iterated_integral_evalf_impl()`, `GiNaC::Li_evalf()`, `GiNaC::S_evalf()`, `GiNaC::integration_kernel::series_coeff()`, `GiNaC::multiple_polylog_kernel::series_coeff_impl()`, `GiNaC::ELi_kernel::series_coeff_impl()`, `GiNaC::Ebar_kernel::series_coeff_impl()`, `GiNaC::Kronecker_dtau_kernel::series_coeff_impl()`, `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`, and `GiNaC::subvalue()`.

### 6.48.3.10 evalm()

```
ex GiNaC::ex::evalm ( ) const [inline]
```

References bp.

Referenced by `GiNaC::add::evalm()`, `GiNaC::pseries::evalm()`, `GiNaC::power::evalm()`, `GiNaC::evalm()`, and `is_zero_matrix()`.

### 6.48.3.11 eval\_ncmul()

```
ex GiNaC::ex::eval_ncmul (
    const exvector & v ) const [inline]
```

References bp.

Referenced by `GiNaC::integral::eval_ncmul()`, `GiNaC::relational::eval_ncmul()`, and `GiNaC::mul::eval_ncmul()`.

### 6.48.3.12 eval\_integ()

```
ex GiNaC::ex::eval_integ ( ) const [inline]
```

References bp.

Referenced by `GiNaC::integral::eval_integ()`, `GiNaC::pseries::eval_integ()`, and `GiNaC::eval_integ()`.

## 6.48.3.13 print()

```
void GiNaC::ex::print (
    const print\_context & c,
    unsigned level = 0 ) const
```

Print expression to stream.

The formatting of the output is determined by the kind of [print\\_context](#) object that is passed. Possible formattings include ginsh-parsable output (the default), tree-like output for debugging, and C++ source.

See also

[print\\_context](#)

References [bp](#), and [c](#).

Referenced by [GiNaC::abs\\_print\\_csrc\\_float\(\)](#), [GiNaC::abs\\_print\\_latex\(\)](#), [GiNaC::conjugate\\_print\\_latex\(\)](#), [GiNaC::integral::do\\_print\(\)](#), [GiNaC::relational::do\\_print\(\)](#), [GiNaC::mul::do\\_print\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::do\\_print\(\)](#), [GiNaC::ELi\\_kernel::do\\_print\(\)](#), [GiNaC::Ebar\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::do\\_print\(\)](#), [GiNaC::modular\\_form\\_kernel::do\\_print\(\)](#), [GiNaC::user\\_defined\\_kernel::do\\_print\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::idx::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::power::do\\_print\\_dflt\(\)](#), [GiNaC::integral::do\\_print\\_latex\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::relational::do\\_print\\_python\\_repr\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::power::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [GiNaC::pseries::do\\_print\\_python\\_repr\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), [GiNaC::factorial\\_print\\_dflt\\_latex\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::imag\\_part\\_print\\_latex\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::expair::print\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::idx::print\\_index\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [GiNaC::power::print\\_power\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::print\\_sym\\_pow\(\)](#), [GiNaC::real\\_part\\_print\\_latex\(\)](#), [GiNaC::S\\_print\\_latex\(\)](#), and [GiNaC::zeta1\\_print\\_latex\(\)](#).

## 6.48.3.14 dbgprint()

```
void GiNaC::ex::dbgprint ( ) const
```

Little wrapper around print to be called within a debugger.

References [bp](#).

## 6.48.3.15 dbgprinttree()

```
void GiNaC::ex::dbgprinttree ( ) const
```

Little wrapper around printtree to be called within a debugger.

References [bp](#).

## 6.48.3.16 info()

```
bool GiNaC::ex::info (
    unsigned int ) const [inline]
```

References bp.

Referenced by GiNaC::abs\_eval(), GiNaC::abs\_info(), GiNaC::abs\_power(), GiNaC::acos\_eval(), GiNaC::acosh\_←  
\_eval(), GiNaC::asin\_eval(), GiNaC::asinh\_conjugate(), GiNaC::asinh\_eval(), GiNaC::atan2\_eval(), GiNaC\_←  
::atan\_conjugate(), GiNaC::atan\_eval(), GiNaC::atanh\_eval(), GiNaC::beta\_eval(), GiNaC::beta\_series(), Gi\_←  
NaC::mul::can\_be\_further\_expanded(), GiNaC::mul::can\_make\_flat(), GiNaC::pseries::conjugate(), GiNaC\_←  
::power::conjugate(), content(), GiNaC::cos\_eval(), GiNaC::cosh\_eval(), GiNaC::csgn\_series(), GiNaC::matrix\_←  
::determinant(), GiNaC::divide(), GiNaC::divide\_in\_z(), GiNaC::add::do\_print\_csrc(), GiNaC::power::do\_print\_←  
csrc(), GiNaC::idx::do\_print\_csrc(), GiNaC::eta\_eval(), GiNaC::eta\_evalf(), GiNaC::eta\_series(), GiNaC::power\_←  
::eval(), GiNaC::exp\_eval(), GiNaC::exp\_power(), GiNaC::power::expand(), GiNaC::func\_arg\_info(), GiNaC::\_←  
G2\_eval(), GiNaC::G2\_evalf(), GiNaC::G3\_eval(), GiNaC::G3\_evalf(), GiNaC::gcd(), GiNaC::H\_eval(), GiNa\_←  
C::H\_evalf(), GiNaC::power::has(), GiNaC::heur\_gcd(), GiNaC::idx::idx(), GiNaC::pseries::imag\_part(), GiNa\_←  
C::power::imag\_part(), GiNaC::add::info(), GiNaC::mul::info(), GiNaC::power::info(), GiNaC::multiple\_polylog\_←  
kernel::is\_numeric(), GiNaC::ELi\_kernel::is\_numeric(), GiNaC::Ebar\_kernel::is\_numeric(), GiNaC::Kronecker\_←  
dtau\_kernel::is\_numeric(), GiNaC::Kronecker\_dz\_kernel::is\_numeric(), GiNaC::Eisenstein\_kernel::is\_numeric(),  
GiNaC::Eisenstein\_h\_kernel::is\_numeric(), GiNaC::modular\_form\_kernel::is\_numeric(), GiNaC::user\_defined\_←  
kernel::is\_numeric(), GiNaC::power::is\_polynomial(), GiNaC::iterated\_integral2\_eval(), GiNaC::iterated\_integral3\_←  
\_eval(), GiNaC::iterated\_integral\_evalf\_impl(), GiNaC::lcm(), GiNaC::lcmcoeff(), GiNaC::lgamma\_conjugate(),  
GiNaC::lgamma\_eval(), GiNaC::lgamma\_series(), GiNaC::Li2\_conjugate(), GiNaC::Li2\_eval(), GiNaC::Li2\_←  
series(), GiNaC::Li\_eval(), GiNaC::Li\_evalf(), GiNaC::Li\_series(), GiNaC::log\_conjugate(), GiNaC::log\_eval(),  
GiNaC::log\_expand(), GiNaC::log\_imag\_part(), GiNaC::log\_real\_part(), GiNaC::log\_series(), GiNaC::lsolve(),  
GiNaC::power::normal(), GiNaC::Order\_imag\_part(), GiNaC::matrix::pow(), GiNaC::prem(), GiNaC::psi1\_eval(),  
GiNaC::psi1\_series(), GiNaC::psi2\_eval(), GiNaC::psi2\_series(), GiNaC::quo(), GiNaC::pseries::real\_part(), Gi\_←  
NaC::power::real\_part(), GiNaC::rem(), GiNaC::resultant(), GiNaC::S\_eval(), GiNaC::S\_evalf(), GiNaC::S\_series(),  
GiNaC::mul::series(), GiNaC::power::series(), GiNaC::sin\_eval(), GiNaC::sinh\_eval(), GiNaC::sprem(), GiNaC\_←  
::step\_series(), subs(), GiNaC::tan\_eval(), GiNaC::tanh\_eval(), GiNaC::tgamma\_eval(), GiNaC::tgamma\_series(),  
GiNaC::expairseq::to\_polynomial(), GiNaC::power::to\_polynomial(), GiNaC::expairseq::to\_rational(), GiNaC\_←  
::power::to\_rational(), GiNaC::matrix::trace(), GiNaC::tryfactsubs(), unitcontprim(), GiNaC::zeta2\_deriv(), and  
GiNaC::zeta2\_eval().

## 6.48.3.17 nops()

```
size_t GiNaC::ex::nops ( ) const [inline]
```

References bp.

Referenced by GiNaC::abs\_expand(), GiNaC::matrix::add\_indexed(), GiNaC::algebraic\_match\_mul\_with\_mul(),  
GiNaC::ncmul::append\_factors(), GiNaC::basic::collect(), GiNaC::collect\_symbols(), GiNaC::color\_trace(), Gi\_←  
NaC::ncmul::count\_factors(), GiNaC::csgn\_eval(), GiNaC::const\_postorder\_iterator::descend(), GiNaC::divide(),  
end(), GiNaC::indexed::eval(), GiNaC::exp\_expand(), GiNaC::integral::expand(), GiNaC::mul::expand(), GiNa\_←  
C::indexed::expand(), find(), GiNaC::find\_common\_factor(), GiNaC::G2\_eval(), GiNaC::G2\_evalf(), GiNaC::\_←  
G3\_eval(), GiNaC::G3\_evalf(), GiNaC::gcd\_pf\_mul(), GiNaC::get\_all\_dummy\_indices\_safely(), GiNaC::get\_←  
first\_symbol(), GiNaC::get\_symbol\_stats(), GiNaC::H\_evalf(), GiNaC::hasindex(), GiNaC::haswild(), GiNaC\_←  
::const\_preorder\_iterator::increment(), GiNaC::lcmcoeff(), GiNaC::Li2\_series(), GiNaC::Li\_deriv(), GiNaC::Li\_←  
eval(), GiNaC::Li\_evalf(), GiNaC::log\_expand(), GiNaC::lsolve(), GiNaC::basic::match(), GiNaC::multiply\_lcm(),  
GiNaC::nops(), postorder\_begin(), preorder\_begin(), GiNaC::product\_to\_exvector(), GiNaC::rename\_dummy\_←  
\_indices\_uniquely(), GiNaC::reposition\_dummy\_indices(), GiNaC::integral::series(), GiNaC::simplify\_indexed(),  
GiNaC::step\_eval(), GiNaC::symminfo::symminfo(), traverse\_postorder(), traverse\_preorder(), GiNaC::zeta1\_←  
evalf(), and GiNaC::zeta2\_evalf().

## 6.48.3.18 op()

```
ex GiNaC::ex::op (
    size_t i ) const [inline]
```

References bp.

Referenced by GiNaC::abs\_eval(), GiNaC::matrix::add\_indexed(), GiNaC::algebraic\_match\_mul\_with\_mul(), GiNaC::ncmul::append\_factors(), GiNaC::atan\_series(), GiNaC::atanh\_series(), GiNaC::mul::can\_be\_further\_expanded(), GiNaC::basic::collect(), GiNaC::collect\_symbols(), GiNaC::color\_trace(), GiNaC::su3f::contract\_with(), GiNaC::su3d::contract\_with(), GiNaC::cos\_eval(), GiNaC::cosh\_eval(), GiNaC::ncmul::count\_factors(), GiNaC::csgn\_eval(), GiNaC::decomp\_rational(), denom(), GiNaC::const\_postorder\_iterator::descend(), GiNaC::divide(), GiNaC::divide\_in\_z(), GiNaC::epsilon\_tensor(), GiNaC::power::eval(), GiNaC::indexed::eval(), GiNaC::integral::eval\_integ(), GiNaC::exp\_eval(), GiNaC::integral::expand(), GiNaC::mul::expand(), GiNaC::indexed::expand(), find(), GiNaC::find\_common\_factor(), GiNaC::matrix::fraction\_free\_elimination(), GiNaC::G2\_eval(), GiNaC::G2\_evalf(), GiNaC::G3\_eval(), GiNaC::G3\_evalf(), GiNaC::gcd\_pf\_mul(), GiNaC::gcd\_pf\_pow(), GiNaC::gcd\_pf\_pow\_pow(), GiNaC::get\_all\_dummy\_indices\_safely(), GiNaC::get\_first\_symbol(), GiNaC::clifford::get\_metric(), GiNaC::H\_evalf(), GiNaC::power::has(), GiNaC::hasindex(), GiNaC::haswild(), GiNaC::const\_preorder\_iterator::increment(), GiNaC::lcmcoeff(), GiNaC::Li2\_series(), GiNaC::Li\_deriv(), GiNaC::Li\_evalf(), GiNaC::log\_eval(), GiNaC::lorentz\_eps(), GiNaC::lsolve(), GiNaC::basic::match(), GiNaC::multiply\_lcm(), GiNaC::power::normal(), normal(), GiNaC::basic::normal(), numer(), GiNaC::op(), GiNaC::op0\_is\_equal::operator>(), GiNaC::ex\_base::is\_less::operator>(), GiNaC::const\_iterator::operator\*(), GiNaC::const\_iterator::operator[](), GiNaC::product\_to\_exvector(), GiNaC::replace\_with\_symbol(), GiNaC::reposition\_dummy\_indices(), GiNaC::clifford::same\_metric(), GiNaC::matrix::scalar\_mul\_indexed(), GiNaC::simplify\_indexed(), GiNaC::sin\_eval(), GiNaC::sinh\_eval(), GiNaC::spmapkey::spmapkey(), GiNaC::sqrfree\_parfrac(), GiNaC::step\_eval(), subs(), GiNaC::symminfo::symminfo(), GiNaC::tan\_eval(), GiNaC::tanh\_eval(), traverse\_postorder(), traverse\_preorder(), GiNaC::tryfactsubs(), and GiNaC::zeta2\_deriv().

## 6.48.3.19 operator[]() [1/4]

```
ex GiNaC::ex::operator[] (
    const ex & index ) const [inline]
```

References bp.

## 6.48.3.20 operator[]() [2/4]

```
ex GiNaC::ex::operator[] (
    size_t i ) const [inline]
```

References bp.

## 6.48.3.21 let\_op()

```
ex & GiNaC::ex::let_op (
    size_t i )
```

Return modifiable operand/member at position i.

References bp, and makewriteable().

**6.48.3.22 operator[]()** [3/4]

```
ex & GiNaC::ex::operator[] (
    const ex & index )
```

References bp, and makewritable().

**6.48.3.23 operator[]()** [4/4]

```
ex & GiNaC::ex::operator[] (
    size_t i )
```

References bp, and makewritable().

**6.48.3.24 lhs()**

```
ex GiNaC::ex::lhs ( ) const
```

Left hand side of relational expression.

References bp.

Referenced by GiNaC::fsolve(), GiNaC::lhs(), GiNaC::pseries::pseries(), and GiNaC::Eisenstein\_h\_kernel::series().

**6.48.3.25 rhs()**

```
ex GiNaC::ex::rhs ( ) const
```

Right hand side of relational expression.

References bp.

Referenced by GiNaC::fsolve(), GiNaC::pseries::pseries(), GiNaC::rhs(), and GiNaC::Eisenstein\_h\_kernel::series().

**6.48.3.26 conjugate()**

```
ex GiNaC::ex::conjugate ( ) const [inline]
```

References bp.

Referenced by GiNaC::abs\_expl\_derivative(), GiNaC::abs\_power(), GiNaC::acos\_conjugate(), GiNaC::acosh↵  
\_conjugate(), GiNaC::asin\_conjugate(), GiNaC::asinh\_conjugate(), GiNaC::atan\_conjugate(), GiNaC::atanh↵  
conjugate(), GiNaC::integral::conjugate(), GiNaC::pseries::conjugate(), GiNaC::add::conjugate(), GiNaC::power↵  
::conjugate(), GiNaC::expair::conjugate(), GiNaC::conjugate(), GiNaC::conjugate\_eval(), GiNaC::cos\_conjugate(),  
GiNaC::cosh\_conjugate(), GiNaC::exp\_conjugate(), GiNaC::lgamma\_conjugate(), GiNaC::Li2\_conjugate(), GiNa↵  
C::log\_conjugate(), GiNaC::sin\_conjugate(), GiNaC::sinh\_conjugate(), GiNaC::tan\_conjugate(), GiNaC::tanh↵  
conjugate(), and GiNaC::tgamma\_conjugate().

## 6.48.3.27 real\_part()

```
ex GiNaC::ex::real_part ( ) const [inline]
```

References bp.

Referenced by GiNaC::abs\_eval(), GiNaC::asinh\_conjugate(), GiNaC::atan\_conjugate(), GiNaC::conjugate\_real↵\_part(), GiNaC::mul::find\_real\_imag(), GiNaC::pseries::imag\_part(), GiNaC::power::imag\_part(), GiNaC::pseries↵::real\_part(), GiNaC::add::real\_part(), GiNaC::power::real\_part(), GiNaC::real\_part(), and GiNaC::real\_part\_eval().

## 6.48.3.28 imag\_part()

```
ex GiNaC::ex::imag_part ( ) const [inline]
```

References bp.

Referenced by GiNaC::acos\_conjugate(), GiNaC::acosh\_conjugate(), GiNaC::asin\_conjugate(), GiNaC::asinh↵\_conjugate(), GiNaC::atan\_conjugate(), GiNaC::atanh\_conjugate(), GiNaC::conjugate\_imag\_part(), GiNaC::mul↵::find\_real\_imag(), GiNaC::add::imag\_part(), GiNaC::power::imag\_part(), GiNaC::imag\_part(), GiNaC::imag\_part↵\_eval(), GiNaC::lgamma\_conjugate(), GiNaC::Li2\_conjugate(), GiNaC::log\_conjugate(), and GiNaC::power::real↵\_part().

## 6.48.3.29 has()

```
bool GiNaC::ex::has (
    const ex & pattern,
    unsigned options = 0 ) const [inline]
```

References bp, and options.

Referenced by GiNaC::power::degree(), GiNaC::integral::eval(), GiNaC::integral::eval\_integ(), GiNaC::integral↵::expand(), GiNaC::basic::has(), GiNaC::has(), GiNaC::power::is\_polynomial(), and GiNaC::power::ldegree().

## 6.48.3.30 find()

```
bool GiNaC::ex::find (
    const ex & pattern,
    exset & found ) const
```

Find all occurrences of a pattern.

The found matches are appended to the "found" list. If the expression itself matches the pattern, the children are not further examined. This function returns true when any matches were found.

References find(), match(), nops(), and op().

Referenced by GiNaC::divide\_in\_z(), find(), GiNaC::find(), GiNaC::pseries::mul\_series(), and GiNaC::replace↵\_with\_symbol().

**6.48.3.31 match()** [1/2]

```
bool GiNaC::ex::match (
    const ex & pattern ) const
```

Check whether expression matches a specified pattern.

References bp.

Referenced by find(), GiNaC::power::has(), GiNaC::basic::match(), GiNaC::match(), and GiNaC::tryfactsubs().

**6.48.3.32 match()** [2/2]

```
bool GiNaC::ex::match (
    const ex & pattern,
    exmap & repls ) const [inline]
```

References bp.

**6.48.3.33 subs()** [1/3]

```
ex GiNaC::ex::subs (
    const exmap & m,
    unsigned options = 0 ) const [inline]
```

References bp, m, and options.

Referenced by GiNaC::adaptivesimpson(), GiNaC::mul::algebraic\_subs\_mul(), GiNaC::atan\_series(), GiNaC::atanh\_series(), GiNaC::beta\_series(), GiNaC::collect\_common\_factors(), GiNaC::integral::conjugate(), GiNaC::csgn\_series(), denom(), GiNaC::integral::derivative(), GiNaC::divide\_in\_z(), GiNaC::EllipticE\_series(), GiNaC::EllipticK\_series(), GiNaC::eta\_series(), GiNaC::integral::eval\_integ(), GiNaC::integral::evalf(), GiNaC::expand\_dummy\_sum(), GiNaC::fsolve(), GiNaC::gcd(), GiNaC::clifford::get\_metric(), GiNaC::H\_evalf(), GiNaC::heur\_gcd\_z(), GiNaC::user\_defined\_kernel::is\_numeric(), GiNaC::user\_defined\_kernel::Laurent\_series(), GiNaC::lgamma\_series(), GiNaC::Li2\_series(), GiNaC::Li\_series(), GiNaC::log\_series(), GiNaC::power::normal(), normal(), numer(), numer\_denom(), GiNaC::psi1\_series(), GiNaC::psi2\_series(), GiNaC::rename\_dummy\_indices(), GiNaC::rename\_dummy\_indices\_uniquely(), GiNaC::tensor::replace\_contr\_index(), GiNaC::replace\_with\_symbol(), GiNaC::reposition\_dummy\_indices(), GiNaC::S\_series(), GiNaC::integral::series(), GiNaC::power::series(), GiNaC::basic::series(), GiNaC::step\_series(), GiNaC::idx::subs(), GiNaC::pseries::subs(), GiNaC::relational::subs(), GiNaC::power::subs(), GiNaC::clifford::subs(), GiNaC::basic::subs(), GiNaC::subs(), GiNaC::basic::subs\_one\_level(), GiNaC::container< C >::subchildren(), GiNaC::subsvalue(), GiNaC::symm(), GiNaC::symmetrize\_cyclic(), GiNaC::tan\_series(), GiNaC::tanh\_series(), and GiNaC::tgamma\_series().



**6.48.3.34** subs() [2/3]

```
ex GiNaC::ex::subs (
    const lst & ls,
    const lst & lr,
    unsigned options = 0 ) const
```

Substitute objects in an expression (syntactic substitution) and return the result as a new expression.

References GiNaC::container< C >::begin(), bp, GiNaC::container< C >::end(), GINAC\_ASSERT, lr, m, GiNaC::container< C >::nops(), options, GiNaC::subs\_options::pattern\_is\_not\_product, and GiNaC::subs\_options::pattern\_is\_product.

**6.48.3.35** subs() [3/3]

```
ex GiNaC::ex::subs (
    const ex & e,
    unsigned options = 0 ) const
```

Substitute objects in an expression (syntactic substitution) and return the result as a new expression.

There are two valid types of replacement arguments: 1) a relational like object==ex and 2) a list of relationals lst{object1==ex1,object2==ex2,...}.

References bp, GINAC\_ASSERT, info(), GiNaC::info\_flags::list, m, op(), options, GiNaC::subs\_options::pattern\_is\_not\_product, GiNaC::subs\_options::pattern\_is\_product, r, and GiNaC::info\_flags::relation\_equal.

**6.48.3.36** map() [1/2]

```
ex GiNaC::ex::map (
    map_function & f ) const [inline]
```

References bp.

Referenced by GiNaC::color\_trace(), and GiNaC::expand\_dummy\_sum().

**6.48.3.37** map() [2/2]

```
ex GiNaC::ex::map (
    ex(*) (const ex &e) f ) const
```

#### 6.48.3.38 accept()

```
void GiNaC::ex::accept (
    visitor & v ) const [inline]
```

References bp.

Referenced by `traverse_postorder()`, and `traverse_preorder()`.

#### 6.48.3.39 traverse\_preorder()

```
void GiNaC::ex::traverse_preorder (
    visitor & v ) const
```

Traverse expression tree with given visitor, preorder traversal.

References `accept()`, `n`, `nops()`, `op()`, and `traverse_preorder()`.

Referenced by `traverse()`, and `traverse_preorder()`.

#### 6.48.3.40 traverse\_postorder()

```
void GiNaC::ex::traverse_postorder (
    visitor & v ) const
```

Traverse expression tree with given visitor, postorder traversal.

References `accept()`, `n`, `nops()`, `op()`, and `traverse_postorder()`.

Referenced by `traverse_postorder()`.

#### 6.48.3.41 traverse()

```
void GiNaC::ex::traverse (
    visitor & v ) const [inline]
```

References `traverse_preorder()`.

## 6.48.3.42 is\_polynomial()

```
bool GiNaC::ex::is_polynomial (
    const ex & vars ) const
```

Check whether expression is a polynomial.

References bp.

Referenced by GiNaC::power::is\_polynomial(), and GiNaC::is\_polynomial().

## 6.48.3.43 degree()

```
int GiNaC::ex::degree (
    const ex & s ) const [inline]
```

References bp.

Referenced by GiNaC::basic::collect(), GiNaC::power::degree(), GiNaC::degree(), GiNaC::divide(), GiNaC::divide\_in\_z(), GiNaC::get\_symbol\_stats(), GiNaC::heur\_gcd\_z(), lcoeff(), GiNaC::prem(), GiNaC::quo(), GiNaC::rem(), GiNaC::resultant(), GiNaC::sprem(), GiNaC::sqfree\_parfrac(), and GiNaC::sr\_gcd().

## 6.48.3.44 ldegree()

```
int GiNaC::ex::ldegree (
    const ex & s ) const [inline]
```

References bp.

Referenced by GiNaC::gcd\_pf\_pow(), GiNaC::get\_symbol\_stats(), GiNaC::power::ldegree(), GiNaC::ldegree(), GiNaC::Order\_series(), GiNaC::resultant(), GiNaC::mul::series(), GiNaC::power::series(), and tcoeff().

## 6.48.3.45 coeff()

```
ex GiNaC::ex::coeff (
    const ex & s,
    int n = 1 ) const [inline]
```

References bp, and n.

Referenced by GiNaC::Bernoulli\_polynomial(), GiNaC::pseries::coeff(), GiNaC::add::coeff(), GiNaC::mul::coeff(), GiNaC::coeff(), GiNaC::basic::collect(), GiNaC::pseries::convert\_to\_poly(), GiNaC::divide(), GiNaC::divide\_in\_z(), GiNaC::pseries::eval\_integ(), GiNaC::pseries::evalm(), GiNaC::pseries::expand(), GiNaC::power::expand\_add\_2(), GiNaC::generalised\_Bernoulli\_number(), GiNaC::add::imag\_part(), lcoeff(), GiNaC::lsolve(), GiNaC::pseries::normal(), GiNaC::prem(), GiNaC::pseries::print\_series(), GiNaC::quo(), GiNaC::add::real\_part(), GiNaC::rem(), GiNaC::resultant(), GiNaC::mul::series(), GiNaC::integration\_kernel::series\_coeff(), GiNaC::simplify\_indexed(), GiNaC::sprem(), GiNaC::sqfree\_parfrac(), and tcoeff().

**6.48.3.46 lcoeff()**

```
ex GiNaC::ex::lcoeff (
    const ex & s ) const [inline]
```

References `coeff()`, and `degree()`.

Referenced by `content()`, `GiNaC::get_symbol_stats()`, and `unit()`.

**6.48.3.47 tcoeff()**

```
ex GiNaC::ex::tcoeff (
    const ex & s ) const [inline]
```

References `coeff()`, and `ldegree()`.

**6.48.3.48 expand()**

```
ex GiNaC::ex::expand (
    unsigned options = 0 ) const
```

References `bp`, `GiNaC::status_flags::expanded`, and `options`.

Referenced by `GiNaC::abs_expand()`, `GiNaC::binomial_sym()`, `GiNaC::color_trace()`, `content()`, `GiNaC::matrix::determinant_minor()`, `GiNaC::divide()`, `GiNaC::divide_in_z()`, `GiNaC::exp_expand()`, `GiNaC::integral::expand()`, `GiNaC::pseries::expand()`, `GiNaC::power::expand()`, `GiNaC::mul::expand()`, `GiNaC::indexed::expand()`, `GiNaC::expand()`, `GiNaC::expand_dummy_sum()`, `GiNaC::ncmul::expandchildren()`, `GiNaC::mul::expandchildren()`, `GiNaC::mul::find_real_imag()`, `GiNaC::frac_cancel()`, `GiNaC::gcd()`, `GiNaC::heur_gcd_z()`, `GiNaC::log_expand()`, `GiNaC::lsolve()`, `GiNaC::expand_map_function::operator()`, `GiNaC::prem()`, `GiNaC::quo()`, `GiNaC::rem()`, `GiNaC::resultant()`, `GiNaC::mul::series()`, `GiNaC::power::series()`, `GiNaC::basic::series()`, `GiNaC::simplify_indexed()`, `GiNaC::sprem()`, `GiNaC::sqrfree_parfrac()`, `GiNaC::matrix::trace()`, `unit()`, and `unitcontprim()`.

**6.48.3.49 collect()**

```
ex GiNaC::ex::collect (
    const ex & s,
    bool distributed = false ) const [inline]
```

References `bp`.

Referenced by `GiNaC::matrix::charpoly()`, and `GiNaC::collect()`.

**6.48.3.50 diff()**

```
ex GiNaC::ex::diff (
    const symbol & s,
    unsigned nth = 1 ) const
```

Compute partial derivative of an expression.

## Parameters

<i>s</i>	symbol by which the expression is derived
<i>nth</i>	order of derivative (default 1)

## Returns

partial derivative as a new expression

References bp.

Referenced by `GiNaC::abs_expl_derivative()`, `GiNaC::conjugate_expl_derivative()`, `GiNaC::integral::derivative()`, `GiNaC::power::derivative()`, `GiNaC::basic::diff()`, `GiNaC::diff()`, `GiNaC::fsolve()`, `GiNaC::imag_part_expl_derivative()`, `GiNaC::log_series()`, `GiNaC::Order_expl_derivative()`, `GiNaC::real_part_expl_derivative()`, `GiNaC::basic::series()`, and `GiNaC::sqrfree_yun()`.

6.48.3.51 `series()`

```
ex GiNaC::ex::series (
    const ex & r,
    int order,
    unsigned options = 0 ) const
```

Compute the truncated series expansion of an expression.

This function returns an expression containing an object of class `pseries` to represent the series. If the series does not terminate within the given truncation order, the last term of the series will be an order term.

## Parameters

<i>r</i>	expansion relation, lhs holds variable and rhs holds point
<i>order</i>	truncation order of series calculations
<i>options</i>	of class <a href="#">series_options</a>

## Returns

an expression holding a `pseries` object

References `GiNaC::_ex0`, `bp`, `options`, `order`, and `r`.

Referenced by `GiNaC::Bernoulli_polynomial()`, `GiNaC::EllipticE_series()`, `GiNaC::EllipticK_series()`, `GiNaC::generalised_Bernoulli_number()`, `GiNaC::modular_form_kernel::is_numeric()`, `GiNaC::integration_kernel::Laurent_series()`, `GiNaC::Eisenstein_kernel::Laurent_series()`, `GiNaC::Eisenstein_h_kernel::Laurent_series()`, `GiNaC::modular_form_kernel::Laurent_series()`, `GiNaC::user_defined_kernel::Laurent_series()`, `GiNaC::Li2_series()`, `GiNaC::Li_series()`, `GiNaC::log_series()`, `GiNaC::S_series()`, `GiNaC::pseries::series()`, `GiNaC::add::series()`, `GiNaC::mul::series()`, `GiNaC::integral::series()`, `GiNaC::power::series()`, `GiNaC::Eisenstein_kernel::series()`, `GiNaC::Eisenstein_h_kernel::series()`, and `GiNaC::series()`.

6.48.3.52 `normal()`

```
ex GiNaC::ex::normal ( ) const
```

Normalization of rational functions.

This function converts an expression to its normal form "numerator/denominator", where numerator and denominator are (relatively prime) polynomials. Any subexpressions which are not rational functions (like non-rational numbers, non-integer powers or functions like `sin()`, `cos()` etc.) are replaced by temporary symbols which are re-substituted by the (normalized) subexpressions before `normal()` returns (this way, any expression can be treated as a rational function). `normal()` is applied recursively to arguments of functions etc.

**Returns**

normalized expression

References `bp`, `GINAC_ASSERT`, `GiNaC::subs_options::no_pattern`, `GiNaC::container< C >::nops()`, `GiNaC::container< C >::op()`, `op()`, and `subs()`.

Referenced by `GiNaC::matrix::determinant()`, `GiNaC::matrix::fraction_free_elimination()`, `GiNaC::matrix::gauss_elimination()`, `GiNaC::pseries::normal()`, `GiNaC::normal()`, and `GiNaC::matrix::trace()`.

6.48.3.53 `to_rational()`

```
ex GiNaC::ex::to_rational (
    exmap & repl ) const
```

Rationalization of non-rational functions.

This function converts a general expression to a rational function by replacing all non-rational subexpressions (like non-rational numbers, non-integer powers or functions like `sin()`, `cos()` etc.) to temporary symbols. This makes it possible to use functions like `gcd()` and `divide()` on non-rational functions by applying `to_rational()` on the arguments, calling the desired function and re-substituting the temporary symbols in the result. To make the last step possible, all temporary symbols and their associated expressions are collected in the map specified by the `repl` parameter, ready to be passed as an argument to `ex::subs()`.

**Parameters**

<i>repl</i>	collects all temporary symbols and their replacements
-------------	---

**Returns**

rationalized expression

References `bp`.

Referenced by `GiNaC::matrix::fraction_free_elimination()`, `GiNaC::power::to_polynomial()`, `GiNaC::expairseq::to_rational()`, `GiNaC::power::to_rational()`, and `GiNaC::to_rational()`.

#### 6.48.3.54 to\_polynomial()

```
ex GiNaC::ex::to_polynomial (
    exmap & repl ) const
```

References bp.

Referenced by GiNaC::find\_common\_factor(), GiNaC::expairseq::to\_polynomial(), GiNaC::power::to\_polynomial(), and GiNaC::to\_polynomial().

#### 6.48.3.55 numer()

```
ex GiNaC::ex::numer ( ) const
```

Get numerator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then the numerator is returned.

See also

[ex::normal](#)

Returns

numerator

References bp, GINAC\_ASSERT, GiNaC::subs\_options::no\_pattern, GiNaC::container< C >::nops(), GiNaC::container< C >::op(), op(), and subs().

Referenced by GiNaC::numer().

#### 6.48.3.56 denom()

```
ex GiNaC::ex::denom ( ) const
```

Get denominator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then the denominator is returned.

See also

[ex::normal](#)

Returns

denominator

References bp, GINAC\_ASSERT, GiNaC::subs\_options::no\_pattern, GiNaC::container< C >::nops(), GiNaC::container< C >::op(), op(), and subs().

Referenced by GiNaC::denom().

### 6.48.3.57 numer\_denom()

```
ex GiNaC::ex::numer_denom ( ) const
```

Get numerator and denominator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then a list [numerator, denominator] is returned.

See also

[ex::normal](#)

Returns

a list [numerator, denominator]

References `bp`, `GINAC_ASSERT`, `GiNaC::subs_options::no_pattern`, `GiNaC::container< C >::nops()`, `GiNaC::container< C >::op()`, and `subs()`.

Referenced by `GiNaC::matrix::fraction_free_elimination()`, and `GiNaC::numer_denom()`.

### 6.48.3.58 unit()

```
ex GiNaC::ex::unit (
    const ex & x ) const
```

Compute unit part (= sign of leading coefficient) of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The product of unit part, content part, and primitive part is the polynomial itself.

Parameters

$x$	main variable
-----	---------------

Returns

unit part

See also

[ex::content](#), [ex::primpart](#), [ex::unitcontprim](#)

References `GiNaC::_ex1`, `GiNaC::_ex_1`, `c`, `expand()`, `GiNaC::get_first_symbol()`, `lcoeff()`, `GiNaC::info_flags::negative`, and `x`.

Referenced by `content()`, `GiNaC::frac_cancel()`, `primpart()`, and `unitcontprim()`.



## 6.48.3.59 content()

```
ex GiNaC::ex::content (
    const ex & x ) const
```

Compute content part (= unit normal GCD of all coefficients) of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The product of unit part, content part, and primitive part is the polynomial itself.

## Parameters

x	main variable
---	---------------

## Returns

content part

## See also

[ex::unit](#), [ex::primpart](#), [ex::unitcontprim](#)

References [GiNaC::\\_ex0](#), [c](#), [expand\(\)](#), [GiNaC::gcd\(\)](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [integer\\_content\(\)](#), [is\\_zero\(\)](#), [lcoeff\(\)](#), [GiNaC::info\\_flags::negative](#), [r](#), [unit\(\)](#), and [x](#).

Referenced by [GiNaC::sr\\_gcd\(\)](#), and [unitcontprim\(\)](#).

## 6.48.3.60 integer\_content()

```
numeric GiNaC::ex::integer_content ( ) const
```

Compute the integer content (= GCD of all numeric coefficients) of an expanded polynomial.

For a polynomial with rational coefficients, this returns  $g/l$  where  $g$  is the GCD of the coefficients' numerators and  $l$  is the LCM of the coefficients' denominators.

## Returns

integer content

References [bp](#).

Referenced by [content\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::gcd\(\)](#), and [GiNaC::heur\\_gcd\\_z\(\)](#).

## 6.48.3.61 primpart() [1/2]

```
ex GiNaC::ex::primpart (
    const ex & x ) const
```

Compute primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The result will be a unit-normal polynomial with a content part of 1. The product of unit part, content part, and primitive part is the polynomial itself.

**Parameters**

<code>x</code>	main variable
----------------	---------------

**Returns**

primitive part

**See also**

[ex::unit](#), [ex::content](#), [ex::unitcontprim](#)

References [c](#), [unitcontprim\(\)](#), and [x](#).

Referenced by [GiNaC::sr\\_gcd\(\)](#).

**6.48.3.62 primpart()** [2/2]

```
ex GiNaC::ex::primpart (
    const ex & x,
    const ex & c ) const
```

Compute primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$  when the content part is already known.

This function is faster in computing the primitive part than the previous function.

**Parameters**

<code>x</code>	main variable
<code>c</code>	previously computed content part

**Returns**

primitive part

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [is\\_zero\(\)](#), [GiNaC::quo\(\)](#), [unit\(\)](#), and [x](#).

**6.48.3.63 unitcontprim()**

```
void GiNaC::ex::unitcontprim (
    const ex & x,
    ex & u,
    ex & c,
    ex & p ) const
```

Compute unit part, content part, and primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The product of the three parts is the polynomial itself.

## Parameters

$x$	main variable
$u$	unit part (returned)
$c$	content part (returned)
$p$	primitive part (returned)

## See also

[ex::unit](#), [ex::content](#), [ex::primpart](#)

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::abs\(\)](#), [c](#), [content\(\)](#), [expand\(\)](#), [info\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::quo\(\)](#), [unit\(\)](#), and [x](#).

Referenced by [GiNaC::gcd\(\)](#), and [primpart\(\)](#).

## 6.48.3.64 smod()

```
ex GiNaC::ex::smod (
    const numeric & xi ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::interpolate\(\)](#).

## 6.48.3.65 max\_coefficient()

```
numeric GiNaC::ex::max_coefficient ( ) const
```

Return maximum (absolute value) coefficient of a polynomial.

This function is used internally by [heur\\_gcd\(\)](#).

## Returns

maximum coefficient

## See also

[heur\\_gcd](#)

References [bp](#).

Referenced by [GiNaC::heur\\_gcd\\_z\(\)](#).

6.48.3.66 `get_free_indices()`

```
exvector GiNaC::ex::get_free_indices ( ) const [inline]
```

References bp.

Referenced by `antisymmetrize()`, `GiNaC::get_all_dummy_indices_safely()`, `GiNaC::integral::get_free_indices()`, `GiNaC::ncmul::get_free_indices()`, `GiNaC::add::get_free_indices()`, `GiNaC::mul::get_free_indices()`, `GiNaC::clifford::get_metric()`, `GiNaC::clifford::same_metric()`, `symmetrize()`, and `symmetrize_cyclic()`.

6.48.3.67 `simplify_indexed()` [1/2]

```
ex GiNaC::ex::simplify_indexed (
    unsigned options = 0 ) const
```

Simplify/canonicalize expression containing indexed objects.

This performs contraction of dummy indices where possible and checks whether the free indices in sums are consistent.

## Parameters

<i>options</i>	Simplification options (currently unused)
----------------	---

## Returns

simplified expression

References `GiNaC::simplify_indexed()`.

Referenced by `GiNaC::simplify_indexed()`.

6.48.3.68 `simplify_indexed()` [2/2]

```
ex GiNaC::ex::simplify_indexed (
    const scalar_products & sp,
    unsigned options = 0 ) const
```

Simplify/canonicalize expression containing indexed objects.

This performs contraction of dummy indices where possible, checks whether the free indices in sums are consistent, and automatically replaces scalar products by known values if desired.

## Parameters

<i>sp</i>	Scalar products to be replaced automatically
<i>options</i>	Simplification options (currently unused)

**Returns**

simplified expression

References `GiNaC::simplify_indexed()`.

**6.48.3.69 compare()**

```
int GiNaC::ex::compare (
    const ex & other ) const [inline]
```

References `bp`, and `share()`.

Referenced by `GiNaC::expair::compare()`, `GiNaC::expair::is_less()`, `GiNaC::expair_rest_is_less::operator()`, `GiNaC::error_and_integral_is_less::operator()`, `GiNaC::ex_is_less::operator()`, `GiNaC::term_info_is_less::operator()`, `GiNaC::symm_info_is_less_by_symmterm::operator()`, `GiNaC::symm_info_is_less_by_orig::operator()`, `GiNaC::spmapkey::operator<()`, and `GiNaC::spmapkey::spmapkey()`.

**6.48.3.70 is\_equal()**

```
bool GiNaC::ex::is_equal (
    const ex & other ) const [inline]
```

References `bp`, and `share()`.

Referenced by `GiNaC::abs_power()`, `GiNaC::acos_eval()`, `GiNaC::acosh_eval()`, `GiNaC::matrix::add_indexed()`, `GiNaC::asin_eval()`, `GiNaC::atan2_eval()`, `GiNaC::atan_eval()`, `GiNaC::atanh_eval()`, `GiNaC::atanh_series()`, `GiNaC::beta_eval()`, `GiNaC::pseries::coeff()`, `GiNaC::power::coeff()`, `GiNaC::add::combine_ex_with_coeff_to_pair()`, `GiNaC::mul::combine_pair_with_coeff_to_pair()`, `GiNaC::pseries::degree()`, `GiNaC::power::degree()`, `GiNaC::divide()`, `GiNaC::divide_in_z()`, `GiNaC::add::do_print_csrc()`, `GiNaC::power::do_print_csrc()`, `GiNaC::mul::do_print_csrc()`, `GiNaC::power::do_print_csrc_cl_N()`, `GiNaC::power::do_print_dflt()`, `GiNaC::power::do_print_latex()`, `GiNaC::epsilon_tensor()`, `GiNaC::power::eval()`, `GiNaC::matrix::eval_indexed()`, `GiNaC::tensdelta::eval_indexed()`, `GiNaC::tensmetric::eval_indexed()`, `GiNaC::mul::expand()`, `GiNaC::find_common_factor()`, `GiNaC::frac_cancel()`, `GiNaC::gcd()`, `GiNaC::gcd_pf_pow()`, `GiNaC::gcd_pf_pow_pow()`, `GiNaC::make_flat_inserter::handle_factor()`, `GiNaC::power::imag_part()`, `GiNaC::expair::is_canonical_numeric()`, `GiNaC::pseries::is_compatible_to()`, `GiNaC::idx::is_dummy_pair_same_type()`, `GiNaC::expair::is_equal()`, `is_zero()`, `GiNaC::pseries::lddegree()`, `GiNaC::power::lddegree()`, `GiNaC::Li2_eval()`, `GiNaC::Li2_series()`, `GiNaC::Li_eval()`, `GiNaC::log_eval()`, `GiNaC::lorentz_eps()`, `GiNaC::idx::match_same_type()`, `GiNaC::minimal_dim()`, `GiNaC::pseries::mul_series()`, `GiNaC::multiply_lcm()`, `GiNaC::power::normal()`, `GiNaC::idx_is_equal_ignore_dim::operator()`, `GiNaC::ex_is_equal::operator()`, `GiNaC::op0_is_equal::operator()`, `GiNaC::spmapkey::operator==()`, `GiNaC::add::print_add()`, `GiNaC::mul::print_overall_coeff()`, `GiNaC::product_to_exvector()`, `GiNaC::quo()`, `GiNaC::power::real_part()`, `GiNaC::mul::recombine_pair_to_ex()`, `GiNaC::rem()`, `GiNaC::replace_with_symbol()`, `GiNaC::clifford::same_metric()`, `GiNaC::pseries::series()`, `GiNaC::power::series()`, `GiNaC::simplify_indexed()`, `GiNaC::add::split_ex_to_pair()`, and `GiNaC::sqrfree_yun()`.

### 6.48.3.71 is\_zero()

```
bool GiNaC::ex::is_zero ( ) const [inline]
```

References `GiNaC::_ex0`, and `is_equal()`.

Referenced by `GiNaC::acos_conjugate()`, `GiNaC::acos_eval()`, `GiNaC::acosh_conjugate()`, `GiNaC::acosh_eval()`, `GiNaC::pseries::add_series()`, `GiNaC::asin_conjugate()`, `GiNaC::asin_eval()`, `GiNaC::asinh_eval()`, `GiNaC::atan()`, `GiNaC::atan2_eval()`, `GiNaC::atan_eval()`, `GiNaC::atanh_conjugate()`, `GiNaC::atanh_eval()`, `GiNaC::add::coeff()`, `content()`, `GiNaC::cosh_eval()`, `GiNaC::add::degree()`, `GiNaC::fderivative::derivative()`, `GiNaC::function::derivative()`, `GiNaC::matrix::determinant()`, `GiNaC::matrix::determinant_minor()`, `GiNaC::basic::diff()`, `GiNaC::divide()`, `GiNaC::divide_in_z()`, `GiNaC::add::do_print_csrc()`, `GiNaC::add::eval()`, `GiNaC::power::eval()`, `GiNaC::indexed::eval()`, `GiNaC::function::eval()`, `GiNaC::pseries::evalm()`, `GiNaC::exp_eval()`, `GiNaC::pseries::expand()`, `GiNaC::mul::expand()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_add_2()`, `GiNaC::find_common_factor()`, `GiNaC::mul::find_real_imag()`, `GiNaC::frac_cancel()`, `GiNaC::gcd()`, `GiNaC::heur_gcd_z()`, `GiNaC::add::imag_part()`, `GiNaC::add::info()`, `GiNaC::interpolate()`, `GiNaC::is_zero()`, `is_zero_matrix()`, `GiNaC::add::lddegree()`, `GiNaC::lgamma_conjugate()`, `GiNaC::Li2_conjugate()`, `GiNaC::Li2_eval()`, `GiNaC::Li2_series()`, `GiNaC::Li3_eval()`, `GiNaC::Li_series()`, `GiNaC::log()`, `GiNaC::log_conjugate()`, `GiNaC::log_eval()`, `GiNaC::log_series()`, `GiNaC::matrix::markowitz_elimination()`, `GiNaC::pseries::mul_series()`, `GiNaC::pseries::normal()`, `GiNaC::Order_eval()`, `GiNaC::prem()`, `primpart()`, `GiNaC::add::print_add()`, `GiNaC::pseries::print_series()`, `GiNaC::quo()`, `GiNaC::add::real_part()`, `GiNaC::rem()`, `GiNaC::S_series()`, `GiNaC::symbol::series()`, `GiNaC::power::series()`, `GiNaC::basic::series()`, `GiNaC::simplify_indexed()`, `GiNaC::simplify_indexed_product()`, `GiNaC::sinh_eval()`, `GiNaC::add::smod()`, `GiNaC::sprem()`, `GiNaC::sqfree_yun()`, `GiNaC::tanh_eval()`, `unitcontprim()`, and `GiNaC::indexed::validate()`.

### 6.48.3.72 is\_zero\_matrix()

```
bool GiNaC::ex::is_zero_matrix ( ) const
```

Check whether expression is zero or zero matrix.

References `evalm()`, and `is_zero()`.

### 6.48.3.73 symmetrize() [1/2]

```
ex GiNaC::ex::symmetrize ( ) const
```

Symmetrize expression over its free indices.

References `get_free_indices()`, and `GiNaC::symmetrize()`.

Referenced by `GiNaC::symmetrize()`.

### 6.48.3.74 symmetrize() [2/2]

```
ex GiNaC::ex::symmetrize (
    const lst & l ) const
```

Symmetrize expression over a list of objects (symbols, indices).

References `GiNaC::container< C >::begin()`, `GiNaC::container< C >::end()`, and `GiNaC::symm()`.

**6.48.3.75 antisymmetrize()** [1/2]

```
ex GiNaC::ex::antisymmetrize ( ) const
```

Antisymmetrize expression over its free indices.

References GiNaC::antisymmetrize(), and get\_free\_indices().

Referenced by GiNaC::antisymmetrize().

**6.48.3.76 antisymmetrize()** [2/2]

```
ex GiNaC::ex::antisymmetrize (
    const lst & l ) const
```

Antisymmetrize expression over a list of objects (symbols, indices).

References GiNaC::container< C >::begin(), GiNaC::container< C >::end(), and GiNaC::symm().

**6.48.3.77 symmetrize\_cyclic()** [1/2]

```
ex GiNaC::ex::symmetrize_cyclic ( ) const
```

Symmetrize expression by cyclic permutation over its free indices.

References get\_free\_indices(), and GiNaC::symmetrize\_cyclic().

Referenced by GiNaC::symmetrize\_cyclic().

**6.48.3.78 symmetrize\_cyclic()** [2/2]

```
ex GiNaC::ex::symmetrize_cyclic (
    const lst & l ) const
```

Symmetrize expression by cyclic permutation over a list of objects (symbols, indices).

References GiNaC::container< C >::begin(), GiNaC::container< C >::end(), and GiNaC::symmetrize\_cyclic().

#### 6.48.3.79 return\_type()

```
unsigned GiNaC::ex::return_type ( ) const [inline]
```

References bp.

Referenced by `GiNaC::ncmul::append_factors()`, `GiNaC::ncmul::count_factors()`, `GiNaC::power::eval()`, `GiNaC::ncmul::eval()`, `GiNaC::exmul()`, `GiNaC::matrix::mul_scalar()`, `GiNaC::integral::return_type()`, `GiNaC::relational::return_type()`, `GiNaC::power::return_type()`, and `GiNaC::indexed::return_type()`.

#### 6.48.3.80 return\_type\_tinfo()

```
return_type_t GiNaC::ex::return_type_tinfo ( ) const [inline]
```

References bp.

Referenced by `GiNaC::color_trace()`, `GiNaC::integral::return_type_tinfo()`, `GiNaC::relational::return_type_tinfo()`, `GiNaC::power::return_type_tinfo()`, and `GiNaC::indexed::return_type_tinfo()`.

#### 6.48.3.81 gethash()

```
unsigned GiNaC::ex::gethash ( ) const [inline]
```

References bp.

Referenced by `GiNaC::idx::calchash()`, `GiNaC::relational::calchash()`, `GiNaC::basic::calchash()`, and `GiNaC::function::calchash()`.

#### 6.48.3.82 construct\_from\_basic()

```
ptr< basic > GiNaC::ex::construct_from_basic (
    const basic & other ) [static], [private]
```

Helper function for the ex-from-basic constructor.

This is where `GiNaC`'s automatic evaluator and memory management are implemented.

See also

[ex::ex\(const basic &\)](#)

References bp, `GiNaC::basic::duplicate()`, `GiNaC::status_flags::dynallocated`, `GiNaC::basic::eval()`, `GiNaC::status_flags::evaluated`, `GiNaC::basic::flags`, `GiNaC::refcounted::get_refcount()`, and `GINAC_ASSERT`.



6.48.3.83 `construct_from_int()`

```
basic & GiNaC::ex::construct_from_int (
    int i ) [static], [private]
```

References `GiNaC::_num0_p`, `GiNaC::_num10_p`, `GiNaC::_num11_p`, `GiNaC::_num12_p`, `GiNaC::_num1_p`, `GiNaC::_num2_p`, `GiNaC::_num3_p`, `GiNaC::_num4_p`, `GiNaC::_num5_p`, `GiNaC::_num6_p`, `GiNaC::_num7_p`, `GiNaC::_num8_p`, `GiNaC::_num9_p`, `GiNaC::_num_10_p`, `GiNaC::_num_11_p`, `GiNaC::_num_12_p`, `GiNaC::_num_1_p`, `GiNaC::_num_2_p`, `GiNaC::_num_3_p`, `GiNaC::_num_4_p`, `GiNaC::_num_5_p`, `GiNaC::_num_6_p`, `GiNaC::_num_7_p`, `GiNaC::_num_8_p`, and `GiNaC::_num_9_p`.

Referenced by `construct_from_longlong()`.

6.48.3.84 `construct_from_uint()`

```
basic & GiNaC::ex::construct_from_uint (
    unsigned int i ) [static], [private]
```

References `GiNaC::_num0_p`, `GiNaC::_num10_p`, `GiNaC::_num11_p`, `GiNaC::_num12_p`, `GiNaC::_num1_p`, `GiNaC::_num2_p`, `GiNaC::_num3_p`, `GiNaC::_num4_p`, `GiNaC::_num5_p`, `GiNaC::_num6_p`, `GiNaC::_num7_p`, `GiNaC::_num8_p`, and `GiNaC::_num9_p`.

Referenced by `construct_from_ulonglong()`.

6.48.3.85 `construct_from_long()`

```
basic & GiNaC::ex::construct_from_long (
    long i ) [static], [private]
```

References `GiNaC::_num0_p`, `GiNaC::_num10_p`, `GiNaC::_num11_p`, `GiNaC::_num12_p`, `GiNaC::_num1_p`, `GiNaC::_num2_p`, `GiNaC::_num3_p`, `GiNaC::_num4_p`, `GiNaC::_num5_p`, `GiNaC::_num6_p`, `GiNaC::_num7_p`, `GiNaC::_num8_p`, `GiNaC::_num9_p`, `GiNaC::_num_10_p`, `GiNaC::_num_11_p`, `GiNaC::_num_12_p`, `GiNaC::_num_1_p`, `GiNaC::_num_2_p`, `GiNaC::_num_3_p`, `GiNaC::_num_4_p`, `GiNaC::_num_5_p`, `GiNaC::_num_6_p`, `GiNaC::_num_7_p`, `GiNaC::_num_8_p`, and `GiNaC::_num_9_p`.

6.48.3.86 `construct_from_ulong()`

```
basic & GiNaC::ex::construct_from_ulong (
    unsigned long i ) [static], [private]
```

References `GiNaC::_num0_p`, `GiNaC::_num10_p`, `GiNaC::_num11_p`, `GiNaC::_num12_p`, `GiNaC::_num1_p`, `GiNaC::_num2_p`, `GiNaC::_num3_p`, `GiNaC::_num4_p`, `GiNaC::_num5_p`, `GiNaC::_num6_p`, `GiNaC::_num7_p`, `GiNaC::_num8_p`, and `GiNaC::_num9_p`.

**6.48.3.87 construct\_from\_longlong()**

```
basic & GiNaC::ex::construct_from_longlong (
    long long i ) [static], [private]
```

References `construct_from_int()`.

**6.48.3.88 construct\_from\_ulonglong()**

```
basic & GiNaC::ex::construct_from_ulonglong (
    unsigned long long i ) [static], [private]
```

References `construct_from_uint()`.

**6.48.3.89 construct\_from\_double()**

```
basic & GiNaC::ex::construct_from_double (
    double d ) [static], [private]
```

**6.48.3.90 construct\_from\_string\_and\_lst()**

```
static ptr<basic> GiNaC::ex::construct_from_string_and_lst (
    const std::string & s,
    const ex & l ) [static], [private]
```

**6.48.3.91 makewriteable()**

```
void GiNaC::ex::makewriteable ( ) [private]
```

Make this ex writable (if more than one ex handle the same basic) by unlinking the object and creating an unshared copy of it.

References `bp`, `GiNaC::status_flags::dynallocated`, and `GINAC_ASSERT`.

Referenced by `let_op()`, and `operator[]()`.

## 6.48.3.92 share()

```
void GiNaC::ex::share (
    const ex & other ) const [private]
```

Share equal objects between expressions.

## See also

[ex::compare\(const ex &\)](#)

References [bp](#), and [GiNaC::status\\_flags::not\\_shareable](#).

Referenced by [compare\(\)](#), and [is\\_equal\(\)](#).

## 6.48.4 Friends And Related Function Documentation

## 6.48.4.1 archive\_node

```
friend class archive\_node [friend]
```

## 6.48.4.2 are\_ex\_trivially\_equal

```
bool are_ex_trivially_equal (
    const ex & e1,
    const ex & e2 ) [friend]
```

Compare two objects of class quickly without doing a deep tree traversal.

## Returns

"true" if they are equal "false" if equality cannot be established quickly (e1 and e2 may still be equal, in this case).

## 6.48.4.3 ex\_to

```
template<class T >
const T& ex_to (
    const ex & e ) [friend]
```

Return a reference to the basic-derived class T object embedded in an expression.

This is fast but unsafe: the result is undefined if the expression does not contain a T object at its top level. Hence, you should generally check the type of e first. Also, you shouldn't cache the returned reference because [GiNaC's](#) garbage collector may destroy the referenced object any time it's used in another expression.

## Parameters

<i>e</i>	expression
----------	------------

## Returns

reference to object of class T

## See also

[is\\_exactly\\_a<class T>\(\)](#)

## 6.48.4.4 is\_a

```
template<class T >
bool is_a (
    const ex & obj ) [friend]
```

Check if *ex* is a handle to a T, including base classes.

## 6.48.4.5 is\_exactly\_a

```
template<class T >
bool is_exactly_a (
    const ex & obj ) [friend]
```

Check if *ex* is a handle to a T, not including base classes.

## 6.48.5 Member Data Documentation

## 6.48.5.1 bp

```
ptr<basic> GiNaC::ex::bp [mutable], [private]
```

pointer to basic object managed by this

Referenced by [accept\(\)](#), [GiNaC::archive\\_node::archive\\_node\(\)](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [coeff\(\)](#), [collect\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [construct\\_from\\_basic\(\)](#), [dbgprint\(\)](#), [dbgprinttree\(\)](#), [degree\(\)](#), [denom\(\)](#), [diff\(\)](#), [eval\(\)](#), [eval\\_↵](#)  
[integ\(\)](#), [eval\\_ncmul\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [ex\(\)](#), [GiNaC::ex\\_to\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [gethash\(\)](#), [has\(\)](#), [GiNaC::archive\\_node::has\\_same\\_ex\\_as\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [integer\\_content\(\)](#), [GiNaC::is\\_a\(\)](#), [is\\_equal\(\)](#), [GiNaC::is\\_↵](#)  
[\\_exactly\\_a\(\)](#), [is\\_polynomial\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [lhs\(\)](#), [makewriteable\(\)](#), [map\(\)](#), [match\(\)](#), [max\\_coefficient\(\)](#), [nops\(\)](#), [normal\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), [op\(\)](#), [operator\[\]\(\)](#), [print\(\)](#), [GiNaC::archive\\_node::printraw\(\)](#), [real\\_part\(\)](#), [return\\_↵](#)  
[\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [rhs\(\)](#), [series\(\)](#), [share\(\)](#), [smod\(\)](#), [subs\(\)](#), [swap\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

The documentation for this class was generated from the following files:

- [ex.h](#)
- [ex.cpp](#)
- [indexed.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)
- [symmetry.cpp](#)

## 6.49 GiNaC::ex\_base\_is\_less Struct Reference

### Public Member Functions

- `bool operator()` (const `ex` &lh, const `ex` &rh) const

### 6.49.1 Member Function Documentation

#### 6.49.1.1 operator()

```
bool GiNaC::ex_base_is_less::operator() (
    const ex & lh,
    const ex & rh ) const    [inline]
```

References `GiNaC::ex::op()`.

The documentation for this struct was generated from the following file:

- [indexed.cpp](#)

## 6.50 GiNaC::ex\_is\_equal Struct Reference

```
#include <ex.h>
```

### Public Member Functions

- `bool operator()` (const `ex` &lh, const `ex` &rh) const

### 6.50.1 Member Function Documentation

#### 6.50.1.1 operator()

```
bool GiNaC::ex_is_equal::operator() (
    const ex & lh,
    const ex & rh ) const    [inline]
```

References `GiNaC::ex::is_equal()`.

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.51 GiNaC::ex\_is\_less Struct Reference

```
#include <ex.h>
```

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

#### 6.51.1 Member Function Documentation

##### 6.51.1.1 operator()

```
bool GiNaC::ex_is_less::operator() (
    const ex & lh,
    const ex & rh ) const [inline]
```

References [GiNaC::ex::compare\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.52 GiNaC::ex\_swap Struct Reference

```
#include <ex.h>
```

### Public Member Functions

- void [operator\(\)](#) ([ex](#) &lh, [ex](#) &rh) const

#### 6.52.1 Member Function Documentation

##### 6.52.1.1 operator()

```
void GiNaC::ex_swap::operator() (
    ex & lh,
    ex & rh ) const [inline]
```

References [GiNaC::ex::swap\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.53 GiNaC::expair Class Reference

A pair of expressions.

```
#include <expair.h>
```

### Public Member Functions

- [expair](#) ()
- [expair](#) (const [ex](#) &r, const [ex](#) &c)  
*Construct an expair from two ex.*
- bool [is\\_equal](#) (const [expair](#) &other) const  
*Member-wise check for canonical ordering equality.*
- bool [is\\_less](#) (const [expair](#) &other) const  
*Member-wise check for canonical ordering lessness.*
- int [compare](#) (const [expair](#) &other) const  
*Member-wise check for canonical ordering.*
- void [print](#) (std::ostream &os) const
- bool [is\\_canonical\\_numeric](#) () const  
*True if this is of the form (numeric,ex(1)).*
- void [swap](#) ([expair](#) &other)  
*Swap contents with other expair.*
- const [expair](#) [conjugate](#) () const

### Public Attributes

- [ex](#) [rest](#)  
*first member of pair, an arbitrary expression*
- [ex](#) [coeff](#)  
*second member of pair, must be numeric*

#### 6.53.1 Detailed Description

A pair of expressions.

This is similar to STL's `pair<>`. It is slightly extended since we need to account for methods like `.compare()`. Also, since this is meant for use by class `expairseq` it must satisfy the invariance that the member `coeff` must be of type `numeric`.

#### 6.53.2 Constructor & Destructor Documentation

##### 6.53.2.1 `expair()` [1/2]

```
GiNaC::expair::expair ( ) [inline]
```

Referenced by `conjugate()`.

### 6.53.2.2 `expair()` [2/2]

```
GiNaC::expair::expair (
    const ex & r,
    const ex & c ) [inline]
```

Construct an `expair` from two `ex`.

References `coeff`, and `GINAC_ASSERT`.

## 6.53.3 Member Function Documentation

### 6.53.3.1 `is_equal()`

```
bool GiNaC::expair::is_equal (
    const expair & other ) const [inline]
```

Member-wise check for canonical ordering equality.

References `coeff`, `GiNaC::ex::is_equal()`, and `rest`.

Referenced by `GiNaC::mul::expair_needs_further_processing()`.

### 6.53.3.2 `is_less()`

```
bool GiNaC::expair::is_less (
    const expair & other ) const [inline]
```

Member-wise check for canonical ordering lessness.

References `coeff`, `GiNaC::ex::compare()`, and `rest`.

Referenced by `GiNaC::expair_is_less::operator()()`.

### 6.53.3.3 `compare()`

```
int GiNaC::expair::compare (
    const expair & other ) const [inline]
```

Member-wise check for canonical ordering.

References `coeff`, `GiNaC::ex::compare()`, and `rest`.



## 6.53.3.4 print()

```
void GiNaC::expair::print (
    std::ostream & os ) const
```

References `c`, `coeff`, `GiNaC::ex::print()`, and `rest`.

## 6.53.3.5 is\_canonical\_numeric()

```
bool GiNaC::expair::is_canonical_numeric ( ) const [inline]
```

True if this is of the form `(numeric,ex(1))`.

References `coeff`, `GINAC_ASSERT`, `GiNaC::ex::is_equal()`, and `rest`.

## 6.53.3.6 swap()

```
void GiNaC::expair::swap (
    expair & other ) [inline]
```

Swap contents with other `expair`.

References `coeff`, `rest`, and `GiNaC::ex::swap()`.

Referenced by `GiNaC::mul::derivative()`, `GiNaC::expair_swap::operator()()`, and `GiNaC::swap()`.

## 6.53.3.7 conjugate()

```
const expair GiNaC::expair::conjugate ( ) const
```

References `GiNaC::are_ex_trivially_equal()`, `coeff`, `GiNaC::ex::conjugate()`, `expair()`, and `rest`.

## 6.53.4 Member Data Documentation

## 6.53.4.1 rest

```
ex GiNaC::expair::rest
```

first member of pair, an arbitrary expression

Referenced by `GiNaC::add::combine_pair_with_coeff_to_pair()`, `GiNaC::mul::combine_pair_with_coeff_to_pair()`, `compare()`, `conjugate()`, `is_canonical_numeric()`, `is_equal()`, `is_less()`, `GiNaC::expair_rest_is_less::operator()()`, `print()`, `GiNaC::add::recombine_pair_to_ex()`, `GiNaC::mul::recombine_pair_to_ex()`, and `swap()`.

#### 6.53.4.2 `coeff`

`ex` `GiNaC::expair::coeff`

second member of pair, must be numeric

Referenced by `GiNaC::mul::can_make_flat()`, `GiNaC::add::combine_pair_with_coeff_to_pair()`, `GiNaC::mul::combine_pair_with_coeff_to_pair()`, `compare()`, `conjugate()`, `expair()`, `GiNaC::power::expand_mul()`, `is_canonical_numeric()`, `is_equal()`, `is_less()`, `print()`, `GiNaC::add::recombine_pair_to_ex()`, `GiNaC::mul::recombine_pair_to_ex()`, and `swap()`.

The documentation for this class was generated from the following files:

- [expair.h](#)
- [expair.cpp](#)

### 6.54 `GiNaC::expair_is_less` Struct Reference

Function object for insertion into third argument of STL's `sort()` etc.

```
#include <expair.h>
```

#### Public Member Functions

- `bool operator() (const expair &lh, const expair &rh) const`

#### 6.54.1 Detailed Description

Function object for insertion into third argument of STL's `sort()` etc.

#### 6.54.2 Member Function Documentation

##### 6.54.2.1 `operator()()`

```
bool GiNaC::expair_is_less::operator() (
    const expair & lh,
    const expair & rh ) const [inline]
```

References `GiNaC::expair::is_less()`.

The documentation for this struct was generated from the following file:

- [expair.h](#)

## 6.55 GiNaC::expair\_rest\_is\_less Struct Reference

Function object not caring about the numerical coefficients for insertion into third argument of STL's sort().

```
#include <expair.h>
```

### Public Member Functions

- `bool operator() (const expair &lh, const expair &rh) const`

#### 6.55.1 Detailed Description

Function object not caring about the numerical coefficients for insertion into third argument of STL's sort().

Note that this does not define a strict weak ordering since for any symbol  $x$  we have neither  $3*x < 2*x$  or  $2*x < 3*x$ . Handle with care!

#### 6.55.2 Member Function Documentation

##### 6.55.2.1 operator()

```
bool GiNaC::expair_rest_is_less::operator() (
    const expair & lh,
    const expair & rh ) const [inline]
```

References `GiNaC::ex::compare()`, and `GiNaC::expair::rest`.

The documentation for this struct was generated from the following file:

- [expair.h](#)

## 6.56 GiNaC::expair\_swap Struct Reference

```
#include <expair.h>
```

### Public Member Functions

- `void operator() (expair &lh, expair &rh) const`

#### 6.56.1 Member Function Documentation

### 6.56.1.1 operator()

```
void GiNaC::expair_swap::operator() (
    expair & lh,
    expair & rh ) const [inline]
```

References GiNaC::expair::swap().

The documentation for this struct was generated from the following file:

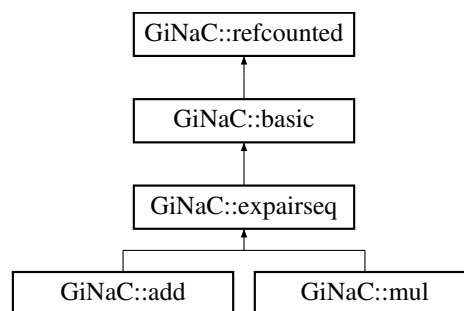
- [expair.h](#)

## 6.57 GiNaC::expairseq Class Reference

A sequence of class expair.

```
#include <expairseq.h>
```

Inheritance diagram for GiNaC::expairseq:



### Public Member Functions

- [expairseq](#) (const [ex](#) &lh, const [ex](#) &rh)
- [expairseq](#) (const [exvector](#) &v)
- [expairseq](#) (const [epvector](#) &v, const [ex](#) &oc, bool do\_index\_renaming=false)
- [expairseq](#) ([epvector](#) &&vp, const [ex](#) &oc, bool do\_index\_renaming=false)
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- size\_t [nops](#) () const override  
*Number of operands/members.*
- [ex](#) [op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex](#) [map](#) ([map\\_function](#) &f) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- [ex](#) [eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex](#) [to\\_rational](#) ([exmap](#) &repl) const override

Implementation of `ex::to_rational()` for `expairseqs`.

- `ex to_polynomial` (`exmap` &`repl`) const override

Implementation of `ex::to_polynomial()` for `expairseqs`.

- bool `match` (const `ex` &`pattern`, `exmap` &`repl_1st`) const override

Check whether the expression matches a given pattern.

- `ex subs` (const `exmap` &`m`, unsigned `options=0`) const override

Substitute a set of objects by arbitrary expressions.

- `ex conjugate` () const override
- void `archive` (`archive_node` &`n`) const override

Save (serialize) the object into archive node.

- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override

Load (deserialize) the object from an archive node.

## Protected Member Functions

- bool `is_equal_same_type` (const `basic` &`other`) const override

Returns true if two objects of same type are equal.

- unsigned `return_type` () const override
- unsigned `calchash` () const override

Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.

- `ex expand` (unsigned `options=0`) const override

Expand expression, i.e.

- virtual `ex thisexpairseq` (const `epvector` &`v`, const `ex` &`oc`, bool `do_index_renaming=false`) const
- virtual `ex thisexpairseq` (`epvector` &&`vp`, const `ex` &`oc`, bool `do_index_renaming=false`) const
- virtual void `printseq` (const `print_context` &`c`, char `delim`, unsigned `this_precedence`, unsigned `upper_precedence`) const
- virtual void `printpair` (const `print_context` &`c`, const `expair` &`p`, unsigned `upper_precedence`) const
- virtual `expair split_ex_to_pair` (const `ex` &`e`) const
- virtual `expair combine_ex_with_coeff_to_pair` (const `ex` &`e`, const `ex` &`c`) const
- virtual `expair combine_pair_with_coeff_to_pair` (const `expair` &`p`, const `ex` &`c`) const
- virtual `ex recombine_pair_to_ex` (const `expair` &`p`) const
- virtual bool `expair_needs_further_processing` (`epp` it)
- virtual `ex default_overall_coeff` () const
- virtual void `combine_overall_coeff` (const `ex` &`c`)
- virtual void `combine_overall_coeff` (const `ex` &`c1`, const `ex` &`c2`)
- virtual bool `can_make_flat` (const `expair` &`p`) const
- void `do_print` (const `print_context` &`c`, unsigned `level`) const
- void `do_print_tree` (const `print_tree` &`c`, unsigned `level`) const
- void `construct_from_2_ex` (const `ex` &`lh`, const `ex` &`rh`)
- void `construct_from_2_expairseq` (const `expairseq` &`s1`, const `expairseq` &`s2`)
- void `construct_from_expairseq_ex` (const `expairseq` &`s`, const `ex` &`e`)
- void `construct_from_exvector` (const `exvector` &`v`)
- void `construct_from_epvector` (const `epvector` &`v`, bool `do_index_renaming=false`)
- void `construct_from_epvector` (`epvector` &&`v`, bool `do_index_renaming=false`)
- void `make_flat` (const `exvector` &`v`)
- void `make_flat` (const `epvector` &`v`, bool `do_index_renaming=false`)
- void `canonicalize` ()
- void `combine_same_terms_sorted_seq` ()
- bool `is_canonical` () const
- `epvector expandchildren` (unsigned `options`) const
- `epvector evalchildren` () const
- `epvector subschildren` (const `exmap` &`m`, unsigned `options=0`) const

## Protected Attributes

- [epvector seq](#)
- [ex overall\\_coeff](#)

### 6.57.1 Detailed Description

A sequence of class `expair`.

This is used for time-critical classes like sums and products of terms since handling a list of `coeff` and `rest` is much faster than handling a list of products or powers, respectively. (Not incidentally, Maple does it the same way, maybe others too.) The semantics is (at least) twofold: one for addition and one for multiplication and several methods have to be overridden by derived classes to reflect the change in semantics. However, most functionality turns out to be shared between addition and multiplication, which is the reason why there is this base class.

### 6.57.2 Constructor & Destructor Documentation

#### 6.57.2.1 `expairseq()` [1/4]

```
GiNaC::expairseq::expairseq (
    const ex & lh,
    const ex & rh )
```

#### 6.57.2.2 `expairseq()` [2/4]

```
GiNaC::expairseq::expairseq (
    const exvector & v )
```

#### 6.57.2.3 `expairseq()` [3/4]

```
GiNaC::expairseq::expairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false )
```

#### 6.57.2.4 `expairseq()` [4/4]

```
GiNaC::expairseq::expairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false )
```

### 6.57.3 Member Function Documentation

#### 6.57.3.1 precedence()

```
unsigned GiNaC::expairseq::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

#### 6.57.3.2 info()

```
bool GiNaC::expairseq::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

#### 6.57.3.3 nops()

```
size_t GiNaC::expairseq::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

Referenced by [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::add::get\\_free\\_indices\(\)](#), and [GiNaC::mul::get\\_free\\_indices\(\)](#).

#### 6.57.3.4 op()

```
ex GiNaC::expairseq::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

Referenced by [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::add::do\\_print\\_python←\\_repr\(\)](#), [GiNaC::add::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), [GiNaC::add::series\(\)](#), and [GiNaC::mul←::series\(\)](#).

#### 6.57.3.5 map()

```
ex GiNaC::expairseq::map (
    map_function & f ) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

#### 6.57.3.6 eval()

```
ex GiNaC::expairseq::eval ( ) const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

#### 6.57.3.7 to\_rational()

```
ex GiNaC::expairseq::to_rational (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for expairseqs.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), [thisexpairseq\(\)](#), and [GiNaC::ex::to\\_rational\(\)](#).



## 6.57.3.8 to\_polynomial()

```
ex GiNaC::expairseq::to_polynomial (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for expairseqs.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex1](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), [thisexpairseq\(\)](#), and [GiNaC::ex::to\\_polynomial\(\)](#).

## 6.57.3.9 match()

```
bool GiNaC::expairseq::match (
    const ex & pattern,
    exmap & repl_lst ) const [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to `repl_lst`.

Reimplemented from [GiNaC::basic](#).

## 6.57.3.10 subs()

```
ex GiNaC::expairseq::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The `ex` returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

## 6.57.3.11 conjugate()

```
ex GiNaC::expairseq::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

#### 6.57.3.12 archive()

```
void GiNaC::expairseq::archive (
    archive_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

#### 6.57.3.13 read\_archive()

```
void GiNaC::expairseq::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

#### 6.57.3.14 is\_equal\_same\_type()

```
bool GiNaC::expairseq::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

**6.57.3.15** `return_type()`

```
unsigned GiNaC::expairseq::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

**6.57.3.16** `calchash()`

```
unsigned GiNaC::expairseq::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

**6.57.3.17** `expand()`

```
ex GiNaC::expairseq::expand (
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

**6.57.3.18** `thisexpairseq()` [1/2]

```
virtual ex GiNaC::expairseq::thisexpairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false ) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#), and [GiNaC::add](#).

Referenced by `to_polynomial()`, and `to_rational()`.

**6.57.3.19 thisexpairseq()** [2/2]

```
virtual ex GiNaC::expairseq::thisexpairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false ) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#), and [GiNaC::add](#).

**6.57.3.20 printseq()**

```
virtual void GiNaC::expairseq::printseq (
    const print_context & c,
    char delim,
    unsigned this_precedence,
    unsigned upper_precedence ) const [protected], [virtual]
```

**6.57.3.21 printpair()**

```
virtual void GiNaC::expairseq::printpair (
    const print_context & c,
    const expair & p,
    unsigned upper_precedence ) const [protected], [virtual]
```

**6.57.3.22 split\_ex\_to\_pair()**

```
virtual expair GiNaC::expairseq::split_ex_to_pair (
    const ex & e ) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#), and [GiNaC::add](#).

Referenced by [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**6.57.3.23 combine\_ex\_with\_coeff\_to\_pair()**

```
virtual expair GiNaC::expairseq::combine_ex_with_coeff_to_pair (
    const ex & e,
    const ex & c ) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#), and [GiNaC::add](#).

#### 6.57.3.24 combine\_pair\_with\_coeff\_to\_pair()

```
virtual expair GiNaC::expairseq::combine_pair_with_coeff_to_pair (  
    const expair & p,  
    const ex & c ) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#), and [GiNaC::add](#).

#### 6.57.3.25 recombine\_pair\_to\_ex()

```
virtual ex GiNaC::expairseq::recombine_pair_to_ex (  
    const expair & p ) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#), and [GiNaC::add](#).

Referenced by [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

#### 6.57.3.26 expair\_needs\_further\_processing()

```
virtual bool GiNaC::expairseq::expair_needs_further_processing (  
    exp it ) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

#### 6.57.3.27 default\_overall\_coeff()

```
virtual ex GiNaC::expairseq::default_overall_coeff ( ) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

Referenced by [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

#### 6.57.3.28 combine\_overall\_coeff() [1/2]

```
virtual void GiNaC::expairseq::combine_overall_coeff (  
    const ex & c ) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

**6.57.3.29 combine\_overall\_coeff()** [2/2]

```
virtual void GiNaC::expairseq::combine_overall_coeff (
    const ex & c1,
    const ex & c2 ) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

**6.57.3.30 can\_make\_flat()**

```
virtual bool GiNaC::expairseq::can_make_flat (
    const expair & p ) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

**6.57.3.31 do\_print()**

```
void GiNaC::expairseq::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

**6.57.3.32 do\_print\_tree()**

```
void GiNaC::expairseq::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

**6.57.3.33 construct\_from\_2\_ex()**

```
void GiNaC::expairseq::construct_from_2_ex (
    const ex & lh,
    const ex & rh ) [protected]
```

Referenced by [GiNaC::add::add\(\)](#), and [GiNaC::mul::mul\(\)](#).

**6.57.3.34 construct\_from\_2\_expairseq()**

```
void GiNaC::expairseq::construct_from_2_expairseq (
    const expairseq & s1,
    const expairseq & s2 ) [protected]
```

**6.57.3.35 construct\_from\_expairseq\_ex()**

```
void GiNaC::expairseq::construct_from_expairseq_ex (
    const expairseq & s,
    const ex & e ) [protected]
```

**6.57.3.36 construct\_from\_exvector()**

```
void GiNaC::expairseq::construct_from_exvector (
    const exvector & v ) [protected]
```

Referenced by GiNaC::add::add(), and GiNaC::mul::mul().

**6.57.3.37 construct\_from\_epvector()** [1/2]

```
void GiNaC::expairseq::construct_from_epvector (
    const epvector & v,
    bool do_index_renaming = false ) [protected]
```

Referenced by GiNaC::add::add(), and GiNaC::mul::mul().

**6.57.3.38 construct\_from\_epvector()** [2/2]

```
void GiNaC::expairseq::construct_from_epvector (
    epvector && v,
    bool do_index_renaming = false ) [protected]
```

**6.57.3.39 make\_flat()** [1/2]

```
void GiNaC::expairseq::make_flat (
    const exvector & v ) [protected]
```

**6.57.3.40 make\_flat()** [2/2]

```
void GiNaC::expairseq::make_flat (
    const epvector & v,
    bool do_index_renaming = false ) [protected]
```

#### 6.57.3.41 canonicalize()

```
void GiNaC::expairseq::canonicalize ( ) [protected]
```

#### 6.57.3.42 combine\_same\_terms\_sorted\_seq()

```
void GiNaC::expairseq::combine_same_terms_sorted_seq ( ) [protected]
```

#### 6.57.3.43 is\_canonical()

```
bool GiNaC::expairseq::is_canonical ( ) const [protected]
```

Referenced by `GiNaC::add::add()`, and `GiNaC::mul::mul()`.

#### 6.57.3.44 expandchildren()

```
epvector GiNaC::expairseq::expandchildren (
    unsigned options ) const [protected]
```

Referenced by `GiNaC::add::expand()`.

#### 6.57.3.45 evalchildren()

```
epvector GiNaC::expairseq::evalchildren ( ) const [protected]
```

Referenced by `GiNaC::add::eval()`.

#### 6.57.3.46 subschildren()

```
epvector GiNaC::expairseq::subschildren (
    const exmap & m,
    unsigned options = 0 ) const [protected]
```

### 6.57.4 Member Data Documentation



## 6.57.4.1 seq

`epvector` `GiNaC::expairseq::seq` [protected]

Referenced by `GiNaC::add::coeff()`, `GiNaC::add::degree()`, `GiNaC::add::derivative()`, `GiNaC::mul::do_print()`, `GiNaC::add::do_print_csrc()`, `GiNaC::mul::do_print_latex()`, `GiNaC::add::eval()`, `GiNaC::mul::eval()`, `GiNaC::add::eval_ncmul()`, `GiNaC::add::evalm()`, `GiNaC::power::expand()`, `GiNaC::mul::expand()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_add_2()`, `GiNaC::add::imag_part()`, `GiNaC::add::info()`, `GiNaC::add::integer_content()`, `GiNaC::mul::integer_content()`, `GiNaC::add::is_polynomial()`, `GiNaC::add::ldegree()`, `GiNaC::add::max_coefficient()`, `GiNaC::mul::max_coefficient()`, `GiNaC::add::normal()`, `GiNaC::mul::normal()`, `GiNaC::add::print_add()`, `GiNaC::add::real_part()`, `GiNaC::add::return_type()`, `GiNaC::add::return_type_tinfo()`, `GiNaC::add::series()`, `GiNaC::mul::series()`, `GiNaC::add::smod()`, `GiNaC::mul::smod()`, `to_polynomial()`, and `to_rational()`.

## 6.57.4.2 overall\_coeff

`ex` `GiNaC::expairseq::overall_coeff` [protected]

Referenced by `GiNaC::add::add()`, `GiNaC::add::coeff()`, `GiNaC::add::combine_ex_with_coeff_to_pair()`, `GiNaC::add::degree()`, `GiNaC::add::do_print_csrc()`, `GiNaC::mul::do_print_csrc()`, `GiNaC::add::eval()`, `GiNaC::mul::eval()`, `GiNaC::power::eval()`, `GiNaC::add::evalm()`, `GiNaC::add::expand()`, `GiNaC::power::expand()`, `GiNaC::mul::expand()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_add_2()`, `GiNaC::add::imag_part()`, `GiNaC::add::info()`, `GiNaC::add::integer_content()`, `GiNaC::mul::integer_content()`, `GiNaC::add::ldegree()`, `GiNaC::add::max_coefficient()`, `GiNaC::mul::max_coefficient()`, `GiNaC::mul::mul()`, `GiNaC::add::normal()`, `GiNaC::mul::normal()`, `GiNaC::add::print_add()`, `GiNaC::mul::print_overall_coeff()`, `GiNaC::add::real_part()`, `GiNaC::add::series()`, `GiNaC::mul::series()`, `GiNaC::add::smod()`, `GiNaC::mul::smod()`, `GiNaC::add::split_ex_to_pair()`, `to_polynomial()`, and `to_rational()`.

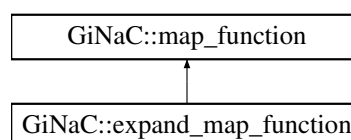
The documentation for this class was generated from the following files:

- [expairseq.h](#)
- [normal.cpp](#)

## 6.58 GiNaC::expand\_map\_function Struct Reference

Function object to be applied by [basic::expand\(\)](#).

Inheritance diagram for `GiNaC::expand_map_function`:



## Public Member Functions

- [expand\\_map\\_function](#) (unsigned o)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Attributes

- unsigned [options](#)

## Additional Inherited Members

### 6.58.1 Detailed Description

Function object to be applied by [basic::expand\(\)](#).

### 6.58.2 Constructor & Destructor Documentation

#### 6.58.2.1 `expand_map_function()`

```
GiNaC::expand_map_function::expand_map_function (
    unsigned o ) [inline]
```

### 6.58.3 Member Function Documentation

#### 6.58.3.1 `operator>()`

```
ex GiNaC::expand_map_function::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::ex::expand\(\)](#), and [options](#).

### 6.58.4 Member Data Documentation

#### 6.58.4.1 `options`

```
unsigned GiNaC::expand_map_function::options
```

The documentation for this struct was generated from the following file:

- [basic.cpp](#)

## 6.59 GiNaC::expand\_options Class Reference

Flags to control the behavior of [expand\(\)](#).

```
#include <flags.h>
```

### Public Types

- enum { [expand\\_indexed](#) = 0x0001, [expand\\_function\\_args](#) = 0x0002, [expand\\_rename\\_idx](#) = 0x0004, [expand\\_transcendental](#) = 0x0008 }

### 6.59.1 Detailed Description

Flags to control the behavior of [expand\(\)](#).

### 6.59.2 Member Enumeration Documentation

#### 6.59.2.1 anonymous enum

anonymous enum

#### Enumerator

<a href="#">expand_indexed</a>	expands (a+b).i to a.i+b.i
<a href="#">expand_function_args</a>	expands the arguments of functions
<a href="#">expand_rename_idx</a>	used internally by <a href="#">mul::expand()</a>
<a href="#">expand_transcendental</a>	expands transcendental functions like log and exp

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.60 GiNaC::factor\_options Class Reference

Flags to control the polynomial factorization.

```
#include <flags.h>
```

### Public Types

- enum { [polynomial](#) = 0x0000, [all](#) = 0x0001 }

### 6.60.1 Detailed Description

Flags to control the polynomial factorization.

### 6.60.2 Member Enumeration Documentation

#### 6.60.2.1 anonymous enum

anonymous enum

##### Enumerator

polynomial	factor only expressions that are polynomials
all	factor all polynomial subexpressions

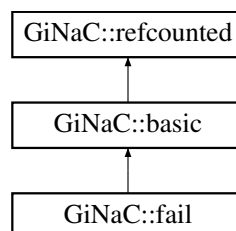
The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.61 GiNaC::fail Class Reference

```
#include <fail.h>
```

Inheritance diagram for GiNaC::fail:



### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

### Additional Inherited Members

#### 6.61.1 Member Function Documentation

6.61.1.1 `return_type()`

```
unsigned GiNaC::fail::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative\\_composite](#).

6.61.1.2 `do_print()`

```
void GiNaC::fail::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following file:

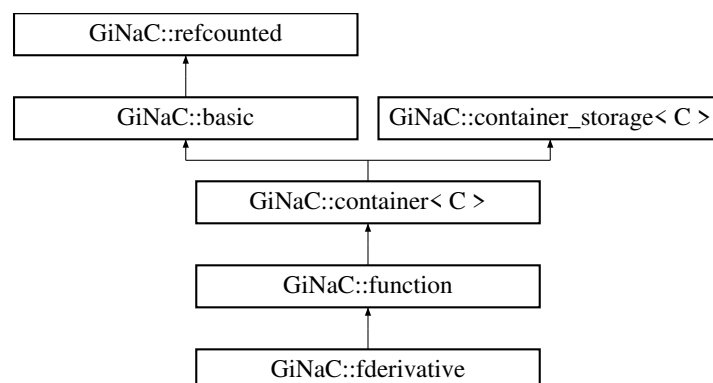
- [fail.h](#)

## 6.62 GiNaC::fderivative Class Reference

This class represents the (abstract) derivative of a symbolic function.

```
#include <fderivative.h>
```

Inheritance diagram for `GiNaC::fderivative`:



## Public Member Functions

- **fderivative** (unsigned ser, unsigned param, const **exvector** &args)  
*Construct derivative with respect to one parameter.*
- **fderivative** (unsigned ser, const **paramset** &params, const **exvector** &args)  
*Construct derivative with respect to multiple parameters.*
- **fderivative** (unsigned ser, const **paramset** &params, **exvector** &&v)
- void **print** (const **print\_context** &c, unsigned level=0) const override  
*Output to stream.*
- **ex eval** () const override  
*Perform automatic non-interruptive term rewriting rules.*
- **ex series** (const **relational** &r, int **order**, unsigned **options**=0) const override  
*The series expansion of derivatives falls back to Taylor expansion.*
- **ex thiscontainer** (const **exvector** &v) const override
- **ex thiscontainer** (**exvector** &&v) const override
- void **archive** (**archive\_node** &n) const override  
*Archive the object.*
- void **read\_archive** (const **archive\_node** &n, **lst** &syms) override  
*Load (deserialize) the object from an archive node.*
- const **paramset** & **derivatives** () const  
*Expose this object's derivative structure.*

## Protected Member Functions

- **ex derivative** (const **symbol** &s) const override  
*Implementation of `ex::diff()` for derivatives.*
- bool **is\_equal\_same\_type** (const **basic** &other) const override  
*Returns true if two objects of same type are equal.*
- bool **match\_same\_type** (const **basic** &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- void **do\_print** (const **print\_context** &c, unsigned level) const
- void **do\_print\_latex** (const **print\_context** &c, unsigned level) const
- void **do\_print\_csrc** (const **print\_csrc** &c, unsigned level) const
- void **do\_print\_tree** (const **print\_tree** &c, unsigned level) const

## Protected Attributes

- **paramset** **parameter\_set**  
*Set of parameter numbers with respect to which to take the derivative.*

## Additional Inherited Members

### 6.62.1 Detailed Description

This class represents the (abstract) derivative of a symbolic function.

It is used to represent the derivatives of functions that do not have a derivative or series expansion procedure defined.

## 6.62.2 Constructor & Destructor Documentation

### 6.62.2.1 fderivative() [1/3]

```
GiNaC::fderivative::fderivative (
    unsigned ser,
    unsigned param,
    const exvector & args )
```

Construct derivative with respect to one parameter.

#### Parameters

<i>ser</i>	Serial number of function
<i>param</i>	Number of parameter with respect to which to take the derivative
<i>args</i>	Arguments of derivative function

References `parameter_set`.

Referenced by `derivative()`, and `thiscontainer()`.

### 6.62.2.2 fderivative() [2/3]

```
GiNaC::fderivative::fderivative (
    unsigned ser,
    const paramset & params,
    const exvector & args )
```

Construct derivative with respect to multiple parameters.

#### Parameters

<i>ser</i>	Serial number of function
<i>params</i>	Set of numbers of parameters with respect to which to take the derivative
<i>args</i>	Arguments of derivative function

### 6.62.2.3 fderivative() [3/3]

```
GiNaC::fderivative::fderivative (
    unsigned ser,
    const paramset & params,
    exvector && v )
```

### 6.62.3 Member Function Documentation

#### 6.62.3.1 `print()`

```
void GiNaC::fderivative::print (
    const print\_context & c,
    unsigned level = 0 ) const [override], [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of `*this` and the dynamic type of the supplied print context.

Parameters

<code>c</code>	print context object that describes the output formatting
<code>level</code>	value that is used to identify the precedence or indentation level for placing parentheses and formatting

Reimplemented from [GiNaC::basic](#).

References `c`, and `GiNaC::basic::print()`.

#### 6.62.3.2 `eval()`

```
ex GiNaC::fderivative::eval ( ) const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References `GiNaC::basic::hold()`, `parameter_set`, `GiNaC::function::pderivative()`, `GiNaC::function::registered_↔functions()`, `GiNaC::container_storage< C >::seq`, and `GiNaC::function::serial`.

#### 6.62.3.3 `series()`

```
ex GiNaC::fderivative::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

The series expansion of derivatives falls back to Taylor expansion.

See also

[basic::series](#)

Reimplemented from [GiNaC::basic](#).

References `options`, `order`, `r`, and `GiNaC::basic::series()`.



#### 6.62.3.4 thiscontainer() [1/2]

```
ex GiNaC::fderivative::thiscontainer (
    const exvector & v ) const [override]
```

References [fderivative\(\)](#), [parameter\\_set](#), and [GiNaC::function::serial](#).

#### 6.62.3.5 thiscontainer() [2/2]

```
ex GiNaC::fderivative::thiscontainer (
    exvector && v ) const [override]
```

References [fderivative\(\)](#), [parameter\\_set](#), and [GiNaC::function::serial](#).

#### 6.62.3.6 archive()

```
void GiNaC::fderivative::archive (
    archive\_node & n ) const [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::container< C >::end\(\)](#), [n](#), and [parameter\\_set](#).

#### 6.62.3.7 read\_archive()

```
void GiNaC::fderivative::read_archive (
    const archive\_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::container< C >](#).

References [n](#), and [parameter\\_set](#).

### 6.62.3.8 derivative()

```
ex GiNaC::fderivative::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for derivatives.

It applies the chain rule.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [fderivative\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [parameter\\_set](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::function::serial](#).

### 6.62.3.9 is\_equal\_same\_type()

```
bool GiNaC::fderivative::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), and [parameter\\_set](#).

### 6.62.3.10 match\_same\_type()

```
bool GiNaC::fderivative::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [parameter\\_set](#).

## 6.62.3.11 derivatives()

```
const paramset & GiNaC::fderivative::derivatives ( ) const
```

Expose this object's derivative structure.

Parameter numbers occurring more than once stand for repeated differentiation with respect to that parameter. If a symbolic function  $f(x,y)$  is differentiated with respect to  $x$ , this method will return  $\{0\}$ . If  $f(x,y)$  is differentiated twice with respect to  $y$ , it will return  $\{1,1\}$ . (This corresponds to the way this object is printed.)

## Returns

multiset of function's parameter numbers that are abstractly differentiated.

References parameter\_set.

## 6.62.3.12 do\_print()

```
void GiNaC::fderivative::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References `c`, `GiNaC::container< C >::end()`, `parameter_set`, `GiNaC::container< C >::precedence()`, `GiNaC::function::precedence()`, `GiNaC::container< C >::printseq()`, `GiNaC::function::registered_functions()`, and `GiNaC::function::serial`.

## 6.62.3.13 do\_print\_latex()

```
void GiNaC::fderivative::do_print_latex (
    const print_context & c,
    unsigned level ) const [protected]
```

References `c`, `GiNaC::container< C >::end()`, `order`, `parameter_set`, `GiNaC::container< C >::precedence()`, `GiNaC::function::precedence()`, `GiNaC::container< C >::printseq()`, `GiNaC::function::registered_functions()`, and `GiNaC::function::serial`.

## 6.62.3.14 do\_print\_csrc()

```
void GiNaC::fderivative::do_print_csrc (
    const print_csrc & c,
    unsigned level ) const [protected]
```

References `c`, `GiNaC::container< C >::end()`, `parameter_set`, `GiNaC::container< C >::precedence()`, `GiNaC::function::precedence()`, `GiNaC::container< C >::printseq()`, `GiNaC::function::registered_functions()`, and `GiNaC::function::serial`.

### 6.62.3.15 do\_print\_tree()

```
void GiNaC::fderivative::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::container< C >::nops\(\)](#), [parameter\\_set](#), [GiNaC::function::registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::function::serial](#).

## 6.62.4 Member Data Documentation

### 6.62.4.1 parameter\_set

```
paramset GiNaC::fderivative::parameter_set [protected]
```

Set of parameter numbers with respect to which to take the derivative.

Referenced by [archive\(\)](#), [derivative\(\)](#), [derivatives\(\)](#), [do\\_print\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_tree\(\)](#), [eval\(\)](#), [fderivative\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), and [thiscontainer\(\)](#).

The documentation for this class was generated from the following files:

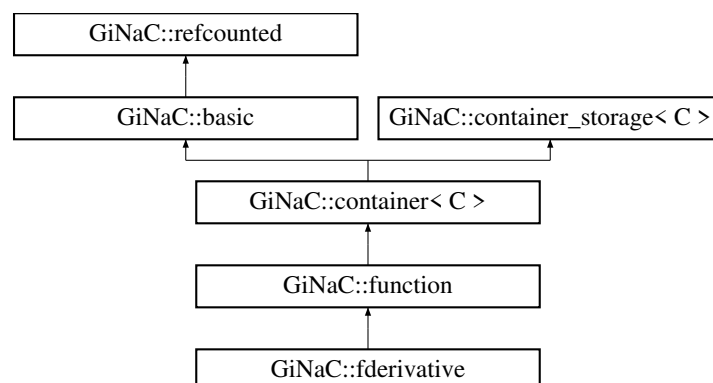
- [fderivative.h](#)
- [fderivative.cpp](#)

## 6.63 GiNaC::function Class Reference

The class function is used to implement builtin functions like sin, cos...

```
#include <function.h>
```

Inheritance diagram for GiNaC::function:



## Public Member Functions

- [function](#) (unsigned ser)
- [function](#) (unsigned ser, const [ex](#) &param1)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12, const [ex](#) &param13)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12, const [ex](#) &param13, const [ex](#) &param14)
- [function](#) (unsigned ser, const [exprseq](#) &es)
- [function](#) (unsigned ser, const [exvector](#) &v)
- [function](#) (unsigned ser, [exvector](#) &&v)
- void [print](#) (const [print\\_context](#) &c, unsigned level=0) const override  
*Output to stream.*
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- [ex expand](#) (unsigned [options](#)=0) const override  
*Expand expression, i.e.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex eval\\_ncmul](#) (const [exvector](#) &v) const override  
*This method is defined to be in line with behavior of [function::return\\_type\(\)](#)*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*Implementation of [ex::series](#) for functions.*
- [ex thiscontainer](#) (const [exvector](#) &v) const override
- [ex thiscontainer](#) ([exvector](#) &&v) const override
- [ex conjugate](#) () const override  
*Implementation of [ex::conjugate](#) for functions.*

- `ex real_part ()` const override  
*Implementation of `ex::real_part` for functions.*
- `ex imag_part ()` const override  
*Implementation of `ex::imag_part` for functions.*
- `void archive (archive_node &n)` const override  
*Archive the object.*
- `void read_archive (const archive_node &n, lst &syms)` override  
*Construct object from `archive_node`.*
- `bool info (unsigned inf)` const override  
*Implementation of `ex::info` for functions.*
- `ex power (const ex &exp)` const
- `unsigned get_serial ()` const
- `std::string get_name ()` const  
*Return the print name of the function.*

### Static Public Member Functions

- static unsigned `register_new (function_options const &opt)`
- static unsigned `find_function (const std::string &name, unsigned nparams)`  
*Find serial number of function by name and number of parameters.*
- static `std::vector< function_options > get_registered_functions ()`

### Static Public Attributes

- static unsigned `current_serial = 0`  
*This can be used as a hook for external applications.*

### Protected Member Functions

- `ex derivative (const symbol &s)` const override  
*Implementation of `ex::diff()` for functions.*
- `bool is_equal_same_type (const basic &other)` const override  
*Returns true if two objects of same type are equal.*
- `bool match_same_type (const basic &other)` const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `unsigned return_type ()` const override
- `return_type_t return_type_tinfo ()` const override
- `ex pderivative (unsigned diff_param)` const
- `ex expl_derivative (const symbol &s)` const
- `bool lookup_remember_table (ex &result)` const
- `void store_remember_table (ex const &result)` const

### Static Protected Member Functions

- static `std::vector< function_options > & registered_functions ()`

### Protected Attributes

- unsigned `serial`

## Friends

- class [remember\\_table\\_entry](#)

## Additional Inherited Members

### 6.63.1 Detailed Description

The class function is used to implement builtin functions like sin, cos...  
and user defined functions

### 6.63.2 Constructor & Destructor Documentation

#### 6.63.2.1 function() [1/18]

```
GiNaC::function::function (  
    unsigned ser )
```

#### 6.63.2.2 function() [2/18]

```
GiNaC::function::function (  
    unsigned ser,  
    const ex & param1 )
```

#### 6.63.2.3 function() [3/18]

```
GiNaC::function::function (  
    unsigned ser,  
    const ex & param1,  
    const ex & param2 )
```

#### 6.63.2.4 function() [4/18]

```
GiNaC::function::function (  
    unsigned ser,  
    const ex & param1,  
    const ex & param2,  
    const ex & param3 )
```

#### 6.63.2.5 function() [5/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4 )
```

#### 6.63.2.6 function() [6/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5 )
```

#### 6.63.2.7 function() [7/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6 )
```

#### 6.63.2.8 function() [8/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7 )
```



### 6.63.2.9 function() [9/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8 )
```

### 6.63.2.10 function() [10/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9 )
```

### 6.63.2.11 function() [11/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10 )
```

**6.63.2.12** `function()` [12/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11 )
```

**6.63.2.13** `function()` [13/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11,
    const ex & param12 )
```

**6.63.2.14** `function()` [14/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11,
    const ex & param12,
    const ex & param13 )
```

**6.63.2.15** `function()` [15/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11,
    const ex & param12,
    const ex & param13,
    const ex & param14 )
```

**6.63.2.16** `function()` [16/18]

```
GiNaC::function::function (
    unsigned ser,
    const exprseq & es )
```

References `GiNaC::basic::clearflag()`, and `GiNaC::status_flags::evaluated`.

**6.63.2.17** `function()` [17/18]

```
GiNaC::function::function (
    unsigned ser,
    const exvector & v )
```

**6.63.2.18** `function()` [18/18]

```
GiNaC::function::function (
    unsigned ser,
    exvector && v )
```

**6.63.3 Member Function Documentation****6.63.3.1** `print()`

```
void GiNaC::function::print (
    const print_context & c,
    unsigned level = 0 ) const [override], [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of `*this` and the dynamic type of the supplied print context.

## Parameters

<i>c</i>	print context object that describes the output formatting
<i>level</i>	value that is used to identify the precedence or indentation level for placing parentheses and formatting

Reimplemented from [GiNaC::basic](#).

References `c`, `current_serial`, `GiNaC::basic::flags`, `GiNaC::class_info< OPT >::get_parent()`, `GINAC_ASSERT`, `GiNaC::basic::hashvalue`, `GiNaC::function_options::name`, `GiNaC::container< C >::nops()`, `GiNaC::function_↵  
options::nparams`, `GiNaC::class_info< OPT >::options`, `GiNaC::container< C >::precedence()`, `precedence()`, `GiNaC::function_options::print_dispatch_table`, `GiNaC::function_options::print_use_exvector_args`, `GiNaC_↵  
::container< C >::printseq()`, `registered_functions()`, `GiNaC::container_storage< C >::seq`, `serial`, and `GiNa_↵  
C::function_options::TeX_name`.

6.63.3.2 `precedence()`

```
unsigned GiNaC::function::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< C >](#).

Referenced by `GiNaC::fderivative::do_print()`, `GiNaC::fderivative::do_print_csrc()`, `GiNaC::fderivative::do_print_↵  
latex()`, and `print()`.

6.63.3.3 `expand()`

```
ex GiNaC::function::expand (
    unsigned options = 0 ) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References `current_serial`, `GiNaC::expand()`, `GiNaC::function_options::expand_f`, `GiNaC::expand_options_↵  
::expand_function_args`, `GiNaC::function_options::expand_use_exvector_args`, `GiNaC::status_flags::expanded`, `GINAC_ASSERT`, `GiNaC::function_options::nparams`, `options`, `registered_functions()`, `GiNaC::container_storage< C >::seq`, `serial`, and `GiNaC::basic::setflag()`.

## 6.63.3.4 eval()

```
ex GiNaC::function::eval ( ) const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::canonicalize\(\)](#), [current\\_serial](#), [GiNaC::function\\_options::eval\\_f](#), [GiNaC::function\\_options::eval\\_use\\_exvector\\_args](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::ex](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [lookup\\_remember\\_table\(\)](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), [store\\_remember\\_table\(\)](#), [GiNaC::function\\_options::syntree](#), [thiscontainer\(\)](#), and [GiNaC::function\\_options::use\\_remember](#).

## 6.63.3.5 evalf()

```
ex GiNaC::function::evalf ( ) const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GiNaC::function\\_options::evalf\\_f](#), [GiNaC::function\\_options::evalf\\_params\\_first](#), [GiNaC::function\\_options::evalf\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

Referenced by [GiNaC::iterated\\_integral2\\_eval\(\)](#), and [GiNaC::iterated\\_integral3\\_eval\(\)](#).

## 6.63.3.6 eval\_ncmul()

```
ex GiNaC::function::eval_ncmul (
    const exvector & v ) const [override], [virtual]
```

This method is defined to be in line with behavior of [function::return\\_type\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container\\_storage< C >::seq](#).

## 6.63.3.7 calchash()

```
unsigned GiNaC::function::calchash ( ) const [override], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::ex::gethash\(\)](#), [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::make\\_hash\\_seed\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::rotate\\_left\(\)](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

### 6.63.3.8 series()

```
ex GiNaC::function::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of [ex::series](#) for functions.

@see [ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [options](#), [order](#), [r](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), [GiNaC::basic::series\(\)](#), [GiNaC::function\\_options::series\\_f](#), and [GiNaC::function\\_options::series\\_use\\_exvector\\_args](#).

### 6.63.3.9 thiscontainer() [1/2]

```
ex GiNaC::function::thiscontainer (
    const exvector & v ) const [override]
```

References [serial](#).

Referenced by [eval\(\)](#).

### 6.63.3.10 thiscontainer() [2/2]

```
ex GiNaC::function::thiscontainer (
    exvector && v ) const [override]
```

References [serial](#).

### 6.63.3.11 conjugate()

```
ex GiNaC::function::conjugate ( ) const [override], [virtual]
```

Implementation of [ex::conjugate](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::function\\_options::conjugate\\_f](#), [GiNaC::function\\_options::conjugate\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

6.63.3.12 `real_part()`

```
ex GiNaC::function::real_part ( ) const [override], [virtual]
```

Implementation of [ex::real\\_part](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References `GINAC_ASSERT`, `GiNaC::function_options::nparams`, `GiNaC::basic::real_part()`, `GiNaC::function_options::real_part_f`, `GiNaC::function_options::real_part_use_exvector_args`, `registered_functions()`, `GiNaC::container_storage< C >::seq`, and `serial`.

6.63.3.13 `imag_part()`

```
ex GiNaC::function::imag_part ( ) const [override], [virtual]
```

Implementation of [ex::imag\\_part](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References `GINAC_ASSERT`, `GiNaC::basic::imag_part()`, `GiNaC::function_options::imag_part_f`, `GiNaC::function_options::imag_part_use_exvector_args`, `GiNaC::function_options::nparams`, `registered_functions()`, `GiNaC::container_storage< C >::seq`, and `serial`.

6.63.3.14 `archive()`

```
void GiNaC::function::archive (
    archive_node & n ) const [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::container< C >](#).

References `GINAC_ASSERT`, `n`, `registered_functions()`, and `serial`.

6.63.3.15 `read_archive()`

```
void GiNaC::function::read_archive (
    const archive_node & n,
    lst & sym_lst ) [override], [virtual]
```

Construct object from [archive\\_node](#).

Reimplemented from [GiNaC::container< C >](#).

References `n`, `registered_functions()`, `GiNaC::container_storage< C >::seq`, and `serial`.

#### 6.63.3.16 info()

```
bool GiNaC::function::info (
    unsigned inf ) const [override], [virtual]
```

Implementation of [ex::info](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References GINAC\_ASSERT, GiNaC::basic::info(), GiNaC::function\_options::info\_f, GiNaC::function\_options::info\_use\_exvector\_args, GiNaC::function\_options::nparams, registered\_functions(), GiNaC::container\_storage< C >::seq, and serial.

#### 6.63.3.17 derivative()

```
ex GiNaC::function::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for functions.

It applies the chain rule, except for the Order term function. @see [ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References expl\_derivative(), GiNaC::ex::is\_zero(), pderivative(), and GiNaC::container\_storage< C >::seq.

#### 6.63.3.18 is\_equal\_same\_type()

```
bool GiNaC::function::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::container< C >](#).

References GINAC\_ASSERT, GiNaC::container< C >::is\_equal\_same\_type(), and serial.



6.63.3.19 `match_same_type()`

```
bool GiNaC::function::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References `GINAC_ASSERT`, and `serial`.

6.63.3.20 `return_type()`

```
unsigned GiNaC::function::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References `GiNaC::return_types::commutative`, `GINAC_ASSERT`, `registered_functions()`, `GiNaC::function_options::return_type`, `GiNaC::container_storage< C >::seq`, `serial`, and `GiNaC::function_options::use_return_type`.

6.63.3.21 `return_type_tinfo()`

```
return\_type\_t GiNaC::function::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References `GINAC_ASSERT`, `registered_functions()`, `GiNaC::function_options::return_type_tinfo`, `GiNaC::container_storage< C >::seq`, `serial`, and `GiNaC::function_options::use_return_type`.

6.63.3.22 `pderivative()`

```
ex GiNaC::function::pderivative (
    unsigned diff_param ) const [protected]
```

References `GiNaC::function_options::derivative_f`, `GiNaC::function_options::derivative_use_exvector_args`, `GINAC_ASSERT`, and `GiNaC::function_options::nparams`.

Referenced by `derivative()`, and `GiNaC::fderivative::eval()`.

#### 6.63.3.23 expl\_derivative()

```
ex GiNaC::function::expl_derivative (
    const symbol & s ) const [protected]
```

References GiNaC::function\_options::expl\_derivative\_f, GiNaC::function\_options::expl\_derivative\_use\_exvector\_↵\_args, GINAC\_ASSERT, and GiNaC::function\_options::nparams.

Referenced by derivative().

#### 6.63.3.24 registered\_functions()

```
std::vector< function_options > & GiNaC::function::registered_functions ( ) [static], [protected]
```

Referenced by archive(), conjugate(), GiNaC::fderivative::do\_print(), GiNaC::fderivative::do\_print\_csrc(), GiNaC::↵fderivative::do\_print\_latex(), GiNaC::fderivative::do\_print\_tree(), GiNaC::fderivative::eval(), eval(), evalf(), ex-↵pand(), find\_function(), get\_name(), get\_registered\_functions(), imag\_part(), info(), print(), read\_archive(), real\_↵part(), register\_new(), return\_type(), return\_type\_tinfo(), and series().

#### 6.63.3.25 lookup\_remember\_table()

```
bool GiNaC::function::lookup_remember_table (
    ex & result ) const [protected]
```

References GiNaC::remember\_table::remember\_tables(), and serial.

Referenced by eval().

#### 6.63.3.26 store\_remember\_table()

```
void GiNaC::function::store_remember_table (
    ex const & result ) const [protected]
```

References GiNaC::remember\_table::remember\_tables(), and serial.

Referenced by eval().

#### 6.63.3.27 power()

```
ex GiNaC::function::power (
    const ex & exp ) const
```

References GiNaC::status\_flags::evaluated, GINAC\_ASSERT, GiNaC::function\_options::nparams, GiNaC::↵function\_options::power\_f, and GiNaC::function\_options::power\_use\_exvector\_args.

#### 6.63.3.28 register\_new()

```
unsigned GiNaC::function::register_new (
    function_options const & opt ) [static]
```

References `GiNaC::function_options::functions_with_same_name`, `GiNaC::function_options::name`, `registered_↵  
_functions()`, `GiNaC::function_options::remember_assoc_size`, `GiNaC::function_options::remember_size`, `GiNaC::↵  
function_options::remember_strategy`, `GiNaC::remember_table::remember_tables()`, and `GiNaC::function_↵  
options::use_remember`.

#### 6.63.3.29 find\_function()

```
unsigned GiNaC::function::find_function (
    const std::string & name,
    unsigned nparams ) [static]
```

Find serial number of function by name and number of parameters.

Throws exception if function was not found.

References `registered_functions()`, and `serial`.

#### 6.63.3.30 get\_registered\_functions()

```
static std::vector<function_options> GiNaC::function::get_registered_functions ( ) [inline],  
[static]
```

References `registered_functions()`.

#### 6.63.3.31 get\_serial()

```
unsigned GiNaC::function::get_serial ( ) const [inline]
```

References `serial`.

#### 6.63.3.32 get\_name()

```
std::string GiNaC::function::get_name ( ) const
```

Return the print name of the function.

References `GINAC_ASSERT`, `registered_functions()`, and `serial`.

## 6.63.4 Friends And Related Function Documentation

### 6.63.4.1 remember\_table\_entry

```
friend class remember_table_entry [friend]
```

## 6.63.5 Member Data Documentation

### 6.63.5.1 current\_serial

```
unsigned GiNaC::function::current_serial = 0 [static]
```

This can be used as a hook for external applications.

Referenced by `eval()`, `evalf()`, `expand()`, `print()`, and `series()`.

### 6.63.5.2 serial

```
unsigned GiNaC::function::serial [protected]
```

Referenced by `archive()`, `calchash()`, `conjugate()`, `GiNaC::fderivative::derivative()`, `GiNaC::fderivative::do_print()`, `GiNaC::fderivative::do_print_csrc()`, `GiNaC::fderivative::do_print_latex()`, `GiNaC::fderivative::do_print_tree()`, `GiNaC::fderivative::eval()`, `eval()`, `evalf()`, `expand()`, `find_function()`, `get_name()`, `get_serial()`, `imag_part()`, `info()`, `is_equal_same_type()`, `lookup_remember_table()`, `match_same_type()`, `print()`, `read_archive()`, `real_part()`, `return_type()`, `return_type_tinfo()`, `series()`, `store_remember_table()`, `GiNaC::fderivative::thiscontainer()`, and `thiscontainer()`.

The documentation for this class was generated from the following files:

- [function.h](#)
- [function.cpp](#)

## 6.64 GiNaC::function\_options Class Reference

```
#include <function.h>
```

## Public Member Functions

- [function\\_options](#) ()
- [function\\_options](#) (std::string const &n, std::string const &tn=std::string())
- [function\\_options](#) (std::string const &n, unsigned np)
- [~function\\_options](#) ()
- [void initialize](#) ()
- [function\\_options](#) & [dummy](#) ()
- [function\\_options](#) & [set\\_name](#) (std::string const &n, std::string const &tn=std::string())
- [function\\_options](#) & [latex\\_name](#) (std::string const &tn)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_1](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_2](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_3](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_4](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_5](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_6](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_7](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_8](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_9](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_10](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_11](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_12](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_13](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_14](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_1](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_2](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_3](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_4](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_5](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_6](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_7](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_8](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_9](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_10](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_11](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_12](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_13](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_14](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_1](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_2](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_3](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_4](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_5](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_6](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_7](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_8](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_9](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_10](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_11](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_12](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_13](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_14](#) e)
- [function\\_options](#) & [real\\_part\\_func](#) ([real\\_part\\_funcp\\_1](#) e)
- [function\\_options](#) & [real\\_part\\_func](#) ([real\\_part\\_funcp\\_2](#) e)
- [function\\_options](#) & [real\\_part\\_func](#) ([real\\_part\\_funcp\\_3](#) e)



- Generated by Doxygen

- `function_options` & `evalf_func` (`evalf_funcp_exvector` e)
- `function_options` & `conjugate_func` (`conjugate_funcp_exvector` e)
- `function_options` & `real_part_func` (`real_part_funcp_exvector` e)
- `function_options` & `imag_part_func` (`imag_part_funcp_exvector` e)
- `function_options` & `expand_func` (`expand_funcp_exvector` e)
- `function_options` & `derivative_func` (`derivative_funcp_exvector` e)
- `function_options` & `expl_derivative_func` (`expl_derivative_funcp_exvector` e)
- `function_options` & `power_func` (`power_funcp_exvector` e)
- `function_options` & `series_func` (`series_funcp_exvector` e)
- `function_options` & `info_func` (`info_funcp_exvector` e)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_1` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_2` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_3` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_4` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_5` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_6` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_7` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_8` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_9` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_10` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_11` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_12` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_13` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_14` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_exvector` p)
- `function_options` & `set_return_type` (unsigned rt, const `return_type_t` \*rtt=nullptr)
- `function_options` & `do_not_evalf_params` ()
- `function_options` & `remember` (unsigned size, unsigned assoc\_size=0, unsigned strategy=`remember_strategies::delete_never`)
- `function_options` & `overloaded` (unsigned o)
- `function_options` & `set_symmetry` (const `symmetry` &s)
- `std::string` `get_name` () const
- unsigned `get_nparams` () const

## Protected Member Functions

- bool `has_derivative` () const
- bool `has_power` () const
- void `test_and_set_nparams` (unsigned n)
- void `set_print_func` (unsigned id, `print_funcp` f)



## Protected Attributes

- `std::string` `name`
- `std::string` `TeX_name`
- `unsigned` `nparams`
- `eval_funcp` `eval_f`
- `evalf_funcp` `evalf_f`
- `conjugate_funcp` `conjugate_f`
- `real_part_funcp` `real_part_f`
- `imag_part_funcp` `imag_part_f`
- `expand_funcp` `expand_f`
- `derivative_funcp` `derivative_f`
- `expl_derivative_funcp` `expl_derivative_f`
- `power_funcp` `power_f`
- `series_funcp` `series_f`
- `std::vector< print_funcp >` `print_dispatch_table`
- `info_funcp` `info_f`
- `bool` `evalf_params_first`
- `bool` `use_return_type`
- `unsigned` `return_type`
- `return_type_t` `return_type_tinfo`
- `bool` `use_remember`
- `unsigned` `remember_size`
- `unsigned` `remember_assoc_size`
- `unsigned` `remember_strategy`
- `bool` `eval_use_exvector_args`
- `bool` `evalf_use_exvector_args`
- `bool` `conjugate_use_exvector_args`
- `bool` `real_part_use_exvector_args`
- `bool` `imag_part_use_exvector_args`
- `bool` `expand_use_exvector_args`
- `bool` `derivative_use_exvector_args`
- `bool` `expl_derivative_use_exvector_args`
- `bool` `power_use_exvector_args`
- `bool` `series_use_exvector_args`
- `bool` `print_use_exvector_args`
- `bool` `info_use_exvector_args`
- `unsigned` `functions_with_same_name`
- `ex` `symtree`

## Friends

- `class` `function`
- `class` `fderivative`

### 6.64.1 Constructor & Destructor Documentation

**6.64.1.1 function\_options()** [1/3]

```
GiNaC::function_options::function_options ( )
```

References initialize().

**6.64.1.2 function\_options()** [2/3]

```
GiNaC::function_options::function_options (
    std::string const & n,
    std::string const & tn = std::string() )
```

References initialize(), n, and set\_name().

**6.64.1.3 function\_options()** [3/3]

```
GiNaC::function_options::function_options (
    std::string const & n,
    unsigned np )
```

References initialize(), n, nparams, and set\_name().

**6.64.1.4 ~function\_options()**

```
GiNaC::function_options::~~function_options ( )
```

**6.64.2 Member Function Documentation****6.64.2.1 initialize()**

```
void GiNaC::function_options::initialize ( )
```

References conjugate\_f, conjugate\_use\_exvector\_args, derivative\_f, derivative\_use\_exvector\_args, eval\_f, eval\_use\_exvector\_args, evalf\_f, evalf\_params\_first, evalf\_use\_exvector\_args, expand\_f, expand\_use\_exvector\_args, expl\_derivative\_f, expl\_derivative\_use\_exvector\_args, functions\_with\_same\_name, imag\_part\_f, imag\_part\_use\_exvector\_args, info\_f, info\_use\_exvector\_args, nparams, power\_f, power\_use\_exvector\_args, print\_use\_exvector\_args, real\_part\_f, real\_part\_use\_exvector\_args, series\_f, series\_use\_exvector\_args, set\_name(), symtree, use\_remember, and use\_return\_type.

Referenced by function\_options().

#### 6.64.2.2 dummy()

```
function_options & GiNaC::function_options::dummy ( ) [inline]
```

#### 6.64.2.3 set\_name()

```
function_options & GiNaC::function_options::set_name (
    std::string const & n,
    std::string const & tn = std::string() )
```

References `n`, `name`, and `TeX_name`.

Referenced by `function_options()`, and `initialize()`.

#### 6.64.2.4 latex\_name()

```
function_options & GiNaC::function_options::latex_name (
    std::string const & tn )
```

References `TeX_name`.

#### 6.64.2.5 eval\_func() [1/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_1 e )
```

References `eval_f`, and `test_and_set_nparams()`.

#### 6.64.2.6 eval\_func() [2/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_2 e )
```

References `eval_f`, and `test_and_set_nparams()`.

#### 6.64.2.7 eval\_func() [3/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_3 e )
```

References `eval_f`, and `test_and_set_nparams()`.

**6.64.2.8 eval\_func()** [4/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_4 e )
```

References eval\_f, and test\_and\_set\_nparams().

**6.64.2.9 eval\_func()** [5/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_5 e )
```

References eval\_f, and test\_and\_set\_nparams().

**6.64.2.10 eval\_func()** [6/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_6 e )
```

References eval\_f, and test\_and\_set\_nparams().

**6.64.2.11 eval\_func()** [7/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_7 e )
```

References eval\_f, and test\_and\_set\_nparams().

**6.64.2.12 eval\_func()** [8/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_8 e )
```

References eval\_f, and test\_and\_set\_nparams().

**6.64.2.13 eval\_func()** [9/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_9 e )
```

References eval\_f, and test\_and\_set\_nparams().

**6.64.2.14 eval\_func()** [10/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_10 e )
```

References eval\_f, and test\_and\_set\_nparams().

**6.64.2.15 eval\_func()** [11/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_11 e )
```

References eval\_f, and test\_and\_set\_nparams().

**6.64.2.16 eval\_func()** [12/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_12 e )
```

References eval\_f, and test\_and\_set\_nparams().

**6.64.2.17 eval\_func()** [13/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_13 e )
```

References eval\_f, and test\_and\_set\_nparams().

**6.64.2.18 eval\_func()** [14/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_14 e )
```

References eval\_f, and test\_and\_set\_nparams().

**6.64.2.19 evalf\_func()** [1/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_1 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.20 evalf\_func()** [2/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_2 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.21 evalf\_func()** [3/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_3 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.22 evalf\_func()** [4/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_4 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.23 evalf\_func()** [5/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_5 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.24 evalf\_func()** [6/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_6 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.25 evalf\_func()** [7/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_7 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.26 evalf\_func()** [8/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_8 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.27 evalf\_func()** [9/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_9 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.28 evalf\_func()** [10/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_10 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.29 evalf\_func()** [11/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_11 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.30 evalf\_func()** [12/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_12 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.31 evalf\_func()** [13/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_13 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.32 evalf\_func()** [14/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_14 e )
```

References evalf\_f, and test\_and\_set\_nparams().

**6.64.2.33 conjugate\_func()** [1/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_1 e )
```

References conjugate\_f, and test\_and\_set\_nparams().

**6.64.2.34 conjugate\_func()** [2/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_2 e )
```

References conjugate\_f, and test\_and\_set\_nparams().

**6.64.2.35 conjugate\_func()** [3/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_3 e )
```

References conjugate\_f, and test\_and\_set\_nparams().

**6.64.2.36 conjugate\_func()** [4/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_4 e )
```

References conjugate\_f, and test\_and\_set\_nparams().

**6.64.2.37 conjugate\_func()** [5/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_5 e )
```

References conjugate\_f, and test\_and\_set\_nparams().



**6.64.2.38 conjugate\_func()** [6/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_6 e )
```

References `conjugate_f`, and `test_and_set_nparams()`.

**6.64.2.39 conjugate\_func()** [7/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_7 e )
```

References `conjugate_f`, and `test_and_set_nparams()`.

**6.64.2.40 conjugate\_func()** [8/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_8 e )
```

References `conjugate_f`, and `test_and_set_nparams()`.

**6.64.2.41 conjugate\_func()** [9/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_9 e )
```

References `conjugate_f`, and `test_and_set_nparams()`.

**6.64.2.42 conjugate\_func()** [10/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_10 e )
```

References `conjugate_f`, and `test_and_set_nparams()`.

**6.64.2.43 conjugate\_func()** [11/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_11 e )
```

References `conjugate_f`, and `test_and_set_nparams()`.

**6.64.2.44 conjugate\_func()** [12/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_12 e )
```

References conjugate\_f, and test\_and\_set\_nparams().

**6.64.2.45 conjugate\_func()** [13/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_13 e )
```

References conjugate\_f, and test\_and\_set\_nparams().

**6.64.2.46 conjugate\_func()** [14/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_14 e )
```

References conjugate\_f, and test\_and\_set\_nparams().

**6.64.2.47 real\_part\_func()** [1/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_1 e )
```

References real\_part\_f, and test\_and\_set\_nparams().

**6.64.2.48 real\_part\_func()** [2/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_2 e )
```

References real\_part\_f, and test\_and\_set\_nparams().

**6.64.2.49 real\_part\_func()** [3/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_3 e )
```

References real\_part\_f, and test\_and\_set\_nparams().

**6.64.2.50** `real_part_func()` [4/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_4 e )
```

References `real_part_f`, and `test_and_set_nparams()`.

**6.64.2.51** `real_part_func()` [5/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_5 e )
```

References `real_part_f`, and `test_and_set_nparams()`.

**6.64.2.52** `real_part_func()` [6/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_6 e )
```

References `real_part_f`, and `test_and_set_nparams()`.

**6.64.2.53** `real_part_func()` [7/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_7 e )
```

References `real_part_f`, and `test_and_set_nparams()`.

**6.64.2.54** `real_part_func()` [8/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_8 e )
```

References `real_part_f`, and `test_and_set_nparams()`.

**6.64.2.55** `real_part_func()` [9/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_9 e )
```

References `real_part_f`, and `test_and_set_nparams()`.

**6.64.2.56 real\_part\_func()** [10/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_10 e )
```

References `real_part_f`, and `test_and_set_nparams()`.

**6.64.2.57 real\_part\_func()** [11/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_11 e )
```

References `real_part_f`, and `test_and_set_nparams()`.

**6.64.2.58 real\_part\_func()** [12/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_12 e )
```

References `real_part_f`, and `test_and_set_nparams()`.

**6.64.2.59 real\_part\_func()** [13/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_13 e )
```

References `real_part_f`, and `test_and_set_nparams()`.

**6.64.2.60 real\_part\_func()** [14/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_14 e )
```

References `real_part_f`, and `test_and_set_nparams()`.

**6.64.2.61 imag\_part\_func()** [1/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_1 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.62** `imag_part_func()` [2/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_2 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.63** `imag_part_func()` [3/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_3 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.64** `imag_part_func()` [4/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_4 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.65** `imag_part_func()` [5/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_5 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.66** `imag_part_func()` [6/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_6 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.67** `imag_part_func()` [7/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_7 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.68** `imag_part_func()` [8/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_8 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.69** `imag_part_func()` [9/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_9 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.70** `imag_part_func()` [10/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_10 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.71** `imag_part_func()` [11/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_11 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.72** `imag_part_func()` [12/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_12 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.73** `imag_part_func()` [13/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_13 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.74** `imag_part_func()` [14/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_14 e )
```

References `imag_part_f`, and `test_and_set_nparams()`.

**6.64.2.75** `expand_func()` [1/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_1 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.76** `expand_func()` [2/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_2 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.77** `expand_func()` [3/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_3 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.78** `expand_func()` [4/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_4 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.79** `expand_func()` [5/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_5 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.80** `expand_func()` [6/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_6 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.81** `expand_func()` [7/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_7 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.82** `expand_func()` [8/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_8 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.83** `expand_func()` [9/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_9 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.84** `expand_func()` [10/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_10 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.85** `expand_func()` [11/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_11 e )
```

References `expand_f`, and `test_and_set_nparams()`.



**6.64.2.86 expand\_func()** [12/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_12 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.87 expand\_func()** [13/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_13 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.88 expand\_func()** [14/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_14 e )
```

References `expand_f`, and `test_and_set_nparams()`.

**6.64.2.89 derivative\_func()** [1/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_1 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.90 derivative\_func()** [2/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_2 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.91 derivative\_func()** [3/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_3 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.92 derivative\_func()** [4/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_4 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.93 derivative\_func()** [5/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_5 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.94 derivative\_func()** [6/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_6 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.95 derivative\_func()** [7/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_7 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.96 derivative\_func()** [8/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_8 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.97 derivative\_func()** [9/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_9 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.98 derivative\_func()** [10/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_10 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.99 derivative\_func()** [11/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_11 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.100 derivative\_func()** [12/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_12 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.101 derivative\_func()** [13/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_13 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.102 derivative\_func()** [14/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_14 e )
```

References `derivative_f`, and `test_and_set_nparams()`.

**6.64.2.103 expl\_derivative\_func()** [1/15]

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_1 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.104** `expl_derivative_func()` [2/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_2 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.105** `expl_derivative_func()` [3/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_3 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.106** `expl_derivative_func()` [4/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_4 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.107** `expl_derivative_func()` [5/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_5 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.108** `expl_derivative_func()` [6/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_6 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.109** `expl_derivative_func()` [7/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_7 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.110** `expl_derivative_func()` [8/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_8 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.111** `expl_derivative_func()` [9/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_9 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.112** `expl_derivative_func()` [10/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_10 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.113** `expl_derivative_func()` [11/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_11 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.114** `expl_derivative_func()` [12/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_12 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.115** `expl_derivative_func()` [13/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_13 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.116** `expl_derivative_func()` [14/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_14 e )
```

References `expl_derivative_f`, and `test_and_set_nparams()`.

**6.64.2.117** `power_func()` [1/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_1 e )
```

References `power_f`, and `test_and_set_nparams()`.

**6.64.2.118** `power_func()` [2/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_2 e )
```

References `power_f`, and `test_and_set_nparams()`.

**6.64.2.119** `power_func()` [3/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_3 e )
```

References `power_f`, and `test_and_set_nparams()`.

**6.64.2.120** `power_func()` [4/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_4 e )
```

References `power_f`, and `test_and_set_nparams()`.

**6.64.2.121** `power_func()` [5/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_5 e )
```

References `power_f`, and `test_and_set_nparams()`.

**6.64.2.122 power\_func()** [6/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_6 e )
```

References `power_f`, and `test_and_set_nparams()`.

**6.64.2.123 power\_func()** [7/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_7 e )
```

References `power_f`, and `test_and_set_nparams()`.

**6.64.2.124 power\_func()** [8/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_8 e )
```

References `power_f`, and `test_and_set_nparams()`.

**6.64.2.125 power\_func()** [9/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_9 e )
```

References `power_f`, and `test_and_set_nparams()`.

**6.64.2.126 power\_func()** [10/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_10 e )
```

References `power_f`, and `test_and_set_nparams()`.

**6.64.2.127 power\_func()** [11/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_11 e )
```

References `power_f`, and `test_and_set_nparams()`.

**6.64.2.128 power\_func()** [12/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_12 e )
```

References power\_f, and test\_and\_set\_nparams().

**6.64.2.129 power\_func()** [13/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_13 e )
```

References power\_f, and test\_and\_set\_nparams().

**6.64.2.130 power\_func()** [14/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_14 e )
```

References power\_f, and test\_and\_set\_nparams().

**6.64.2.131 series\_func()** [1/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_1 e )
```

References series\_f, and test\_and\_set\_nparams().

**6.64.2.132 series\_func()** [2/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_2 e )
```

References series\_f, and test\_and\_set\_nparams().

**6.64.2.133 series\_func()** [3/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_3 e )
```

References series\_f, and test\_and\_set\_nparams().



**6.64.2.134 series\_func()** [4/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_4 e )
```

References `series_f`, and `test_and_set_nparams()`.

**6.64.2.135 series\_func()** [5/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_5 e )
```

References `series_f`, and `test_and_set_nparams()`.

**6.64.2.136 series\_func()** [6/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_6 e )
```

References `series_f`, and `test_and_set_nparams()`.

**6.64.2.137 series\_func()** [7/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_7 e )
```

References `series_f`, and `test_and_set_nparams()`.

**6.64.2.138 series\_func()** [8/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_8 e )
```

References `series_f`, and `test_and_set_nparams()`.

**6.64.2.139 series\_func()** [9/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_9 e )
```

References `series_f`, and `test_and_set_nparams()`.

**6.64.2.140 series\_func()** [10/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_10 e )
```

References series\_f, and test\_and\_set\_nparams().

**6.64.2.141 series\_func()** [11/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_11 e )
```

References series\_f, and test\_and\_set\_nparams().

**6.64.2.142 series\_func()** [12/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_12 e )
```

References series\_f, and test\_and\_set\_nparams().

**6.64.2.143 series\_func()** [13/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_13 e )
```

References series\_f, and test\_and\_set\_nparams().

**6.64.2.144 series\_func()** [14/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_14 e )
```

References series\_f, and test\_and\_set\_nparams().

**6.64.2.145 info\_func()** [1/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_1 e )
```

References info\_f, and test\_and\_set\_nparams().

**6.64.2.146** info\_func() [2/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_2 e )
```

References info\_f, and test\_and\_set\_nparams().

**6.64.2.147** info\_func() [3/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_3 e )
```

References info\_f, and test\_and\_set\_nparams().

**6.64.2.148** info\_func() [4/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_4 e )
```

References info\_f, and test\_and\_set\_nparams().

**6.64.2.149** info\_func() [5/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_5 e )
```

References info\_f, and test\_and\_set\_nparams().

**6.64.2.150** info\_func() [6/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_6 e )
```

References info\_f, and test\_and\_set\_nparams().

**6.64.2.151** info\_func() [7/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_7 e )
```

References info\_f, and test\_and\_set\_nparams().

**6.64.2.152** `info_func()` [8/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_8 e )
```

References `info_f`, and `test_and_set_nparams()`.

**6.64.2.153** `info_func()` [9/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_9 e )
```

References `info_f`, and `test_and_set_nparams()`.

**6.64.2.154** `info_func()` [10/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_10 e )
```

References `info_f`, and `test_and_set_nparams()`.

**6.64.2.155** `info_func()` [11/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_11 e )
```

References `info_f`, and `test_and_set_nparams()`.

**6.64.2.156** `info_func()` [12/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_12 e )
```

References `info_f`, and `test_and_set_nparams()`.

**6.64.2.157** `info_func()` [13/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_13 e )
```

References `info_f`, and `test_and_set_nparams()`.

**6.64.2.158** `info_func()` [14/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_14 e )
```

References `info_f`, and `test_and_set_nparams()`.

**6.64.2.159** `eval_func()` [15/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_exvector e )
```

References `eval_f`, and `eval_use_exvector_args`.

**6.64.2.160** `evalf_func()` [15/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_exvector e )
```

References `evalf_f`, and `evalf_use_exvector_args`.

**6.64.2.161** `conjugate_func()` [15/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_exvector e )
```

References `conjugate_f`, and `conjugate_use_exvector_args`.

**6.64.2.162** `real_part_func()` [15/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_exvector e )
```

References `real_part_f`, and `real_part_use_exvector_args`.

**6.64.2.163** `imag_part_func()` [15/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_exvector e )
```

References `imag_part_f`, and `imag_part_use_exvector_args`.

**6.64.2.164** `expand_func()` [15/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_exvector e )
```

References `expand_f`, and `expand_use_exvector_args`.

**6.64.2.165** `derivative_func()` [15/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_exvector e )
```

References `derivative_f`, and `derivative_use_exvector_args`.

**6.64.2.166** `expl_derivative_func()` [15/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_exvector e )
```

References `expl_derivative_f`, and `expl_derivative_use_exvector_args`.

**6.64.2.167** `power_func()` [15/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_exvector e )
```

References `power_f`, and `power_use_exvector_args`.

**6.64.2.168** `series_func()` [15/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_exvector e )
```

References `series_f`, and `series_use_exvector_args`.

**6.64.2.169** `info_func()` [15/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_exvector e )
```

References `info_f`, and `info_use_exvector_args`.

**6.64.2.170 print\_func()** [1/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_1 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.171 print\_func()** [2/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_2 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.172 print\_func()** [3/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_3 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.173 print\_func()** [4/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_4 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.174 print\_func()** [5/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_5 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.175 print\_func()** [6/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_6 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.176 print\_func()** [7/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_7 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.177 print\_func()** [8/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_8 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.178 print\_func()** [9/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_9 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.179 print\_func()** [10/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_10 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().



**6.64.2.180 print\_func()** [11/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_11 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.181 print\_func()** [12/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_12 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.182 print\_func()** [13/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_13 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.183 print\_func()** [14/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_14 p ) [inline]
```

References options, set\_print\_func(), and test\_and\_set\_nparams().

**6.64.2.184 print\_func()** [15/15]

```
template<class Ctx >
function_options& GiNaC::function_options::print_func (
    print_funcp_exvector p ) [inline]
```

References options, print\_use\_exvector\_args, and set\_print\_func().

#### 6.64.2.185 set\_return\_type()

```
function_options & GiNaC::function_options::set_return_type (
    unsigned rt,
    const return_type_t * rtt = nullptr )
```

References `return_type`, `return_type_tinfo`, and `use_return_type`.

#### 6.64.2.186 do\_not\_evalf\_params()

```
function_options & GiNaC::function_options::do_not_evalf_params ( )
```

References `evalf_params_first`.

#### 6.64.2.187 remember()

```
function_options & GiNaC::function_options::remember (
    unsigned size,
    unsigned assoc_size = 0,
    unsigned strategy = remember_strategies::delete_never )
```

References `remember_assoc_size`, `remember_size`, `remember_strategy`, and `use_remember`.

#### 6.64.2.188 overloaded()

```
function_options & GiNaC::function_options::overloaded (
    unsigned o )
```

References `functions_with_same_name`.

#### 6.64.2.189 set\_symmetry()

```
function_options & GiNaC::function_options::set_symmetry (
    const symmetry & s )
```

References `symtree`.

**6.64.2.190 get\_name()**

```
std::string GiNaC::function_options::get_name ( ) const [inline]
```

References name.

**6.64.2.191 get\_nparams()**

```
unsigned GiNaC::function_options::get_nparams ( ) const [inline]
```

References nparams.

**6.64.2.192 has\_derivative()**

```
bool GiNaC::function_options::has_derivative ( ) const [inline], [protected]
```

References derivative\_f.

**6.64.2.193 has\_power()**

```
bool GiNaC::function_options::has_power ( ) const [inline], [protected]
```

References power\_f.

**6.64.2.194 test\_and\_set\_nparams()**

```
void GiNaC::function_options::test_and_set_nparams (
    unsigned n ) [protected]
```

References n, name, and nparams.

Referenced by conjugate\_func(), derivative\_func(), eval\_func(), evalf\_func(), expand\_func(), expl\_derivative\_func(), imag\_part\_func(), info\_func(), power\_func(), print\_func(), real\_part\_func(), and series\_func().

**6.64.2.195 set\_print\_func()**

```
void GiNaC::function_options::set_print_func (
    unsigned id,
    print_funcp f ) [protected]
```

References print\_dispatch\_table.

Referenced by print\_func().

### 6.64.3 Friends And Related Function Documentation

#### 6.64.3.1 function

```
friend class function [friend]
```

#### 6.64.3.2 fderivative

```
friend class fderivative [friend]
```

### 6.64.4 Member Data Documentation

#### 6.64.4.1 name

```
std::string GiNaC::function_options::name [protected]
```

Referenced by `get_name()`, `GiNaC::function::print()`, `GiNaC::function::register_new()`, `set_name()`, and `test_and_set_nparams()`.

#### 6.64.4.2 TeX\_name

```
std::string GiNaC::function_options::TeX_name [protected]
```

Referenced by `latex_name()`, `GiNaC::function::print()`, and `set_name()`.

#### 6.64.4.3 nparams

```
unsigned GiNaC::function_options::nparams [protected]
```

Referenced by `GiNaC::function::conjugate()`, `GiNaC::function::eval()`, `GiNaC::function::evalf()`, `GiNaC::function::expand()`, `GiNaC::function::expl_derivative()`, `function_options()`, `get_nparams()`, `GiNaC::function::imag_part()`, `GiNaC::function::info()`, `initialize()`, `GiNaC::function::pderivative()`, `GiNaC::function::power()`, `GiNaC::function::print()`, `GiNaC::function::real_part()`, `GiNaC::function::series()`, and `test_and_set_nparams()`.

#### 6.64.4.4 eval\_f

`eval_funcp` GiNaC::function\_options::eval\_f [protected]

Referenced by GiNaC::function::eval(), eval\_func(), and initialize().

#### 6.64.4.5 evalf\_f

`evalf_funcp` GiNaC::function\_options::evalf\_f [protected]

Referenced by GiNaC::function::evalf(), evalf\_func(), and initialize().

#### 6.64.4.6 conjugate\_f

`conjugate_funcp` GiNaC::function\_options::conjugate\_f [protected]

Referenced by GiNaC::function::conjugate(), conjugate\_func(), and initialize().

#### 6.64.4.7 real\_part\_f

`real_part_funcp` GiNaC::function\_options::real\_part\_f [protected]

Referenced by initialize(), GiNaC::function::real\_part(), and real\_part\_func().

#### 6.64.4.8 imag\_part\_f

`imag_part_funcp` GiNaC::function\_options::imag\_part\_f [protected]

Referenced by GiNaC::function::imag\_part(), imag\_part\_func(), and initialize().

#### 6.64.4.9 expand\_f

`expand_funcp` GiNaC::function\_options::expand\_f [protected]

Referenced by GiNaC::function::expand(), expand\_func(), and initialize().

#### 6.64.4.10 derivative\_f

`derivative_funcp` `GiNaC::function_options::derivative_f` [protected]

Referenced by `derivative_func()`, `has_derivative()`, `initialize()`, and `GiNaC::function::pderivative()`.

#### 6.64.4.11 expl\_derivative\_f

`expl_derivative_funcp` `GiNaC::function_options::expl_derivative_f` [protected]

Referenced by `GiNaC::function::expl_derivative()`, `expl_derivative_func()`, and `initialize()`.

#### 6.64.4.12 power\_f

`power_funcp` `GiNaC::function_options::power_f` [protected]

Referenced by `has_power()`, `initialize()`, `GiNaC::function::power()`, and `power_func()`.

#### 6.64.4.13 series\_f

`series_funcp` `GiNaC::function_options::series_f` [protected]

Referenced by `initialize()`, `GiNaC::function::series()`, and `series_func()`.

#### 6.64.4.14 print\_dispatch\_table

`std::vector<print_funcp>` `GiNaC::function_options::print_dispatch_table` [protected]

Referenced by `GiNaC::function::print()`, and `set_print_func()`.

#### 6.64.4.15 info\_f

`info_funcp` `GiNaC::function_options::info_f` [protected]

Referenced by `GiNaC::function::info()`, `info_func()`, and `initialize()`.

#### 6.64.4.16 evalf\_params\_first

```
bool GiNaC::function_options::evalf_params_first [protected]
```

Referenced by `do_not_evalf_params()`, `GiNaC::function::evalf()`, and `initialize()`.

#### 6.64.4.17 use\_return\_type

```
bool GiNaC::function_options::use_return_type [protected]
```

Referenced by `initialize()`, `GiNaC::function::return_type()`, `GiNaC::function::return_type_tinfo()`, and `set_return_type()`.

#### 6.64.4.18 return\_type

```
unsigned GiNaC::function_options::return_type [protected]
```

Referenced by `GiNaC::function::return_type()`, and `set_return_type()`.

#### 6.64.4.19 return\_type\_tinfo

```
return_type_t GiNaC::function_options::return_type_tinfo [protected]
```

Referenced by `GiNaC::function::return_type_tinfo()`, and `set_return_type()`.

#### 6.64.4.20 use\_remember

```
bool GiNaC::function_options::use_remember [protected]
```

Referenced by `GiNaC::function::eval()`, `initialize()`, `GiNaC::function::register_new()`, and `remember()`.

#### 6.64.4.21 remember\_size

```
unsigned GiNaC::function_options::remember_size [protected]
```

Referenced by `GiNaC::function::register_new()`, and `remember()`.

#### 6.64.4.22 remember\_assoc\_size

`unsigned GiNaC::function_options::remember_assoc_size [protected]`

Referenced by `GiNaC::function::register_new()`, and `remember()`.

#### 6.64.4.23 remember\_strategy

`unsigned GiNaC::function_options::remember_strategy [protected]`

Referenced by `GiNaC::function::register_new()`, and `remember()`.

#### 6.64.4.24 eval\_use\_exvector\_args

`bool GiNaC::function_options::eval_use_exvector_args [protected]`

Referenced by `GiNaC::function::eval()`, `eval_func()`, and `initialize()`.

#### 6.64.4.25 evalf\_use\_exvector\_args

`bool GiNaC::function_options::evalf_use_exvector_args [protected]`

Referenced by `GiNaC::function::evalf()`, `evalf_func()`, and `initialize()`.

#### 6.64.4.26 conjugate\_use\_exvector\_args

`bool GiNaC::function_options::conjugate_use_exvector_args [protected]`

Referenced by `GiNaC::function::conjugate()`, `conjugate_func()`, and `initialize()`.

#### 6.64.4.27 real\_part\_use\_exvector\_args

`bool GiNaC::function_options::real_part_use_exvector_args [protected]`

Referenced by `initialize()`, `GiNaC::function::real_part()`, and `real_part_func()`.



#### 6.64.4.28 imag\_part\_use\_exvector\_args

```
bool GiNaC::function_options::imag_part_use_exvector_args [protected]
```

Referenced by `GiNaC::function::imag_part()`, `imag_part_func()`, and `initialize()`.

#### 6.64.4.29 expand\_use\_exvector\_args

```
bool GiNaC::function_options::expand_use_exvector_args [protected]
```

Referenced by `GiNaC::function::expand()`, `expand_func()`, and `initialize()`.

#### 6.64.4.30 derivative\_use\_exvector\_args

```
bool GiNaC::function_options::derivative_use_exvector_args [protected]
```

Referenced by `derivative_func()`, `initialize()`, and `GiNaC::function::pderivative()`.

#### 6.64.4.31 expl\_derivative\_use\_exvector\_args

```
bool GiNaC::function_options::expl_derivative_use_exvector_args [protected]
```

Referenced by `GiNaC::function::expl_derivative()`, `expl_derivative_func()`, and `initialize()`.

#### 6.64.4.32 power\_use\_exvector\_args

```
bool GiNaC::function_options::power_use_exvector_args [protected]
```

Referenced by `initialize()`, `GiNaC::function::power()`, and `power_func()`.

#### 6.64.4.33 series\_use\_exvector\_args

```
bool GiNaC::function_options::series_use_exvector_args [protected]
```

Referenced by `initialize()`, `GiNaC::function::series()`, and `series_func()`.

#### 6.64.4.34 `print_use_exvector_args`

```
bool GiNaC::function_options::print_use_exvector_args [protected]
```

Referenced by `initialize()`, `GiNaC::function::print()`, and `print_func()`.

#### 6.64.4.35 `info_use_exvector_args`

```
bool GiNaC::function_options::info_use_exvector_args [protected]
```

Referenced by `GiNaC::function::info()`, `info_func()`, and `initialize()`.

#### 6.64.4.36 `functions_with_same_name`

```
unsigned GiNaC::function_options::functions_with_same_name [protected]
```

Referenced by `initialize()`, `overloaded()`, and `GiNaC::function::register_new()`.

#### 6.64.4.37 `symtree`

```
ex GiNaC::function_options::symtree [protected]
```

Referenced by `GiNaC::function::eval()`, `initialize()`, and `set_symmetry()`.

The documentation for this class was generated from the following files:

- [function.h](#)
- [function.cpp](#)

## 6.65 `GiNaC::G2_SERIAL` Class Reference

Generalized multiple polylogarithm.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.65.1 Detailed Description

Generalized multiple polylogarithm.

### 6.65.2 Member Data Documentation

#### 6.65.2.1 serial

```
unsigned GiNaC::G2_SERIAL::serial [static]
```

#### Initial value:

```
= function::register_new(function_options("G", 2).  
    evalf_func(G2_evalf).  
    eval_func(G2_eval).  
    overloaded(2))
```

Referenced by `GiNaC::G()`.

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## 6.66 GiNaC::G3\_SERIAL Class Reference

Generalized multiple polylogarithm with explicit imaginary parts.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.66.1 Detailed Description

Generalized multiple polylogarithm with explicit imaginary parts.

### 6.66.2 Member Data Documentation

### 6.66.2.1 serial

```
unsigned GiNaC::G3_SERIAL::serial [static]
```

#### Initial value:

```
= function::register_new(function_options("G", 3).
    evalf_func(G3_evalf).
    eval_func(G3_eval).
    overloaded(2))
```

Referenced by `GiNaC::G()`.

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## 6.67 GiNaC::gcd\_options Struct Reference

Flags to control the behavior of `gcd()` and friends.

```
#include <normal.h>
```

### Public Types

- enum { `no_heur_gcd` = 2, `no_part_factored` = 4, `use_sr_gcd` = 8 }

### 6.67.1 Detailed Description

Flags to control the behavior of `gcd()` and friends.

### 6.67.2 Member Enumeration Documentation

#### 6.67.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

<code>no_heur_gcd</code>	Usually <code>GiNaC</code> tries heuristic GCD first, because typically it's much faster than anything else. Even if heuristic algorithm fails, the overhead is negligible w.r.t. cost of computing the GCD by some other method. However, some people dislike it, so here's a flag which tells <code>GiNaC</code> to NOT use the heuristic algorithm.
<code>no_part_factored</code>	<code>GiNaC</code> tries to avoid expanding expressions when computing GCDs. This is a good idea, but some people dislike it. Hence the flag to disable special handling of partially factored polynomials. DON'T SET THIS unless you <i>really</i> know what are you doing!
<code>use_sr_gcd</code>	By default <code>GiNaC</code> uses modular GCD algorithm. Typically it's much faster than PRS (pseudo remainder sequence) algorithm. This flag forces <code>GiNaC</code> to use PRS algorithm

The documentation for this struct was generated from the following file:

- [normal.h](#)

## 6.68 GiNaC::gcdheu\_failed Class Reference

Exception thrown by [heur\\_gcd\(\)](#) to signal failure.

### 6.68.1 Detailed Description

Exception thrown by [heur\\_gcd\(\)](#) to signal failure.

The documentation for this class was generated from the following file:

- [normal.cpp](#)

## 6.69 GiNaC::has\_distance< T > Class Template Reference

SFINAE test for distance.

```
#include <utils_multi_iterator.h>
```

### Public Types

- enum { [value](#) = sizeof(test<T>(0)) == sizeof(yes\_type) }

### Private Types

- typedef char [yes\\_type](#)[1]
- typedef char [no\\_type](#)[2]

### Static Private Member Functions

- template<typename C >  
static [yes\\_type](#) & [test](#) (decltype(std::distance< C >))
- template<typename C >  
static [no\\_type](#) & [test](#) (...)

### 6.69.1 Detailed Description

```
template<typename T>
class GiNaC::has_distance< T >
```

SFINAE test for distance.

## 6.69.2 Member Typedef Documentation

### 6.69.2.1 yes\_type

```
template<typename T >
typedef char GiNaC::has_distance< T >::yes_type[1] [private]
```

### 6.69.2.2 no\_type

```
template<typename T >
typedef char GiNaC::has_distance< T >::no_type[2] [private]
```

## 6.69.3 Member Enumeration Documentation

### 6.69.3.1 anonymous enum

```
template<typename T >
anonymous enum
```

#### Enumerator

value	
-------	--

## 6.69.4 Member Function Documentation

### 6.69.4.1 test() [1/2]

```
template<typename T >
template<typename C >
static yes_type& GiNaC::has_distance< T >::test (
    decltype(std::distance< C >) ) [static], [private]
```

## 6.69.4.2 test() [2/2]

```
template<typename T >
template<typename C >
static no_type& GiNaC::has_distance< T >::test (
    ... ) [static], [private]
```

The documentation for this class was generated from the following file:

- [utils\\_multi\\_iterator.h](#)

## 6.70 GiNaC::has\_options Class Reference

Flags to control the behavior of [has\(\)](#).

```
#include <flags.h>
```

## Public Types

- enum { [algebraic](#) = 0x0001 }

## 6.70.1 Detailed Description

Flags to control the behavior of [has\(\)](#).

## 6.70.2 Member Enumeration Documentation

## 6.70.2.1 anonymous enum

```
anonymous enum
```

## Enumerator

<a href="#">algebraic</a>	enable algebraic matching
---------------------------	---------------------------

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.71 std::hash&lt; GiNaC::ex &gt; Struct Template Reference

Specialization of std::hash() for ex objects.

```
#include <ex.h>
```

## Public Member Functions

- `std::size_t operator()` (const [GiNaC::ex](#) &e) const noexcept

### 6.71.1 Detailed Description

```
template<>
struct std::hash< GiNaC::ex >
```

Specialization of `std::hash()` for `ex` objects.

### 6.71.2 Member Function Documentation

#### 6.71.2.1 operator>()

```
std::size_t std::hash< GiNaC::ex >::operator() (
    const GiNaC::ex & e ) const [inline], [noexcept]
```

The documentation for this struct was generated from the following file:

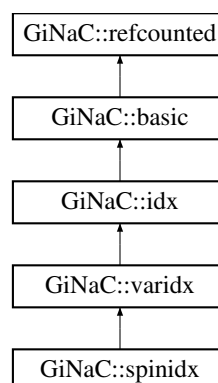
- [ex.h](#)

## 6.72 GiNaC::idx Class Reference

This class holds one index of an indexed object.

```
#include <idx.h>
```

Inheritance diagram for `GiNaC::idx`:





## Public Member Functions

- `idx` (const `ex` &`v`, const `ex` &`dim`)  
*Construct index with given value and dimension.*
- `bool info` (unsigned int) const override  
*Information about the object.*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex map` (map\_function &`f`) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex evalf` () const override  
*By default, `basic::evalf` would evaluate the index value but we don't want `a.1` to become `a`.*
- `ex subs` (const `exmap` &`m`, unsigned `options`=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `void archive` (archive\_node &`n`) const override  
*Save (serialize) the object into archive node.*
- `void read_archive` (const archive\_node &`n`, lst &`syms`) override  
*Load (deserialize) the object from an archive node.*
- `virtual bool is_dummy_pair_same_type` (const `basic` &`other`) const  
*Check whether the index forms a dummy index pair with another index of the same type.*
- `ex get_value` () const  
*Get value of index.*
- `bool is_numeric` () const  
*Check whether the index is numeric.*
- `bool is_symbolic` () const  
*Check whether the index is symbolic.*
- `ex get_dim` () const  
*Get dimension of index space.*
- `bool is_dim_numeric` () const  
*Check whether the dimension is numeric.*
- `bool is_dim_symbolic` () const  
*Check whether the dimension is symbolic.*
- `ex replace_dim` (const `ex` &`new_dim`) const  
*Make a new index with the same value but a different dimension.*
- `ex minimal_dim` (const `idx` &`other`) const  
*Return the minimum of the dimensions of this and another index.*

## Protected Member Functions

- `ex derivative` (const `symbol` &`s`) const override  
*Implementation of `ex::diff()` for an index always returns 0.*
- `bool match_same_type` (const `basic` &`other`) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `unsigned calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void print_index` (const `print_context` &`c`, unsigned level) const
- `void do_print` (const `print_context` &`c`, unsigned level) const
- `void do_print_csrc` (const `print_csrc` &`c`, unsigned level) const
- `void do_print_latex` (const `print_latex` &`c`, unsigned level) const
- `void do_print_tree` (const `print_tree` &`c`, unsigned level) const

## Protected Attributes

- [ex value](#)  
*Expression that constitutes the index (numeric or symbolic name)*
- [ex dim](#)  
*Dimension of space (can be symbolic or numeric)*

### 6.72.1 Detailed Description

This class holds one index of an indexed object.

Indices can theoretically consist of any symbolic expression but they are usually only just a symbol (e.g. "mu", "i") or numeric (integer). Indices belong to a space with a certain numeric or symbolic dimension.

### 6.72.2 Constructor & Destructor Documentation

#### 6.72.2.1 idx()

```
GiNaC::idx::idx (
    const ex & v,
    const ex & dim ) [explicit]
```

Construct index with given value and dimension.

#### Parameters

<i>v</i>	Value of index (numeric or symbolic)
<i>dim</i>	Dimension of index space (numeric or symbolic)

#### Returns

newly constructed index

References `dim`, `GiNaC::ex::info()`, `is_dim_numeric()`, and `GiNaC::info_flags::posint`.

### 6.72.3 Member Function Documentation

#### 6.72.3.1 info()

```
bool GiNaC::idx::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::has\\_indices](#), and [GiNaC::info\\_flags::idx](#).

### 6.72.3.2 nops()

```
size_t GiNaC::idx::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.72.3.3 op()

```
ex GiNaC::idx::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [value](#).

### 6.72.3.4 map()

```
ex GiNaC::idx::map (
    map_function & f ) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::status\\_↔flags::hash\\_calculated](#), and [value](#).

### 6.72.3.5 evalf()

```
ex GiNaC::idx::evalf ( ) const [override], [virtual]
```

By default, [basic::evalf](#) would evaluate the index value but we don't want a.1 to become a.

(1.0).

Reimplemented from [GiNaC::basic](#).

### 6.72.3.6 subs()

```
ex GiNaC::idx::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [m](#), [options](#), [GiNaC::subs\\_options::really\\_subs\\_idx](#), [GiNaC::ex::subs\(\)](#), and [value](#).

Referenced by [GiNaC::spinmetric::contract\\_with\(\)](#).

### 6.72.3.7 archive()

```
void GiNaC::idx::archive (
    archive_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::spinidx](#), and [GiNaC::varidx](#).

References [dim](#), [n](#), and [value](#).

## 6.72.3.8 read\_archive()

```
void GiNaC::idx::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

**Note**

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::spinidx](#), and [GiNaC::varidx](#).

References `dim`, `n`, and `value`.

## 6.72.3.9 derivative()

```
ex GiNaC::idx::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for an index always returns 0.

**See also**

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References `GiNaC::_ex0`.

## 6.72.3.10 match\_same\_type()

```
bool GiNaC::idx::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

**See also**

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::spinidx](#), and [GiNaC::varidx](#).

References `dim`, `GINAC_ASSERT`, and `GiNaC::ex::is_equal()`.

### 6.72.3.11 calchash()

```
unsigned GiNaC::idx::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class `basic` computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::ex::gethash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::make\\_hash\\_seed\(\)](#), [GiNaC::rotate\\_left\(\)](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

### 6.72.3.12 is\_dummy\_pair\_same\_type()

```
bool GiNaC::idx::is_dummy_pair_same_type (
    const basic & other ) const [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented in [GiNaC::spinidx](#), and [GiNaC::varidx](#).

References [dim](#), [GiNaC::ex::is\\_equal\(\)](#), and [value](#).

Referenced by [GiNaC::is\\_dummy\\_pair\(\)](#).

### 6.72.3.13 get\_value()

```
ex GiNaC::idx::get_value ( ) const [inline]
```

Get value of index.

References [value](#).

Referenced by [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), and [GiNaC::spinmetric::eval\\_indexed\(\)](#).

### 6.72.3.14 is\_numeric()

```
bool GiNaC::idx::is_numeric ( ) const [inline]
```

Check whether the index is numeric.

References [value](#).

#### 6.72.3.15 is\_symbolic()

```
bool GiNaC::idx::is_symbolic ( ) const [inline]
```

Check whether the index is symbolic.

References value.

Referenced by GiNaC::spinmetric::contract\_with(), and GiNaC::tensor::replace\_contr\_index().

#### 6.72.3.16 get\_dim()

```
ex GiNaC::idx::get_dim ( ) const [inline]
```

Get dimension of index space.

References dim.

Referenced by GiNaC::matrix::eval\_indexed(), GiNaC::tensdelta::eval\_indexed(), and GiNaC::tensmetric::eval\_indexed().

#### 6.72.3.17 is\_dim\_numeric()

```
bool GiNaC::idx::is_dim_numeric ( ) const [inline]
```

Check whether the dimension is numeric.

References dim.

Referenced by idx().

#### 6.72.3.18 is\_dim\_symbolic()

```
bool GiNaC::idx::is_dim_symbolic ( ) const [inline]
```

Check whether the dimension is symbolic.

References dim.

### 6.72.3.19 replace\_dim()

```
ex GiNaC::idx::replace_dim (
    const ex & new_dim ) const
```

Make a new index with the same value but a different dimension.

References `GiNaC::basic::clearflag()`, `dim`, `GiNaC::basic::duplicate()`, and `GiNaC::status_flags::hash_calculated`.

Referenced by `GiNaC::tensdelta::eval_indexed()`, `GiNaC::tensmetric::eval_indexed()`, and `GiNaC::tensor::replace_↵_contr_index()`.

### 6.72.3.20 minimal\_dim()

```
ex GiNaC::idx::minimal_dim (
    const idx & other ) const
```

Return the minimum of the dimensions of this and another index.

If this is undecidable, throw an exception.

References `dim`, and `GiNaC::minimal_dim()`.

Referenced by `GiNaC::tensdelta::eval_indexed()`, `GiNaC::tensmetric::eval_indexed()`, and `GiNaC::tensor::replace_↵_contr_index()`.

### 6.72.3.21 print\_index()

```
void GiNaC::idx::print_index (
    const print_context & c,
    unsigned level ) const [protected]
```

References `c`, `dim`, `GiNaC::ex::print()`, `GiNaC::print_options::print_index_dimensions`, and `value`.

Referenced by `do_print()`, `GiNaC::varidx::do_print()`, `GiNaC::spinidx::do_print()`, `do_print_latex()`, and `GiNaC_↵::spinidx::do_print_latex()`.

### 6.72.3.22 do\_print()

```
void GiNaC::idx::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References `c`, and `print_index()`.



**6.72.3.23 do\_print\_csrc()**

```
void GiNaC::idx::do_print_csrc (
    const print\_csrc & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::print\(\)](#), and [value](#).

**6.72.3.24 do\_print\_latex()**

```
void GiNaC::idx::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_index\(\)](#).

**6.72.3.25 do\_print\_tree()**

```
void GiNaC::idx::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [dim](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::ex::print\(\)](#), and [value](#).

**6.72.4 Member Data Documentation****6.72.4.1 value**

```
ex GiNaC::idx::value [protected]
```

Expression that constitutes the index (numeric or symbolic name)

Referenced by [archive\(\)](#), [calchash\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spindex::do\\_print\\_tree\(\)](#), [get\\_value\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [is\\_numeric\(\)](#), [is\\_symbolic\(\)](#), [map\(\)](#), [op\(\)](#), [print\\_index\(\)](#), [read\\_archive\(\)](#), and [subs\(\)](#).

#### 6.72.4.2 dim

`ex` GiNaC::idx::dim [protected]

Dimension of space (can be symbolic or numeric)

Referenced by `archive()`, `do_print_tree()`, `GiNaC::varidx::do_print_tree()`, `GiNaC::spinidx::do_print_tree()`, `get_↵  
dim()`, `idx()`, `is_dim_numeric()`, `is_dim_symbolic()`, `is_dummy_pair_same_type()`, `match_same_type()`, `minimal_↵  
dim()`, `print_index()`, `read_archive()`, and `replace_dim()`.

The documentation for this class was generated from the following files:

- [idx.h](#)
- [idx.cpp](#)

### 6.73 GiNaC::idx\_is\_equal\_ignore\_dim Struct Reference

#### Public Member Functions

- `bool operator()` (const `ex` &lh, const `ex` &rh) const

#### 6.73.1 Member Function Documentation

##### 6.73.1.1 operator()

```
bool GiNaC::idx_is_equal_ignore_dim::operator() (
    const ex & lh,
    const ex & rh ) const [inline]
```

References `GiNaC::ex::is_equal()`.

The documentation for this struct was generated from the following file:

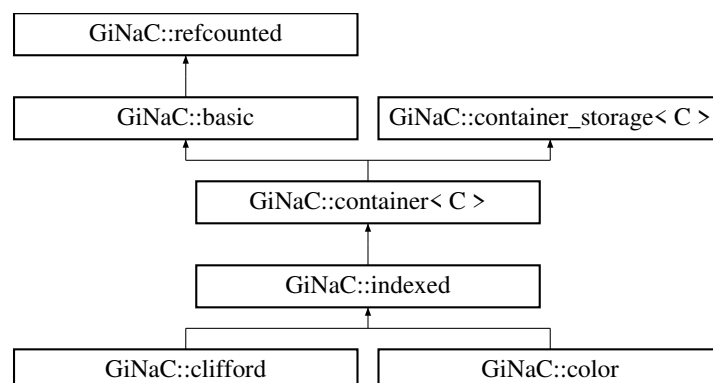
- [indexed.cpp](#)

### 6.74 GiNaC::indexed Class Reference

This class holds an indexed expression.

```
#include <indexed.h>
```

Inheritance diagram for `GiNaC::indexed`:



## Public Member Functions

- [indexed](#) (const [ex](#) &b)  
*Construct indexed object with no index.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1)  
*Construct indexed object with one index.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1, const [ex](#) &i2)  
*Construct indexed object with two indices.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Construct indexed object with three indices.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4)  
*Construct indexed object with four indices.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [ex](#) &i1, const [ex](#) &i2)  
*Construct indexed object with two indices and a specified symmetry.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Construct indexed object with three indices and a specified symmetry.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4)  
*Construct indexed object with four indices and a specified symmetry.*
- [indexed](#) (const [ex](#) &b, const [exvector](#) &iv)  
*Construct indexed object with a specified vector of indices.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [exvector](#) &iv)  
*Construct indexed object with a specified vector of indices and symmetry.*
- [indexed](#) (const [symmetry](#) &symm, const [exprseq](#) &es)
- [indexed](#) (const [symmetry](#) &symm, const [exvector](#) &v)
- [indexed](#) (const [symmetry](#) &symm, [exvector](#) &&v)
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- [exvector get\\_free\\_indices](#) () const override  
*Return a vector containing the free indices of an expression.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &symbols) override  
*Read (a.k.a.*
- bool [all\\_index\\_values\\_are](#) (unsigned inf) const  
*Check whether all index values have a certain property.*
- [exvector get\\_indices](#) () const  
*Return a vector containing the object's indices.*
- [exvector get\\_dummy\\_indices](#) () const  
*Return a vector containing the dummy indices of the object, if any.*
- [exvector get\\_dummy\\_indices](#) (const [indexed](#) &other) const  
*Return a vector containing the dummy indices in the contraction with another indexed object.*
- bool [has\\_dummy\\_index\\_for](#) (const [ex](#) &i) const  
*Check whether the object has an index that forms a dummy index pair with a given index.*
- [ex get\\_symmetry](#) () const  
*Return symmetry properties.*

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for an indexed object always returns 0.*
- `ex thiscontainer` (const `exvector` &v) const override
- `ex thiscontainer` (`exvector` &&v) const override
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- void `printindices` (const `print_context` &c, unsigned level) const
- void `print_indexed` (const `print_context` &c, const char \*openbrace, const char \*closebrace, unsigned level) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `validate` () const  
*Check whether all indices are of class `idx` and validate the symmetry tree.*

## Protected Attributes

- `ex symtree`  
*Index symmetry (tree of symmetry objects)*

## Friends

- `ex simplify_indexed` (const `ex` &e, `exvector` &free\_indices, `exvector` &dummy\_indices, const `scalar_products` &sp)  
*Simplify indexed expression, return list of free indices.*
- `ex simplify_indexed_product` (const `ex` &e, `exvector` &free\_indices, `exvector` &dummy\_indices, const `scalar_products` &sp)  
*Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.*
- bool `reposition_dummy_indices` (`ex` &e, `exvector` &variant\_dummy\_indices, `exvector` &moved\_indices)  
*Raise/lower dummy indices in a single indexed objects to canonicalize their variance.*

## Additional Inherited Members

### 6.74.1 Detailed Description

This class holds an indexed expression.

It consists of a 'base' expression (the expression being indexed) which can be accessed as `op(0)`, and `n` (`n >= 0`) indices (all of class `idx`), accessible as `op(1)..op(n)`.

### 6.74.2 Constructor & Destructor Documentation

#### 6.74.2.1 `indexed()` [1/13]

```
GiNaC::indexed::indexed (
    const ex & b )
```

Construct indexed object with no index.

## Parameters

<i>b</i>	Base expression
----------	-----------------

## Returns

newly constructed indexed object

Referenced by `GiNaC::clifford::get_metric()`, and `thiscontainer()`.

6.74.2.2 `indexed()` [2/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & i1 )
```

Construct indexed object with one index.

The index must be of class `idx`.

## Parameters

<i>b</i>	Base expression
<i>i1</i>	The index

## Returns

newly constructed indexed object

6.74.2.3 `indexed()` [3/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & i1,
    const ex & i2 )
```

Construct indexed object with two indices.

The indices must be of class `idx`.

## Parameters

<i>b</i>	Base expression
<i>i1</i>	First index
<i>i2</i>	Second index

**Returns**

newly constructed indexed object

**6.74.2.4 indexed()** [4/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & i1,
    const ex & i2,
    const ex & i3 )
```

Construct indexed object with three indices.

The indices must be of class idx.

**Parameters**

<i>b</i>	Base expression
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index

**Returns**

newly constructed indexed object

**6.74.2.5 indexed()** [5/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & i1,
    const ex & i2,
    const ex & i3,
    const ex & i4 )
```

Construct indexed object with four indices.

The indices must be of class idx.

**Parameters**

<i>b</i>	Base expression
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index
<i>i4</i>	Fourth index

**Returns**

newly constructed indexed object

**6.74.2.6 indexed()** [6/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const symmetry & symm,
    const ex & i1,
    const ex & i2 )
```

Construct indexed object with two indices and a specified symmetry.

The indices must be of class idx.

**Parameters**

<i>b</i>	Base expression
<i>symm</i>	Symmetry of indices
<i>i1</i>	First index
<i>i2</i>	Second index

**Returns**

newly constructed indexed object

**6.74.2.7 indexed()** [7/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const symmetry & symm,
    const ex & i1,
    const ex & i2,
    const ex & i3 )
```

Construct indexed object with three indices and a specified symmetry.

The indices must be of class idx.

**Parameters**

<i>b</i>	Base expression
<i>symm</i>	Symmetry of indices
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index

**Returns**

newly constructed indexed object

**6.74.2.8 indexed()** [8/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const symmetry & symm,
    const ex & i1,
    const ex & i2,
    const ex & i3,
    const ex & i4 )
```

Construct indexed object with four indices and a specified symmetry.

The indices must be of class *idx*.

**Parameters**

<i>b</i>	Base expression
<i>symm</i>	Symmetry of indices
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index
<i>i4</i>	Fourth index

**Returns**

newly constructed indexed object

**6.74.2.9 indexed()** [9/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const exvector & iv )
```

Construct indexed object with a specified vector of indices.

The indices must be of class *idx*.

**Parameters**

<i>b</i>	Base expression
<i>iv</i>	Vector of indices



**Returns**

newly constructed indexed object

**6.74.2.10 indexed()** [10/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const symmetry & symm,
    const exvector & iv )
```

Construct indexed object with a specified vector of indices and symmetry.

The indices must be of class `idx`.

**Parameters**

<i>b</i>	Base expression
<i>symm</i>	Symmetry of indices
<i>iv</i>	Vector of indices

**Returns**

newly constructed indexed object

**6.74.2.11 indexed()** [11/13]

```
GiNaC::indexed::indexed (
    const symmetry & symm,
    const exprseq & es )
```

**6.74.2.12 indexed()** [12/13]

```
GiNaC::indexed::indexed (
    const symmetry & symm,
    const exvector & v )
```

**6.74.2.13 indexed()** [13/13]

```
GiNaC::indexed::indexed (
    const symmetry & symm,
    exvector && v )
```

### 6.74.3 Member Function Documentation

#### 6.74.3.1 precedence()

```
unsigned GiNaC::indexed::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< C >](#).

Referenced by `print_indexed()`.

#### 6.74.3.2 info()

```
bool GiNaC::indexed::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::container< C >](#).

References `GiNaC::info_flags::has_indices`, `GiNaC::info_flags::indexed`, and `GiNaC::container_storage< C >::seq`.

Referenced by `imag_part()`, and `real_part()`.

#### 6.74.3.3 eval()

```
ex GiNaC::indexed::eval ( ) const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References `GiNaC::_ex0`, `GiNaC::canonicalize()`, `GiNaC::basic::eval_indexed()`, `GiNaC::basic::ex`, `GINAC_ASSE←RT`, `GiNaC::ex::is_zero()`, `GiNaC::ex::nops()`, `GiNaC::ex::op()`, `GiNaC::container_storage< C >::seq`, `symtree`, and `thiscontainer()`.

#### 6.74.3.4 `real_part()`

```
ex GiNaC::indexed::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References `info()`, `GiNaC::container< C >::op()`, and `GiNaC::info_flags::real`.

#### 6.74.3.5 `imag_part()`

```
ex GiNaC::indexed::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References `info()`, `GiNaC::container< C >::op()`, and `GiNaC::info_flags::real`.

#### 6.74.3.6 `get_free_indices()`

```
exvector GiNaC::indexed::get_free_indices ( ) const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References `GiNaC::find_free_and_dummy()`, and `GiNaC::container_storage< C >::seq`.

Referenced by `get_dummy_indices()`.

#### 6.74.3.7 `archive()`

```
void GiNaC::indexed::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) indexed object into archive.

Reimplemented from [GiNaC::container< C >](#).

References `n`, and `symtree`.

#### 6.74.3.8 read\_archive()

```
void GiNaC::indexed::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) indexed object from archive.

Reimplemented from [GiNaC::container< C >](#).

References [n](#), [GiNaC::not\\_symmetric\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_↔symm\(\)](#), [GiNaC::symm\(\)](#), [symtree](#), and [GiNaC::symmetry::validate\(\)](#).

#### 6.74.3.9 derivative()

```
ex GiNaC::indexed::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for an indexed object always returns 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#).

#### 6.74.3.10 thiscontainer() [1/2]

```
ex GiNaC::indexed::thiscontainer (
    const exvector & v ) const [override], [protected]
```

References [indexed\(\)](#), and [symtree](#).

Referenced by [eval\(\)](#), and [expand\(\)](#).

#### 6.74.3.11 thiscontainer() [2/2]

```
ex GiNaC::indexed::thiscontainer (
    exvector && v ) const [override], [protected]
```

References [indexed\(\)](#), and [symtree](#).

6.74.3.12 `return_type()`

```
unsigned GiNaC::indexed::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::container< C >::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

6.74.3.13 `return_type_tinfo()`

```
return_type_t GiNaC::indexed::return_type_tinfo ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container< C >::op\(\)](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#).

6.74.3.14 `expand()`

```
ex GiNaC::indexed::expand (
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex0](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::expand\\_options::expand\\_indexed](#), [GINAC\\_ASSERT](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [options](#), [GiNaC::container\\_storage< C >::seq](#), and [thiscontainer\(\)](#).

6.74.3.15 `all_index_values_are()`

```
bool GiNaC::indexed::all_index_values_are (
    unsigned inf ) const
```

Check whether all index values have a certain property.

See also

class [info\\_flags](#)

References [GiNaC::container\\_storage< C >::seq](#).

**6.74.3.16** `get_indices()`

```
exvector GiNaC::indexed::get_indices ( ) const
```

Return a vector containing the object's indices.

References `GINAC_ASSERT`, and `GiNaC::container_storage< C >::seq`.

**6.74.3.17** `get_dummy_indices()` [1/2]

```
exvector GiNaC::indexed::get_dummy_indices ( ) const
```

Return a vector containing the dummy indices of the object, if any.

References `GiNaC::find_free_and_dummy()`, and `GiNaC::container_storage< C >::seq`.

**6.74.3.18** `get_dummy_indices()` [2/2]

```
exvector GiNaC::indexed::get_dummy_indices (
    const indexed & other ) const
```

Return a vector containing the dummy indices in the contraction with another indexed object.

This is symmetric: `a.get_dummy_indices(b) == b.get_dummy_indices(a)`

References `GiNaC::find_dummy_indices()`, and `get_free_indices()`.

**6.74.3.19** `has_dummy_index_for()`

```
bool GiNaC::indexed::has_dummy_index_for (
    const ex & i ) const
```

Check whether the object has an index that forms a dummy index pair with a given index.

References `GiNaC::is_dummy_pair()`, and `GiNaC::container_storage< C >::seq`.

**6.74.3.20** `get_symmetry()`

```
ex GiNaC::indexed::get_symmetry ( ) const [inline]
```

Return symmetry properties.

References `symtree`.

Referenced by `GiNaC::clifford::get_metric()`.

## 6.74.3.21 printindices()

```
void GiNaC::indexed::printindices (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::clifford::do\\_print\\_dflt\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [do\\_print\\_tree\(\)](#), and [print\\_indexed\(\)](#).

## 6.74.3.22 print\_indexed()

```
void GiNaC::indexed::print_indexed (
    const print\_context & c,
    const char * openbrace,
    const char * closebrace,
    unsigned level ) const [protected]
```

References [c](#), [precedence\(\)](#), [printindices\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).

## 6.74.3.23 do\_print()

```
void GiNaC::indexed::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_indexed\(\)](#).

## 6.74.3.24 do\_print\_latex()

```
void GiNaC::indexed::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_indexed\(\)](#).

## 6.74.3.25 do\_print\_tree()

```
void GiNaC::indexed::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [printindices\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [symtree](#).

#### 6.74.3.26 validate()

```
void GiNaC::indexed::validate ( ) const [protected]
```

Check whether all indices are of class `idx` and validate the symmetry tree.

This function is used internally to make sure that all constructed indexed objects really carry indices and not some other classes.

References `GINAC_ASSERT`, `GiNaC::ex::is_zero()`, `GiNaC::container_storage< C >::seq`, `symtree`, and `GiNaC::symmetry::validate()`.

### 6.74.4 Friends And Related Function Documentation

#### 6.74.4.1 simplify\_indexed

```
ex simplify_indexed (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar_products & sp ) [friend]
```

Simplify indexed expression, return list of free indices.

Referenced by `GiNaC::clifford::get_metric()`, and `GiNaC::clifford::same_metric()`.

#### 6.74.4.2 simplify\_indexed\_product

```
ex simplify_indexed_product (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar_products & sp ) [friend]
```

Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.

#### 6.74.4.3 reposition\_dummy\_indices

```
bool reposition_dummy_indices (
    ex & e,
    exvector & variant_dummy_indices,
    exvector & moved_indices ) [friend]
```

Raise/lower dummy indices in a single indexed objects to canonicalize their variance.



## Parameters

<i>e</i>	Object to work on
<i>variant_dummy_indices</i>	The set of indices that might need repositioning (will be changed by this function)
<i>moved_indices</i>	The set of indices that have been repositioned (will be changed by this function)

## Returns

true if 'e' was changed

## 6.74.5 Member Data Documentation

## 6.74.5.1 symtree

```
ex GiNaC::indexed::symtree [protected]
```

Index symmetry (tree of symmetry objects)

Referenced by `archive()`, `GiNaC::clifford::do_print_tree()`, `do_print_tree()`, `eval()`, `get_symmetry()`, `read_archive()`, `thiscontainer()`, and `validate()`.

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)

## 6.75 GiNaC::info\_flags Class Reference

Possible attributes an object can have.

```
#include <flags.h>
```

## Public Types

- enum {  
[numeric](#), [real](#), [rational](#), [integer](#),  
[crational](#), [cinteger](#), [positive](#), [negative](#),  
[nonnegative](#), [posint](#), [negint](#), [nonnegint](#),  
[even](#), [odd](#), [prime](#), [relation](#),  
[relation\\_equal](#), [relation\\_not\\_equal](#), [relation\\_less](#), [relation\\_less\\_or\\_equal](#),  
[relation\\_greater](#), [relation\\_greater\\_or\\_equal](#), [symbol](#), [list](#),  
[exprseq](#), [polynomial](#), [integer\\_polynomial](#), [cinteger\\_polynomial](#),  
[rational\\_polynomial](#), [crational\\_polynomial](#), [rational\\_function](#), [indexed](#),  
[has\\_indices](#), [idx](#), [expanded](#), [indefinite](#) }

### 6.75.1 Detailed Description

Possible attributes an object can have.

### 6.75.2 Member Enumeration Documentation

#### 6.75.2.1 anonymous enum

anonymous enum

##### Enumerator

numeric	
real	
rational	
integer	
crational	
cinteger	
positive	
negative	
nonnegative	
posint	
negint	
nonnegint	
even	
odd	
prime	
relation	
relation_equal	
relation_not_equal	
relation_less	
relation_less_or_equal	
relation_greater	
relation_greater_or_equal	
symbol	
list	
exprseq	
polynomial	
integer_polynomial	
cinteger_polynomial	
rational_polynomial	
crational_polynomial	
rational_function	
indexed	
has_indices	
idx	
expanded	
indefinite	

The documentation for this class was generated from the following file:

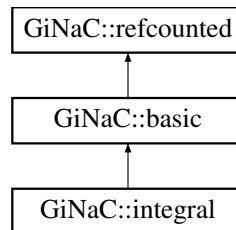
- [flags.h](#)

## 6.76 GiNaC::integral Class Reference

Symbolic integral.

```
#include <integral.h>
```

Inheritance diagram for GiNaC::integral:



### Public Member Functions

- `integral` (const `ex` &x\_, const `ex` &a\_, const `ex` &b\_, const `ex` &f\_)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex evalf` () const override  
*Evaluate object numerically.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*
- int `ldegree` (const `ex` &s) const override  
*Return degree of lowest power in object s.*
- `ex eval_ncmul` (const `exvector` &v) const override
- size\_t `nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex conjugate` () const override
- `ex eval_integ` () const override  
*Evaluate integrals, if result is known.*
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Read (a.k.a.*

## Static Public Attributes

- static int `max_integration_level` = 15
- static `ex` `relative_integration_error` = 1e-8

## Protected Member Functions

- `ex` `derivative` (const `symbol` &s) const override  
*Default implementation of `ex::diff()`.*
- `ex` `series` (const `relational` &r, int `order`, unsigned `options`=0) const override  
*Default implementation of `ex::series()`.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Private Attributes

- `ex` x
- `ex` a
- `ex` b
- `ex` f

## Additional Inherited Members

### 6.76.1 Detailed Description

Symbolic integral.

### 6.76.2 Constructor & Destructor Documentation

#### 6.76.2.1 `integral()`

```
GiNaC::integral::integral (
    const ex & x_,
    const ex & a_,
    const ex & b_,
    const ex & f_ )
```

References x.

Referenced by `derivative()`, `expand()`, and `series()`.

### 6.76.3 Member Function Documentation

### 6.76.3.1 precedence()

```
unsigned GiNaC::integral::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by `do_print_latex()`.

### 6.76.3.2 eval()

```
ex GiNaC::integral::eval ( ) const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References `GiNaC::_ex0`, `a`, `b`, `GiNaC::status_flags::evaluated`, `f`, `GiNaC::basic::flags`, `GiNaC::ex::has()`, `GiNaC::ex::haswild()`, `GiNaC::basic::hold()`, and `x`.

### 6.76.3.3 evalf()

```
ex GiNaC::integral::evalf ( ) const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References `a`, `GiNaC::adaptivesimpson()`, `GiNaC::are_ex_trivially_equal()`, `b`, `GiNaC::ex::evalf()`, `f`, `GiNaC::ex::subs()`, and `x`.

### 6.76.3.4 degree()

```
int GiNaC::integral::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object `s`.

Reimplemented from [GiNaC::basic](#).

References `a`, `b`, and `f`.

#### 6.76.3.5 ldegree()

```
int GiNaC::integral::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References a, b, and f.

#### 6.76.3.6 eval\_ncmul()

```
ex GiNaC::integral::eval_ncmul (
    const exvector & v ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::eval\\_ncmul\(\)](#), and f.

#### 6.76.3.7 nops()

```
size_t GiNaC::integral::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 6.76.3.8 op()

```
ex GiNaC::integral::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References a, b, f, GINAC\_ASSERT, and x.

## 6.76.3.9 let\_op()

```
ex & GiNaC::integral::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References a, b, [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), f, and x.

## 6.76.3.10 expand()

```
ex GiNaC::integral::expand (
    unsigned options = 0 ) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References a, [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), b, [GiNaC::basic::ex](#), [GiNaC::ex::expand\(\)](#), [GiNaC::status\\_flags](#)↔  
↔[::expanded](#), f, [GiNaC::basic::flags](#), [GiNaC::ex::has\(\)](#), [integral\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), options, [Gi-](#)↔  
↔[NaC::basic::setflag\(\)](#), and x.

Referenced by [eval\\_integ\(\)](#).

## 6.76.3.11 get\_free\_indices()

```
exvector GiNaC::integral::get_free_indices ( ) const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References a, b, f, and [GiNaC::ex::get\\_free\\_indices\(\)](#).

## 6.76.3.12 return\_type()

```
unsigned GiNaC::integral::return_type ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References f, and [GiNaC::ex::return\\_type\(\)](#).

**6.76.3.13 return\_type\_tinfo()**

```
return_type_t GiNaC::integral::return_type_tinfo ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [f](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#).

**6.76.3.14 conjugate()**

```
ex GiNaC::integral::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [a](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [b](#), [GiNaC::ex::conjugate\(\)](#), [f](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**6.76.3.15 eval\_integ()**

```
ex GiNaC::integral::eval_integ ( ) const [override], [virtual]
```

Evaluate integrals, if result is known.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::ex::eval\\_integ\(\)](#), [expand\(\)](#), [GiNaC::status\\_flags::expanded](#), [f](#), [GiNaC::basic::flags](#), [GiNaC::ex::has\(\)](#), [GiNaC::log\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**6.76.3.16 archive()**

```
void GiNaC::integral::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [f](#), [n](#), and [x](#).



#### 6.76.3.17 read\_archive()

```
void GiNaC::integral::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [f](#), [n](#), and [x](#).

#### 6.76.3.18 derivative()

```
ex GiNaC::integral::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::ex::diff\(\)](#), [f](#), [integral\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

#### 6.76.3.19 series()

```
ex GiNaC::integral::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [a](#), [b](#), [f](#), [integral\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::nops\(\)](#), [options](#), [order](#), [r](#), [GiNaC::ex::series\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::to\\_int\(\)](#), and [x](#).

**6.76.3.20 do\_print()**

```
void GiNaC::integral::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [a](#), [b](#), [c](#), [f](#), [GiNaC::ex::print\(\)](#), and [x](#).

**6.76.3.21 do\_print\_latex()**

```
void GiNaC::integral::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [a](#), [b](#), [c](#), [f](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), and [x](#).

**6.76.4 Member Data Documentation****6.76.4.1 max\_integration\_level**

```
int GiNaC::integral::max_integration_level = 15 [static]
```

Referenced by [GiNaC::adaptivesimpson\(\)](#).

**6.76.4.2 relative\_integration\_error**

```
ex GiNaC::integral::relative_integration_error = 1e-8 [static]
```

**6.76.4.3 x**

```
ex GiNaC::integral::x [private]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [expand\(\)](#), [integral\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [series\(\)](#).

## 6.76.4.4 a

```
ex GiNaC::integral::a [private]
```

Referenced by archive(), conjugate(), degree(), derivative(), do\_print(), do\_print\_latex(), eval(), eval\_integ(), evalf(), expand(), get\_free\_indices(), ldegree(), let\_op(), op(), read\_archive(), and series().

## 6.76.4.5 b

```
ex GiNaC::integral::b [private]
```

Referenced by archive(), conjugate(), degree(), derivative(), do\_print(), do\_print\_latex(), eval(), eval\_integ(), evalf(), expand(), get\_free\_indices(), ldegree(), let\_op(), op(), read\_archive(), and series().

## 6.76.4.6 f

```
ex GiNaC::integral::f [private]
```

Referenced by archive(), conjugate(), degree(), derivative(), do\_print(), do\_print\_latex(), eval(), eval\_integ(), eval\_↵ncmul(), evalf(), expand(), get\_free\_indices(), ldegree(), let\_op(), op(), read\_archive(), return\_type(), return\_type\_↵\_tinfo(), and series().

The documentation for this class was generated from the following files:

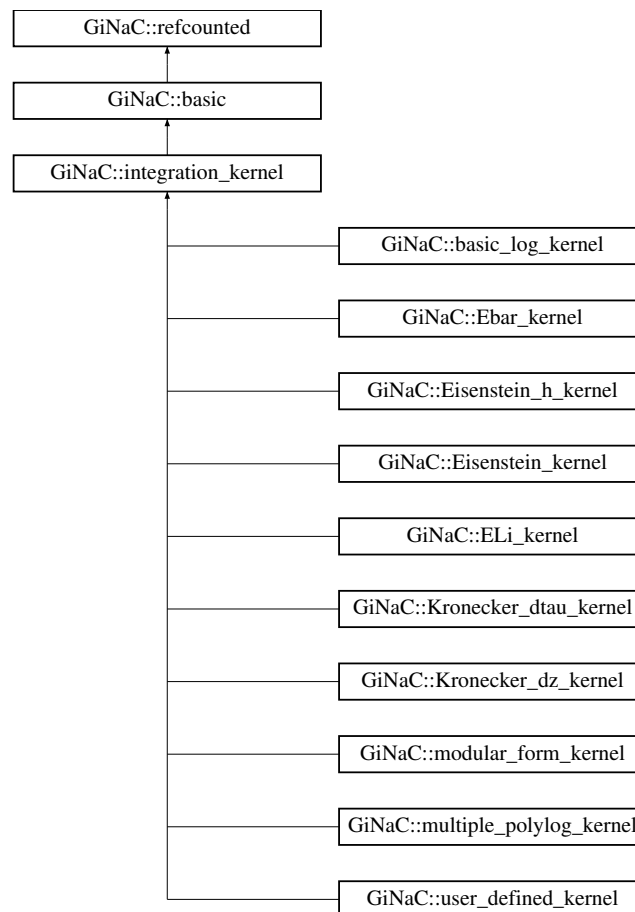
- [integral.h](#)
- [indexed.cpp](#)
- [integral.cpp](#)
- [pseries.cpp](#)

## 6.77 GiNaC::integration\_kernel Class Reference

The base class for integration kernels for iterated integrals.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::integration\_kernel:



## Public Member Functions

- `ex series` (const relational &r, int order, unsigned options=0) const override  
*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool `is_numeric` (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual `ex Laurent_series` (const ex &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual `ex get_numerical_value` (const ex &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t `get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int i) const  
*Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.*
- `cln::cl_N series_coeff` (int i) const  
*Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

## Protected Member Functions

- virtual bool [uses\\_Laurent\\_series](#) () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- virtual `cln::cl_N` [series\\_coeff\\_impl](#) (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex` [get\\_numerical\\_value\\_impl](#) (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void [do\\_print](#) (const `print_context` &c, unsigned level) const

## Protected Attributes

- int [cache\\_step\\_size](#)
- `std::vector< cln::cl_N >` [series\\_vec](#)

### 6.77.1 Detailed Description

The base class for integration kernels for iterated integrals.

This class represents the differential one-form

$$\omega = d\lambda$$

The integration variable is a dummy variable and does not need to be specified.

### 6.77.2 Member Function Documentation

#### 6.77.2.1 `series()`

```
ex GiNaC::integration_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Default implementation of `ex::series()`.

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::modular\\_form\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), and [GiNaC::Eisenstein\\_kernel](#).

References `order`, and `r`.

### 6.77.2.2 has\_trailing\_zero()

```
bool GiNaC::integration_kernel::has_trailing_zero (
    void ) const [virtual]
```

This routine returns true, if the integration kernel has a trailing zero.

### 6.77.2.3 is\_numeric()

```
bool GiNaC::integration_kernel::is_numeric (
    void ) const [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented in [GiNaC::user\\_defined\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::ELi\\_kernel](#), and [GiNaC::multiple\\_polylog\\_kernel](#).

### 6.77.2.4 Laurent\_series()

```
ex GiNaC::integration_kernel::Laurent_series (
    const ex & x,
    int order ) const [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented in [GiNaC::user\\_defined\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), and [GiNaC::Eisenstein\\_kernel](#).

References `n`, `order`, [GiNaC::pow\(\)](#), [GiNaC::ex::series\(\)](#), and `x`.

### 6.77.2.5 get\_numerical\_value()

```
ex GiNaC::integration_kernel::get_numerical_value (
    const ex & lambda,
    int N_trunc = 0 ) const [virtual]
```

Evaluates the integrand at `lambda`.

Reimplemented in [GiNaC::modular\\_form\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Ebar\\_kernel](#), and [GiNaC::ELi\\_kernel](#).

## 6.77.2.6 uses\_Laurent\_series()

```
bool GiNaC::integration_kernel::uses_Laurent_series ( ) const [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented in [GiNaC::user\\_defined\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), and [GiNaC::Eisenstein\\_kernel](#).

## 6.77.2.7 series\_coeff\_impl()

```
cln::cl_N GiNaC::integration_kernel::series_coeff_impl (
    int i ) const [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented in [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::multiple\\_polylog\\_kernel](#), and [GiNaC::basic\\_log\\_kernel](#).

## 6.77.2.8 get\_cache\_size()

```
size_t GiNaC::integration_kernel::get_cache_size (
    void ) const
```

Returns the current size of the cache.

## 6.77.2.9 set\_cache\_step()

```
void GiNaC::integration_kernel::set_cache_step (
    int cache_steps ) const
```

Sets the step size by which the cache is increased.

## 6.77.2.10 get\_series\_coeff()

```
ex GiNaC::integration_kernel::get_series_coeff (
    int i ) const
```

Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.

### 6.77.2.11 series\_coeff()

```
cln::cl_N GiNaC::integration_kernel::series_coeff (
    int i ) const
```

Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.

The method `series_coeff_impl` can be used, if a single coefficient can be computed independently of the others.

The method `Laurent_series` can be used, if it is more efficient to compute a Laurent series in one shot and to determine a range of coefficients from this Laurent series.

References `GiNaC::ex::coeff()`, `GiNaC::ex::evalf()`, and `x`.

### 6.77.2.12 get\_numerical\_value\_impl()

```
ex GiNaC::integration_kernel::get_numerical_value_impl (
    const ex & lambda,
    const ex & pre,
    int shift,
    int N_trunc ) const [protected]
```

The actual implementation for computing a numerical value for the integrand.

References `GiNaC::Digits`, `GiNaC::ex::evalf()`, and `one`.

Referenced by `GiNaC::ELi_kernel::get_numerical_value()`, `GiNaC::Ebar_kernel::get_numerical_value()`, `GiNaC::Kronecker_dtau_kernel::get_numerical_value()`, `GiNaC::Kronecker_dz_kernel::get_numerical_value()`, `GiNaC::Eisenstein_kernel::get_numerical_value()`, `GiNaC::Eisenstein_h_kernel::get_numerical_value()`, and `GiNaC::modular_form_kernel::get_numerical_value()`.

### 6.77.2.13 do\_print()

```
void GiNaC::integration_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References `c`.

## 6.77.3 Member Data Documentation

### 6.77.3.1 cache\_step\_size

```
int GiNaC::integration_kernel::cache_step_size [mutable], [protected]
```



### 6.77.3.2 series\_vec

```
std::vector<cln::cl_N> GiNaC::integration_kernel::series_vec [mutable], [protected]
```

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.78 GiNaC::is\_not\_a\_clifford Struct Reference

Predicate for finding non-clifford objects.

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &e)

### 6.78.1 Detailed Description

Predicate for finding non-clifford objects.

### 6.78.2 Member Function Documentation

#### 6.78.2.1 operator>()

```
bool GiNaC::is_not_a_clifford::operator() (
    const ex & e ) [inline]
```

The documentation for this struct was generated from the following file:

- [clifford.cpp](#)

## 6.79 GiNaC::is\_summation\_idx Struct Reference

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &e)

### 6.79.1 Member Function Documentation

### 6.79.1.1 operator()

```
bool GiNaC::is_summation_idx::operator() (
    const ex & e ) [inline]
```

References [GiNaC::is\\_dummy\\_pair\(\)](#).

The documentation for this struct was generated from the following file:

- [indexed.cpp](#)

## 6.80 GiNaC::iterated\_integral2\_SERIAL Class Reference

Complete elliptic integral of the first kind.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.80.1 Detailed Description

Complete elliptic integral of the first kind.

Complete elliptic integral of the second kind. Iterated integral.

### 6.80.2 Member Data Documentation

#### 6.80.2.1 serial

```
unsigned GiNaC::iterated_integral2_SERIAL::serial [static]
```

#### Initial value:

```
=
    function::register\_new(function_options("iterated_integral", 2).
        eval_func(iterated\_integral2\_eval).
        evalf_func(iterated\_integral2\_evalf).
        do_not_evalf_params().
        overloaded(2))
```

Referenced by [GiNaC::iterated\\_integral\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_elliptic.cpp](#)

## 6.81 GiNaC::iterated\_integral3\_SERIAL Class Reference

Iterated integral with explicit truncation.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.81.1 Detailed Description

Iterated integral with explicit truncation.

### 6.81.2 Member Data Documentation

#### 6.81.2.1 serial

```
unsigned GiNaC::iterated_integral3_SERIAL::serial [static]
```

**Initial value:**

```
=
    function::register_new(function_options("iterated_integral", 3).
        eval_func(iterated_integral3_eval).
        evalf_func(iterated_integral3_evalf).
        do_not_evalf_params().
        overloaded(2))
```

Referenced by `GiNaC::iterated_integral()`.

The documentation for this class was generated from the following files:

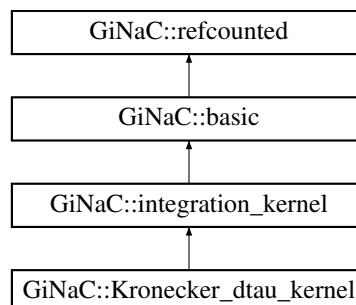
- [inifcns.h](#)
- [inifcns\\_elliptic.cpp](#)

## 6.82 GiNaC::Kronecker\_dtau\_kernel Class Reference

The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).

```
#include <integration_kernel.h>
```

Inheritance diagram for `GiNaC::Kronecker_dtau_kernel`:



## Public Member Functions

- `Kronecker_dtau_kernel` (const `ex` &`n`, const `ex` &`z`, const `ex` &`K=numeric(1)`, const `ex` &`C_norm=numeric(1)`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position  $i$ .*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position  $i$ .*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc=0`) const override  
*Returns the value of the  $g^{(n)}(z, K\tau)$ , where  $\tau$  is given by `qbar`.*

## Protected Member Functions

- `cln::cl_N series_coeff_impl` (int `i`) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `void do_print` (const `print_context` &`c`, unsigned level) const

## Protected Attributes

- `ex n`
- `ex z`
- `ex K`
- `ex C_norm`

### 6.82.1 Detailed Description

The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).

This class represents the differential one-form

$$\omega_{n,K}^{\text{Kronecker},\tau}(z_j) = \frac{C_n K(n-1)}{(2\pi i)^n} g^{(n)}(z_j, K\tau) \frac{d\bar{q}}{\bar{q}}$$

### 6.82.2 Constructor & Destructor Documentation

#### 6.82.2.1 Kronecker\_dtau\_kernel()

```
GiNaC::Kronecker_dtau_kernel::Kronecker_dtau_kernel (
    const ex & n,
    const ex & z,
    const ex & K = numeric(1),
    const ex & C_norm = numeric(1) )
```

### 6.82.3 Member Function Documentation

#### 6.82.3.1 nops()

```
size_t GiNaC::Kronecker_dtau_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 6.82.3.2 op()

```
ex GiNaC::Kronecker_dtau_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References C\_norm, K, n, and z.

#### 6.82.3.3 let\_op()

```
ex & GiNaC::Kronecker_dtau_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References C\_norm, [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), K, n, and z.

#### 6.82.3.4 is\_numeric()

```
bool GiNaC::Kronecker_dtau_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References C\_norm, [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), K, n, [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), and z.

### 6.82.3.5 `get_numerical_value()`

```
ex GiNaC::Kronecker_dtau_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of the  $g^{(n)}(z, K \cdot \tau)$ , where  $\tau$  is given by  $qbar$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References `C_norm`, `GiNaC::ex::evalf()`, `GiNaC::basic::evalf()`, `GiNaC::exp()`, `GiNaC::Ebar_kernel::get_numerical_value()`, `GiNaC::integration_kernel::get_numerical_value_impl()`, `GiNaC::I`, `K`, `n`, `GiNaC::Pi`, `GiNaC::pow()`,  $qbar$ , and  $z$ .

### 6.82.3.6 `series_coeff_impl()`

```
cln::cl_N GiNaC::Kronecker_dtau_kernel::series_coeff_impl (
    int i ) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References `GiNaC::bernoulli()`, `C_norm`, `GiNaC::ex::evalf()`, `GiNaC::basic::evalf()`, `GiNaC::exp()`, `GiNaC::factorial()`, `GiNaC::I`, `K`, `n`, `GiNaC::Pi`, `GiNaC::numeric::to_int()`, `GiNaC::to_int()`, and  $z$ .

### 6.82.3.7 `do_print()`

```
void GiNaC::Kronecker_dtau_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References  $c$ , `C_norm`, `K`, `n`, `GiNaC::ex::print()`, and  $z$ .

## 6.82.4 Member Data Documentation

### 6.82.4.1 `n`

```
ex GiNaC::Kronecker_dtau_kernel::n [protected]
```

Referenced by `do_print()`, `get_numerical_value()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

6.82.4.2 `z`

`ex` `GiNaC::Kronecker_dtau_kernel::z` [protected]

Referenced by `do_print()`, `get_numerical_value()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

6.82.4.3 `K`

`ex` `GiNaC::Kronecker_dtau_kernel::K` [protected]

Referenced by `do_print()`, `get_numerical_value()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

6.82.4.4 `C_norm`

`ex` `GiNaC::Kronecker_dtau_kernel::C_norm` [protected]

Referenced by `do_print()`, `get_numerical_value()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

The documentation for this class was generated from the following files:

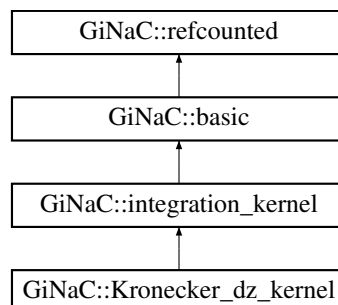
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.83 GiNaC::Kronecker\_dz\_kernel Class Reference

The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .

```
#include <integration_kernel.h>
```

Inheritance diagram for `GiNaC::Kronecker_dz_kernel`:



## Public Member Functions

- `Kronecker_dz_kernel` (const `ex` &`n`, const `ex` &`z_j`, const `ex` &`tau`, const `ex` &`K=numeric(1)`, const `ex` &`C_norm=numeric(1)`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position `i`.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value` (const `ex` &`z`, int `N_trunc=0`) const override  
*Returns the value of the  $g^{(n-1)}(z-z_j, K\tau)$ .*

## Protected Member Functions

- `cl::cl_N_series_coeff_impl` (int `i`) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `void do_print` (const `print_context` &`c`, unsigned `level`) const

## Protected Attributes

- `ex n`
- `ex z_j`
- `ex tau`
- `ex K`
- `ex C_norm`

### 6.83.1 Detailed Description

The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .

This class represents the differential one-form

$$\omega_{n,K}^{\text{Kronecker},z}(z_j, \tau) = C_n (2\pi i)^{2-n} g^{(n-1)}(z - z_j, K\tau) dz$$

### 6.83.2 Constructor & Destructor Documentation

#### 6.83.2.1 Kronecker\_dz\_kernel()

```
GiNaC::Kronecker_dz_kernel::Kronecker_dz_kernel (
    const ex & n,
    const ex & z_j,
    const ex & tau,
    const ex & K = numeric(1),
    const ex & C_norm = numeric(1) )
```



### 6.83.3 Member Function Documentation

#### 6.83.3.1 nops()

```
size_t GiNaC::Kronecker_dz_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 6.83.3.2 op()

```
ex GiNaC::Kronecker_dz_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References C\_norm, K, n, tau, and z\_j.

#### 6.83.3.3 let\_op()

```
ex & GiNaC::Kronecker_dz_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References C\_norm, [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), K, n, tau, and z\_j.

#### 6.83.3.4 is\_numeric()

```
bool GiNaC::Kronecker_dz_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References C\_norm, [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), K, n, [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), tau, and z\_j.

### 6.83.3.5 get\_numerical\_value()

```
ex GiNaC::Kronecker_dz_kernel::get_numerical_value (
    const ex & z,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of the  $g^{(n-1)}(z-z_j, K\tau)$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References `C_norm`, `GiNaC::integration_kernel::get_numerical_value_impl()`, `GiNaC::l`, `n`, `GiNaC::Pi`, and `GiNaC::C::pow()`.

### 6.83.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::Kronecker_dz_kernel::series_coeff_impl (
    int i ) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References `GiNaC::bernoulli()`, `C_norm`, `GiNaC::ex::evalf()`, `GiNaC::basic::evalf()`, `GiNaC::exp()`, `GiNaC::factorial()`, `GiNaC::Ebar_kernel::get_numerical_value()`, `GiNaC::l`, `GiNaC::is_even()`, `GiNaC::is_zero()`, `K`, `n`, `GiNaC::Pi`, `GiNaC::pow()`, `qbar`, `tau`, `GiNaC::numeric::to_int()`, and `z_j`.

### 6.83.3.7 do\_print()

```
void GiNaC::Kronecker_dz_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References `c`, `C_norm`, `K`, `n`, `GiNaC::ex::print()`, `tau`, and `z_j`.

## 6.83.4 Member Data Documentation

### 6.83.4.1 n

```
ex GiNaC::Kronecker_dz_kernel::n [protected]
```

Referenced by `do_print()`, `get_numerical_value()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

6.83.4.2 `z_j`

`ex GiNaC::Kronecker_dz_kernel::z_j` [protected]

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

6.83.4.3 `tau`

`ex GiNaC::Kronecker_dz_kernel::tau` [protected]

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

6.83.4.4 `K`

`ex GiNaC::Kronecker_dz_kernel::K` [protected]

Referenced by `do_print()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

6.83.4.5 `C_norm`

`ex GiNaC::Kronecker_dz_kernel::C_norm` [protected]

Referenced by `do_print()`, `get_numerical_value()`, `is_numeric()`, `let_op()`, `op()`, and `series_coeff_impl()`.

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.84 GiNaC::lanczos\_coeffs Class Reference

## Public Member Functions

- [lanczos\\_coeffs](#) ()
- bool [sufficiently\\_accurate](#) (int digits)
- int [get\\_order](#) () const
- `cln::cl_N` [calc\\_lanczos\\_A](#) (const `cln::cl_N` &) const

## Private Attributes

- `std::vector< cln::cl_N >` \* [current\\_vector](#)

## Static Private Attributes

- static std::vector< cln::cl\_N > \* `coeffs` = nullptr

## 6.84.1 Constructor & Destructor Documentation

### 6.84.1.1 lanczos\_coeffs()

```
GiNaC::lanczos_coeffs::lanczos_coeffs ( )
```

References `coeffs`.

## 6.84.2 Member Function Documentation

### 6.84.2.1 sufficiently\_accurate()

```
bool GiNaC::lanczos_coeffs::sufficiently_accurate (
    int digits )
```

References `coeffs`, and `current_vector`.

Referenced by `GiNaC::lgamma()`, and `GiNaC::tgamma()`.

### 6.84.2.2 get\_order()

```
int GiNaC::lanczos_coeffs::get_order ( ) const [inline]
```

References `current_vector`.

Referenced by `GiNaC::lgamma()`, and `GiNaC::tgamma()`.

### 6.84.2.3 calc\_lanczos\_A()

```
cln::cl_N GiNaC::lanczos_coeffs::calc_lanczos_A (
    const cln::cl_N & x ) const
```

References `current_vector`, and `x`.

Referenced by `GiNaC::lgamma()`, and `GiNaC::tgamma()`.

### 6.84.3 Member Data Documentation

#### 6.84.3.1 coeffs

```
std::vector< cln::cl_N > * GiNaC::lanczos_coeffs::coeffs = nullptr [static], [private]
```

Referenced by lanczos\_coeffs(), and sufficiently\_accurate().

#### 6.84.3.2 current\_vector

```
std::vector<cln::cl_N>* GiNaC::lanczos_coeffs::current_vector [private]
```

Referenced by calc\_lanczos\_A(), get\_order(), and sufficiently\_accurate().

The documentation for this class was generated from the following file:

- [numeric.cpp](#)

## 6.85 std::less< GiNaC::ptr< T > > Struct Template Reference

Specialization of std::less for ptr<T> to enable ordering of ptr<T> objects (e.g.

```
#include <ptr.h>
```

### Public Member Functions

- bool [operator\(\)](#) (const [GiNaC::ptr< T >](#) &lhs, const [GiNaC::ptr< T >](#) &rhs) const

#### 6.85.1 Detailed Description

```
template<class T>
struct std::less< GiNaC::ptr< T > >
```

Specialization of std::less for ptr<T> to enable ordering of ptr<T> objects (e.g.

for the use as std::map keys).

#### 6.85.2 Member Function Documentation

### 6.85.2.1 operator()

```
template<class T >
bool std::less< GiNaC::ptr< T > >::operator() (
    const GiNaC::ptr< T > & lhs,
    const GiNaC::ptr< T > & rhs ) const [inline]
```

References `GiNaC::lhs()`, and `GiNaC::rhs()`.

The documentation for this struct was generated from the following file:

- [ptr.h](#)

## 6.86 GiNaC::library\_init Class Reference

Helper class to initialize the library.

```
#include <ex.h>
```

### Public Member Functions

- [library\\_init\(\)](#)  
*Ctor of static initialization helpers.*
- [~library\\_init\(\)](#)  
*Dtor of static initialization helpers.*

### Static Private Member Functions

- static void [init\\_unarchivers\(\)](#)

### Static Private Attributes

- static int [count](#) = 0  
*How many static objects were created? Only the first one must create the static flyweights on the heap.*

### 6.86.1 Detailed Description

Helper class to initialize the library.

There must be one static object of this class in every object file that makes use of our flyweights in order to guarantee proper initialization. Hence we put it into this file which is included by every relevant file anyways. This is modeled after section 27.4.2.1.6 of the C++ standard, where `cout` and `friends` are set up.

See also

[utils.cpp](#)

## 6.86.2 Constructor & Destructor Documentation

### 6.86.2.1 library\_init()

```
GiNaC::library_init::library_init ( )
```

Ctor of static initialization helpers.

The first call to this is going to initialize the library, the others do nothing.

References `GiNaC::_ex0`, `GiNaC::_ex1`, `GiNaC::_ex10`, `GiNaC::_ex11`, `GiNaC::_ex12`, `GiNaC::_ex120`, `GiNaC::_ex15`, `GiNaC::_ex18`, `GiNaC::_ex1_2`, `GiNaC::_ex1_3`, `GiNaC::_ex1_4`, `GiNaC::_ex2`, `GiNaC::_ex20`, `GiNaC::_ex24`, `GiNaC::_ex25`, `GiNaC::_ex3`, `GiNaC::_ex30`, `GiNaC::_ex4`, `GiNaC::_ex48`, `GiNaC::_ex5`, `GiNaC::_ex6`, `GiNaC::_ex60`, `GiNaC::_ex7`, `GiNaC::_ex8`, `GiNaC::_ex9`, `GiNaC::_ex_1`, `GiNaC::_ex_10`, `GiNaC::_ex_11`, `GiNaC::_ex_12`, `GiNaC::_ex_120`, `GiNaC::_ex_15`, `GiNaC::_ex_18`, `GiNaC::_ex_1_2`, `GiNaC::_ex_1_3`, `GiNaC::_ex_1_4`, `GiNaC::_ex_2`, `GiNaC::_ex_20`, `GiNaC::_ex_24`, `GiNaC::_ex_25`, `GiNaC::_ex_3`, `GiNaC::_ex_30`, `GiNaC::_ex_4`, `GiNaC::_ex_48`, `GiNaC::_ex_5`, `GiNaC::_ex_6`, `GiNaC::_ex_60`, `GiNaC::_ex_7`, `GiNaC::_ex_8`, `GiNaC::_ex_9`, `GiNaC::_num0_bp`, `GiNaC::_num0_p`, `GiNaC::_num10_p`, `GiNaC::_num11_p`, `GiNaC::_num120_p`, `GiNaC::_num12_p`, `GiNaC::_num15_p`, `GiNaC::_num18_p`, `GiNaC::_num1_2_p`, `GiNaC::_num1_3_p`, `GiNaC::_num1_4_p`, `GiNaC::_num1_p`, `GiNaC::_num20_p`, `GiNaC::_num24_p`, `GiNaC::_num25_p`, `GiNaC::_num2_p`, `GiNaC::_num30_p`, `GiNaC::_num3_p`, `GiNaC::_num48_p`, `GiNaC::_num4_p`, `GiNaC::_num5_p`, `GiNaC::_num60_p`, `GiNaC::_num6_p`, `GiNaC::_num7_p`, `GiNaC::_num8_p`, `GiNaC::_num9_p`, `GiNaC::_num_10_p`, `GiNaC::_num_11_p`, `GiNaC::_num_120_p`, `GiNaC::_num_12_p`, `GiNaC::_num_15_p`, `GiNaC::_num_18_p`, `GiNaC::_num_1_2_p`, `GiNaC::_num_1_3_p`, `GiNaC::_num_1_4_p`, `GiNaC::_num_1_p`, `GiNaC::_num_20_p`, `GiNaC::_num_24_p`, `GiNaC::_num_25_p`, `GiNaC::_num_2_p`, `GiNaC::_num_30_p`, `GiNaC::_num_3_p`, `GiNaC::_num_48_p`, `GiNaC::_num_4_p`, `GiNaC::_num_5_p`, `GiNaC::_num_60_p`, `GiNaC::_num_6_p`, `GiNaC::_num_7_p`, `GiNaC::_num_8_p`, `GiNaC::_num_9_p`, and `count`.

### 6.86.2.2 ~library\_init()

```
GiNaC::library_init::~~library_init ( )
```

Dtor of static initialization helpers.

The last call to this is going to shut down the library, the others do nothing.

References `GiNaC::_ex0`, `GiNaC::_ex1`, `GiNaC::_ex10`, `GiNaC::_ex11`, `GiNaC::_ex12`, `GiNaC::_ex120`, `GiNaC::_ex15`, `GiNaC::_ex18`, `GiNaC::_ex1_2`, `GiNaC::_ex1_3`, `GiNaC::_ex1_4`, `GiNaC::_ex2`, `GiNaC::_ex20`, `GiNaC::_ex24`, `GiNaC::_ex25`, `GiNaC::_ex3`, `GiNaC::_ex30`, `GiNaC::_ex4`, `GiNaC::_ex48`, `GiNaC::_ex5`, `GiNaC::_ex6`, `GiNaC::_ex60`, `GiNaC::_ex7`, `GiNaC::_ex8`, `GiNaC::_ex9`, `GiNaC::_ex_1`, `GiNaC::_ex_10`, `GiNaC::_ex_11`, `GiNaC::_ex_12`, `GiNaC::_ex_120`, `GiNaC::_ex_15`, `GiNaC::_ex_18`, `GiNaC::_ex_1_2`, `GiNaC::_ex_1_3`, `GiNaC::_ex_1_4`, `GiNaC::_ex_2`, `GiNaC::_ex_20`, `GiNaC::_ex_24`, `GiNaC::_ex_25`, `GiNaC::_ex_3`, `GiNaC::_ex_30`, `GiNaC::_ex_4`, `GiNaC::_ex_48`, `GiNaC::_ex_5`, `GiNaC::_ex_6`, `GiNaC::_ex_60`, `GiNaC::_ex_7`, `GiNaC::_ex_8`, `GiNaC::_ex_9`, and `count`.

## 6.86.3 Member Function Documentation

### 6.86.3.1 `init_unarchivers()`

```
void GiNaC::library_init::init_unarchivers ( ) [static], [private]
```

## 6.86.4 Member Data Documentation

### 6.86.4.1 `count`

```
int GiNaC::library_init::count = 0 [static], [private]
```

How many static objects were created? Only the first one must create the static flyweights on the heap.

Referenced by `library_init()`, and `~library_init()`.

The documentation for this class was generated from the following files:

- [ex.h](#)
- [utils.cpp](#)

## 6.87 `GiNaC::make_flat_inserter` Class Reference

Class to handle the renaming of dummy indices.

```
#include <expairseq.h>
```

### Public Member Functions

- [make\\_flat\\_inserter](#) (const [epvector](#) &epv, bool b)
- [make\\_flat\\_inserter](#) (const [exvector](#) &v, bool b)
- [ex\\_handle\\_factor](#) (const [ex](#) &x, const [ex](#) &coeff)

### Private Member Functions

- void [combine\\_indices](#) (const [exvector](#) &dummies\_of\_factor)

### Private Attributes

- bool [do\\_renaming](#)
- [exvector](#) [used\\_indices](#)



## 6.87.1 Detailed Description

Class to handle the renaming of dummy indices.

It holds a vector of indices that are being used in the expression so far. If the same index occurs again as a dummy index in a factor, it is to be renamed. Unless dummy index renaming was switched off, of course ;-).

## 6.87.2 Constructor & Destructor Documentation

### 6.87.2.1 make\_flat\_inserter() [1/2]

```
GiNaC::make_flat_inserter::make_flat_inserter (
    const epvector & epv,
    bool b ) [inline]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [combine\\_indices\(\)](#), and [do\\_renaming](#).

### 6.87.2.2 make\_flat\_inserter() [2/2]

```
GiNaC::make_flat_inserter::make_flat_inserter (
    const exvector & v,
    bool b ) [inline]
```

References [combine\\_indices\(\)](#), and [do\\_renaming](#).

## 6.87.3 Member Function Documentation

### 6.87.3.1 handle\_factor()

```
ex GiNaC::make_flat_inserter::handle_factor (
    const ex & x,
    const ex & coeff ) [inline]
```

References [GiNaC::coeff\(\)](#), [combine\\_indices\(\)](#), [do\\_renaming](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::is\\_ex\\_equal\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [used\\_indices](#), and [x](#).

Referenced by [GiNaC::ncmul::eval\(\)](#).

### 6.87.3.2 combine\_indices()

```
void GiNaC::make_flat_inserter::combine_indices (
    const exvector & dummies_of_factor ) [inline], [private]
```

References used\_indices.

Referenced by handle\_factor(), and make\_flat\_inserter().

## 6.87.4 Member Data Documentation

### 6.87.4.1 do\_renaming

```
bool GiNaC::make_flat_inserter::do_renaming [private]
```

Referenced by handle\_factor(), and make\_flat\_inserter().

### 6.87.4.2 used\_indices

```
exvector GiNaC::make_flat_inserter::used_indices [private]
```

Referenced by combine\_indices(), and handle\_factor().

The documentation for this class was generated from the following file:

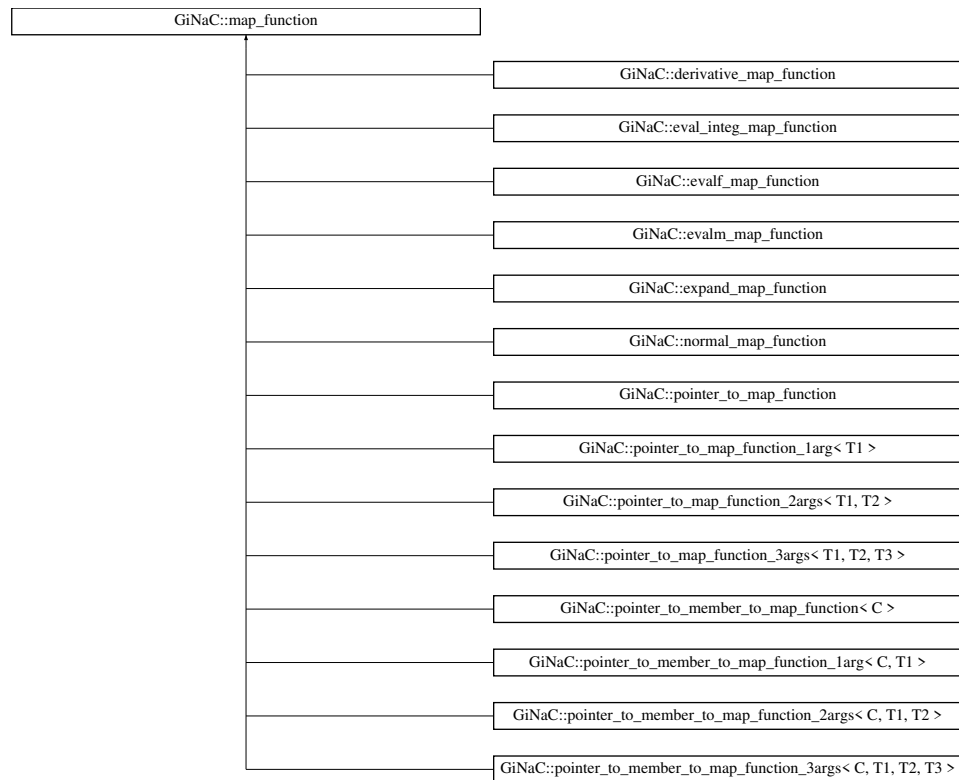
- [expairseq.h](#)

## 6.88 GiNaC::map\_function Struct Reference

Function object for map().

```
#include <basic.h>
```

Inheritance diagram for GiNaC::map\_function:



## Public Types

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## Public Member Functions

- virtual [~map\\_function](#) ()
- virtual [ex](#) [operator](#)() (const [ex](#) &*e*)=0

### 6.88.1 Detailed Description

Function object for `map()`.

### 6.88.2 Member Typedef Documentation

#### 6.88.2.1 [argument\\_type](#)

```
typedef const ex& GiNaC::map\_function::argument\_type
```

### 6.88.2.2 result\_type

```
typedef ex GiNaC::map\_function::result\_type
```

## 6.88.3 Constructor & Destructor Documentation

### 6.88.3.1 ~map\_function()

```
virtual GiNaC::map\_function::~~map\_function ( ) \[inline\], \[virtual\]
```

## 6.88.4 Member Function Documentation

### 6.88.4.1 operator>()

```
virtual ex GiNaC::map\_function::operator() (
    const ex & e ) \[pure virtual\]
```

Implemented in [GiNaC::normal\\_map\\_function](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >](#), [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >](#), [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >](#), [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >](#), [GiNaC::pointer\\_to\\_map\\_function](#), [GiNaC::expand\\_map\\_function](#), [GiNaC::derivative\\_map\\_function](#), [GiNaC::eval\\_integ\\_map\\_function](#), [GiNaC::evalm\\_map\\_function](#), and [GiNaC::evalf\\_map\\_function](#).

The documentation for this struct was generated from the following file:

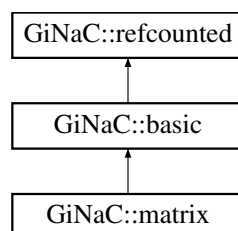
- [basic.h](#)

## 6.89 GiNaC::matrix Class Reference

Symbolic matrices.

```
#include <matrix.h>
```

Inheritance diagram for [GiNaC::matrix](#):



## Public Member Functions

- [matrix](#) (unsigned [r](#), unsigned [c](#))  
*Very common ctor.*
- [matrix](#) (unsigned [r](#), unsigned [c](#), const [lst](#) &l)  
*Construct matrix from (flat) list of elements.*
- [matrix](#) (std::initializer\_list< std::initializer\_list< [ex](#) >> l)  
*Construct a matrix from an 2 dimensional initializer list.*
- [size\\_t nops](#) () const override  
*nops is defined to be rows x columns.*
- [ex op](#) (size\_t i) const override  
*returns matrix entry at position (i/col, icol).*
- [ex & let\\_op](#) (size\_t i) override  
*returns writable matrix entry at position (i/col, icol).*
- [ex evalm](#) () const override  
*Evaluate sums, products and integer powers of matrices.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [ex eval\\_indexed](#) (const [basic](#) &i) const override  
*Automatic symbolic evaluation of an indexed matrix.*
- [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const override  
*Sum of two indexed matrices.*
- [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const override  
*Product of an indexed matrix with a number.*
- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of an indexed matrix with something else.*
- [ex conjugate](#) () const override  
*Complex conjugate every matrix entry.*
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*
- unsigned [rows](#) () const  
*Get number of rows.*
- unsigned [cols](#) () const  
*Get number of columns.*
- [matrix add](#) (const [matrix](#) &other) const  
*Sum of matrices.*
- [matrix sub](#) (const [matrix](#) &other) const  
*Difference of matrices.*
- [matrix mul](#) (const [matrix](#) &other) const  
*Product of matrices.*
- [matrix mul](#) (const [numeric](#) &other) const  
*Product of matrix and scalar.*
- [matrix mul\\_scalar](#) (const [ex](#) &other) const  
*Product of matrix and scalar expression.*
- [matrix pow](#) (const [ex](#) &expn) const  
*Power of a matrix.*
- const [ex & operator\(\)](#) (unsigned ro, unsigned co) const

- operator()* to access elements for reading.
- **ex** & **operator()** (unsigned ro, unsigned co)
- operator()* to access elements for writing.
- **matrix** & **set** (unsigned ro, unsigned co, const **ex** &value)
- **matrix transpose** () const
- Transposed of an  $m \times n$  matrix, producing a new  $n \times m$  matrix object that represents the transposed.*
- **ex determinant** (unsigned algo=**determinant\_algo::automatic**) const
- Determinant of square matrix.*
- **ex trace** () const
- Trace of a matrix.*
- **ex charpoly** (const **ex** &lambda) const
- Characteristic Polynomial.*
- **matrix inverse** () const
- Inverse of this matrix, with automatic algorithm selection.*
- **matrix inverse** (unsigned algo) const
- Inverse of this matrix.*
- **matrix solve** (const **matrix** &vars, const **matrix** &rhs, unsigned algo=**solve\_algo::automatic**) const
- Solve a linear system consisting of a  $m \times n$  matrix and a  $m \times p$  right hand side by applying an elimination scheme to the augmented matrix.*
- unsigned **rank** () const
- Compute the rank of this matrix.*
- unsigned **rank** (unsigned **solve\_algo**) const
- Compute the rank of this matrix using the given algorithm, which should be a member of enum **solve\_algo**.*
- bool **is\_zero\_matrix** () const
- Function to check that all elements of the matrix are zero.*

## Protected Member Functions

- **matrix** (unsigned **r**, unsigned **c**, const **exvector** &m2)
- Ctor from representation, for internal use only.*
- **matrix** (unsigned **r**, unsigned **c**, **exvector** &&m2)
- bool **match\_same\_type** (const **basic** &other) const override
- Returns true if the attributes of two objects are similar enough for a match.*
- unsigned **return\_type** () const override
- **ex determinant\_minor** () const
- Recursive determinant for small matrices having at least one symbolic entry.*
- std::vector< unsigned > **echelon\_form** (unsigned algo, int **n**)
- int **gauss\_elimination** (const bool det=false)
- Perform the steps of an ordinary Gaussian elimination to bring the  $m \times n$  matrix into an upper echelon form.*
- int **division\_free\_elimination** (const bool det=false)
- Perform the steps of division free elimination to bring the  $m \times n$  matrix into an upper echelon form.*
- int **fraction\_free\_elimination** (const bool det=false)
- Perform the steps of Bareiss' one-step fraction free elimination to bring the matrix into an upper echelon form.*
- std::vector< unsigned > **markowitz\_elimination** (unsigned **n**)
- int **pivot** (unsigned ro, unsigned co, bool symbolic=true)
- Partial pivoting method for matrix elimination schemes.*
- void **print\_elements** (const **print\_context** &c, const char \*row\_start, const char \*row\_end, const char \*row↔\_sep, const char \*col\_sep) const
- void **do\_print** (const **print\_context** &c, unsigned level) const
- void **do\_print\_latex** (const **print\_latex** &c, unsigned level) const
- void **do\_print\_python\_repr** (const **print\_python\_repr** &c, unsigned level) const

## Protected Attributes

- unsigned [row](#)  
*number of rows*
- unsigned [col](#)  
*number of columns*
- [exvector](#) [m](#)  
*representation (cols indexed first)*

### 6.89.1 Detailed Description

Symbolic matrices.

### 6.89.2 Constructor & Destructor Documentation

#### 6.89.2.1 [matrix\(\)](#) [1/5]

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c )
```

Very common ctor.

Initializes to r x c-dimensional zero-matrix.

#### Parameters

<i>r</i>	number of rows
<i>c</i>	number of cols

References [GiNaC::status\\_flags::not\\_shareable](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [add\(\)](#), [conjugate\(\)](#), [determinant\(\)](#), [imag\\_part\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [real\\_part\(\)](#), [sub\(\)](#), [subs\(\)](#), and [transpose\(\)](#).

#### 6.89.2.2 [matrix\(\)](#) [2/5]

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c,
    const lst & l )
```

Construct matrix from (flat) list of elements.

If the list has fewer elements than the matrix, the remaining matrix elements are set to zero. If the list has more elements than the matrix, the excessive elements are thrown away.

References [c](#), [m](#), [GiNaC::status\\_flags::not\\_shareable](#), [r](#), [GiNaC::basic::setflag\(\)](#), and [x](#).

**6.89.2.3 matrix()** [3/5]

```
GiNaC::matrix::matrix (
    std::initializer_list< std::initializer_list< ex >> l )
```

Construct a matrix from an 2 dimensional initializer list.

Throws an exception if some row has a different length than all the others.

References `c`, `col`, `m`, `GiNaC::status_flags::not_shareable`, `r`, `row`, and `GiNaC::basic::setflag()`.

**6.89.2.4 matrix()** [4/5]

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c,
    const exvector & m2 ) [protected]
```

Ctor from representation, for internal use only.

References `GiNaC::status_flags::not_shareable`, and `GiNaC::basic::setflag()`.

**6.89.2.5 matrix()** [5/5]

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c,
    exvector && m2 ) [protected]
```

References `GiNaC::status_flags::not_shareable`, and `GiNaC::basic::setflag()`.

**6.89.3 Member Function Documentation****6.89.3.1 nops()**

```
size_t GiNaC::matrix::nops ( ) const [override], [virtual]
```

`nops` is defined to be rows x columns.

Reimplemented from [GiNaC::basic](#).

References `col`, and `row`.

Referenced by `add_indexed()`, `let_op()`, `op()`, and `scalar_mul_indexed()`.



### 6.89.3.2 op()

```
ex GiNaC::matrix::op (
    size_t i ) const [override], [virtual]
```

returns matrix entry at position (i/col, icol).

Reimplemented from [GiNaC::basic](#).

References GINAC\_ASSERT, m, and nops().

Referenced by add\_indexed(), and scalar\_mul\_indexed().

### 6.89.3.3 let\_op()

```
ex & GiNaC::matrix::let_op (
    size_t i ) [override], [virtual]
```

returns writable matrix entry at position (i/col, icol).

Reimplemented from [GiNaC::basic](#).

References GiNaC::basic::ensure\_if\_modifiable(), GINAC\_ASSERT, m, and nops().

### 6.89.3.4 evalm()

```
ex GiNaC::matrix::evalm ( ) const [inline], [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

### 6.89.3.5 subs()

```
ex GiNaC::matrix::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References c, col, m, matrix(), options, r, and row.

Referenced by fraction\_free\_elimination().

### 6.89.3.6 eval\_indexed()

```
ex GiNaC::matrix::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed matrix.

Reimplemented from [GiNaC::basic](#).

References `col`, `GiNaC::idx::get_dim()`, `GiNaC::idx::get_value()`, `GINAC_ASSERT`, `GiNaC::basic::hold()`, `GiNaC::is_dummy_pair()`, `GiNaC::ex::is_equal()`, `GiNaC::info_flags::nonnegint`, `GiNaC::basic::nops()`, `GiNaC::basic::op()`, `row`, `GiNaC::to_int()`, and `trace()`.

### 6.89.3.7 add\_indexed()

```
ex GiNaC::matrix::add_indexed (
    const ex & self,
    const ex & other ) const [override], [virtual]
```

Sum of two indexed matrices.

Reimplemented from [GiNaC::basic](#).

References `GINAC_ASSERT`, `GiNaC::ex::is_equal()`, `nops()`, `GiNaC::ex::nops()`, `op()`, and `GiNaC::ex::op()`.

### 6.89.3.8 scalar\_mul\_indexed()

```
ex GiNaC::matrix::scalar_mul_indexed (
    const ex & self,
    const numeric & other ) const [override], [virtual]
```

Product of an indexed matrix with a number.

Reimplemented from [GiNaC::basic](#).

References `GINAC_ASSERT`, `mul()`, `nops()`, `op()`, and `GiNaC::ex::op()`.

### 6.89.3.9 contract\_with()

```
bool GiNaC::matrix::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed matrix with something else.

Reimplemented from [GiNaC::basic](#).

References `GiNaC::_ex1`, `GINAC_ASSERT`, `GiNaC::is_dummy_pair()`, `mul()`, and `transpose()`.

#### 6.89.3.10 conjugate()

```
ex GiNaC::matrix::conjugate ( ) const [override], [virtual]
```

Complex conjugate every matrix entry.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [col](#), [m](#), [matrix\(\)](#), [row](#), and [x](#).

#### 6.89.3.11 real\_part()

```
ex GiNaC::matrix::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [col](#), [m](#), [matrix\(\)](#), and [row](#).

#### 6.89.3.12 imag\_part()

```
ex GiNaC::matrix::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [col](#), [m](#), [matrix\(\)](#), and [row](#).

#### 6.89.3.13 archive()

```
void GiNaC::matrix::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [col](#), [m](#), [n](#), and [row](#).

#### 6.89.3.14 read\_archive()

```
void GiNaC::matrix::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References col, m, n, and row.

#### 6.89.3.15 match\_same\_type()

```
bool GiNaC::matrix::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References col, cols(), GINAC\_ASSERT, row, and rows().

#### 6.89.3.16 return\_type()

```
unsigned GiNaC::matrix::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#).

## 6.89.3.17 rows()

```
unsigned GiNaC::matrix::rows ( ) const [inline]
```

Get number of rows.

References row.

Referenced by `division_free_elimination()`, `fraction_free_elimination()`, `gauss_elimination()`, `GiNaC::lsolve()`, `match_same_type()`, `mul()`, `solve()`, and `transpose()`.

## 6.89.3.18 cols()

```
unsigned GiNaC::matrix::cols ( ) const [inline]
```

Get number of columns.

References col.

Referenced by `determinant_minor()`, `division_free_elimination()`, `fraction_free_elimination()`, `gauss_elimination()`, `GiNaC::lsolve()`, `match_same_type()`, `mul()`, `solve()`, and `transpose()`.

## 6.89.3.19 add()

```
matrix GiNaC::matrix::add (
    const matrix & other ) const
```

Sum of matrices.

## Exceptions

<i>logic_error</i>	(incompatible matrices)
--------------------	-------------------------

References col, m, `matrix()`, and row.

Referenced by `GiNaC::add::evalm()`.

## 6.89.3.20 sub()

```
matrix GiNaC::matrix::sub (
    const matrix & other ) const
```

Difference of matrices.

## Exceptions

<i>logic_error</i>	(incompatible matrices)
--------------------	-------------------------

References col, m, matrix(), and row.

## 6.89.3.21 mul() [1/2]

```
matrix GiNaC::matrix::mul (
    const matrix & other ) const
```

Product of matrices.

## Exceptions

<i>logic_error</i>	(incompatible matrices)
--------------------	-------------------------

References c, col, cols(), GiNaC::is\_zero(), m, matrix(), row, and rows().

Referenced by charpoly(), contract\_with(), GiNaC::ncmul::evalm(), pow(), and scalar\_mul\_indexed().

## 6.89.3.22 mul() [2/2]

```
matrix GiNaC::matrix::mul (
    const numeric & other ) const
```

Product of matrix and scalar.

References c, col, m, matrix(), r, and row.

## 6.89.3.23 mul\_scalar()

```
matrix GiNaC::matrix::mul_scalar (
    const ex & other ) const
```

Product of matrix and scalar expression.

References c, col, GiNaC::return\_types::commutative, m, matrix(), r, GiNaC::ex::return\_type(), and row.

## 6.89.3.24 pow()

```
matrix GiNaC::matrix::pow (
    const ex & expn ) const
```

Power of a matrix.

Currently handles integer exponents only.

References GiNaC::\_ex1, GiNaC::\_num1\_p, GiNaC::\_num2\_p, col, GiNaC::ex::info(), GiNaC::info\_flags::integer, inverse(), GiNaC::numeric::is\_odd(), GiNaC::numeric::is\_zero(), mul(), GiNaC::numeric::mul(), GiNaC::info\_flags::negative, r, and row.

## 6.89.3.25 operator() [1/2]

```
const ex & GiNaC::matrix::operator() (
    unsigned ro,
    unsigned co ) const
```

operator() to access elements for reading.

## Parameters

<i>ro</i>	row of element
<i>co</i>	column of element

## Exceptions

<i>range_error</i>	(index out of range)
--------------------	----------------------

References col, m, and row.

## 6.89.3.26 operator() [2/2]

```
ex & GiNaC::matrix::operator() (
    unsigned ro,
    unsigned co )
```

operator() to access elements for writing.

## Parameters

<i>ro</i>	row of element
<i>co</i>	column of element

## Exceptions

<code>range_error</code>	(index out of range)
--------------------------	----------------------

References `col`, `GiNaC::basic::ensure_if_modifiable()`, `m`, and `row`.

6.89.3.27 `set()`

```
matrix& GiNaC::matrix::set (
    unsigned ro,
    unsigned co,
    const ex & value ) [inline]
```

References `value`.

6.89.3.28 `transpose()`

```
matrix GiNaC::matrix::transpose ( ) const
```

Transposed of an  $m \times n$  matrix, producing a new  $n \times m$  matrix object that represents the transposed.

References `c`, `cols()`, `m`, `matrix()`, `r`, and `rows()`.

Referenced by `contract_with()`.

6.89.3.29 `determinant()`

```
ex GiNaC::matrix::determinant (
    unsigned algo = determinant_algo::automatic ) const
```

Determinant of square matrix.

This routine doesn't actually calculate the determinant, it only implements some heuristics about which algorithm to run. If all the elements of the matrix are elements of an integral domain the determinant is also in that integral domain and the result is expanded only. If one or more elements are from a quotient field the determinant is usually also in that quotient field and the result is normalized before it is returned. This implies that the determinant of the symbolic 2x2 matrix  $\begin{bmatrix} a/(a-b) & 1 \\ b/(a-b) & 1 \end{bmatrix}$  is returned as unity. (In this respect, it behaves like MapleV and unlike Mathematica.)

## Parameters

<code>algo</code>	allows to chose an algorithm
-------------------	------------------------------



**Returns**

the determinant as a new expression

**Exceptions**

<i>logic_error</i>	(matrix not square)
--------------------	---------------------

**See also**

[determinant\\_algo](#)

References `GiNaC::ex0`, `GiNaC::determinant_algo::automatic`, `GiNaC::determinant_algo::bareiss`, `c`, `col`, `GiNaC::info_flags::crational_polynomial`, `determinant_minor()`, `GiNaC::determinant_algo::divfree`, `division_free_elimination()`, `fraction_free_elimination()`, `GiNaC::determinant_algo::gauss`, `gauss_elimination()`, `GINAC_ASSERT`, `GiNaC::ex::info()`, `GiNaC::ex::is_zero()`, `GiNaC::is_zero()`, `GiNaC::determinant_algo::laplace`, `m`, `matrix()`, `GiNaC::ex::normal()`, `GiNaC::info_flags::numeric`, `GiNaC::permutation_sign()`, `r`, `GiNaC::info_flags::rational_function`, and `row`.

Referenced by `charpoly()`.

**6.89.3.30 trace()**

```
ex GiNaC::matrix::trace ( ) const
```

Trace of a matrix.

The result is normalized if it is in some quotient field and expanded only otherwise. This implies that the trace of the symbolic 2x2 matrix  $\begin{bmatrix} a/(a-b), x \\ y, b/(b-a) \end{bmatrix}$  is recognized to be unity.

**Returns**

the sum of diagonal elements

**Exceptions**

<i>logic_error</i>	(matrix not square)
--------------------	---------------------

References `col`, `GiNaC::info_flags::crational_polynomial`, `GiNaC::ex::expand()`, `GiNaC::ex::info()`, `m`, `GiNaC::ex::normal()`, `r`, `GiNaC::info_flags::rational_function`, and `row`.

Referenced by `charpoly()`, and `eval_indexed()`.

**6.89.3.31 charpoly()**

```
ex GiNaC::matrix::charpoly (
    const ex & lambda ) const
```

Characteristic Polynomial.

Following mathematica notation the characteristic polynomial of a matrix  $M$  is defined as the determinant of  $(M - \lambda * 1)$  where  $1$  stands for the unit matrix of the same dimension as  $M$ . Note that some CASs define it with a sign inside the determinant which gives rise to an overall sign if the dimension is odd. This method returns the characteristic polynomial collected in powers of  $\lambda$  as a new expression.

#### Returns

characteristic polynomial as new expression

#### Exceptions

<code>logic_error</code>	(matrix not square)
--------------------------	---------------------

#### See also

[matrix::determinant\(\)](#)

References `c`, `col`, `GiNaC::ex::collect()`, `determinant()`, `GiNaC::basic::ex`, `m`, `mul()`, `GiNaC::info_flags::numeric`, `poly`, `r`, `row`, and `trace()`.

#### 6.89.3.32 `inverse()` [1/2]

```
matrix GiNaC::matrix::inverse ( ) const
```

Inverse of this matrix, with automatic algorithm selection.

References `GiNaC::solve_algo::automatic`.

Referenced by `pow()`.

#### 6.89.3.33 `inverse()` [2/2]

```
matrix GiNaC::matrix::inverse (
    unsigned algo ) const
```

Inverse of this matrix.

#### Parameters

<code>algo</code>	selects the algorithm (one of <a href="#">solve_algo</a> )
-------------------	--

**Returns**

the inverted matrix

**Exceptions**

<i>logic_error</i>	(matrix not square)
<i>runtime_error</i>	(singular matrix)

References `GiNaC::_ex1`, `c`, `col`, `r`, `row`, and `solve()`.

**6.89.3.34 solve()**

```
matrix GiNaC::matrix::solve (
    const matrix & vars,
    const matrix & rhs,
    unsigned algo = solve_algo::automatic ) const
```

Solve a linear system consisting of a  $m \times n$  matrix and a  $m \times p$  right hand side by applying an elimination scheme to the augmented matrix.

**Parameters**

<i>vars</i>	$n \times p$ matrix, all elements must be symbols
<i>rhs</i>	$m \times p$ matrix
<i>algo</i>	selects the solving algorithm

**Returns**

$n \times p$  solution matrix

**Exceptions**

<i>logic_error</i>	(incompatible matrices)
<i>invalid_argument</i>	(1st argument must be matrix of symbols)
<i>runtime_error</i>	(inconsistent linear system)

**See also**

[solve\\_algo](#)

References `c`, `cols()`, `echelon_form()`, `GiNaC::basic::info()`, `m`, `n`, `GiNaC::basic::normal()`, `r`, `GiNaC::rhs()`, `rows()`, and `GiNaC::info_flags::symbol`.

Referenced by `inverse()`, `GiNaC::lsolve()`, and `GiNaC::sqrfree_parfrac()`.

**6.89.3.35 rank()** [1/2]

```
unsigned GiNaC::matrix::rank ( ) const
```

Compute the rank of this matrix.

References `GiNaC::solve_algo::automatic`.

**6.89.3.36 rank()** [2/2]

```
unsigned GiNaC::matrix::rank (
    unsigned solve_algo ) const
```

Compute the rank of this matrix using the given algorithm, which should be a member of enum `solve_algo`.

References `col`, `echelon_form()`, `GINAC_ASSERT`, `m`, `r`, and `row`.

**6.89.3.37 is\_zero\_matrix()**

```
bool GiNaC::matrix::is_zero_matrix ( ) const
```

Function to check that all elements of the matrix are zero.

References `m`.

**6.89.3.38 determinant\_minor()**

```
ex GiNaC::matrix::determinant_minor ( ) const [protected]
```

Recursive determinant for small matrices having at least one symbolic entry.

The basic algorithm, known as Laplace-expansion, is enhanced by some bookkeeping to avoid calculation of the same submatrices ("minors") more than once. According to W.M.Gentleman and S.C.Johnson this algorithm is better than elimination schemes for matrices of sparse multivariate polynomials and also for matrices of dense univariate polynomials if the matrix' dimension is larger than 7.

**Returns**

the determinant as a new expression (in expanded form)

**See also**

[matrix::determinant\(\)](#)

References `GiNaC::_ex0`, `GiNaC::_ex1`, `c`, `cols()`, `GiNaC::ex::expand()`, `GiNaC::ex::is_zero()`, `GiNaC::is_zero()`, `m`, `n`, and `r`.

Referenced by `determinant()`.

## 6.89.3.39 echelon\_form()

```
std::vector< unsigned > GiNaC::matrix::echelon_form (
    unsigned algo,
    int n ) [protected]
```

References GiNaC::solve\_algo::automatic, GiNaC::solve\_algo::bareiss, c, col, GiNaC::solve\_algo::divfree, division\_free\_elimination(), fraction\_free\_elimination(), GiNaC::solve\_algo::gauss, gauss\_elimination(), m, GiNaC::solve\_algo::markowitz, markowitz\_elimination(), n, GiNaC::info\_flags::numeric, r, and row.

Referenced by rank(), and solve().

## 6.89.3.40 gauss\_elimination()

```
int GiNaC::matrix::gauss_elimination (
    const bool det = false ) [protected]
```

Perform the steps of an ordinary Gaussian elimination to bring the m x n matrix into an upper echelon form.

The algorithm is ok for matrices with numeric coefficients but quite unsuited for symbolic matrices.

## Parameters

<i>det</i>	may be set to true to save a lot of space if one is only interested in the diagonal elements (i.e. for calculating determinants). The others are set to zero in this case.
------------	--

## Returns

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References GiNaC::ex0, c, cols(), GiNaC::basic::ensure\_if\_modifiable(), GINAC\_ASSERT, GiNaC::basic::info(), GiNaC::is\_zero(), m, n, GiNaC::ex::normal(), GiNaC::info\_flags::numeric, pivot(), r, and rows().

Referenced by determinant(), and echelon\_form().

## 6.89.3.41 division\_free\_elimination()

```
int GiNaC::matrix::division_free_elimination (
    const bool det = false ) [protected]
```

Perform the steps of division free elimination to bring the m x n matrix into an upper echelon form.

## Parameters

<i>det</i>	may be set to true to save a lot of space if one is only interested in the diagonal elements (i.e. for calculating determinants). The others are set to zero in this case.
------------	--

**Returns**

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References `GiNaC::_ex0`, `c`, `cols()`, `GiNaC::basic::ensure_if_modifiable()`, `GINAC_ASSERT`, `m`, `n`, `pivot()`, `r`, and `rows()`.

Referenced by `determinant()`, and `echelon_form()`.

**6.89.3.42 fraction\_free\_elimination()**

```
int GiNaC::matrix::fraction_free_elimination (
    const bool det = false ) [protected]
```

Perform the steps of Bareiss' one-step fraction free elimination to bring the matrix into an upper echelon form.

Fraction free elimination means that divide is used straightforwardly, without computing GCDs first. This is possible, since we know the divisor at each step.

**Parameters**

<i>det</i>	may be set to true to save a lot of space if one is only interested in the last element (i.e. for calculating determinants). The others are set to zero in this case.
------------	---

**Returns**

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References `GiNaC::_ex0`, `GiNaC::_ex1`, `c`, `cols()`, `GiNaC::divide()`, `GiNaC::basic::ensure_if_modifiable()`, `GiNaC::basic::expand()`, `GINAC_ASSERT`, `GiNaC::is_zero()`, `m`, `n`, `GiNaC::subs_options::no_pattern`, `GiNaC::ex::normal()`, `GiNaC::ex::numer_denom()`, `GiNaC::ex::op()`, `r`, `rows()`, `subs()`, and `GiNaC::ex::to_rational()`.

Referenced by `determinant()`, and `echelon_form()`.

**6.89.3.43 markowitz\_elimination()**

```
std::vector< unsigned > GiNaC::matrix::markowitz_elimination (
    unsigned n ) [protected]
```

References `GiNaC::_ex0`, `c`, `col`, `GINAC_ASSERT`, `GiNaC::ex::is_zero()`, `GiNaC::is_zero()`, `k`, `m`, `n`, `GiNaC::basic::normal()`, `r`, `row`, and `std::swap()`.

Referenced by `echelon_form()`.

## 6.89.3.44 pivot()

```
int GiNaC::matrix::pivot (
    unsigned ro,
    unsigned co,
    bool symbolic = true ) [protected]
```

Partial pivoting method for matrix elimination schemes.

Usual pivoting (symbolic==false) returns the index to the element with the largest absolute value in column ro and swaps the current row with the one where the element was found. With (symbolic==true) it does the same thing with the first non-zero element.

## Parameters

<i>ro</i>	is the row from where to begin
<i>co</i>	is the column to be inspected
<i>symbolic</i>	signal if we want the first non-zero element to be pivoted (true) or the one with the largest absolute value (false).

## Returns

0 if no interchange occurred, -1 if all are zero (usually signaling a degeneracy) and positive integer k means that rows ro and k were swapped.

References GiNaC::abs(), c, col, GiNaC::basic::ensure\_if\_modifiable(), GiNaC::basic::expand(), GINAC\_ASSERT, GiNaC::numeric::is\_zero(), GiNaC::is\_zero(), k, m, row, and GiNaC::swap().

Referenced by division\_free\_elimination(), and gauss\_elimination().

## 6.89.3.45 print\_elements()

```
void GiNaC::matrix::print_elements (
    const print_context & c,
    const char * row_start,
    const char * row_end,
    const char * row_sep,
    const char * col_sep ) const [protected]
```

References c, col, m, and row.

Referenced by do\_print(), do\_print\_latex(), and do\_print\_python\_repr().

## 6.89.3.46 do\_print()

```
void GiNaC::matrix::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References c, and print\_elements().

**6.89.3.47 do\_print\_latex()**

```
void GiNaC::matrix::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), [col](#), and [print\\_elements\(\)](#).

**6.89.3.48 do\_print\_python\_repr()**

```
void GiNaC::matrix::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_elements\(\)](#).

**6.89.4 Member Data Documentation****6.89.4.1 row**

```
unsigned GiNaC::matrix::row [protected]
```

number of rows

Referenced by [add\(\)](#), [archive\(\)](#), [charpoly\(\)](#), [conjugate\(\)](#), [determinant\(\)](#), [echelon\\_form\(\)](#), [eval\\_indexed\(\)](#), [imag\\_part\(\)](#), [inverse\(\)](#), [markowitz\\_elimination\(\)](#), [match\\_same\\_type\(\)](#), [matrix\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [nops\(\)](#), [operator\(\)](#), [pivot\(\)](#), [pow\(\)](#), [print\\_elements\(\)](#), [rank\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [rows\(\)](#), [sub\(\)](#), [subs\(\)](#), and [trace\(\)](#).

**6.89.4.2 col**

```
unsigned GiNaC::matrix::col [protected]
```

number of columns

Referenced by [add\(\)](#), [archive\(\)](#), [charpoly\(\)](#), [cols\(\)](#), [conjugate\(\)](#), [determinant\(\)](#), [do\\_print\\_latex\(\)](#), [echelon\\_form\(\)](#), [eval\\_indexed\(\)](#), [imag\\_part\(\)](#), [inverse\(\)](#), [markowitz\\_elimination\(\)](#), [match\\_same\\_type\(\)](#), [matrix\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [nops\(\)](#), [operator\(\)](#), [pivot\(\)](#), [pow\(\)](#), [print\\_elements\(\)](#), [rank\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [sub\(\)](#), [subs\(\)](#), and [trace\(\)](#).



## 6.89.4.3 m

`exvector` GiNaC::matrix::m [protected]

representation (cols indexed first)

Referenced by `add()`, `archive()`, `charpoly()`, `conjugate()`, `determinant()`, `determinant_minor()`, `division_free_`  
`elimination()`, `echelon_form()`, `fraction_free_elimination()`, `gauss_elimination()`, `imag_part()`, `is_zero_matrix()`, `let_`  
`_op()`, `markowitz_elimination()`, `matrix()`, `mul()`, `mul_scalar()`, `op()`, `operator()`, `pivot()`, `print_elements()`, `rank()`,  
`read_archive()`, `real_part()`, `solve()`, `sub()`, `subs()`, `trace()`, and `transpose()`.

The documentation for this class was generated from the following files:

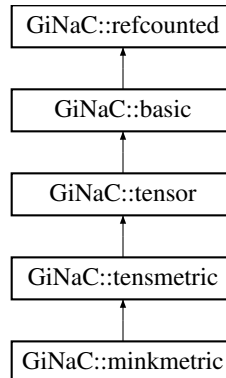
- [matrix.h](#)
- [matrix.cpp](#)

## 6.90 GiNaC::minkmetric Class Reference

This class represents a Minkowski metric tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::minkmetric:



### Public Member Functions

- `minkmetric` (bool `pos_sig`)  
*Construct Lorentz metric tensor with given signature.*
- bool `info` (unsigned int) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed Lorentz metric tensor.*
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &`syms`) override  
*Read (a.k.a.*

## Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

## Private Attributes

- bool [pos\\_sig](#)  
*If true, the metric is  $\text{diag}(-1, 1, 1, \dots)$ .*

## Additional Inherited Members

### 6.90.1 Detailed Description

This class represents a Minkowski metric tensor.

It has all the properties of a metric tensor and is (as a matrix) equal to  $\text{diag}(1, -1, -1, \dots)$  or  $\text{diag}(-1, 1, 1, \dots)$ .

### 6.90.2 Constructor & Destructor Documentation

#### 6.90.2.1 [minkmetric\(\)](#)

```
GiNaC::minkmetric::minkmetric (
    bool pos_sig )
```

Construct Lorentz metric tensor with given signature.

### 6.90.3 Member Function Documentation

#### 6.90.3.1 [info\(\)](#)

```
bool GiNaC::minkmetric::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::info\\_flags::real](#).

## 6.90.3.2 eval\_indexed()

```
ex GiNaC::minkmetric::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed Lorentz metric tensor.

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [pos\\_sig](#), and [GiNaC::to\\_int\(\)](#).

## 6.90.3.3 archive()

```
void GiNaC::minkmetric::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), and [pos\\_sig](#).

## 6.90.3.4 read\_archive()

```
void GiNaC::minkmetric::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), and [pos\\_sig](#).

## 6.90.3.5 return\_type()

```
unsigned GiNaC::minkmetric::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::return\\_types::commutative](#).

### 6.90.3.6 do\_print()

```
void GiNaC::minkmetric::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

### 6.90.3.7 do\_print\_latex()

```
void GiNaC::minkmetric::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

## 6.90.4 Member Data Documentation

### 6.90.4.1 pos\_sig

```
bool GiNaC::minkmetric::pos_sig [private]
```

If true, the metric is  $\text{diag}(-1,1,1\dots)$ .

Otherwise it is  $\text{diag}(1,-1,-1,\dots)$ .

Referenced by `archive()`, `eval_indexed()`, and `read_archive()`.

The documentation for this class was generated from the following files:

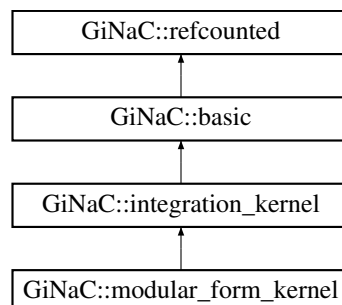
- [tensor.h](#)
- [tensor.cpp](#)

## 6.91 GiNaC::modular\_form\_kernel Class Reference

A kernel corresponding to a polynomial in Eisenstein series.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::modular\_form\_kernel:



## Public Member Functions

- `modular_form_kernel` (const `ex` &`k`, const `ex` &`P`, const `ex` &`C_norm=numeric(1)`)
- `ex series` (const `relational` &`r`, int `order`, unsigned `options=0`) const override  
*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position i.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex Laurent_series` (const `ex` &`qbar`, int `order`) const override  
*Returns the Laurent series, starting possibly with the pole term.*
- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc=0`) const override  
*Returns the value of the modular form.*
- `ex q_expansion_modular_form` (const `ex` &`q`, int `order`) const

## Protected Member Functions

- `bool uses_Laurent_series` () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method Laurent\_series needs to be implemented).*
- `void do_print` (const `print_context` &`c`, unsigned `level`) const

## Protected Attributes

- `ex k`
- `ex P`
- `ex C_norm`

### 6.91.1 Detailed Description

A kernel corresponding to a polynomial in Eisenstein series.

This class represents the differential one-form

$$\omega^{\text{modular}}(P_k(\eta_{k_1}^{(1)}, \dots, \eta_{k_r}^{(r)})) = C_k P_k(\eta_{k_1}^{(1)}, \dots, \eta_{k_r}^{(r)}) \frac{d\tilde{q}_N}{\tilde{q}_N}.$$

### 6.91.2 Constructor & Destructor Documentation

### 6.91.2.1 modular\_form\_kernel()

```
GiNaC::modular_form_kernel::modular_form_kernel (
    const ex & k,
    const ex & P,
    const ex & C_norm = numeric(1) )
```

## 6.91.3 Member Function Documentation

### 6.91.3.1 series()

```
ex GiNaC::modular_form_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.

Reimplemented from [GiNaC::integration\\_kernel](#).

References order, P, qbar, and r.

Referenced by [q\\_expansion\\_modular\\_form\(\)](#).

### 6.91.3.2 nops()

```
size_t GiNaC::modular_form_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.91.3.3 op()

```
ex GiNaC::modular_form_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References C\_norm, k, and P.

## 6.91.3.4 let\_op()

```
ex & GiNaC::modular_form_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References C\_norm, GiNaC::basic::ensure\_if\_modifiable(), k, and P.

## 6.91.3.5 is\_numeric()

```
bool GiNaC::modular_form_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References C\_norm, GiNaC::ex::evalf(), GiNaC::basic::evalf(), GiNaC::ex::info(), k, GiNaC::info\_flags::nonnegint, GiNaC::info\_flags::numeric, P, qbar, GiNaC::ex::series(), GiNaC::series\_to\_poly(), and GiNaC::basic::subs().

## 6.91.3.6 Laurent\_series()

```
ex GiNaC::modular_form_kernel::Laurent_series (
    const ex & x,
    int order ) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References C\_norm, order, P, qbar, GiNaC::ex::series(), and GiNaC::series\_to\_poly().

## 6.91.3.7 get\_numerical\_value()

```
ex GiNaC::modular_form_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References C\_norm, GiNaC::integration\_kernel::get\_numerical\_value\_impl(), and qbar.

### 6.91.3.8 uses\_Laurent\_series()

```
bool GiNaC::modular_form_kernel::uses_Laurent_series ( ) const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 6.91.3.9 q\_expansion\_modular\_form()

```
ex GiNaC::modular_form_kernel::q_expansion_modular_form (
    const ex & q,
    int order ) const
```

References `series()`.

### 6.91.3.10 do\_print()

```
void GiNaC::modular_form_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References `c`, `C_norm`, `k`, `P`, and `GiNaC::ex::print()`.

## 6.91.4 Member Data Documentation

### 6.91.4.1 k

```
ex GiNaC::modular_form_kernel::k [protected]
```

Referenced by `do_print()`, `is_numeric()`, `let_op()`, and `op()`.

### 6.91.4.2 P

```
ex GiNaC::modular_form_kernel::P [protected]
```

Referenced by `do_print()`, `is_numeric()`, `Laurent_series()`, `let_op()`, `op()`, and `series()`.



### 6.91.4.3 C\_norm

`ex` `GiNaC::modular_form_kernel::C_norm` [protected]

Referenced by `do_print()`, `get_numerical_value()`, `is_numeric()`, `Laurent_series()`, `let_op()`, and `op()`.

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.92 GiNaC::basic\_partition\_generator::mpartition2 Struct Reference

```
#include <utils.h>
```

### Public Member Functions

- [mpartition2](#) (unsigned `n_`, unsigned `m_`)
- bool [next\\_partition](#) ()

### Public Attributes

- `std::vector< unsigned >` [x](#)
- unsigned [n](#)
- unsigned [m](#)

### 6.92.1 Constructor & Destructor Documentation

#### 6.92.1.1 mpartition2()

```
GiNaC::basic_partition_generator::mpartition2::mpartition2 (  
    unsigned n_,  
    unsigned m_ ) [inline]
```

References `k`, `m`, `n`, and `x`.

### 6.92.2 Member Function Documentation

### 6.92.2.1 next\_partition()

```
bool GiNaC::basic_partition_generator::mpartition2::next_partition ( ) [inline]
```

References k, m, and x.

Referenced by GiNaC::partition\_with\_zero\_parts\_generator::next(), and GiNaC::partition\_generator::next().

## 6.92.3 Member Data Documentation

### 6.92.3.1 x

```
std::vector<unsigned> GiNaC::basic_partition_generator::mpartition2::x
```

Referenced by GiNaC::partition\_with\_zero\_parts\_generator::get(), GiNaC::partition\_generator::get(), mpartition2(), and next\_partition().

### 6.92.3.2 n

```
unsigned GiNaC::basic_partition_generator::mpartition2::n
```

Referenced by mpartition2(), and GiNaC::partition\_with\_zero\_parts\_generator::next().

### 6.92.3.3 m

```
unsigned GiNaC::basic_partition_generator::mpartition2::m
```

Referenced by GiNaC::partition\_with\_zero\_parts\_generator::get(), GiNaC::partition\_generator::get(), mpartition2(), GiNaC::partition\_with\_zero\_parts\_generator::next(), and next\_partition().

The documentation for this struct was generated from the following file:

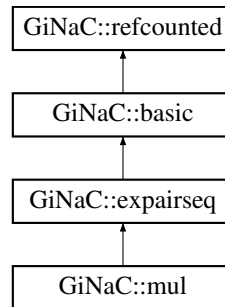
- [utils.h](#)

## 6.93 GiNaC::mul Class Reference

Product of expressions.

```
#include <mul.h>
```

Inheritance diagram for GiNaC::mul:



### Public Member Functions

- `mul` (const `ex` &lh, const `ex` &rh)
- `mul` (const `exvector` &v)
- `mul` (const `epvector` &v)
- `mul` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false)
- `mul` (`epvector` &&vp)
- `mul` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false)
- `mul` (const `ex` &lh, const `ex` &mh, const `ex` &rh)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthesizing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*
- int `ldegree` (const `ex` &s) const override  
*Return degree of lowest power in object s.*
- `ex coeff` (const `ex` &s, int n=1) const override  
*Return coefficient of degree n in object s.*
- bool `has` (const `ex` &other, unsigned `options`=0) const override  
*Test for occurrence of a pattern.*
- `ex eval` () const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalf` () const override  
*Evaluate object numerically.*
- `ex real_part` () const override
- `ex imag_part` () const override
- `ex evalm` () const override  
*Evaluate sums, products and integer powers of matrices.*
- `ex series` (const `relational` &s, int `order`, unsigned `options`=0) const override  
*Implementation of `ex::series()` for product.*

- `ex normal` (`exmap &repl`, `exmap &rev_lookup`, `lst &modifier`) const override  
*Implementation of `ex::normal()` for a product.*
- `numeric integer_content` () const override
- `ex smod` (const `numeric &xi`) const override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient` () const override  
*Implementation `ex::max_coefficient()`.*
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- `ex conjugate` () const override
- `ex algebraic_subs_mul` (const `exmap &m`, unsigned `options`) const

## Protected Member Functions

- `ex derivative` (const `symbol &s`) const override  
*Implementation of `ex::diff()` for a product.*
- `ex eval_ncmul` (const `exvector &v`) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex thisexpairseq` (const `epvector &v`, const `ex &oc`, bool `do_index_renaming=false`) const override
- `ex thisexpairseq` (`epvector &&vp`, const `ex &oc`, bool `do_index_renaming=false`) const override
- `expair split_ex_to_pair` (const `ex &e`) const override
- `expair combine_ex_with_coeff_to_pair` (const `ex &e`, const `ex &c`) const override
- `expair combine_pair_with_coeff_to_pair` (const `expair &p`, const `ex &c`) const override
- `ex recombine_pair_to_ex` (const `expair &p`) const override
- bool `expair_needs_further_processing` (`epp` it) override
- `ex default_overall_coeff` () const override
- void `combine_overall_coeff` (const `ex &c`) override
- void `combine_overall_coeff` (const `ex &c1`, const `ex &c2`) override
- bool `can_make_flat` (const `expair &p`) const override
- `ex expand` (unsigned `options=0`) const override  
*Expand expression, i.e.*
- void `find_real_imag` (`ex &`, `ex &`) const
- void `print_overall_coeff` (const `print_context &c`, const char \*`mul_sym`) const
- void `do_print` (const `print_context &c`, unsigned `level`) const
- void `do_print_latex` (const `print_latex &c`, unsigned `level`) const
- void `do_print_csrc` (const `print_csrc &c`, unsigned `level`) const
- void `do_print_python_repr` (const `print_python_repr &c`, unsigned `level`) const
- `epvector expandchildren` (unsigned `options`) const  
*Member-wise expand the expairs representing this sequence.*

## Static Protected Member Functions

- static bool `can_be_further_expanded` (const `ex &e`)

## Friends

- class `add`
- class `ncmul`
- class `power`

## Additional Inherited Members

### 6.93.1 Detailed Description

Product of expressions.

### 6.93.2 Constructor & Destructor Documentation

#### 6.93.2.1 mul() [1/7]

```
GiNaC::mul::mul (
    const ex & lh,
    const ex & rh )
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_↔\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

Referenced by [do\\_print\\_latex\(\)](#).

#### 6.93.2.2 mul() [2/7]

```
GiNaC::mul::mul (
    const exvector & v )
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_exvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_↔\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

#### 6.93.2.3 mul() [3/7]

```
GiNaC::mul::mul (
    const epvector & v )
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_↔\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.93.2.4 mul()** [4/7]

```
GiNaC::mul::mul (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false )
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.93.2.5 mul()** [5/7]

```
GiNaC::mul::mul (
    epvector && vp )
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.93.2.6 mul()** [6/7]

```
GiNaC::mul::mul (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false )
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.93.2.7 mul()** [7/7]

```
GiNaC::mul::mul (
    const ex & lh,
    const ex & mh,
    const ex & rh )
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_exvector\(\)](#), [factors](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.93.3 Member Function Documentation**

#### 6.93.3.1 precedence()

```
unsigned GiNaC::mul::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::expairseq](#).

Referenced by `do_print()`, `do_print_csrc()`, `do_print_latex()`, and `print_overall_coeff()`.

#### 6.93.3.2 info()

```
bool GiNaC::mul::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::expairseq](#).

References `GiNaC::_num1_p`, `GiNaC::factor()`, `GiNaC::numeric::info()`, `GiNaC::ex::info()`, `GiNaC::is_negative()`, `GiNaC::is_positive()`, and `GiNaC::real()`.

#### 6.93.3.3 is\_polynomial()

```
bool GiNaC::mul::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

#### 6.93.3.4 degree()

```
int GiNaC::mul::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object `s`.

Reimplemented from [GiNaC::basic](#).

#### 6.93.3.5 ldegree()

```
int GiNaC::mul::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

#### 6.93.3.6 coeff()

```
ex GiNaC::mul::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::coeff\(\)](#), and [n](#).

Referenced by [print\\_overall\\_coeff\(\)](#).

#### 6.93.3.7 has()

```
bool GiNaC::mul::has (
    const ex & pattern,
    unsigned options = 0 ) const [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::has\(\)](#), [GiNaC::nops\(\)](#), and [options](#).



## 6.93.3.8 eval()

```
ex GiNaC::mul::eval ( ) const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following  $x, x_1, x_2, \dots$  stand for a symbolic variables of type `ex` and  $c, c_1, c_2, \dots$  stand for such expressions that contain a plain number.

- $*(\dots, x; 0) \rightarrow 0$
- $*(+(x_1, x_2, \dots); c) \rightarrow *(*(x_1, c), *(x_2, c), \dots)$
- $*(x; 1) \rightarrow x$
- $*(; c) \rightarrow c$

Reimplemented from [GiNaC::expairseq](#).

References `GiNaC::_ex0`, `GiNaC::_ex1`, `GiNaC::_num1_p`, `GiNaC::_num_1_p`, `c`, `GiNaC::basic::clearflag()`, `GiNaC::coeff()`, `GiNaC::add::combine_pair_with_coeff_to_pair()`, `GINAC_ASSERT`, `GiNaC::numeric::is_integer()`, `GiNaC::numeric::is_pos_integer()`, last, likely, `GiNaC::numeric::mul()`, `GiNaC::expairseq::overall_coeff`, `GiNaC::expairseq::seq`, and unlikely.

## 6.93.3.9 evalf()

```
ex GiNaC::mul::evalf ( ) const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

## 6.93.3.10 real\_part()

```
ex GiNaC::mul::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

## 6.93.3.11 imag\_part()

```
ex GiNaC::mul::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 6.93.3.12 evalm()

```
ex GiNaC::mul::evalm ( ) const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), and [m](#).

### 6.93.3.13 series()

```
ex GiNaC::mul::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for product.

This performs series multiplication when multiplying series.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::begin\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::factor\(\)](#), [GINAC\\_↵  
ASSERT](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::ldegree\(\)](#), [GiNaC::pseries::mul\\_const\(\)](#), [GiNaC↵  
C::pseries::mul\\_series\(\)](#), [GiNaC::expairseq::op\(\)](#), [options](#), [order](#), [GiNaC::expairseq::overall\\_coeff](#), [r](#), [recombine\\_↵  
pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::ex::series\(\)](#), and [GiNaC::to\\_int\(\)](#).

### 6.93.3.14 normal()

```
ex GiNaC::mul::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for a product.

It cancels common factors from fractions.

See also

[ex::normal\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::frac\\_cancel\(\)](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC↵  
::container< C >::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

## 6.93.3.15 integer\_content()

```
numeric GiNaC::mul::integer_content ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

## 6.93.3.16 smod()

```
ex GiNaC::mul::smod (
    const numeric & xi ) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

## Parameters

$xi$	modulus
------	---------

## Returns

mapped polynomial

## See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [GiNaC::smod\(\)](#).

## 6.93.3.17 max\_coefficient()

```
numeric GiNaC::mul::max_coefficient ( ) const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

## See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

**6.93.3.18** `get_free_indices()`

```
exvector GiNaC::mul::get_free_indices ( ) const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

**6.93.3.19** `conjugate()`

```
ex GiNaC::mul::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [c](#), and [x](#).

**6.93.3.20** `derivative()`

```
ex GiNaC::mul::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a product.

It applies the product rule.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::begin\(\)](#), [GiNaC::basic::diff\(\)](#), [GiNaC::pow\(\)](#), and [GiNaC::expair::swap\(\)](#).

**6.93.3.21** `eval_ncmul()`

```
ex GiNaC::mul::eval_ncmul (
    const exvector & v ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::eval\\_ncmul\(\)](#).

**6.93.3.22** `return_type()`

```
unsigned GiNaC::mul::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

**6.93.3.23** `return_type_tinfo()`

```
return_type_t GiNaC::mul::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.93.3.24** `thisexpairseq()` [1/2]

```
ex GiNaC::mul::thisexpairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

**6.93.3.25** `thisexpairseq()` [2/2]

```
ex GiNaC::mul::thisexpairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

**6.93.3.26** `split_ex_to_pair()`

```
expair GiNaC::mul::split_ex_to_pair (
    const ex & e ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::power::basis](#), and [GiNaC::power::exponent](#).

**6.93.3.27 combine\_ex\_with\_coeff\_to\_pair()**

```

expair GiNaC::mul::combine_ex_with_coeff_to_pair (
    const ex & e,
    const ex & c ) const [override], [protected], [virtual]

```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [c](#), [GINAC\\_ASSERT](#), and [GiNaC::pow\(\)](#).

**6.93.3.28 combine\_pair\_with\_coeff\_to\_pair()**

```

expair GiNaC::mul::combine_pair_with_coeff_to_pair (
    const expair & p,
    const ex & c ) const [override], [protected], [virtual]

```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [c](#), [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::pow\(\)](#), and [GiNaC::expair::rest](#).

**6.93.3.29 recombine\_pair\_to\_ex()**

```

ex GiNaC::mul::recombine_pair_to_ex (
    const expair & p ) const [override], [protected], [virtual]

```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::expair::coeff](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::expair::rest](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [integer\\_content\(\)](#), [max\\_coefficient\(\)](#), [normal\(\)](#), [series\(\)](#), and [smod\(\)](#).

**6.93.3.30 expair\_needs\_further\_processing()**

```

bool GiNaC::mul::expair_needs_further_processing (
    epp it ) [override], [protected], [virtual]

```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), and [GiNaC::expair::is\\_equal\(\)](#).

#### 6.93.3.31 default\_overall\_coeff()

```
ex GiNaC::mul::default_overall_coeff ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#).

#### 6.93.3.32 combine\_overall\_coeff() [1/2]

```
void GiNaC::mul::combine_overall_coeff (
    const ex & c ) [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [c](#), and [GINAC\\_ASSERT](#).

#### 6.93.3.33 combine\_overall\_coeff() [2/2]

```
void GiNaC::mul::combine_overall_coeff (
    const ex & c1,
    const ex & c2 ) [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GINAC\\_ASSERT](#).

#### 6.93.3.34 can\_make\_flat()

```
bool GiNaC::mul::can_make_flat (
    const expair & p ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), and [GiNaC::ex::info\(\)](#).

### 6.93.3.35 expand()

```
ex GiNaC::mul::expand (
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_num0\\_p](#), [GiNaC::ex::expand\(\)](#), [factors](#), [GiNaC::get\\_all\\_dummy\\_indices\\_↵safely\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [n](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::ex::op\(\)](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.36 algebraic\_subs\_mul()

```
ex GiNaC::mul::algebraic_subs_mul (
    const exmap & m,
    unsigned options ) const
```

References [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [m](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), [options](#), [GiNaC::pow\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::tryfactsubs\(\)](#).

### 6.93.3.37 find\_real\_imag()

```
void GiNaC::mul::find_real_imag (
    ex & rp,
    ex & ip ) const [protected]
```

References [GiNaC::ex::expand\(\)](#), [GiNaC::factor\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [GiNaC::ex::real\\_part\(\)](#).

### 6.93.3.38 print\_overall\_coeff()

```
void GiNaC::mul::print_overall_coeff (
    const print_context & c,
    const char * mul_sym ) const [protected]
```

References [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [c](#), [coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::expairseq::overall\\_↵coeff](#), [precedence\(\)](#), and [GiNaC::ex::print\(\)](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).



**6.93.3.39 do\_print()**

```
void GiNaC::mul::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References `c`, `precedence()`, `GiNaC::ex::print()`, `print_overall_coeff()`, `recombine_pair_to_ex()`, and `GiNaC::expairseq::seq`.

**6.93.3.40 do\_print\_latex()**

```
void GiNaC::mul::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

References `c`, `GINAC_ASSERT`, `mul()`, `precedence()`, `print_overall_coeff()`, `recombine_pair_to_ex()`, and `GiNaC::expairseq::seq`.

**6.93.3.41 do\_print\_csrc()**

```
void GiNaC::mul::do_print_csrc (
    const print_csrc & c,
    unsigned level ) const [protected]
```

References `GiNaC::_ex1`, `GiNaC::_ex_1`, `c`, `GiNaC::ex::is_equal()`, `GiNaC::expairseq::overall_coeff`, `precedence()`, and `GiNaC::ex::print()`.

**6.93.3.42 do\_print\_python\_repr()**

```
void GiNaC::mul::do_print_python_repr (
    const print_python_repr & c,
    unsigned level ) const [protected]
```

References `c`, `GiNaC::nops()`, `GiNaC::op()`, and `GiNaC::ex::print()`.

**6.93.3.43 can\_be\_further\_expanded()**

```
bool GiNaC::mul::can_be_further_expanded (
    const ex & e ) [static], [protected]
```

References `GiNaC::ex::info()`, and `GiNaC::ex::op()`.

#### 6.93.3.44 `expandchildren()`

```
epvector GiNaC::mul::expandchildren (
    unsigned options ) const [protected]
```

Member-wise expand the expairs representing this sequence.

This must be overridden from `expairseq::expandchildren()` and done iteratively in order to allow for early cancellations and thus save memory.

See also

[mul::expand\(\)](#)

Returns

epvector containing expanded pairs, empty if no members had to be changed.

References `GiNaC::are_ex_trivially_equal()`, `GiNaC::ex::begin()`, `GiNaC::ex::expand()`, `GiNaC::factor()`, `last`, and `options`.

### 6.93.4 Friends And Related Function Documentation

#### 6.93.4.1 `add`

```
friend class add [friend]
```

#### 6.93.4.2 `ncmul`

```
friend class ncmul [friend]
```

#### 6.93.4.3 `power`

```
friend class power [friend]
```

The documentation for this class was generated from the following files:

- [mul.h](#)
- [indexed.cpp](#)
- [mul.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.94 GiNaC::multi\_iterator\_counter< T > Class Template Reference

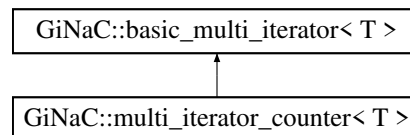
The class `multi_iterator_counter` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N$$

.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for `GiNaC::multi_iterator_counter< T >`:



### Public Member Functions

- `multi_iterator_counter` (void)  
*Default constructor.*
- `multi_iterator_counter` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k.*
- `multi_iterator_counter` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- `basic_multi_iterator< T > & init` (void)  
*Initialize the multi-index to*  

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$
- `basic_multi_iterator< T > & operator++` (int)  
*The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.*

### Friends

- `template<class TT >`  
`std::ostream & operator<<` (std::ostream &os, const `multi_iterator_counter< TT > &v`)

### Additional Inherited Members

#### 6.94.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_counter< T >
```

The class `multi_iterator_counter` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N$$

.

## 6.94.2 Constructor & Destructor Documentation

### 6.94.2.1 multi\_iterator\_counter() [1/3]

```
template<class T >
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
    void ) [inline]
```

Default constructor.

### 6.94.2.2 multi\_iterator\_counter() [2/3]

```
template<class T >
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
    T B,
    T N,
    size_t k ) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

### 6.94.2.3 multi\_iterator\_counter() [3/3]

```
template<class T >
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
    T B,
    T N,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

## 6.94.3 Member Function Documentation

### 6.94.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

## 6.94.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References k.

## 6.94.4 Friends And Related Function Documentation

## 6.94.4.1 operator&lt;&lt;

```
template<class T >
template<class TT >
std::ostream& operator<< (
    std::ostream & os,
    const multi_iterator_counter< TT > & v ) [friend]
```

The documentation for this class was generated from the following file:

- [utils\\_multi\\_iterator.h](#)

## 6.95 GiNaC::multi\_iterator\_counter\_indv&lt; T &gt; Class Template Reference

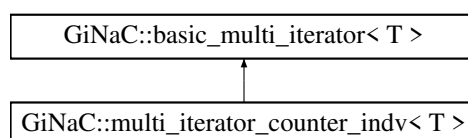
The class [multi\\_iterator\\_counter\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N_j$$

.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_counter\_indv< T >:



## Public Member Functions

- [multi\\_iterator\\_counter\\_indv](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_counter\\_indv](#) (T B, const std::vector< T > &Nv, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- [multi\\_iterator\\_counter\\_indv](#) (T B, const std::vector< T > &Nv, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > &init (void)  
*Initialize the multi-index to*  
$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$
  
.
- [basic\\_multi\\_iterator](#)< T > &operator++ (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Protected Attributes

- std::vector< T > Nv

## Friends

- template<class TT >  
std::ostream &operator<< (std::ostream &os, const [multi\\_iterator\\_counter\\_indv](#)< TT > &v)

## 6.95.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_counter_indv< T >
```

The class [multi\\_iterator\\_counter\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N_j$$

## 6.95.2 Constructor & Destructor Documentation

### 6.95.2.1 multi\_iterator\_counter\_indv() [1/3]

```
template<class T >
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
    void ) [inline]
```

Default constructor.

## 6.95.2.2 multi\_iterator\_counter\_indv() [2/3]

```
template<class T >
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
    T B,
    const std::vector< T > & Nv,
    size_t k ) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

## 6.95.2.3 multi\_iterator\_counter\_indv() [3/3]

```
template<class T >
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
    T B,
    const std::vector< T > & Nv,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

## 6.95.3 Member Function Documentation

## 6.95.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter_indv< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

## 6.95.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter_indv< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.

If n is in the last configuration and the increment operator ++ is applied to n, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References k.

## 6.95.4 Friends And Related Function Documentation

### 6.95.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream& operator<< (
    std::ostream & os,
    const multi_iterator_counter_indv< TT > & v ) [friend]
```

## 6.95.5 Member Data Documentation

### 6.95.5.1 Nv

```
template<class T >
std::vector<T> GiNaC::multi_iterator_counter_indv< T >::Nv [protected]
```

The documentation for this class was generated from the following file:

- [utils\\_multi\\_iterator.h](#)

## 6.96 GiNaC::multi\_iterator\_ordered< T > Class Template Reference

The class [multi\\_iterator\\_ordered](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N$$

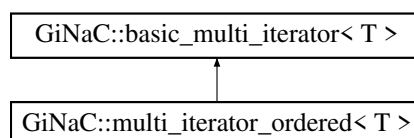
and

$$i_j < i_{j+1}.$$

It is assumed that  $k > 0$  and  $N - B \geq k$ .

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_ordered< T >:





## Public Member Functions

- [multi\\_iterator\\_ordered](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_ordered](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k.*
- [multi\\_iterator\\_ordered](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to*  
$$(n_1, n_2, n_3, \dots, n_k) = (B + 0, B + 1, B + 2, \dots, B + k - 1)$$
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Friends

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_ordered](#)< TT > &v)

## Additional Inherited Members

### 6.96.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_ordered< T >
```

The class [multi\\_iterator\\_ordered](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N$$

and

$$i_j < i_{j+1}.$$

It is assumed that  $k > 0$  and  $N - B \geq k$ .

### 6.96.2 Constructor & Destructor Documentation

#### 6.96.2.1 multi\_iterator\_ordered() [1/3]

```
template<class T >
GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered (
    void ) [inline]
```

Default constructor.

### 6.96.2.2 multi\_iterator\_ordered() [2/3]

```
template<class T >
GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered (
    T B,
    T N,
    size_t k ) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

### 6.96.2.3 multi\_iterator\_ordered() [3/3]

```
template<class T >
GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered (
    T B,
    T N,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

## 6.96.3 Member Function Documentation

### 6.96.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to

$$(n_1, n_2, n_3, \dots, n_k) = (B + 0, B + 1, B + 2, \dots, B + k - 1)$$

.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

### 6.96.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.

If n is in the last configuration and the increment operator ++ is applied to n, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References k.

## 6.96.4 Friends And Related Function Documentation

### 6.96.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream& operator<< (
    std::ostream & os,
    const multi_iterator_ordered< TT > & v ) [friend]
```

The documentation for this class was generated from the following file:

- [utils\\_multi\\_iterator.h](#)

## 6.97 GiNaC::multi\_iterator\_ordered\_eq< T > Class Template Reference

The class [multi\\_iterator\\_ordered\\_eq](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N$$

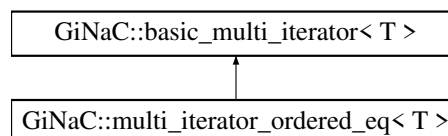
and

$$i_j \leq i_{j+1}.$$

It is assumed that  $k > 0$ .

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_ordered\_eq< T >:



### Public Member Functions

- [multi\\_iterator\\_ordered\\_eq](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_ordered\\_eq](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- [multi\\_iterator\\_ordered\\_eq](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & init (void)  
*Initialize the multi-index to*  

$$(n_1, n_2, \dots, n_k) = (B, B, \dots, B)$$
- [basic\\_multi\\_iterator](#)< T > & operator++ (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Friends

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const multi_iterator_ordered_eq< TT > &v)`

## Additional Inherited Members

### 6.97.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_ordered_eq< T >
```

The class `multi_iterator_ordered_eq` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N$$

and

$$i_j \leq i_{j+1}.$$

It is assumed that  $k > 0$ .

### 6.97.2 Constructor & Destructor Documentation

#### 6.97.2.1 multi\_iterator\_ordered\_eq() [1/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
    void ) [inline]
```

Default constructor.

#### 6.97.2.2 multi\_iterator\_ordered\_eq() [2/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
    T B,
    T N,
    size_t k ) [inline], [explicit]
```

Construct a `multi_iterator` with upper limit `N` and size `k`.

## 6.97.2.3 multi\_iterator\_ordered\_eq() [3/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
    T B,
    T N,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

## 6.97.3 Member Function Documentation

## 6.97.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to

$$(n_1, n_2, \dots, n_k) = (B, B, \dots, B)$$

.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

## 6.97.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.

If `n` is in the last configuration and the increment operator `++` is applied to `n`, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References `k`.

## 6.97.4 Friends And Related Function Documentation

### 6.97.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream& operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq< TT > & v ) [friend]
```

The documentation for this class was generated from the following file:

- [utils\\_multi\\_iterator.h](#)

## 6.98 GiNaC::multi\_iterator\_ordered\_eq\_indv< T > Class Template Reference

The class [multi\\_iterator\\_ordered\\_eq\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N_j$$

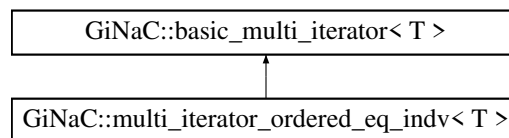
and

$$i_j \leq i_{j+1}.$$

.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_ordered\_eq\_indv< T >:



### Public Member Functions

- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (T B, const std::vector< T > &Nv, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k.*
- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (T B, const std::vector< T > &Nv, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator< T > & init](#) (void)  
*Initialize the multi-index to*  

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, B, \dots, B)$$
- [basic\\_multi\\_iterator< T > & operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

### Protected Attributes

- std::vector< T > Nv

## Friends

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const multi_iterator_ordered_eq_indv< TT > &v)`

## 6.98.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_ordered_eq_indv< T >
```

The class `multi_iterator_ordered_eq_indv` defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N_j$$

and

$$i_j \leq i_{j+1}.$$

## 6.98.2 Constructor & Destructor Documentation

### 6.98.2.1 multi\_iterator\_ordered\_eq\_indv() [1/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
    void ) [inline]
```

Default constructor.

### 6.98.2.2 multi\_iterator\_ordered\_eq\_indv() [2/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
    T B,
    const std::vector< T > & Nv,
    size_t k ) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

### 6.98.2.3 multi\_iterator\_ordered\_eq\_indv() [3/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
    T B,
    const std::vector< T > & Nv,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

### 6.98.3 Member Function Documentation

#### 6.98.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq_indv< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, B, \dots, B)$$

.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

#### 6.98.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq_indv< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References  $k$ .

### 6.98.4 Friends And Related Function Documentation

#### 6.98.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream& operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq_indv< TT > & v ) [friend]
```

### 6.98.5 Member Data Documentation



## 6.98.5.1 Nv

```
template<class T >
std::vector<T> GiNaC::multi_iterator_ordered_eq_indv< T >::Nv [protected]
```

The documentation for this class was generated from the following file:

- [utils\\_multi\\_iterator.h](#)

## 6.99 GiNaC::multi\_iterator\_permutation&lt; T &gt; Class Template Reference

The class [multi\\_iterator\\_permutation](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , for which

$$B \leq i_j < N$$

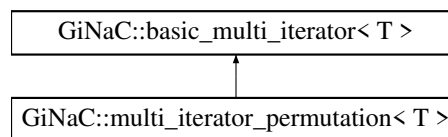
and

$$i_i \neq i_j$$

In particular, if  $N - B = k$ , [multi\\_iterator\\_permutation](#) loops over all permutations of  $k$  elements.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_permutation< T >:



## Public Member Functions

- [multi\\_iterator\\_permutation](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_permutation](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k.*
- [multi\\_iterator\\_permutation](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator< T > &init](#) (void)  
*Initialize the multi-index to*

$$(n_1, n_2, n_3, \dots, n_k) = (B + 0, B + 1, B + 2, \dots, B + k - 1)$$

- [basic\\_multi\\_iterator< T > &operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*
- int [get\\_sign](#) (void) const

*Returns the sign of the permutation, defined by*

$$(-1)^{n_{inv}},$$

*where  $n_{inv}$  is the number of inversions, e.g.*

## Friends

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const multi_iterator_permutation< TT > &v)`

## Additional Inherited Members

### 6.99.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_permutation< T >
```

The class `multi_iterator_permutation` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , for which

$$B \leq i_j < N$$

and

$$i_i \neq i_j$$

In particular, if  $N - B = k$ , `multi_iterator_permutation` loops over all permutations of  $k$  elements.

### 6.99.2 Constructor & Destructor Documentation

#### 6.99.2.1 multi\_iterator\_permutation() [1/3]

```
template<class T >
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
    void ) [inline]
```

Default constructor.

#### 6.99.2.2 multi\_iterator\_permutation() [2/3]

```
template<class T >
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
    T B,
    T N,
    size_t k ) [inline], [explicit]
```

Construct a `multi_iterator` with upper limit  $N$  and size  $k$ .

## 6.99.2.3 multi\_iterator\_permutation() [3/3]

```
template<class T >
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
    T B,
    T N,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

## 6.99.3 Member Function Documentation

## 6.99.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_permutation< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to

$$(n_1, n_2, n_3, \dots, n_k) = (B + 0, B + 1, B + 2, \dots, B + k - 1)$$

.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

## 6.99.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_permutation< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References  $k$ .

### 6.99.3.3 get\_sign()

```
template<class T >
int GiNaC::multi_iterator_permutation< T >::get_sign (
    void ) const [inline]
```

Returns the sign of the permutation, defined by

$$(-1)^{n_{inv}},$$

where  $n_{inv}$  is the number of inversions, e.g.

the number of pairs  $i < j$  for which

$$n_i > n_j.$$

References k.

## 6.99.4 Friends And Related Function Documentation

### 6.99.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream& operator<< (
    std::ostream & os,
    const multi_iterator_permutation< TT > & v ) [friend]
```

The documentation for this class was generated from the following file:

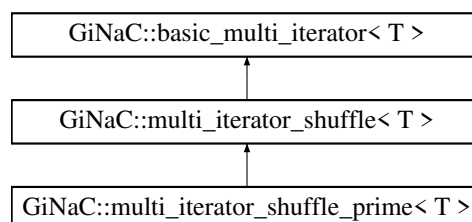
- [utils\\_multi\\_iterator.h](#)

## 6.100 GiNaC::multi\_iterator\_shuffle< T > Class Template Reference

The class [multi\\_iterator\\_shuffle](#) defines a multi\_iterator, which runs over all shuffles of a and b.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_shuffle< T >:



## Public Member Functions

- [multi\\_iterator\\_shuffle](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_shuffle](#) (const std::vector< T > &a, const std::vector< T > &b)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to the first shuffle.*
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Protected Attributes

- size\_t [N\\_internal](#)
- std::vector< size\_t > [v\\_internal](#)
- std::vector< T > [v\\_orig](#)

## Friends

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_shuffle](#)< TT > &v)

### 6.100.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_shuffle< T >
```

The class [multi\\_iterator\\_shuffle](#) defines a multi\_iterator, which runs over all shuffles of a and b.

### 6.100.2 Constructor & Destructor Documentation

#### 6.100.2.1 multi\_iterator\_shuffle() [1/2]

```
template<class T >
GiNaC::multi_iterator_shuffle< T >::multi_iterator_shuffle (
    void ) [inline]
```

Default constructor.

### 6.100.2.2 multi\_iterator\_shuffle() [2/2]

```
template<class T >
GiNaC::multi_iterator_shuffle< T >::multi_iterator_shuffle (
    const std::vector< T > & a,
    const std::vector< T > & b ) [inline], [explicit]
```

Construct from a vector.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::multi\\_iterator\\_shuffle< T >::N\\_internal](#), [GiNaC::basic\\_multi\\_iterator< T >::v](#), [GiNaC::multi\\_iterator\\_shuffle< T >::v\\_internal](#), and [GiNaC::multi\\_iterator\\_shuffle< T >::v\\_orig](#).

## 6.100.3 Member Function Documentation

### 6.100.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to the first shuffle.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

Reimplemented in [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#).

### 6.100.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.

If `n` is in the last configuration and the increment operator `++` is applied to `n`, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References `k`.

## 6.100.4 Friends And Related Function Documentation

#### 6.100.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream& operator<< (
    std::ostream & os,
    const multi_iterator_shuffle< TT > & v ) [friend]
```

### 6.100.5 Member Data Documentation

#### 6.100.5.1 N\_internal

```
template<class T >
size_t GiNaC::multi_iterator_shuffle< T >::N_internal [protected]
```

Referenced by `GiNaC::multi_iterator_shuffle< T >::multi_iterator_shuffle()`.

#### 6.100.5.2 v\_internal

```
template<class T >
std::vector<size_t> GiNaC::multi_iterator_shuffle< T >::v_internal [protected]
```

Referenced by `GiNaC::multi_iterator_shuffle< T >::multi_iterator_shuffle()`.

#### 6.100.5.3 v\_orig

```
template<class T >
std::vector<T> GiNaC::multi_iterator_shuffle< T >::v_orig [protected]
```

Referenced by `GiNaC::multi_iterator_shuffle< T >::multi_iterator_shuffle()`.

The documentation for this class was generated from the following file:

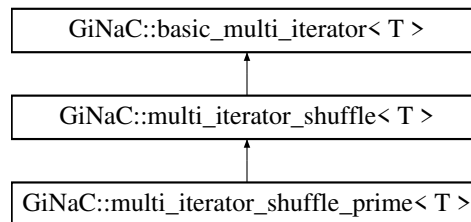
- [utils\\_multi\\_iterator.h](#)

## 6.101 GiNaC::multi\_iterator\_shuffle\_prime< T > Class Template Reference

The class [multi\\_iterator\\_shuffle\\_prime](#) defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_shuffle\_prime< T >:



### Public Member Functions

- [multi\\_iterator\\_shuffle\\_prime](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_shuffle\\_prime](#) (const std::vector< T > &a, const std::vector< T > &b)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > &init (void)  
*Initialize the multi-index to the first shuffle.*

### Friends

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_shuffle\\_prime](#)< TT > &v)

### Additional Inherited Members

#### 6.101.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_shuffle_prime< T >
```

The class [multi\\_iterator\\_shuffle\\_prime](#) defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).

#### 6.101.2 Constructor & Destructor Documentation



#### 6.101.2.1 multi\_iterator\_shuffle\_prime() [1/2]

```
template<class T >
GiNaC::multi_iterator_shuffle_prime< T >::multi_iterator_shuffle_prime (
    void ) [inline]
```

Default constructor.

#### 6.101.2.2 multi\_iterator\_shuffle\_prime() [2/2]

```
template<class T >
GiNaC::multi_iterator_shuffle_prime< T >::multi_iterator_shuffle_prime (
    const std::vector< T > & a,
    const std::vector< T > & b ) [inline], [explicit]
```

Construct from a vector.

### 6.101.3 Member Function Documentation

#### 6.101.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle_prime< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to the first shuffle.

Reimplemented from [GiNaC::multi\\_iterator\\_shuffle< T >](#).

### 6.101.4 Friends And Related Function Documentation

#### 6.101.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream& operator<< (
    std::ostream & os,
    const multi_iterator_shuffle_prime< TT > & v ) [friend]
```

The documentation for this class was generated from the following file:

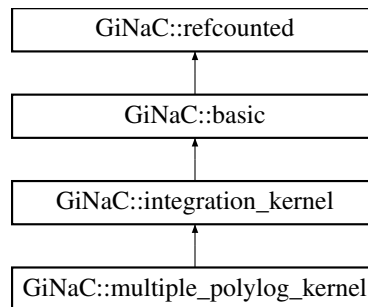
- [utils\\_multi\\_iterator.h](#)

## 6.102 GiNaC::multiple\_polylog\_kernel Class Reference

The integration kernel for multiple polylogarithms.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::multiple\_polylog\_kernel:



### Public Member Functions

- [multiple\\_polylog\\_kernel](#) (const [ex](#) &[z](#))
- [size\\_t nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t [i](#)) const override  
*Return operand/member at position [i](#).*
- [ex & let\\_op](#) (size\_t [i](#)) override  
*Return modifiable operand/member at position [i](#).*
- [bool is\\_numeric](#) (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*

### Protected Member Functions

- [cln::cl\\_N series\\_coeff\\_impl](#) (int [i](#)) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- [void do\\_print](#) (const [print\\_context](#) &[c](#), unsigned [level](#)) const

### Protected Attributes

- [ex z](#)

#### 6.102.1 Detailed Description

The integration kernel for multiple polylogarithms.

This class represents the differential one-form

$$\omega^{\text{mpl}}(z) = \frac{d\lambda}{\lambda - z}$$

For the case  $z = 0$  the class [basic\\_log\\_kernel](#) should be used.

## 6.102.2 Constructor & Destructor Documentation

### 6.102.2.1 multiple\_polylog\_kernel()

```
GiNaC::multiple_polylog_kernel::multiple_polylog_kernel (
    const ex & z )
```

## 6.102.3 Member Function Documentation

### 6.102.3.1 nops()

```
size_t GiNaC::multiple_polylog_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.102.3.2 op()

```
ex GiNaC::multiple_polylog_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [z](#).

### 6.102.3.3 let\_op()

```
ex & GiNaC::multiple_polylog_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), and [z](#).

#### 6.102.3.4 `is_numeric()`

```
bool GiNaC::multiple_polylog_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), and [z](#).

#### 6.102.3.5 `series_coeff_impl()`

```
cln::cl_N GiNaC::multiple_polylog_kernel::series_coeff_impl (
    int i ) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), and [z](#).

#### 6.102.3.6 `do_print()`

```
void GiNaC::multiple_polylog_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::ex::print\(\)](#), and [z](#).

### 6.102.4 Member Data Documentation

#### 6.102.4.1 `z`

```
ex GiNaC::multiple_polylog_kernel::z [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

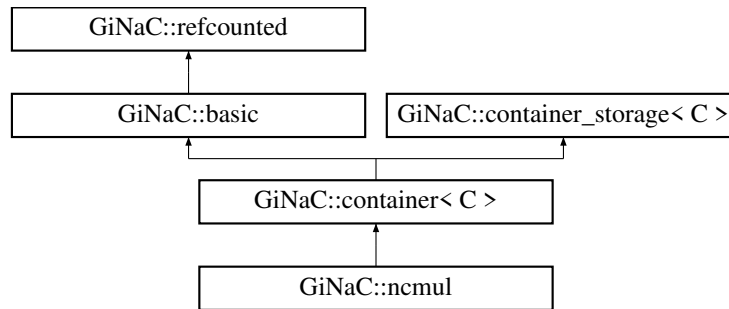
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.103 GiNaC::ncmul Class Reference

Non-commutative product of expressions.

```
#include <ncmul.h>
```

Inheritance diagram for GiNaC::ncmul:



### Public Member Functions

- `ncmul` (const `ex` &lh, const `ex` &rh)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4, const `ex` &f5)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4, const `ex` &f5, const `ex` &f6)
- `ncmul` (const `exvector` &v)
- `ncmul` (`exvector` &&v)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*
- int `ldegree` (const `ex` &s) const override  
*Return degree of lowest power in object s.*
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- `ex coeff` (const `ex` &s, int `n`=1) const override  
*Return coefficient of degree n in object s.*
- `ex eval` () const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalm` () const override  
*Evaluate sums, products and integer powers of matrices.*
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- `ex thiscontainer` (const `exvector` &v) const override
- `ex thiscontainer` (`exvector` &&v) const override
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- const `exvector` & `get_factors` () const

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for a non-commutative product.*
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_csrc` (const `print_context` &c, unsigned level) const
- `size_t count_factors` (const `ex` &e) const
- void `append_factors` (`exvector` &v, const `ex` &e) const
- `exvector expandchildren` (unsigned `options`) const

## Friends

- class `power`
- `ex reeval_ncmul` (const `exvector` &v)
- `ex hold_ncmul` (const `exvector` &v)

## Additional Inherited Members

### 6.103.1 Detailed Description

Non-commutative product of expressions.

### 6.103.2 Constructor & Destructor Documentation

#### 6.103.2.1 `ncmul()` [1/7]

```
GiNaC::ncmul::ncmul (
    const ex & lh,
    const ex & rh )
```

#### 6.103.2.2 `ncmul()` [2/7]

```
GiNaC::ncmul::ncmul (
    const ex & f1,
    const ex & f2,
    const ex & f3 )
```

### 6.103.2.3 ncmul() [3/7]

```
GiNaC::ncmul::ncmul (
    const ex & f1,
    const ex & f2,
    const ex & f3,
    const ex & f4 )
```

### 6.103.2.4 ncmul() [4/7]

```
GiNaC::ncmul::ncmul (
    const ex & f1,
    const ex & f2,
    const ex & f3,
    const ex & f4,
    const ex & f5 )
```

### 6.103.2.5 ncmul() [5/7]

```
GiNaC::ncmul::ncmul (
    const ex & f1,
    const ex & f2,
    const ex & f3,
    const ex & f4,
    const ex & f5,
    const ex & f6 )
```

### 6.103.2.6 ncmul() [6/7]

```
GiNaC::ncmul::ncmul (
    const exvector & v )
```

### 6.103.2.7 ncmul() [7/7]

```
GiNaC::ncmul::ncmul (
    exvector && v )
```

## 6.103.3 Member Function Documentation

#### 6.103.3.1 precedence()

```
unsigned GiNaC::ncmul::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< C >](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_csrc\(\)](#).

#### 6.103.3.2 info()

```
bool GiNaC::ncmul::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::container< C >](#).

#### 6.103.3.3 degree()

```
int GiNaC::ncmul::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object *s*.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::is\\_equal\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

#### 6.103.3.4 ldegree()

```
int GiNaC::ncmul::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object *s*.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::is\\_equal\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).



## 6.103.3.5 expand()

```
ex GiNaC::ncmul::expand (
    unsigned options = 0 ) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [k](#), [GiNaC::container< C >::op\(\)](#), [options](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

## 6.103.3.6 coeff()

```
ex GiNaC::ncmul::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [GiNaC::basic::is\\_equal\(\)](#), [n](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::add::coeff\(\)](#).

## 6.103.3.7 eval()

```
ex GiNaC::ncmul::eval ( ) const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following x, x1, x2,... stand for a symbolic variables of type ex and c, c1, c2... stand for such expressions that contain a plain number.

- `ncmul(...*(x1,x2),...,ncmul(x3,x4),...)` -> `ncmul(...,x1,x2,...,x3,x4,...)` (associativity)
- `ncmul(x)` -> `x`
- `ncmul()` -> `1`
- `ncmul(...,c1,...,c2,...)` -> `*(c1,c2,ncmul(...))` (pull out commutative elements)
- `ncmul(x1,y1,x2,y2)` -> `*(ncmul(x1,x2),ncmul(y1,y2))` (collect elements of same type)
- `ncmul(x1,x2,x3,...)` -> `x::eval_ncmul(x1,x2,x3,...)`

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [append\\_factors\(\)](#), [GiNaC::return\\_types::commutative](#), [count\\_factors\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::factor\(\)](#), [factors](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::return\\_types::noncommutative\\_composite](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), [GiNaC::ex::return\\_type\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

**6.103.3.8 evalm()**

```
ex GiNaC::ncmul::evalm ( ) const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::begin\(\)](#), [GiNaC::matrix::mul\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.103.3.9 get\_free\_indices()**

```
exvector GiNaC::ncmul::get_free_indices ( ) const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::container< C >::nops\(\)](#), and [GiNaC::container< C >::op\(\)](#).

**6.103.3.10 thiscontainer()** [1/2]

```
ex GiNaC::ncmul::thiscontainer (
    const exvector & v ) const [override]
```

**6.103.3.11 thiscontainer()** [2/2]

```
ex GiNaC::ncmul::thiscontainer (
    exvector && v ) const [override]
```

**6.103.3.12 conjugate()**

```
ex GiNaC::ncmul::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::is\\_clifford\\_tinfo\(\)](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::container< C >::nops\(\)](#), [return\\_type\(\)](#), and [return\\_type\\_tinfo\(\)](#).

**6.103.3.13 real\_part()**

```
ex GiNaC::ncmul::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::basic::real\\_part\(\)](#).

**6.103.3.14 imag\_part()**

```
ex GiNaC::ncmul::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::basic::imag\\_part\(\)](#).

**6.103.3.15 derivative()**

```
ex GiNaC::ncmul::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a non-commutative product.

It applies the product rule.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::ex::swap\(\)](#).

**6.103.3.16 return\_type()**

```
unsigned GiNaC::ncmul::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::container< C >::end\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::return\\_types::noncommutative\\_composite](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [conjugate\(\)](#).

**6.103.3.17 return\_type\_tinfo()**

```
return_type_t GiNaC::ncmul::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [conjugate\(\)](#).

**6.103.3.18 do\_print()**

```
void GiNaC::ncmul::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [precedence\(\)](#), and [GiNaC::container< C >::printseq\(\)](#).

**6.103.3.19 do\_print\_csrc()**

```
void GiNaC::ncmul::do_print_csrc (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [precedence\(\)](#), and [GiNaC::container< C >::printseq\(\)](#).

**6.103.3.20 count\_factors()**

```
size_t GiNaC::ncmul::count_factors (
    const ex & e ) const [protected]
```

References [GiNaC::return\\_types::commutative](#), [factors](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

Referenced by [eval\(\)](#).

**6.103.3.21 append\_factors()**

```
void GiNaC::ncmul::append_factors (
    exvector & v,
    const ex & e ) const [protected]
```

References [GiNaC::return\\_types::commutative](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

Referenced by [eval\(\)](#).

#### 6.103.3.22 expandchildren()

```
exvector GiNaC::ncmul::expandchildren (
    unsigned options ) const [protected]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::ex::expand\(\)](#), [options](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [expand\(\)](#).

#### 6.103.3.23 get\_factors()

```
const exvector & GiNaC::ncmul::get_factors ( ) const
```

References [GiNaC::container\\_storage< C >::seq](#).

### 6.103.4 Friends And Related Function Documentation

#### 6.103.4.1 power

```
friend class power [friend]
```

#### 6.103.4.2 reeval\_ncmul

```
ex reeval_ncmul (
    const exvector & v ) [friend]
```

#### 6.103.4.3 hold\_ncmul

```
ex hold_ncmul (
    const exvector & v ) [friend]
```

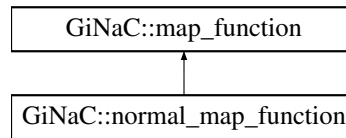
The documentation for this class was generated from the following files:

- [ncmul.h](#)
- [indexed.cpp](#)
- [ncmul.cpp](#)

## 6.104 GiNaC::normal\_map\_function Struct Reference

Function object to be applied by [basic::normal\(\)](#).

Inheritance diagram for GiNaC::normal\_map\_function:



### Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &[e](#)) override

### Additional Inherited Members

#### 6.104.1 Detailed Description

Function object to be applied by [basic::normal\(\)](#).

#### 6.104.2 Member Function Documentation

##### 6.104.2.1 operator()

```
ex GiNaC::normal_map_function::operator() (  
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::normal\(\)](#).

The documentation for this struct was generated from the following file:

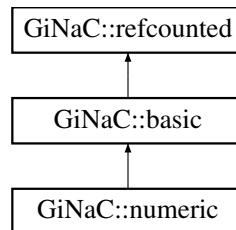
- [normal.cpp](#)

## 6.105 GiNaC::numeric Class Reference

This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.

```
#include <numeric.h>
```

Inheritance diagram for GiNaC::numeric:



### Public Member Functions

- [numeric](#) (int i)
- [numeric](#) (unsigned int i)
- [numeric](#) (long i)
- [numeric](#) (unsigned long i)
- [numeric](#) (long long i)
- [numeric](#) (unsigned long long i)
- [numeric](#) (long [numer](#), long [denom](#))  
*Constructor for rational numerics a/b.*
- [numeric](#) (double d)
- [numeric](#) (const char \*)  
*ctor from C-style string.*
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- bool [is\\_polynomial](#) (const [ex](#) &var) const override  
*Check whether this is a polynomial in the given variables.*
- int [degree](#) (const [ex](#) &s) const override  
*Return degree of highest power in object s.*
- int [ldegree](#) (const [ex](#) &s) const override  
*Return degree of lowest power in object s.*
- [ex](#) [coeff](#) (const [ex](#) &s, int n=1) const override  
*Return coefficient of degree n in object s.*
- bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const override  
*Disassemble real part and imaginary part to scan for the occurrence of a single number.*
- [ex](#) [eval](#) () const override  
*Evaluation of numbers doesn't do anything at all.*
- [ex](#) [evalf](#) () const override  
*Cast numeric into a floating-point object.*
- [ex](#) [subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [ex](#) [normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const override  
*Implementation of [ex::normal\(\)](#) for a numeric.*

- `ex to_rational` (`exmap` &repl) const override  
*Implementation of `ex::to_rational()` for a numeric.*
- `ex to_polynomial` (`exmap` &repl) const override  
*Implementation of `ex::to_polynomial()` for a numeric.*
- `numeric integer_content` () const override
- `ex smod` (const `numeric` &xi) const override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient` () const override  
*Implementation `ex::max_coefficient()`.*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Read (a.k.a.*
- const `numeric add` (const `numeric` &other) const  
*Numerical addition method.*
- const `numeric sub` (const `numeric` &other) const  
*Numerical subtraction method.*
- const `numeric mul` (const `numeric` &other) const  
*Numerical multiplication method.*
- const `numeric div` (const `numeric` &other) const  
*Numerical division method.*
- const `numeric power` (const `numeric` &other) const  
*Numerical exponentiation.*
- const `numeric & add_dyn` (const `numeric` &other) const  
*Numerical addition method.*
- const `numeric & sub_dyn` (const `numeric` &other) const  
*Numerical subtraction method.*
- const `numeric & mul_dyn` (const `numeric` &other) const  
*Numerical multiplication method.*
- const `numeric & div_dyn` (const `numeric` &other) const  
*Numerical division method.*
- const `numeric & power_dyn` (const `numeric` &other) const  
*Numerical exponentiation.*
- const `numeric & operator=` (int i)
- const `numeric & operator=` (unsigned int i)
- const `numeric & operator=` (long i)
- const `numeric & operator=` (unsigned long i)
- const `numeric & operator=` (double d)
- const `numeric & operator=` (const char \*s)
- const `numeric inverse` () const  
*Inverse of a number.*
- `numeric step` () const  
*Return the step function of a numeric.*
- int `csgn` () const  
*Return the complex half-plane (left or right) in which the number lies.*
- int `compare` (const `numeric` &other) const  
*This method establishes a canonical order on all numbers.*
- bool `is_equal` (const `numeric` &other) const



- bool `is_zero` () const  
*True if object is zero.*
- bool `is_positive` () const  
*True if object is not complex and greater than zero.*
- bool `is_negative` () const  
*True if object is not complex and less than zero.*
- bool `is_integer` () const  
*True if object is a non-complex integer.*
- bool `is_pos_integer` () const  
*True if object is an exact integer greater than zero.*
- bool `is_nonneg_integer` () const  
*True if object is an exact integer greater or equal zero.*
- bool `is_even` () const  
*True if object is an exact even integer.*
- bool `is_odd` () const  
*True if object is an exact odd integer.*
- bool `is_prime` () const  
*Probabilistic primality test.*
- bool `is_rational` () const  
*True if object is an exact rational number, may even be complex (denominator may be unity).*
- bool `is_real` () const  
*True if object is a real integer, rational or float (but not complex).*
- bool `is_cinteger` () const  
*True if object is element of the domain of integers extended by  $i$ , i.e.*
- bool `is_crational` () const  
*True if object is an exact rational number, may even be complex (denominator may be unity).*
- bool `operator==` (const `numeric` &other) const
- bool `operator!=` (const `numeric` &other) const
- bool `operator<` (const `numeric` &other) const  
*Numerical comparison: less.*
- bool `operator<=` (const `numeric` &other) const  
*Numerical comparison: less or equal.*
- bool `operator>` (const `numeric` &other) const  
*Numerical comparison: greater.*
- bool `operator>=` (const `numeric` &other) const  
*Numerical comparison: greater or equal.*
- int `to_int` () const  
*Converts numeric types to machine's int.*
- long `to_long` () const  
*Converts numeric types to machine's long.*
- double `to_double` () const  
*Converts numeric types to machine's double.*
- `cln::cl_N to_cl_N` () const  
*Returns a new CLN object of type `cl_N`, representing the value of `*this`.*
- const `numeric real` () const  
*Real part of a number.*
- const `numeric imag` () const  
*Imaginary part of a number.*
- const `numeric numer` () const  
*Numerator.*
- const `numeric denom` () const

*Denominator.*

- `int int_length () const`

*Size in binary notation.*

- `numeric (const cln::cl_N &z)`

*Ctor from CLN types.*

## Protected Member Functions

- `ex derivative (const symbol &s) const` override

*Implementation of `ex::diff` for a numeric always returns 0.*

- `bool is_equal_same_type (const basic &other) const` override

*Returns true if two objects of same type are equal.*

- `unsigned calchash () const` override

*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*

- `void print_numeric (const print_context &c, const char *par_open, const char *par_close, const char *imag←_sym, const char *mul_sym, unsigned level) const`
- `void do_print (const print_context &c, unsigned level) const`
- `void do_print_latex (const print_latex &c, unsigned level) const`
- `void do_print_csrc (const print_csrc &c, unsigned level) const`
- `void do_print_csrc_cl_N (const print_csrc_cl_N &c, unsigned level) const`
- `void do_print_tree (const print_tree &c, unsigned level) const`
- `void do_print_python_repr (const print_python_repr &c, unsigned level) const`

## Protected Attributes

- `cln::cl_N value`

### 6.105.1 Detailed Description

This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.

Objects of this type may directly be created by the user.

### 6.105.2 Constructor & Destructor Documentation

#### 6.105.2.1 `numeric()` [1/10]

```
GiNaC::numeric::numeric (
    int i )
```

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `GiNaC::basic::setflag()`, and `value`.

Referenced by `add()`, `conjugate()`, `denom()`, `div()`, `evalf()`, `imag()`, `imag_part()`, `inverse()`, `mul()`, `numer()`, `operator=()`, `power()`, `real()`, `real_part()`, `step()`, and `sub()`.

**6.105.2.2 numeric()** [2/10]

```
GiNaC::numeric::numeric (
    unsigned int i )
```

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `GiNaC::basic::setflag()`, and `value`.

**6.105.2.3 numeric()** [3/10]

```
GiNaC::numeric::numeric (
    long i )
```

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `GiNaC::basic::setflag()`, and `value`.

**6.105.2.4 numeric()** [4/10]

```
GiNaC::numeric::numeric (
    unsigned long i )
```

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `GiNaC::basic::setflag()`, and `value`.

**6.105.2.5 numeric()** [5/10]

```
GiNaC::numeric::numeric (
    long long i )
```

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `GiNaC::basic::setflag()`, and `value`.

**6.105.2.6 numeric()** [6/10]

```
GiNaC::numeric::numeric (
    unsigned long long i )
```

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `GiNaC::basic::setflag()`, and `value`.

**6.105.2.7 numeric()** [7/10]

```
GiNaC::numeric::numeric (
    long numer,
    long denom )
```

Constructor for rational numerics `a/b`.

## Exceptions

<code>overflow_error</code>	(division by zero)
-----------------------------	--------------------

References `denom()`, `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `numer()`, `GiNaC::basic::setflag()`, and `value`.

### 6.105.2.8 `numeric()` [8/10]

```
GiNaC::numeric::numeric (
    double d )
```

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `GiNaC::basic::setflag()`, and `value`.

### 6.105.2.9 `numeric()` [9/10]

```
GiNaC::numeric::numeric (
    const char * s )
```

ctor from C-style string.

It also accepts complex numbers in [GiNaC](#) notation like "2+5\*I".

References `GiNaC::Digits`, `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `GiNaC::basic::setflag()`, and `value`.

### 6.105.2.10 `numeric()` [10/10]

```
GiNaC::numeric::numeric (
    const cln::cl_N & z ) [explicit]
```

Ctor from CLN types.

This is for the initiated user or internal use only.

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `GiNaC::basic::setflag()`, and `value`.

## 6.105.3 Member Function Documentation

## 6.105.3.1 precedence()

```
unsigned GiNaC::numeric::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by `print_numeric()`.

## 6.105.3.2 info()

```
bool GiNaC::numeric::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References `GiNaC::info_flags::cinteger`, `GiNaC::info_flags::cinteger_polynomial`, `GiNaC::info_flags::crational`, `GiNaC::info_flags::crational_polynomial`, `GiNaC::info_flags::even`, `GiNaC::info_flags::expanded`, `GiNaC::info_flags::integer`, `GiNaC::info_flags::integer_polynomial`, `is_cinteger()`, `is_crational()`, `is_even()`, `is_integer()`, `is_negative()`, `is_nonneg_integer()`, `is_odd()`, `is_pos_integer()`, `is_positive()`, `is_prime()`, `is_rational()`, `is_real()`, `is_zero()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::negint`, `GiNaC::info_flags::nonnegative`, `GiNaC::info_flags::nonnegint`, `GiNaC::info_flags::numeric`, `GiNaC::info_flags::odd`, `GiNaC::info_flags::polynomial`, `GiNaC::info_flags::posint`, `GiNaC::info_flags::positive`, `GiNaC::info_flags::prime`, `GiNaC::info_flags::rational`, `GiNaC::info_flags::rational_function`, `GiNaC::info_flags::rational_polynomial`, and `GiNaC::info_flags::real`.

Referenced by `GiNaC::abs_power()`, `GiNaC::csgn_power()`, `GiNaC::mul::info()`, and `GiNaC::zeta1_eval()`.

## 6.105.3.3 is\_polynomial()

```
bool GiNaC::numeric::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

#### 6.105.3.4 degree()

```
int GiNaC::numeric::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

#### 6.105.3.5 ldegree()

```
int GiNaC::numeric::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

#### 6.105.3.6 coeff()

```
ex GiNaC::numeric::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), and n.

Referenced by [GiNaC::pseries::mul\\_const\(\)](#).

#### 6.105.3.7 has()

```
bool GiNaC::numeric::has (
    const ex & other,
    unsigned options = 0 ) const [override], [virtual]
```

Disassemble real part and imaginary part to scan for the occurrence of a single number.

Also handles the imaginary unit. It ignores the sign on both this and the argument, which may lead to what might appear as funny results:  $(2+i).has(-2) \rightarrow true$ . But this is consistent, since we also would like to have  $(-2+i).has(2) \rightarrow true$  and we want to think about the sign as a multiplicative factor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_num0\\_p](#), [GiNaC::i](#), [imag\(\)](#), [is\\_equal\(\)](#), [is\\_real\(\)](#), [is\\_zero\(\)](#), and [real\(\)](#).

### 6.105.3.8 eval()

```
ex GiNaC::numeric::eval ( ) const [override], [virtual]
```

Evaluation of numbers doesn't do anything at all.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

### 6.105.3.9 evalf()

```
ex GiNaC::numeric::evalf ( ) const [override], [virtual]
```

Cast numeric into a floating-point object.

For example `exact numeric(1)` is returned as a `1.000000000000000000000000` and so on according to how `Digits` is currently set. In case the object already was a floating point number the precision is trimmed to match the currently set default.

#### Returns

an ex-handle to a numeric.

Reimplemented from [GiNaC::basic](#).

References `numeric()`, and `value`.

### 6.105.3.10 subs()

```
ex GiNaC::numeric::subs (
    const exmap & m,
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References `m`, `options`, and [GiNaC::basic::subs\\_one\\_level\(\)](#).

### 6.105.3.11 normal()

```
ex GiNaC::numeric::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for a numeric.

It splits complex numbers into  $re+I*im$  and replaces  $I$  and non-rational real numbers with a temporary symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [denom\(\)](#), [GiNaC::I](#), [imag\(\)](#), [is\\_integer\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [numer\(\)](#), [real\(\)](#), and [GiNaC::replace\\_↔with\\_symbol\(\)](#).

### 6.105.3.12 to\_rational()

```
ex GiNaC::numeric::to_rational (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for a numeric.

It splits complex numbers into  $re+I*im$  and replaces  $I$  and non-rational real numbers with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::I](#), [imag\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [real\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

### 6.105.3.13 to\_polynomial()

```
ex GiNaC::numeric::to_polynomial (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for a numeric.

It splits complex numbers into  $re+I*im$  and replaces  $I$  and non-integer real numbers with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::I](#), [imag\(\)](#), [is\\_integer\(\)](#), [is\\_real\(\)](#), [real\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).



#### 6.105.3.14 integer\_content()

```
numeric GiNaC::numeric::integer_content ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#).

#### 6.105.3.15 smod()

```
ex GiNaC::numeric::smod (
    const numeric & xi ) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

**Parameters**

$xi$	modulus
------	---------

**Returns**

mapped polynomial

**See also**

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::smod\(\)](#).

**6.105.3.16 max\_coefficient()**

```
numeric GiNaC::numeric::max_coefficient ( ) const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

**See also**

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#).

**6.105.3.17 conjugate()**

```
ex GiNaC::numeric::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::conjugate\(\)](#), [is\\_real\(\)](#), [numeric\(\)](#), and [value](#).

**6.105.3.18 real\_part()**

```
ex GiNaC::numeric::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [numeric\(\)](#), and [value](#).

## 6.105.3.19 imag\_part()

`ex` GiNaC::numeric::imag\_part ( ) const [override], [virtual]

Reimplemented from [GiNaC::basic](#).

References `numeric()`, and `value`.

## 6.105.3.20 archive()

```
void GiNaC::numeric::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References `n`, `value`, and `GiNaC::write_real_float()`.

## 6.105.3.21 read\_archive()

```
void GiNaC::numeric::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References `c`, `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, `n`, `GiNaC::read_real_float()`, `GiNaC::basic::setflag()`, and `value`.

## 6.105.3.22 derivative()

```
ex GiNaC::numeric::derivative (
    const symbol & s ) const [inline], [override], [protected], [virtual]
```

Implementation of `ex::diff` for a numeric always returns 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

### 6.105.3.23 `is_equal_same_type()`

```
bool GiNaC::numeric::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls `compare_same_type()`. The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from `GiNaC::basic`.

References `GINAC_ASSERT`, and `is_equal()`.

### 6.105.3.24 `calchash()`

```
unsigned GiNaC::numeric::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.

The method inherited from class `basic` computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from `GiNaC::basic`.

References `GiNaC::golden_ratio_hash()`, `GiNaC::status_flags::hash_calculated`, `GiNaC::basic::hashvalue`, `GiNaC::basic::setflag()`, and `value`.

### 6.105.3.25 `add()`

```
const numeric GiNaC::numeric::add (
    const numeric & other ) const
```

Numerical addition method.

Adds argument to `*this` and returns result as a numeric object.

References `numeric()`, and `value`.

Referenced by `GiNaC::operator+()`, `GiNaC::operator++()`, `GiNaC::operator+=()`, and `GiNaC::operator--()`.

## 6.105.3.26 sub()

```
const numeric GiNaC::numeric::sub (
    const numeric & other ) const
```

Numerical subtraction method.

Subtracts argument from \*this and returns result as a numeric object.

References numeric(), and value.

Referenced by GiNaC::operator-(), and GiNaC::operator-=().

## 6.105.3.27 mul()

```
const numeric GiNaC::numeric::mul (
    const numeric & other ) const
```

Numerical multiplication method.

Multiplies \*this and argument and returns result as a numeric object.

References numeric(), and value.

Referenced by GiNaC::mul::eval(), GiNaC::power::eval(), GiNaC::power::expand\_add\_2(), GiNaC::lcm(), GiNaC::operator\*(), GiNaC::operator\*=( ), GiNaC::operator-(), and GiNaC::matrix::pow().

## 6.105.3.28 div()

```
const numeric GiNaC::numeric::div (
    const numeric & other ) const
```

Numerical division method.

Divides \*this by argument and returns result as a numeric object.

## Exceptions

<i>overflow_error</i>	(division by zero)
-----------------------	--------------------

References numeric(), and value.

Referenced by GiNaC::power::eval(), GiNaC::multinomial\_coefficient(), GiNaC::operator/(), GiNaC::operator/=( ), and GiNaC::basic::series().

**6.105.3.29 power()**

```
const numeric GiNaC::numeric::power (
    const numeric & other ) const
```

Numerical exponentiation.

Raises \*this to the power given as argument and returns result as a numeric object.

References GiNaC::\_num0\_p, GiNaC::\_num1\_p, numeric(), and value.

Referenced by GiNaC::binomial(), and GiNaC::power::eval().

**6.105.3.30 add\_dyn()**

```
const numeric & GiNaC::numeric::add_dyn (
    const numeric & other ) const
```

Numerical addition method.

Adds argument to \*this and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References GiNaC::\_num0\_p, and value.

**6.105.3.31 sub\_dyn()**

```
const numeric & GiNaC::numeric::sub_dyn (
    const numeric & other ) const
```

Numerical subtraction method.

Subtracts argument from \*this and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References GiNaC::\_num0\_p, and value.

**6.105.3.32 mul\_dyn()**

```
const numeric & GiNaC::numeric::mul_dyn (
    const numeric & other ) const
```

Numerical multiplication method.

Multiplies \*this and argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References GiNaC::\_num1\_p, and value.

Referenced by GiNaC::power::expand\_add\_2().

**6.105.3.33 div\_dyn()**

```
const numeric & GiNaC::numeric::div_dyn (
    const numeric & other ) const
```

Numerical division method.

Divides \*this by argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

## Exceptions

<code>overflow_error</code>	(division by zero)
-----------------------------	--------------------

References GiNaC::\_num1\_p, and value.

## 6.105.3.34 power\_dyn()

```
const numeric & GiNaC::numeric::power_dyn (
    const numeric & other ) const
```

Numerical exponentiation.

Raises \*this to the power given as argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References GiNaC::\_num0\_p, GiNaC::\_num1\_p, and value.

## 6.105.3.35 operator=() [1/6]

```
const numeric & GiNaC::numeric::operator= (
    int i )
```

References numeric().

Referenced by operator=().

## 6.105.3.36 operator=() [2/6]

```
const numeric & GiNaC::numeric::operator= (
    unsigned int i )
```

References numeric(), and operator=().

## 6.105.3.37 operator=() [3/6]

```
const numeric & GiNaC::numeric::operator= (
    long i )
```

References numeric(), and operator=().

**6.105.3.38 operator=()** [4/6]

```
const numeric & GiNaC::numeric::operator= (
    unsigned long i )
```

References numeric(), and operator=().

**6.105.3.39 operator=()** [5/6]

```
const numeric & GiNaC::numeric::operator= (
    double d )
```

References numeric(), and operator=().

**6.105.3.40 operator=()** [6/6]

```
const numeric & GiNaC::numeric::operator= (
    const char * s )
```

References numeric(), and operator=().

**6.105.3.41 inverse()**

```
const numeric GiNaC::numeric::inverse ( ) const
```

Inverse of a number.

References numeric(), and value.

Referenced by GiNaC::divide\_in\_z(), GiNaC::power::eval(), GiNaC::heur\_gcd\_z(), GiNaC::interpolate(), and GiNaC::psi1\_eval().

**6.105.3.42 step()**

```
numeric GiNaC::numeric::step ( ) const
```

Return the step function of a numeric.

The imaginary part of it is ignored because the step function is generally considered real but a numeric may develop a small imaginary part due to rounding errors.

References numeric(), r, and value.



**6.105.3.43 csgn()**

```
int GiNaC::numeric::csgn ( ) const
```

Return the complex half-plane (left or right) in which the number lies.

$\text{csgn}(x) == 0$  for  $x == 0$ ,  $\text{csgn}(x) == 1$  for  $\text{Re}(x) > 0$  or  $\text{Re}(x) = 0$  and  $\text{Im}(x) > 0$ ,  $\text{csgn}(x) == -1$  for  $\text{Re}(x) < 0$  or  $\text{Re}(x) = 0$  and  $\text{Im}(x) < 0$ .

**See also**

`numeric::compare(const numeric &other)`

References `r`, and `value`.

**6.105.3.44 compare()**

```
int GiNaC::numeric::compare (
    const numeric & other ) const
```

This method establishes a canonical order on all numbers.

For complex numbers this is not possible in a mathematically consistent way but we need to establish some order and it ought to be fast. So we simply define it to be compatible with our method `csgn`.

**Returns**

`csgn(*this-other)`

**See also**

[numeric::csgn\(\)](#)

References `value`.

Referenced by `GiNaC::power::eval()`.

**6.105.3.45 is\_equal()**

```
bool GiNaC::numeric::is_equal (
    const numeric & other ) const
```

References `value`.

Referenced by `GiNaC::cos_eval()`, `GiNaC::power::eval()`, `GiNaC::exp_eval()`, `has()`, `is_equal_same_type()`, `GiNaC::sin_eval()`, `GiNaC::tan_eval()`, and `GiNaC::zeta1_eval()`.

**6.105.3.46 is\_zero()**

```
bool GiNaC::numeric::is_zero ( ) const
```

True if object is zero.

References value.

Referenced by `GiNaC::asinh_conjugate()`, `GiNaC::atan()`, `GiNaC::atan_conjugate()`, `GiNaC::cosh_eval()`, `GiNaC::csgn_eval()`, `GiNaC::csgn_series()`, `GiNaC::power::eval()`, `GiNaC::fsolve()`, `GiNaC::gcd()`, `has()`, `info()`, `GiNaC::iquo()`, `GiNaC::irem()`, `GiNaC::matrix::pivot()`, `GiNaC::matrix::pow()`, `GiNaC::pseries::power_const()`, `GiNaC::sinh_eval()`, `GiNaC::step_eval()`, `GiNaC::step_series()`, `GiNaC::tanh_eval()`, and `GiNaC::zeta1_eval()`.

**6.105.3.47 is\_positive()**

```
bool GiNaC::numeric::is_positive ( ) const
```

True if object is not complex and greater than zero.

References value.

Referenced by `GiNaC::power::eval()`, `info()`, `GiNaC::psi1_eval()`, and `GiNaC::tgamma_eval()`.

**6.105.3.48 is\_negative()**

```
bool GiNaC::numeric::is_negative ( ) const
```

True if object is not complex and less than zero.

References value.

Referenced by `GiNaC::bernoulli()`, `GiNaC::beta_eval()`, `GiNaC::eta_eval()`, `GiNaC::eta_evalf()`, `info()`, and `GiNaC::pseries::power_const()`.

**6.105.3.49 is\_integer()**

```
bool GiNaC::numeric::is_integer ( ) const
```

True if object is a non-complex integer.

References value.

Referenced by `GiNaC::bernoulli()`, `GiNaC::beta_eval()`, `GiNaC::binomial_sym()`, `GiNaC::mul::eval()`, `GiNaC::power::eval()`, `GiNaC::gcd()`, `info()`, `GiNaC::iquo()`, `GiNaC::irem()`, `GiNaC::lcm()`, `GiNaC::mod()`, `normal()`, `GiNaC::psi1_eval()`, `GiNaC::smod()`, `GiNaC::tgamma_eval()`, `to_int()`, `to_long()`, `to_polynomial()`, and `GiNaC::zeta1_eval()`.

**6.105.3.50 is\_pos\_integer()**

```
bool GiNaC::numeric::is_pos_integer ( ) const
```

True if object is an exact integer greater than zero.

References value.

Referenced by `GiNaC::mul::eval()`, `GiNaC::power::eval()`, and `info()`.

**6.105.3.51 is\_nonneg\_integer()**

```
bool GiNaC::numeric::is_nonneg_integer ( ) const
```

True if object is an exact integer greater or equal zero.

References value.

Referenced by `GiNaC::binomial_sym()`, `info()`, and `print_numeric()`.

**6.105.3.52 is\_even()**

```
bool GiNaC::numeric::is_even ( ) const
```

True if object is an exact even integer.

References value.

Referenced by `info()`, `GiNaC::kronecker_symbol()`, and `GiNaC::tgamma_eval()`.

**6.105.3.53 is\_odd()**

```
bool GiNaC::numeric::is_odd ( ) const
```

True if object is an exact odd integer.

References value.

Referenced by `info()`, `GiNaC::is_discriminant_of_quadratic_number_field()`, and `GiNaC::matrix::pow()`.

**6.105.3.54 is\_prime()**

```
bool GiNaC::numeric::is_prime ( ) const
```

Probabilistic primality test.

**Returns**

true if object is exact integer and prime.

References value.

Referenced by info().

**6.105.3.55 is\_rational()**

```
bool GiNaC::numeric::is_rational ( ) const
```

True if object is an exact rational number, may even be complex (denominator may be unity).

References value.

Referenced by GiNaC::power::eval(), info(), GiNaC::multiply\_lcm(), normal(), and to\_rational().

**6.105.3.56 is\_real()**

```
bool GiNaC::numeric::is_real ( ) const
```

True if object is a real integer, rational or float (but not complex).

References value.

Referenced by GiNaC::atan(), GiNaC::beta\_eval(), conjugate(), GiNaC::csgn\_eval(), denom(), do\_print\_csrc(), do\_print\_csrc\_cl\_N(), GiNaC::eta\_eval(), GiNaC::eta\_evalf(), GiNaC::power::eval(), GiNaC::fsolve(), has(), info(), is\_cinteger(), is\_crational(), normal(), numer(), operator<(), operator<=(), operator>(), operator>=(), GiNaC::step\_eval(), to\_double(), to\_polynomial(), and to\_rational().

**6.105.3.57 is\_cinteger()**

```
bool GiNaC::numeric::is_cinteger ( ) const
```

True if object is element of the domain of integers extended by I, i.e.

is of the form  $a+b*I$ , where a and b are integers.

References is\_real(), and value.

Referenced by info().

**6.105.3.58 is\_crational()**

```
bool GiNaC::numeric::is_crational ( ) const
```

True if object is an exact rational number, may even be complex (denominator may be unity).

References `is_real()`, and `value`.

Referenced by `GiNaC::power::eval()`, and `info()`.

**6.105.3.59 operator==( )**

```
bool GiNaC::numeric::operator== (
    const numeric & other ) const
```

References `value`.

**6.105.3.60 operator!=( )**

```
bool GiNaC::numeric::operator!= (
    const numeric & other ) const
```

References `value`.

**6.105.3.61 operator<( )**

```
bool GiNaC::numeric::operator< (
    const numeric & other ) const
```

Numerical comparison: less.

**Exceptions**

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References `is_real()`, and `value`.

**6.105.3.62 operator<=( )**

```
bool GiNaC::numeric::operator<= (
    const numeric & other ) const
```

Numerical comparison: less or equal.

## Exceptions

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References `is_real()`, and `value`.

6.105.3.63 `operator>()`

```
bool GiNaC::numeric::operator> (
    const numeric & other ) const
```

Numerical comparison: greater.

## Exceptions

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References `is_real()`, and `value`.

6.105.3.64 `operator>=()`

```
bool GiNaC::numeric::operator>= (
    const numeric & other ) const
```

Numerical comparison: greater or equal.

## Exceptions

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References `is_real()`, and `value`.

6.105.3.65 `to_int()`

```
int GiNaC::numeric::to_int ( ) const
```

Converts numeric types to machine's int.

You should check with `is_integer()` if the number is really an integer before calling this method. You may also consider checking the range first.

References `GINAC_ASSERT`, `is_integer()`, and `value`.

Referenced by `GiNaC::bernoulli()`, `GiNaC::binomial_sym()`, `GiNaC::power::eval()`, `GiNaC::power::expand()`, `GiNaC::C::Kronecker_dtau_kernel::series_coeff_impl()`, and `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`.

**6.105.3.66 to\_long()**

```
long GiNaC::numeric::to_long ( ) const
```

Converts numeric types to machine's long.

You should check with [is\\_integer\(\)](#) if the number is really an integer before calling this method. You may also consider checking the range first.

References GINAC\_ASSERT, is\_integer(), and value.

Referenced by GiNaC::power::expand(), and GiNaC::power::expand\_add().

**6.105.3.67 to\_double()**

```
double GiNaC::numeric::to_double ( ) const
```

Converts numeric types to machine's double.

You should check with [is\\_real\(\)](#) if the number is really not complex before calling this method.

References GINAC\_ASSERT, is\_real(), and value.

**6.105.3.68 to\_cl\_N()**

```
cln::cl_N GiNaC::numeric::to_cl_N ( ) const
```

Returns a new CLN object of type cl\_N, representing the value of \*this.

This method may be used when mixing [GiNaC](#) and CLN in one project.

References value.

Referenced by GiNaC::atan(), GiNaC::gcd(), GiNaC::iquo(), GiNaC::irem(), GiNaC::lcm(), GiNaC::mod(), and GiNaC::smod().

**6.105.3.69 real()**

```
const numeric GiNaC::numeric::real ( ) const
```

Real part of a number.

References numeric(), and value.

Referenced by GiNaC::csgn\_eval(), GiNaC::power::eval(), has(), normal(), GiNaC::pseries::power\_const(), GiNaC::step\_eval(), to\_polynomial(), and to\_rational().



**6.105.3.70 imag()**

```
const numeric GiNaC::numeric::imag ( ) const
```

Imaginary part of a number.

References numeric(), and value.

Referenced by GiNaC::csgn\_eval(), has(), normal(), GiNaC::step\_eval(), to\_polynomial(), and to\_rational().

**6.105.3.71 numer()**

```
const numeric GiNaC::numeric::numer ( ) const
```

Numerator.

Computes the numerator of rational numbers, rationalized numerator of complex if real and imaginary part are both rational numbers (i.e `numer(4/3+5/6*I) == 8+5*I`), the number carrying the sign in all other cases.

References is\_real(), GiNaC::lcm(), numeric(), r, and value.

Referenced by GiNaC::power::eval(), GiNaC::frac\_cancel(), normal(), and numeric().

**6.105.3.72 denom()**

```
const numeric GiNaC::numeric::denom ( ) const
```

Denominator.

Computes the denominator of rational numbers, common integer denominator of complex if real and imaginary part are both rational numbers (i.e `denom(4/3+5/6*I) == 6`), one in all other cases.

References GiNaC::\_num1\_p, is\_real(), GiNaC::lcm(), numeric(), r, and value.

Referenced by GiNaC::power::eval(), GiNaC::frac\_cancel(), normal(), and numeric().

**6.105.3.73 int\_length()**

```
int GiNaC::numeric::int_length ( ) const
```

Size in binary notation.

For integers, this is the smallest  $n \geq 0$  such that  $-2^n \leq x < 2^n$ . If  $x > 0$ , this is the unique  $n > 0$  such that  $2^{n-1} \leq x < 2^n$ .

**Returns**

number of bits (excluding sign) needed to represent that number in two's complement if it is an integer, 0 otherwise.

References value.

Referenced by GiNaC::heur\_gcd\_z().

#### 6.105.3.74 print\_numeric()

```
void GiNaC::numeric::print_numeric (
    const print\_context & c,
    const char * par_open,
    const char * par_close,
    const char * imag_sym,
    const char * mul_sym,
    unsigned level ) const [protected]
```

References [c](#), [is\\_nonneg\\_integer\(\)](#), [precedence\(\)](#), [GiNaC::print\\_real\\_number\(\)](#), [r](#), [GiNaC::print\\_context::s](#), and [value](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\\_repr\(\)](#).

#### 6.105.3.75 do\_print()

```
void GiNaC::numeric::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_numeric\(\)](#).

#### 6.105.3.76 do\_print\_latex()

```
void GiNaC::numeric::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_numeric\(\)](#).

#### 6.105.3.77 do\_print\_csrc()

```
void GiNaC::numeric::do_print_csrc (
    const print\_csrc & c,
    unsigned level ) const [protected]
```

References [c](#), [is\\_real\(\)](#), [GiNaC::print\\_real\\_csrc\(\)](#), and [value](#).

**6.105.3.78 do\_print\_csrc\_cl\_N()**

```
void GiNaC::numeric::do_print_csrc_cl_N (
    const print\_csrc\_cl\_N & c,
    unsigned level ) const [protected]
```

References [c](#), [is\\_real\(\)](#), [GiNaC::print\\_real\\_cl\\_N\(\)](#), and [value](#).

**6.105.3.79 do\_print\_tree()**

```
void GiNaC::numeric::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), and [value](#).

**6.105.3.80 do\_print\_python\_repr()**

```
void GiNaC::numeric::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_numeric\(\)](#).

**6.105.4 Member Data Documentation****6.105.4.1 value**

```
cln::cl_N GiNaC::numeric::value [protected]
```

Referenced by [add\(\)](#), [add\\_dyn\(\)](#), [archive\(\)](#), [calchash\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [csgn\(\)](#), [denom\(\)](#), [div\(\)](#), [div\\_dyn\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [do\\_print\\_tree\(\)](#), [evalf\(\)](#), [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_C↵LASS\\_OPT\(\)](#), [imag\(\)](#), [imag\\_part\(\)](#), [int\\_length\(\)](#), [inverse\(\)](#), [is\\_cinteger\(\)](#), [is\\_crational\(\)](#), [is\\_equal\(\)](#), [is\\_even\(\)](#), [is↵\\_integer\(\)](#), [is\\_negative\(\)](#), [is\\_nonneg\\_integer\(\)](#), [is\\_odd\(\)](#), [is\\_pos\\_integer\(\)](#), [is\\_positive\(\)](#), [is\\_prime\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [is\\_zero\(\)](#), [mul\(\)](#), [mul\\_dyn\(\)](#), [number\(\)](#), [numeric\(\)](#), [operator!=\(\)](#), [operator<\(\)](#), [operator<=\(\)](#), [operator==\(\)](#), [operator>\(\)](#), [operator>=\(\)](#), [power\(\)](#), [power\\_dyn\(\)](#), [print\\_numeric\(\)](#), [read\\_archive\(\)](#), [real\(\)](#), [real\\_part\(\)](#), [step\(\)](#), [sub\(\)](#), [sub\\_dyn\(\)](#), [to\\_cl\\_N\(\)](#), [to\\_double\(\)](#), [to\\_int\(\)](#), and [to\\_long\(\)](#).

The documentation for this class was generated from the following files:

- [numeric.h](#)
- [normal.cpp](#)
- [numeric.cpp](#)

## 6.106 GiNaC::op0\_is\_equal Struct Reference

```
#include <ex.h>
```

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

### 6.106.1 Member Function Documentation

#### 6.106.1.1 operator()

```
bool GiNaC::op0_is_equal::operator() (
    const ex & lh,
    const ex & rh ) const [inline]
```

References [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

The documentation for this struct was generated from the following file:

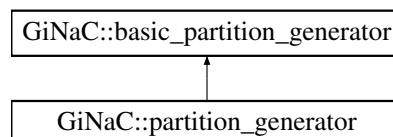
- [ex.h](#)

## 6.107 GiNaC::partition\_generator Class Reference

Generate all bounded combinatorial partitions of an integer n with exactly m parts (not including zero parts) in non-decreasing order.

```
#include <utils.h>
```

Inheritance diagram for `GiNaC::partition_generator`:



### Public Member Functions

- [partition\\_generator](#) (unsigned n\_, unsigned m\_)
- const std::vector< unsigned > & [get](#) () const
- bool [next](#) ()

## Private Attributes

- `std::vector< unsigned > partition`
- `bool current_updated`

## Additional Inherited Members

### 6.107.1 Detailed Description

Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (not including zero parts) in non-decreasing order.

### 6.107.2 Constructor & Destructor Documentation

#### 6.107.2.1 partition\_generator()

```
GiNaC::partition_generator::partition_generator (
    unsigned n_,
    unsigned m_ ) [inline]
```

### 6.107.3 Member Function Documentation

#### 6.107.3.1 get()

```
const std::vector<unsigned>& GiNaC::partition_generator::get ( ) const [inline]
```

References `current_updated`, `GiNaC::basic_partition_generator::mpartition2::m`, `GiNaC::basic_partition_generator::mpgen`, `partition`, and `GiNaC::basic_partition_generator::mpartition2::x`.

#### 6.107.3.2 next()

```
bool GiNaC::partition_generator::next ( ) [inline]
```

References `current_updated`, `GiNaC::basic_partition_generator::mpgen`, and `GiNaC::basic_partition_generator::mpartition2::next_partition()`.

### 6.107.4 Member Data Documentation

#### 6.107.4.1 partition

```
std::vector<unsigned> GiNaC::partition_generator::partition [mutable], [private]
```

Referenced by `get()`.

#### 6.107.4.2 current\_updated

```
bool GiNaC::partition_generator::current_updated [mutable], [private]
```

Referenced by `get()`, and `next()`.

The documentation for this class was generated from the following file:

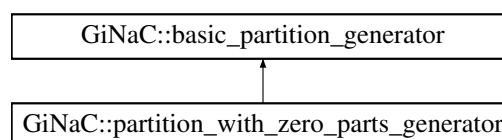
- [utils.h](#)

### 6.108 GiNaC::partition\_with\_zero\_parts\_generator Class Reference

Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (including zero parts) in non-decreasing order.

```
#include <utils.h>
```

Inheritance diagram for `GiNaC::partition_with_zero_parts_generator`:



#### Public Member Functions

- [partition\\_with\\_zero\\_parts\\_generator](#) (unsigned  $n$ \_, unsigned  $m$ \_)
- const std::vector< unsigned > & [get](#) () const
- bool [next](#) ()

#### Private Attributes

- unsigned [m](#)
- std::vector< unsigned > [partition](#)
- bool [current\\_updated](#)

## Additional Inherited Members

### 6.108.1 Detailed Description

Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (including zero parts) in non-decreasing order.

### 6.108.2 Constructor & Destructor Documentation

#### 6.108.2.1 partition\_with\_zero\_parts\_generator()

```
GiNaC::partition_with_zero_parts_generator::partition_with_zero_parts_generator (
    unsigned n_,
    unsigned m_ ) [inline]
```

### 6.108.3 Member Function Documentation

#### 6.108.3.1 get()

```
const std::vector<unsigned>& GiNaC::partition_with_zero_parts_generator::get ( ) const [inline]
```

References `current_updated`, `GiNaC::basic_partition_generator::mpartition2::m`, `m`, `GiNaC::basic_partition_generator::mpgen`, `partition`, and `GiNaC::basic_partition_generator::mpartition2::x`.

Referenced by `GiNaC::power::expand_add()`.

#### 6.108.3.2 next()

```
bool GiNaC::partition_with_zero_parts_generator::next ( ) [inline]
```

References `current_updated`, `GiNaC::basic_partition_generator::mpartition2::m`, `m`, `GiNaC::basic_partition_generator::mpgen`, `GiNaC::basic_partition_generator::mpartition2::n`, and `GiNaC::basic_partition_generator::mpartition2::next_partition()`.

### 6.108.4 Member Data Documentation

## 6.108.4.1 m

```
unsigned GiNaC::partition_with_zero_parts_generator::m [private]
```

Referenced by `get()`, and `next()`.

## 6.108.4.2 partition

```
std::vector<unsigned> GiNaC::partition_with_zero_parts_generator::partition [mutable], [private]
```

Referenced by `get()`.

## 6.108.4.3 current\_updated

```
bool GiNaC::partition_with_zero_parts_generator::current_updated [mutable], [private]
```

Referenced by `get()`, and `next()`.

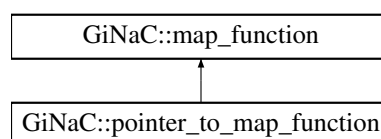
The documentation for this class was generated from the following file:

- [utils.h](#)

## 6.109 GiNaC::pointer\_to\_map\_function Class Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_map_function`:



## Public Member Functions

- [pointer\\_to\\_map\\_function](#) (`ex x(const ex &)`)
- [ex operator\(\)](#) (`const ex &e`) override

## Protected Attributes

- [ex\(\\* ptr\)](#) (`const ex &`)



## Additional Inherited Members

## 6.109.1 Constructor &amp; Destructor Documentation

## 6.109.1.1 pointer\_to\_map\_function()

```
GiNaC::pointer_to_map_function::pointer_to_map_function (
    ex xconst ex & ) [inline], [explicit]
```

## 6.109.2 Member Function Documentation

## 6.109.2.1 operator&gt;()

```
ex GiNaC::pointer_to_map_function::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [ptr](#).

## 6.109.3 Member Data Documentation

## 6.109.3.1 ptr

```
ex(* GiNaC::pointer_to_map_function::ptr) (const ex &) [protected]
```

Referenced by [operator>\(\)](#).

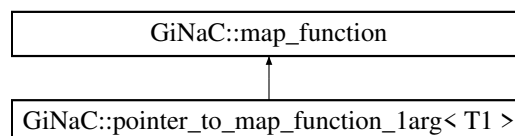
The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.110 GiNaC::pointer\_to\_map\_function\_1arg&lt; T1 &gt; Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_map\_function\_1arg< T1 >:



## Public Member Functions

- [pointer\\_to\\_map\\_function\\_1arg](#) ([ex](#) x(const [ex](#) &, T1), T1 a1)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Protected Attributes

- [ex](#)(\* [ptr](#))(const [ex](#) &, T1)
- T1 [arg1](#)

## Additional Inherited Members

### 6.110.1 Constructor & Destructor Documentation

#### 6.110.1.1 [pointer\\_to\\_map\\_function\\_1arg\(\)](#)

```
template<class T1>
GiNaC::pointer_to_map_function_larg< T1 >::pointer_to_map_function_larg (
    ex xconst ex &, T1,
    T1 a1 ) [inline], [explicit]
```

### 6.110.2 Member Function Documentation

#### 6.110.2.1 [operator\(\)](#)

```
template<class T1>
ex GiNaC::pointer_to_map_function_larg< T1 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >::arg1](#), and [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >::ptr](#).

### 6.110.3 Member Data Documentation

## 6.110.3.1 ptr

```
template<class T1>
ex(* GiNaC::pointer_to_map_function_larg< T1 >::ptr) (const ex &, T1) [protected]
```

Referenced by GiNaC::pointer\_to\_map\_function\_1arg< T1 >::operator()().

## 6.110.3.2 arg1

```
template<class T1>
T1 GiNaC::pointer_to_map_function_larg< T1 >::arg1 [protected]
```

Referenced by GiNaC::pointer\_to\_map\_function\_1arg< T1 >::operator()().

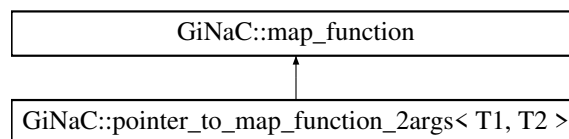
The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.111 GiNaC::pointer\_to\_map\_function\_2args&lt; T1, T2 &gt; Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >:



## Public Member Functions

- [pointer\\_to\\_map\\_function\\_2args](#) ([ex](#) x(const [ex](#) &, T1, T2), T1 a1, T2 a2)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Protected Attributes

- [ex\(\\* ptr\)](#)(const [ex](#) &, T1, T2)
- T1 [arg1](#)
- T2 [arg2](#)

## Additional Inherited Members

## 6.111.1 Constructor &amp; Destructor Documentation

#### 6.111.1.1 `pointer_to_map_function_2args()`

```
template<class T1 , class T2 >
GiNaC::pointer_to_map_function_2args< T1, T2 >::pointer_to_map_function_2args (
    ex xconst ex &, T1, T2,
    T1 a1,
    T2 a2 ) [inline], [explicit]
```

### 6.111.2 Member Function Documentation

#### 6.111.2.1 `operator()`

```
template<class T1 , class T2 >
ex GiNaC::pointer_to_map_function_2args< T1, T2 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::arg1](#), [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::arg2](#), and [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::ptr](#).

### 6.111.3 Member Data Documentation

#### 6.111.3.1 `ptr`

```
template<class T1 , class T2 >
ex(* GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >::ptr) (const ex &, T1, T2) [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::operator\(\)](#).

#### 6.111.3.2 `arg1`

```
template<class T1 , class T2 >
T1 GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::operator\(\)](#).

## 6.111.3.3 arg2

```
template<class T1 , class T2 >
T2 GiNaC::pointer_to_map_function_2args< T1, T2 >::arg2 [protected]
```

Referenced by `GiNaC::pointer_to_map_function_2args< T1, T2 >::operator()()`.

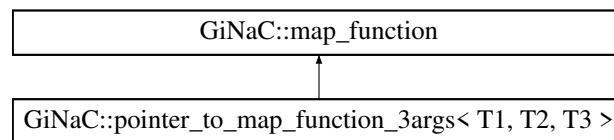
The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.112 GiNaC::pointer\_to\_map\_function\_3args&lt; T1, T2, T3 &gt; Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_map_function_3args< T1, T2, T3 >`:



## Public Member Functions

- `pointer_to_map_function_3args` (`ex x(const ex &, T1, T2, T3), T1 a1, T2 a2, T3 a3`)
- `ex operator()` (`const ex &e`) override

## Protected Attributes

- `ex(* ptr)(const ex &, T1, T2, T3)`
- `T1 arg1`
- `T2 arg2`
- `T3 arg3`

## Additional Inherited Members

## 6.112.1 Constructor &amp; Destructor Documentation

## 6.112.1.1 pointer\_to\_map\_function\_3args()

```
template<class T1 , class T2 , class T3 >
GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::pointer_to_map_function_3args (
    ex xconst ex &, T1, T2, T3,
    T1 a1,
    T2 a2,
    T3 a3 ) [inline], [explicit]
```

## 6.112.2 Member Function Documentation

### 6.112.2.1 operator()

```
template<class T1 , class T2 , class T3 >
ex GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::arg1](#), [GiNaC::pointer\\_to\\_map\\_function\\_↵  
3args< T1, T2, T3 >::arg2](#), [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::arg3](#), and [GiNaC::pointer\\_↵  
to\\_map\\_function\\_3args< T1, T2, T3 >::ptr](#).

## 6.112.3 Member Data Documentation

### 6.112.3.1 ptr

```
template<class T1 , class T2 , class T3 >
ex(* GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >::ptr) (const ex &, T1, T2, T3) [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::operator\(\)](#).

### 6.112.3.2 arg1

```
template<class T1 , class T2 , class T3 >
T1 GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::operator\(\)](#).

### 6.112.3.3 arg2

```
template<class T1 , class T2 , class T3 >
T2 GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >::arg2 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::operator\(\)](#).

## 6.112.3.4 arg3

```
template<class T1 , class T2 , class T3 >
T3 GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::arg3 [protected]
```

Referenced by `GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::operator()`.

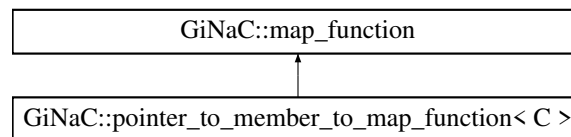
The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.113 GiNaC::pointer\_to\_member\_to\_map\_function&lt; C &gt; Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_member_to_map_function< C >`:



## Public Member Functions

- [pointer\\_to\\_member\\_to\\_map\\_function](#) (`ex(C::*member)(const ex &), C &obj`)
- [ex operator\(\)](#) (`const ex &e`) override

## Protected Attributes

- [ex\(C::\\* ptr\)](#) (`const ex &`)
- `C & c`

## Additional Inherited Members

## 6.113.1 Constructor &amp; Destructor Documentation

## 6.113.1.1 pointer\_to\_member\_to\_map\_function()

```
template<class C >
GiNaC::pointer_to_member_to_map_function< C >::pointer_to_member_to_map_function (
    ex(C::*)(const ex &) member,
    C & obj ) [inline], [explicit]
```

### 6.113.2 Member Function Documentation

#### 6.113.2.1 operator()

```
template<class C >
ex GiNaC::pointer_to_member_to_map_function< C >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >::c](#).

### 6.113.3 Member Data Documentation

#### 6.113.3.1 ptr

```
template<class C >
ex(C::* GiNaC::pointer\_to\_member\_to\_map\_function< C >::ptr) (const ex &) [protected]
```

#### 6.113.3.2 c

```
template<class C >
C& GiNaC::pointer\_to\_member\_to\_map\_function< C >::c [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >::operator\(\)](#).

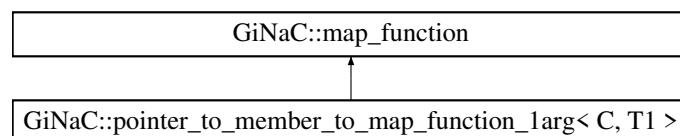
The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.114 GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >](#):





## Public Member Functions

- [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg](#) ([ex](#)(C::\*[member](#))(const [ex](#) &, T1), C &[obj](#), T1 [a1](#))
- [ex operator\(\)](#) (const [ex](#) &[e](#)) override

## Protected Attributes

- [ex](#)(C::\* [ptr](#) )(const [ex](#) &, T1)
- C & [c](#)
- T1 [arg1](#)

## Additional Inherited Members

### 6.114.1 Constructor & Destructor Documentation

#### 6.114.1.1 [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg\(\)](#)

```
template<class C , class T1 >
GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::pointer_to_member_to_map_function_1arg
(
    ex(C::*)(const ex &, T1) member,
    C & obj,
    T1 a1 ) [inline], [explicit]
```

### 6.114.2 Member Function Documentation

#### 6.114.2.1 [operator\(\)](#)

```
template<class C , class T1 >
ex GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >::arg1](#), and [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >::c](#).

### 6.114.3 Member Data Documentation

## 6.114.3.1 ptr

```
template<class C , class T1 >
ex(C::* GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::ptr) (const ex &, T1) [protected]
```

## 6.114.3.2 c

```
template<class C , class T1 >
C& GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::c [protected]
```

Referenced by `GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::operator>()()`.

## 6.114.3.3 arg1

```
template<class C , class T1 >
T1 GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::arg1 [protected]
```

Referenced by `GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::operator>()()`.

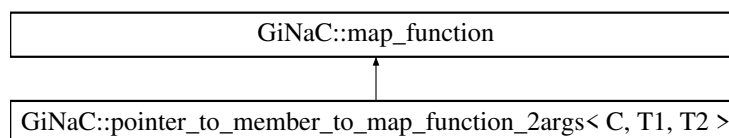
The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.115 GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >`:



### Public Member Functions

- [pointer\\_to\\_member\\_to\\_map\\_function\\_2args](#) (`ex(C::*member)(const ex &, T1, T2), C &obj, T1 a1, T2 a2)`
- [ex operator\(\)](#) (`(const ex &e)` override)

## Protected Attributes

- `ex(C::* ptr)(const ex &, T1, T2)`
- `C & c`
- `T1 arg1`
- `T2 arg2`

## Additional Inherited Members

### 6.115.1 Constructor & Destructor Documentation

#### 6.115.1.1 pointer\_to\_member\_to\_map\_function\_2args()

```
template<class C , class T1 , class T2 >
GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::pointer_to_member_to_map_function_2args
(
    ex(C::*)(const ex &, T1, T2) member,
    C & obj,
    T1 a1,
    T2 a2 ) [inline], [explicit]
```

### 6.115.2 Member Function Documentation

#### 6.115.2.1 operator()

```
template<class C , class T1 , class T2 >
ex GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::arg1](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::arg2](#), and [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::c](#).

### 6.115.3 Member Data Documentation

## 6.115.3.1 ptr

```
template<class C , class T1 , class T2 >
ex(C::* GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::ptr) (const ex &, T1, T2)
[protected]
```

## 6.115.3.2 c

```
template<class C , class T1 , class T2 >
C& GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::c [protected]
```

Referenced by GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >::operator()().

## 6.115.3.3 arg1

```
template<class C , class T1 , class T2 >
T1 GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::arg1 [protected]
```

Referenced by GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >::operator()().

## 6.115.3.4 arg2

```
template<class C , class T1 , class T2 >
T2 GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::arg2 [protected]
```

Referenced by GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >::operator()().

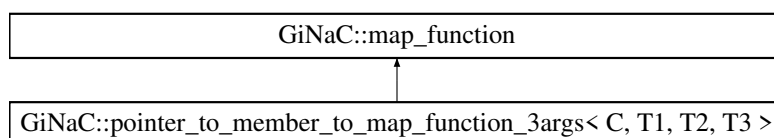
The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.116 GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >:



## Public Member Functions

- [pointer\\_to\\_member\\_to\\_map\\_function\\_3args](#) ([ex](#)(C::[\\*member](#))(const [ex](#) &, T1, T2, T3), C &[obj](#), T1 [a1](#), T2 [a2](#), T3 [a3](#))
- [ex operator\(\)](#) (const [ex](#) &[e](#)) override

## Protected Attributes

- [ex](#)(C::[\\* ptr](#))(const [ex](#) &, T1, T2, T3)
- C & [c](#)
- T1 [arg1](#)
- T2 [arg2](#)
- T3 [arg3](#)

## Additional Inherited Members

### 6.116.1 Constructor & Destructor Documentation

#### 6.116.1.1 pointer\_to\_member\_to\_map\_function\_3args()

```
template<class C , class T1 , class T2 , class T3 >
GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::pointer_to_member_to_map_function_3args
(
    ex(C::\*ptr)(const ex &, T1, T2, T3) member,
    C & obj,
    T1 a1,
    T2 a2,
    T3 a3 ) [inline], [explicit]
```

### 6.116.2 Member Function Documentation

#### 6.116.2.1 operator()

```
template<class C , class T1 , class T2 , class T3 >
ex GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::arg1](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::arg2](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::arg3](#), and [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::c](#).

### 6.116.3 Member Data Documentation

#### 6.116.3.1 ptr

```
template<class C , class T1 , class T2 , class T3 >
ex(C::* GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::ptr) (const ex &, T1,
T2, T3) [protected]
```

#### 6.116.3.2 c

```
template<class C , class T1 , class T2 , class T3 >
C& GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::c [protected]
```

Referenced by `GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::operator()()`.

#### 6.116.3.3 arg1

```
template<class C , class T1 , class T2 , class T3 >
T1 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg1 [protected]
```

Referenced by `GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::operator()()`.

#### 6.116.3.4 arg2

```
template<class C , class T1 , class T2 , class T3 >
T2 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg2 [protected]
```

Referenced by `GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::operator()()`.

#### 6.116.3.5 arg3

```
template<class C , class T1 , class T2 , class T3 >
T3 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg3 [protected]
```

Referenced by `GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::operator()()`.

The documentation for this class was generated from the following file:

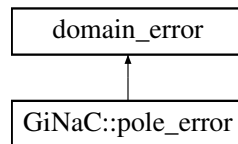
- [ex.h](#)

## 6.117 GiNaC::pole\_error Class Reference

Exception class thrown when a singularity is encountered.

```
#include <numeric.h>
```

Inheritance diagram for GiNaC::pole\_error:



### Public Member Functions

- `pole_error` (const std::string &what\_arg, int degree)  
*ctor for `pole_error` exception class.*
- int `degree` () const  
*Return the degree of the `pole_error` exception class.*

### Private Attributes

- int `deg`

#### 6.117.1 Detailed Description

Exception class thrown when a singularity is encountered.

#### 6.117.2 Constructor & Destructor Documentation

##### 6.117.2.1 pole\_error()

```
GiNaC::pole_error::pole_error (  
    const std::string & what_arg,  
    int degree ) [explicit]
```

ctor for `pole_error` exception class.

#### 6.117.3 Member Function Documentation

### 6.117.3.1 degree()

```
int GiNaC::pole_error::degree ( ) const
```

Return the degree of the [pole\\_error](#) exception class.

References [deg](#).

## 6.117.4 Member Data Documentation

### 6.117.4.1 deg

```
int GiNaC::pole_error::deg [private]
```

Referenced by [degree\(\)](#).

The documentation for this class was generated from the following files:

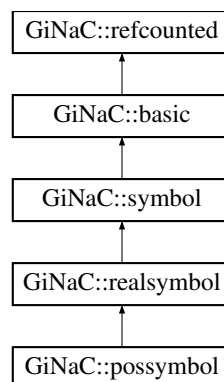
- [numeric.h](#)
- [utils.cpp](#)

## 6.118 GiNaC::possymbol Class Reference

Specialization of [symbol](#) to real positive domain.

```
#include <symbol.h>
```

Inheritance diagram for `GiNaC::possymbol`:



### Public Member Functions

- [possymbol](#) ()
- [possymbol](#) (const std::string &initname)
- [possymbol](#) (const std::string &initname, const std::string &texname)
- unsigned [get\\_domain](#) () const override
- [possymbol](#) \* [duplicate](#) () const override

*Create a clone of this object on the heap.*



## Additional Inherited Members

### 6.118.1 Detailed Description

Specialization of symbol to real positive domain.

### 6.118.2 Constructor & Destructor Documentation

#### 6.118.2.1 possymbol() [1/3]

```
GiNaC::possymbol::possymbol ( )
```

Referenced by duplicate().

#### 6.118.2.2 possymbol() [2/3]

```
GiNaC::possymbol::possymbol (
    const std::string & initname ) [explicit]
```

#### 6.118.2.3 possymbol() [3/3]

```
GiNaC::possymbol::possymbol (
    const std::string & initname,
    const std::string & texname )
```

### 6.118.3 Member Function Documentation

#### 6.118.3.1 get\_domain()

```
unsigned GiNaC::possymbol::get_domain ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::realsymbol](#).

References [GiNaC::domain::positive](#).

### 6.118.3.2 duplicate()

```
possymbol* GiNaC::possymbol::duplicate ( ) const [inline], [override], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the refcounted construction of an ex from a basic.

Reimplemented from [GiNaC::realsymbol](#).

References [GiNaC::status\\_flags::dynallocated](#), [possymbol\(\)](#), and [GiNaC::basic::setflag\(\)](#).

The documentation for this class was generated from the following files:

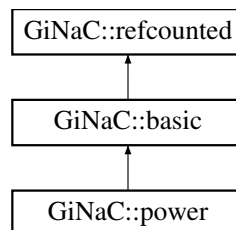
- [symbol.h](#)
- [symbol.cpp](#)

## 6.119 GiNaC::power Class Reference

This class holds a two-component object, a basis and an exponent representing exponentiation.

```
#include <power.h>
```

Inheritance diagram for [GiNaC::power](#):



### Public Member Functions

- [power](#) (const [ex](#) &lh, const [ex](#) &rh)
- `template<typename T >`  
[power](#) (const [ex](#) &lh, const T &rh)
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- size\_t [nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex map](#) ([map\\_function](#) &f) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- bool [is\\_polynomial](#) (const [ex](#) &var) const override

- Check whether this is a polynomial in the given variables.*

  - `int degree` (const `ex` &`s`) const override
  - Return degree of highest power in object `s`.*
  - `int ldegree` (const `ex` &`s`) const override
  - Return degree of lowest power in object `s`.*
  - `ex coeff` (const `ex` &`s`, int `n=1`) const override
  - Return coefficient of degree `n` in object `s`.*
  - `ex eval` () const override
  - Perform automatic term rewriting rules in this class.*
  - `ex evalf` () const override
  - Evaluate object numerically.*
  - `ex evalm` () const override
  - Evaluate sums, products and integer powers of matrices.*
  - `ex series` (const `relational` &`s`, int `order`, unsigned `options=0`) const override
  - Implementation of `ex::series()` for powers.*
  - `ex subs` (const `exmap` &`m`, unsigned `options=0`) const override
  - Substitute a set of objects by arbitrary expressions.*
  - `bool has` (const `ex` &`other`, unsigned `options=0`) const override
  - Test for occurrence of a pattern.*
  - `ex normal` (`exmap` &`repl`, `exmap` &`rev_lookup`, `lst` &`modifier`) const override
  - Implementation of `ex::normal()` for powers.*
  - `ex to_rational` (`exmap` &`repl`) const override
  - Implementation of `ex::to_rational()` for powers.*
  - `ex to_polynomial` (`exmap` &`repl`) const override
  - Implementation of `ex::to_polynomial()` for powers.*
  - `ex conjugate` () const override
  - `ex real_part` () const override
  - `ex imag_part` () const override
  - `void archive` (`archive_node` &`n`) const override
  - Save (a.k.a.*
  - `void read_archive` (const `archive_node` &`n`, `lst` &`syms`) override
  - Read (a.k.a.*

## Protected Member Functions

- `ex derivative` (const `symbol` &`s`) const override
- Implementation of `ex::diff()` for a power.*
- `ex eval_ncmul` (const `exvector` &`v`) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex expand` (unsigned `options=0`) const override
- Expand expression, i.e.*
- `void print_power` (const `print_context` &`c`, const char \*`powersymbol`, const char \*`openbrace`, const char \*`closebrace`, unsigned `level`) const
- `void do_print_dflt` (const `print_dflt` &`c`, unsigned `level`) const
- `void do_print_latex` (const `print_latex` &`c`, unsigned `level`) const
- `void do_print_csrc` (const `print_csrc` &`c`, unsigned `level`) const
- `void do_print_python` (const `print_python` &`c`, unsigned `level`) const
- `void do_print_python_repr` (const `print_python_repr` &`c`, unsigned `level`) const
- `void do_print_csrc_cl_N` (const `print_csrc_cl_N` &`c`, unsigned `level`) const

## Static Protected Member Functions

- static `ex expand_add` (const `add` &a, long `n`, unsigned `options`)  
*expand  $a^n$  where  $a$  is an add and  $n$  is a positive integer.*
- static `ex expand_add_2` (const `add` &a, unsigned `options`)  
*Special case of `power::expand_add`.*
- static `ex expand_mul` (const `mul` &m, const `numeric` &n, unsigned `options`, bool from\_expand=false)  
*Expand factors of  $m$  in  $m^n$  where  $m$  is a mul and  $n$  is an integer.*

## Protected Attributes

- `ex basis`
- `ex exponent`

## Friends

- class `mul`

### 6.119.1 Detailed Description

This class holds a two-component object, a basis and an exponent representing exponentiation.

### 6.119.2 Constructor & Destructor Documentation

#### 6.119.2.1 `power()` [1/2]

```
GiNaC::power::power (
    const ex & lh,
    const ex & rh ) [inline]
```

Referenced by `do_print_latex()`, `expand_add()`, `expand_add_2()`, and `subs()`.

#### 6.119.2.2 `power()` [2/2]

```
template<typename T >
GiNaC::power::power (
    const ex & lh,
    const T & rh ) [inline]
```

### 6.119.3 Member Function Documentation

## 6.119.3.1 precedence()

```
unsigned GiNaC::power::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by `print_power()`.

## 6.119.3.2 info()

```
bool GiNaC::power::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References `basis`, `GiNaC::info_flags::integer_polynomial`, `GiNaC::basic::clearflag()`, `GiNaC::info_flags::crational`, `polynomial`, `GiNaC::info_flags::even`, `GiNaC::status_flags::expanded`, `GiNaC::info_flags::expanded`, `exponent`, `GiNaC::basic::flags`, `GiNaC::status_flags::has_indices`, `GiNaC::info_flags::has_indices`, `GiNaC::status_flags::has_no_indices`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `GiNaC::info_flags::integer_polynomial`, `GiNaC::info_flags::nonnegative`, `GiNaC::info_flags::nonnegint`, `GiNaC::info_flags::polynomial`, `GiNaC::info_flags::positive`, `GiNaC::info_flags::rational_function`, `GiNaC::info_flags::rational_polynomial`, `GiNaC::info_flags::real`, and `GiNaC::basic::setflag()`.

## 6.119.3.3 nops()

```
size_t GiNaC::power::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

## 6.119.3.4 op()

```
ex GiNaC::power::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position `i`.

Reimplemented from [GiNaC::basic](#).

References `basis`, `exponent`, and `GINAC_ASSERT`.

### 6.119.3.5 map()

```
ex GiNaC::power::map (
    map_function & f ) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), and [exponent](#).

### 6.119.3.6 is\_polynomial()

```
bool GiNaC::power::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), [GiNaC::ex::has\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_polynomial\(\)](#), and [GiNaC::info\\_flags::nonnegint](#).

### 6.119.3.7 degree()

```
int GiNaC::power::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::ex::degree\(\)](#), [exponent](#), [GiNaC::ex::has\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::is\\_integer\(\)](#), and [GiNaC::to\\_int\(\)](#).

### 6.119.3.8 ldegree()

```
int GiNaC::power::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), [GiNaC::ex::has\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::is\\_integer\(\)](#), [GiNaC::ex::ldegree\(\)](#), and [GiNaC::to\\_int\(\)](#).

## 6.119.3.9 coeff()

```
ex GiNaC::power::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [basis](#), [exponent](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::is\\_integer\(\)](#), [n](#), and [GiNaC::to\\_int\(\)](#).

Referenced by [expand\(\)](#), and [expand\\_add\(\)](#).

## 6.119.3.10 eval()

```
ex GiNaC::power::eval ( ) const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following x, x1, x2,... stand for a symbolic variables of type ex and c, c1, c2... stand for such expressions that contain a plain number.

- $x^0 \rightarrow 1$  (also handles  $0^0$ )
- $x^1 \rightarrow x$
- $0^c \rightarrow 0$  or exception (depending on the real part of c)
- $1^x \rightarrow 1$
- $c_1^{c_2} \rightarrow c_1^n c_1^{c_2-n}$  (so that  $0 < (c_2-n) < 1$ , try to evaluate roots, possibly in numerator and denominator of c1)
- $x^{(x,c_1),c_2} \rightarrow x^{x,c_1 c_2}$  if x is positive and c1 is real.
- $x^{(x,c_1),c_2} \rightarrow x^{x,c_1 c_2}$  (c2 integer or  $-1 < c_1 \leq 1$  or  $(c_1 = -1 \text{ and } c_2 > 0)$ , case  $c_1 = 1$  should not happen, see below!)
- $(x,y,z)^c \rightarrow x^c y^c z^c$  (if c integer)
- $(x,c_1)^{c_2} \rightarrow x^{c_2} c_1^{c_2}$  ( $c_1 > 0$ )
- $(x,c_1)^{c_2} \rightarrow x^{-c_2} c_1^{c_2}$  ( $c_1 < 0$ )

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::abs\(\)](#), [basis](#), [c](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::return\\_types::commutative](#), [GiNaC::numeric::compare\(\)](#), [GiNaC::numeric::denom\(\)](#), [GiNaC::numeric::div\(\)](#), [GiNaC::status\\_flags::evaluated](#), [expand\\_mul\(\)](#), [exponent](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::integer\\_content\(\)](#), [GiNaC::numeric::inverse\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::numeric::is\\_crational\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::is\\_negative\(\)](#), [GiNaC::numeric::is\\_pos\\_integer\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [GiNaC::numeric::is\\_rational\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [likely](#), [m](#), [GiNaC::numeric::mul\(\)](#), [n](#), [GiNaC::numeric::numerator\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::numeric::power\(\)](#), [r](#), [GiNaC::numeric::real\(\)](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::return\\_type\(\)](#), [GiNaC::basic::setflag\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

### 6.119.3.11 evalf()

```
ex GiNaC::power::evalf ( ) const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::ex::evalf\(\)](#), and [exponent](#).

### 6.119.3.12 evalm()

```
ex GiNaC::power::evalm ( ) const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::ex::evalm\(\)](#), [exponent](#), and [GiNaC::pow\(\)](#).

### 6.119.3.13 series()

```
ex GiNaC::power::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for powers.

This performs Laurent expansion of reciprocals of series at singularities.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [basis](#), [GiNaC::exp\(\)](#), [GiNaC::ex::expand\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::lddegree\(\)](#), [GiNaC::log\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [r](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::ex::series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::to\\_int\(\)](#).



## 6.119.3.14 subs()

```
ex GiNaC::power::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::subs\\_options::algebraic](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [exponent](#), [m](#), [GiNaC::subs\\_↵options::no\\_pattern](#), [options](#), [GiNaC::pow\(\)](#), [power\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), and [Gi↵NaC::tryfactsubs\(\)](#).

## 6.119.3.15 has()

```
bool GiNaC::power::has (
    const ex & pattern,
    unsigned options = 0 ) const [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::has\\_options::algebraic](#), [basis](#), [exponent](#), [GiNaC::basic::has\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_↵\\_flags::integer](#), [GiNaC::ex::match\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::ex::op\(\)](#), [options](#), and [GiNaC::info\\_flags\\_↵::posint](#).

## 6.119.3.16 normal()

```
ex GiNaC::power::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for powers.

It normalizes the basis, distributes integer exponents to numerator and denominator, and replaces non-integer powers by temporary symbols.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [basis](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC\\_↵::container< C >::op\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::subs\(\)](#).

### 6.119.3.17 to\_rational()

```
ex GiNaC::power::to_rational (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for powers.

It replaces non-integer powers by temporary symbols.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::pow\(\)](#), [GiNaC::replace\\_with\\_↵symbol\(\)](#), and [GiNaC::ex::to\\_rational\(\)](#).

### 6.119.3.18 to\_polynomial()

```
ex GiNaC::power::to_polynomial (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for powers.

It replaces non-posint powers by temporary symbols.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex\\_1](#), [basis](#), [GiNaC::collect\\_common\\_factors\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_↵flags::negint](#), [GiNaC::info\\_flags::posint](#), [GiNaC::pow\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::ex::to\\_polynomial\(\)](#), and [GiNaC::ex::to\\_rational\(\)](#).

### 6.119.3.19 conjugate()

```
ex GiNaC::power::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [GiNaC::ex::conjugate\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC\\_↵::info\\_flags::integer](#), and [GiNaC::info\\_flags::positive](#).

### 6.119.3.20 real\_part()

```
ex GiNaC::power::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::abs\(\)](#), [basis](#), [GiNaC::binomial\(\)](#), [c](#), [GiNaC::cos\(\)](#), [GiNaC::exp\(\)](#), [exponent](#), [Gi\\_↵NaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::log\(\)](#), [n](#), [GiNa\\_↵C::info\\_flags::nonnegative](#), [GiNaC::numer\(\)](#), [GiNaC::pow\(\)](#), [GiNaC::ex::real\\_part\(\)](#), and [GiNaC::to\\_long\(\)](#).

6.119.3.21 `imag_part()`

```
ex GiNaC::power::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::abs\(\)](#), [basis](#), [GiNaC::binomial\(\)](#), [c](#), [GiNaC::exp\(\)](#), [exponent](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::log\(\)](#), [n](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::numer\(\)](#), [GiNaC::pow\(\)](#), [GiNaC::ex::real\\_part\(\)](#), [GiNaC::sin\(\)](#), and [GiNaC::to\\_long\(\)](#).

6.119.3.22 `archive()`

```
void GiNaC::power::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), and [n](#).

6.119.3.23 `read_archive()`

```
void GiNaC::power::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), and [n](#).

6.119.3.24 `derivative()`

```
ex GiNaC::power::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a power.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [basis](#), [GiNaC::ex::diff\(\)](#), [exponent](#), [GiNaC::log\(\)](#), and [GiNaC::pow\(\)](#).

**6.119.3.25 eval\_ncmul()**

```
ex GiNaC::power::eval_ncmul (
    const exvector & v ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.119.3.26 return\_type()**

```
unsigned GiNaC::power::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [basis](#), and [GiNaC::ex::return\\_type\(\)](#).

**6.119.3.27 return\_type\_tinfo()**

```
return\_type\_t GiNaC::power::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [basis](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#).

**6.119.3.28 expand()**

```
ex GiNaC::power::expand (
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex1](#), [GiNaC::ex\\_1](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [coeff\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\\_add\(\)](#), [expand\\_mul\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::indefinite](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_integer\(\)](#), [m](#), [GiNaC::info\\_flags::negative](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::status\\_flags::purely\\_indefinite](#), [r](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), and [GiNaC::numeric::to\\_long\(\)](#).

Referenced by [expand\\_add\(\)](#), and [expand\\_add\\_2\(\)](#).

## 6.119.3.29 print\_power()

```
void GiNaC::power::print_power (
    const print_context & c,
    const char * powersymbol,
    const char * openbrace,
    const char * closebrace,
    unsigned level ) const [protected]
```

References `basis`, `c`, `exponent`, `precedence()`, and `GiNaC::ex::print()`.

Referenced by `do_print_dflt()`, `do_print_latex()`, and `do_print_python()`.

## 6.119.3.30 do\_print\_dflt()

```
void GiNaC::power::do_print_dflt (
    const print_dflt & c,
    unsigned level ) const [protected]
```

References `GiNaC::_ex1_2`, `basis`, `c`, `exponent`, `GiNaC::ex::is_equal()`, `GiNaC::ex::print()`, and `print_power()`.

## 6.119.3.31 do\_print\_latex()

```
void GiNaC::power::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

References `GiNaC::_ex1_2`, `basis`, `c`, `exponent`, `GiNaC::ex::is_equal()`, `GiNaC::is_negative()`, `power()`, `GiNaC::ex::print()`, and `print_power()`.

## 6.119.3.32 do\_print\_csrc()

```
void GiNaC::power::do_print_csrc (
    const print_csrc & c,
    unsigned level ) const [protected]
```

References `GiNaC::_ex_1`, `basis`, `c`, `GiNaC::exp()`, `exponent`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `GiNaC::ex::is_equal()`, `GiNaC::ex::print()`, `GiNaC::print_sym_pow()`, and `GiNaC::to_int()`.

**6.119.3.33 do\_print\_python()**

```
void GiNaC::power::do_print_python (
    const print\_python & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_power\(\)](#).

**6.119.3.34 do\_print\_python\_repr()**

```
void GiNaC::power::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [basis](#), [c](#), [exponent](#), and [GiNaC::ex::print\(\)](#).

**6.119.3.35 do\_print\_csrc\_cl\_N()**

```
void GiNaC::power::do_print_csrc_cl_N (
    const print\_csrc\_cl\_N & c,
    unsigned level ) const [protected]
```

References [GiNaC::\\_ex\\_1](#), [basis](#), [c](#), [exponent](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::print\(\)](#).

**6.119.3.36 expand\_add()**

```
ex GiNaC::power::expand_add (
    const add & a,
    long n,
    unsigned options ) [static], [protected]
```

expand  $a^n$  where  $a$  is an [add](#) and  $n$  is a positive integer.

See also

[power::expand](#)

References [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [basis](#), [GiNaC::binomial\(\)](#), [c](#), [coeff\(\)](#), [expand\(\)](#), [expand\\_add\\_2\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GiNaC::factor\(\)](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::get\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_pos\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [k](#), [mul](#), [GiNaC::multinomial\\_coefficient\(\)](#), [n](#), [GiNaC::composition\\_generator::next\(\)](#), [GiNaC::expairseq::nops\(\)](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::pow\(\)](#), [power\(\)](#), [r](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), and [GiNaC::numeric::to\\_long\(\)](#).

Referenced by [expand\(\)](#).

6.119.3.37 `expand_add_2()`

```
ex GiNaC::power::expand_add_2 (
    const add & a,
    unsigned options ) [static], [protected]
```

Special case of [power::expand\\_add](#).

Expands  $a^2$  where  $a$  is an `add`.

See also

[power::expand\\_add](#)

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex2](#), [GiNaC::\\_num2\\_p](#), [basis](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::add::combine\\_↵  
pair\\_with\\_coeff\\_to\\_pair\(\)](#), [expand\(\)](#), [expand\\_mul\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GINAC\\_ASSERT](#),  
[GiNaC::is\\_pos\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [last](#), [mul](#), [GiNaC::numeric::mul\(\)](#), [GiNaC::numeric::mul\\_dyn\(\)](#), [Gi↵  
NaC::expairseq::nops\(\)](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [power\(\)](#), [r](#), [GiNaC::expairseq::seq](#), and [GiNaC↵  
::basic::setflag\(\)](#).

Referenced by [expand\\_add\(\)](#).

6.119.3.38 `expand_mul()`

```
ex GiNaC::power::expand_mul (
    const mul & m,
    const numeric & n,
    unsigned options,
    bool from_expand = false ) [static], [protected]
```

Expand factors of  $m$  in  $m^n$  where  $m$  is a `mul` and  $n$  is an integer.

See also

[power::expand](#)

References [GiNaC::\\_ex1](#), [GiNaC::expair::coeff](#), [GiNaC::basic::ex](#), [GiNaC::expand\\_options::expand\\_rename\\_idx](#),  
[GiNaC::status\\_flags::expanded](#), [GiNaC::get\\_all\\_dummy\\_indices\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::info\\_flags::has\\_↵  
indices](#), [m](#), [n](#), [options](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [eval\(\)](#), [expand\(\)](#), and [expand\\_add\\_2\(\)](#).

## 6.119.4 Friends And Related Function Documentation

6.119.4.1 `mul`

```
friend class mul [friend]
```

Referenced by [expand\\_add\(\)](#), and [expand\\_add\\_2\(\)](#).

### 6.119.5 Member Data Documentation

#### 6.119.5.1 basis

`ex GiNaC::power::basis` [protected]

Referenced by `archive()`, `coeff()`, `conjugate()`, `degree()`, `derivative()`, `do_print_csrc()`, `do_print_csrc_cl_N()`, `do_print_dflt()`, `do_print_latex()`, `do_print_python_repr()`, `eval()`, `evalf()`, `evalm()`, `expand()`, `expand_add()`, `expand_add_2()`, `has()`, `imag_part()`, `info()`, `is_polynomial()`, `ldegree()`, `map()`, `normal()`, `op()`, `print_power()`, `read_archive()`, `real_part()`, `return_type()`, `return_type_tinfo()`, `series()`, `GiNaC::mul::split_ex_to_pair()`, `subs()`, `to_polynomial()`, and `to_rational()`.

#### 6.119.5.2 exponent

`ex GiNaC::power::exponent` [protected]

Referenced by `archive()`, `coeff()`, `conjugate()`, `degree()`, `derivative()`, `do_print_csrc()`, `do_print_csrc_cl_N()`, `do_print_dflt()`, `do_print_latex()`, `do_print_python_repr()`, `eval()`, `evalf()`, `evalm()`, `expand()`, `expand_add()`, `expand_add_2()`, `has()`, `imag_part()`, `info()`, `is_polynomial()`, `ldegree()`, `map()`, `normal()`, `op()`, `print_power()`, `read_archive()`, `real_part()`, `series()`, `GiNaC::mul::split_ex_to_pair()`, `subs()`, `to_polynomial()`, and `to_rational()`.

The documentation for this class was generated from the following files:

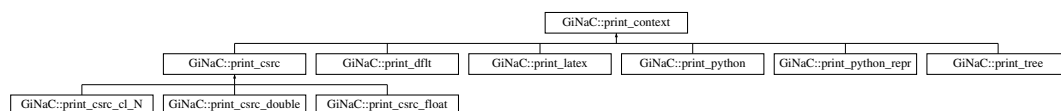
- [power.h](#)
- [normal.cpp](#)
- [power.cpp](#)
- [pseries.cpp](#)

## 6.120 GiNaC::print\_context Class Reference

Base class for `print_contexts`.

```
#include <print.h>
```

Inheritance diagram for `GiNaC::print_context`:



### Public Member Functions

- [print\\_context](#) (std::ostream &, unsigned `options`=0)
- [~print\\_context](#) ()



## Public Attributes

- `std::ostream & s`  
*stream to output to*
- `unsigned options`  
*option flags*

### 6.120.1 Detailed Description

Base class for print\_contexts.

### 6.120.2 Constructor & Destructor Documentation

#### 6.120.2.1 print\_context()

```
GiNaC::print_context::print_context (
    std::ostream & os,
    unsigned options = 0 )
```

#### 6.120.2.2 ~print\_context()

```
virtual GiNaC::print_context::~~print_context ( ) [inline], [virtual]
```

### 6.120.3 Member Data Documentation

#### 6.120.3.1 s

```
std::ostream& GiNaC::print_context::s
```

stream to output to

Referenced by `GiNaC::numeric::print_numeric()`.

### 6.120.3.2 options

```
unsigned GiNaC::print_context::options
```

option flags

Referenced by `GiNaC::get_print_options()`, `GiNaC::set_print_context()`, and `GiNaC::set_print_options()`.

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)

## 6.121 GiNaC::print\_context\_options Class Reference

This class stores information about a registered [print\\_context](#) class.

```
#include <print.h>
```

### Public Member Functions

- [print\\_context\\_options](#) (const char \*n, const char \*p, unsigned i)
- const char \* [get\\_name](#) () const
- const char \* [get\\_parent\\_name](#) () const
- unsigned [get\\_id](#) () const

### Private Attributes

- const char \* [name](#)  
*Class name.*
- const char \* [parent\\_name](#)  
*Name of superclass.*
- unsigned [id](#)  
*ID number (assigned automatically).*

### 6.121.1 Detailed Description

This class stores information about a registered [print\\_context](#) class.

### 6.121.2 Constructor & Destructor Documentation

### 6.121.2.1 print\_context\_options()

```
GiNaC::print_context_options::print_context_options (
    const char * n,
    const char * p,
    unsigned i ) [inline]
```

## 6.121.3 Member Function Documentation

### 6.121.3.1 get\_name()

```
const char* GiNaC::print_context_options::get_name ( ) const [inline]
```

References name.

### 6.121.3.2 get\_parent\_name()

```
const char* GiNaC::print_context_options::get_parent_name ( ) const [inline]
```

References parent\_name.

### 6.121.3.3 get\_id()

```
unsigned GiNaC::print_context_options::get_id ( ) const [inline]
```

References id.

## 6.121.4 Member Data Documentation

### 6.121.4.1 name

```
const char* GiNaC::print_context_options::name [private]
```

Class name.

Referenced by get\_name().

#### 6.121.4.2 parent\_name

```
const char* GiNaC::print_context_options::parent_name [private]
```

Name of superclass.

Referenced by `get_parent_name()`.

#### 6.121.4.3 id

```
unsigned GiNaC::print_context_options::id [private]
```

ID number (assigned automatically).

Referenced by `get_id()`.

The documentation for this class was generated from the following file:

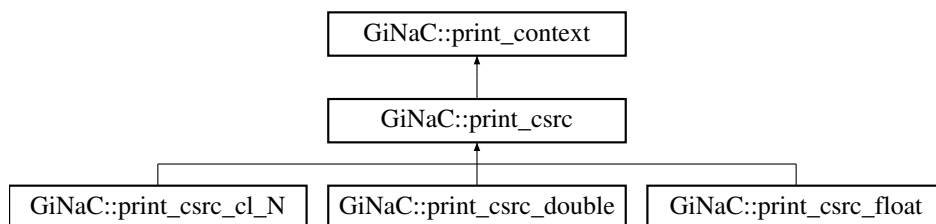
- [print.h](#)

## 6.122 GiNaC::print\_csrc Class Reference

Base context for C source output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc:



### Public Member Functions

- [print\\_csrc](#) (std::ostream &, unsigned [options](#)=0)

### Additional Inherited Members

#### 6.122.1 Detailed Description

Base context for C source output.

## 6.122.2 Constructor & Destructor Documentation

### 6.122.2.1 print\_csrc()

```
GiNaC::print_csrc::print_csrc (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

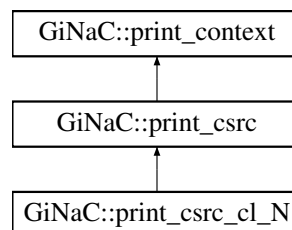
- [print.h](#)
- [print.cpp](#)

## 6.123 GiNaC::print\_csrc\_cl\_N Class Reference

Context for C source output using CLN numbers.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc\_cl\_N:



### Public Member Functions

- [print\\_csrc\\_cl\\_N](#) (std::ostream &, unsigned options=0)

### Additional Inherited Members

### 6.123.1 Detailed Description

Context for C source output using CLN numbers.

### 6.123.2 Constructor & Destructor Documentation

### 6.123.2.1 `print_csrc_cl_N()`

```
GiNaC::print_csrc_cl_N::print_csrc_cl_N (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

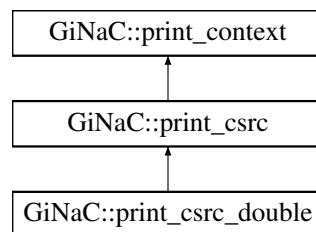
- [print.h](#)
- [print.cpp](#)

## 6.124 `GiNaC::print_csrc_double` Class Reference

Context for C source output using double precision.

```
#include <print.h>
```

Inheritance diagram for `GiNaC::print_csrc_double`:



### Public Member Functions

- [print\\_csrc\\_double](#) (std::ostream &, unsigned options=0)

### Additional Inherited Members

### 6.124.1 Detailed Description

Context for C source output using double precision.

### 6.124.2 Constructor & Destructor Documentation

## 6.124.2.1 print\_csrc\_double()

```
GiNaC::print_csrc_double::print_csrc_double (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

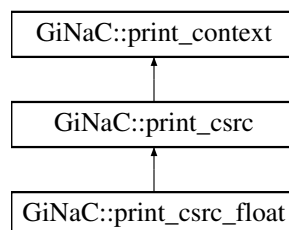
- [print.h](#)
- [print.cpp](#)

## 6.125 GiNaC::print\_csrc\_float Class Reference

Context for C source output using float precision.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc\_float:



## Public Member Functions

- [print\\_csrc\\_float](#) (std::ostream &, unsigned options=0)

## Additional Inherited Members

## 6.125.1 Detailed Description

Context for C source output using float precision.

## 6.125.2 Constructor &amp; Destructor Documentation

### 6.125.2.1 `print_csrc_float()`

```
GiNaC::print_csrc_float::print_csrc_float (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

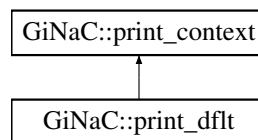
- [print.h](#)
- [print.cpp](#)

## 6.126 `GiNaC::print_dflt` Class Reference

Context for default (ginsh-parsable) output.

```
#include <print.h>
```

Inheritance diagram for `GiNaC::print_dflt`:



### Public Member Functions

- [print\\_dflt](#) (std::ostream &, unsigned options=0)

### Additional Inherited Members

### 6.126.1 Detailed Description

Context for default (ginsh-parsable) output.

### 6.126.2 Constructor & Destructor Documentation

#### 6.126.2.1 `print_dflt()`

```
GiNaC::print_dflt::print_dflt (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)



## 6.127 GiNaC::print\_functor Class Reference

This class represents a print method for a certain algebraic class and [print\\_context](#) type.

```
#include <print.h>
```

### Public Member Functions

- [print\\_functor](#) ()
- [print\\_functor](#) (const [print\\_functor](#) &other)
- [print\\_functor](#) (std::unique\_ptr< [print\\_functor\\_impl](#) > impl\_)
- template<class T, class C >  
  [print\\_functor](#) (void f(const T &, const C &, unsigned))
- template<class T, class C >  
  [print\\_functor](#) (void(T::\*f)(const C &, unsigned) const)
- [print\\_functor](#) & [operator=](#) (const [print\\_functor](#) &other)
- void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const
- bool [is\\_valid](#) () const

### Private Attributes

- std::unique\_ptr< [print\\_functor\\_impl](#) > impl

### 6.127.1 Detailed Description

This class represents a print method for a certain algebraic class and [print\\_context](#) type.

Its main purpose is to hide the difference between member functions and nonmember functions behind one unified [operator\(\)](#) interface. [print\\_functor](#) has value semantics and acts as a smart pointer (with deep copy) to a class derived from [print\\_functor\\_impl](#) which implements the actual function call.

### 6.127.2 Constructor & Destructor Documentation

#### 6.127.2.1 [print\\_functor](#)() [1/5]

```
GiNaC::print_functor::print_functor ( ) [inline]
```

#### 6.127.2.2 [print\\_functor](#)() [2/5]

```
GiNaC::print_functor::print_functor (
    const print\_functor & other ) [inline]
```

**6.127.2.3 print\_functor()** [3/5]

```
GiNaC::print_functor::print_functor (
    std::unique_ptr< print\_functor\_impl > impl_ ) [inline]
```

**6.127.2.4 print\_functor()** [4/5]

```
template<class T , class C >
GiNaC::print_functor::print_functor (
    void fconst T &, const C &, unsigned ) [inline]
```

**6.127.2.5 print\_functor()** [5/5]

```
template<class T , class C >
GiNaC::print_functor::print_functor (
    void(T::*)(const C &, unsigned) const f ) [inline]
```

**6.127.3 Member Function Documentation****6.127.3.1 operator=()**

```
print\_functor& GiNaC::print_functor::operator= (
    const print\_functor & other ) [inline]
```

References impl.

**6.127.3.2 operator()()**

```
void GiNaC::print_functor::operator() (
    const basic & obj,
    const print\_context & c,
    unsigned level ) const [inline]
```

References c.

## 6.127.3.3 is\_valid()

```
bool GiNaC::print_functor::is_valid ( ) const [inline]
```

References impl.

## 6.127.4 Member Data Documentation

## 6.127.4.1 impl

```
std::unique_ptr<print_functor_impl> GiNaC::print_functor::impl [private]
```

Referenced by is\_valid(), and operator=().

The documentation for this class was generated from the following file:

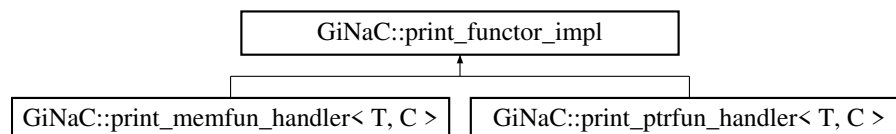
- [print.h](#)

## 6.128 GiNaC::print\_functor\_impl Class Reference

Base class for [print\\_functor](#) handlers.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_functor\_impl:



## Public Member Functions

- virtual [~print\\_functor\\_impl](#) ()
- virtual [print\\_functor\\_impl](#) \* [duplicate](#) () const =0
- virtual void [operator](#)() (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const =0

## 6.128.1 Detailed Description

Base class for [print\\_functor](#) handlers.

## 6.128.2 Constructor & Destructor Documentation

### 6.128.2.1 `~print_functor_impl()`

```
virtual GiNaC::print_functor_impl::~~print_functor_impl ( ) [inline], [virtual]
```

## 6.128.3 Member Function Documentation

### 6.128.3.1 `duplicate()`

```
virtual print\_functor\_impl* GiNaC::print_functor_impl::duplicate ( ) const [pure virtual]
```

Implemented in [GiNaC::print\\_memfun\\_handler< T, C >](#), and [GiNaC::print\\_ptrfun\\_handler< T, C >](#).

### 6.128.3.2 `operator()`

```
virtual void GiNaC::print_functor_impl::operator() (
    const basic & obj,
    const print\_context & c,
    unsigned level ) const [pure virtual]
```

Implemented in [GiNaC::print\\_memfun\\_handler< T, C >](#), and [GiNaC::print\\_ptrfun\\_handler< T, C >](#).

The documentation for this class was generated from the following file:

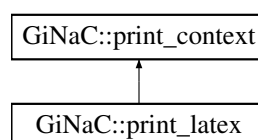
- [print.h](#)

## 6.129 GiNaC::print\_latex Class Reference

Context for latex-parsable output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_latex:



## Public Member Functions

- [print\\_latex](#) (std::ostream &, unsigned [options](#)=0)

## Additional Inherited Members

### 6.129.1 Detailed Description

Context for latex-parsable output.

### 6.129.2 Constructor & Destructor Documentation

#### 6.129.2.1 print\_latex()

```
GiNaC::print_latex::print_latex (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

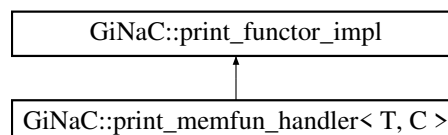
- [print.h](#)
- [print.cpp](#)

## 6.130 GiNaC::print\_memfun\_handler< T, C > Class Template Reference

[print\\_functor](#) handler for member functions of class T, context type C

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_memfun\_handler< T, C >:



## Public Types

- typedef void(T::\* [F](#)) (const C &[c](#), unsigned level) const

## Public Member Functions

- [print\\_memfun\\_handler](#) ([F f\\_](#))
- [print\\_memfun\\_handler](#) \* [duplicate](#) () const override
- void [operator](#)() (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const override

## Private Attributes

- [F f](#)

### 6.130.1 Detailed Description

```
template<class T, class C>
class GiNaC::print_memfun_handler< T, C >
```

[print\\_functor](#) handler for member functions of class T, context type C

### 6.130.2 Member Typedef Documentation

#### 6.130.2.1 F

```
template<class T , class C >
typedef void(T::* GiNaC::print\_memfun\_handler< T, C >::F) (const C &c, unsigned level) const
```

### 6.130.3 Constructor & Destructor Documentation

#### 6.130.3.1 [print\\_memfun\\_handler](#)()

```
template<class T , class C >
GiNaC::print\_memfun\_handler< T, C >::print_memfun_handler (
    F f\_ ) [inline]
```

Referenced by [GiNaC::print\\_memfun\\_handler](#)< T, C >::duplicate().

### 6.130.4 Member Function Documentation

## 6.130.4.1 duplicate()

```
template<class T , class C >
print_memfun_handler* GiNaC::print_memfun_handler< T, C >::duplicate ( ) const [inline],
[override], [virtual]
```

Implements [GiNaC::print\\_funcutor\\_impl](#).

References [GiNaC::print\\_memfun\\_handler< T, C >::print\\_memfun\\_handler\(\)](#).

## 6.130.4.2 operator()

```
template<class T , class C >
void GiNaC::print_memfun_handler< T, C >::operator() (
    const basic & obj,
    const print_context & c,
    unsigned level ) const [inline], [override], [virtual]
```

Implements [GiNaC::print\\_funcutor\\_impl](#).

References [c](#), and [GiNaC::print\\_memfun\\_handler< T, C >::f](#).

## 6.130.5 Member Data Documentation

## 6.130.5.1 f

```
template<class T , class C >
F GiNaC::print_memfun_handler< T, C >::f [private]
```

Referenced by [GiNaC::print\\_memfun\\_handler< T, C >::operator\(\)](#).

The documentation for this class was generated from the following file:

- [print.h](#)

## 6.131 GiNaC::print\_options Class Reference

Flags to control the behavior of a [print\\_context](#).

```
#include <print.h>
```

## Public Types

- enum { [print\\_index\\_dimensions](#) = 0x0001 }

### 6.131.1 Detailed Description

Flags to control the behavior of a [print\\_context](#).

### 6.131.2 Member Enumeration Documentation

#### 6.131.2.1 anonymous enum

anonymous enum

##### Enumerator

print_index_dimensions	print the dimensions of indices
------------------------	---------------------------------

The documentation for this class was generated from the following file:

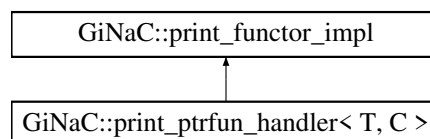
- [print.h](#)

## 6.132 GiNaC::print\_ptrfun\_handler< T, C > Class Template Reference

[print\\_functor](#) handler for pointer-to-functions of class T, context type C

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_ptrfun\_handler< T, C >:



### Public Types

- typedef void(\* [F](#)) (const T &, const C &, unsigned)

### Public Member Functions

- [print\\_ptrfun\\_handler](#) (F f\_)
- [print\\_ptrfun\\_handler](#) \* [duplicate](#) () const override
- void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const override



## Private Attributes

- [F f](#)

### 6.132.1 Detailed Description

```
template<class T, class C>
class GiNaC::print_ptrfun_handler< T, C >
```

[print\\_functor](#) handler for pointer-to-functions of class T, context type C

### 6.132.2 Member Typedef Documentation

#### 6.132.2.1 F

```
template<class T , class C >
typedef void(* GiNaC::print\_ptrfun\_handler< T, C >::F) (const T &, const C &, unsigned)
```

### 6.132.3 Constructor & Destructor Documentation

#### 6.132.3.1 print\_ptrfun\_handler()

```
template<class T , class C >
GiNaC::print\_ptrfun\_handler< T, C >::print_ptrfun_handler (
    F f_ ) [inline]
```

Referenced by [GiNaC::print\\_ptrfun\\_handler](#)< T, C >::duplicate().

### 6.132.4 Member Function Documentation

#### 6.132.4.1 duplicate()

```
template<class T , class C >
print\_ptrfun\_handler* GiNaC::print\_ptrfun\_handler< T, C >::duplicate ( ) const [inline],
[override], [virtual]
```

Implements [GiNaC::print\\_functor\\_impl](#).

References [GiNaC::print\\_ptrfun\\_handler](#)< T, C >::print\_ptrfun\_handler().

#### 6.132.4.2 operator()

```
template<class T , class C >
void GiNaC::print_ptrfun_handler< T, C >::operator() (
    const basic & obj,
    const print_context & c,
    unsigned level ) const [inline], [override], [virtual]
```

Implements [GiNaC::print\\_func\\_impl](#).

References [c](#), and [GiNaC::print\\_ptrfun\\_handler< T, C >::f](#).

#### 6.132.5 Member Data Documentation

##### 6.132.5.1 f

```
template<class T , class C >
F GiNaC::print_ptrfun_handler< T, C >::f [private]
```

Referenced by [GiNaC::print\\_ptrfun\\_handler< T, C >::operator\(\)](#).

The documentation for this class was generated from the following file:

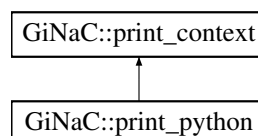
- [print.h](#)

### 6.133 GiNaC::print\_python Class Reference

Context for python pretty-print output.

```
#include <print.h>
```

Inheritance diagram for [GiNaC::print\\_python](#):



#### Public Member Functions

- [print\\_python](#) (std::ostream &, unsigned [options](#)=0)

## Additional Inherited Members

### 6.133.1 Detailed Description

Context for python pretty-print output.

### 6.133.2 Constructor & Destructor Documentation

#### 6.133.2.1 print\_python()

```

GiNaC::print_python::print_python (
    std::ostream & os,
    unsigned options = 0 )

```

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)

## 6.134 GiNaC::print\_python\_repr Class Reference

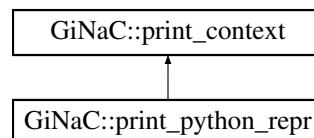
Context for python-parsable output.

```

#include <print.h>

```

Inheritance diagram for GiNaC::print\_python\_repr:



## Public Member Functions

- [print\\_python\\_repr](#) (std::ostream &, unsigned options=0)

## Additional Inherited Members

### 6.134.1 Detailed Description

Context for python-parsable output.

## 6.134.2 Constructor & Destructor Documentation

### 6.134.2.1 `print_python_repr()`

```
GiNaC::print_python_repr::print_python_repr (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

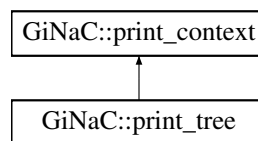
- [print.h](#)
- [print.cpp](#)

## 6.135 GiNaC::print\_tree Class Reference

Context for tree-like output for debugging.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_tree:



### Public Member Functions

- [print\\_tree](#) (unsigned d)
- [print\\_tree](#) (std::ostream &, unsigned options=0, unsigned d=4)

### Public Attributes

- const unsigned [delta\\_indent](#)  
*size of indentation step*

### 6.135.1 Detailed Description

Context for tree-like output for debugging.

### 6.135.2 Constructor & Destructor Documentation

#### 6.135.2.1 `print_tree()` [1/2]

```
GiNaC::print_tree::print_tree (
    unsigned d )
```

#### 6.135.2.2 `print_tree()` [2/2]

```
GiNaC::print_tree::print_tree (
    std::ostream & os,
    unsigned options = 0,
    unsigned d = 4 )
```

### 6.135.3 Member Data Documentation

#### 6.135.3.1 `delta_indent`

```
const unsigned GiNaC::print_tree::delta_indent
```

size of indentation step

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)

## 6.136 GiNaC::archive\_node::property Struct Reference

Archived property (data type, name and associated data)

```
#include <archive.h>
```

### Public Member Functions

- [property](#) ()
- [property](#) ([archive\\_atom](#) n, [property\\_type](#) t, unsigned v)

### Public Attributes

- [property\\_type](#) type  
*Data type of property.*
- [archive\\_atom](#) name  
*Name of property.*
- unsigned [value](#)  
*Stored value.*

### 6.136.1 Detailed Description

Archived property (data type, name and associated data)

### 6.136.2 Constructor & Destructor Documentation

#### 6.136.2.1 `property()` [1/2]

```
GiNaC::archive_node::property::property ( ) [inline]
```

#### 6.136.2.2 `property()` [2/2]

```
GiNaC::archive_node::property::property (
    archive_atom n,
    property_type t,
    unsigned v ) [inline]
```

### 6.136.3 Member Data Documentation

#### 6.136.3.1 `type`

```
property_type GiNaC::archive_node::property::type
```

Data type of property.

#### 6.136.3.2 `name`

```
archive_atom GiNaC::archive_node::property::name
```

Name of property.

### 6.136.3.3 value

unsigned GiNaC::archive\_node::property::value

Stored value.

The documentation for this struct was generated from the following file:

- [archive.h](#)

## 6.137 GiNaC::archive\_node::property\_info Struct Reference

Information about a stored property.

```
#include <archive.h>
```

### Public Member Functions

- [property\\_info](#) ()
- [property\\_info](#) ([property\\_type](#) t, const std::string &n, unsigned c=1)

### Public Attributes

- [property\\_type](#) type  
*Data type of property.*
- std::string [name](#)  
*Name of property.*
- unsigned [count](#)  
*Number of occurrences.*

### 6.137.1 Detailed Description

Information about a stored property.

A vector of these structures is returned by [get\\_properties\(\)](#).

See also

[get\\_properties](#)

### 6.137.2 Constructor & Destructor Documentation

#### 6.137.2.1 `property_info()` [1/2]

```
GiNaC::archive_node::property_info::property_info ( ) [inline]
```

#### 6.137.2.2 `property_info()` [2/2]

```
GiNaC::archive_node::property_info::property_info (
    property\_type t,
    const std::string & n,
    unsigned c = 1 ) [inline]
```

### 6.137.3 Member Data Documentation

#### 6.137.3.1 `type`

[property\\_type](#) GiNaC::archive\_node::property\_info::type

Data type of property.

#### 6.137.3.2 `name`

std::string GiNaC::archive\_node::property\_info::name

Name of property.

#### 6.137.3.3 `count`

unsigned GiNaC::archive\_node::property\_info::count

Number of occurrences.

The documentation for this struct was generated from the following file:

- [archive.h](#)

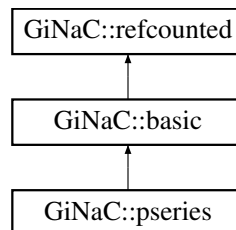


## 6.138 GiNaC::pseries Class Reference

This class holds a extended truncated power series (positive and negative integer powers).

```
#include <pseries.h>
```

Inheritance diagram for GiNaC::pseries:



### Public Member Functions

- **pseries** (const **ex** &rel\_, const **epvector** &ops\_)  
*Construct pseries from a vector of coefficients and powers.*
- **pseries** (const **ex** &rel\_, **epvector** &&ops\_)
- unsigned **precedence** () const override  
*Return relative operator precedence (for parenthezing output).*
- size\_t **nops** () const override  
*Return the number of operands including a possible order term.*
- **ex op** (size\_t i) const override  
*Return the ith term in the series when represented as a sum.*
- int **degree** (const **ex** &s) const override  
*Return degree of highest power of the series.*
- int **ldegree** (const **ex** &s) const override  
*Return degree of lowest power of the series.*
- **ex coeff** (const **ex** &s, int n=1) const override  
*Return coefficient of degree n in power series if s is the expansion variable.*
- **ex collect** (const **ex** &s, bool distributed=false) const override  
*Does nothing.*
- **ex eval** () const override  
*Perform coefficient-wise automatic term rewriting rules in this class.*
- **ex evalf** () const override  
*Evaluate coefficients numerically.*
- **ex series** (const **relational** &r, int order, unsigned options=0) const override  
*Re-expansion of a pseries object.*
- **ex subs** (const **exmap** &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- **ex normal** (**exmap** &repl, **exmap** &rev\_lookup, **lst** &modifier) const override  
*Implementation of **ex::normal()** for pseries.*
- **ex expand** (unsigned options=0) const override  
*Implementation of **ex::expand()** for a power series.*
- **ex conjugate** () const override
- **ex real\_part** () const override
- **ex imag\_part** () const override

- [ex eval\\_integ](#) () const override  
*Evaluate integrals, if result is known.*
- [ex evalm](#) () const override  
*Evaluate sums, products and integer powers of matrices.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &[syms](#)) override  
*Read (a.k.a.*
- [ex get\\_var](#) () const  
*Get the expansion variable.*
- [ex get\\_point](#) () const  
*Get the expansion point.*
- [ex convert\\_to\\_poly](#) (bool no\_order=false) const  
*Convert the pseries object to an ordinary polynomial.*
- bool [is\\_compatible\\_to](#) (const [pseries](#) &other) const  
*Check whether series is compatible to another series (expansion variable and point are the same.*
- bool [is\\_zero](#) () const  
*Check whether series has the value zero.*
- bool [is\\_terminating](#) () const  
*Returns true if there is no order term, i.e.*
- [ex coeffop](#) (size\_t i) const  
*Get coefficients and exponents.*
- [ex exponop](#) (size\_t i) const
- [ex add\\_series](#) (const [pseries](#) &other) const  
*Add one series object to another, producing a pseries object that represents the sum.*
- [ex mul\\_const](#) (const [numeric](#) &other) const  
*Multiply a pseries object with a numeric constant, producing a pseries object that represents the product.*
- [ex mul\\_series](#) (const [pseries](#) &other) const  
*Multiply one pseries object to another, producing a pseries object that represents the product.*
- [ex power\\_const](#) (const [numeric](#) &p, int deg) const  
*Compute the p-th power of a series.*
- [pseries shift\\_exponents](#) (int deg) const  
*Return a new pseries object with the powers shifted by deg.*

## Protected Member Functions

- [ex derivative](#) (const [symbol](#) &s) const override  
*Implementation of `ex::diff()` for a power series.*
- void [print\\_series](#) (const [print\\_context](#) &c, const char \*openbrace, const char \*closebrace, const char \*mul←  
\_sym, const char \*pow\_sym, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

## Protected Attributes

- [epvector seq](#)  
Vector of {coefficient, power} pairs.
- [ex var](#)  
Series variable (holds a symbol)
- [ex point](#)  
Expansion point.

### 6.138.1 Detailed Description

This class holds a extended truncated power series (positive and negative integer powers).

It consists of expression coefficients (only non-zero coefficients are stored), an expansion variable and an expansion point. Other classes must provide members to convert into this type.

### 6.138.2 Constructor & Destructor Documentation

#### 6.138.2.1 pseries() [1/2]

```
GiNaC::pseries::pseries (
    const ex & rel_,
    const epvector & ops_ )
```

Construct pseries from a vector of coefficients and powers.

[expair.rest](#) holds the coefficient, [expair.coeff](#) holds the power. The powers must be integers (positive or negative) and in ascending order; the last coefficient can be Order(\_ex1) to represent a truncated, non-terminating series.

#### Parameters

<a href="#">rel</a> ↔ —	expansion variable and point (must hold a relational)
<a href="#">ops</a> ↔ —	vector of {coefficient, power} pairs (coefficient must not be zero)

#### Returns

newly constructed pseries

References GINAC\_ASSERT, GiNaC::is\_order\_function(), GiNaC::ex::lhs(), point, GiNaC::ex::rhs(), seq, and var.

Referenced by add\_series(), derivative(), mul\_const(), mul\_series(), normal(), power\_const(), series(), and shift\_↔exponents().

**6.138.2.2 pseries()** [2/2]

```
GiNaC::pseries::pseries (
    const ex & rel_,
    epvector && ops_ )
```

References GINAC\_ASSERT, GiNaC::is\_order\_function(), GiNaC::ex::lhs(), point, GiNaC::ex::rhs(), seq, and var.

**6.138.3 Member Function Documentation****6.138.3.1 precedence()**

```
unsigned GiNaC::pseries::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by print\_series().

**6.138.3.2 nops()**

```
size_t GiNaC::pseries::nops ( ) const [override], [virtual]
```

Return the number of operands including a possible order term.

Reimplemented from [GiNaC::basic](#).

References seq.

Referenced by coeffop(), exponop(), and GiNaC::log\_series().

**6.138.3.3 op()**

```
ex GiNaC::pseries::op (
    size_t i ) const [override], [virtual]
```

Return the ith term in the series when represented as a sum.

Reimplemented from [GiNaC::basic](#).

References coeff(), GiNaC::is\_order\_function(), point, GiNaC::pow(), seq, and var.

## 6.138.3.4 degree()

```
int GiNaC::pseries::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power of the series.

This is usually the exponent of the Order term. If s is not the expansion variable of the series, the series is examined termwise.

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [seq](#), and [var](#).

Referenced by [mul\\_series\(\)](#), and [series\(\)](#).

## 6.138.3.5 ldegree()

```
int GiNaC::pseries::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power of the series.

This is usually the exponent of the leading term. If s is not the expansion variable of the series, the series is examined termwise. If s is the expansion variable but the expansion point is not zero the series is not expanded to find the degree. I.e.:  $(1-x) + (1-x)^2 + \text{Order}((1-x)^3)$  has `ldegree(x)` 1, not 0.

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [seq](#), and [var](#).

Referenced by [GiNaC::log\\_series\(\)](#), [mul\\_series\(\)](#), and [power\\_const\(\)](#).

## 6.138.3.6 coeff()

```
ex GiNaC::pseries::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in power series if s is the expansion variable.

If the expansion point is nonzero, by definition the n=1 coefficient in s of  $a+b*(s-z)+c*(s-z)^2+\text{Order}((s-z)^3)$  is b (assuming the expansion took place in the s in the first place). If s is not the expansion variable, an attempt is made to convert the series to a polynomial and return the corresponding coefficient from there.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::basic::compare\(\)](#), [convert\\_to\\_poly\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [n](#), [seq](#), and [var](#).

Referenced by [degree\(\)](#), [ldegree\(\)](#), [GiNaC::log\\_series\(\)](#), [mul\\_series\(\)](#), [op\(\)](#), [power\\_const\(\)](#), and [read\\_archive\(\)](#).

**6.138.3.7 collect()**

```
ex GiNaC::pseries::collect (
    const ex & s,
    bool distributed = false ) const [override], [virtual]
```

Does nothing.

Reimplemented from [GiNaC::basic](#).

**6.138.3.8 eval()**

```
ex GiNaC::pseries::eval ( ) const [override], [virtual]
```

Perform coefficient-wise automatic term rewriting rules in this class.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

**6.138.3.9 evalf()**

```
ex GiNaC::pseries::evalf ( ) const [override], [virtual]
```

Evaluate coefficients numerically.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [point](#), [seq](#), [GiNaC::basic::setflag\(\)](#), and [var](#).

**6.138.3.10 series()**

```
ex GiNaC::pseries::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Re-expansion of a pseries object.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [convert\\_to\\_poly\(\)](#), [degree\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [options](#), [order](#), [point](#), [pseries\(\)](#), [r](#), [seq](#), [GiNaC::ex::series\(\)](#), [GiNaC::to\\_int\(\)](#), and [var](#).

## 6.138.3.11 subs()

```
ex GiNaC::pseries::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [convert\\_to\\_poly\(\)](#), [m](#), [options](#), [point](#), [seq](#), [GiNaC::ex::subs\(\)](#), and [var](#).

## 6.138.3.12 normal()

```
ex GiNaC::pseries::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for pseries.

It normalizes each coefficient and replaces the series by a temporary symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [n](#), [GiNaC::ex::normal\(\)](#), [point](#), [pseries\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [seq](#), and [var](#).

## 6.138.3.13 expand()

```
ex GiNaC::pseries::expand (
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of [ex::expand\(\)](#) for a power series.

It expands all the terms individually and returns the resulting series as a new pseries.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::ex::is\\_zero\(\)](#), [options](#), [point](#), [seq](#), [GiNaC::basic::setflag\(\)](#), and [var](#).

**6.138.3.14 conjugate()**

```
ex GiNaC::pseries::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::conjugateepvector\(\)](#), [GiNaC::ex::info\(\)](#), [point](#), [GiNaC::info\\_flags::real](#), [seq](#), and [var](#).

**6.138.3.15 real\_part()**

```
ex GiNaC::pseries::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::info\(\)](#), [point](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), [seq](#), and [var](#).

**6.138.3.16 imag\_part()**

```
ex GiNaC::pseries::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::info\(\)](#), [point](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), [seq](#), and [var](#).

**6.138.3.17 eval\_integ()**

```
ex GiNaC::pseries::eval_integ ( ) const [override], [virtual]
```

Evaluate integrals, if result is known.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::eval\\_integ\(\)](#), [point](#), [seq](#), and [var](#).

**6.138.3.18 evalm()**

```
ex GiNaC::pseries::evalm ( ) const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::evalm\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [point](#), [seq](#), and [var](#).



### 6.138.3.19 archive()

```
void GiNaC::pseries::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), [point](#), [seq](#), and [var](#).

### 6.138.3.20 read\_archive()

```
void GiNaC::pseries::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [n](#), [point](#), [seq](#), and [var](#).

### 6.138.3.21 derivative()

```
ex GiNaC::pseries::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a power series.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [c](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [pseries\(\)](#), [seq](#), and [var](#).

**6.138.3.22 get\_var()**

```
ex GiNaC::pseries::get_var ( ) const [inline]
```

Get the expansion variable.

References var.

**6.138.3.23 get\_point()**

```
ex GiNaC::pseries::get_point ( ) const [inline]
```

Get the expansion point.

References point.

**6.138.3.24 convert\_to\_poly()**

```
ex GiNaC::pseries::convert_to_poly (
    bool no_order = false ) const
```

Convert the pseries object to an ordinary polynomial.

**Parameters**

<i>no_order</i>	flag: discard higher order terms
-----------------	----------------------------------

References GiNaC::ex::coeff(), GiNaC::is\_order\_function(), point, GiNaC::pow(), seq, and var.

Referenced by coeff(), series(), and subs().

**6.138.3.25 is\_compatible\_to()**

```
bool GiNaC::pseries::is_compatible_to (
    const pseries & other ) const [inline]
```

Check whether series is compatible to another series (expansion variable and point are the same).

References GiNaC::ex::is\_equal(), point, and var.

Referenced by add\_series(), and mul\_series().

#### 6.138.3.26 is\_zero()

```
bool GiNaC::pseries::is_zero ( ) const [inline]
```

Check whether series has the value zero.

References seq.

Referenced by power\_const().

#### 6.138.3.27 is\_terminating()

```
bool GiNaC::pseries::is_terminating ( ) const
```

Returns true if there is no order term, i.e.

the series terminates and false otherwise.

References GiNaC::is\_order\_function(), and seq.

Referenced by GiNaC::is\_terminating(), and GiNaC::log\_series().

#### 6.138.3.28 coeffop()

```
ex GiNaC::pseries::coeffop (
    size_t i ) const
```

Get coefficients and exponents.

References nops(), and seq.

#### 6.138.3.29 exponop()

```
ex GiNaC::pseries::exponop (
    size_t i ) const
```

References nops(), and seq.

#### 6.138.3.30 add\_series()

```
ex GiNaC::pseries::add_series (
    const pseries & other ) const
```

Add one series object to another, producing a pseries object that represents the sum.

**Parameters**

<i>other</i>	pseries object to add with
--------------	----------------------------

**Returns**

the sum as a pseries

References `GiNaC::_ex0`, `GiNaC::_ex1`, `is_compatible_to()`, `GiNaC::is_order_function()`, `GiNaC::ex::is_zero()`, `point`, `pseries()`, `seq`, `GiNaC::to_int()`, and `var`.

Referenced by `GiNaC::log_series()`.

**6.138.3.31 mul\_const()**

```
ex GiNaC::pseries::mul_const (
    const numeric & other ) const
```

Multiply a pseries object with a numeric constant, producing a pseries object that represents the product.

**Parameters**

<i>other</i>	constant to multiply with
--------------	---------------------------

**Returns**

the product as a pseries

References `GiNaC::numeric::coeff()`, `GiNaC::is_order_function()`, `point`, `pseries()`, `seq`, and `var`.

Referenced by `GiNaC::mul::series()`.

**6.138.3.32 mul\_series()**

```
ex GiNaC::pseries::mul_series (
    const pseries & other ) const
```

Multiply one pseries object to another, producing a pseries object that represents the product.

**Parameters**

<i>other</i>	pseries object to multiply with
--------------	---------------------------------

**Returns**

the product as a pseries

References `GiNaC::_ex0`, `GiNaC::_ex1`, `coeff()`, `degree()`, `GiNaC::ex::find()`, `is_compatible_to()`, `GiNaC::ex::is_equal()`, `GiNaC::is_order_function()`, `GiNaC::ex::is_zero()`, `ldegree()`, `point`, `pseries()`, `seq`, `GiNaC::to_int()`, and `var`.

Referenced by `GiNaC::mul::series()`.

**6.138.3.33 power\_const()**

```
ex GiNaC::pseries::power_const (
    const numeric & p,
    int deg ) const
```

Compute the p-th power of a series.

**Parameters**

<i>p</i>	power to compute
<i>deg</i>	truncation order of series calculation

References `GiNaC::_ex0`, `GiNaC::_ex1`, `c`, `coeff()`, `GiNaC::is_integer()`, `GiNaC::numeric::is_negative()`, `GiNaC::is_order_function()`, `is_zero()`, `GiNaC::numeric::is_zero()`, `ldegree()`, `point`, `GiNaC::pow()`, `pseries()`, `GiNaC::numeric::real()`, `seq`, `GiNaC::to_int()`, and `var`.

**6.138.3.34 shift\_exponents()**

```
pseries GiNaC::pseries::shift_exponents (
    int deg ) const
```

Return a new pseries object with the powers shifted by deg.

References `point`, `pseries()`, `seq`, and `var`.

**6.138.3.35 print\_series()**

```
void GiNaC::pseries::print_series (
    const print_context & c,
    const char * openbrace,
    const char * closebrace,
    const char * mul_sym,
    const char * pow_sym,
    unsigned level ) const [protected]
```

References `GiNaC::_ex1`, `c`, `GiNaC::ex::coeff()`, `GiNaC::is_order_function()`, `GiNaC::ex::is_zero()`, `GiNaC::info_flags::negative`, `GiNaC::info_flags::numeric`, `point`, `GiNaC::info_flags::positive`, `GiNaC::pow()`, `precedence()`, `GiNaC::ex::print()`, `GiNaC::basic::print()`, `seq`, and `var`.

Referenced by `do_print()`, `do_print_latex()`, and `do_print_python()`.

#### 6.138.3.36 do\_print()

```
void GiNaC::pseries::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_series\(\)](#).

#### 6.138.3.37 do\_print\_latex()

```
void GiNaC::pseries::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_series\(\)](#).

#### 6.138.3.38 do\_print\_tree()

```
void GiNaC::pseries::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [point](#), [GiNaC::ex::print\(\)](#), [seq](#), and [var](#).

#### 6.138.3.39 do\_print\_python()

```
void GiNaC::pseries::do_print_python (
    const print\_python & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_series\(\)](#).

#### 6.138.3.40 do\_print\_python\_repr()

```
void GiNaC::pseries::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), [point](#), [GiNaC::ex::print\(\)](#), [seq](#), and [var](#).

### 6.138.4 Member Data Documentation

#### 6.138.4.1 seq

`epvector` GiNaC::pseries::seq [protected]

Vector of {coefficient, power} pairs.

Referenced by `add_series()`, `archive()`, `coeff()`, `coeffop()`, `conjugate()`, `convert_to_poly()`, `degree()`, `derivative()`, `do_print_python_repr()`, `do_print_tree()`, `eval_integ()`, `evalf()`, `evalm()`, `expand()`, `exponop()`, `imag_part()`, `is_terminating()`, `is_zero()`, `ldegree()`, `mul_const()`, `mul_series()`, `nops()`, `normal()`, `op()`, `power_const()`, `print_series()`, `pseries()`, `read_archive()`, `real_part()`, `series()`, `shift_exponents()`, and `subs()`.

#### 6.138.4.2 var

`ex` GiNaC::pseries::var [protected]

Series variable (holds a symbol)

Referenced by `add_series()`, `archive()`, `coeff()`, `conjugate()`, `convert_to_poly()`, `degree()`, `derivative()`, `do_print_python_repr()`, `do_print_tree()`, `eval_integ()`, `evalf()`, `evalm()`, `expand()`, `get_var()`, `imag_part()`, `is_compatible_to()`, `ldegree()`, `mul_const()`, `mul_series()`, `normal()`, `op()`, `power_const()`, `print_series()`, `pseries()`, `read_archive()`, `real_part()`, `series()`, `shift_exponents()`, and `subs()`.

#### 6.138.4.3 point

`ex` GiNaC::pseries::point [protected]

Expansion point.

Referenced by `add_series()`, `archive()`, `conjugate()`, `convert_to_poly()`, `derivative()`, `do_print_python_repr()`, `do_print_tree()`, `eval_integ()`, `evalf()`, `evalm()`, `expand()`, `get_point()`, `imag_part()`, `is_compatible_to()`, `mul_const()`, `mul_series()`, `normal()`, `op()`, `power_const()`, `print_series()`, `pseries()`, `read_archive()`, `real_part()`, `series()`, `shift_exponents()`, and `subs()`.

The documentation for this class was generated from the following files:

- [pseries.h](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.139 GiNaC::psi1\_SERIAL Class Reference

Polylogarithm and multiple polylogarithm.

```
#include <inifcns.h>
```

## Static Public Attributes

- static unsigned [serial](#)

### 6.139.1 Detailed Description

Polylogarithm and multiple polylogarithm.

Nielsen's generalized polylogarithm. Harmonic polylogarithm. Gamma-function. Beta-function. Psi-function (aka digamma-function).

### 6.139.2 Member Data Documentation

#### 6.139.2.1 [serial](#)

```
unsigned GiNaC::psil_SERIAL::serial [static]
```

**Initial value:**

```
=
    function::register_new(function_options("psi", 1).
        eval_func(psil_eval).
        evalf_func(psil_evalf).
        derivative_func(psil_deriv).
        series_func(psil_series).
        latex_name("\\psi").
        overloaded(2))
```

Referenced by `GiNaC::psi()`.

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_gamma.cpp](#)

## 6.140 GiNaC::psi2\_SERIAL Class Reference

Derivatives of Psi-function (aka polygamma-functions).

```
#include <inifcns.h>
```

## Static Public Attributes

- static unsigned [serial](#)



### 6.140.1 Detailed Description

Derivatives of Psi-function (aka polygamma-functions).

### 6.140.2 Member Data Documentation

#### 6.140.2.1 serial

```
unsigned GiNaC::psi2_SERIAL::serial [static]
```

**Initial value:**

```
=
    function::register_new(function_options("psi", 2).
        eval_func(psi2_eval).
        evalf_func(psi2_evalf).
        derivative_func(psi2_deriv).
        series_func(psi2_series).
        latex_name("\\psi").
        overloaded(2))
```

Referenced by GiNaC::psi().

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_gamma.cpp](#)

## 6.141 GiNaC::ptr< T > Class Template Reference

Class of (intrusively) reference-counted pointers that support copy-on-write semantics.

```
#include <ptr.h>
```

### Public Member Functions

- [ptr](#) (T \*t) noexcept  
*Bind ptr to newly created object, start reference counting.*
- [ptr](#) (T &t) noexcept  
*Bind ptr to existing reference-counted object.*
- [ptr](#) (const [ptr](#) &other) noexcept
- [~ptr](#) ()
- [ptr](#) & [operator=](#) (const [ptr](#) &other)
- T & [operator\\*](#) () const noexcept
- T \* [operator->](#) () const noexcept
- void [makewritable](#) ()  
*Announce your intention to modify the object bound to this ptr.*
- void [swap](#) ([ptr](#) &other) noexcept  
*Swap the bound object of this ptr with another ptr.*
- template<class U >  
bool [operator==](#) (const [ptr](#)< U > &rhs) const noexcept
- template<class U >  
bool [operator!=](#) (const [ptr](#)< U > &rhs) const noexcept

## Private Attributes

- `T * p`

## Friends

- `struct std::less< ptr< T > >`
- `T * get_pointer (const ptr &x) noexcept`
- `template<class U >`  
`bool operator== (const ptr &lhs, const U *rhs) noexcept`
- `template<class U >`  
`bool operator!= (const ptr &lhs, const U *rhs) noexcept`
- `template<class U >`  
`bool operator== (const U *lhs, const ptr &rhs) noexcept`
- `template<class U >`  
`bool operator!= (const U *lhs, const ptr &rhs) noexcept`
- `std::ostream & operator<< (std::ostream &os, const ptr< T > &rhs)`

### 6.141.1 Detailed Description

```
template<class T>
class GiNaC::ptr< T >
```

Class of (intrusively) reference-counted pointers that support copy-on-write semantics.

Requirements for T: must support the refcounted interface (usually by being derived from refcounted) `T* T↔::duplicate()` member function (only if `makewriteable()` is used)

### 6.141.2 Constructor & Destructor Documentation

#### 6.141.2.1 ptr() [1/3]

```
template<class T>
GiNaC::ptr< T >::ptr (
    T * t ) [inline], [noexcept]
```

Bind ptr to newly created object, start reference counting.

#### 6.141.2.2 ptr() [2/3]

```
template<class T>
GiNaC::ptr< T >::ptr (
    T & t ) [inline], [explicit], [noexcept]
```

Bind ptr to existing reference-counted object.

### 6.141.2.3 ptr() [3/3]

```
template<class T>
GiNaC::ptr< T >::ptr (
    const ptr< T > & other ) [inline], [noexcept]
```

### 6.141.2.4 ~ptr()

```
template<class T>
GiNaC::ptr< T >::~~ptr ( ) [inline]
```

## 6.141.3 Member Function Documentation

### 6.141.3.1 operator=()

```
template<class T>
ptr& GiNaC::ptr< T >::operator= (
    const ptr< T > & other ) [inline]
```

### 6.141.3.2 operator\*()

```
template<class T>
T& GiNaC::ptr< T >::operator* ( ) const [inline], [noexcept]
```

### 6.141.3.3 operator->()

```
template<class T>
T* GiNaC::ptr< T >::operator-> ( ) const [inline], [noexcept]
```

### 6.141.3.4 makewritable()

```
template<class T>
void GiNaC::ptr< T >::makewritable ( ) [inline]
```

Announce your intention to modify the object bound to this ptr.

This ensures that the object is not shared by any other ptrs.

#### 6.141.3.5 swap()

```
template<class T>
void GiNaC::ptr< T >::swap (
    ptr< T > & other ) [inline], [noexcept]
```

Swap the bound object of this ptr with another ptr.

#### 6.141.3.6 operator==()

```
template<class T>
template<class U >
bool GiNaC::ptr< T >::operator== (
    const ptr< U > & rhs ) const [inline], [noexcept]
```

#### 6.141.3.7 operator!=(())

```
template<class T>
template<class U >
bool GiNaC::ptr< T >::operator!= (
    const ptr< U > & rhs ) const [inline], [noexcept]
```

### 6.141.4 Friends And Related Function Documentation

#### 6.141.4.1 std::less< ptr< T > >

```
template<class T>
friend struct std::less< ptr< T > > [friend]
```

#### 6.141.4.2 get\_pointer

```
template<class T>
T* get_pointer (
    const ptr< T > & x ) [friend]
```

Referenced by GiNaC::ptr< GiNaC::basic >::operator!=(()), and GiNaC::ptr< GiNaC::basic >::operator==().

#### 6.141.4.3 operator== [1/2]

```
template<class T>
template<class U >
bool operator== (
    const ptr< T > & lhs,
    const U * rhs ) [friend]
```

#### 6.141.4.4 operator!= [1/2]

```
template<class T>
template<class U >
bool operator!= (
    const ptr< T > & lhs,
    const U * rhs ) [friend]
```

#### 6.141.4.5 operator== [2/2]

```
template<class T>
template<class U >
bool operator== (
    const U * lhs,
    const ptr< T > & rhs ) [friend]
```

#### 6.141.4.6 operator!= [2/2]

```
template<class T>
template<class U >
bool operator!= (
    const U * lhs,
    const ptr< T > & rhs ) [friend]
```

#### 6.141.4.7 operator<<

```
template<class T>
std::ostream& operator<< (
    std::ostream & os,
    const ptr< T > & rhs ) [friend]
```

### 6.141.5 Member Data Documentation

### 6.141.5.1 p

```
template<class T>
T* GiNaC::ptr< T >::p [private]
```

Referenced by `GiNaC::ptr< GiNaC::basic >::makewritable()`, `GiNaC::ptr< GiNaC::basic >::operator!=()`, `GiNaC::ptr< GiNaC::basic >::operator*()`, `GiNaC::ptr< GiNaC::basic >::operator->()`, `GiNaC::ptr< GiNaC::basic >::operator=()`, `GiNaC::ptr< GiNaC::basic >::operator==()`, `GiNaC::ptr< GiNaC::basic >::ptr()`, `GiNaC::ptr< GiNaC::basic >::swap()`, and `GiNaC::ptr< GiNaC::basic >::~~ptr()`.

The documentation for this class was generated from the following file:

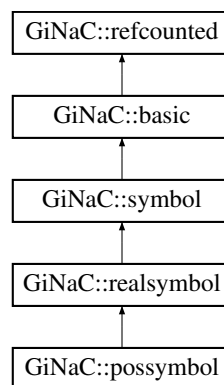
- [ptr.h](#)

## 6.142 GiNaC::realsymbol Class Reference

Specialization of symbol to real domain.

```
#include <symbol.h>
```

Inheritance diagram for `GiNaC::realsymbol`:



### Public Member Functions

- [realsymbol](#) ()
- [realsymbol](#) (const std::string &initname)
- [realsymbol](#) (const std::string &initname, const std::string &texname)
- unsigned [get\\_domain](#) () const override
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- [realsymbol](#) \* [duplicate](#) () const override

*Create a clone of this object on the heap.*

### Additional Inherited Members

#### 6.142.1 Detailed Description

Specialization of symbol to real domain.

## 6.142.2 Constructor & Destructor Documentation

### 6.142.2.1 realsymbol() [1/3]

```
GiNaC::realsymbol::realsymbol ( )
```

Referenced by `duplicate()`.

### 6.142.2.2 realsymbol() [2/3]

```
GiNaC::realsymbol::realsymbol (
    const std::string & initname ) [explicit]
```

### 6.142.2.3 realsymbol() [3/3]

```
GiNaC::realsymbol::realsymbol (
    const std::string & initname,
    const std::string & texname )
```

## 6.142.3 Member Function Documentation

### 6.142.3.1 get\_domain()

```
unsigned GiNaC::realsymbol::get_domain ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

Reimplemented in [GiNaC::possymbol](#).

References [GiNaC::domain::real](#).

### 6.142.3.2 conjugate()

```
ex GiNaC::realsymbol::conjugate ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

#### 6.142.3.3 `real_part()`

```
ex GiNaC::realsymbol::real_part ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

#### 6.142.3.4 `imag_part()`

```
ex GiNaC::realsymbol::imag_part ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

#### 6.142.3.5 `duplicate()`

```
realsymbol* GiNaC::realsymbol::duplicate ( ) const [inline], [override], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the refcounted construction of an ex from a basic.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::possymbol](#).

References [GiNaC::status\\_flags::dynallocated](#), [realsymbol\(\)](#), and [GiNaC::basic::setflag\(\)](#).

The documentation for this class was generated from the following files:

- [symbol.h](#)
- [symbol.cpp](#)

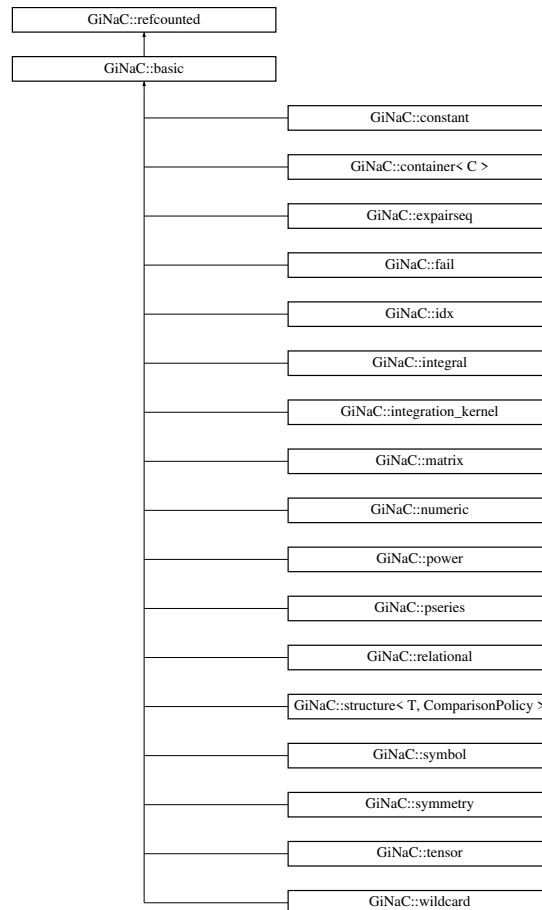


## 6.143 GiNaC::refcounted Class Reference

Base class for reference-counted objects.

```
#include <ptr.h>
```

Inheritance diagram for GiNaC::refcounted:



### Public Member Functions

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Private Attributes

- unsigned int [refcount](#)  
*reference counter*

#### 6.143.1 Detailed Description

Base class for reference-counted objects.

## 6.143.2 Constructor & Destructor Documentation

### 6.143.2.1 refcounted()

```
GiNaC::refcounted::refcounted ( ) [inline], [noexcept]
```

## 6.143.3 Member Function Documentation

### 6.143.3.1 add\_reference()

```
unsigned int GiNaC::refcounted::add_reference ( ) [inline], [noexcept]
```

References refcount.

### 6.143.3.2 remove\_reference()

```
unsigned int GiNaC::refcounted::remove_reference ( ) [inline], [noexcept]
```

References refcount.

### 6.143.3.3 get\_refcount()

```
unsigned int GiNaC::refcounted::get_refcount ( ) const [inline], [noexcept]
```

References refcount.

Referenced by `GiNaC::ex::construct_from_basic()`, and `GiNaC::basic::~~basic()`.

### 6.143.3.4 set\_refcount()

```
void GiNaC::refcounted::set_refcount (
    unsigned int r ) [inline], [noexcept]
```

References `r`, and `refcount`.

### 6.143.4 Member Data Documentation

#### 6.143.4.1 refcount

```
unsigned int GiNaC::refcounted::refcount [private]
```

reference counter

Referenced by `add_reference()`, `get_refcount()`, `remove_reference()`, and `set_refcount()`.

The documentation for this class was generated from the following file:

- [ptr.h](#)

## 6.144 GiNaC::registered\_class\_options Class Reference

This class stores information about a registered [GiNaC](#) class.

```
#include <registrar.h>
```

### Public Member Functions

- [registered\\_class\\_options](#) (const char \*n, const char \*p, const std::type\_info &ti)
- const char \* [get\\_name](#) () const
- const char \* [get\\_parent\\_name](#) () const
- std::type\_info const \* [get\\_id](#) () const
- const std::vector< [print\\_functor](#) > & [get\\_print\\_dispatch\\_table](#) () const
- template<class Ctx, class T, class C >  
[registered\\_class\\_options](#) & [print\\_func](#) (void f(const T &, const C &c, unsigned))
- template<class Ctx, class T, class C >  
[registered\\_class\\_options](#) & [print\\_func](#) (void(T::\*f)(const C &, unsigned))
- template<class Ctx >  
[registered\\_class\\_options](#) & [print\\_func](#) (const [print\\_functor](#) &f)
- void [set\\_print\\_func](#) (unsigned id, const [print\\_functor](#) &f)

### Private Attributes

- const char \* [name](#)  
*Class name.*
- const char \* [parent\\_name](#)  
*Name of superclass.*
- std::type\_info const \* [tinfo\\_key](#)  
*Type information key.*
- std::vector< [print\\_functor](#) > [print\\_dispatch\\_table](#)  
*Method table for print() dispatch.*

### 6.144.1 Detailed Description

This class stores information about a registered [GiNaC](#) class.

### 6.144.2 Constructor & Destructor Documentation

#### 6.144.2.1 registered\_class\_options()

```
GiNaC::registered_class_options::registered_class_options (
    const char * n,
    const char * p,
    const std::type_info & ti ) [inline]
```

### 6.144.3 Member Function Documentation

#### 6.144.3.1 get\_name()

```
const char* GiNaC::registered_class_options::get_name ( ) const [inline]
```

References name.

#### 6.144.3.2 get\_parent\_name()

```
const char* GiNaC::registered_class_options::get_parent_name ( ) const [inline]
```

References parent\_name.

#### 6.144.3.3 get\_id()

```
std::type_info const* GiNaC::registered_class_options::get_id ( ) const [inline]
```

References tinfo\_key.

6.144.3.4 `get_print_dispatch_table()`

```
const std::vector<print_functor>& GiNaC::registered_class_options::get_print_dispatch_table (
) const [inline]
```

References `print_dispatch_table`.

6.144.3.5 `print_func()` [1/3]

```
template<class Ctx , class T , class C >
registered_class_options& GiNaC::registered_class_options::print_func (
    void fconst T &, const C &c, unsigned ) [inline]
```

References `options`, and `set_print_func()`.

6.144.3.6 `print_func()` [2/3]

```
template<class Ctx , class T , class C >
registered_class_options& GiNaC::registered_class_options::print_func (
    void(T::*)(const C &, unsigned) f ) [inline]
```

References `options`, and `set_print_func()`.

6.144.3.7 `print_func()` [3/3]

```
template<class Ctx >
registered_class_options& GiNaC::registered_class_options::print_func (
    const print_functor & f ) [inline]
```

References `options`, and `set_print_func()`.

6.144.3.8 `set_print_func()`

```
void GiNaC::registered_class_options::set_print_func (
    unsigned id,
    const print_functor & f ) [inline]
```

References `print_dispatch_table`.

Referenced by `print_func()`.

#### 6.144.4 Member Data Documentation

##### 6.144.4.1 name

```
const char* GiNaC::registered_class_options::name [private]
```

Class name.

Referenced by `get_name()`.

##### 6.144.4.2 parent\_name

```
const char* GiNaC::registered_class_options::parent_name [private]
```

Name of superclass.

Referenced by `get_parent_name()`.

##### 6.144.4.3 tinfo\_key

```
std::type_info const* GiNaC::registered_class_options::tinfo_key [private]
```

Type information key.

Referenced by `get_id()`.

##### 6.144.4.4 print\_dispatch\_table

```
std::vector<print_func_t> GiNaC::registered_class_options::print_dispatch_table [private]
```

Method table for `print()` dispatch.

Referenced by `get_print_dispatch_table()`, and `set_print_func()`.

The documentation for this class was generated from the following file:

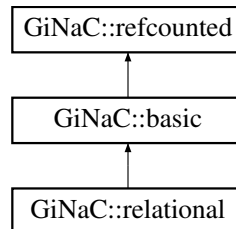
- [registrar.h](#)

## 6.145 GiNaC::relational Class Reference

This class holds a relation consisting of two expressions and a logical relation between them.

```
#include <relational.h>
```

Inheritance diagram for GiNaC::relational:



### Classes

- struct [safe\\_bool\\_helper](#)

### Public Types

- enum [operators](#) {  
[equal](#), [not\\_equal](#), [less](#), [less\\_or\\_equal](#),  
[greater](#), [greater\\_or\\_equal](#) }

### Public Member Functions

- [relational](#) (const [ex](#) &[lhs](#), const [ex](#) &[rhs](#), [operators](#) oper=[equal](#))
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- size\_t [nops](#) () const override  
*Number of operands/members.*
- [ex](#) op (size\_t i) const override  
*Return operand/member at position i.*
- [ex](#) map ([map\\_function](#) &[f](#)) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- [ex](#) subs (const [exmap](#) &[m](#), unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- void [archive](#) ([archive\\_node](#) &[n](#)) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &[n](#), [lst](#) &[syms](#)) override  
*Read (a.k.a.*
- [ex](#) lhs () const
- [ex](#) rhs () const
- [operator safe\\_bool](#) () const  
*Cast the relational into a Boolean, mainly for evaluation within an if-statement.*
- [safe\\_bool operator!](#) () const

## Protected Member Functions

- `ex eval_ncmul` (const `exvector` &`v`) const override
- `bool match_same_type` (const `basic` &`other`) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `unsigned return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `unsigned calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void do_print` (const `print_context` &`c`, unsigned level) const
- `void do_print_python_repr` (const `print_python_repr` &`c`, unsigned level) const

## Protected Attributes

- `ex lh`
- `ex rh`
- `operators o`

## Private Types

- `typedef void(safe_bool_helper::* safe_bool)` ()

## Private Member Functions

- `safe_bool make_safe_bool` (bool) const

### 6.145.1 Detailed Description

This class holds a relation consisting of two expressions and a logical relation between them.

### 6.145.2 Member Typedef Documentation

#### 6.145.2.1 `safe_bool`

```
typedef void(safe_bool_helper::* GiNaC::relational::safe_bool) () [private]
```

### 6.145.3 Member Enumeration Documentation

#### 6.145.3.1 `operators`

```
enum GiNaC::relational::operators
```



## Enumerator

equal	
not_equal	
less	
less_or_equal	
greater	
greater_or_equal	

## 6.145.4 Constructor &amp; Destructor Documentation

## 6.145.4.1 relational()

```
GiNaC::relational::relational (
    const ex & lhs,
    const ex & rhs,
    operators oper = equal )
```

Referenced by subs().

## 6.145.5 Member Function Documentation

## 6.145.5.1 precedence()

```
unsigned GiNaC::relational::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by do\_print().

## 6.145.5.2 info()

```
bool GiNaC::relational::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

## See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [equal](#), [greater](#), [greater\\_or\\_equal](#), [less](#), [less\\_or\\_equal](#), [not\\_equal](#), [o](#), [GiNaC::info\\_flags::relation](#), [GiNaC::info\\_flags::relation\\_equal](#), [GiNaC::info\\_flags::relation\\_greater](#), [GiNaC::info\\_flags::relation\\_greater\\_or\\_equal](#), [GiNaC::info\\_flags::relation\\_less](#), [GiNaC::info\\_flags::relation\\_less\\_or\\_equal](#), and [GiNaC::info\\_flags::relation\\_not\\_equal](#).

#### 6.145.5.3 nops()

```
size_t GiNaC::relational::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 6.145.5.4 op()

```
ex GiNaC::relational::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References GINAC\_ASSERT, lh, and rh.

#### 6.145.5.5 map()

```
ex GiNaC::relational::map (
    map_function & f ) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References GiNaC::are\_ex\_trivially\_equal(), lh, o, and rh.

#### 6.145.5.6 subs()

```
ex GiNaC::relational::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References GiNaC::are\_ex\_trivially\_equal(), lh, m, o, options, relational(), rh, GiNaC::ex::subs(), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

## 6.145.5.7 archive()

```
void GiNaC::relational::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a. serialize) object into archive.  
Reimplemented from [GiNaC::basic](#).  
References lh, n, o, and rh.

## 6.145.5.8 read\_archive()

```
void GiNaC::relational::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a. deserialize) object from archive.  
Reimplemented from [GiNaC::basic](#).  
References lh, n, o, and rh.

## 6.145.5.9 eval\_ncmul()

```
ex GiNaC::relational::eval_ncmul (
    const exvector & v ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).  
References [GiNaC::ex::eval\\_ncmul\(\)](#), and lh.

## 6.145.5.10 match\_same\_type()

```
bool GiNaC::relational::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).  
References GINAC\_ASSERT, and o.

**6.145.5.11 return\_type()**

```
unsigned GiNaC::relational::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References GINAC\_ASSERT, lh, GiNaC::ex::return\_type(), and rh.

**6.145.5.12 return\_type\_tinfo()**

```
return_type_t GiNaC::relational::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References GINAC\_ASSERT, lh, GiNaC::ex::return\_type\_tinfo(), and rh.

**6.145.5.13 calchash()**

```
unsigned GiNaC::relational::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References equal, GiNaC::status\_flags::evaluated, GiNaC::basic::flags, GiNaC::ex::gethash(), greater, greater\_↔  
or\_equal, GiNaC::status\_flags::hash\_calculated, GiNaC::basic::hashvalue, less, less\_or\_equal, lh, GiNaC::make\_↔  
\_hash\_seed(), not\_equal, o, rh, GiNaC::rotate\_left(), and GiNaC::basic::setflag().

**6.145.5.14 do\_print()**

```
void GiNaC::relational::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References c, lh, o, precedence(), GiNaC::ex::print(), GiNaC::print\_operator(), and rh.

## 6.145.5.15 do\_print\_python\_repr()

```
void GiNaC::relational::do_print_python_repr (
    const print_python_repr & c,
    unsigned level ) const [protected]
```

References `c`, `lh`, `o`, `GiNaC::ex::print()`, `GiNaC::print_operator()`, and `rh`.

## 6.145.5.16 lhs()

```
ex GiNaC::relational::lhs ( ) const [inline]
```

References `lh`.

Referenced by `GiNaC::atan_series()`, `GiNaC::atanh_series()`, `GiNaC::beta_series()`, `GiNaC::Li2_series()`, `GiNaC::log_series()`, `GiNaC::tan_series()`, and `GiNaC::tanh_series()`.

## 6.145.5.17 rhs()

```
ex GiNaC::relational::rhs ( ) const [inline]
```

References `rh`.

Referenced by `GiNaC::atan_series()`, `GiNaC::atanh_series()`, `GiNaC::Li2_series()`, and `GiNaC::log_series()`.

## 6.145.5.18 make\_safe\_bool()

```
relational::safe_bool GiNaC::relational::make_safe_bool (
    bool cond ) const [private]
```

References `GiNaC::relational::safe_bool_helper::nonnull()`.

Referenced by operator `safe_bool()`, and operator `!()`.

## 6.145.5.19 operator safe\_bool()

```
GiNaC::relational::operator relational::safe_bool ( ) const
```

Cast the relational into a Boolean, mainly for evaluation within an if-statement.

Note that  $(a < b) == \text{false}$  does not imply  $(a \geq b) == \text{true}$  in the general symbolic case. A false result means the comparison is either false or undecidable (except of course for `!=`, where true means either unequal or undecidable).

References `GiNaC::_num0_p`, `equal`, `greater`, `greater_or_equal`, `GiNaC::is_zero()`, `less`, `less_or_equal`, `lh`, `make_safe_bool()`, `not_equal`, `o`, and `rh`.

#### 6.145.5.20 operator!()

```
relational::safe_bool GiNaC::relational::operator! ( ) const [inline]
```

References `make_safe_bool()`.

### 6.145.6 Member Data Documentation

#### 6.145.6.1 lh

```
ex GiNaC::relational::lh [protected]
```

Referenced by `archive()`, `calchash()`, `do_print()`, `do_print_python_repr()`, `eval_ncmul()`, `lhs()`, `map()`, `op()`, `operator safe_bool()`, `read_archive()`, `return_type()`, `return_type_tinfo()`, and `subs()`.

#### 6.145.6.2 rh

```
ex GiNaC::relational::rh [protected]
```

Referenced by `archive()`, `calchash()`, `do_print()`, `do_print_python_repr()`, `map()`, `op()`, `operator safe_bool()`, `read_archive()`, `return_type()`, `return_type_tinfo()`, `rhs()`, and `subs()`.

#### 6.145.6.3 o

```
operators GiNaC::relational::o [protected]
```

Referenced by `archive()`, `calchash()`, `do_print()`, `do_print_python_repr()`, `info()`, `map()`, `match_same_type()`, `operator safe_bool()`, `read_archive()`, and `subs()`.

The documentation for this class was generated from the following files:

- [relational.h](#)
- [relational.cpp](#)

## 6.146 GiNaC::remember\_strategies Class Reference

Strategies how to clean up the function remember cache.

```
#include <flags.h>
```

## Public Types

- enum { [delete\\_never](#), [delete\\_lru](#), [delete\\_lfu](#), [delete\\_cyclic](#) }

### 6.146.1 Detailed Description

Strategies how to clean up the function remember cache.

See also

[remember\\_table](#)

### 6.146.2 Member Enumeration Documentation

#### 6.146.2.1 anonymous enum

anonymous enum

#### Enumerator

<code>delete_never</code>	Let table grow indefinitely.
<code>delete_lru</code>	Least recently used.
<code>delete_lfu</code>	Least frequently used.
<code>delete_cyclic</code>	First (oldest) one in list.

The documentation for this class was generated from the following file:

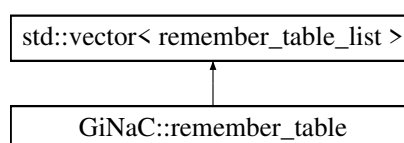
- [flags.h](#)

## 6.147 GiNaC::remember\_table Class Reference

The remember table is organized like an n-fold associative cache in a microprocessor.

```
#include <remember.h>
```

Inheritance diagram for GiNaC::remember\_table:



## Public Member Functions

- [remember\\_table](#) ()
- [remember\\_table](#) (unsigned s, unsigned as, unsigned strat)
- bool [lookup\\_entry](#) (function const &f, [ex](#) &result) const
- void [add\\_entry](#) (function const &f, [ex](#) const &result)
- void [clear\\_all\\_entries](#) ()
- void [show\\_statistics](#) (std::ostream &os, unsigned level) const

## Static Public Member Functions

- static std::vector< [remember\\_table](#) > & [remember\\_tables](#) ()

## Protected Member Functions

- void [init\\_table](#) ()

## Protected Attributes

- unsigned [table\\_size](#)
- unsigned [max\\_assoc\\_size](#)
- unsigned [remember\\_strategy](#)

### 6.147.1 Detailed Description

The remember table is organized like an n-fold associative cache in a microprocessor.

The table has a width of 's' (which is rounded to `table_size`, some power of 2 near 's', internally) and a depth of 'as' (unless you choose that entries are never discarded). The place where an entry is stored depends on the hashvalue of the parameters of the function (this corresponds to the address of byte to be cached). The '`log_2(table_size)`' least significant bits of this hashvalue give the slot in which the entry will be stored or looked up. Each slot can take up to 'as' entries. If a slot is full, an older entry is removed by one of the following strategies:

- oldest entry (the first one in the list)
- least recently used (the one with the lowest '`last_access`')
- least frequently used (the one with the lowest '`successful_hits`') or all entries are kept which means that the table grows indefinitely.

### 6.147.2 Constructor & Destructor Documentation

#### 6.147.2.1 `remember_table()` [1/2]

```
GiNaC::remember_table::remember_table ( )
```

References `GiNaC::remember_strategies::delete_never`, `max_assoc_size`, `remember_strategy`, and `table_size`.



### 6.147.2.2 remember\_table() [2/2]

```
GiNaC::remember_table::remember_table (
    unsigned s,
    unsigned as,
    unsigned strat )
```

References `init_table()`, `GiNaC::log2()`, and `table_size`.

## 6.147.3 Member Function Documentation

### 6.147.3.1 lookup\_entry()

```
bool GiNaC::remember_table::lookup_entry (
    function const & f,
    ex & result ) const
```

References `GINAC_ASSERT`, and `table_size`.

### 6.147.3.2 add\_entry()

```
void GiNaC::remember_table::add_entry (
    function const & f,
    ex const & result )
```

References `GINAC_ASSERT`, and `table_size`.

### 6.147.3.3 clear\_all\_entries()

```
void GiNaC::remember_table::clear_all_entries ( )
```

References `init_table()`.

### 6.147.3.4 show\_statistics()

```
void GiNaC::remember_table::show_statistics (
    std::ostream & os,
    unsigned level ) const
```

#### 6.147.3.5 remember\_tables()

```
std::vector< remember_table > & GiNaC::remember_table::remember_tables ( ) [static]
```

Referenced by `GiNaC::function::lookup_remember_table()`, `GiNaC::function::register_new()`, and `GiNaC::function::store_remember_table()`.

#### 6.147.3.6 init\_table()

```
void GiNaC::remember_table::init_table ( ) [protected]
```

References `max_assoc_size`, `remember_strategy`, and `table_size`.

Referenced by `clear_all_entries()`, and `remember_table()`.

### 6.147.4 Member Data Documentation

#### 6.147.4.1 table\_size

```
unsigned GiNaC::remember_table::table_size [protected]
```

Referenced by `add_entry()`, `init_table()`, `lookup_entry()`, and `remember_table()`.

#### 6.147.4.2 max\_assoc\_size

```
unsigned GiNaC::remember_table::max_assoc_size [protected]
```

Referenced by `init_table()`, and `remember_table()`.

#### 6.147.4.3 remember\_strategy

```
unsigned GiNaC::remember_table::remember_strategy [protected]
```

Referenced by `init_table()`, and `remember_table()`.

The documentation for this class was generated from the following files:

- [remember.h](#)
- [remember.cpp](#)

## 6.148 GiNaC::remember\_table\_entry Class Reference

A single entry in the remember table of a function.

```
#include <remember.h>
```

### Public Member Functions

- [remember\\_table\\_entry](#) ([function](#) const &*f*, [ex](#) const &*r*)
- bool [is\\_equal](#) ([function](#) const &*f*) const
- [ex](#) [get\\_result](#) () const
- unsigned long [get\\_last\\_access](#) () const
- unsigned long [get\\_successful\\_hits](#) () const

### Protected Attributes

- unsigned [hashvalue](#)
- [exvector](#) [seq](#)
- [ex](#) [result](#)
- unsigned long [last\\_access](#)
- unsigned [successful\\_hits](#)

### Static Protected Attributes

- static unsigned long [access\\_counter](#) = 0

#### 6.148.1 Detailed Description

A single entry in the remember table of a function.

Needs to be a friend of class `function` to access '`seq`'. '`last_access`' and '`successful_hits`' are updated at each successful '`is_equal`'.

#### 6.148.2 Constructor & Destructor Documentation

##### 6.148.2.1 `remember_table_entry()`

```
GiNaC::remember_table_entry::remember_table_entry (  
    function const & f,  
    ex const & r )
```

References `access_counter`, `last_access`, and `successful_hits`.

### 6.148.3 Member Function Documentation

#### 6.148.3.1 is\_equal()

```
bool GiNaC::remember_table_entry::is_equal (
    function const & f ) const
```

References `access_counter`, `GINAC_ASSERT`, `hashvalue`, `last_access`, `seq`, and `successful_hits`.

#### 6.148.3.2 get\_result()

```
ex GiNaC::remember_table_entry::get_result ( ) const [inline]
```

References `result`.

#### 6.148.3.3 get\_last\_access()

```
unsigned long GiNaC::remember_table_entry::get_last_access ( ) const [inline]
```

References `last_access`.

#### 6.148.3.4 get\_successful\_hits()

```
unsigned long GiNaC::remember_table_entry::get_successful_hits ( ) const [inline]
```

### 6.148.4 Member Data Documentation

#### 6.148.4.1 hashvalue

```
unsigned GiNaC::remember_table_entry::hashvalue [protected]
```

Referenced by `is_equal()`.

#### 6.148.4.2 seq

`exvector` GiNaC::remember\_table\_entry::seq [protected]

Referenced by `is_equal()`.

#### 6.148.4.3 result

`ex` GiNaC::remember\_table\_entry::result [protected]

Referenced by `get_result()`.

#### 6.148.4.4 last\_access

`unsigned long` GiNaC::remember\_table\_entry::last\_access [mutable], [protected]

Referenced by `get_last_access()`, `is_equal()`, and `remember_table_entry()`.

#### 6.148.4.5 successful\_hits

`unsigned` GiNaC::remember\_table\_entry::successful\_hits [mutable], [protected]

Referenced by `is_equal()`, and `remember_table_entry()`.

#### 6.148.4.6 access\_counter

`unsigned long` GiNaC::remember\_table\_entry::access\_counter = 0 [static], [protected]

Referenced by `is_equal()`, and `remember_table_entry()`.

The documentation for this class was generated from the following files:

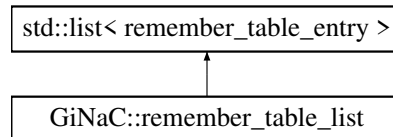
- [remember.h](#)
- [remember.cpp](#)

## 6.149 GiNaC::remember\_table\_list Class Reference

A list of entries in the remember table having some least significant bits of the hashvalue in common.

```
#include <remember.h>
```

Inheritance diagram for GiNaC::remember\_table\_list:



### Public Member Functions

- [remember\\_table\\_list](#) (unsigned *as*, unsigned *strat*)
- void [add\\_entry](#) (function const &*f*, *ex* const &*result*)
- bool [lookup\\_entry](#) (function const &*f*, *ex* &*result*) const

### Protected Attributes

- unsigned [max\\_assoc\\_size](#)
- unsigned [remember\\_strategy](#)

### 6.149.1 Detailed Description

A list of entries in the remember table having some least significant bits of the hashvalue in common.

### 6.149.2 Constructor & Destructor Documentation

#### 6.149.2.1 remember\_table\_list()

```
GiNaC::remember_table_list::remember_table_list (
    unsigned as,
    unsigned strat )
```

References [max\\_assoc\\_size](#), and [remember\\_strategy](#).

### 6.149.3 Member Function Documentation

## 6.149.3.1 add\_entry()

```
void GiNaC::remember_table_list::add_entry (
    function const & f,
    ex const & result )
```

References GiNaC::remember\_strategies::delete\_cyclic, GiNaC::remember\_strategies::delete\_lfu, GiNaC::remember\_strategies::delete\_lru, GiNaC::remember\_strategies::delete\_never, GINAC\_ASSERT, max\_assoc\_size, and remember\_strategy.

## 6.149.3.2 lookup\_entry()

```
bool GiNaC::remember_table_list::lookup_entry (
    function const & f,
    ex & result ) const
```

## 6.149.4 Member Data Documentation

## 6.149.4.1 max\_assoc\_size

```
unsigned GiNaC::remember_table_list::max_assoc_size [protected]
```

Referenced by add\_entry(), and remember\_table\_list().

## 6.149.4.2 remember\_strategy

```
unsigned GiNaC::remember_table_list::remember_strategy [protected]
```

Referenced by add\_entry(), and remember\_table\_list().

The documentation for this class was generated from the following files:

- [remember.h](#)
- [remember.cpp](#)

## 6.150 GiNaC::return\_type\_t Struct Reference

To distinguish between different kinds of non-commutative objects.

```
#include <registrar.h>
```

## Public Member Functions

- bool `operator<` (const `return_type_t` &other) const  
*Strict weak ordering (so one can put `return_type_t`'s into a STL container).*
- bool `operator==` (const `return_type_t` &other) const
- bool `operator!=` (const `return_type_t` &other) const

## Public Attributes

- `std::type_info` const \* `tinfo`  
*to distinguish between non-commutative objects of different type.*
- unsigned `rl`  
*to distinguish between non-commutative objects of the same type.*

### 6.150.1 Detailed Description

To distinguish between different kinds of non-commutative objects.

### 6.150.2 Member Function Documentation

#### 6.150.2.1 `operator<()`

```
bool GiNaC::return_type_t::operator< (
    const return_type_t & other ) const [inline]
```

Strict weak ordering (so one can put `return_type_t`'s into a STL container).

References `rl`, and `tinfo`.

#### 6.150.2.2 `operator==()`

```
bool GiNaC::return_type_t::operator== (
    const return_type_t & other ) const [inline]
```

References `rl`, and `tinfo`.

Referenced by `operator!=()`.



### 6.150.2.3 operator!=(())

```
bool GiNaC::return_type_t::operator!=(
    const return_type_t & other ) const [inline]
```

References operator==(()).

## 6.150.3 Member Data Documentation

### 6.150.3.1 tinfo

```
std::type_info const* GiNaC::return_type_t::tinfo
```

to distinguish between non-commutative objects of different type.

Referenced by `GiNaC::is_clifford_tinfo()`, `GiNaC::is_color_tinfo()`, `GiNaC::make_return_type_t()`, `operator<()`, `operator==(())`, and `GiNaC::basic::return_type_tinfo()`.

### 6.150.3.2 rl

```
unsigned GiNaC::return_type_t::rl
```

to distinguish between non-commutative objects of the same type.

Think of gamma matrices with different representation labels.

Referenced by `GiNaC::get_representation_label()`, `GiNaC::make_return_type_t()`, `operator<()`, `operator==(())`, and `GiNaC::basic::return_type_tinfo()`.

The documentation for this struct was generated from the following file:

- [registrar.h](#)

## 6.151 GiNaC::return\_types Class Reference

```
#include <flags.h>
```

### Public Types

- enum { `commutative`, `noncommutative`, `noncommutative_composite` }

## 6.151.1 Member Enumeration Documentation

### 6.151.1.1 anonymous enum

```
anonymous enum
```

## Enumerator

commutative	
noncommutative	
noncommutative_composite	

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.152 GiNaC::relational::safe\_bool\_helper Struct Reference

### Public Member Functions

- void [nonnull](#) ()

### 6.152.1 Member Function Documentation

#### 6.152.1.1 nonnull()

```
void GiNaC::relational::safe_bool_helper::nonnull ( ) [inline]
```

Referenced by `GiNaC::relational::make_safe_bool()`.

The documentation for this struct was generated from the following file:

- [relational.h](#)

## 6.153 GiNaC::scalar\_products Class Reference

Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).

```
#include <indexed.h>
```

## Public Member Functions

- void [add](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &sp)  
*Register scalar product pair and its value.*
- void [add](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim, const [ex](#) &sp)  
*Register scalar product pair and its value for a specific space dimension.*
- void [add\\_vectors](#) (const [lst](#) &l, const [ex](#) &dim=[wild](#)())  
*Register list of vectors.*
- void [clear](#) ()  
*Clear all registered scalar products.*
- bool [is\\_defined](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim) const  
*Check whether scalar product pair is defined.*
- [ex](#) [evaluate](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim) const  
*Return value of defined scalar product pair.*
- void [debugprint](#) () const

## Protected Attributes

- [spm](#)

### 6.153.1 Detailed Description

Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).

See also

[simplify\\_indexed](#)

### 6.153.2 Member Function Documentation

#### 6.153.2.1 [add\(\)](#) [1/2]

```
void GiNaC::scalar_products::add (
    const ex & v1,
    const ex & v2,
    const ex & sp )
```

Register scalar product pair and its value.

References [spm](#).

Referenced by [add\\_vectors\(\)](#).

**6.153.2.2 add()** [2/2]

```
void GiNaC::scalar_products::add (
    const ex & v1,
    const ex & v2,
    const ex & dim,
    const ex & sp )
```

Register scalar product pair and its value for a specific space dimension.

References spm.

**6.153.2.3 add\_vectors()**

```
void GiNaC::scalar_products::add_vectors (
    const lst & l,
    const ex & dim = wild() )
```

Register list of vectors.

This adds all possible pairs of products  $a.i * b.i$  with the value  $a*b$  (note that this is not a scalar vector product but an ordinary product of scalars).

References add().

**6.153.2.4 clear()**

```
void GiNaC::scalar_products::clear ( )
```

Clear all registered scalar products.

References spm.

**6.153.2.5 is\_defined()**

```
bool GiNaC::scalar_products::is_defined (
    const ex & v1,
    const ex & v2,
    const ex & dim ) const
```

Check whether scalar product pair is defined.

References spm.

Referenced by GiNaC::simplify\_indexed\_product().

## 6.153.2.6 evaluate()

```
ex GiNaC::scalar_products::evaluate (
    const ex & v1,
    const ex & v2,
    const ex & dim ) const
```

Return value of defined scalar product pair.

References spm.

Referenced by GiNaC::simplify\_indexed\_product().

## 6.153.2.7 debugprint()

```
void GiNaC::scalar_products::debugprint ( ) const
```

References k, and spm.

## 6.153.3 Member Data Documentation

## 6.153.3.1 spm

```
spmap GiNaC::scalar_products::spm [protected]
```

Referenced by add(), clear(), debugprint(), evaluate(), and is\_defined().

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)

## 6.154 GiNaC::series\_options Class Reference

Flags to control series expansion.

```
#include <flags.h>
```

## Public Types

- enum { [suppress\\_branchcut](#) = 0x0001 }

## 6.154.1 Detailed Description

Flags to control series expansion.

## 6.154.2 Member Enumeration Documentation

## 6.154.2.1 anonymous enum

```
anonymous enum
```

## Enumerator

suppress_branchcut	Suppress branch cuts in series expansion. Branch cuts manifest themselves as step functions, if this option is not passed. If it is passed and expansion at a point on a cut is performed, then the analytic continuation of the function is expanded.
--------------------	--

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.155 GiNaC::solve\_algo Class Reference

Switch to control algorithm for linear system solving.

```
#include <flags.h>
```

### Public Types

- enum {  
[automatic](#), [gauss](#), [divfree](#), [bareiss](#),  
[markowitz](#) }

### 6.155.1 Detailed Description

Switch to control algorithm for linear system solving.

### 6.155.2 Member Enumeration Documentation

#### 6.155.2.1 anonymous enum

```
anonymous enum
```

## Enumerator

automatic	Let the system choose. A heuristics is applied for automatic determination of a suitable algorithm.
gauss	<p>Gauss elimination. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)} / m_{k,k}^{(k)}$ <p>This algorithm is well-suited for numerical matrices but generally suffers from the expensive division (and computation of GCDs) at each step.</p>

## Enumerator

divfree	<p>Division-free elimination. This is a modification of Gauss elimination where the division by the pivot element is not carried out. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}$ <p>This algorithm is only there for the purpose of cross-checks. It suffers from exponential intermediate expression swell. Use it only for small systems.</p>
bareiss	<p>Bareiss fraction-free elimination. This is a modification of Gauss elimination where the division by the pivot element is <i>delayed</i> until it can be carried out without computing GCDs. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = (m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}) / m_{k-1,k-1}^{(k-1)}$ <p>(We have set <math>m_{-1,-1}^{(-1)} = 1</math> in order to avoid a case distinction in above formula.) It can be shown that nothing more than polynomial long division is needed for carrying out the division. This is generally the fastest algorithm for solving linear systems. In contrast to division-free elimination it only has a linear expression swell. For two-dimensional systems, the two algorithms are equivalent, however.</p>
markowitz	<p>Markowitz-ordered Gaussian elimination. Same as the usual Gaussian elimination, but with additional effort spent on selecting pivots that minimize fill-in. Faster than the methods above for large sparse matrices (particularly with symbolic coefficients), otherwise slightly slower than Gaussian elimination.</p>

The documentation for this class was generated from the following file:

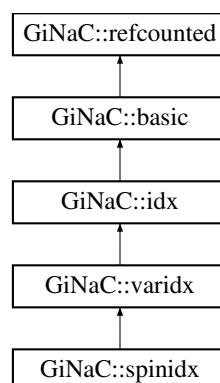
- [flags.h](#)

## 6.156 GiNaC::spindex Class Reference

This class holds a spinor index that can be dotted or undotted and that also has a variance.

```
#include <idx.h>
```

Inheritance diagram for GiNaC::spindex:



## Public Member Functions

- `spinidx` (const `ex` &`v`, const `ex` &`dim`=2, bool `covariant`=false, bool `dotted`=false)  
*Construct index with given value, dimension, variance and dot.*
- bool `is_dummy_pair_same_type` (const `basic` &`other`) const override  
*Check whether the index forms a dummy index pair with another index of the same type.*
- `ex conjugate` () const override
- void `archive` (`archive_node` &`n`) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override  
*Load (deserialize) the object from an archive node.*
- bool `is_dotted` () const  
*Check whether the index is dotted.*
- bool `is_undotted` () const  
*Check whether the index is not dotted.*
- `ex toggle_dot` () const  
*Make a new index with the same value and variance but the opposite dottedness.*
- `ex toggle_variance_dot` () const  
*Make a new index with the same value but opposite variance and dottedness.*

## Protected Member Functions

- bool `match_same_type` (const `basic` &`other`) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- void `do_print` (const `print_context` &`c`, unsigned level) const
- void `do_print_latex` (const `print_latex` &`c`, unsigned level) const
- void `do_print_tree` (const `print_tree` &`c`, unsigned level) const

## Protected Attributes

- bool `dotted`

### 6.156.1 Detailed Description

This class holds a spinor index that can be dotted or undotted and that also has a variance.

This is used in the Weyl-van-der-Waerden formalism where the dot indicates complex conjugation. There is an associated (asymmetric) metric tensor that can be used to raise/lower spinor indices.

### 6.156.2 Constructor & Destructor Documentation

#### 6.156.2.1 `spinidx()`

```
GiNaC::spinidx::spinidx (
    const ex & v,
    const ex & dim = 2,
    bool covariant = false,
    bool dotted = false )
```

Construct index with given value, dimension, variance and dot.



## Parameters

<i>v</i>	Value of index (numeric or symbolic)
<i>dim</i>	Dimension of index space (numeric or symbolic)
<i>covariant</i>	Make covariant index (default is contravariant)
<i>dotted</i>	Make covariant dotted (default is undotted)

## Returns

newly constructed index

## 6.156.3 Member Function Documentation

## 6.156.3.1 is\_dummy\_pair\_same\_type()

```
bool GiNaC::spinidx::is_dummy_pair_same_type (
    const basic & other ) const [override], [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented from [GiNaC::varidx](#).

References dotted.

## 6.156.3.2 conjugate()

```
ex GiNaC::spinidx::conjugate ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References toggle\_dot().

## 6.156.3.3 archive()

```
void GiNaC::spinidx::archive (
    archive_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::varidx](#).

References dotted, and n.

#### 6.156.3.4 read\_archive()

```
void GiNaC::spinidx::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

##### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::varidx](#).

References dotted, and n.

#### 6.156.3.5 match\_same\_type()

```
bool GiNaC::spinidx::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

##### See also

[basic::match](#)

Reimplemented from [GiNaC::varidx](#).

References dotted, and GINAC\_ASSERT.

#### 6.156.3.6 is\_dotted()

```
bool GiNaC::spinidx::is_dotted ( ) const [inline]
```

Check whether the index is dotted.

References dotted.

### 6.156.3.7 is\_undotted()

```
bool GiNaC::spinidx::is_undotted ( ) const [inline]
```

Check whether the index is not dotted.

References dotted.

### 6.156.3.8 toggle\_dot()

```
ex GiNaC::spinidx::toggle_dot ( ) const
```

Make a new index with the same value and variance but the opposite dottedness.

References GiNaC::basic::clearflag(), dotted, GiNaC::basic::duplicate(), and GiNaC::status\_flags::hash\_calculated.

Referenced by conjugate().

### 6.156.3.9 toggle\_variance\_dot()

```
ex GiNaC::spinidx::toggle_variance_dot ( ) const
```

Make a new index with the same value but opposite variance and dottedness.

References GiNaC::basic::clearflag(), GiNaC::varidx::covariant, dotted, GiNaC::basic::duplicate(), and GiNaC::status\_flags::hash\_calculated.

### 6.156.3.10 do\_print()

```
void GiNaC::spinidx::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References c, GiNaC::varidx::covariant, dotted, and GiNaC::idx::print\_index().

### 6.156.3.11 do\_print\_latex()

```
void GiNaC::spinidx::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

References c, dotted, and GiNaC::idx::print\_index().

### 6.156.3.12 do\_print\_tree()

```
void GiNaC::spinidx::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::varidx::covariant](#), [GiNaC::idx::dim](#), [dotted](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::ex::print\(\)](#), and [GiNaC::idx::value](#).

## 6.156.4 Member Data Documentation

### 6.156.4.1 dotted

```
bool GiNaC::spinidx::dotted [protected]
```

Referenced by [archive\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_tree\(\)](#), [is\\_dotted\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [is\\_undotted\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [toggle\\_dot\(\)](#), and [toggle\\_variance\\_dot\(\)](#).

The documentation for this class was generated from the following files:

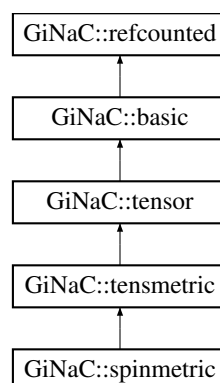
- [idx.h](#)
- [idx.cpp](#)

## 6.157 GiNaC::spinmetric Class Reference

This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::spinmetric:



## Public Member Functions

- bool [info](#) (unsigned *inf*) const override  
*Information about the object.*
- [ex eval\\_indexed](#) (const [basic](#) &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of an indexed spinor metric with something else.*

## Protected Member Functions

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

## Additional Inherited Members

### 6.157.1 Detailed Description

This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.

If indexed, it must have exactly two indices of the same type which must be of class `spinidx` or a subclass and have dimension 2.

### 6.157.2 Member Function Documentation

#### 6.157.2.1 `info()`

```
bool GiNaC::spinmetric::info (
    unsigned inf ) const    [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::info\\_flags::real](#).

### 6.157.2.2 eval\_indexed()

```
ex GiNaC::spinmetric::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed metric tensor.

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), and [GiNaC::to\\_int\(\)](#).

### 6.157.2.3 contract\_with()

```
bool GiNaC::spinmetric::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed spinor metric with something else.

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex2](#), [GiNaC::\\_ex\\_2](#), [GiNaC::delta\\_tensor\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::idx::is\\_symbolic\(\)](#), and [GiNaC::idx::subs\(\)](#).

### 6.157.2.4 do\_print()

```
void GiNaC::spinmetric::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

### 6.157.2.5 do\_print\_latex()

```
void GiNaC::spinmetric::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

- [tensor.h](#)
- [tensor.cpp](#)

## 6.158 GiNaC::spmapkey Class Reference

```
#include <indexed.h>
```

### Public Member Functions

- [spmapkey](#) ()
- [spmapkey](#) (const [ex](#) &[v1](#), const [ex](#) &[v2](#), const [ex](#) &[dim=wild\(\)](#))
- bool [operator==](#) (const [spmapkey](#) &other) const
- bool [operator<](#) (const [spmapkey](#) &other) const
- void [debugprint](#) () const

### Protected Attributes

- [ex v1](#)
- [ex v2](#)
- [ex dim](#)

## 6.158.1 Constructor & Destructor Documentation

### 6.158.1.1 [spmapkey\(\)](#) [1/2]

```
GiNaC::spmapkey::spmapkey ( ) [inline]
```

### 6.158.1.2 [spmapkey\(\)](#) [2/2]

```
GiNaC::spmapkey::spmapkey (
    const ex & v1,
    const ex & v2,
    const ex & dim = wild\(\) )
```

References [GiNaC::ex::compare\(\)](#), [GiNaC::ex::op\(\)](#), [v1](#), and [v2](#).

## 6.158.2 Member Function Documentation

#### 6.158.2.1 operator==()

```
bool GiNaC::spmapkey::operator== (
    const spmapkey & other ) const
```

References [dim](#), [GiNaC::ex::is\\_equal\(\)](#), [v1](#), and [v2](#).

#### 6.158.2.2 operator<()

```
bool GiNaC::spmapkey::operator< (
    const spmapkey & other ) const
```

References [GiNaC::ex::compare\(\)](#), [dim](#), [v1](#), and [v2](#).

#### 6.158.2.3 debugprint()

```
void GiNaC::spmapkey::debugprint ( ) const
```

References [dim](#), [v1](#), and [v2](#).

### 6.158.3 Member Data Documentation

#### 6.158.3.1 v1

[ex](#) [GiNaC::spmapkey::v1](#) [protected]

Referenced by [debugprint\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [spmapkey\(\)](#).

#### 6.158.3.2 v2

[ex](#) [GiNaC::spmapkey::v2](#) [protected]

Referenced by [debugprint\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [spmapkey\(\)](#).



## 6.158.3.3 dim

```
ex GiNaC::spmapkey::dim [protected]
```

Referenced by `debugprint()`, `operator<()`, and `operator==()`.

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)

## 6.159 GiNaC::status\_flags Class Reference

Flags to store information about the state of an object.

```
#include <flags.h>
```

## Public Types

- enum {  
[dynallocated](#) = 0x0001, [evaluated](#) = 0x0002, [expanded](#) = 0x0004, [hash\\_calculated](#) = 0x0008,  
[not\\_shareable](#) = 0x0010, [has\\_indices](#) = 0x0020, [has\\_no\\_indices](#) = 0x0040, [is\\_positive](#) = 0x0080,  
[is\\_negative](#) = 0x0100, [purely\\_indefinite](#) = 0x0200 }

## 6.159.1 Detailed Description

Flags to store information about the state of an object.

See also

[basic::flags](#)

## 6.159.2 Member Enumeration Documentation

## 6.159.2.1 anonymous enum

```
anonymous enum
```

## Enumerator

<code>dynallocated</code>	heap-allocated (i.e. created by <code>new</code> if we want to be clever and bypass the stack,  See also  <a href="#">ex::construct_from_basic()</a> )
<code>evaluated</code>	<a href="#">.eval()</a> has already done its job
<code>expanded</code>	<a href="#">.expand(0)</a> has already done its job (other <a href="#">expand()</a> options ignore this flag)
<code>hash_calculated</code>	<a href="#">.calchash()</a> has already done its job
<code>not_shareable</code>	don't share instances of this object between different expressions unless explicitly asked to (used by <a href="#">ex::compare()</a> )

The documentation for this class was generated from the following file:

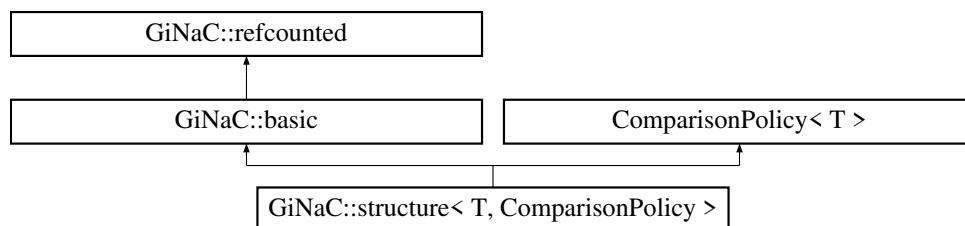
- [flags.h](#)

## 6.160 GiNaC::structure< T, ComparisonPolicy > Class Template Reference

Wrapper template for making [GiNaC](#) classes out of C++ structures.

```
#include <structure.h>
```

Inheritance diagram for GiNaC::structure< T, ComparisonPolicy >:



### Public Member Functions

- [structure](#) (const T &t)  
*Construct structure as a copy of a given C++ structure.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalm](#) () const override  
*Evaluate sums, products and integer powers of matrices.*
- [ex eval\\_indexed](#) (const [basic](#) &i) const override  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- void [print](#) (const [print\\_context](#) &c, unsigned level=0) const override  
*Output to stream.*
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- size\_t [nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex operator\[\]](#) (const [ex](#) &index) const override
- [ex operator\[\]](#) (size\_t i) const override
- [ex & let\\_op](#) (size\_t i) override  
*Return modifiable operand/member at position i.*
- [ex & operator\[\]](#) (const [ex](#) &index) override
- [ex & operator\[\]](#) (size\_t i) override
- bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const override  
*Test for occurrence of a pattern.*
- bool [match](#) (const [ex](#) &pattern, [exmap](#) &repl\_list) const override

- Check whether the expression matches a given pattern.*
- `ex subs` (const `exmap` &m, unsigned `options`=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex map` (`map_function` &f) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `int degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*
- `int ldegree` (const `ex` &s) const override  
*Return degree of lowest power in object s.*
- `ex coeff` (const `ex` &s, int `n`=1) const override  
*Return coefficient of degree n in object s.*
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- `ex collect` (const `ex` &s, bool `distributed`=false) const override  
*Sort expanded expression in terms of powers of some object(s).*
- `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const override  
*Default implementation of `ex::series()`.*
- `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const override  
*Default implementation of `ex::normal()`.*
- `ex to_rational` (`exmap` &repl) const override  
*Default implementation of `ex::to_rational()`.*
- `ex to_polynomial` (`exmap` &repl) const override
- `numeric integer_content` () const override
- `ex smod` (const `numeric` &xi) const override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient` () const override  
*Implementation `ex::max_coefficient()`.*
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- `ex add_indexed` (const `ex` &self, const `ex` &other) const override  
*Add two indexed expressions.*
- `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const override  
*Multiply an indexed expression with a scalar.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Try to contract two indexed expressions that appear in the same product.*
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- const T \* `operator->` () const
- T & `get_struct` ()
- const T & `get_struct` () const

## Protected Member Functions

- `ex eval_ncmul` (const `exvector` &v) const override
- `bool match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `ex derivative` (const `symbol` &s) const override  
*Default implementation of `ex::diff()`.*
- `bool is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*

## Static Private Member Functions

- static const char \* [get\\_class\\_name](#) ()

## Private Attributes

- T [obj](#)

## Additional Inherited Members

### 6.160.1 Detailed Description

```
template<class T, template< class > class ComparisonPolicy>
class GiNaC::structure< T, ComparisonPolicy >
```

Wrapper template for making [GiNaC](#) classes out of C++ structures.

### 6.160.2 Constructor & Destructor Documentation

#### 6.160.2.1 structure()

```
template<class T , template< class > class ComparisonPolicy>
GiNaC::structure< T, CP >::structure (
    const T & t ) [inline]
```

Construct structure as a copy of a given C++ structure.

Default constructor.

### 6.160.3 Member Function Documentation

#### 6.160.3.1 get\_class\_name()

```
template<class T , template< class > class ComparisonPolicy>
static const char* GiNaC::structure< T, ComparisonPolicy >::get_class_name ( ) [inline],
[static], [private]
```

**6.160.3.2 eval()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval ( ) const [inline], [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

**6.160.3.3 evalm()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::evalm ( ) const [inline], [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::evalm\(\)](#).

**6.160.3.4 eval\_ncmul()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval_ncmul (
    const exvector & v ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::hold\\_ncmul\(\)](#).

**6.160.3.5 eval\_indexed()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval_indexed (
    const basic & i ) const [inline], [override], [virtual]
```

Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

**6.160.3.6 print()**

```
template<class T , template< class > class ComparisonPolicy>
void GiNaC::structure< T, ComparisonPolicy >::print (
    const print_context & c,
    unsigned level = 0 ) const [inline], [override], [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

## Parameters

<i>c</i>	print context object that describes the output formatting
<i>level</i>	value that is used to identify the precedence or indentation level for placing parentheses and formatting

Reimplemented from [GiNaC::basic](#).

References *c*.

## 6.160.3.7 precedence()

```
template<class T , template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::precedence ( ) const  [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

## 6.160.3.8 info()

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::info (
    unsigned inf ) const  [inline], [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

## 6.160.3.9 nops()

```
template<class T , template< class > class ComparisonPolicy>
size_t GiNaC::structure< T, ComparisonPolicy >::nops ( ) const  [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

**6.160.3.10 op()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::op (
    size_t i ) const [inline], [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::op\(\)](#).

**6.160.3.11 operator[]()** [1/4]

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::operator[] (
    const ex & index ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.12 operator[]()** [2/4]

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::operator[] (
    size_t i ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.13 let\_op()**

```
template<class T , template< class > class ComparisonPolicy>
ex& GiNaC::structure< T, ComparisonPolicy >::let_op (
    size_t i ) [inline], [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

**6.160.3.14 operator[]()** [3/4]

```
template<class T , template< class > class ComparisonPolicy>
ex& GiNaC::structure< T, ComparisonPolicy >::operator[] (
    const ex & index ) [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.15** `operator[]()` [4/4]

```
template<class T , template< class > class ComparisonPolicy>
ex& GiNaC::structure< T, ComparisonPolicy >::operator[] (
    size_t i ) [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.16** `has()`

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::has (
    const ex & pattern,
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::has\(\)](#), and [options](#).

**6.160.3.17** `match()`

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::match (
    const ex & pattern,
    exmap & repl_lst ) const [inline], [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to `repl_lst`.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::match\(\)](#).



## 6.160.3.18 match\_same\_type()

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::match_same_type (
    const basic & other ) const [inline], [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

## 6.160.3.19 subs()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::subs (
    const exmap & m,
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References m, options, and [GiNaC::subs\(\)](#).

## 6.160.3.20 map()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::map (
    map_function & f ) const [inline], [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

**6.160.3.21 degree()**

```
template<class T , template< class > class ComparisonPolicy>
int GiNaC::structure< T, ComparisonPolicy >::degree (
    const ex & s ) const [inline], [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::degree\(\)](#).

**6.160.3.22 ldegree()**

```
template<class T , template< class > class ComparisonPolicy>
int GiNaC::structure< T, ComparisonPolicy >::ldegree (
    const ex & s ) const [inline], [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ldegree\(\)](#).

**6.160.3.23 coeff()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::coeff (
    const ex & s,
    int n = 1 ) const [inline], [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::coeff\(\)](#), and n.

**6.160.3.24 expand()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::expand (
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expand\(\)](#), and options.

**6.160.3.25 collect()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::collect (
    const ex & s,
    bool distributed = false ) const [inline], [override], [virtual]
```

Sort expanded expression in terms of powers of some object(s).

## Parameters

<i>s</i>	object(s) to sort in
<i>distributed</i>	recursive or distributed form (only used when s is a list)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::collect\(\)](#).

## 6.160.3.26 derivative()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::derivative (
    const symbol & s ) const [inline], [override], [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

## 6.160.3.27 series()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [options](#), [order](#), [r](#), and [GiNaC::series\(\)](#).

### 6.160.3.28 normal()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [inline], [override], [virtual]
```

Default implementation of [ex::normal\(\)](#).

It normalizes the children and replaces the object with a temporary symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::normal\(\)](#).

### 6.160.3.29 to\_rational()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::to_rational (
    exmap & repl ) const [inline], [override], [virtual]
```

Default implementation of [ex::to\\_rational\(\)](#).

This replaces the object with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::to\\_rational\(\)](#).

### 6.160.3.30 to\_polynomial()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::to_polynomial (
    exmap & repl ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::to\\_polynomial\(\)](#).

## 6.160.3.31 integer\_content()

```
template<class T , template< class > class ComparisonPolicy>
numeric GiNaC::structure< T, ComparisonPolicy >::integer_content ( ) const [inline], [override],
[virtual]
```

Reimplemented from [GiNaC::basic](#).

## 6.160.3.32 smod()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::smod (
    const numeric & xi ) const [inline], [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

## Parameters

$xi$	modulus
------	---------

## Returns

mapped polynomial

## See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

## 6.160.3.33 max\_coefficient()

```
template<class T , template< class > class ComparisonPolicy>
numeric GiNaC::structure< T, ComparisonPolicy >::max_coefficient ( ) const [inline], [override],
[virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

## See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

#### 6.160.3.34 `get_free_indices()`

```
template<class T , template< class > class ComparisonPolicy>
exvector GiNaC::structure< T, ComparisonPolicy >::get_free_indices ( ) const [inline], [override],
[virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

#### 6.160.3.35 `add_indexed()`

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::add_indexed (
    const ex & self,
    const ex & other ) const [inline], [override], [virtual]
```

Add two indexed expressions.

They are guaranteed to be of class indexed (or a subclass) and their indices are compatible. This function is used internally by [simplify\\_indexed\(\)](#).

##### Parameters

<i>self</i>	First indexed expression; its base object is *this
<i>other</i>	Second indexed expression

##### Returns

sum of self and other

##### See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

#### 6.160.3.36 `scalar_mul_indexed()`

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::scalar_mul_indexed (
    const ex & self,
    const numeric & other ) const [inline], [override], [virtual]
```

Multiply an indexed expression with a scalar.

This function is used internally by [simplify\\_indexed\(\)](#).

## Parameters

<i>self</i>	Indexed expression; its base object is *this
<i>other</i>	Numeric value

## Returns

product of self and other

## See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

## 6.160.3.37 contract\_with()

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [inline], [override], [virtual]
```

Try to contract two indexed expressions that appear in the same product.

If a contraction exists, the function overwrites one or both of the expressions and returns true. Otherwise it returns false. It is guaranteed that both expressions are of class indexed (or a subclass) and that at least one dummy index has been found. This functions is used internally by [simplify\\_indexed\(\)](#).

## Parameters

<i>self</i>	Pointer to first indexed expression; its base object is *this
<i>other</i>	Pointer to second indexed expression
<i>v</i>	The complete vector of factors

## Returns

true if the contraction was successful, false otherwise

## See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.38** `return_type()`

```
template<class T , template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::return_type ( ) const [inline], [override],
[virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

**6.160.3.39** `return_type_tinfo()`

```
template<class T , template< class > class ComparisonPolicy>
return\_type\_t GiNaC::structure< T, ComparisonPolicy >::return_type_tinfo ( ) const [inline],
[override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [r](#).

**6.160.3.40** `is_equal_same_type()`

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::is_equal_same_type (
    const basic & other ) const [inline], [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [GiNaC::structure< T, ComparisonPolicy >::obj](#).

**6.160.3.41** `calchash()`

```
template<class T , template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::calchash ( ) const [inline], [override],
[protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class [basic](#) computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).



#### 6.160.3.42 operator->()

```
template<class T , template< class > class ComparisonPolicy>
const T& GiNaC::structure< T, ComparisonPolicy >::operator-> ( ) const [inline]
```

References GiNaC::structure< T, ComparisonPolicy >::obj.

#### 6.160.3.43 get\_struct() [1/2]

```
template<class T , template< class > class ComparisonPolicy>
T& GiNaC::structure< T, ComparisonPolicy >::get_struct ( ) [inline]
```

References GiNaC::structure< T, ComparisonPolicy >::obj.

#### 6.160.3.44 get\_struct() [2/2]

```
template<class T , template< class > class ComparisonPolicy>
const T& GiNaC::structure< T, ComparisonPolicy >::get_struct ( ) const [inline]
```

References GiNaC::structure< T, ComparisonPolicy >::obj.

### 6.160.4 Member Data Documentation

#### 6.160.4.1 obj

```
template<class T , template< class > class ComparisonPolicy>
T GiNaC::structure< T, ComparisonPolicy >::obj [private]
```

Referenced by GiNaC::structure< T, ComparisonPolicy >::get\_struct(), GiNaC::structure< T, ComparisonPolicy >::is\_equal\_same\_type(), and GiNaC::structure< T, ComparisonPolicy >::operator->().

The documentation for this class was generated from the following file:

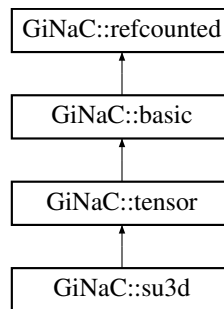
- [structure.h](#)

## 6.161 GiNaC::su3d Class Reference

This class represents the tensor of symmetric su(3) structure constants.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3d:



### Public Member Functions

- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of indexed symmetric structure constant.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed symmetric structure constant with something else.*

### Protected Member Functions

- unsigned `return_type` () const override
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Additional Inherited Members

#### 6.161.1 Detailed Description

This class represents the tensor of symmetric su(3) structure constants.

#### 6.161.2 Member Function Documentation

## 6.161.2.1 eval\_indexed()

```
ex GiNaC::su3d::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of indexed symmetric structure constant.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex1\\_3](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex\\_1\\_2](#), [GiNaC::\\_ex\\_1\\_3](#), [GiNaC::\\_ex\\_6](#), [CMPINDICES](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::sqrt\(\)](#), [std::swap\(\)](#), and [GiNaC::to\\_int\(\)](#).

## 6.161.2.2 contract\_with()

```
bool GiNaC::su3d::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed symmetric structure constant with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::color\\_T\(\)](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::get\\_representation\\_label\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::op\(\)](#), and [GiNaC::permute\\_free\\_index\\_to\\_front\(\)](#).

## 6.161.2.3 return\_type()

```
unsigned GiNaC::su3d::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

## 6.161.2.4 do\_print()

```
void GiNaC::su3d::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

### 6.161.2.5 do\_print\_latex()

```
void GiNaC::su3d::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

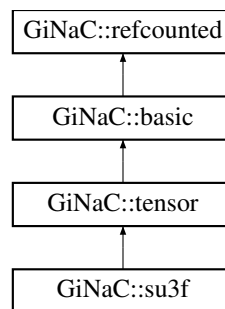
- [color.h](#)
- [color.cpp](#)

## 6.162 GiNaC::su3f Class Reference

This class represents the tensor of antisymmetric su(3) structure constants.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3f:



### Public Member Functions

- [ex eval\\_indexed](#) (const [basic](#) &i) const override  
*Automatic symbolic evaluation of indexed antisymmetric structure constant.*
- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of an indexed antisymmetric structure constant with something else.*

### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

### Additional Inherited Members

#### 6.162.1 Detailed Description

This class represents the tensor of antisymmetric su(3) structure constants.

## 6.162.2 Member Function Documentation

### 6.162.2.1 eval\_indexed()

```
ex GiNaC::su3f::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of indexed antisymmetric structure constant.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex\\_1\\_2](#), [CMPINDICES](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::sqrt\(\)](#), [std::swap\(\)](#), and [GiNaC::to\\_int\(\)](#).

### 6.162.2.2 contract\_with()

```
bool GiNaC::su3f::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed antisymmetric structure constant with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::color\\_T\(\)](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::get\\_representation\\_label\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::l](#), [GiNaC::ex::op\(\)](#), and [GiNaC::permute\\_free\\_index\\_to\\_front\(\)](#).

### 6.162.2.3 return\_type()

```
unsigned GiNaC::su3f::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 6.162.2.4 do\_print()

```
void GiNaC::su3f::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

#### 6.162.2.5 do\_print\_latex()

```
void GiNaC::su3f::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

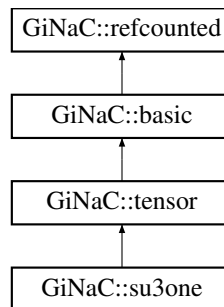
- [color.h](#)
- [color.cpp](#)

### 6.163 GiNaC::su3one Class Reference

This class represents the su(3) unity element.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3one:



#### Protected Member Functions

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

#### Additional Inherited Members

#### 6.163.1 Detailed Description

This class represents the su(3) unity element.

#### 6.163.2 Member Function Documentation

## 6.163.2.1 do\_print()

```
void GiNaC::su3one::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

## 6.163.2.2 do\_print\_latex()

```
void GiNaC::su3one::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following file:

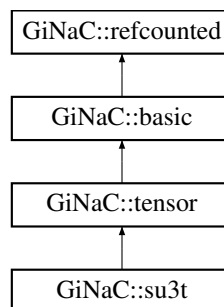
- [color.h](#)

## 6.164 GiNaC::su3t Class Reference

This class represents an su(3) generator.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3t:



## Public Member Functions

- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of generator with something else.*

## Protected Member Functions

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

## Additional Inherited Members

### 6.164.1 Detailed Description

This class represents an  $su(3)$  generator.

### 6.164.2 Member Function Documentation

#### 6.164.2.1 `contract_with()`

```
bool GiNaC::su3t::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of generator with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::color\\_ONE\(\)](#), [GiNaC::color\\_trace\(\)](#), and [GINAC\\_ASSERT](#).

#### 6.164.2.2 `do_print()`

```
void GiNaC::su3t::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

#### 6.164.2.3 `do_print_latex()`

```
void GiNaC::su3t::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

- [color.h](#)
- [color.cpp](#)

## 6.165 GiNaC::subs\_options Class Reference

Flags to control the behavior of [subs\(\)](#).

```
#include <flags.h>
```



## Public Types

- enum {  
[no\\_pattern](#) = 0x0001, [subs\\_no\\_pattern](#) = 0x0001, [algebraic](#) = 0x0002, [subs\\_algebraic](#) = 0x0002,  
[pattern\\_is\\_product](#) = 0x0004, [pattern\\_is\\_not\\_product](#) = 0x0008, [no\\_index\\_renaming](#) = 0x0010,  
[really\\_subs\\_idx](#) = 0x0020 }

### 6.165.1 Detailed Description

Flags to control the behavior of [subs\(\)](#).

### 6.165.2 Member Enumeration Documentation

#### 6.165.2.1 anonymous enum

anonymous enum

#### Enumerator

<a href="#">no_pattern</a>	disable pattern matching
<a href="#">subs_no_pattern</a>	
<a href="#">algebraic</a>	enable algebraic substitutions
<a href="#">subs_algebraic</a>	
<a href="#">pattern_is_product</a>	used internally by <a href="#">expairseq::subschildren()</a>
<a href="#">pattern_is_not_product</a>	used internally by <a href="#">expairseq::subschildren()</a>
<a href="#">no_index_renaming</a>	
<a href="#">really_subs_idx</a>	

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.166 GiNaC::sy\_is\_less Class Reference

### Public Member Functions

- [sy\\_is\\_less](#) (exvector::iterator v\_)
- bool [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

### Private Attributes

- exvector::iterator [v](#)

## 6.166.1 Constructor & Destructor Documentation

### 6.166.1.1 `sy_is_less()`

```
GiNaC::sy_is_less::sy_is_less (
    exvector::iterator v_ ) [inline]
```

## 6.166.2 Member Function Documentation

### 6.166.2.1 `operator()()`

```
bool GiNaC::sy_is_less::operator() (
    const ex & lh,
    const ex & rh ) const [inline]
```

References `GINAC_ASSERT`, and `v`.

## 6.166.3 Member Data Documentation

### 6.166.3.1 `v`

```
exvector::iterator GiNaC::sy_is_less::v [private]
```

Referenced by `operator()()`.

The documentation for this class was generated from the following file:

- [symmetry.cpp](#)

## 6.167 GiNaC::sy\_swap Class Reference

### Public Member Functions

- [sy\\_swap](#) (exvector::iterator v\_, bool &s)
- void [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh)

## Public Attributes

- bool & [swapped](#)

## Private Attributes

- [exvector::iterator](#) [v](#)

## 6.167.1 Constructor & Destructor Documentation

### 6.167.1.1 sy\_swap()

```
GiNaC::sy_swap::sy_swap (
    exvector::iterator v_,
    bool & s ) [inline]
```

## 6.167.2 Member Function Documentation

### 6.167.2.1 operator>()()

```
void GiNaC::sy_swap::operator() (
    const ex & lh,
    const ex & rh ) [inline]
```

References [GINAC\\_ASSERT](#), [swapped](#), and [v](#).

## 6.167.3 Member Data Documentation

### 6.167.3.1 v

```
exvector::iterator GiNaC::sy_swap::v [private]
```

Referenced by [operator>\(\)\(\)](#).

### 6.167.3.2 swapped

```
bool& GiNaC::sy_swap::swapped
```

Referenced by operator()().

The documentation for this class was generated from the following file:

- [symmetry.cpp](#)

## 6.168 GiNaC::sym\_desc Struct Reference

This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".

### Public Member Functions

- [sym\\_desc](#) (const [ex](#) &s)  
*Initialize symbol, leave other variables uninitialized.*
- bool [operator<](#) (const [sym\\_desc](#) &x) const  
*Comparison operator for sorting.*

### Public Attributes

- [ex](#) [sym](#)  
*Reference to symbol.*
- int [deg\\_a](#)  
*Highest degree of symbol in polynomial "a".*
- int [deg\\_b](#)  
*Highest degree of symbol in polynomial "b".*
- int [ldeg\\_a](#)  
*Lowest degree of symbol in polynomial "a".*
- int [ldeg\\_b](#)  
*Lowest degree of symbol in polynomial "b".*
- int [max\\_deg](#)  
*Maximum of deg\_a and deg\_b (Used for sorting)*
- size\_t [max\\_lcnops](#)  
*Maximum number of terms of leading coefficient of symbol in both polynomials.*

### 6.168.1 Detailed Description

This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".

A vector of these structures with information about all symbols in two polynomials can be created with the function [get\\_symbol\\_stats\(\)](#).

See also

[get\\_symbol\\_stats](#)

## 6.168.2 Constructor & Destructor Documentation

### 6.168.2.1 sym\_desc()

```
GiNaC::sym_desc::sym_desc (
    const ex & s ) [inline]
```

Initialize symbol, leave other variables uninitialized.

## 6.168.3 Member Function Documentation

### 6.168.3.1 operator<()

```
bool GiNaC::sym_desc::operator< (
    const sym\_desc & x ) const [inline]
```

Comparison operator for sorting.

References `max_deg`, `max_lcnops`, and `x`.

## 6.168.4 Member Data Documentation

### 6.168.4.1 sym

```
ex GiNaC::sym_desc::sym
```

Reference to symbol.

### 6.168.4.2 deg\_a

```
int GiNaC::sym_desc::deg_a
```

Highest degree of symbol in polynomial "a".

#### 6.168.4.3 deg\_b

```
int GiNaC::sym_desc::deg_b
```

Highest degree of symbol in polynomial "b".

#### 6.168.4.4 ldeg\_a

```
int GiNaC::sym_desc::ldeg_a
```

Lowest degree of symbol in polynomial "a".

#### 6.168.4.5 ldeg\_b

```
int GiNaC::sym_desc::ldeg_b
```

Lowest degree of symbol in polynomial "b".

#### 6.168.4.6 max\_deg

```
int GiNaC::sym_desc::max_deg
```

Maximum of deg\_a and deg\_b (Used for sorting)

Referenced by operator<().

#### 6.168.4.7 max\_lcnops

```
size_t GiNaC::sym_desc::max_lcnops
```

Maximum number of terms of leading coefficient of symbol in both polynomials.

Referenced by operator<().

The documentation for this struct was generated from the following file:

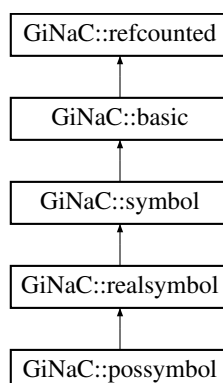
- [normal.cpp](#)

## 6.169 GiNaC::symbol Class Reference

Basic CAS symbol.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::symbol:



### Public Member Functions

- [symbol](#) (const std::string &initname)
- [symbol](#) (const std::string &initname, const std::string &texname)
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex series](#) (const [relational](#) &s, int [order](#), unsigned [options](#)=0) const override  
*Implementation of [ex::series\(\)](#) for symbols.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const override  
*Implementation of [ex::normal\(\)](#) for symbols.*
- [ex to\\_rational](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_rational\(\)](#) for symbols.*
- [ex to\\_polynomial](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_polynomial\(\)](#) for symbols.*
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_polynomial](#) (const [ex](#) &var) const override  
*Check whether this is a polynomial in the given variables.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*
- virtual unsigned [get\\_domain](#) () const
- void [set\\_name](#) (const std::string &n)
- void [set\\_TeX\\_name](#) (const std::string &n)
- std::string [get\\_name](#) () const
- std::string [get\\_TeX\\_name](#) () const

## Protected Member Functions

- `ex_derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for single differentiation of a symbol.*
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const

## Protected Attributes

- unsigned `serial`  
*unique serial number for comparison*
- std::string `name`  
*printname of this symbol*
- std::string `TeX_name`  
*LaTeX name of this symbol.*

## Static Private Attributes

- static unsigned `next_serial` = 0

### 6.169.1 Detailed Description

Basic CAS symbol.

It has a name because it must know how to output itself.

### 6.169.2 Constructor & Destructor Documentation

#### 6.169.2.1 `symbol()` [1/2]

```
GiNaC::symbol::symbol (
    const std::string & initname ) [explicit]
```

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, and `GiNaC::basic::setflag()`.



6.169.2.2 `symbol()` [2/2]

```
GiNaC::symbol::symbol (
    const std::string & initname,
    const std::string & texname )
```

References `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, and `GiNaC::basic::setflag()`.

## 6.169.3 Member Function Documentation

6.169.3.1 `info()`

```
bool GiNaC::symbol::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References `GiNaC::info_flags::cinteger_polynomial`, `GiNaC::info_flags::crational_polynomial`, `GiNaC::info_flags::expanded`, `get_domain()`, `GiNaC::info_flags::has_indices`, `GiNaC::info_flags::integer_polynomial`, `GiNaC::info_flags::nonnegative`, `GiNaC::info_flags::polynomial`, `GiNaC::domain::positive`, `GiNaC::info_flags::positive`, `GiNaC::info_flags::rational_function`, `GiNaC::info_flags::rational_polynomial`, `GiNaC::domain::real`, `GiNaC::info_flags::real`, and `GiNaC::info_flags::symbol`.

Referenced by `GiNaC::conjugate_expl_derivative()`, `GiNaC::imag_part_expl_derivative()`, and `GiNaC::real_part_expl_derivative()`.

6.169.3.2 `eval()`

```
ex GiNaC::symbol::eval ( ) const [inline], [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.3 evalf()

```
ex GiNaC::symbol::evalf ( ) const [inline], [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.4 series()

```
ex GiNaC::symbol::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for symbols.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GINAC\\_ASSERT](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [order](#), and [r](#).

### 6.169.3.5 subs()

```
ex GiNaC::symbol::subs (
    const exmap & m,
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [m](#), [options](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

### 6.169.3.6 normal()

```
ex GiNaC::symbol::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for symbols.

This returns the unmodified symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#).

### 6.169.3.7 to\_rational()

```
ex GiNaC::symbol::to_rational (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for symbols.

This returns the unmodified symbol.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.8 to\_polynomial()

```
ex GiNaC::symbol::to_polynomial (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for symbols.

This returns the unmodified symbol.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.9 conjugate()

```
ex GiNaC::symbol::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::realsymbol](#).

#### 6.169.3.10 real\_part()

```
ex GiNaC::symbol::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::realsymbol](#).

#### 6.169.3.11 imag\_part()

```
ex GiNaC::symbol::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::realsymbol](#).

#### 6.169.3.12 is\_polynomial()

```
bool GiNaC::symbol::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

#### 6.169.3.13 archive()

```
void GiNaC::symbol::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

Archive the object.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References n, name, and TeX\_name.

#### 6.169.3.14 read\_archive()

```
void GiNaC::symbol::read_archive (
    const archive_node & n,
    lst & sym_lst ) [override], [virtual]
```

Read (a.k.a.

Read object from [archive\\_node](#).

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::dynallocated](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [n](#), [name](#), [next\\_serial](#), [serial](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

#### 6.169.3.15 derivative()

```
ex GiNaC::symbol::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for single differentiation of a symbol.

It returns 1 or 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), and [GiNaC::basic::compare\\_same\\_type\(\)](#).

#### 6.169.3.16 is\_equal\_same\_type()

```
bool GiNaC::symbol::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [serial](#).

Referenced by [series\(\)](#).

#### 6.169.3.17 calchash()

```
unsigned GiNaC::symbol::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class `basic` computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::make\\_hash\\_seed\(\)](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

#### 6.169.3.18 get\_domain()

```
virtual unsigned GiNaC::symbol::get_domain ( ) const [inline], [virtual]
```

Reimplemented in [GiNaC::possymbol](#), and [GiNaC::realsymbol](#).

References [GiNaC::domain::complex](#).

Referenced by [do\\_print\\_tree\(\)](#), and [info\(\)](#).

#### 6.169.3.19 set\_name()

```
void GiNaC::symbol::set_name (
    const std::string & n ) [inline]
```

References [n](#), and [name](#).

#### 6.169.3.20 set\_TeX\_name()

```
void GiNaC::symbol::set_TeX_name (
    const std::string & n ) [inline]
```

References [n](#), and [TeX\\_name](#).

### 6.169.3.21 get\_name()

```
std::string GiNaC::symbol::get_name ( ) const
```

References name, and serial.

Referenced by do\_print(), and get\_TeX\_name().

### 6.169.3.22 get\_TeX\_name()

```
std::string GiNaC::symbol::get_TeX_name ( ) const
```

References GiNaC::get\_default\_TeX\_name(), get\_name(), and TeX\_name.

### 6.169.3.23 do\_print()

```
void GiNaC::symbol::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References c, and get\_name().

### 6.169.3.24 do\_print\_latex()

```
void GiNaC::symbol::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

References c, GiNaC::get\_default\_TeX\_name(), name, serial, and TeX\_name.

### 6.169.3.25 do\_print\_tree()

```
void GiNaC::symbol::do_print_tree (
    const print_tree & c,
    unsigned level ) const [protected]
```

References c, GiNaC::basic::flags, get\_domain(), GiNaC::basic::hashvalue, name, and serial.

#### 6.169.3.26 do\_print\_python\_repr()

```
void GiNaC::symbol::do_print_python_repr (
    const print_python_repr & c,
    unsigned level ) const [protected]
```

References `c`, `name`, `serial`, and `TeX_name`.

### 6.169.4 Member Data Documentation

#### 6.169.4.1 serial

```
unsigned GiNaC::symbol::serial [protected]
```

unique serial number for comparison

Referenced by `calchash()`, `do_print_latex()`, `do_print_python_repr()`, `do_print_tree()`, `get_name()`, `is_equal_same_type()`, and `read_archive()`.

#### 6.169.4.2 name

```
std::string GiNaC::symbol::name [mutable], [protected]
```

printname of this symbol

Referenced by `archive()`, `do_print_latex()`, `do_print_python_repr()`, `do_print_tree()`, `get_name()`, `read_archive()`, and `set_name()`.

#### 6.169.4.3 TeX\_name

```
std::string GiNaC::symbol::TeX_name [protected]
```

LaTeX name of this symbol.

Referenced by `archive()`, `do_print_latex()`, `do_print_python_repr()`, `get_TeX_name()`, `read_archive()`, and `set_TeX_name()`.



#### 6.169.4.4 next\_serial

```
unsigned GiNaC::symbol::next_serial = 0 [static], [private]
```

Referenced by read\_archive().

The documentation for this class was generated from the following files:

- [symbol.h](#)
- [normal.cpp](#)
- [pseries.cpp](#)
- [symbol.cpp](#)

## 6.170 GiNaC::symbolset Class Reference

### Public Member Functions

- [symbolset](#) (const [ex](#) &e)
- bool [has](#) (const [ex](#) &e) const

### Private Member Functions

- void [insert\\_symbols](#) (const [ex](#) &e)

### Private Attributes

- [exset](#) s

## 6.170.1 Constructor & Destructor Documentation

### 6.170.1.1 symbolset()

```
GiNaC::symbolset::symbolset (  
    const ex & e ) [inline], [explicit]
```

References [insert\\_symbols\(\)](#).

## 6.170.2 Member Function Documentation

### 6.170.2.1 insert\_symbols()

```
void GiNaC::symbolset::insert_symbols (
    const ex & e ) [inline], [private]
```

References s.

Referenced by symbolset().

### 6.170.2.2 has()

```
bool GiNaC::symbolset::has (
    const ex & e ) const [inline]
```

References s.

Referenced by GiNaC::lsolve().

## 6.170.3 Member Data Documentation

### 6.170.3.1 s

```
exset GiNaC::symbolset::s [private]
```

Referenced by has(), and insert\_symbols().

The documentation for this class was generated from the following file:

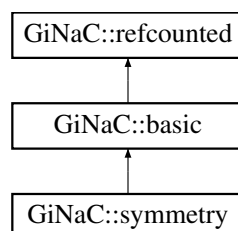
- [inifcns.cpp](#)

## 6.171 GiNaC::symmetry Class Reference

This class describes the symmetry of a group of indices.

```
#include <symmetry.h>
```

Inheritance diagram for GiNaC::symmetry:



## Public Types

- enum [symmetry\\_type](#) { [none](#), [symmetric](#), [antisymmetric](#), [cyclic](#) }  
*Type of symmetry.*

## Public Member Functions

- [symmetry](#) (unsigned i)  
*Create leaf node that represents one index.*
- [symmetry](#) ([symmetry\\_type](#) t, const [symmetry](#) &c1, const [symmetry](#) &c2)  
*Create node with two children.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &[syms](#)) override  
*Read (a.k.a.*
- [symmetry\\_type](#) [get\\_type](#) () const  
*Get symmetry type.*
- void [set\\_type](#) ([symmetry\\_type](#) t)  
*Set symmetry type.*
- [symmetry](#) & [add](#) (const [symmetry](#) &c)  
*Add child node, check index sets for consistency.*
- void [validate](#) (unsigned n)  
*Verify that all indices of this node are in the range [0..n-1].*
- bool [has\\_symmetry](#) () const  
*Check whether this node actually represents any kind of symmetry.*
- bool [has\\_nonsymmetric](#) () const  
*Check whether this node involves anything non symmetric.*
- bool [has\\_cyclic](#) () const  
*Check whether this node involves a cyclic symmetry.*

## Protected Member Functions

- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

## Private Attributes

- [symmetry\\_type](#) type  
*Type of symmetry described by this node.*
- std::set< unsigned > [indices](#)  
*Sorted union set of all indices handled by this node.*
- [exvector](#) children  
*Vector of child nodes.*

## Friends

- class [sy\\_is\\_less](#)
- class [sy\\_swap](#)
- int [canonicalize](#) (exvector::iterator v, const [symmetry](#) &symm)

*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*

## Additional Inherited Members

### 6.171.1 Detailed Description

This class describes the symmetry of a group of indices.

These objects can be grouped into a tree to form complex mixed symmetries.

### 6.171.2 Member Enumeration Documentation

#### 6.171.2.1 symmetry\_type

```
enum GiNaC::symmetry::symmetry\_type
```

Type of symmetry.

#### Enumerator

none	no symmetry properties
symmetric	totally symmetric
antisymmetric	totally antisymmetric
cyclic	cyclic symmetry

### 6.171.3 Constructor & Destructor Documentation

#### 6.171.3.1 symmetry() [1/2]

```
GiNaC::symmetry::symmetry (
    unsigned i )
```

Create leaf node that represents one index.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [indices](#), and [GiNaC::basic::setflag\(\)](#).

### 6.171.3.2 symmetry() [2/2]

```
GiNaC::symmetry::symmetry (
    symmetry_type t,
    const symmetry & c1,
    const symmetry & c2 )
```

Create node with two children.

References `add()`, `GiNaC::status_flags::evaluated`, `GiNaC::status_flags::expanded`, and `GiNaC::basic::setflag()`.

## 6.171.4 Member Function Documentation

### 6.171.4.1 archive()

```
void GiNaC::symmetry::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

Archive the object.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References `children`, `indices`, `n`, and `type`.

### 6.171.4.2 read\_archive()

```
void GiNaC::symmetry::read_archive (
    const archive_node & n,
    lst & sym_lst ) [override], [virtual]
```

Read (a.k.a.

Construct object from [archive\\_node](#).

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References `add()`, `indices`, `n`, and `type`.

#### 6.171.4.3 calchash()

```
unsigned GiNaC::symmetry::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References children, GiNaC::status\_flags::evaluated, GiNaC::basic::flags, GiNaC::status\_flags::hash\_calculated, GiNaC::basic::hashvalue, indices, GiNaC::make\_hash\_seed(), none, GiNaC::rotate\_left(), GiNaC::basic::setflag(), and type.

#### 6.171.4.4 get\_type()

```
symmetry_type GiNaC::symmetry::get_type ( ) const [inline]
```

Get symmetry type.

References type.

#### 6.171.4.5 set\_type()

```
void GiNaC::symmetry::set_type (
    symmetry_type t ) [inline]
```

Set symmetry type.

References type.

Referenced by GiNaC::sy\_anti(), GiNaC::sy\_cycl(), and GiNaC::sy\_symm().

#### 6.171.4.6 add()

```
symmetry & GiNaC::symmetry::add (
    const symmetry & c )
```

Add child node, check index sets for consistency.

References c, children, GINAC\_ASSERT, indices, none, and type.

Referenced by read\_archive(), GiNaC::sy\_anti(), GiNaC::sy\_cycl(), GiNaC::sy\_none(), GiNaC::sy\_symm(), symmetry(), and validate().

#### 6.171.4.7 validate()

```
void GiNaC::symmetry::validate (
    unsigned n )
```

Verify that all indices of this node are in the range [0..n-1].

This function throws an exception if the verification fails. If the top node has a type != none and no children, add all indices in the range [0..n-1] as children.

References add(), indices, n, none, and type.

Referenced by GiNaC::indexed::read\_archive(), and GiNaC::indexed::validate().

#### 6.171.4.8 has\_symmetry()

```
bool GiNaC::symmetry::has_symmetry ( ) const [inline]
```

Check whether this node actually represents any kind of symmetry.

References children, none, and type.

#### 6.171.4.9 has\_nonsymmetric()

```
bool GiNaC::symmetry::has_nonsymmetric ( ) const
```

Check whether this node involves anything non symmetric.

References antisymmetric, children, cyclic, and type.

#### 6.171.4.10 has\_cyclic()

```
bool GiNaC::symmetry::has_cyclic ( ) const
```

Check whether this node involves a cyclic symmetry.

References children, cyclic, and type.

#### 6.171.4.11 do\_print()

```
void GiNaC::symmetry::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References antisymmetric, c, children, cyclic, indices, none, symmetric, and type.

#### 6.171.4.12 do\_print\_tree()

```
void GiNaC::symmetry::do_print_tree (
    const print_tree & c,
    unsigned level ) const [protected]
```

References antisymmetric, c, children, cyclic, GiNaC::basic::flags, GiNaC::basic::hashvalue, indices, none, symmetric, and type.

### 6.171.5 Friends And Related Function Documentation

#### 6.171.5.1 sy\_is\_less

```
friend class sy_is_less [friend]
```

#### 6.171.5.2 sy\_swap

```
friend class sy_swap [friend]
```

#### 6.171.5.3 canonicalize

```
int canonicalize (
    exvector::iterator v,
    const symmetry & symm ) [friend]
```

Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.

##### Parameters

<i>v</i>	Start of expression vector
<i>symm</i>	Root node of symmetry tree

##### Returns

the overall sign introduced by the reordering (+1, -1 or 0) or numeric\_limits<int>::max() if nothing changed

### 6.171.6 Member Data Documentation



### 6.171.6.1 type

`symmetry_type` GiNaC::symmetry::type [private]

Type of symmetry described by this node.

Referenced by `add()`, `archive()`, `calchash()`, `do_print()`, `do_print_tree()`, `get_type()`, `has_cyclic()`, `has_nonsymmetric()`, `has_symmetry()`, `read_archive()`, `set_type()`, and `validate()`.

### 6.171.6.2 indices

`std::set<unsigned>` GiNaC::symmetry::indices [private]

Sorted union set of all indices handled by this node.

Referenced by `add()`, `archive()`, `calchash()`, `do_print()`, `do_print_tree()`, `read_archive()`, `symmetry()`, and `validate()`.

### 6.171.6.3 children

`exvector` GiNaC::symmetry::children [private]

Vector of child nodes.

Referenced by `add()`, `archive()`, `calchash()`, `do_print()`, `do_print_tree()`, `has_cyclic()`, `has_nonsymmetric()`, and `has_symmetry()`.

The documentation for this class was generated from the following files:

- [symmetry.h](#)
- [symmetry.cpp](#)

## 6.172 GiNaC::symminfo Class Reference

This structure stores the individual symmetrized terms obtained during the simplification of sums.

### Public Member Functions

- [symminfo](#) ()
- [symminfo](#) (const [ex](#) &symmterm\_, const [ex](#) &orig\_, size\_t num\_)

## Public Attributes

- [ex symmterm](#)  
*symmetrized term*
- [ex coeff](#)  
*coefficient of symmetrized term*
- [ex orig](#)  
*original term*
- `size_t num`  
*how many symmetrized terms resulted from the original term*

### 6.172.1 Detailed Description

This structure stores the individual symmetrized terms obtained during the simplification of sums.

### 6.172.2 Constructor & Destructor Documentation

#### 6.172.2.1 symminfo() [1/2]

```
GiNaC::symminfo::symminfo ( ) [inline]
```

#### 6.172.2.2 symminfo() [2/2]

```
GiNaC::symminfo::symminfo (
    const ex & symmterm_,
    const ex & orig_,
    size_t num_ ) [inline]
```

References `coeff`, `GiNaC::ex::nops()`, `GiNaC::ex::op()`, and `symmterm`.

### 6.172.3 Member Data Documentation

#### 6.172.3.1 symmterm

```
ex GiNaC::symminfo::symmterm
```

symmetrized term

Referenced by `GiNaC::symminfo_is_less_by_symmterm::operator()()`, and `symminfo()`.

6.172.3.2 `coeff`

`ex` `GiNaC::symminfo::coeff`

coefficient of symmetrized term

Referenced by `symminfo()`.

6.172.3.3 `orig`

`ex` `GiNaC::symminfo::orig`

original term

Referenced by `GiNaC::symminfo_is_less_by_orig::operator()()`.

6.172.3.4 `num`

`size_t` `GiNaC::symminfo::num`

how many symmetrized terms resulted from the original term

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.173 GiNaC::symminfo\_is\_less\_by\_orig Class Reference

## Public Member Functions

- `bool operator() (const symminfo &si1, const symminfo &si2) const`

## 6.173.1 Member Function Documentation

6.173.1.1 `operator()()`

```
bool GiNaC::symminfo_is_less_by_orig::operator() (
    const symminfo & si1,
    const symminfo & si2 ) const [inline]
```

References `GiNaC::ex::compare()`, and `GiNaC::symminfo::orig`.

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.174 GiNaC::symminfo\_is\_less\_by\_symmterm Class Reference

### Public Member Functions

- `bool operator()` (const `symminfo` &si1, const `symminfo` &si2) const

### 6.174.1 Member Function Documentation

#### 6.174.1.1 operator()

```
bool GiNaC::symminfo_is_less_by_symmterm::operator() (
    const symminfo & si1,
    const symminfo & si2 ) const [inline]
```

References `GiNaC::ex::compare()`, and `GiNaC::symminfo::symmterm`.

The documentation for this class was generated from the following file:

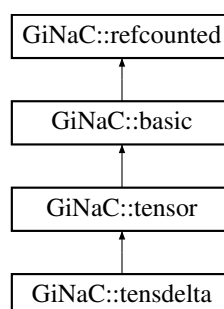
- `indexed.cpp`

## 6.175 GiNaC::tensdelta Class Reference

This class represents the delta tensor.

```
#include <tensor.h>
```

Inheritance diagram for `GiNaC::tensdelta`:



### Public Member Functions

- `bool info` (unsigned inf) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed delta tensor.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed delta tensor with something else.*

## Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

## Additional Inherited Members

### 6.175.1 Detailed Description

This class represents the delta tensor.

If indexed, it must have exactly two indices of the same type.

### 6.175.2 Member Function Documentation

#### 6.175.2.1 [info\(\)](#)

```
bool GiNaC::tensdelta::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

#### 6.175.2.2 [eval\\_indexed\(\)](#)

```
ex GiNaC::tensdelta::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed delta tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::idx::get\\_dim\(\)](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [m](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::to\\_int\(\)](#).

### 6.175.2.3 contract\_with()

```
bool GiNaC::tensdelta::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed delta tensor with something else.

Reimplemented from [GiNaC::basic](#).

References GINAC\_ASSERT, and GiNaC::tensor::replace\_contr\_index().

### 6.175.2.4 return\_type()

```
unsigned GiNaC::tensdelta::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::tensor](#).

References GiNaC::return\_types::commutative.

### 6.175.2.5 do\_print()

```
void GiNaC::tensdelta::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

### 6.175.2.6 do\_print\_latex()

```
void GiNaC::tensdelta::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

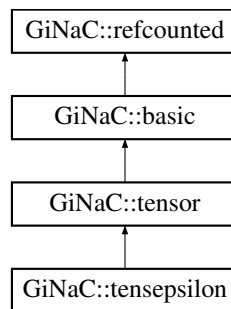
- [tensor.h](#)
- [tensor.cpp](#)

## 6.176 GiNaC::tensepsilon Class Reference

This class represents the totally antisymmetric epsilon tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensepsilon:



### Public Member Functions

- `tensepsilon` (bool `minkowski`, bool `pos_sig`)
- bool `info` (unsigned `inf`) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &`i`) const override  
*Automatic symbolic evaluation of an indexed epsilon tensor.*
- bool `contract_with` (exvector::iterator `self`, exvector::iterator `other`, `exvector` &`v`) const override  
*Contraction of epsilon tensor with something else.*
- void `archive` (`archive_node` &`n`) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override  
*Read (a.k.a.*

### Protected Member Functions

- unsigned `return_type` () const override
- void `do_print` (const `print_context` &`c`, unsigned `level`) const
- void `do_print_latex` (const `print_latex` &`c`, unsigned `level`) const

### Private Attributes

- bool `minkowski`  
*If true, tensor is in Minkowski-type space.*
- bool `pos_sig`  
*If true, the metric is assumed to be diag(-1,1,1...).*

## Additional Inherited Members

### 6.176.1 Detailed Description

This class represents the totally antisymmetric epsilon tensor.

If indexed, all indices must be of the same type and their number must be equal to the dimension of the index space.

### 6.176.2 Constructor & Destructor Documentation

#### 6.176.2.1 tensespsilon()

```
GiNaC::tensespsilon::tensespsilon (
    bool minkowski,
    bool pos_sig )
```

### 6.176.3 Member Function Documentation

#### 6.176.3.1 info()

```
bool GiNaC::tensespsilon::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

#### 6.176.3.2 eval\_indexed()

```
ex GiNaC::tensespsilon::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed epsilon tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_zero\(\)](#), [minkowski](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::permutation\\_sign\(\)](#), [pos\\_sig](#), [GiNaC::to\\_int\(\)](#), and [x](#).



### 6.176.3.3 contract\_with()

```
bool GiNaC::tensepsilon::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of epsilon tensor with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::delta\\_tensor\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::lorentz\\_g\(\)](#), [GiNaC::metric\\_tensor\(\)](#), [minkowski](#), and [pos\\_sig](#).

### 6.176.3.4 archive()

```
void GiNaC::tensepsilon::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [minkowski](#), [n](#), and [pos\\_sig](#).

### 6.176.3.5 read\_archive()

```
void GiNaC::tensepsilon::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [minkowski](#), [n](#), and [pos\\_sig](#).

### 6.176.3.6 return\_type()

```
unsigned GiNaC::tensepsilon::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::tensor](#).

References [GiNaC::return\\_types::commutative](#).

#### 6.176.3.7 do\_print()

```
void GiNaC::tensepsilon::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

#### 6.176.3.8 do\_print\_latex()

```
void GiNaC::tensepsilon::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

### 6.176.4 Member Data Documentation

#### 6.176.4.1 minkowski

```
bool GiNaC::tensepsilon::minkowski [private]
```

If true, tensor is in Minkowski-type space.

Otherwise it is in a Euclidean space.

Referenced by [archive\(\)](#), [contract\\_with\(\)](#), [eval\\_indexed\(\)](#), and [read\\_archive\(\)](#).

#### 6.176.4.2 pos\_sig

```
bool GiNaC::tensepsilon::pos_sig [private]
```

If true, the metric is assumed to be  $\text{diag}(-1, 1, 1, \dots)$ .

Otherwise it is  $\text{diag}(1, -1, -1, \dots)$ . This is only relevant if `minkowski = true`.

Referenced by [archive\(\)](#), [contract\\_with\(\)](#), [eval\\_indexed\(\)](#), and [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

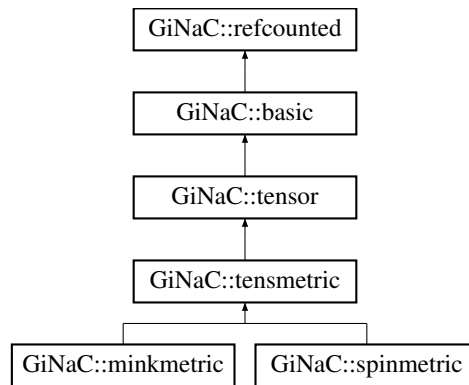
- [tensor.h](#)
- [tensor.cpp](#)

## 6.177 GiNaC::tensmetric Class Reference

This class represents a general metric tensor which can be used to raise/lower indices.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensmetric:



### Public Member Functions

- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- ex [eval\\_indexed](#) (const [basic](#) &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of an indexed metric tensor with something else.*

### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

### Additional Inherited Members

#### 6.177.1 Detailed Description

This class represents a general metric tensor which can be used to raise/lower indices.

If indexed, it must have exactly two indices of the same type which must be of class `varidx` or a subclass.

#### 6.177.2 Member Function Documentation

### 6.177.2.1 info()

```
bool GiNaC::tensmetric::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::spinmetric](#), and [GiNaC::minkmetric](#).

References [GiNaC::info\\_flags::real](#).

### 6.177.2.2 eval\_indexed()

```
ex GiNaC::tensmetric::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed metric tensor.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::spinmetric](#), and [GiNaC::minkmetric](#).

References [GiNaC::delta\\_tensor\(\)](#), [GiNaC::idx::get\\_dim\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::varidx::is\\_covariant\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [m](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::basic::subs\(\)](#).

### 6.177.2.3 contract\_with()

```
bool GiNaC::tensmetric::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed metric tensor with something else.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::spinmetric](#).

References [GINAC\\_ASSERT](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

6.177.2.4 `return_type()`

```
unsigned GiNaC::tensmetric::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::tensor](#).

Reimplemented in [GiNaC::minkmetric](#).

References [GiNaC::return\\_types::commutative](#).

6.177.2.5 `do_print()`

```
void GiNaC::tensmetric::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

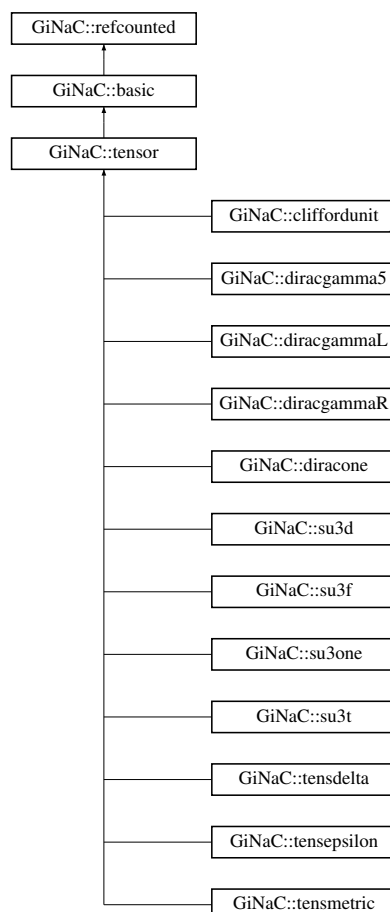
- [tensor.h](#)
- [tensor.cpp](#)

6.178 [GiNaC::tensor](#) Class Reference

This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.

```
#include <tensor.h>
```

Inheritance diagram for [GiNaC::tensor](#):



## Public Member Functions

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Protected Member Functions

- unsigned [return\\_type](#) () const override

## Additional Inherited Members

### 6.178.1 Detailed Description

This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.

They are represented without indices. To attach indices to them, wrap them in an object of class [indexed](#).

### 6.178.2 Member Function Documentation

#### 6.178.2.1 [return\\_type\(\)](#)

```
unsigned GiNaC::tensor::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::tensepsilon](#), [GiNaC::minkmetric](#), [GiNaC::tensmetric](#), and [GiNaC::tensdelta](#).

References [GiNaC::return\\_types::noncommutative\\_composite](#).

#### 6.178.2.2 [replace\\_contr\\_index\(\)](#)

```
bool GiNaC::tensor::replace_contr_index (
    exvector::iterator self,
    exvector::iterator other ) const
```

Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).

References [GiNaC::\\_ex1](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::idx::is\\_symbolic\(\)](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::tensdelta::contract\\_with\(\)](#), and [GiNaC::tensmetric::contract\\_with\(\)](#).

The documentation for this class was generated from the following files:

- [tensor.h](#)
- [tensor.cpp](#)

## 6.179 GiNaC::terminfo Class Reference

This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.

### Public Member Functions

- [terminfo](#) (const [ex](#) &orig\_, const [ex](#) &symm\_)

### Public Attributes

- [ex orig](#)  
*original term*
- [ex symm](#)  
*symmetrized term*

### 6.179.1 Detailed Description

This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.

### 6.179.2 Constructor & Destructor Documentation

#### 6.179.2.1 terminfo()

```
GiNaC::terminfo::terminfo (  
    const ex & orig_,  
    const ex & symm_ ) [inline]
```

### 6.179.3 Member Data Documentation

#### 6.179.3.1 orig

[ex](#) GiNaC::terminfo::orig

original term

### 6.179.3.2 `symm`

`ex` `GiNaC::terminfo::symm`

symmetrized term

Referenced by `GiNaC::terminfo_is_less::operator()`.

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.180 `GiNaC::terminfo_is_less` Class Reference

### Public Member Functions

- `bool operator()` (const `terminfo` &ti1, const `terminfo` &ti2) const

### 6.180.1 Member Function Documentation

#### 6.180.1.1 `operator()`

```
bool GiNaC::terminfo_is_less::operator() (
    const terminfo & ti1,
    const terminfo & ti2 ) const [inline]
```

References `GiNaC::ex::compare()`, and `GiNaC::terminfo::symm`.

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.181 `GiNaC::class_info< OPT >::tree_node` Struct Reference

### Public Member Functions

- `tree_node` (`class_info` \*i)
- void `add_child` (`tree_node` \*n)

### Public Attributes

- `std::vector< tree_node * >` `children`
- `class_info` \* `info`



## 6.181.1 Constructor & Destructor Documentation

### 6.181.1.1 tree\_node()

```
template<class OPT>
GiNaC::class_info< OPT >::tree_node::tree_node (
    class_info * i ) [inline]
```

## 6.181.2 Member Function Documentation

### 6.181.2.1 add\_child()

```
template<class OPT>
void GiNaC::class_info< OPT >::tree_node::add_child (
    tree_node * n ) [inline]
```

References `GiNaC::class_info< OPT >::tree_node::children`, and `n`.

## 6.181.3 Member Data Documentation

### 6.181.3.1 children

```
template<class OPT>
std::vector<tree_node *> GiNaC::class_info< OPT >::tree_node::children
```

Referenced by `GiNaC::class_info< OPT >::tree_node::add_child()`.

### 6.181.3.2 info

```
template<class OPT>
class_info* GiNaC::class_info< OPT >::tree_node::info
```

The documentation for this struct was generated from the following file:

- [class\\_info.h](#)

## 6.182 GiNaC::unarchive\_table\_t Class Reference

```
#include <archive.h>
```

### Public Member Functions

- [unarchive\\_table\\_t\(\)](#)
- [~unarchive\\_table\\_t\(\)](#)
- [synthesize\\_func find](#) (const std::string &classname) const
- void [insert](#) (const std::string &classname, [synthesize\\_func](#) f)

### Static Private Attributes

- static int [usecount](#) = 0
- static [unarchive\\_map\\_t](#) \* [unarch\\_map](#) = nullptr

## 6.182.1 Constructor & Destructor Documentation

### 6.182.1.1 unarchive\_table\_t()

```
GiNaC::unarchive_table_t::unarchive_table_t ( )
```

References [unarch\\_map](#), and [usecount](#).

### 6.182.1.2 ~unarchive\_table\_t()

```
GiNaC::unarchive_table_t::~~unarchive_table_t ( )
```

References [unarch\\_map](#), and [usecount](#).

## 6.182.2 Member Function Documentation

### 6.182.2.1 find()

```
synthesize\_func GiNaC::unarchive_table_t::find (
    const std::string & classname ) const
```

References [unarch\\_map](#).

Referenced by [GiNaC::find\\_factory\\_fcn\(\)](#).

## 6.182.2.2 insert()

```
void GiNaC::unarchive_table_t::insert (
    const std::string & classname,
    synthesize_func f )
```

References unarch\_map.

## 6.182.3 Member Data Documentation

## 6.182.3.1 usecount

```
int GiNaC::unarchive_table_t::usecount = 0 [static], [private]
```

Referenced by unarchive\_table\_t(), and ~unarchive\_table\_t().

## 6.182.3.2 unarch\_map

```
unarchive_map_t * GiNaC::unarchive_table_t::unarch_map = nullptr [static], [private]
```

Referenced by find(), insert(), unarchive\_table\_t(), and ~unarchive\_table\_t().

The documentation for this class was generated from the following files:

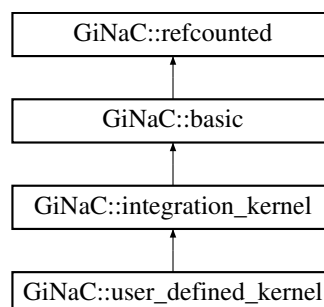
- [archive.h](#)
- [archive.cpp](#)

## 6.183 GiNaC::user\_defined\_kernel Class Reference

A user-defined integration kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::user\_defined\_kernel:



## Public Member Functions

- `user_defined_kernel` (const `ex` &`f`, const `ex` &`x`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position `i`.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex Laurent_series` (const `ex` &`x`, int `order`) const override  
*Returns the Laurent series, starting possibly with the pole term.*

## Protected Member Functions

- `bool uses_Laurent_series` () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `void do_print` (const `print_context` &`c`, unsigned `level`) const

## Protected Attributes

- `ex f`
- `ex x`

### 6.183.1 Detailed Description

A user-defined integration kernel.

The input is an expression  $f$ , depending on a variable  $x$ . It is assumed that  $f$  has a Laurent expansion around  $x = 0$  and maximally a simple pole at  $x = 0$ .

### 6.183.2 Constructor & Destructor Documentation

#### 6.183.2.1 `user_defined_kernel()`

```
GiNaC::user_defined_kernel::user_defined_kernel (
    const ex & f,
    const ex & x )
```

### 6.183.3 Member Function Documentation

### 6.183.3.1 nops()

```
size_t GiNaC::user_defined_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.183.3.2 op()

```
ex GiNaC::user_defined_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References f, and x.

### 6.183.3.3 let\_op()

```
ex & GiNaC::user_defined_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), f, and x.

### 6.183.3.4 is\_numeric()

```
bool GiNaC::user_defined_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), f, [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::subs\(\)](#), and x.

### 6.183.3.5 Laurent\_series()

```
ex GiNaC::user_defined_kernel::Laurent_series (
    const ex & x,
    int order ) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References `f`, `order`, `GiNaC::ex::series()`, `GiNaC::ex::subs()`, and `x`.

### 6.183.3.6 uses\_Laurent\_series()

```
bool GiNaC::user_defined_kernel::uses_Laurent_series ( ) const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 6.183.3.7 do\_print()

```
void GiNaC::user_defined_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References `c`, `f`, `GiNaC::ex::print()`, and `x`.

## 6.183.4 Member Data Documentation

### 6.183.4.1 f

```
ex GiNaC::user_defined_kernel::f [protected]
```

Referenced by `do_print()`, `is_numeric()`, `Laurent_series()`, `let_op()`, and `op()`.

## 6.183.4.2 x

```
ex GiNaC::user_defined_kernel::x [protected]
```

Referenced by `do_print()`, `is_numeric()`, `Laurent_series()`, `let_op()`, and `op()`.

The documentation for this class was generated from the following files:

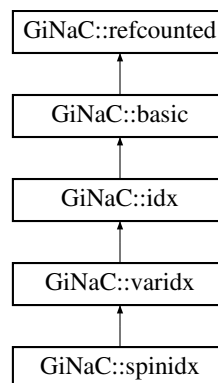
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.184 GiNaC::varidx Class Reference

This class holds an index with a variance (co- or contravariant).

```
#include <idx.h>
```

Inheritance diagram for GiNaC::varidx:



### Public Member Functions

- **varidx** (const [ex](#) &v, const [ex](#) &dim, bool [covariant](#)=false)  
*Construct index with given value, dimension and variance.*
- bool **is\_dummy\_pair\_same\_type** (const [basic](#) &other) const override  
*Check whether the index forms a dummy index pair with another index of the same type.*
- void **archive** ([archive\\_node](#) &n) const override  
*Save (serialize) the object into archive node.*
- void **read\_archive** (const [archive\\_node](#) &n, [lst](#) &[syms](#)) override  
*Load (deserialize) the object from an archive node.*
- bool **is\_covariant** () const  
*Check whether the index is covariant.*
- bool **is\_contravariant** () const  
*Check whether the index is contravariant (not covariant).*
- **ex toggle\_variance** () const  
*Make a new index with the same value but the opposite variance.*

## Protected Member Functions

- bool `match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const

## Protected Attributes

- bool `covariant`  
 *$x.mu$ , default is contravariant:  $x \sim mu$*

### 6.184.1 Detailed Description

This class holds an index with a variance (co- or contravariant).

There is an associated metric tensor that can be used to raise/lower indices.

### 6.184.2 Constructor & Destructor Documentation

#### 6.184.2.1 `varidx()`

```
GiNaC::varidx::varidx (
    const ex & v,
    const ex & dim,
    bool covariant = false )
```

Construct index with given value, dimension and variance.

#### Parameters

<code>v</code>	Value of index (numeric or symbolic)
<code>dim</code>	Dimension of index space (numeric or symbolic)
<code>covariant</code>	Make covariant index (default is contravariant)

#### Returns

newly constructed index

### 6.184.3 Member Function Documentation



### 6.184.3.1 is\_dummy\_pair\_same\_type()

```
bool GiNaC::varidx::is_dummy_pair_same_type (
    const basic & other ) const [override], [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented from [GiNaC::idx](#).

Reimplemented in [GiNaC::spinidx](#).

References covariant.

### 6.184.3.2 archive()

```
void GiNaC::varidx::archive (
    archive\_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::idx](#).

Reimplemented in [GiNaC::spinidx](#).

References covariant, and n.

### 6.184.3.3 read\_archive()

```
void GiNaC::varidx::read_archive (
    const archive\_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::idx](#).

Reimplemented in [GiNaC::spinidx](#).

References covariant, and n.

#### 6.184.3.4 `match_same_type()`

```
bool GiNaC::varidx::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::idx](#).

Reimplemented in [GiNaC::spinidx](#).

References covariant, and GINAC\_ASSERT.

#### 6.184.3.5 `is_covariant()`

```
bool GiNaC::varidx::is_covariant ( ) const [inline]
```

Check whether the index is covariant.

References covariant.

Referenced by [GiNaC::tensmetric::eval\\_indexed\(\)](#).

#### 6.184.3.6 `is_contravariant()`

```
bool GiNaC::varidx::is_contravariant ( ) const [inline]
```

Check whether the index is contravariant (not covariant).

References covariant.

#### 6.184.3.7 `toggle_variance()`

```
ex GiNaC::varidx::toggle_variance ( ) const
```

Make a new index with the same value but the opposite variance.

References [GiNaC::basic::clearflag\(\)](#), covariant, [GiNaC::basic::duplicate\(\)](#), and [GiNaC::status\\_flags::hash\\_↔](#) calculated.

## 6.184.3.8 do\_print()

```
void GiNaC::varidx::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References `c`, `covariant`, and `GiNaC::idx::print_index()`.

## 6.184.3.9 do\_print\_tree()

```
void GiNaC::varidx::do_print_tree (
    const print_tree & c,
    unsigned level ) const [protected]
```

References `c`, `covariant`, `GiNaC::idx::dim`, `GiNaC::basic::flags`, `GiNaC::basic::hashvalue`, `GiNaC::ex::print()`, and `GiNaC::idx::value`.

## 6.184.4 Member Data Documentation

## 6.184.4.1 covariant

```
bool GiNaC::varidx::covariant [protected]
```

`x.mu`, default is contravariant:  $x \sim \mu$

Referenced by `archive()`, `do_print()`, `GiNaC::spinidx::do_print()`, `do_print_tree()`, `GiNaC::spinidx::do_print_tree()`, `is_contravariant()`, `is_covariant()`, `is_dummy_pair_same_type()`, `match_same_type()`, `read_archive()`, `toggle_variance()`, and `GiNaC::spinidx::toggle_variance_dot()`.

The documentation for this class was generated from the following files:

- [idx.h](#)
- [idx.cpp](#)

## 6.185 GiNaC::visitor Class Reference

Degenerate base class for visitors.

```
#include <basic.h>
```

## Protected Member Functions

- virtual [~visitor\(\)](#)

### 6.185.1 Detailed Description

Degenerate base class for visitors.

basic and derivative classes support Robert C. Martin's Acyclic Visitor pattern (cf. <http://condor.depaul.edu/dmumaugh/OOT/Design-Principles/acv.pdf> or chapter 10 of Andrei Alexandrescu's "Modern C++ Design").

### 6.185.2 Constructor & Destructor Documentation

#### 6.185.2.1 ~visitor()

```
virtual GiNaC::visitor::~~visitor ( ) [inline], [protected], [virtual]
```

The documentation for this class was generated from the following file:

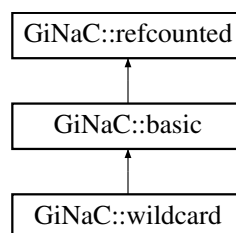
- [basic.h](#)

## 6.186 GiNaC::wildcard Class Reference

This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).

```
#include <wildcard.h>
```

Inheritance diagram for GiNaC::wildcard:



### Public Member Functions

- [wildcard](#) (unsigned [label](#))  
*Construct wildcard with specified label.*
- bool [match](#) (const [ex](#) &pattern, [exmap](#) &repl\_lst) const override  
*Check whether the expression matches a given pattern.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*
- unsigned [get\\_label](#) () const

## Protected Member Functions

- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

## Private Attributes

- unsigned [label](#)  
*Label used to distinguish different wildcards.*

## Additional Inherited Members

### 6.186.1 Detailed Description

This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).

An integer label is used to identify different wildcards.

### 6.186.2 Constructor & Destructor Documentation

#### 6.186.2.1 wildcard()

```
GiNaC::wildcard::wildcard (
    unsigned label )
```

Construct wildcard with specified label.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), and [GiNaC::basic::setflag\(\)](#).

### 6.186.3 Member Function Documentation

#### 6.186.3.1 match()

```
bool GiNaC::wildcard::match (
    const ex & pattern,
    exmap & repl_lst ) const [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to `repl_lst`.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::is\\_equal\(\)](#).

### 6.186.3.2 archive()

```
void GiNaC::wildcard::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References label, and n.

### 6.186.3.3 read\_archive()

```
void GiNaC::wildcard::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), label, n, and [GiNaC::basic::setflag\(\)](#).

### 6.186.3.4 calchash()

```
unsigned GiNaC::wildcard::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), label, [GiNaC::make\\_hash\\_seed\(\)](#), and [GiNaC::basic::setflag\(\)](#).

### 6.186.3.5 get\_label()

```
unsigned GiNaC::wildcard::get_label ( ) const [inline]
```

References label.

### 6.186.3.6 do\_print()

```
void GiNaC::wildcard::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [label](#).

### 6.186.3.7 do\_print\_tree()

```
void GiNaC::wildcard::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), and [label](#).

### 6.186.3.8 do\_print\_python\_repr()

```
void GiNaC::wildcard::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), and [label](#).

## 6.186.4 Member Data Documentation

### 6.186.4.1 label

```
unsigned GiNaC::wildcard::label [private]
```

Label used to distinguish different wildcards.

Referenced by [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_label\(\)](#), and [read\\_↔archive\(\)](#).

The documentation for this class was generated from the following files:

- [wildcard.h](#)
- [wildcard.cpp](#)

## 6.187 GiNaC::zeta1\_SERIAL Class Reference

Complex conjugate.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.187.1 Detailed Description

Complex conjugate.

Real part. Imaginary part. Absolute value. Step function. Complex sign. Eta function:  $\log(a*b) == \log(a) + \log(b) + \eta(a, b)$ . Sine. Cosine. Tangent. Exponential function. Natural logarithm. Inverse sine (arc sine). Inverse cosine (arc cosine). Inverse tangent (arc tangent). Inverse tangent with two arguments. Hyperbolic Sine. Hyperbolic Cosine. Hyperbolic Tangent. Inverse hyperbolic Sine (area hyperbolic sine). Inverse hyperbolic Cosine (area hyperbolic cosine). Inverse hyperbolic Tangent (area hyperbolic tangent). Dilogarithm. Trilogarithm. Derivatives of Riemann's Zeta-function. Multiple zeta value including Riemann's zeta-function.

### 6.187.2 Member Data Documentation

#### 6.187.2.1 serial

```
unsigned GiNaC::zeta1_SERIAL::serial [static]
```

**Initial value:**

```
= function::register_new(function_options("zeta", 1).
    evalf_func(zetal_evalf).
    eval_func(zetal_eval).
    derivative_func(zetal_deriv).
    print_func<print_latex>(zetal_print_latex).
    do_not_evalf_params().
    overloaded(2))
```

Referenced by `GiNaC::zeta()`.

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## 6.188 GiNaC::zeta2\_SERIAL Class Reference

Alternating Euler sum or colored MZV.

```
#include <inifcns.h>
```



## Static Public Attributes

- static unsigned [serial](#)

### 6.188.1 Detailed Description

Alternating Euler sum or colored MZV.

### 6.188.2 Member Data Documentation

#### 6.188.2.1 [serial](#)

unsigned GiNaC::zeta2\_SERIAL::serial [static]

#### Initial value:

```
= function::register_new(function_options("zeta", 2).
    evalf_func(zeta2_evalf).
    eval_func(zeta2_eval).
    derivative_func(zeta2_deriv).
    print_func<print_latex>(zeta2_print_latex).
    do_not_evalf_params().
    overloaded(2))
```

Referenced by `GiNaC::zeta()`.

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)



## Chapter 7

# File Documentation

### 7.1 add.cpp File Reference

Implementation of [GiNaC](#)'s sums of expressions.

```
#include "add.h"
#include "mul.h"
#include "archive.h"
#include "operators.h"
#include "matrix.h"
#include "utils.h"
#include "clifford.h"
#include "ncmul.h"
#include "compiler.h"
#include <iostream>
#include <limits>
#include <stdexcept>
#include <string>
```

#### Namespaces

- [GiNaC](#)

#### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (add, expairseq, print\_func< print\_context >(&add::do\_print). print\_func< print\_latex >(&add::do\_print\_latex). print\_func< print\_csrc >(&add::do\_print\_csrc). print\_func< print\_tree >(&add::do\_print\_tree). print\_func< print\_python\_repr >(&add::do\_print\_python\_repr)) add
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (add)

#### 7.1.1 Detailed Description

Implementation of [GiNaC](#)'s sums of expressions.

## 7.2 add.h File Reference

Interface to [GiNaC](#)'s sums of expressions.

```
#include "expairseq.h"
```

### Classes

- class [GiNaC::add](#)  
*Sum of expressions.*

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (add)

#### 7.2.1 Detailed Description

Interface to [GiNaC](#)'s sums of expressions.

## 7.3 archive.cpp File Reference

Archiving of [GiNaC](#) expressions.

```
#include "archive.h"  
#include "registrar.h"  
#include "ex.h"  
#include "lst.h"  
#include "version.h"  
#include <iostream>  
#include <stdexcept>
```

### Namespaces

- [GiNaC](#)

## Functions

- static void [GiNaC::write\\_unsigned](#) (std::ostream &os, unsigned val)  
*Write unsigned integer quantity to stream.*
- static unsigned [GiNaC::read\\_unsigned](#) (std::istream &is)  
*Read unsigned integer quantity from stream.*
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const archive\_node &n)  
*Write [archive\\_node](#) to binary data stream.*
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const archive &ar)  
*Write archive to binary data stream.*
- std::istream & [GiNaC::operator>>](#) (std::istream &is, archive\_node &n)  
*Read [archive\\_node](#) from binary data stream.*
- std::istream & [GiNaC::operator>>](#) (std::istream &is, archive &ar)  
*Read archive from binary data stream.*
- static synthesizer\_func [GiNaC::find\\_factory\\_fcn](#) (const std::string &name)

### 7.3.1 Detailed Description

Archiving of [GiNaC](#) expressions.

## 7.4 archive.h File Reference

Archiving of [GiNaC](#) expressions.

```
#include "ex.h"
#include <iosfwd>
#include <map>
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::archive\\_node](#)  
*This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class).*
- struct [GiNaC::archive\\_node::property\\_info](#)  
*Information about a stored property.*
- struct [GiNaC::archive\\_node::property](#)  
*Archived property (data type, name and associated data)*
- struct [GiNaC::archive\\_node::archive\\_node\\_cit\\_range](#)
- class [GiNaC::unarchive\\_table\\_t](#)
- class [GiNaC::archive](#)  
*This class holds archived versions of [GiNaC](#) expressions (class ex).*
- struct [GiNaC::archive::archived\\_ex](#)  
*Archived expression descriptor.*

## Namespaces

- [GiNaC](#)

## Macros

- `#define GINAC_DECLARE_UNARCHIVER(classname)`  
*Helper macros to register a class with (un)archiving (a.k.a.*
- `#define GINAC_BIND_UNARCHIVER(classname)`

## Typedefs

- `typedef unsigned GiNaC::archive_node_id`  
*Numerical ID value to refer to an [archive\\_node](#).*
- `typedef unsigned GiNaC::archive_atom`  
*Numerical ID value to refer to a string.*
- `typedef basic *(* GiNaC::synthesize_func) ()`
- `typedef std::map< std::string, synthesize_func > GiNaC::unarchive_map_t`

## Functions

- `std::ostream & GiNaC::operator<< (std::ostream &os, const archive &ar)`  
*Write archive to binary data stream.*
- `std::istream & GiNaC::operator>> (std::istream &is, archive &ar)`  
*Read archive from binary data stream.*

## Variables

- `static unarchive_table_t GiNaC::unarch_table_instance`

### 7.4.1 Detailed Description

Archiving of [GiNaC](#) expressions.

### 7.4.2 Macro Definition Documentation

## 7.4.2.1 GINAC\_DECLARE\_UNARCHIVER

```
#define GINAC_DECLARE_UNARCHIVER(  
    classname )
```

**Value:**

```
class classname ## _unarchiver  
{  
    static int usecount;  
public:  
    static GiNaC::basic* create();  
    classname ## _unarchiver();  
    ~ classname ## _unarchiver();  
};  
static classname ## _unarchiver classname ## _unarchiver_instance
```

Helper macros to register a class with (un)archiving (a.k.a. (de)serialization).

Usage: put

```
GINAC_DECLARE_UNARCHIVER(myclass);
```

into the header file (in the global or namespace scope), and

```
GINAC_BIND_UNARCHIVER(myclass);
```

into the source file.

Effect: the 'myclass' (being a class derived directly or indirectly from [GiNaC::basic](#)) can be archived and unarchived.

Note: you need to use `GINAC_{DECLARE,BIND}_UNARCHIVER` incantations in order to make your class (un)archivable *even if your class does not overload 'read\_archive' method*. Sorry for inconvenience.

How it works:

The 'basic' class has a 'read\_archive' virtual method which reads an expression from archive. Derived classes can overload that method. There's a small problem, though. On unarchiving all we have is a set of named byte streams. In C++ the class name (as written in the source code) has nothing to do with its actual type. Thus, we need establish a correspondence ourselves. To do so we maintain a 'class\_name' => 'function\_pointer' table (see the `unarchive_table_t` class above). Every function in this table is supposed to create a new object of the 'class\_name' type. The 'archive\_node' class uses that table to construct an object of correct type. Next it invokes `read_archive` virtual method of newly created object, which does the actual job.

Note: this approach is very simple-minded (it does not handle classes with same names from different namespaces, multiple inheritance, etc), but it happens to work surprisingly well.

## 7.4.2.2 GINAC\_BIND\_UNARCHIVER

```
#define GINAC_BIND_UNARCHIVER(  
    classname )
```

**Value:**

```
classname ## _unarchiver::classname ## _unarchiver()  
{  
    static GiNaC::unarchive_table_t table;  
    if (usecount++ == 0) {  
        table.insert(std::string(#classname),  
            &(classname ## _unarchiver::create));  
    }  
}  
GiNaC::basic* classname ## _unarchiver::create()  
{  
    return new classname();  
}  
classname ## _unarchiver::~~ classname ## _unarchiver() { }  
int classname ## _unarchiver::usecount = 0
```

## 7.5 assertion.h File Reference

Assertion macro definition.

### Macros

- `#define GINAC_ASSERT(X) ((void)0)`  
*Assertion macro for checking invariances.*

### 7.5.1 Detailed Description

Assertion macro definition.

### 7.5.2 Macro Definition Documentation

#### 7.5.2.1 GINAC\_ASSERT

```
#define GINAC_ASSERT(  
    X ) ((void)0)
```

Assertion macro for checking invariances.

Referenced by `GiNaC::acos_deriv()`, `GiNaC::acosh_deriv()`, `GiNaC::add::add()`, `GiNaC::symmetry::add()`, `GiNaC::remember_table_list::add_entry()`, `GiNaC::remember_table::add_entry()`, `GiNaC::matrix::add_indexed()`, `GiNaC::algebraic_match_mul_with_mul()`, `GiNaC::function::archive()`, `GiNaC::asin_deriv()`, `GiNaC::asinh_deriv()`, `GiNaC::atan2_deriv()`, `GiNaC::atan_deriv()`, `GiNaC::atan_series()`, `GiNaC::atanh_deriv()`, `GiNaC::atanh_series()`, `GiNaC::beta_deriv()`, `GiNaC::beta_series()`, `GiNaC::mul::can_make_flat()`, `GiNaC::canonicalize()`, `GiNaC::clifford::clifford()`, `GiNaC::pseries::coeff()`, `GiNaC::add::combine_ex_with_coeff_to_pair()`, `GiNaC::mul::combine_ex_with_coeff_to_pair()`, `GiNaC::mul::combine_overall_coeff()`, `GiNaC::add::combine_pair_with_coeff_to_pair()`, `GiNaC::mul::combine_pair_with_coeff_to_pair()`, `GiNaC::function::conjugate()`, `GiNaC::ex::construct_from_basic()`, `GiNaC::matrix::contract_with()`, `GiNaC::tensdelta::contract_with()`, `GiNaC::tensmetric::contract_with()`, `GiNaC::su3f::contract_with()`, `GiNaC::su3f::contract_with()`, `GiNaC::su3d::contract_with()`, `GiNaC::spinmetric::contract_with()`, `GiNaC::tensepsilon::contract_with()`, `GiNaC::cos_deriv()`, `GiNaC::cosh_deriv()`, `GiNaC::ex::denom()`, `GiNaC::matrix::determinant()`, `GiNaC::matrix::division_free_elimination()`, `GiNaC::mul::do_print_latex()`, `GiNaC::add::eval()`, `GiNaC::mul::eval()`, `GiNaC::power::eval()`, `GiNaC::ncmul::eval()`, `GiNaC::indexed::eval()`, `GiNaC::function::eval()`, `GiNaC::matrix::eval_indexed()`, `GiNaC::tensdelta::eval_indexed()`, `GiNaC::tensmetric::eval_indexed()`, `GiNaC::minkmetric::eval_indexed()`, `GiNaC::su3f::eval_indexed()`, `GiNaC::su3d::eval_indexed()`, `GiNaC::spinmetric::eval_indexed()`, `GiNaC::tensepsilon::eval_indexed()`, `GiNaC::function::evalf()`, `GiNaC::ex::ex()`, `GiNaC::ex_to()`, `GiNaC::exp_deriv()`, `GiNaC::expair::expair()`, `GiNaC::indexed::expand()`, `GiNaC::function::expand()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_add_2()`, `GiNaC::power::expand_mul()`, `GiNaC::function::exp_derivative()`, `GiNaC::frac_cancel()`, `GiNaC::matrix::fraction_free_elimination()`, `GiNaC::matrix::gauss_elimination()`, `GiNaC::gcd_pf_mul()`, `GiNaC::gcd_pf_pow()`, `GiNaC::indexed::get_indices()`, `GiNaC::function::get_name()`, `GiNaC::H_deriv()`, `GiNaC::function::imag_part()`, `GiNaC::function::info()`, `GiNaC::add::integer_content()`, `GiNaC::mul::integer_content()`, `GiNaC::expair::is_canonical_numeric()`, `GiNaC::remember_table_entry::is_equal()`, `GiNaC::constant::is_equal_same_type()`, `GiNaC::symbol::is_equal_same_type()`, `GiNaC::fderivative::is_equal_same_type()`, `GiNaC::numeric::is_equal_same_type()`, `GiNaC::container< C >::is_equal_same_type()`, `GiNaC::structure< T, ComparisonPolicy >::is_equal_same_type()`, `GiNaC::function::is_equal_same_type()`, `GiNaC::matrix::let_op()`, `GiNaC::clifford::let_op()`, `GiNaC::container< C >::let_op()`, `GiNaC::lgamma_deriv()`, `GiNaC::Li2_deriv()`, `GiNaC::Li_deriv()`, `GiNaC::Li_eval()`, `GiNaC::log_deriv()`, `GiNaC::log_series()`, `GiNaC::remember_table::lookup_entry()`, `GiNaC::solve()`, `GiNaC::ex::makewriteable()`, `GiNaC::matrix::markowitz_elimination()`, `GiNaC::color::match_same_type()`, `GiNaC::clifford::match_same_type()`, `GiNaC::idx::match_same_type()`, `GiNaC::relational::match_same_type()`, `GiNaC::matrix::match_same_type()`, `GiNaC::fderivative::match_same_type()`,



[GiNaC::varidx::match\\_same\\_type\(\)](#), [GiNaC::spindidx::match\\_same\\_type\(\)](#), [GiNaC::function::match\\_same\\_type\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::ex::numer\(\)](#), [GiNaC::ex::numer\\_denom\(\)](#), [GiNaC::integral::op\(\)](#), [GiNaC::idx::op\(\)](#), [GiNaC::matrix::op\(\)](#), [GiNaC::power::op\(\)](#), [GiNaC::relational::op\(\)](#), [GiNaC::clifford::op\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::sy\\_is\\_less::operator\(\)](#), [GiNaC::sy\\_swap::operator\(\)](#), [GiNaC::Order\\_series\(\)](#), [GiNaC::function::pderivative\(\)](#), [GiNaC::permute\\_free\\_index\\_to\\_front\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::function::power\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::psi1\\_deriv\(\)](#), [GiNaC::psi2\\_deriv\(\)](#), [GiNaC::ptr< GiNaC::basic >::ptr\(\)](#), [GiNaC::matrix::rank\(\)](#), [GiNaC::function::real\\_part\(\)](#), [GiNaC::rename\\_dummy\\_indices\(\)](#), [GiNaC::relational::return\\_type\(\)](#), [GiNaC::ncmul::return\\_type\(\)](#), [GiNaC::function::return\\_type\(\)](#), [GiNaC::relational::return\\_type\\_tinfo\(\)](#), [GiNaC::function::return\\_type\\_tinfo\(\)](#), [GiNaC::S\\_deriv\(\)](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::function::series\(\)](#), [GiNaC::simplify\\_indexed\\_product\(\)](#), [GiNaC::sin\\_deriv\(\)](#), [GiNaC::sinh\\_deriv\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::ex::swap\(\)](#), [GiNaC::tan\\_deriv\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\\_deriv\(\)](#), [GiNaC::tanh\\_series\(\)](#), [GiNaC::tgamma\\_deriv\(\)](#), [GiNaC::numeric::to\\_double\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), [GiNaC::numeric::to\\_long\(\)](#), [GiNaC::indexed::validate\(\)](#), [GiNaC::zeta1\\_deriv\(\)](#), [GiNaC::zeta2\\_deriv\(\)](#), [GiNaC::zetaderiv\\_deriv\(\)](#), and [GiNaC::basic::~~basic\(\)](#).

## 7.6 basic.cpp File Reference

Implementation of [GiNaC's ABC](#).

```

#include "basic.h"
#include "ex.h"
#include "numeric.h"
#include "power.h"
#include "add.h"
#include "symbol.h"
#include "lst.h"
#include "ncmul.h"
#include "relational.h"
#include "operators.h"
#include "wildcard.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include "inifcns.h"
#include <iostream>
#include <stdexcept>
#include <typeinfo>

```

### Classes

- struct [GiNaC::evalf\\_map\\_function](#)  
Function object to be applied by [basic::evalf\(\)](#).
- struct [GiNaC::evalm\\_map\\_function](#)  
Function object to be applied by [basic::evalm\(\)](#).
- struct [GiNaC::eval\\_integ\\_map\\_function](#)  
Function object to be applied by [basic::eval\\_integ\(\)](#).
- struct [GiNaC::derivative\\_map\\_function](#)  
Function object to be applied by [basic::derivative\(\)](#).
- struct [GiNaC::expand\\_map\\_function](#)  
Function object to be applied by [basic::expand\(\)](#).

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (basic, void, print\_func< print\_context >(&basic::do\_print). print\_func< print\_tree >(&basic::do\_print\_tree). print\_func< print\_python\_repr >(&basic::do\_print\_python\_repr)) basic

*basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by duplicate()), so it can copy the tinfo\_key and the hash value.*

## Variables

- [GiNaC::evalm\\_map\\_function](#) [GiNaC::map\\_evalm](#)
- [GiNaC::eval\\_integ\\_map\\_function](#) [GiNaC::map\\_eval\\_integ](#)

### 7.6.1 Detailed Description

Implementation of [GiNaC](#)'s ABC.

## 7.7 basic.h File Reference

Interface to [GiNaC](#)'s ABC.

```
#include "flags.h"
#include "ptr.h"
#include "assertion.h"
#include "registrar.h"
#include <cstdlib>
#include <map>
#include <set>
#include <typeinfo>
#include <vector>
#include <utility>
```

## Classes

- struct [GiNaC::map\\_function](#)  
*Function object for map().*
- class [GiNaC::visitor](#)  
*Degenerate base class for visitors.*
- class [GiNaC::basic](#)  
*This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.*

## Namespaces

- [GiNaC](#)

## Typedefs

- typedef std::vector< ex > [GiNaC::exvector](#)
- typedef std::set< ex, ex\_is\_less > [GiNaC::exset](#)
- typedef std::map< ex, ex, ex\_is\_less > [GiNaC::exmap](#)

## Functions

- template<class T >  
bool [GiNaC::is\\_a](#) (const basic &obj)  
*Check if obj is a T, including base classes.*
- template<class T >  
bool [GiNaC::is\\_exactly\\_a](#) (const basic &obj)  
*Check if obj is a T, not including base classes.*
- template<class B , typename... Args>  
B & [GiNaC::dynallocate](#) (Args &&... args)  
*Constructs a new (class basic or derived) B object on the heap.*
- template<class B >  
B & [GiNaC::dynallocate](#) (std::initializer\_list< ex > il)  
*Constructs a new (class basic or derived) B object on the heap.*

### 7.7.1 Detailed Description

Interface to [GiNaC](#)'s ABC.

## 7.8 class\_info.h File Reference

Helper templates to provide per-class information for class hierarchies.

```
#include <cstddef>
#include <cstring>
#include <iomanip>
#include <iostream>
#include <map>
#include <stdexcept>
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::class\\_info< OPT >](#)
- struct [GiNaC::class\\_info< OPT >::tree\\_node](#)

## Namespaces

- [GiNaC](#)

### 7.8.1 Detailed Description

Helper templates to provide per-class information for class hierarchies.

## 7.9 clifford.cpp File Reference

Implementation of [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

```
#include "clifford.h"
#include "ex.h"
#include "idx.h"
#include "ncmul.h"
#include "symbol.h"
#include "numeric.h"
#include "symmetry.h"
#include "lst.h"
#include "relational.h"
#include "operators.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "matrix.h"
#include "archive.h"
#include "utils.h"
#include <stdexcept>
```

## Classes

- struct [GiNaC::is\\_not\\_a\\_clifford](#)  
*Predicate for finding non-clifford objects.*

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (clifford, indexed, print\_func< print\_dflt >(&clifford::do\_print\_dflt). print\_func< print\_latex >(&clifford::do\_print\_latex). print\_func< print\_tree >(&clifford::do\_print\_tree)) GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT(diracone
- [GiNaC::print\\_func< print\\_dflt >](#) (&diracone::do\_print). print\_func< print\_latex >(&diracone
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (clifford)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (cliffordunit)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (diracone)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (diracgamma)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (diracgamma5)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (diracgammaL)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (diracgammaR)
- static bool [GiNaC::is\\_dirac\\_slash](#) (const ex &seq0)
- static void [GiNaC::base\\_and\\_index](#) (const ex &c, ex &b, ex &i)  
*This function decomposes  $\gamma \sim \mu \rightarrow (1, \mu)$  and  $a \rightarrow (a.ix, ix)$*
- ex [GiNaC::dirac\\_ONE](#) (unsigned char rl=0)  
*Create a Clifford unity object.*
- static unsigned [GiNaC::get\\_dim\\_uint](#) (const ex &e)
- ex [GiNaC::clifford\\_unit](#) (const ex &mu, const ex &metr, unsigned char rl=0)  
*Create a Clifford unit object.*
- ex [GiNaC::dirac\\_gamma](#) (const ex &mu, unsigned char rl=0)  
*Create a Dirac gamma object.*
- ex [GiNaC::dirac\\_gamma5](#) (unsigned char rl=0)  
*Create a Dirac gamma5 object.*
- ex [GiNaC::dirac\\_gammaL](#) (unsigned char rl=0)  
*Create a Dirac gammaL object.*
- ex [GiNaC::dirac\\_gammaR](#) (unsigned char rl=0)  
*Create a Dirac gammaR object.*
- ex [GiNaC::dirac\\_slash](#) (const ex &e, const ex &dim, unsigned char rl=0)  
*Create a term of the form  $e_\mu * \gamma \sim \mu$  with a unique index  $\mu$ .*
- static unsigned char [GiNaC::get\\_representation\\_label](#) (const return\_type\_t &ti)  
*Extract representation label from tinfo key (as returned by return\_type\_tinfo()).*
- static ex [GiNaC::trace\\_string](#) (exvector::const\_iterator ix, size\_t num)  
*Take trace of a string of an even number of Dirac gammas given a vector of indices.*
- ex [GiNaC::dirac\\_trace](#) (const ex &e, const std::set< unsigned char > &rls, const ex &trONE=4)  
*Calculate dirac traces over the specified set of representation labels.*
- ex [GiNaC::dirac\\_trace](#) (const ex &e, const lst &rl, const ex &trONE=4)  
*Calculate dirac traces over the specified list of representation labels.*
- ex [GiNaC::dirac\\_trace](#) (const ex &e, unsigned char rl=0, const ex &trONE=4)  
*Calculate the trace of an expression containing gamma objects with a specified representation label.*
- ex [GiNaC::canonicalize\\_clifford](#) (const ex &e)  
*Bring all products of clifford objects in an expression into a canonical order.*
- ex [GiNaC::clifford\\_star\\_bar](#) (const ex &e, bool do\_bar, unsigned options)  
*An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).*
- ex [GiNaC::clifford\\_prime](#) (const ex &e)  
*Automorphism of the Clifford algebra, simply changes signs of all clifford units.*
- ex [GiNaC::remove\\_dirac\\_ONE](#) (const ex &e, unsigned char rl=0, unsigned options=0)  
*Replaces [dirac\\_ONE](#)'s (with a representation\_label no less than rl) in e with 1.*
- int [GiNaC::clifford\\_max\\_label](#) (const ex &e, bool ignore\_ONE=false)  
*Returns the maximal representation label of a clifford object if e contains at least one, otherwise returns -1.*
- ex [GiNaC::clifford\\_norm](#) (const ex &e)

- *Calculation of the norm in the Clifford algebra.*  
ex [GiNaC::clifford\\_inverse](#) (const ex &e)
- *Calculation of the inverse in the Clifford algebra.*  
ex [GiNaC::lst\\_to\\_clifford](#) (const ex &v, const ex &mu, const ex &metr, unsigned char rl=0)
- *List or vector conversion into the Clifford vector.*  
ex [GiNaC::lst\\_to\\_clifford](#) (const ex &v, const ex &e)
- static ex [GiNaC::get\\_clifford\\_comp](#) (const ex &e, const ex &c, bool root=true)
- *Auxiliary structure to define a function for stripping one Clifford unit from vectors.*  
lst [GiNaC::clifford\\_to\\_lst](#) (const ex &e, const ex &c, bool algebraic=true)
- *An inverse function to [lst\\_to\\_clifford\(\)](#).*  
ex [GiNaC::clifford\\_moebius\\_map](#) (const ex &a, const ex &b, const ex &c, const ex &d, const ex &v, const ex &G, unsigned char rl=0)
- *Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b|c d) in linear spaces with arbitrary signature.*  
ex [GiNaC::clifford\\_moebius\\_map](#) (const ex &M, const ex &v, const ex &G, unsigned char rl=0)
- *The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*

## Variables

- [GiNaC::tensor](#)

## 7.9.1 Detailed Description

Implementation of [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

## 7.10 clifford.h File Reference

Interface to [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

```
#include "indexed.h"
#include "tensor.h"
#include "symbol.h"
#include "idx.h"
#include <set>
```

## Classes

- class [GiNaC::clifford](#)  
*This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).*
- class [GiNaC::diracone](#)  
*This class represents the Clifford algebra unity element.*
- class [GiNaC::cliffordunit](#)  
*This class represents the Clifford algebra generators (units).*
- class [GiNaC::diracgamma](#)  
*This class represents the Dirac gamma Lorentz vector.*
- class [GiNaC::diracgamma5](#)  
*This class represents the Dirac gamma5 object which anticommutes with all other gammas.*
- class [GiNaC::diracgammaL](#)  
*This class represents the Dirac gammaL object which behaves like 1/2 (1-gamma5).*
- class [GiNaC::diracgammaR](#)  
*This class represents the Dirac gammaL object which behaves like 1/2 (1+gamma5).*

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (clifford)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (diracone)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (cliffordunit)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (diracgamma)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (diracgamma5)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (diracgammaL)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (diracgammaR)
- [bool GiNaC::is\\_clifford\\_tinfo](#) (const return\_type\_t &ti)  
*Check whether a given [return\\_type\\_t](#) object (as returned by [return\\_type\\_tinfo\(\)](#)) is that of a clifford object (with an arbitrary representation label).*
- [ex GiNaC::dirac\\_ONE](#) (unsigned char rl=0)  
*Create a Clifford unity object.*
- [ex GiNaC::clifford\\_unit](#) (const ex &mu, const ex &metr, unsigned char rl=0)  
*Create a Clifford unit object.*
- [ex GiNaC::dirac\\_gamma](#) (const ex &mu, unsigned char rl=0)  
*Create a Dirac gamma object.*
- [ex GiNaC::dirac\\_gamma5](#) (unsigned char rl=0)  
*Create a Dirac gamma5 object.*
- [ex GiNaC::dirac\\_gammaL](#) (unsigned char rl=0)  
*Create a Dirac gammaL object.*
- [ex GiNaC::dirac\\_gammaR](#) (unsigned char rl=0)  
*Create a Dirac gammaR object.*
- [ex GiNaC::dirac\\_slash](#) (const ex &e, const ex &dim, unsigned char rl=0)  
*Create a term of the form  $e_{\mu} * \gamma_{\sim \mu}$  with a unique index  $\mu$ .*
- [ex GiNaC::dirac\\_trace](#) (const ex &e, const std::set< unsigned char > &rls, const ex &trONE=4)  
*Calculate dirac traces over the specified set of representation labels.*
- [ex GiNaC::dirac\\_trace](#) (const ex &e, const lst &rl, const ex &trONE=4)  
*Calculate dirac traces over the specified list of representation labels.*
- [ex GiNaC::dirac\\_trace](#) (const ex &e, unsigned char rl=0, const ex &trONE=4)  
*Calculate the trace of an expression containing gamma objects with a specified representation label.*
- [ex GiNaC::canonicalize\\_clifford](#) (const ex &e)  
*Bring all products of clifford objects in an expression into a canonical order.*
- [ex GiNaC::clifford\\_prime](#) (const ex &e)  
*Automorphism of the Clifford algebra, simply changes signs of all clifford units.*
- [ex GiNaC::clifford\\_star\\_bar](#) (const ex &e, bool do\_bar, unsigned options)  
*An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).*
- [ex GiNaC::clifford\\_bar](#) (const ex &e)  
*Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.*
- [ex GiNaC::clifford\\_star](#) (const ex &e)  
*Reversion of the Clifford algebra, reverse the order of all clifford units in ncmul.*
- [ex GiNaC::remove\\_dirac\\_ONE](#) (const ex &e, unsigned char rl=0, unsigned options=0)  
*Replaces [dirac\\_ONE](#)'s (with a [representation\\_label](#) no less than  $rl$ ) in  $e$  with 1.*
- [int GiNaC::clifford\\_max\\_label](#) (const ex &e, bool ignore\_ONE=false)  
*Returns the maximal representation label of a clifford object if  $e$  contains at least one, otherwise returns -1.*
- [ex GiNaC::clifford\\_norm](#) (const ex &e)

*Calculation of the norm in the Clifford algebra.*

- ex [GiNaC::clifford\\_inverse](#) (const ex &e)

*Calculation of the inverse in the Clifford algebra.*

- ex [GiNaC::lst\\_to\\_clifford](#) (const ex &v, const ex &mu, const ex &metr, unsigned char rl=0)

*List or vector conversion into the Clifford vector.*

- ex [GiNaC::lst\\_to\\_clifford](#) (const ex &v, const ex &e)
- lst [GiNaC::clifford\\_to\\_lst](#) (const ex &e, const ex &c, bool algebraic=true)

*An inverse function to [lst\\_to\\_clifford\(\)](#).*

- ex [GiNaC::clifford\\_moebius\\_map](#) (const ex &a, const ex &b, const ex &c, const ex &d, const ex &v, const ex &G, unsigned char rl=0)

*Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b\c d) in linear spaces with arbitrary signature.*

- ex [GiNaC::clifford\\_moebius\\_map](#) (const ex &M, const ex &v, const ex &G, unsigned char rl=0)

*The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*

### 7.10.1 Detailed Description

Interface to [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

## 7.11 color.cpp File Reference

Implementation of [GiNaC](#)'s color (SU(3) Lie algebra) objects.

```
#include "color.h"
#include "idx.h"
#include "ncmul.h"
#include "symmetry.h"
#include "operators.h"
#include "numeric.h"
#include "mul.h"
#include "power.h"
#include "symbol.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
#include <stdexcept>
```

### Namespaces

- [GiNaC](#)

### Macros

- #define [TEST\\_PERMUTATION](#)(A, B, C, P)
- #define [CMPINDICES](#)(A, B, C) ((v[0] == (A)) && (v[1] == (B)) && (v[2] == (C)))



## Functions

- `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (su3one, tensor, print\_func< print\_dflt >(&su3one::do\_print), print\_func< print\_latex >(&su3one::do\_print\_latex)) GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT(su3t
- `GiNaC::print_func< print_dflt >` (&su3t::do\_print), print\_func< print\_latex >(&su3t
- `GiNaC::GINAC_BIND_UNARCHIVER` (color)
- `GiNaC::GINAC_BIND_UNARCHIVER` (su3one)
- `GiNaC::GINAC_BIND_UNARCHIVER` (su3t)
- `GiNaC::GINAC_BIND_UNARCHIVER` (su3f)
- `GiNaC::GINAC_BIND_UNARCHIVER` (su3d)
- static ex `GiNaC::permute_free_index_to_front` (const exvector &iv3, const exvector &iv2, int &sig)
 

*Given a vector iv3 of three indices and a vector iv2 of two indices that is a subset of iv3, return the (free) index that is in iv3 but not in iv2 and the sign introduced by permuting that index to the front.*
- ex `GiNaC::color_ONE` (unsigned char rl=0)
 

*Create the su(3) unity element.*
- ex `GiNaC::color_T` (const ex &a, unsigned char rl=0)
 

*Create an su(3) generator.*
- ex `GiNaC::color_f` (const ex &a, const ex &b, const ex &c)
 

*Create an su(3) antisymmetric structure constant.*
- ex `GiNaC::color_d` (const ex &a, const ex &b, const ex &c)
 

*Create an su(3) symmetric structure constant.*
- ex `GiNaC::color_h` (const ex &a, const ex &b, const ex &c)
 

*This returns the linear combination d.a.b.c+l\*f.a.b.c.*
- static bool `GiNaC::is_color_tinfo` (const return\_type\_t &ti)
 

*Check whether a given tinfo key (as returned by return\_type\_tinfo()) is that of a color object (with an arbitrary representation label).*
- static unsigned char `GiNaC::get_representation_label` (const return\_type\_t &ti)
 

*Extract representation label from tinfo key (as returned by return\_type\_tinfo()).*
- ex `GiNaC::color_trace` (const ex &e, const std::set< unsigned char > &rls)
 

*Calculate color traces over the specified set of representation labels.*
- ex `GiNaC::color_trace` (const ex &e, const lst &rl)
 

*Calculate color traces over the specified list of representation labels.*
- ex `GiNaC::color_trace` (const ex &e, unsigned char rl=0)
 

*Calculate the trace of an expression containing color objects with a specified representation label.*

### 7.11.1 Detailed Description

Implementation of `GiNaC`'s color (SU(3) Lie algebra) objects.

### 7.11.2 Macro Definition Documentation

### 7.11.2.1 TEST\_PERMUTATION

```
#define TEST_PERMUTATION(
    A,
    B,
    C,
    P )
```

**Value:**

```
if (iv3[B].is_equal(iv2[0]) && iv3[C].is_equal(iv2[1])) { \
    sig = P; \
    return iv3[A]; \
}
```

Referenced by `GiNaC::permute_free_index_to_front()`.

### 7.11.2.2 CMPINDICES

```
#define CMPINDICES(
    A,
    B,
    C ) ((v[0] == (A)) && (v[1] == (B)) && (v[2] == (C)))
```

Referenced by `GiNaC::su3f::eval_indexed()`, and `GiNaC::su3d::eval_indexed()`.

## 7.12 color.h File Reference

Interface to [GiNaC](#)'s color (SU(3) Lie algebra) objects.

```
#include "indexed.h"
#include "tensor.h"
#include <set>
```

### Classes

- class [GiNaC::color](#)  
*This class holds a generator  $T_a$  or the unity element of the Lie algebra of SU(3), as used for calculations in quantum chromodynamics.*
- class [GiNaC::su3one](#)  
*This class represents the su(3) unity element.*
- class [GiNaC::su3t](#)  
*This class represents an su(3) generator.*
- class [GiNaC::su3f](#)  
*This class represents the tensor of antisymmetric su(3) structure constants.*
- class [GiNaC::su3d](#)  
*This class represents the tensor of symmetric su(3) structure constants.*

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (color)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (su3one)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (su3t)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (su3f)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (su3d)
- ex [GiNaC::color\\_ONE](#) (unsigned char rl=0)  
*Create the su(3) unity element.*
- ex [GiNaC::color\\_T](#) (const ex &a, unsigned char rl=0)  
*Create an su(3) generator.*
- ex [GiNaC::color\\_f](#) (const ex &a, const ex &b, const ex &c)  
*Create an su(3) antisymmetric structure constant.*
- ex [GiNaC::color\\_d](#) (const ex &a, const ex &b, const ex &c)  
*Create an su(3) symmetric structure constant.*
- ex [GiNaC::color\\_h](#) (const ex &a, const ex &b, const ex &c)  
*This returns the linear combination d.a.b.c+l\*f.a.b.c.*
- ex [GiNaC::color\\_trace](#) (const ex &e, const std::set< unsigned char > &rls)  
*Calculate color traces over the specified set of representation labels.*
- ex [GiNaC::color\\_trace](#) (const ex &e, const lst &rls)  
*Calculate color traces over the specified list of representation labels.*
- ex [GiNaC::color\\_trace](#) (const ex &e, unsigned char rl=0)  
*Calculate the trace of an expression containing color objects with a specified representation label.*

### 7.12.1 Detailed Description

Interface to [GiNaC](#)'s color (SU(3) Lie algebra) objects.

## 7.13 compiler.h File Reference

Definition of optimizing macros.

## Macros

- `#define unlikely(cond) (cond)`
- `#define likely(cond) (cond)`
- `#define attribute_deprecated`

### 7.13.1 Detailed Description

Definition of optimizing macros.

## 7.13.2 Macro Definition Documentation

### 7.13.2.1 unlikely

```
#define unlikely(  
    cond ) (cond)
```

Referenced by `GiNaC::add::eval()`, and `GiNaC::mul::eval()`.

### 7.13.2.2 likely

```
#define likely(  
    cond ) (cond)
```

Referenced by `GiNaC::add::combine_ex_with_coeff_to_pair()`, `GiNaC::mul::eval()`, and `GiNaC::power::eval()`.

### 7.13.2.3 attribute\_deprecated

```
#define attribute_deprecated
```

## 7.14 constant.cpp File Reference

Implementation of [GiNaC](#)'s constant types and some special constants.

```
#include "constant.h"  
#include "numeric.h"  
#include "ex.h"  
#include "archive.h"  
#include "utils.h"  
#include "inifcns.h"  
#include <iostream>  
#include <stdexcept>  
#include <string>
```

### Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (constant, basic, print\_func< print\_context >(&constant::do\_print). print\_func< print\_latex >(&constant::do\_print\_latex). print\_func< print\_tree >(&constant::do\_print\_tree). print\_func< print\_python\_repr >(&constant::do\_print\_python\_repr)) constant
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (constant)

## Variables

- const constant [GiNaC::Pi](#) ("Pi", PiEvalf, "\i", domain::positive)  
*Pi.*
- const constant [GiNaC::Euler](#) ("Euler", EulerEvalf, "\amma\_E", domain::positive)  
*Euler's constant.*
- const constant [GiNaC::Catalan](#) ("Catalan", CatalanEvalf, "G", domain::positive)  
*Catalan's constant.*

### 7.14.1 Detailed Description

Implementation of [GiNaC](#)'s constant types and some special constants.

## 7.15 constant.h File Reference

Interface to [GiNaC](#)'s constant types and some special constants.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include <string>
```

## Classes

- class [GiNaC::constant](#)  
*This class holds constants, symbols with specific numerical value.*

## Namespaces

- [GiNaC](#)

## Typedefs

- typedef `ex(* GiNaC::evalffunctype)()`

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (constant)

### 7.15.1 Detailed Description

Interface to [GiNaC](#)'s constant types and some special constants.

## 7.16 container.h File Reference

Wrapper template for making [GiNaC](#) classes out of STL containers.

```
#include "ex.h"
#include "print.h"
#include "archive.h"
#include "assertion.h"
#include "compiler.h"
#include <algorithm>
#include <iterator>
#include <list>
#include <memory>
#include <stdexcept>
#include <vector>
```

### Classes

- class [GiNaC::container\\_storage< C >](#)  
*Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.*
- class [GiNaC::container< C >](#)  
*Wrapper template for making [GiNaC](#) classes out of STL containers.*

### Namespaces

- [GiNaC](#)

### 7.16.1 Detailed Description

Wrapper template for making [GiNaC](#) classes out of STL containers.

## 7.17 crc32.h File Reference

CRC32 hash function.

### Namespaces

- [GiNaC](#)

## Functions

- static unsigned [GiNaC::crc32](#) (const char \*c, const unsigned len, const unsigned crcinit)

## Variables

- static unsigned const [GiNaC::crctab](#) [256]

### 7.17.1 Detailed Description

CRC32 hash function.

Shamelessly stolen from GNU coreutils (cksum.c).

## 7.18 ex.cpp File Reference

Implementation of [GiNaC](#)'s light-weight expression handles.

```
#include "ex.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "numeric.h"
#include "matrix.h"
#include "power.h"
#include "lst.h"
#include "relational.h"
#include "utils.h"
#include <iostream>
#include <stdexcept>
```

## Namespaces

- [GiNaC](#)

### 7.18.1 Detailed Description

Implementation of [GiNaC](#)'s light-weight expression handles.

## 7.19 ex.h File Reference

Interface to [GiNaC](#)'s light-weight expression handles.

```
#include "basic.h"
#include "ptr.h"
#include <functional>
#include <iosfwd>
#include <iterator>
#include <memory>
#include <stack>
```

## Classes

- class [GiNaC::library\\_init](#)  
*Helper class to initialize the library.*
- class [GiNaC::ex](#)  
*Lightweight wrapper for [GiNaC](#)'s symbolic objects.*
- class [GiNaC::const\\_iterator](#)
- struct [GiNaC::internal::\\_iter\\_rep](#)
- class [GiNaC::const\\_preorder\\_iterator](#)
- class [GiNaC::const\\_postorder\\_iterator](#)
- struct [GiNaC::ex\\_is\\_less](#)
- struct [GiNaC::ex\\_is\\_equal](#)
- struct [GiNaC::op0\\_is\\_equal](#)
- struct [GiNaC::ex\\_swap](#)
- class [GiNaC::pointer\\_to\\_map\\_function](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >](#)
- struct [std::hash< GiNaC::ex >](#)  
*Specialization of `std::hash()` for `ex` objects.*
- struct [std::equal\\_to< GiNaC::ex >](#)  
*Specialization of `std::equal_to()` for `ex` objects.*

## Namespaces

- [GiNaC](#)
- [GiNaC::internal](#)
- [std](#)

## Functions

- bool [GiNaC::are\\_ex\\_trivially\\_equal](#) (const `ex` &`e1`, const `ex` &`e2`)  
*Compare two objects of class quickly without doing a deep tree traversal.*
- `std::ostream` & [GiNaC::operator<<](#) (`std::ostream` &`os`, const `exvector` &`e`)
- `std::ostream` & [GiNaC::operator<<](#) (`std::ostream` &`os`, const `exset` &`e`)
- `std::ostream` & [GiNaC::operator<<](#) (`std::ostream` &`os`, const `exmap` &`e`)
- `size_t` [GiNaC::nops](#) (const `ex` &`thisex`)
- `ex` [GiNaC::expand](#) (const `ex` &`thisex`, unsigned `options`=0)
- `ex` [GiNaC::conjugate](#) (const `ex` &`thisex`)
- `ex` [GiNaC::real\\_part](#) (const `ex` &`thisex`)
- `ex` [GiNaC::imag\\_part](#) (const `ex` &`thisex`)
- bool [GiNaC::has](#) (const `ex` &`thisex`, const `ex` &`pattern`, unsigned `options`=0)
- bool [GiNaC::find](#) (const `ex` &`thisex`, const `ex` &`pattern`, `exset` &`found`)
- bool [GiNaC::is\\_polynomial](#) (const `ex` &`thisex`, const `ex` &`vars`)
- int [GiNaC::degree](#) (const `ex` &`thisex`, const `ex` &`s`)
- int [GiNaC::ldegree](#) (const `ex` &`thisex`, const `ex` &`s`)
- `ex` [GiNaC::coeff](#) (const `ex` &`thisex`, const `ex` &`s`, int `n`=1)
- `ex` [GiNaC::numer](#) (const `ex` &`thisex`)



- ex [GiNaC::denom](#) (const ex &thisex)
- ex [GiNaC::numer\\_denom](#) (const ex &thisex)
- ex [GiNaC::normal](#) (const ex &thisex)
- ex [GiNaC::to\\_rational](#) (const ex &thisex, exmap &repl)
- ex [GiNaC::to\\_polynomial](#) (const ex &thisex, exmap &repl)
- ex [GiNaC::collect](#) (const ex &thisex, const ex &s, bool distributed=false)
- ex [GiNaC::eval](#) (const ex &thisex)
- ex [GiNaC::evalf](#) (const ex &thisex)
- ex [GiNaC::evalm](#) (const ex &thisex)
- ex [GiNaC::eval\\_integ](#) (const ex &thisex)
- ex [GiNaC::diff](#) (const ex &thisex, const symbol &s, unsigned nth=1)
- ex [GiNaC::series](#) (const ex &thisex, const ex &r, int [order](#), unsigned [options](#)=0)
- bool [GiNaC::match](#) (const ex &thisex, const ex &pattern, exmap &repl\_lst)
- ex [GiNaC::simplify\\_indexed](#) (const ex &thisex, unsigned [options](#)=0)
- ex [GiNaC::simplify\\_indexed](#) (const ex &thisex, const scalar\_products &sp, unsigned [options](#)=0)
- ex [GiNaC::symmetrize](#) (const ex &thisex)
- ex [GiNaC::symmetrize](#) (const ex &thisex, const lst &l)
- ex [GiNaC::antisymmetrize](#) (const ex &thisex)
- ex [GiNaC::antisymmetrize](#) (const ex &thisex, const lst &l)
- ex [GiNaC::symmetrize\\_cyclic](#) (const ex &thisex)
- ex [GiNaC::symmetrize\\_cyclic](#) (const ex &thisex, const lst &l)
- ex [GiNaC::op](#) (const ex &thisex, size\_t i)
- ex [GiNaC::lhs](#) (const ex &thisex)
- ex [GiNaC::rhs](#) (const ex &thisex)
- bool [GiNaC::is\\_zero](#) (const ex &thisex)
- void [GiNaC::swap](#) (ex &e1, ex &e2)
- ex [GiNaC::subs](#) (const ex &thisex, const exmap &m, unsigned [options](#)=0)
- ex [GiNaC::subs](#) (const ex &thisex, const lst &ls, const lst &lr, unsigned [options](#)=0)
- ex [GiNaC::subs](#) (const ex &thisex, const ex &e, unsigned [options](#)=0)
- template<class T >  
bool [GiNaC::is\\_a](#) (const ex &obj)  
*Check if ex is a handle to a T, including base classes.*
- template<class T >  
bool [GiNaC::is\\_exactly\\_a](#) (const ex &obj)  
*Check if ex is a handle to a T, not including base classes.*
- template<class T >  
const T & [GiNaC::ex\\_to](#) (const ex &e)  
*Return a reference to the basic-derived class T object embedded in an expression.*
- template<>  
void [std::swap](#) ([GiNaC::ex](#) &a, [GiNaC::ex](#) &b)  
*Specialization of [std::swap\(\)](#) for ex objects.*

## Variables

- static library\_init [GiNaC::library\\_initializer](#)  
*For construction of flyweights, etc.*
- const basic \* [GiNaC::\\_num0\\_bp](#)

## 7.19.1 Detailed Description

Interface to [GiNaC](#)'s light-weight expression handles.

## 7.20 excompiler.cpp File Reference

Functions to facilitate the conversion of a `ex` to a function pointer suited for fast numerical integration.

```
#include "excompiler.h"
#include "ex.h"
#include "lst.h"
#include "operators.h"
#include "relational.h"
#include "symbol.h"
#include <cstdlib>
#include <fstream>
#include <ios>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
```

### Namespaces

- [GiNaC](#)

### Functions

- void [GiNaC::compile\\_ex](#) (const `ex` &expr, const `symbol` &sym, `FUNC_P_1P` &fp, const `std::string` filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const `ex` &expr, const `symbol` &sym1, const `symbol` &sym2, `FUNC_P_2P` &fp, const `std::string` filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const `lst` &exprs, const `lst` &symbols, `FUNC_P_CUBA` &fp, const `std::string` filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::link\\_ex](#) (const `std::string` filename, `FUNC_P_1P` &fp)  
*Opens an existing so-file and returns a function pointer of type `FUNC_P_1P` to the contained function.*
- void [GiNaC::link\\_ex](#) (const `std::string` filename, `FUNC_P_2P` &fp)  
*Opens an existing so-file and returns a function pointer of type `FUNC_P_2P` to the contained function.*
- void [GiNaC::link\\_ex](#) (const `std::string` filename, `FUNC_P_CUBA` &fp)  
*Opens an existing so-file and returns a function pointer of type `FUNC_P_CUBA` to the contained function.*
- void [GiNaC::unlink\\_ex](#) (const `std::string` filename)  
*Closes all linked .so files that have the supplied filename.*

### 7.20.1 Detailed Description

Functions to facilitate the conversion of a `ex` to a function pointer suited for fast numerical integration.

## 7.21 excompiler.h File Reference

Functions to facilitate the conversion of a `ex` to a function pointer suited for fast numerical integration.

```
#include "lst.h"
#include <string>
```

## Namespaces

- [GiNaC](#)

## Typedefs

- typedef double(\* [GiNaC::FUNCP\\_1P](#)) (double)  
*Function pointer with one function parameter.*
- typedef double(\* [GiNaC::FUNCP\\_2P](#)) (double, double)  
*Function pointer with two function parameters.*
- typedef void(\* [GiNaC::FUNCP\\_CUBA](#)) (const int \*, const double[], const int \*, double[])  
*Function pointer for use with the CUBA library (<http://www.feynarts.de/cuba>).*

## Functions

- void [GiNaC::compile\\_ex](#) (const ex &expr, const symbol &sym, FUNCP\_1P &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const ex &expr, const symbol &sym1, const symbol &sym2, FUNCP\_2P &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const lst &exprs, const lst &syms, FUNCP\_CUBA &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::link\\_ex](#) (const std::string filename, FUNCP\_1P &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_1P to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, FUNCP\_2P &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_2P to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, FUNCP\_CUBA &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_CUBA to the contained function.*
- void [GiNaC::unlink\\_ex](#) (const std::string filename)  
*Closes all linked .so files that have the supplied filename.*

### 7.21.1 Detailed Description

Functions to facilitate the conversion of a ex to a function pointer suited for fast numerical integration.

## 7.22 expair.cpp File Reference

Implementation of expression pairs (building blocks of expairseq).

```
#include "expair.h"
#include "operators.h"
#include <iostream>
```

## Namespaces

- [GiNaC](#)

### 7.22.1 Detailed Description

Implementation of expression pairs (building blocks of `expairseq`).

## 7.23 `expair.h` File Reference

Definition of expression pairs (building blocks of `expairseq`).

```
#include "ex.h"  
#include "numeric.h"  
#include "print.h"
```

### Classes

- class [GiNaC::expair](#)  
*A pair of expressions.*
- struct [GiNaC::expair\\_is\\_less](#)  
*Function object for insertion into third argument of STL's `sort()` etc.*
- struct [GiNaC::expair\\_rest\\_is\\_less](#)  
*Function object not caring about the numerical coefficients for insertion into third argument of STL's `sort()`.*
- struct [GiNaC::expair\\_swap](#)

### Namespaces

- [GiNaC](#)

### Functions

- void [GiNaC::swap](#) (`expair &e1`, `expair &e2`)

### 7.23.1 Detailed Description

Definition of expression pairs (building blocks of `expairseq`).

## 7.24 expairseq.cpp File Reference

Implementation of sequences of expression pairs.

```
#include "expairseq.h"
#include "lst.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "relational.h"
#include "wildcard.h"
#include "archive.h"
#include "operators.h"
#include "utils.h"
#include "hash_seed.h"
#include "indexed.h"
#include "compiler.h"
#include <algorithm>
#include <iostream>
#include <iterator>
#include <memory>
#include <stdexcept>
#include <string>
```

### 7.24.1 Detailed Description

Implementation of sequences of expression pairs.

## 7.25 expairseq.h File Reference

Interface to sequences of expression pairs.

```
#include "expair.h"
#include "indexed.h"
#include <vector>
```

### Classes

- class [GiNaC::expairseq](#)  
*A sequence of class expair.*
- class [GiNaC::make\\_flat\\_inserter](#)  
*Class to handle the renaming of dummy indices.*

### Namespaces

- [GiNaC](#)

## Typedefs

- `typedef std::vector< expair > GiNaC::epvector`  
*expair-vector*
- `typedef epvector::iterator GiNaC::epp`  
*expair-vector pointer*

## Functions

- `epvector * GiNaC::conjugateepvector` (const epvector &)  
*Complex conjugate every element of an epvector.*

### 7.25.1 Detailed Description

Interface to sequences of expression pairs.

## 7.26 exprseq.cpp File Reference

Implementation of [GiNaC](#)'s `exprseq`.

```
#include "exprseq.h"
```

## Namespaces

- [GiNaC](#)

### 7.26.1 Detailed Description

Implementation of [GiNaC](#)'s `exprseq`.

## 7.27 exprseq.h File Reference

Definition of [GiNaC](#)'s `exprseq`.

```
#include "container.h"  
#include <vector>
```

## Namespaces

- [GiNaC](#)

## Typedefs

- typedef container< std::vector > [GiNaC::exprseq](#)

### 7.27.1 Detailed Description

Definition of [GiNaC](#)'s exprseq.

## 7.28 factor.cpp File Reference

Polynomial factorization (implementation).

```
#include "factor.h"
#include "ex.h"
#include "numeric.h"
#include "operators.h"
#include "inifcns.h"
#include "symbol.h"
#include "relational.h"
#include "power.h"
#include "mul.h"
#include "normal.h"
#include "add.h"
#include <algorithm>
#include <limits>
#include <list>
#include <vector>
#include <stack>
#include <cln/cln.h>
```

## Namespaces

- [GiNaC](#)

## Macros

- #define [DCOUT](#)(str)
- #define [DCOUTVAR](#)(var)
- #define [DCOUT2](#)(str, var)
- #define [USE\\_SAME\\_DEGREE\\_FACTOR](#)

## Functions

- ex [GiNaC::factor](#) (const ex &poly, unsigned options)  
*Interface function to the outside world.*

### 7.28.1 Detailed Description

Polynomial factorization (implementation).

The interface function `factor()` at the end of this file is defined in the [GiNaC](#) namespace. All other utility functions and classes are defined in an additional anonymous namespace.

Factorization starts by doing a square free factorization and making the coefficients integer. Then, depending on the number of free variables it proceeds either in dedicated univariate or multivariate factorization code.

Univariate factorization does a modular factorization via Berlekamp's algorithm and distinct degree factorization. Hensel lifting is used at the end.

Multivariate factorization uses the univariate factorization (applying a evaluation homomorphism first) and Hensel lifting raises the answer to the multivariate domain. The Hensel lifting code is completely distinct from the code used by the univariate factorization.

Algorithms used can be found in [Wan] An Improved Multivariate Polynomial Factoring Algorithm, P.S.Wang, Mathematics of Computation, Vol. 32, No. 144 (1978) 1215–1231. [GCL] Algorithms for Computer Algebra, K.O.Geddes, S.R.Czapor, G.Labahn, Springer Verlag, 1992. [Mig] Some Useful Bounds, M.Mignotte, In "Computer Algebra, Symbolic and Algebraic Computation" (B.Buchberger et al., eds.), pp. 259-263, Springer-Verlag, New York, 1982.

### 7.28.2 Macro Definition Documentation

#### 7.28.2.1 DCOUT

```
#define DCOUT(  
    str )
```

#### 7.28.2.2 DCOUTVAR

```
#define DCOUTVAR(  
    var )
```

#### 7.28.2.3 DCOUT2

```
#define DCOUT2(  
    str,  
    var )
```



## 7.28.2.4 USE\_SAME\_DEGREE\_FACTOR

```
#define USE_SAME_DEGREE_FACTOR
```

## 7.28.3 Variable Documentation

## 7.28.3.1 value

```
const bool value = false [static]
```

Referenced by `GiNaC::archive_node::add_bool()`, `GiNaC::archive_node::add_ex()`, `GiNaC::archive_node::add_string()`, `GiNaC::archive_node::add_unsigned()`, `GiNaC::Li2_()`, `GiNaC::matrix::set()`, and `GiNaC::subvalue()`.

## 7.28.3.2 r

```
size_t r [private]
```

Referenced by `GiNaC::matrix::charpoly()`, `GiNaC::collect_common_factors()`, `GiNaC::ex::content()`, `GiNaC::numeric::csgn()`, `GiNaC::numeric::denom()`, `GiNaC::matrix::determinant()`, `GiNaC::matrix::determinant_minor()`, `GiNaC::divide()`, `GiNaC::divide_in_z()`, `GiNaC::matrix::division_free_elimination()`, `GiNaC::matrix::echelon_form()`, `GiNaC::power::eval()`, `GiNaC::power::expand()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_add_2()`, `GiNaC::matrix::fraction_free_elimination()`, `GiNaC::matrix::gauss_elimination()`, `GiNaC::idx_symmetrization()`, `GiNaC::matrix::inverse()`, `GiNaC::iquo()`, `GiNaC::lsolve()`, `GiNaC::matrix::markowitz_elimination()`, `GiNaC::matrix::matrix()`, `GiNaC::matrix::mul()`, `GiNaC::matrix::mul_scalar()`, `GiNaC::numeric::numer()`, `GiNaC::Order_series()`, `GiNaC::matrix::pow()`, `GiNaC::prem()`, `GiNaC::numeric::print_numeric()`, `GiNaC::quo()`, `GiNaC::matrix::rank()`, `GiNaC::reduced_matrix()`, `GiNaC::rem()`, `GiNaC::structure< T, ComparisonPolicy >::return_type_tinfo()`, `GiNaC::symbol::series()`, `GiNaC::pseries::series()`, `GiNaC::add::series()`, `GiNaC::integration_kernel::series()`, `GiNaC::mul::series()`, `GiNaC::fderivative::series()`, `GiNaC::integral::series()`, `GiNaC::power::series()`, `GiNaC::structure< T, ComparisonPolicy >::series()`, `GiNaC::ex::series()`, `GiNaC::basic::series()`, `GiNaC::Eisenstein_kernel::series()`, `GiNaC::modular_form_kernel::series()`, `GiNaC::function::series()`, `GiNaC::series()`, `GiNaC::refcounted::set_refcount()`, `GiNaC::simplify_indexed_product()`, `GiNaC::matrix::solve()`, `GiNaC::sprem()`, `GiNaC::sr_gcd()`, `GiNaC::numeric::step()`, `GiNaC::sub_matrix()`, `GiNaC::matrix::subs()`, `GiNaC::ex::subs()`, `GiNaC::symbolic_matrix()`, `GiNaC::matrix::trace()`, `GiNaC::matrix::transpose()`, `GiNaC::unit_matrix()`, and `GiNaC::zeta1_evalf()`.

## 7.28.3.3 c

```
size_t c [private]
```

Referenced by `GiNaC::abs_print_csrc_float()`, `GiNaC::abs_print_latex()`, `GiNaC::symmetry::add()`, `GiNaC::bernoulli()`, `GiNaC::matrix::charpoly()`, `GiNaC::mul::coeff()`, `GiNaC::ncmul::coeff()`, `GiNaC::color_d()`, `GiNaC::color_f()`, `GiNaC::color_h()`, `GiNaC::add::combine_ex_with_coeff_to_pair()`, `GiNaC::mul::combine_ex_with_coeff_to_pair()`, `GiNaC::mul::combine_overall_coeff()`, `GiNaC::add::combine_pair_with_coeff_to_pair()`, `GiNaC::mul::combine_pair_with_coeff_to_pair()`, `GiNaC::mul::conjugate()`, `GiNaC::conjugate_print_latex()`, `GiNaC::ex::content()`, `GiNaC::crc32()`, `GiNaC::pseries::derivative()`, `GiNaC::matrix::determinant()`, `GiNaC::matrix::determinant_minor()`, `GiNaC::divide_in_z()`, `GiNaC::matrix::division_free_elimination()`, `GiNaC::wildcard::do_print()`, `GiNaC::constant::do_print()`, `GiNaC::integral::do_print()`, `GiNaC::relational::do_print()`, `GiNaC::fderivative::do_print()`, `GiNaC::symbol::do_print()`, `GiNaC::ncmul::do_print()`, `GiNaC::add::do_print()`, `GiNaC::integration_kernel::do_print()`, `GiNaC::symmetry::do_print()`, `GiNaC::mul::do_print()`, `GiNaC::idx::do_print()`, `GiNaC::pseries::do_print()`, `GiNaC::matrix::do_print()`, `GiNaC::basic_log_kernel::do_print()`, `GiNaC::varidx::do_print()`, `GiNaC::multiple_polylog_kernel::do_print()`, `GiNaC::numeric::do_print()`, `GiNaC::indexed::do_print()`, `GiNaC::spinidx::do_print()`, `GiNaC::ELi_kernel::do_print()`, `GiNaC::container< C >::do_print()`, `GiNaC::Ebar_kernel::do_print()`, `GiNaC::basic::do_print()`, `GiNaC::Kronecker_dtau_kernel::do_print()`, `GiNaC::Kronecker_dz_kernel::do_print()`, `GiNaC::Eisenstein_kernel::do_print()`, `GiNaC::Eisenstein_h_kernel::do_print()`, `GiNaC::modular_form_kernel::do_print()`, `GiNaC::user_defined_kernel::do_print()`, `GiNaC::fderivative::do_print_csrc()`, `GiNaC::ncmul::do_print_csrc()`, `GiNaC::add::do_print_csrc()`, `GiNaC::power::do_print_csrc()`, `GiNaC::idx::do_print_csrc()`, `GiNaC::mul::do_print_csrc()`, `GiNaC::numeric::do_print_csrc()`, `GiNaC::power::do_print_csrc_cl_N()`, `GiNaC::numeric::do_print_csrc_cl_N()`, `GiNaC::clifford::do_print_dflt()`, `GiNaC::power::do_print_dflt()`, `GiNaC::constant::do_print_latex()`, `GiNaC::integral::do_print_latex()`, `GiNaC::fderivative::do_print_latex()`, `GiNaC::clifford::do_print_latex()`, `GiNaC::symbol::do_print_latex()`, `GiNaC::add::do_print_latex()`, `GiNaC::power::do_print_latex()`, `GiNaC::mul::do_print_latex()`, `GiNaC::idx::do_print_latex()`, `GiNaC::pseries::do_print_latex()`, `GiNaC::matrix::do_print_latex()`, `GiNaC::numeric::do_print_latex()`, `GiNaC::indexed::do_print_latex()`, `GiNaC::spinidx::do_print_latex()`, `GiNaC::power::do_print_python()`, `GiNaC::pseries::do_print_python()`, `GiNaC::container< C >::do_print_python()`, `GiNaC::wildcard::do_print_python_repr()`, `GiNaC::constant::do_print_python_repr()`, `GiNaC::relational::do_print_python_repr()`, `GiNaC::symbol::do_print_python_repr()`, `GiNaC::add::do_print_python_repr()`, `GiNaC::power::do_print_python_repr()`, `GiNaC::mul::do_print_python_repr()`, `GiNaC::matrix::do_print_python_repr()`, `GiNaC::pseries::do_print_python_repr()`, `GiNaC::numeric::do_print_python_repr()`, `GiNaC::container< C >::do_print_python_repr()`, `GiNaC::basic::do_print_python_repr()`, `GiNaC::wildcard::do_print_tree()`, `GiNaC::constant::do_print_tree()`, `GiNaC::clifford::do_print_tree()`, `GiNaC::fderivative::do_print_tree()`, `GiNaC::symbol::do_print_tree()`, `GiNaC::symmetry::do_print_tree()`, `GiNaC::idx::do_print_tree()`, `GiNaC::pseries::do_print_tree()`, `GiNaC::varidx::do_print_tree()`, `GiNaC::numeric::do_print_tree()`, `GiNaC::indexed::do_print_tree()`, `GiNaC::spinidx::do_print_tree()`, `GiNaC::container< C >::do_print_tree()`, `GiNaC::basic::do_print_tree()`, `GiNaC::matrix::echelon_form()`, `GiNaC::EllipticE_print_latex()`, `GiNaC::EllipticK_print_latex()`, `GiNaC::mul::eval()`, `GiNaC::power::eval()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_add_2()`, `GiNaC::factorial_print_dflt_latex()`, `GiNaC::matrix::fraction_free_elimination()`, `GiNaC::matrix::gauss_elimination()`, `GiNaC::H_print_latex()`, `GiNaC::power::imag_part()`, `GiNaC::imag_part_print_latex()`, `GiNaC::add::integer_content()`, `GiNaC::matrix::inverse()`, `GiNaC::lcmcoeff()`, `GiNaC::Li_print_latex()`, `GiNaC::lsolve()`, `GiNaC::matrix::markowitz_elimination()`, `GiNaC::matrix::matrix()`, `GiNaC::matrix::mul()`, `GiNaC::matrix::mul_scalar()`, `GiNaC::error_and_integral_is_less::operator()`, `GiNaC::print_ptrfun_handler< T, C >::operator()`, `GiNaC::print_memfun_handler< T, C >::operator()`, `GiNaC::print_funcutor::operator()`, `GiNaC::matrix::pivot()`, `GiNaC::pseries::power_const()`, `GiNaC::ex::primpart()`, `GiNaC::fderivative::print()`, `GiNaC::expair::print()`, `GiNaC::ex::print()`, `GiNaC::structure< T, ComparisonPolicy >::print()`, `GiNaC::basic::print()`, `GiNaC::function::print()`, `GiNaC::add::print_add()`, `GiNaC::basic::print_dispatch()`, `GiNaC::matrix::print_elements()`, `GiNaC::idx::print_index()`, `GiNaC::indexed::print_indexed()`, `GiNaC::print_integer_csrc()`, `GiNaC::numeric::print_numeric()`, `GiNaC::print_operator()`, `GiNaC::mul::print_overall_coeff()`, `GiNaC::power::print_power()`, `GiNaC::print_real_cl_N()`, `GiNaC::print_real_csrc()`, `GiNaC::print_real_number()`, `GiNaC::pseries::print_series()`, `GiNaC::print_sym_pow()`, `GiNaC::indexed::printindices()`, `GiNaC::container< C >::printseq()`, `GiNaC::numeric::read_archive()`, `GiNaC::power::real_part()`, `GiNaC::real_part_print_latex()`, `GiNaC::reduced_matrix()`, `GiNaC::S_print_latex()`, `GiNaC::set_print_context()`, `GiNaC::matrix::solve()`, `GiNaC::sr_gcd()`, `GiNaC::sub_matrix()`, `GiNaC::matrix::subs()`, `GiNaC::clifford::subs()`, `GiNaC::symbolic_matrix()`, `GiNaC::matrix::transpose()`, `GiNaC::ex::unit()`, `GiNaC::unit_matrix()`, `GiNaC::ex::unitcontprim()`, `GiNaC::zeta1_print_latex()`, and `GiNaC::zeta2_print_latex()`.

## 7.28.3.4 m

```
mvec m [private]
```

Referenced by `GiNaC::mul::algebraic_subs_mul()`, `GiNaC::charpoly()`, `GiNaC::Eisenstein_h_kernel::coefficient↵_an()`, `GiNaC::cols()`, `GiNaC::convert_H_to_Li()`, `GiNaC::determinant()`, `GiNaC::power::eval()`, `GiNaC::tensdelta↵::eval_indexed()`, `GiNaC::tensmetric::eval_indexed()`, `GiNaC::evalf()`, `GiNaC::add::evalm()`, `GiNaC::mul::evalm()`, `GiNaC::power::expand()`, `GiNaC::expand()`, `GiNaC::power::expand_mul()`, `GiNaC::fibonacci()`, `GiNaC::H_deriv()`, `GiNaC::H_eval()`, `GiNaC::H_evalf()`, `GiNaC::H_print_latex()`, `GiNaC::H_series()`, `GiNaC::inverse()`, `GiNaC↵::lgamma_series()`, `GiNaC::Li_deriv()`, `GiNaC::Li_eval()`, `GiNaC::Li_evalf()`, `GiNaC::Li_print_latex()`, `GiNaC::Li↵_series()`, `GiNaC::nops()`, `GiNaC::Order_eval()`, `GiNaC::psi1_series()`, `GiNaC::psi2_eval()`, `GiNaC::psi2_series()`, `GiNaC::rank()`, `GiNaC::reduced_matrix()`, `GiNaC::reposition_dummy_indices()`, `GiNaC::resultant()`, `GiNaC::rows()`, `GiNaC::S_eval()`, `GiNaC::smod()`, `GiNaC::sub_matrix()`, `GiNaC::symbol::subs()`, `GiNaC::idx::subs()`, `GiNaC↵::pseries::subs()`, `GiNaC::relational::subs()`, `GiNaC::power::subs()`, `GiNaC::clifford::subs()`, `GiNaC::numeric::subs()`, `GiNaC::container< C >::subs()`, `GiNaC::ex::subs()`, `GiNaC::structure< T, ComparisonPolicy >::subs()`, `Gi↵NaC::basic::subs()`, `GiNaC::subs()`, `GiNaC::basic::subs_one_level()`, `GiNaC::container< C >::subschilchildren()`, `GiNaC::tgamma_series()`, `GiNaC::trace()`, `GiNaC::transpose()`, `GiNaC::zeta1_deriv()`, `GiNaC::zeta1_eval()`, `GiNa↵C::zeta1_print_latex()`, `GiNaC::zeta2_deriv()`, `GiNaC::zeta2_eval()`, and `GiNaC::zeta2_print_latex()`.

## 7.28.3.5 lr

```
umodpoly lr[2] [private]
```

Referenced by `GiNaC::ex::subs()`, and `GiNaC::subs()`.

## 7.28.3.6 cache

```
vector<vector<umodpoly> > cache [private]
```

## 7.28.3.7 factors

```
upvec factors [private]
```

Referenced by `GiNaC::ncmul::count_factors()`, `GiNaC::ncmul::eval()`, `GiNaC::mul::expand()`, `GiNaC::mul::mul()`, `GiNaC::sqrfree()`, and `GiNaC::sqrfree_yun()`.

## 7.28.3.8 one

```
umodpoly one [private]
```

Referenced by `GiNaC::integration_kernel::get_numerical_value_impl()`, and `GiNaC::iterated_integral_evalf_impl()`.

## 7.28.3.9 n

```
size_t n [private]
```

Referenced by `GiNaC::class_info< OPT >::tree_node::add_child()`, `GiNaC::archive::add_node()`, `GiNaC::wildcard::archive()`, `GiNaC::color::archive()`, `GiNaC::clifford::archive()`, `GiNaC::constant::archive()`, `GiNaC::idx::archive()`, `GiNaC::integral::archive()`, `GiNaC::symbol::archive()`, `GiNaC::relational::archive()`, `GiNaC::pseries::archive()`, `GiNaC::matrix::archive()`, `GiNaC::fderivative::archive()`, `GiNaC::symmetry::archive()`, `GiNaC::power::archive()`, `GiNaC::minkmetric::archive()`, `GiNaC::numeric::archive()`, `GiNaC::varidx::archive()`, `GiNaC::container< C >::archive()`, `GiNaC::indexed::archive()`, `GiNaC::tensepsilon::archive()`, `GiNaC::spinidx::archive()`, `GiNaC::basic::archive()`, `GiNaC::archive::archive()`, `GiNaC::function::archive()`, `GiNaC::bernoulli()`, `GiNaC::binomial()`, `GiNaC::pseries::coeff()`, `GiNaC::add::coeff()`, `GiNaC::mul::coeff()`, `GiNaC::power::coeff()`, `GiNaC::ncmul::coeff()`, `GiNaC::numeric::coeff()`, `GiNaC::structure< T, ComparisonPolicy >::coeff()`, `GiNaC::ex::coeff()`, `GiNaC::basic::coeff()`, `GiNaC::coeff()`, `GiNaC::Eisenstein_h_kernel::coefficient_an()`, `GiNaC::basic::collect()`, `GiNaC::const_postorder_iterator::const_postorder_iterator()`, `GiNaC::const_preorder_iterator::const_preorder_iterator()`, `GiNaC::composition_generator::coolmulti::coolmulti()`, `GiNaC::matrix::determinant_minor()`, `GiNaC::dirichlet_character()`, `GiNaC::matrix::division_free_elimination()`, `GiNaC::doublefactorial()`, `GiNaC::class_info< OPT >::dump_tree()`, `GiNaC::matrix::echelon_form()`, `GiNaC::power::eval()`, `GiNaC::mul::expand()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_mul()`, `GiNaC::factorial()`, `GiNaC::fibonacci()`, `GiNaC::frac_cancel()`, `GiNaC::matrix::fraction_free_elimination()`, `GiNaC::function_options::function_options()`, `GiNaC::matrix::gauss_elimination()`, `GiNaC::gcd()`, `GiNaC::golden_ratio_hash()`, `GiNaC::H_eval()`, `GiNaC::ifactor()`, `GiNaC::power::imag_part()`, `GiNaC::is_discriminant_of_quadratic_number_field()`, `GiNaC::kronecker_symbol()`, `GiNaC::integration_kernel::Laurent_series()`, `GiNaC::log2()`, `GiNaC::log_series()`, `GiNaC::basic::map()`, `GiNaC::matrix::markowitz_elimination()`, `GiNaC::multinomial_coefficient()`, `GiNaC::pseries::normal()`, `GiNaC::add::normal()`, `GiNaC::mul::normal()`, `GiNaC::const_iterator::operator+()`, `GiNaC::const_iterator::operator+=()`, `GiNaC::const_iterator::operator-()`, `GiNaC::const_iterator::operator=()`, `GiNaC::operator<<()`, `GiNaC::operator>>()`, `GiNaC::const_iterator::operator[]()`, `GiNaC::primitive_dirichlet_character()`, `GiNaC::psi2_deriv()`, `GiNaC::psi2_eval()`, `GiNaC::psi2_evalf()`, `GiNaC::psi2_series()`, `GiNaC::wildcard::read_archive()`, `GiNaC::color::read_archive()`, `GiNaC::clifford::read_archive()`, `GiNaC::constant::read_archive()`, `GiNaC::idx::read_archive()`, `GiNaC::integral::read_archive()`, `GiNaC::symbol::read_archive()`, `GiNaC::relational::read_archive()`, `GiNaC::pseries::read_archive()`, `GiNaC::fderivative::read_archive()`, `GiNaC::matrix::read_archive()`, `GiNaC::symmetry::read_archive()`, `GiNaC::power::read_archive()`, `GiNaC::minkmetric::read_archive()`, `GiNaC::numeric::read_archive()`, `GiNaC::container< C >::read_archive()`, `GiNaC::varidx::read_archive()`, `GiNaC::indexed::read_archive()`, `GiNaC::tensepsilon::read_archive()`, `GiNaC::spinidx::read_archive()`, `GiNaC::function::read_archive()`, `GiNaC::power::real_part()`, `GiNaC::container_storage< C >::reserve()`, `GiNaC::rotate_left()`, `GiNaC::S_deriv()`, `GiNaC::S_eval()`, `GiNaC::S_evalf()`, `GiNaC::S_print_latex()`, `GiNaC::S_series()`, `GiNaC::basic::series()`, `GiNaC::symbol::set_name()`, `GiNaC::function_options::set_name()`, `GiNaC::symbol::set_TeX_name()`, `GiNaC::matrix::solve()`, `GiNaC::function_options::test_and_set_nparams()`, `GiNaC::tgamma_eval()`, `GiNaC::ex::traverse_postorder()`, `GiNaC::ex::traverse_preorder()`, `GiNaC::symmetry::validate()`, `GiNaC::write_real_float()`, `GiNaC::zetaderiv_deriv()`, and `GiNaC::zetaderiv_eval()`.

## 7.28.3.10 len

```
size_t len [private]
```

Referenced by `GiNaC::crc32()`.

## 7.28.3.11 last

```
size_t last [private]
```

Referenced by `GiNaC::antisymmetrize()`, `GiNaC::canonicalize()`, `GiNaC::cyclic_permutation()`, `GiNaC::mul::eval()`, `GiNaC::power::expand_add_2()`, `GiNaC::mul::expandchildren()`, `GiNaC::find_free_and_dummy()`, `GiNaC::permutation_sign()`, `GiNaC::shaker_sort()`, `GiNaC::symm()`, `GiNaC::symmetrize()`, and `GiNaC::symmetrize_cyclic()`.

## 7.28.3.12 k

```
vector<int> k [private]
```

Referenced by GiNaC::bernoulli(), GiNaC::Bernoulli\_polynomial(), GiNaC::binomial(), GiNaC::scalar\_products(), GiNaC::debugprint(), GiNaC::divide\_in\_z(), GiNaC::EllipticE\_deriv(), GiNaC::EllipticE\_eval(), GiNaC::EllipticE\_evalf(), GiNaC::EllipticE\_print\_latex(), GiNaC::EllipticE\_series(), GiNaC::EllipticK\_deriv(), GiNaC::EllipticK\_eval(), GiNaC::EllipticK\_evalf(), GiNaC::EllipticK\_print\_latex(), GiNaC::EllipticK\_series(), GiNaC::ncmul::expand(), GiNaC::power::expand\_add(), GiNaC::find\_common\_factor(), GiNaC::generalised\_Bernoulli\_number(), GiNaC::multi\_iterator\_permutation< T >::get\_sign(), GiNaC::log2(), GiNaC::matrix::markowitz\_elimination(), GiNaC::basic\_partition\_generator::mpartition2::mpartition2(), GiNaC::basic\_partition\_generator::mpartition2::next\_partition(), GiNaC::composition\_generator::coolmulti::next\_permutation(), GiNaC::multi\_iterator\_ordered< T >::operator++(), GiNaC::multi\_iterator\_ordered\_eq< T >::operator++(), GiNaC::multi\_iterator\_ordered\_eq\_indv< T >::operator++(), GiNaC::multi\_iterator\_counter< T >::operator++(), GiNaC::multi\_iterator\_counter\_indv< T >::operator++(), GiNaC::multi\_iterator\_permutation< T >::operator++(), GiNaC::multi\_iterator\_shuffle< T >::operator++(), GiNaC::matrix::pivot(), GiNaC::reposition\_dummy\_indices(), GiNaC::resultant(), GiNaC::ELi\_kernel::series\_coeff\_impl(), GiNaC::Ebar\_kernel::series\_coeff\_impl(), GiNaC::simplify\_indexed(), and GiNaC::sqrfree\_pfrac().

## 7.28.3.13 poly

```
upoly poly
```

Referenced by GiNaC::matrix::charpoly(), and GiNaC::factor().

## 7.28.3.14 x

```
ex x
```

Referenced by GiNaC::abs(), GiNaC::acos(), GiNaC::acos\_conjugate(), GiNaC::acos\_deriv(), GiNaC::acos\_eval(), GiNaC::acos\_evalf(), GiNaC::acosh(), GiNaC::acosh\_conjugate(), GiNaC::acosh\_deriv(), GiNaC::acosh\_eval(), GiNaC::acosh\_evalf(), GiNaC::adaptivesimpson(), GiNaC::asin(), GiNaC::asin\_conjugate(), GiNaC::asin\_deriv(), GiNaC::asin\_eval(), GiNaC::asin\_evalf(), GiNaC::asinh(), GiNaC::asinh\_conjugate(), GiNaC::asinh\_deriv(), GiNaC::asinh\_eval(), GiNaC::asinh\_evalf(), GiNaC::atan(), GiNaC::atan2\_deriv(), GiNaC::atan2\_eval(), GiNaC::atan2\_evalf(), GiNaC::atan\_conjugate(), GiNaC::atan\_deriv(), GiNaC::atan\_eval(), GiNaC::atan\_evalf(), GiNaC::atanh(), GiNaC::atanh\_conjugate(), GiNaC::atanh\_deriv(), GiNaC::atanh\_eval(), GiNaC::atanh\_evalf(), GiNaC::beta\_deriv(), GiNaC::beta\_eval(), GiNaC::beta\_evalf(), GiNaC::binomial\_conjugate(), GiNaC::binomial\_eval(), GiNaC::binomial\_evalf(), GiNaC::binomial\_real\_part(), GiNaC::binomial\_sym(), GiNaC::lanczos\_coeffs::calc\_lanczos\_A(), GiNaC::basic::collect(), GiNaC::matrix::conjugate(), GiNaC::mul::conjugate(), GiNaC::container< C >::conjugate(), GiNaC::ex::content(), GiNaC::convert\_H\_to\_Li(), GiNaC::cos(), GiNaC::cos\_conjugate(), GiNaC::cos\_deriv(), GiNaC::cos\_eval(), GiNaC::cos\_evalf(), GiNaC::cos\_imag\_part(), GiNaC::cos\_real\_part(), GiNaC::cosh(), GiNaC::cosh\_conjugate(), GiNaC::cosh\_deriv(), GiNaC::cosh\_eval(), GiNaC::cosh\_evalf(), GiNaC::cosh\_imag\_part(), GiNaC::cosh\_real\_part(), GiNaC::csgn(), GiNaC::decomp\_rational(), GiNaC::denom(), GiNaC::divide(), GiNaC::divide\_in\_z(), GiNaC::eta\_conjugate(), GiNaC::eta\_eval(), GiNaC::eta\_evalf(), GiNaC::eta\_imag\_part(), GiNaC::eta\_series(), GiNaC::tensepsilon::eval\_indexed(), GiNaC::exp(), GiNaC::exp\_conjugate(), GiNaC::exp\_deriv(), GiNaC::exp\_eval(), GiNaC::exp\_evalf(), GiNaC::exp\_imag\_part(), GiNaC::exp\_power(), GiNaC::exp\_real\_part(), GiNaC::factorial\_conjugate(), GiNaC::factorial\_eval(), GiNaC::factorial\_evalf(), GiNaC::factorial\_print\_dflt\_latex(), GiNaC::factorial\_real\_part(), GiNaC::find\_common\_factor(), GiNaC::frac\_cancel(), GiNaC::fsolve(), GiNaC::G(), GiNaC::G2\_eval(), GiNaC::G2\_evalf(), GiNaC::G3\_eval(), GiNaC::G3\_evalf(), GiNaC::gcd(), GiNaC::get\_first\_symbol(), GiNaC::guess\_precision(), GiNaC::H\_deriv(), GiNaC::H\_eval(), GiNaC::H\_evalf(), GiNaC::H\_print\_latex(), GiNaC::H\_series(), GiNaC::make\_flat\_inserter::handle\_factor(), GiNaC::hasindex(), GiNaC::haswild(),

GiNaC::heur\_gcd\_z(), GiNaC::imag(), GiNaC::interpolate(), GiNaC::inverse(), GiNaC::is\_cinteger(), GiNaC::is\_↵  
 crational(), GiNaC::is\_even(), GiNaC::is\_integer(), GiNaC::is\_negative(), GiNaC::is\_nonneg\_integer(), GiNaC::is\_↵  
 \_odd(), GiNaC::is\_pos\_integer(), GiNaC::is\_positive(), GiNaC::is\_prime(), GiNaC::is\_rational(), GiNaC::is\_real(),  
 GiNaC::is\_the\_function(), GiNaC::is\_the\_function< G\_SERIAL >(), GiNaC::is\_the\_function< iterated\_integral↵  
 \_SERIAL >(), GiNaC::is\_the\_function< psi\_SERIAL >(), GiNaC::is\_the\_function< zeta\_SERIAL >(), GiNaC::is\_↵  
 zero(), GiNaC::isqrt(), GiNaC::lgamma(), GiNaC::lgamma\_conjugate(), GiNaC::lgamma\_deriv(), GiNaC::↵  
 lgamma\_eval(), GiNaC::lgamma\_evalf(), GiNaC::Li2(), GiNaC::Li2\_conjugate(), GiNaC::Li2\_deriv(), GiNaC::↵  
 Li2\_eval(), GiNaC::Li2\_evalf(), GiNaC::Li2\_projection(), GiNaC::Li2\_series(), GiNaC::Li3\_eval(), GiNaC::Li\_↵  
 deriv(), GiNaC::Li\_eval(), GiNaC::Li\_evalf(), GiNaC::Li\_print\_latex(), GiNaC::Li\_series(), GiNaC::log(), GiNaC::log\_↵  
 conjugate(), GiNaC::log\_deriv(), GiNaC::log\_eval(), GiNaC::log\_evalf(), GiNaC::log\_imag\_part(), GiNaC::log\_↵  
 real\_part(), GiNaC::make\_real\_float(), GiNaC::matrix::matrix(), GiNaC::numer(), GiNaC::sym\_desc::operator<(),  
 GiNaC::Order\_conjugate(), GiNaC::Order\_eval(), GiNaC::Order\_imag\_part(), GiNaC::Order\_real\_part(), GiNaC::↵  
 Order\_series(), GiNaC::pow(), GiNaC::prem(), GiNaC::ex::primpart(), GiNaC::print\_integer\_csrc(), GiNaC::↵  
 print\_real\_cl\_N(), GiNaC::print\_real\_csrc(), GiNaC::print\_real\_number(), GiNaC::print\_sym\_pow(), GiNaC::psi1\_↵  
 \_deriv(), GiNaC::psi1\_eval(), GiNaC::psi1\_evalf(), GiNaC::psi2\_deriv(), GiNaC::psi2\_eval(), GiNaC::psi2\_evalf(),  
 GiNaC::quo(), GiNaC::read\_real\_float(), GiNaC::real(), GiNaC::rem(), GiNaC::S\_deriv(), GiNaC::S\_eval(), GiNaC::↵  
 S\_evalf(), GiNaC::S\_print\_latex(), GiNaC::S\_series(), GiNaC::sin(), GiNaC::sin\_conjugate(), GiNaC::sin\_deriv(),  
 GiNaC::sin\_eval(), GiNaC::sin\_evalf(), GiNaC::sin\_imag\_part(), GiNaC::sin\_real\_part(), GiNaC::sinh(), GiNaC::↵  
 sinh\_conjugate(), GiNaC::sinh\_deriv(), GiNaC::sinh\_eval(), GiNaC::sinh\_evalf(), GiNaC::sinh\_imag\_part(), Gi↵  
 NaC::sinh\_real\_part(), GiNaC::sprem(), GiNaC::sqrfree(), GiNaC::sqrfree\_parfrac(), GiNaC::sqrfree\_yun(), GiNaC::↵  
 sqrt(), GiNaC::sr\_gcd(), GiNaC::step(), GiNaC::tan(), GiNaC::tan\_conjugate(), GiNaC::tan\_deriv(), GiNaC::tan\_↵  
 \_eval(), GiNaC::tan\_evalf(), GiNaC::tan\_imag\_part(), GiNaC::tan\_real\_part(), GiNaC::tan\_series(), GiNaC::tanh(),  
 GiNaC::tanh\_conjugate(), GiNaC::tanh\_deriv(), GiNaC::tanh\_eval(), GiNaC::tanh\_evalf(), GiNaC::tanh\_imag\_↵  
 part(), GiNaC::tanh\_real\_part(), GiNaC::tanh\_series(), GiNaC::tgamma(), GiNaC::tgamma\_conjugate(), GiNaC::↵  
 tgamma\_deriv(), GiNaC::tgamma\_eval(), GiNaC::tgamma\_evalf(), GiNaC::to\_double(), GiNaC::to\_int(), GiNaC::↵  
 to\_long(), GiNaC::ex::unit(), GiNaC::unit\_matrix(), GiNaC::ex::unitcontprim(), GiNaC::zeta(), GiNaC::zeta1\_evalf(),  
 GiNaC::zeta2\_evalf(), GiNaC::zetaderiv\_deriv(), and GiNaC::zetaderiv\_eval().

### 7.28.3.15 evalpoint

```
int evalpoint
```

### 7.28.3.16 R

```
cl_modint_ring R
```

### 7.28.3.17 syms

```
exset syms
```

Referenced by GiNaC::lsolve().

## 7.28.3.18 options

unsigned options

Referenced by `GiNaC::abs_expand()`, `GiNaC::mul::algebraic_subs_mul()`, `GiNaC::atan_series()`, `GiNaC::atanh_series()`, `GiNaC::beta_series()`, `GiNaC::csgn_series()`, `GiNaC::determinant()`, `GiNaC::exp_expand()`, `GiNaC::integral::expand()`, `GiNaC::pseries::expand()`, `GiNaC::ncmul::expand()`, `GiNaC::add::expand()`, `GiNaC::power::expand()`, `GiNaC::mul::expand()`, `GiNaC::indexed::expand()`, `GiNaC::structure< T, ComparisonPolicy >::expand()`, `GiNaC::ex::expand()`, `GiNaC::basic::expand()`, `GiNaC::function::expand()`, `GiNaC::expand()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_add_2()`, `GiNaC::power::expand_mul()`, `GiNaC::ncmul::expandchildren()`, `GiNaC::mul::expandchildren()`, `GiNaC::factor()`, `GiNaC::gcd()`, `GiNaC::mul::has()`, `GiNaC::power::has()`, `GiNaC::ex::has()`, `GiNaC::structure< T, ComparisonPolicy >::has()`, `GiNaC::basic::has()`, `GiNaC::has()`, `GiNaC::lgamma_series()`, `GiNaC::Li2_series()`, `GiNaC::log_expand()`, `GiNaC::log_series()`, `GiNaC::lsolve()`, `GiNaC::expand_map_function::operator()`, `GiNaC::registered_class_options::print_func()`, `GiNaC::function_options::print_func()`, `GiNaC::psi1_series()`, `GiNaC::psi2_series()`, `GiNaC::S_series()`, `GiNaC::pseries::series()`, `GiNaC::add::series()`, `GiNaC::mul::series()`, `GiNaC::integral::series()`, `GiNaC::fderivative::series()`, `GiNaC::power::series()`, `GiNaC::structure< T, ComparisonPolicy >::series()`, `GiNaC::ex::series()`, `GiNaC::function::series()`, `GiNaC::series()`, `GiNaC::set_print_context()`, `GiNaC::set_print_func()`, `GiNaC::set_print_options()`, `GiNaC::simplify_indexed()`, `GiNaC::step_series()`, `GiNaC::symbol::subs()`, `GiNaC::idx::subs()`, `GiNaC::matrix::subs()`, `GiNaC::pseries::subs()`, `GiNaC::relational::subs()`, `GiNaC::power::subs()`, `GiNaC::clifford::subs()`, `GiNaC::numeric::subs()`, `GiNaC::container< C >::subs()`, `GiNaC::ex::subs()`, `GiNaC::structure< T, ComparisonPolicy >::subs()`, `GiNaC::basic::subs()`, `GiNaC::subs()`, `GiNaC::basic::subs_one_level()`, `GiNaC::container< C >::subchildren()`, `GiNaC::tan_series()`, `GiNaC::tanh_series()`, and `GiNaC::tgamma_series()`.

## 7.29 factor.h File Reference

Polynomial factorization.

## Namespaces

- [GiNaC](#)

## Functions

- ex [GiNaC::factor](#) (const ex &[poly](#), unsigned [options](#))  
*Interface function to the outside world.*

## 7.29.1 Detailed Description

Polynomial factorization.

## 7.30 fail.cpp File Reference

Implementation of class signaling failure of operation.

```
#include "fail.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
```

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (fail, basic, print\_func< print\_context >(&fail←::do\_print). print\_func< print\_tree >(&fail::do\_print\_tree)) [GINAC\\_BIND\\_UNARCHIVER](#)(fail)

### 7.30.1 Detailed Description

Implementation of class signaling failure of operation.

Considered somewhat obsolete (most of this can be replaced by exceptions).

## 7.31 fail.h File Reference

Interface to class signaling failure of operation.

```
#include "basic.h"
#include "archive.h"
```

## Classes

- class [GiNaC::fail](#)

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (fail)

### 7.31.1 Detailed Description

Interface to class signaling failure of operation.

Considered obsolete somewhat obsolete (most of this can be replaced by exceptions).



## 7.32 fderivative.cpp File Reference

Implementation of abstract derivatives of functions.

```
#include "fderivative.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
```

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (fderivative, function, print\_func< print\_context >(&fderivative::do\_print). print\_func< print\_latex >(&fderivative::do\_print\_latex). print\_func< print\_csrc >(&fderivative::do\_print\_csrc). print\_func< print\_tree >(&fderivative::do\_print\_tree)) fderivative
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (fderivative)

#### 7.32.1 Detailed Description

Implementation of abstract derivatives of functions.

## 7.33 fderivative.h File Reference

Interface to abstract derivatives of functions.

```
#include "function.h"
#include <set>
```

### Classes

- class [GiNaC::fderivative](#)  
*This class represents the (abstract) derivative of a symbolic function.*

### Namespaces

- [GiNaC](#)

### Typedefs

- typedef std::multiset< unsigned > [GiNaC::paramset](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (fderivative)

### 7.33.1 Detailed Description

Interface to abstract derivatives of functions.

## 7.34 flags.h File Reference

Collection of all flags used through the [GiNaC](#) framework.

## Classes

- class [GiNaC::expand\\_options](#)  
*Flags to control the behavior of [expand\(\)](#).*
- class [GiNaC::has\\_options](#)  
*Flags to control the behavior of [has\(\)](#).*
- class [GiNaC::subs\\_options](#)  
*Flags to control the behavior of [subs\(\)](#).*
- class [GiNaC::domain](#)  
*Domain of an object.*
- class [GiNaC::series\\_options](#)  
*Flags to control series expansion.*
- class [GiNaC::determinant\\_algo](#)  
*Switch to control algorithm for determinant computation.*
- class [GiNaC::solve\\_algo](#)  
*Switch to control algorithm for linear system solving.*
- class [GiNaC::status\\_flags](#)  
*Flags to store information about the state of an object.*
- class [GiNaC::info\\_flags](#)  
*Possible attributes an object can have.*
- class [GiNaC::return\\_types](#)
- class [GiNaC::remember\\_strategies](#)  
*Strategies how to clean up the function remember cache.*
- class [GiNaC::factor\\_options](#)  
*Flags to control the polynomial factorization.*

## Namespaces

- [GiNaC](#)

### 7.34.1 Detailed Description

Collection of all flags used through the [GiNaC](#) framework.

## 7.35 function.cpp File Reference

Implementation of class of symbolic functions.

```
#include "function.h"
#include "operators.h"
#include "fderivative.h"
#include "ex.h"
#include "lst.h"
#include "symmetry.h"
#include "print.h"
#include "power.h"
#include "archive.h"
#include "inifcns.h"
#include "utils.h"
#include "hash_seed.h"
#include "remember.h"
#include <iostream>
#include <limits>
#include <list>
#include <stdexcept>
#include <string>
```

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (function)

#### 7.35.1 Detailed Description

Implementation of class of symbolic functions.

## 7.36 function.h File Reference

Interface to class of symbolic functions.

```
#include "exprseq.h"
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::function\\_options](#)
- class [GiNaC::do\\_taylor](#)

*Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.*

- class [GiNaC::function](#)

*The class function is used to implement builtin functions like sin, cos...*

## Namespaces

- [GiNaC](#)

## Macros

- #define [DECLARE\\_FUNCTION\\_1P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_2P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_3P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_4P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_5P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_6P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_7P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_8P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_9P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_10P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_11P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_12P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_13P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_14P](#)(NAME)
- #define [REGISTER\\_FUNCTION](#)(NAME, OPT)
- #define [is\\_ex\\_the\\_function](#)(OBJ, FUNCNAME) ([GiNaC::is\\_the\\_function](#)<FUNCNAME##\_SERIAL>(OBJ))

## Typedefs

- typedef ex(\* [GiNaC::eval\\_funcp](#)) ()
- typedef ex(\* [GiNaC::evalf\\_funcp](#)) ()
- typedef ex(\* [GiNaC::conjugate\\_funcp](#)) ()
- typedef ex(\* [GiNaC::real\\_part\\_funcp](#)) ()
- typedef ex(\* [GiNaC::imag\\_part\\_funcp](#)) ()
- typedef ex(\* [GiNaC::expand\\_funcp](#)) ()
- typedef ex(\* [GiNaC::derivative\\_funcp](#)) ()
- typedef ex(\* [GiNaC::expl\\_derivative\\_funcp](#)) ()
- typedef ex(\* [GiNaC::power\\_funcp](#)) ()
- typedef ex(\* [GiNaC::series\\_funcp](#)) ()
- typedef void(\* [GiNaC::print\\_funcp](#)) ()
- typedef bool(\* [GiNaC::info\\_funcp](#)) ()
- typedef ex(\* [GiNaC::eval\\_funcp\\_1](#)) (const ex &)
- typedef ex(\* [GiNaC::evalf\\_funcp\\_1](#)) (const ex &)
- typedef ex(\* [GiNaC::conjugate\\_funcp\\_1](#)) (const ex &)
- typedef ex(\* [GiNaC::real\\_part\\_funcp\\_1](#)) (const ex &)
- typedef ex(\* [GiNaC::imag\\_part\\_funcp\\_1](#)) (const ex &)
- typedef ex(\* [GiNaC::expand\\_funcp\\_1](#)) (const ex &, unsigned)

- [illegible]



- [illegible]







- [illegible]

- typedef ex(\* [GiNaC::evalf\\_funcp\\_exvector](#)) (const exvector &)
- typedef ex(\* [GiNaC::conjugate\\_funcp\\_exvector](#)) (const exvector &)
- typedef ex(\* [GiNaC::real\\_part\\_funcp\\_exvector](#)) (const exvector &)
- typedef ex(\* [GiNaC::imag\\_part\\_funcp\\_exvector](#)) (const exvector &)
- typedef ex(\* [GiNaC::expand\\_funcp\\_exvector](#)) (const exvector &, unsigned)
- typedef ex(\* [GiNaC::derivative\\_funcp\\_exvector](#)) (const exvector &, unsigned)
- typedef ex(\* [GiNaC::expl\\_derivative\\_funcp\\_exvector](#)) (const exvector &, const symbol &)
- typedef ex(\* [GiNaC::power\\_funcp\\_exvector](#)) (const exvector &, const ex &)
- typedef ex(\* [GiNaC::series\\_funcp\\_exvector](#)) (const exvector &, const relational &, int, unsigned)
- typedef void(\* [GiNaC::print\\_funcp\\_exvector](#)) (const exvector &, const print\_context &)
- typedef bool(\* [GiNaC::info\\_funcp\\_exvector](#)) (const exvector &, unsigned)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (function)
- template<typename T >  
bool [GiNaC::is\\_the\\_function](#) (const ex &x)

### 7.36.1 Detailed Description

Interface to class of symbolic functions.

### 7.36.2 Macro Definition Documentation

#### 7.36.2.1 DECLARE\_FUNCTION\_1P

```
#define DECLARE_FUNCTION_1P (
    NAME )
```

##### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 1; \
template< typename T1 > const GiNaC::function NAME( const T1 & p1 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1) ); \
}
```

#### 7.36.2.2 DECLARE\_FUNCTION\_2P

```
#define DECLARE_FUNCTION_2P (
    NAME )
```

##### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 2; \
template< typename T1, typename T2 > const GiNaC::function NAME( const T1 & p1, const T2 & \
    p2 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), \
        GiNaC::ex(p2) ); \
}
```

## 7.36.2.3 DECLARE\_FUNCTION\_3P

```
#define DECLARE_FUNCTION_3P(  
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 3; \
template< typename T1, typename T2, typename T3 > const GiNaC::function NAME( const T1 & p1,  
    const T2 & p2, const T3 & p3 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),  
        GiNaC::ex(p2), GiNaC::ex(p3) ); \
}
```

## 7.36.2.4 DECLARE\_FUNCTION\_4P

```
#define DECLARE_FUNCTION_4P(  
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 4; \
template< typename T1, typename T2, typename T3, typename T4 > const  
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),  
        GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4) ); \
}
```

## 7.36.2.5 DECLARE\_FUNCTION\_5P

```
#define DECLARE_FUNCTION_5P(  
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 5; \
template< typename T1, typename T2, typename T3, typename T4, typename T5 > const  
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 &  
        p5 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),  
        GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4),  
        GiNaC::ex(p5) ); \
}
```

## 7.36.2.6 DECLARE\_FUNCTION\_6P

```
#define DECLARE_FUNCTION_6P(
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 6; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6 > const
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 &
        p5, const T6 & p6 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),
        GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4),
        GiNaC::ex(p5), GiNaC::ex(p6) ); \
}
```

## 7.36.2.7 DECLARE\_FUNCTION\_7P

```
#define DECLARE_FUNCTION_7P(
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 7; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7 > const
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 &
        p5, const T6 & p6, const T7 & p7 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),
        GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4),
        GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7) ); \
}
```

## 7.36.2.8 DECLARE\_FUNCTION\_8P

```
#define DECLARE_FUNCTION_8P(
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 8; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8 > const GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 &
    p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),
        GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4),
        GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7),
        GiNaC::ex(p8) ); \
}
```

## 7.36.2.9 DECLARE\_FUNCTION\_9P

```
#define DECLARE_FUNCTION_9P(
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 9; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9 > const GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3,
    const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),
    GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4),
    GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7),
    GiNaC::ex(p8), GiNaC::ex(p9) ); \
}
```

## 7.36.2.10 DECLARE\_FUNCTION\_10P

```
#define DECLARE_FUNCTION_10P(
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 10; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9, typename T10 > const GiNaC::function NAME( const T1 & p1, const T2 & p2,
    const T3 & p3, const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9,
    const T10 & p10 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),
    GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4),
    GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7),
    GiNaC::ex(p8), GiNaC::ex(p9), GiNaC::ex(p10) ); \
}
```

## 7.36.2.11 DECLARE\_FUNCTION\_11P

```
#define DECLARE_FUNCTION_11P(
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 11; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9, typename T10, typename T11 > const GiNaC::function NAME( const T1 & p1,
    const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8,
    const T9 & p9, const T10 & p10, const T11 & p11 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),
    GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4),
    GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7),
    GiNaC::ex(p8), GiNaC::ex(p9), GiNaC::ex(p10),
    GiNaC::ex(p11) ); \
}
```

## 7.36.2.12 DECLARE\_FUNCTION\_12P

```
#define DECLARE_FUNCTION_12P(
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 12; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9, typename T10, typename T11, typename T12 > const
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 &
        p5, const T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12
        & p12 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),
        GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4),
        GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7),
        GiNaC::ex(p8), GiNaC::ex(p9), GiNaC::ex(p10),
        GiNaC::ex(p11), GiNaC::ex(p12) ); \
}
```

## 7.36.2.13 DECLARE\_FUNCTION\_13P

```
#define DECLARE_FUNCTION_13P(
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 13; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9, typename T10, typename T11, typename T12, typename T13 > const
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 &
        p5, const T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12
        & p12, const T13 & p13 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),
        GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4),
        GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7),
        GiNaC::ex(p8), GiNaC::ex(p9), GiNaC::ex(p10),
        GiNaC::ex(p11), GiNaC::ex(p12), GiNaC::ex(p13) ); \
}
```

## 7.36.2.14 DECLARE\_FUNCTION\_14P

```
#define DECLARE_FUNCTION_14P(
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 14; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14 > const
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 &
        p5, const T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12
        & p12, const T13 & p13, const T14 & p14 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1),
        GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4),
        GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7),
        GiNaC::ex(p8), GiNaC::ex(p9), GiNaC::ex(p10),
        GiNaC::ex(p11), GiNaC::ex(p12), GiNaC::ex(p13),
        GiNaC::ex(p14) ); \
}
```

## 7.36.2.15 REGISTER\_FUNCTION

```
#define REGISTER_FUNCTION(
    NAME,
    OPT )
```

**Value:**

```
unsigned NAME##_SERIAL::serial = \
    GiNaC::function::register_new(
        GiNaC::function_options( #NAME, NAME##_NPARAMS ).OPT );
```

## 7.36.2.16 is\_ex\_the\_function

```
#define is_ex_the_function(
    OBJ,
    FUNCNAME ) (GiNaC::is_the_function<FUNCNAME##_SERIAL> (OBJ) )
```

Referenced by `GiNaC::abs_eval()`, `GiNaC::cos_eval()`, `GiNaC::cosh_eval()`, `GiNaC::exp_eval()`, `GiNaC::is_order_↵  
function()`, `GiNaC::log_eval()`, `GiNaC::replace_with_symbol()`, `GiNaC::sin_eval()`, `GiNaC::sinh_eval()`, `GiNaC::tan↵  
_eval()`, and `GiNaC::tanh_eval()`.

## 7.37 ginac.h File Reference

This include file includes all other public [GiNaC](#) headers.

```
#include "version.h"
#include "basic.h"
#include "ex.h"
#include "normal.h"
#include "archive.h"
#include "print.h"
#include "constant.h"
#include "fail.h"
#include "integral.h"
#include "lst.h"
#include "matrix.h"
#include "numeric.h"
#include "power.h"
#include "relational.h"
#include "structure.h"
#include "symbol.h"
#include "pseries.h"
#include "wildcard.h"
#include "symmetry.h"
#include "expair.h"
#include "expairseq.h"
#include "add.h"
#include "mul.h"
#include "exprseq.h"
#include "function.h"
```

```
#include "ncmul.h"
#include "inifcns.h"
#include "fderivative.h"
#include "operators.h"
#include "hash_map.h"
#include "idx.h"
#include "indexed.h"
#include "tensor.h"
#include "color.h"
#include "clifford.h"
#include "factor.h"
#include "integration_kernel.h"
#include "excompiler.h"
#include "parser.h"
```

### 7.37.1 Detailed Description

This include file includes all other public [GiNaC](#) headers.

## 7.38 hash\_map.h File Reference

Replacement for map<> using hash tables.

```
#include <unordered_map>
```

### Namespaces

- [GiNaC](#)

### Typedefs

- `template<typename T, class Hash = std::hash<ex>, class KeyEqual = std::equal_to<ex>, class Allocator = std::allocator<std::pair<const ex, T>>>`  
using [GiNaC::exhashmap](#) = `std::unordered_map< ex, T, Hash, KeyEqual, Allocator >`

### 7.38.1 Detailed Description

Replacement for map<> using hash tables.

## 7.39 hash\_seed.h File Reference

Type-specific hash seed.

```
#include <typeinfo>
#include <cstring>
#include "utils.h"
```



## Namespaces

- [GiNaC](#)

## Functions

- static unsigned [GiNaC::make\\_hash\\_seed](#) (const std::type\_info &tinfo)  
*We need a hash function which gives different values for objects of different types.*

### 7.39.1 Detailed Description

Type-specific hash seed.

## 7.40 idx.cpp File Reference

Implementation of [GiNaC](#)'s indices.

```
#include "idx.h"
#include "symbol.h"
#include "lst.h"
#include "relational.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <iostream>
#include <sstream>
#include <stdexcept>
```

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (idx, basic, print\_func< print\_context >(&idx::do\_print). print\_func< print\_latex >(&idx::do\_print\_latex). print\_func< print\_csrc >(&idx::do\_print\_csrc). print\_func< print\_tree >(&idx::do\_print\_tree)) GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT(varidx
- [GiNaC::print\\_func< print\\_context >](#) (&varidx::do\_print). print\_func< print\_latex >(&varidx
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (idx)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (varidx)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (spinidx)
- bool [GiNaC::is\\_dummy\\_pair](#) (const idx &i1, const idx &i2)  
*Check whether two indices form a dummy pair.*
- bool [GiNaC::is\\_dummy\\_pair](#) (const ex &e1, const ex &e2)  
*Check whether two expressions form a dummy index pair.*
- void [GiNaC::find\\_free\\_and\\_dummy](#) (exvector::const\_iterator it, exvector::const\_iterator itend, exvector &out\_free, exvector &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- ex [GiNaC::minimal\\_dim](#) (const ex &dim1, const ex &dim2)  
*Return the minimum of two index dimensions.*

## Variables

- [GiNaC::idx](#)

### 7.40.1 Detailed Description

Implementation of [GiNaC](#)'s indices.

## 7.41 idx.h File Reference

Interface to [GiNaC](#)'s indices.

```
#include "ex.h"
#include "numeric.h"
```

## Classes

- class [GiNaC::idx](#)  
*This class holds one index of an indexed object.*
- class [GiNaC::varidx](#)  
*This class holds an index with a variance (co- or contravariant).*
- class [GiNaC::spinidx](#)  
*This class holds a spinor index that can be dotted or undotted and that also has a variance.*

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (idx)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (varidx)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (spinidx)
- bool [GiNaC::is\\_dummy\\_pair](#) (const idx &i1, const idx &i2)  
*Check whether two indices form a dummy pair.*
- bool [GiNaC::is\\_dummy\\_pair](#) (const ex &e1, const ex &e2)  
*Check whether two expressions form a dummy index pair.*
- void [GiNaC::find\\_free\\_and\\_dummy](#) (exvector::const\_iterator it, exvector::const\_iterator itend, exvector &out\_free, exvector &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- void [GiNaC::find\\_free\\_and\\_dummy](#) (const exvector &v, exvector &out\_free, exvector &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- void [GiNaC::find\\_dummy\\_indices](#) (const exvector &v, exvector &out\_dummy)  
*Given a vector of indices, find the dummy indices.*
- size\_t [GiNaC::count\\_dummy\\_indices](#) (const exvector &v)  
*Count the number of dummy index pairs in an index vector.*
- size\_t [GiNaC::count\\_free\\_indices](#) (const exvector &v)  
*Count the number of dummy index pairs in an index vector.*
- ex [GiNaC::minimal\\_dim](#) (const ex &dim1, const ex &dim2)  
*Return the minimum of two index dimensions.*

### 7.41.1 Detailed Description

Interface to [GiNaC](#)'s indices.

## 7.42 indexed.cpp File Reference

Implementation of [GiNaC](#)'s indexed expressions.

```
#include "indexed.h"
#include "idx.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "power.h"
#include "relational.h"
#include "symmetry.h"
#include "operators.h"
#include "lst.h"
#include "archive.h"
#include "symbol.h"
#include "utils.h"
#include "integral.h"
#include "matrix.h"
#include "inifcns.h"
#include <iostream>
#include <limits>
#include <sstream>
#include <stdexcept>
```

### Classes

- struct [GiNaC::idx\\_is\\_equal\\_ignore\\_dim](#)
- struct [GiNaC::is\\_summation\\_idx](#)
- struct [GiNaC::ex\\_base\\_is\\_less](#)
- class [GiNaC::terminfo](#)

*This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.*

- class [GiNaC::terminfo\\_is\\_less](#)
- class [GiNaC::symminfo](#)

*This structure stores the individual symmetrized terms obtained during the simplification of sums.*

- class [GiNaC::symminfo\\_is\\_less\\_by\\_symmterm](#)
- class [GiNaC::symminfo\\_is\\_less\\_by\\_orig](#)

### Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (indexed, exprseq, print\_func< print\_context >(&indexed::do\_print). print\_func< print\_latex >(&indexed::do\_print\_latex). print\_func< print\_tree >(&indexed::do\_print\_tree)) indexed
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (indexed)
- static bool [GiNaC::indices\\_consistent](#) (const exvector &v1, const exvector &v2)  
*Check whether two sorted index vectors are consistent (i.e.*
- template<class T >  
size\_t [GiNaC::number\\_of\\_type](#) (const exvector &v)
- template<class T >  
static ex [GiNaC::rename\\_dummy\\_indices](#) (const ex &e, exvector &global\_dummy\_indices, exvector &local\_↵\_dummy\_indices)  
*Rename dummy indices in an expression.*
- static void [GiNaC::find\\_variant\\_indices](#) (const exvector &v, exvector &variant\_indices)  
*Given a set of indices, extract those of class varidx.*
- bool [GiNaC::reposition\\_dummy\\_indices](#) (ex &e, exvector &variant\_dummy\_indices, exvector &moved\_↵\_indices)  
*Raise/lower dummy indices in a single indexed objects to canonicalize their variance.*
- static void [GiNaC::product\\_to\\_exvector](#) (const ex &e, exvector &v, bool &non\_commutative)
- template<class T >  
ex [GiNaC::idx\\_symmetrization](#) (const ex &r, const exvector &local\_dummy\_indices)
- ex [GiNaC::simplify\\_indexed](#) (const ex &e, exvector &free\_indices, exvector &dummy\_indices, const scalar\_↵\_products &sp)  
*Simplify indexed expression, return list of free indices.*
- ex [GiNaC::simplify\\_indexed\\_product](#) (const ex &e, exvector &free\_indices, exvector &dummy\_indices, const scalar\_products &sp)  
*Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.*
- bool [GiNaC::hasindex](#) (const ex &x, const ex &sym)
- exvector [GiNaC::get\\_all\\_dummy\\_indices\\_safely](#) (const ex &e)  
*More reliable version of the form.*
- exvector [GiNaC::get\\_all\\_dummy\\_indices](#) (const ex &e)  
*Returns all dummy indices from the exvector.*
- lst [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const exvector &va, const exvector &vb)  
*Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.*
- ex [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const exvector &va, const exvector &vb, const ex &b)  
*Same as above, where va and vb contain the indices of a and b and are sorted.*
- ex [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const ex &a, const ex &b)  
*Returns b with all dummy indices, which are common with a, renamed.*
- ex [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (exvector &va, const ex &b, bool modify\_va=false)  
*Returns b with all dummy indices, which are listed in va, renamed if modify\_va is set to TRUE all dummy indices of b will be appended to va.*
- ex [GiNaC::expand\\_dummy\\_sum](#) (const ex &e, bool subs\_idx=false)  
*This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.*

### 7.42.1 Detailed Description

Implementation of [GiNaC](#)'s indexed expressions.

## 7.43 indexed.h File Reference

Interface to [GiNaC](#)'s indexed expressions.

```
#include "exprseq.h"
#include "wildcard.h"
#include <map>
```

### Classes

- class [GiNaC::indexed](#)  
*This class holds an indexed expression.*
- class [GiNaC::spmapkey](#)
- class [GiNaC::scalar\\_products](#)  
*Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).*

### Namespaces

- [GiNaC](#)

### Typedefs

- typedef std::map< spmapkey, ex > [GiNaC::spmap](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (indexed)
- exvector [GiNaC::get\\_all\\_dummy\\_indices](#) (const ex &e)  
*Returns all dummy indices from the exvector.*
- exvector [GiNaC::get\\_all\\_dummy\\_indices\\_safely](#) (const ex &e)  
*More reliable version of the form.*
- ex [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (exvector &va, const ex &b, bool modify\_va=false)  
*Returns b with all dummy indices, which are listed in va, renamed if modify\_va is set to TRUE all dummy indices of b will be appended to va.*
- ex [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const ex &a, const ex &b)  
*Returns b with all dummy indices, which are common with a, renamed.*
- ex [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const exvector &va, const exvector &vb, const ex &b)  
*Same as above, where va and vb contain the indices of a and b and are sorted.*
- lst [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const exvector &va, const exvector &vb)  
*Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.*
- ex [GiNaC::expand\\_dummy\\_sum](#) (const ex &e, bool subs\_idx=false)  
*This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.*

#### 7.43.1 Detailed Description

Interface to [GiNaC](#)'s indexed expressions.

## 7.44 inifcns.cpp File Reference

Implementation of [GiNaC](#)'s initially known functions.

```
#include "inifcns.h"
#include "ex.h"
#include "constant.h"
#include "lst.h"
#include "fderivative.h"
#include "matrix.h"
#include "mul.h"
#include "power.h"
#include "operators.h"
#include "relational.h"
#include "pseries.h"
#include "symbol.h"
#include "symmetry.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

### Classes

- class [GiNaC::symbolset](#)

### Namespaces

- [GiNaC](#)

### Functions

- static ex [GiNaC::conjugate\\_evalf](#) (const ex &arg)
- static ex [GiNaC::conjugate\\_eval](#) (const ex &arg)
- static void [GiNaC::conjugate\\_print\\_latex](#) (const ex &arg, const print\_context &c)
- static ex [GiNaC::conjugate\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::conjugate\\_expl\\_derivative](#) (const ex &arg, const symbol &s)
- static ex [GiNaC::conjugate\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::conjugate\\_imag\\_part](#) (const ex &arg)
- static bool [GiNaC::func\\_arg\\_info](#) (const ex &arg, unsigned inf)
- static bool [GiNaC::conjugate\\_info](#) (const ex &arg, unsigned inf)
- [GiNaC::REGISTER\\_FUNCTION](#) (conjugate\_function, eval\_func(conjugate\_eval). evalf\_func(conjugate\_evalf). expl\_derivative\_func(conjugate\_expl\_derivative). info\_func(conjugate\_info). print\_func< print\_context >(conjugate\_print\_latex). conjugate\_func(conjugate\_conjugate). real\_part\_func(conjugate\_real\_part). imag\_part\_func(conjugate\_imag\_part). set\_name("conjugate","conjugate"))
- static ex [GiNaC::real\\_part\\_evalf](#) (const ex &arg)
- static ex [GiNaC::real\\_part\\_eval](#) (const ex &arg)
- static void [GiNaC::real\\_part\\_print\\_latex](#) (const ex &arg, const print\_context &c)
- static ex [GiNaC::real\\_part\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::real\\_part\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::real\\_part\\_imag\\_part](#) (const ex &arg)
- static ex [GiNaC::real\\_part\\_expl\\_derivative](#) (const ex &arg, const symbol &s)

- [GiNaC::REGISTER\\_FUNCTION](#) (real\_part\_function, eval\_func(real\_part\_eval). evalf\_func(real\_part\_↵evalf). expl\_derivative\_func(real\_part\_expl\_derivative). print\_func< print\_latex >(real\_part\_print\_latex). conjugate\_func(real\_part\_conjugate). real\_part\_func(real\_part\_real\_part). imag\_part\_func(real\_part\_↵imag\_part). set\_name("real\_part","real\_part"))
- static ex [GiNaC::imag\\_part\\_evalf](#) (const ex &arg)
- static ex [GiNaC::imag\\_part\\_eval](#) (const ex &arg)
- static void [GiNaC::imag\\_part\\_print\\_latex](#) (const ex &arg, const print\_context &c)
- static ex [GiNaC::imag\\_part\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::imag\\_part\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::imag\\_part\\_imag\\_part](#) (const ex &arg)
- static ex [GiNaC::imag\\_part\\_expl\\_derivative](#) (const ex &arg, const symbol &s)
- [GiNaC::REGISTER\\_FUNCTION](#) (imag\_part\_function, eval\_func(imag\_part\_eval). evalf\_func(imag\_part\_↵evalf). expl\_derivative\_func(imag\_part\_expl\_derivative). print\_func< print\_latex >(imag\_part\_print\_latex). conjugate\_func(imag\_part\_conjugate). real\_part\_func(imag\_part\_real\_part). imag\_part\_func(imag\_part\_↵imag\_part). set\_name("imag\_part","imag\_part"))
- static ex [GiNaC::abs\\_evalf](#) (const ex &arg)
- static ex [GiNaC::abs\\_eval](#) (const ex &arg)
- static ex [GiNaC::abs\\_expand](#) (const ex &arg, unsigned [options](#))
- static ex [GiNaC::abs\\_expl\\_derivative](#) (const ex &arg, const symbol &s)
- static void [GiNaC::abs\\_print\\_latex](#) (const ex &arg, const print\_context &c)
- static void [GiNaC::abs\\_print\\_csrc\\_float](#) (const ex &arg, const print\_context &c)
- static ex [GiNaC::abs\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::abs\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::abs\\_imag\\_part](#) (const ex &arg)
- static ex [GiNaC::abs\\_power](#) (const ex &arg, const ex &exp)
- bool [GiNaC::abs\\_info](#) (const ex &arg, unsigned inf)
- [GiNaC::REGISTER\\_FUNCTION](#) (abs, eval\_func(abs\_eval). evalf\_func(abs\_evalf). expand\_func(abs\_↵expand). expl\_derivative\_func(abs\_expl\_derivative). info\_func(abs\_info). print\_func< print\_latex >(abs\_↵\_print\_latex). print\_func< print\_csrc\_float >(abs\_print\_csrc\_float). print\_func< print\_csrc\_double >(abs\_↵\_print\_csrc\_float). conjugate\_func(abs\_conjugate). real\_part\_func(abs\_real\_part). imag\_part\_func(abs\_↵imag\_part). power\_func(abs\_power))
- static ex [GiNaC::step\\_evalf](#) (const ex &arg)
- static ex [GiNaC::step\\_eval](#) (const ex &arg)
- static ex [GiNaC::step\\_series](#) (const ex &arg, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::step\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::step\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::step\\_imag\\_part](#) (const ex &arg)
- [GiNaC::REGISTER\\_FUNCTION](#) (step, eval\_func(step\_eval). evalf\_func(step\_evalf). series\_func(step\_↵series). conjugate\_func(step\_conjugate). real\_part\_func(step\_real\_part). imag\_part\_func(step\_imag\_part))
- static ex [GiNaC::csgn\\_evalf](#) (const ex &arg)
- static ex [GiNaC::csgn\\_eval](#) (const ex &arg)
- static ex [GiNaC::csgn\\_series](#) (const ex &arg, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::csgn\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::csgn\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::csgn\\_imag\\_part](#) (const ex &arg)
- static ex [GiNaC::csgn\\_power](#) (const ex &arg, const ex &exp)
- [GiNaC::REGISTER\\_FUNCTION](#) (csgn, eval\_func(csgn\_eval). evalf\_func(csgn\_evalf). series\_func(csgn\_↵series). conjugate\_func(csgn\_conjugate). real\_part\_func(csgn\_real\_part). imag\_part\_func(csgn\_imag\_↵part). power\_func(csgn\_power))
- static ex [GiNaC::eta\\_evalf](#) (const ex &x, const ex &y)
- static ex [GiNaC::eta\\_eval](#) (const ex &x, const ex &y)
- static ex [GiNaC::eta\\_series](#) (const ex &x, const ex &y, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::eta\\_conjugate](#) (const ex &x, const ex &y)
- static ex [GiNaC::eta\\_real\\_part](#) (const ex &x, const ex &y)
- static ex [GiNaC::eta\\_imag\\_part](#) (const ex &x, const ex &y)

- `GiNaC::REGISTER_FUNCTION` (`eta`, `eval_func(eta_eval)`. `evalf_func(eta_evalf)`. `series_func(eta_series)`. `latex_name("\eta")`. `set_symmetry(sy_symm(0, 1))`. `conjugate_func(eta_conjugate)`. `real_part_func(eta_↵real_part)`. `imag_part_func(eta_imag_part)`)
- static ex `GiNaC::Li2_evalf` (const ex &`x`)
- static ex `GiNaC::Li2_eval` (const ex &`x`)
- static ex `GiNaC::Li2_deriv` (const ex &`x`, unsigned `deriv_param`)
- static ex `GiNaC::Li2_series` (const ex &`x`, const relational &`rel`, int `order`, unsigned `options`)
- static ex `GiNaC::Li2_conjugate` (const ex &`x`)
- `GiNaC::REGISTER_FUNCTION` (`Li2`, `eval_func(Li2_eval)`. `evalf_func(Li2_evalf)`. `derivative_func(Li2_deriv)`. `series_func(Li2_series)`. `conjugate_func(Li2_conjugate)`. `latex_name("\mathrm{Li}_2")`)
- static ex `GiNaC::Li3_eval` (const ex &`x`)
- `GiNaC::REGISTER_FUNCTION` (`Li3`, `eval_func(Li3_eval)`. `latex_name("\mathrm{Li}_3")`)
- static ex `GiNaC::zetaderiv_eval` (const ex &`n`, const ex &`x`)
- static ex `GiNaC::zetaderiv_deriv` (const ex &`n`, const ex &`x`, unsigned `deriv_param`)
- `GiNaC::REGISTER_FUNCTION` (`zetaderiv`, `eval_func(zetaderiv_eval)`. `derivative_func(zetaderiv_deriv)`. `latex_name("\eta^{\mathrm{rime}}")`)
- static ex `GiNaC::factorial_evalf` (const ex &`x`)
- static ex `GiNaC::factorial_eval` (const ex &`x`)
- static void `GiNaC::factorial_print_dflt_latex` (const ex &`x`, const print\_context &`c`)
- static ex `GiNaC::factorial_conjugate` (const ex &`x`)
- static ex `GiNaC::factorial_real_part` (const ex &`x`)
- static ex `GiNaC::factorial_imag_part` (const ex &`x`)
- `GiNaC::REGISTER_FUNCTION` (`factorial`, `eval_func(factorial_eval)`. `evalf_func(factorial_evalf)`. `print_func< print_dflt >(factorial_print_dflt_latex)`. `print_func< print_latex >(factorial_print_dflt_latex)`. `conjugate_↵func(factorial_conjugate)`. `real_part_func(factorial_real_part)`. `imag_part_func(factorial_imag_part)`)
- static ex `GiNaC::binomial_evalf` (const ex &`x`, const ex &`y`)
- static ex `GiNaC::binomial_sym` (const ex &`x`, const numeric &`y`)
- static ex `GiNaC::binomial_eval` (const ex &`x`, const ex &`y`)
- static ex `GiNaC::binomial_conjugate` (const ex &`x`, const ex &`y`)
- static ex `GiNaC::binomial_real_part` (const ex &`x`, const ex &`y`)
- static ex `GiNaC::binomial_imag_part` (const ex &`x`, const ex &`y`)
- `GiNaC::REGISTER_FUNCTION` (`binomial`, `eval_func(binomial_eval)`. `evalf_func(binomial_evalf)`. `conjugate_↵_func(binomial_conjugate)`. `real_part_func(binomial_real_part)`. `imag_part_func(binomial_imag_part)`)
- static ex `GiNaC::Order_eval` (const ex &`x`)
- static ex `GiNaC::Order_series` (const ex &`x`, const relational &`r`, int `order`, unsigned `options`)
- static ex `GiNaC::Order_conjugate` (const ex &`x`)
- static ex `GiNaC::Order_real_part` (const ex &`x`)
- static ex `GiNaC::Order_imag_part` (const ex &`x`)
- static ex `GiNaC::Order_expl_derivative` (const ex &`arg`, const symbol &`s`)
- `GiNaC::REGISTER_FUNCTION` (`Order`, `eval_func(Order_eval)`. `series_func(Order_series)`. `latex_↵name("\mathrm{O}")`. `expl_derivative_func(Order_expl_derivative)`. `conjugate_func(Order_conjugate)`. `real_↵part_func(Order_real_part)`. `imag_part_func(Order_imag_part)`)
- ex `GiNaC::Isolve` (const ex &`eqns`, const ex &`symbols`, unsigned `options=solve_algo::automatic`)  
*Factorial function.*
- const numeric `GiNaC::fsolve` (const ex &`f`, const symbol &`x`, const numeric &`x1`, const numeric &`x2`)  
*Find a real root of real-valued function f(x) numerically within a given interval.*

## Variables

- unsigned `GiNaC::force_include_tgamma` = `tgamma_SERIAL::serial`
- unsigned `GiNaC::force_include_zeta1` = `zeta1_SERIAL::serial`



### 7.44.1 Detailed Description

Implementation of [GiNaC](#)'s initially known functions.

## 7.45 inifcns.h File Reference

Interface to [GiNaC](#)'s initially known functions.

```
#include "numeric.h"
#include "function.h"
#include "ex.h"
```

### Classes

- class [GiNaC::zeta1\\_SERIAL](#)  
*Complex conjugate.*
- class [GiNaC::zeta2\\_SERIAL](#)  
*Alternating Euler sum or colored MZV.*
- class [GiNaC::G2\\_SERIAL](#)  
*Generalized multiple polylogarithm.*
- class [GiNaC::G3\\_SERIAL](#)  
*Generalized multiple polylogarithm with explicit imaginary parts.*
- class [GiNaC::psi1\\_SERIAL](#)  
*Polylogarithm and multiple polylogarithm.*
- class [GiNaC::psi2\\_SERIAL](#)  
*Derivatives of Psi-function (aka polygamma-functions).*
- class [GiNaC::iterated\\_integral2\\_SERIAL](#)  
*Complete elliptic integral of the first kind.*
- class [GiNaC::iterated\\_integral3\\_SERIAL](#)  
*Iterated integral with explicit truncation.*

### Namespaces

- [GiNaC](#)

### Functions

- `template<typename T1 >`  
function [GiNaC::zeta](#) (const T1 &p1)
- `template<typename T1 , typename T2 >`  
function [GiNaC::zeta](#) (const T1 &p1, const T2 &p2)
- `template<>`  
bool [GiNaC::is\\_the\\_function](#)< [zeta\\_SERIAL](#) > (const ex &x)
- `template<typename T1 , typename T2 >`  
function [GiNaC::G](#) (const T1 &x, const T2 &y)
- `template<typename T1 , typename T2 , typename T3 >`  
function [GiNaC::G](#) (const T1 &x, const T2 &s, const T3 &y)

- `template<>`  
`bool GiNaC::is_the_function< G_SERIAL > (const ex &x)`
- `template<typename T1 >`  
`function GiNaC::psi (const T1 &p1)`
- `template<typename T1 , typename T2 >`  
`function GiNaC::psi (const T1 &p1, const T2 &p2)`
- `template<>`  
`bool GiNaC::is_the_function< psi_SERIAL > (const ex &x)`
- `template<typename T1 , typename T2 >`  
`function GiNaC::iterated_integral (const T1 &kernel_lst, const T2 &lambda)`
- `template<typename T1 , typename T2 , typename T3 >`  
`function GiNaC::iterated_integral (const T1 &kernel_lst, const T2 &lambda, const T3 &N_trunc)`
- `template<>`  
`bool GiNaC::is_the_function< iterated_integral_SERIAL > (const ex &x)`
- `ex GiNaC::solve` (const ex &eqns, const ex &symbols, unsigned `options=solve_algo::automatic`)  
*Factorial function.*
- `const numeric GiNaC::fsolve` (const ex &f, const symbol &x, const numeric &x1, const numeric &x2)  
*Find a real root of real-valued function  $f(x)$  numerically within a given interval.*
- `bool GiNaC::is_order_function` (const ex &e)  
*Check whether a function is the Order ( $O(n)$ ) function.*
- `ex GiNaC::convert_H_to_Li` (const ex &parameterlst, const ex &arg)  
*Converts a given list containing parameters for  $H$  in Remiddi/Vermaseren notation into the corresponding GiNaC functions.*

### 7.45.1 Detailed Description

Interface to GiNaC's initially known functions.

## 7.46 inifcns\_elliptic.cpp File Reference

Implementation of some special functions related to elliptic curves.

```
#include "inifcns.h"
#include "add.h"
#include "constant.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "operators.h"
#include "power.h"
#include "pseries.h"
#include "relational.h"
#include "symbol.h"
#include "utils.h"
#include "wildcard.h"
#include "integration_kernel.h"
#include "utils_multi_iterator.h"
#include <cln/cln.h>
#include <sstream>
#include <stdexcept>
#include <vector>
#include <cmath>
```

## Namespaces

- [GiNaC](#)

## Functions

- static ex [GiNaC::EllipticK\\_evalf](#) (const ex &k)
- static ex [GiNaC::EllipticK\\_eval](#) (const ex &k)
- static ex [GiNaC::EllipticK\\_deriv](#) (const ex &k, unsigned deriv\_param)
- static ex [GiNaC::EllipticK\\_series](#) (const ex &k, const relational &rel, int [order](#), unsigned [options](#))
- static void [GiNaC::EllipticK\\_print\\_latex](#) (const ex &k, const print\_context &c)
- [GiNaC::REGISTER\\_FUNCTION](#) (EllipticK, evalf\_func(EllipticK\_evalf). eval\_func(EllipticK\_eval). derivative↵\_func(EllipticK\_deriv). series\_func(EllipticK\_series). print\_func< print\_latex >(EllipticK\_print\_latex). do\_↵not\_evalf\_params())
- static ex [GiNaC::EllipticE\\_evalf](#) (const ex &k)
- static ex [GiNaC::EllipticE\\_eval](#) (const ex &k)
- static ex [GiNaC::EllipticE\\_deriv](#) (const ex &k, unsigned deriv\_param)
- static ex [GiNaC::EllipticE\\_series](#) (const ex &k, const relational &rel, int [order](#), unsigned [options](#))
- static void [GiNaC::EllipticE\\_print\\_latex](#) (const ex &k, const print\_context &c)
- [GiNaC::REGISTER\\_FUNCTION](#) (EllipticE, evalf\_func(EllipticE\_evalf). eval\_func(EllipticE\_eval). derivative↵\_func(EllipticE\_deriv). series\_func(EllipticE\_series). print\_func< print\_latex >(EllipticE\_print\_latex). do\_↵not\_evalf\_params())
- static ex [GiNaC::iterated\\_integral\\_evalf\\_impl](#) (const ex &kernel\_lst, const ex &lambda, const ex &N\_trunc)
- static ex [GiNaC::iterated\\_integral2\\_evalf](#) (const ex &kernel\_lst, const ex &lambda)
- static ex [GiNaC::iterated\\_integral3\\_evalf](#) (const ex &kernel\_lst, const ex &lambda, const ex &N\_trunc)
- static ex [GiNaC::iterated\\_integral2\\_eval](#) (const ex &kernel\_lst, const ex &lambda)
- static ex [GiNaC::iterated\\_integral3\\_eval](#) (const ex &kernel\_lst, const ex &lambda, const ex &N\_trunc)

### 7.46.1 Detailed Description

Implementation of some special functions related to elliptic curves.

The functions are: complete elliptic integral of the first kind [EllipticK\(k\)](#) complete elliptic integral of the second kind [EllipticE\(k\)](#) iterated integral [iterated\\_integral\(a,y\)](#) or [iterated\\_integral\(a,y,N\\_trunc\)](#)

Some remarks:

- All formulae used can be looked up in the following publication: [WW] Numerical evaluation of iterated integrals related to elliptic Feynman integrals, M.Walden, S.Weinzierl, arXiv:2010.05271
- When these routines and methods are used for scientific work that leads to publication in a scientific journal, please refer to this program as : M.Walden, S.Weinzierl, "Numerical evaluation of iterated integrals related to elliptic Feynman integrals", arXiv:2010.05271
- As these routines build on the core part of [GiNaC](#), it is also polite to acknowledge C. Bauer, A. Frink, R. Kreckel, "Introduction to the GiNaC Framework for Symbolic Computation within the C++ Programming Language", J. Symbolic Computations 33, 1 (2002), cs.sc/0004015

## 7.47 inifcns\_gamma.cpp File Reference

Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff.

```
#include "inifcns.h"
#include "constant.h"
#include "pseries.h"
#include "numeric.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "symbol.h"
#include "symmetry.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

### Namespaces

- [GiNaC](#)

### Functions

- static ex [GiNaC::lgamma\\_evalf](#) (const ex &[x](#))
- static ex [GiNaC::lgamma\\_eval](#) (const ex &[x](#))
 

*Evaluation of  $\lgamma(x)$ , the natural logarithm of the Gamma function.*
- static ex [GiNaC::lgamma\\_deriv](#) (const ex &[x](#), unsigned deriv\_param)
- static ex [GiNaC::lgamma\\_series](#) (const ex &arg, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::lgamma\\_conjugate](#) (const ex &[x](#))
- [GiNaC::REGISTER\\_FUNCTION](#) (lgamma, eval\_func(lgamma\_eval). evalf\_func(lgamma\_evalf). derivative↔\_func(lgamma\_deriv). series\_func(lgamma\_series). conjugate\_func(lgamma\_conjugate). latex\_name("\og\amma"))
- static ex [GiNaC::tgamma\\_evalf](#) (const ex &[x](#))
- static ex [GiNaC::tgamma\\_eval](#) (const ex &[x](#))
 

*Evaluation of  $tgamma(x)$ , the true Gamma function.*
- static ex [GiNaC::tgamma\\_deriv](#) (const ex &[x](#), unsigned deriv\_param)
- static ex [GiNaC::tgamma\\_series](#) (const ex &arg, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::tgamma\\_conjugate](#) (const ex &[x](#))
- [GiNaC::REGISTER\\_FUNCTION](#) (tgamma, eval\_func(tgamma\_eval). evalf\_func(tgamma\_evalf). derivative↔\_func(tgamma\_deriv). series\_func(tgamma\_series). conjugate\_func(tgamma\_conjugate). latex↔\_name("\amma"))
- static ex [GiNaC::beta\\_evalf](#) (const ex &[x](#), const ex &[y](#))
- static ex [GiNaC::beta\\_eval](#) (const ex &[x](#), const ex &[y](#))
- static ex [GiNaC::beta\\_deriv](#) (const ex &[x](#), const ex &[y](#), unsigned deriv\_param)
- static ex [GiNaC::beta\\_series](#) (const ex &arg1, const ex &arg2, const relational &rel, int [order](#), unsigned [options](#))
- [GiNaC::REGISTER\\_FUNCTION](#) (beta, eval\_func(beta\_eval). evalf\_func(beta\_evalf). derivative↔\_func(beta↔\_deriv). series\_func(beta\_series). latex\_name("\athrm{B}"). set\_symmetry(sy\_symm(0, 1)))
- static ex [GiNaC::psi1\\_evalf](#) (const ex &[x](#))
- static ex [GiNaC::psi1\\_eval](#) (const ex &[x](#))
 

*Evaluation of digamma-function  $\psi(x)$ .*
- static ex [GiNaC::psi1\\_deriv](#) (const ex &[x](#), unsigned deriv\_param)

- static ex [GiNaC::psi1\\_series](#) (const ex &arg, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::psi2\\_evalf](#) (const ex &n, const ex &x)
- static ex [GiNaC::psi2\\_eval](#) (const ex &n, const ex &x)  
*Evaluation of polygamma-function  $\psi(n,x)$ .*
- static ex [GiNaC::psi2\\_deriv](#) (const ex &n, const ex &x, unsigned deriv\_param)
- static ex [GiNaC::psi2\\_series](#) (const ex &n, const ex &arg, const relational &rel, int [order](#), unsigned [options](#))

### 7.47.1 Detailed Description

Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff.

## 7.48 inifcns\_nstdsums.cpp File Reference

Implementation of some special functions that have a representation as nested sums.

```
#include "inifcns.h"
#include "add.h"
#include "constant.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "operators.h"
#include "power.h"
#include "pseries.h"
#include "relational.h"
#include "symbol.h"
#include "utils.h"
#include "wildcard.h"
#include <cln/cln.h>
#include <sstream>
#include <stdexcept>
#include <vector>
#include <cmath>
```

### Namespaces

- [GiNaC](#)

### Functions

- static ex [GiNaC::G2\\_evalf](#) (const ex &x\_, const ex &y)
- static ex [GiNaC::G2\\_eval](#) (const ex &x\_, const ex &y)
- static ex [GiNaC::G3\\_evalf](#) (const ex &x\_, const ex &s\_, const ex &y)
- static ex [GiNaC::G3\\_eval](#) (const ex &x\_, const ex &s\_, const ex &y)
- static ex [GiNaC::Li\\_evalf](#) (const ex &m\_, const ex &x\_)
- static ex [GiNaC::Li\\_eval](#) (const ex &m\_, const ex &x\_)
- static ex [GiNaC::Li\\_series](#) (const ex &m, const ex &x, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::Li\\_deriv](#) (const ex &m\_, const ex &x\_, unsigned deriv\_param)
- static void [GiNaC::Li\\_print\\_latex](#) (const ex &m\_, const ex &x\_, const print\_context &c)

- `GiNaC::REGISTER_FUNCTION` (Li, evalf\_func(Li\_evalf). eval\_func(Li\_eval). series\_func(Li\_series). derivative\_func(Li\_deriv). print\_func< print\_latex >(Li\_print\_latex). do\_not\_evalf\_params())
- static ex `GiNaC::S_evalf` (const ex &n, const ex &p, const ex &x)
- static ex `GiNaC::S_eval` (const ex &n, const ex &p, const ex &x)
- static ex `GiNaC::S_series` (const ex &n, const ex &p, const ex &x, const relational &rel, int order, unsigned options)
- static ex `GiNaC::S_deriv` (const ex &n, const ex &p, const ex &x, unsigned deriv\_param)
- static void `GiNaC::S_print_latex` (const ex &n, const ex &p, const ex &x, const print\_context &c)
- `GiNaC::REGISTER_FUNCTION` (S, evalf\_func(S\_evalf). eval\_func(S\_eval). series\_func(S\_series). derivative\_func(S\_deriv). print\_func< print\_latex >(S\_print\_latex). do\_not\_evalf\_params())
- static ex `GiNaC::H_evalf` (const ex &x1, const ex &x2)
- static ex `GiNaC::H_eval` (const ex &m\_, const ex &x)
- static ex `GiNaC::H_series` (const ex &m, const ex &x, const relational &rel, int order, unsigned options)
- static ex `GiNaC::H_deriv` (const ex &m\_, const ex &x, unsigned deriv\_param)
- static void `GiNaC::H_print_latex` (const ex &m\_, const ex &x, const print\_context &c)
- `GiNaC::REGISTER_FUNCTION` (H, evalf\_func(H\_evalf). eval\_func(H\_eval). series\_func(H\_series). derivative\_func(H\_deriv). print\_func< print\_latex >(H\_print\_latex). do\_not\_evalf\_params())
- ex `GiNaC::convert_H_to_Li` (const ex &parameterlst, const ex &arg)

*Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding GiNaC functions.*

- static ex `GiNaC::zeta1_evalf` (const ex &x)
- static ex `GiNaC::zeta1_eval` (const ex &m)
- static ex `GiNaC::zeta1_deriv` (const ex &m, unsigned deriv\_param)
- static void `GiNaC::zeta1_print_latex` (const ex &m\_, const print\_context &c)
- static ex `GiNaC::zeta2_evalf` (const ex &x, const ex &s)
- static ex `GiNaC::zeta2_eval` (const ex &m, const ex &s\_)
- static ex `GiNaC::zeta2_deriv` (const ex &m, const ex &s, unsigned deriv\_param)
- static void `GiNaC::zeta2_print_latex` (const ex &m\_, const ex &s\_, const print\_context &c)

## 7.48.1 Detailed Description

Implementation of some special functions that have a representation as nested sums.

The functions are: classical polylogarithm  $\text{Li}(n,x)$  multiple polylogarithm  $\text{Li}(\{m_1,\dots,m_k\},\{x_1,\dots,x_k\})$   $G(\{a_1,\dots,a_k\},y)$  or  $G(\{a_1,\dots,a_k\},\{s_1,\dots,s_k\},y)$  Nielsen's generalized polylogarithm  $S(n,p,x)$  harmonic polylogarithm  $H(m,x)$  or  $H(\{m_1,\dots,m_k\},x)$  multiple zeta value  $\text{zeta}(m)$  or  $\text{zeta}(\{m_1,\dots,m_k\})$  alternating Euler sum  $\text{zeta}(m,s)$  or  $\text{zeta}(\{m_1,\dots,m_k\},\{s_1,\dots,s_k\})$

Some remarks:

- All formulae used can be looked up in the following publications: [Kol] Nielsen's Generalized Polylogarithms, K.S.Kolbig, SIAM J.Math.Anal. 17 (1986), pp. 1232-1258. [Cra] Fast Evaluation of Multiple Zeta Sums, R.E.Crandall, Math.Comp. 67 (1998), pp. 1163-1172. [ReV] Harmonic Polylogarithms, E.Remiddi, J.A. Vermaseren, Int.J.Mod.Phys. A15 (2000), pp. 725-754 [BBB] Special Values of Multiple Polylogarithms, J.Borwein, D.Bradley, D.Broadhurst, P.Lisonek, Trans.Amer.Math.Soc. 353/3 (2001), pp. 907-941 [VSW] Numerical evaluation of multiple polylogarithms, J.Vollinga, S.Weinzierl, hep-ph/0410259
- The order of parameters and arguments of Li and zeta is defined according to the nested sums representation. The parameters for H are understood as in [ReV]. They can be in expanded — only 0, 1 and -1 — or in compactified — a string with zeros in front of 1 or -1 is written as a single number — notation.
- All functions can be numerically evaluated with arguments in the whole complex plane. The parameters for Li, zeta and S must be positive integers. If you want to have an alternating Euler sum, you have to give the signs of the parameters as a second argument s to  $\text{zeta}(m,s)$  containing 1 and -1.

- The calculation of classical polylogarithms is speeded up by using Bernoulli numbers and look-up tables. S uses look-up tables as well. The zeta function applies the algorithms in [Cra] and [BBB] for speed up. Multiple polylogarithms use Hoelder convolution [BBB].
- The functions have no means to do a series expansion into nested sums. To do this, you have to convert these functions into the appropriate objects from the nestedsums library, do the expansion and convert the result back.
- Numerical testing of this implementation has been performed by doing a comparison of results between this software and the commercial M..... 4.1. Multiple zeta values have been checked by means of evaluations into simple zeta values. Harmonic polylogarithms have been checked by comparison to  $S(n,p,x)$  for corresponding parameter combinations and by continuity checks around  $|x|=1$  along with comparisons to corresponding zeta functions. Multiple polylogarithms were checked against H and zeta and by means of shuffle and quasi-shuffle relations.

## 7.49 inifcns\_trans.cpp File Reference

Implementation of transcendental (and trigonometric and hyperbolic) functions.

```
#include "inifcns.h"
#include "ex.h"
#include "constant.h"
#include "add.h"
#include "mul.h"
#include "numeric.h"
#include "power.h"
#include "operators.h"
#include "relational.h"
#include "symbol.h"
#include "pseries.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

### Namespaces

- [GiNaC](#)

### Functions

- static ex [GiNaC::exp\\_valf](#) (const ex &x)
- static ex [GiNaC::exp\\_eval](#) (const ex &x)
- static ex [GiNaC::exp\\_expand](#) (const ex &arg, unsigned [options](#))
- static ex [GiNaC::exp\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::exp\\_real\\_part](#) (const ex &x)
- static ex [GiNaC::exp\\_imag\\_part](#) (const ex &x)
- static ex [GiNaC::exp\\_conjugate](#) (const ex &x)
- static ex [GiNaC::exp\\_power](#) (const ex &x, const ex &a)
- [GiNaC::REGISTER\\_FUNCTION](#) (exp, eval\_func(exp\_eval). evalf\_func(exp\_valf). expand\_func(exp\_expand). derivative\_func(exp\_deriv). real\_part\_func(exp\_real\_part). imag\_part\_func(exp\_imag\_part). conjugate\_func(exp\_conjugate). power\_func(exp\_power). latex\_name("xp"))
- static ex [GiNaC::log\\_valf](#) (const ex &x)

- static ex [GiNaC::log\\_eval](#) (const ex &[x](#))
- static ex [GiNaC::log\\_deriv](#) (const ex &[x](#), unsigned deriv\_param)
- static ex [GiNaC::log\\_series](#) (const ex &arg, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::log\\_real\\_part](#) (const ex &[x](#))
- static ex [GiNaC::log\\_imag\\_part](#) (const ex &[x](#))
- static ex [GiNaC::log\\_expand](#) (const ex &arg, unsigned [options](#))
- static ex [GiNaC::log\\_conjugate](#) (const ex &[x](#))
- [GiNaC::REGISTER\\_FUNCTION](#) (log, eval\_func(log\_eval). evalf\_func(log\_evalf). expand\_func(log\_expand). derivative\_func(log\_deriv). series\_func(log\_series). real\_part\_func(log\_real\_part). imag\_part\_func(log\_↵  
imag\_part). conjugate\_func(log\_conjugate). latex\_name("\n"))
- static ex [GiNaC::sin\\_evalf](#) (const ex &[x](#))
- static ex [GiNaC::sin\\_eval](#) (const ex &[x](#))
- static ex [GiNaC::sin\\_deriv](#) (const ex &[x](#), unsigned deriv\_param)
- static ex [GiNaC::sin\\_real\\_part](#) (const ex &[x](#))
- static ex [GiNaC::sin\\_imag\\_part](#) (const ex &[x](#))
- static ex [GiNaC::sin\\_conjugate](#) (const ex &[x](#))
- [GiNaC::REGISTER\\_FUNCTION](#) (sin, eval\_func(sin\_eval). evalf\_func(sin\_evalf). derivative\_func(sin\_deriv). real\_part\_func(sin\_real\_part). imag\_part\_func(sin\_imag\_part). conjugate\_func(sin\_conjugate). latex\_↵  
name("\in"))
- static ex [GiNaC::cos\\_evalf](#) (const ex &[x](#))
- static ex [GiNaC::cos\\_eval](#) (const ex &[x](#))
- static ex [GiNaC::cos\\_deriv](#) (const ex &[x](#), unsigned deriv\_param)
- static ex [GiNaC::cos\\_real\\_part](#) (const ex &[x](#))
- static ex [GiNaC::cos\\_imag\\_part](#) (const ex &[x](#))
- static ex [GiNaC::cos\\_conjugate](#) (const ex &[x](#))
- [GiNaC::REGISTER\\_FUNCTION](#) (cos, eval\_func(cos\_eval). evalf\_func(cos\_evalf). derivative\_func(cos\_↵  
deriv). real\_part\_func(cos\_real\_part). imag\_part\_func(cos\_imag\_part). conjugate\_func(cos\_conjugate). latex\_name("\os"))
- static ex [GiNaC::tan\\_evalf](#) (const ex &[x](#))
- static ex [GiNaC::tan\\_eval](#) (const ex &[x](#))
- static ex [GiNaC::tan\\_deriv](#) (const ex &[x](#), unsigned deriv\_param)
- static ex [GiNaC::tan\\_real\\_part](#) (const ex &[x](#))
- static ex [GiNaC::tan\\_imag\\_part](#) (const ex &[x](#))
- static ex [GiNaC::tan\\_series](#) (const ex &[x](#), const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::tan\\_conjugate](#) (const ex &[x](#))
- [GiNaC::REGISTER\\_FUNCTION](#) (tan, eval\_func(tan\_eval). evalf\_func(tan\_evalf). derivative\_func(tan\_↵  
deriv). series\_func(tan\_series). real\_part\_func(tan\_real\_part). imag\_part\_func(tan\_imag\_part). conjugate\_↵  
\_func(tan\_conjugate). latex\_name("\an"))
- static ex [GiNaC::asin\\_evalf](#) (const ex &[x](#))
- static ex [GiNaC::asin\\_eval](#) (const ex &[x](#))
- static ex [GiNaC::asin\\_deriv](#) (const ex &[x](#), unsigned deriv\_param)
- static ex [GiNaC::asin\\_conjugate](#) (const ex &[x](#))
- [GiNaC::REGISTER\\_FUNCTION](#) (asin, eval\_func(asin\_eval). evalf\_func(asin\_evalf). derivative\_func(asin\_↵  
deriv). conjugate\_func(asin\_conjugate). latex\_name("\rcsin"))
- static ex [GiNaC::acos\\_evalf](#) (const ex &[x](#))
- static ex [GiNaC::acos\\_eval](#) (const ex &[x](#))
- static ex [GiNaC::acos\\_deriv](#) (const ex &[x](#), unsigned deriv\_param)
- static ex [GiNaC::acos\\_conjugate](#) (const ex &[x](#))
- [GiNaC::REGISTER\\_FUNCTION](#) (acos, eval\_func(acos\_eval). evalf\_func(acos\_evalf). derivative\_↵  
func(acos\_deriv). conjugate\_func(acos\_conjugate). latex\_name("\rccos"))
- static ex [GiNaC::atan\\_evalf](#) (const ex &[x](#))
- static ex [GiNaC::atan\\_eval](#) (const ex &[x](#))
- static ex [GiNaC::atan\\_deriv](#) (const ex &[x](#), unsigned deriv\_param)
- static ex [GiNaC::atan\\_series](#) (const ex &arg, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::atan\\_conjugate](#) (const ex &[x](#))



- [GiNaC::REGISTER\\_FUNCTION](#) (atan, eval\_func(atan\_eval). evalf\_func(atan\_evalf). derivative\_func(atan↵  
\_deriv). series\_func(atan\_series). conjugate\_func(atan\_conjugate). latex\_name("\rctan"))
- static ex [GiNaC::atan2\\_evalf](#) (const ex &y, const ex &x)
- static ex [GiNaC::atan2\\_eval](#) (const ex &y, const ex &x)
- static ex [GiNaC::atan2\\_deriv](#) (const ex &y, const ex &x, unsigned deriv\_param)
- [GiNaC::REGISTER\\_FUNCTION](#) (atan2, eval\_func(atan2\_eval). evalf\_func(atan2\_evalf). derivative\_↵  
func(atan2\_deriv))
- static ex [GiNaC::sinh\\_evalf](#) (const ex &x)
- static ex [GiNaC::sinh\\_eval](#) (const ex &x)
- static ex [GiNaC::sinh\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::sinh\\_real\\_part](#) (const ex &x)
- static ex [GiNaC::sinh\\_imag\\_part](#) (const ex &x)
- static ex [GiNaC::sinh\\_conjugate](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (sinh, eval\_func(sinh\_eval). evalf\_func(sinh\_evalf). derivative\_func(sinh↵  
\_deriv). real\_part\_func(sinh\_real\_part). imag\_part\_func(sinh\_imag\_part). conjugate\_func(sinh\_conjugate).  
latex\_name("\sinh"))
- static ex [GiNaC::cosh\\_evalf](#) (const ex &x)
- static ex [GiNaC::cosh\\_eval](#) (const ex &x)
- static ex [GiNaC::cosh\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::cosh\\_real\\_part](#) (const ex &x)
- static ex [GiNaC::cosh\\_imag\\_part](#) (const ex &x)
- static ex [GiNaC::cosh\\_conjugate](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (cosh, eval\_func(cosh\_eval). evalf\_func(cosh\_evalf). derivative\_↵  
func(cosh\_deriv). real\_part\_func(cosh\_real\_part). imag\_part\_func(cosh\_imag\_part). conjugate\_↵  
func(cosh\_conjugate). latex\_name("\cosh"))
- static ex [GiNaC::tanh\\_evalf](#) (const ex &x)
- static ex [GiNaC::tanh\\_eval](#) (const ex &x)
- static ex [GiNaC::tanh\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::tanh\\_series](#) (const ex &x, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::tanh\\_real\\_part](#) (const ex &x)
- static ex [GiNaC::tanh\\_imag\\_part](#) (const ex &x)
- static ex [GiNaC::tanh\\_conjugate](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (tanh, eval\_func(tanh\_eval). evalf\_func(tanh\_evalf). derivative\_func(tanh↵  
\_deriv). series\_func(tanh\_series). real\_part\_func(tanh\_real\_part). imag\_part\_func(tanh\_imag\_part).  
conjugate\_func(tanh\_conjugate). latex\_name("\anh"))
- static ex [GiNaC::asinh\\_evalf](#) (const ex &x)
- static ex [GiNaC::asinh\\_eval](#) (const ex &x)
- static ex [GiNaC::asinh\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::asinh\\_conjugate](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (asinh, eval\_func(asinh\_eval). evalf\_func(asinh\_evalf). derivative\_↵  
func(asinh\_deriv). conjugate\_func(asinh\_conjugate))
- static ex [GiNaC::acosh\\_evalf](#) (const ex &x)
- static ex [GiNaC::acosh\\_eval](#) (const ex &x)
- static ex [GiNaC::acosh\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::acosh\\_conjugate](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (acosh, eval\_func(acosh\_eval). evalf\_func(acosh\_evalf). derivative\_↵  
func(acosh\_deriv). conjugate\_func(acosh\_conjugate))
- static ex [GiNaC::atanh\\_evalf](#) (const ex &x)
- static ex [GiNaC::atanh\\_eval](#) (const ex &x)
- static ex [GiNaC::atanh\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::atanh\\_series](#) (const ex &arg, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::atanh\\_conjugate](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (atanh, eval\_func(atanh\_eval). evalf\_func(atanh\_evalf). derivative\_↵  
func(atanh\_deriv). series\_func(atanh\_series). conjugate\_func(atanh\_conjugate))

### 7.49.1 Detailed Description

Implementation of transcendental (and trigonometric and hyperbolic) functions.

## 7.50 integral.cpp File Reference

Implementation of [GiNaC](#)'s symbolic integral.

```
#include "integral.h"
#include "numeric.h"
#include "symbol.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "inifcns.h"
#include "wildcard.h"
#include "archive.h"
#include "registrar.h"
#include "utils.h"
#include "operators.h"
#include "relational.h"
```

### Classes

- struct [GiNaC::error\\_and\\_integral](#)
- struct [GiNaC::error\\_and\\_integral\\_is\\_less](#)

### Namespaces

- [GiNaC](#)

### Typedefs

- typedef map< error\_and\_integral, ex, error\_and\_integral\_is\_less > [GiNaC::lookup\\_map](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (integral, basic, print\_func< print\_dflt >(&integral::do\_print). print\_func< print\_python >(&integral::do\_print). print\_func< print\_latex >(&integral::do\_print\_latex)) integral
- ex [GiNaC::subvalue](#) (const ex &var, const ex &value, const ex &fun)
- ex [GiNaC::adaptivesimpson](#) (const ex &x, const ex &a\_in, const ex &b\_in, const ex &f, const ex &error)  
*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (integral)

### 7.50.1 Detailed Description

Implementation of [GiNaC](#)'s symbolic integral.

## 7.51 integral.h File Reference

Interface to [GiNaC](#)'s symbolic integral.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

### Classes

- class [GiNaC::integral](#)  
*Symbolic integral.*

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (integral)
- ex [GiNaC::adaptivesimpson](#) (const ex &x, const ex &a\_in, const ex &b\_in, const ex &f, const ex &error)  
*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*

#### 7.51.1 Detailed Description

Interface to [GiNaC](#)'s symbolic integral.

## 7.52 integration\_kernel.cpp File Reference

Implementation of [GiNaC](#)'s integration kernels for iterated integrals.

```
#include "integration_kernel.h"
#include "add.h"
#include "mul.h"
#include "operators.h"
#include "power.h"
#include "relational.h"
#include "symbol.h"
#include "constant.h"
#include "numeric.h"
#include "function.h"
#include "pseries.h"
#include "utils.h"
#include "inifcns.h"
#include <iostream>
#include <stdexcept>
#include <cln/cln.h>
```

## Namespaces

- [GiNaC](#)

## Functions

- ex [GiNaC::ifactor](#) (const numeric &n)  
*Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $lst( lst(p1,...,pr), lst(a1,...,ar) )$  such that  $n = p1^{a1} \dots$*
- bool [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field](#) (const numeric &n)  
*Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.*
- numeric [GiNaC::kronecker\\_symbol](#) (const numeric &a, const numeric &n)  
*Returns the Kronecker symbol  $a: integer\ n: integer$ .*
- numeric [GiNaC::primitive\\_dirichlet\\_character](#) (const numeric &n, const numeric &a)  
*Defines a primitive Dirichlet character through the Kronecker symbol.*
- numeric [GiNaC::dirichlet\\_character](#) (const numeric &n, const numeric &a, const numeric &N)  
*Defines a Dirichlet character through the Kronecker symbol.*
- numeric [GiNaC::generalised\\_Bernoulli\\_number](#) (const numeric &k, const numeric &b)  
*The generalised Bernoulli number.*
- ex [GiNaC::Bernoulli\\_polynomial](#) (const numeric &k, const ex &x)  
*The Bernoulli polynomials.*
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (integration\_kernel, basic, print\_func< print\_↵ context >(&integration\_kernel::do\_print)) integration\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (integration\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (basic\_log\_kernel, integration\_kernel, print\_↵ func< print\_context >(&basic\_log\_kernel::do\_print)) basic\_log\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (basic\_log\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (multiple\_polylog\_kernel, integration\_kernel, print\_func< print\_context >(&multiple\_polylog\_kernel::do\_print)) multiple\_polylog\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (multiple\_polylog\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (ELi\_kernel, integration\_kernel, print\_func< print\_context >(&ELi\_kernel::do\_print)) ELi\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (ELi\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Ebar\_kernel, integration\_kernel, print\_func< print\_context >(&Ebar\_kernel::do\_print)) Ebar\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Ebar\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dtau\_kernel, integration\_kernel, print\_func< print\_context >(&Kronecker\_dtau\_kernel::do\_print)) Kronecker\_dtau\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dtau\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dz\_kernel, integration\_kernel, print\_func< print\_context >(&Kronecker\_dz\_kernel::do\_print)) Kronecker\_dz\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dz\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_kernel, integration\_kernel, print\_↵ func< print\_context >(&Eisenstein\_kernel::do\_print)) Eisenstein\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_h\_kernel, integration\_kernel, print\_func< print\_context >(&Eisenstein\_h\_kernel::do\_print)) Eisenstein\_h\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_h\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (modular\_form\_kernel, integration\_kernel, print\_func< print\_context >(&modular\_form\_kernel::do\_print)) modular\_form\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (modular\_form\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (user\_defined\_kernel, integration\_kernel, print\_func< print\_context >(&user\_defined\_kernel::do\_print)) user\_defined\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (user\_defined\_kernel)

## 7.52.1 Detailed Description

Implementation of [GiNaC](#)'s integration kernels for iterated integrals.

## 7.52.2 Variable Documentation

### 7.52.2.1 qbar

```
ex qbar
```

Referenced by `GiNaC::divide_in_z()`, `GiNaC::ELi_kernel::get_numerical_value()`, `GiNaC::Ebar_kernel::get_numerical_value()`, `GiNaC::Kronecker_dtau_kernel::get_numerical_value()`, `GiNaC::Eisenstein_kernel::get_numerical_value()`, `GiNaC::Eisenstein_h_kernel::get_numerical_value()`, `GiNaC::modular_form_kernel::get_numerical_value()`, `GiNaC::modular_form_kernel::is_numeric()`, `GiNaC::modular_form_kernel::Laurent_series()`, `GiNaC::Eisenstein_kernel::series()`, `GiNaC::Eisenstein_h_kernel::series()`, `GiNaC::modular_form_kernel::series()`, and `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`.

### 7.52.2.2 order

```
int order
```

Referenced by `GiNaC::atan_series()`, `GiNaC::atanh_series()`, `GiNaC::beta_series()`, `GiNaC::fderivative::do_print_latex()`, `GiNaC::EllipticE_series()`, `GiNaC::EllipticK_series()`, `GiNaC::integration_kernel::Laurent_series()`, `GiNaC::Eisenstein_kernel::Laurent_series()`, `GiNaC::Eisenstein_h_kernel::Laurent_series()`, `GiNaC::modular_form_kernel::Laurent_series()`, `GiNaC::user_defined_kernel::Laurent_series()`, `GiNaC::lgamma_series()`, `GiNaC::Li2_series()`, `GiNaC::Li_series()`, `GiNaC::log_series()`, `GiNaC::Order_series()`, `GiNaC::psi1_series()`, `GiNaC::psi2_series()`, `GiNaC::Eisenstein_kernel::q_expansion_modular_form()`, `GiNaC::S_series()`, `GiNaC::symbol::series()`, `GiNaC::pseries::series()`, `GiNaC::add::series()`, `GiNaC::mul::series()`, `GiNaC::integration_kernel::series()`, `GiNaC::integral::series()`, `GiNaC::fderivative::series()`, `GiNaC::power::series()`, `GiNaC::structure< T, ComparisonPolicy >::series()`, `GiNaC::ex::series()`, `GiNaC::basic::series()`, `GiNaC::Eisenstein_kernel::series()`, `GiNaC::Eisenstein_h_kernel::series()`, `GiNaC::modular_form_kernel::series()`, `GiNaC::function::series()`, `GiNaC::series()`, `GiNaC::tan_series()`, `GiNaC::tanh_series()`, and `GiNaC::tgamma_series()`.

### 7.52.2.3 cache\_vec

```
std::vector<ex> cache_vec [static], [protected]
```

### 7.52.2.4 x

```
symbol x [static], [protected]
```

Referenced by `GiNaC::Bernoulli_polynomial()`, `GiNaC::generalised_Bernoulli_number()`, `GiNaC::integration_kernel::Laurent_series()`, `GiNaC::Eisenstein_kernel::Laurent_series()`, `GiNaC::Eisenstein_h_kernel::Laurent_series()`, and `GiNaC::integration_kernel::series_coeff()`.

## 7.53 integration\_kernel.h File Reference

Interface to [GiNaC](#)'s integration kernels for iterated integrals.

```
#include "basic.h"
#include "archive.h"
#include "numeric.h"
#include <cln/complex.h>
#include <vector>
```

### Classes

- class [GiNaC::integration\\_kernel](#)  
The base class for integration kernels for iterated integrals.
- class [GiNaC::basic\\_log\\_kernel](#)  
The basic integration kernel with a logarithmic singularity at the origin.
- class [GiNaC::multiple\\_polylog\\_kernel](#)  
The integration kernel for multiple polylogarithms.
- class [GiNaC::ELi\\_kernel](#)  
The ELi-kernel.
- class [GiNaC::Ebar\\_kernel](#)  
The Ebar-kernel.
- class [GiNaC::Kronecker\\_dtau\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).
- class [GiNaC::Kronecker\\_dz\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .
- class [GiNaC::Eisenstein\\_kernel](#)  
The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .
- class [GiNaC::Eisenstein\\_h\\_kernel](#)  
The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .
- class [GiNaC::modular\\_form\\_kernel](#)  
A kernel corresponding to a polynomial in Eisenstein series.
- class [GiNaC::user\\_defined\\_kernel](#)  
A user-defined integration kernel.

### Namespaces

- [GiNaC](#)

### Functions

- ex [GiNaC::ifactor](#) (const numeric &n)  
Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $lst( lst(p1,...,pr), lst(a1,...,ar) )$  such that  $n = p1^{a1} \dots$
- bool [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field](#) (const numeric &n)  
Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.
- numeric [GiNaC::kronecker\\_symbol](#) (const numeric &a, const numeric &n)  
Returns the Kronecker symbol  $a$ : integer  $n$ : integer.

- numeric [GiNaC::primitive\\_dirichlet\\_character](#) (const numeric &n, const numeric &a)  
*Defines a primitive Dirichlet character through the Kronecker symbol.*
- numeric [GiNaC::dirichlet\\_character](#) (const numeric &n, const numeric &a, const numeric &N)  
*Defines a Dirichlet character through the Kronecker symbol.*
- numeric [GiNaC::generalised\\_Bernoulli\\_number](#) (const numeric &k, const numeric &b)  
*The generalised Bernoulli number.*
- ex [GiNaC::Bernoulli\\_polynomial](#) (const numeric &k, const ex &x)  
*The Bernoulli polynomials.*
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (integration\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (basic\_log\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (multiple\_polylog\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (ELi\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Ebar\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Kronecker\_dtau\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Kronecker\_dz\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Eisenstein\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Eisenstein\_h\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (modular\_form\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (user\_defined\_kernel)

### 7.53.1 Detailed Description

Interface to [GiNaC](#)'s integration kernels for iterated integrals.

## 7.54 Ist.cpp File Reference

Implementation of [GiNaC](#)'s Ist.

```
#include "lst.h"
#include "archive.h"
```

### Namespaces

- [GiNaC](#)

### 7.54.1 Detailed Description

Implementation of [GiNaC](#)'s Ist.

## 7.55 Ist.h File Reference

Definition of [GiNaC](#)'s Ist.

```
#include "container.h"
#include <list>
```

## Namespaces

- [GiNaC](#)

## Typedefs

- typedef container< std::list > [GiNaC::lst](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (lst)

### 7.55.1 Detailed Description

Definition of [GiNaC](#)'s lst.

## 7.56 matrix.cpp File Reference

Implementation of symbolic matrices.

```
#include "matrix.h"
#include "numeric.h"
#include "lst.h"
#include "idx.h"
#include "indexed.h"
#include "add.h"
#include "power.h"
#include "symbol.h"
#include "operators.h"
#include "normal.h"
#include "archive.h"
#include "utils.h"
#include <algorithm>
#include <iostream>
#include <map>
#include <sstream>
#include <stdexcept>
#include <string>
```

## Namespaces

- [GiNaC](#)



## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (matrix, basic, print\_func< print\_context >(&matrix::do\_print). print\_func< print\_latex >(&matrix::do\_print\_latex). print\_func< print\_tree >(&matrix::do\_print\_tree). print\_func< print\_python\_repr >(&matrix::do\_print\_python\_repr)) matrix  
*Default ctor.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (matrix)
- ex [GiNaC::lst\\_to\\_matrix](#) (const lst &l)  
*Convert list of lists to matrix.*
- ex [GiNaC::diag\\_matrix](#) (const lst &l)  
*Convert list of diagonal elements to matrix.*
- ex [GiNaC::diag\\_matrix](#) (std::initializer\_list< ex > l)
- ex [GiNaC::unit\\_matrix](#) (unsigned r, unsigned c)  
*Create an r times c unit matrix.*
- ex [GiNaC::symbolic\\_matrix](#) (unsigned r, unsigned c, const std::string &base\_name, const std::string &tex\_↔ base\_name)  
*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- ex [GiNaC::reduced\\_matrix](#) (const matrix &m, unsigned r, unsigned c)  
*Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*
- ex [GiNaC::sub\\_matrix](#) (const matrix &m, unsigned r, unsigned nr, unsigned c, unsigned nc)  
*Return the nr times nc submatrix starting at position r, c of matrix m.*

### 7.56.1 Detailed Description

Implementation of symbolic matrices.

## 7.57 matrix.h File Reference

Interface to symbolic matrices.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include "compiler.h"
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::matrix](#)  
*Symbolic matrices.*

## Namespaces

- [GiNaC](#)

## Functions

- `GiNaC::GINAC_DECLARE_UNARCHIVER` (matrix)
- `size_t GiNaC::nops` (const matrix &m)
- ex `GiNaC::expand` (const matrix &m, unsigned options=0)
- ex `GiNaC::evalf` (const matrix &m)
- unsigned `GiNaC::rows` (const matrix &m)
- unsigned `GiNaC::cols` (const matrix &m)
- matrix `GiNaC::transpose` (const matrix &m)
- ex `GiNaC::determinant` (const matrix &m, unsigned options=determinant\_algo::automatic)
- ex `GiNaC::trace` (const matrix &m)
- ex `GiNaC::charpoly` (const matrix &m, const ex &lambda)
- matrix `GiNaC::inverse` (const matrix &m)
- matrix `GiNaC::inverse` (const matrix &m, unsigned algo)
- unsigned `GiNaC::rank` (const matrix &m)
- unsigned `GiNaC::rank` (const matrix &m, unsigned solve\_algo)
- ex `GiNaC::lst_to_matrix` (const lst &l)
 

*Convert list of lists to matrix.*
- ex `GiNaC::diag_matrix` (const lst &l)
 

*Convert list of diagonal elements to matrix.*
- ex `GiNaC::diag_matrix` (std::initializer\_list< ex > l)
- ex `GiNaC::unit_matrix` (unsigned r, unsigned c)
 

*Create an r times c unit matrix.*
- ex `GiNaC::unit_matrix` (unsigned x)
 

*Create a x times x unit matrix.*
- ex `GiNaC::symbolic_matrix` (unsigned r, unsigned c, const std::string &base\_name, const std::string &tex\_↔  
base\_name)
 

*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- ex `GiNaC::reduced_matrix` (const matrix &m, unsigned r, unsigned c)
 

*Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*
- ex `GiNaC::sub_matrix` (const matrix &m, unsigned r, unsigned nr, unsigned c, unsigned nc)
 

*Return the nr times nc submatrix starting at position r, c of matrix m.*
- ex `GiNaC::symbolic_matrix` (unsigned r, unsigned c, const std::string &base\_name)
 

*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*

### 7.57.1 Detailed Description

Interface to symbolic matrices.

## 7.58 mul.cpp File Reference

Implementation of `GiNaC`'s products of expressions.

```
#include "mul.h"
#include "add.h"
#include "power.h"
#include "operators.h"
#include "matrix.h"
```

```
#include "indexed.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "symbol.h"
#include "compiler.h"
#include <iostream>
#include <limits>
#include <stdexcept>
#include <vector>
```

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (mul, expairseq, print\_func< print\_context >(&mul::do\_print). print\_func< print\_latex >(&mul::do\_print\_latex). print\_func< print\_csrc >(&mul::do\_print\_csrc). print\_func< print\_tree >(&mul::do\_print\_tree). print\_func< print\_python\_repr >(&mul::do\_print\_python\_repr)) mul
- bool [GiNaC::tryfactsubs](#) (const ex &origfactor, const ex &patternfactor, int &nummatches, exmap &repls)
- bool [GiNaC::algebraic\\_match\\_mul\\_with\\_mul](#) (const mul &e, const ex &pat, exmap &repls, int factor, int &nummatches, const std::vector< bool > &subsed, std::vector< bool > &matched)  
*Checks whether e matches to the pattern pat and the (possibly to be updated) list of replacements repls.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (mul)

### 7.58.1 Detailed Description

Implementation of [GiNaC](#)'s products of expressions.

## 7.59 mul.h File Reference

Interface to [GiNaC](#)'s products of expressions.

```
#include "expairseq.h"
```

## Classes

- class [GiNaC::mul](#)  
*Product of expressions.*

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (mul)

### 7.59.1 Detailed Description

Interface to [GiNaC](#)'s products of expressions.

## 7.60 ncmul.cpp File Reference

Implementation of [GiNaC](#)'s non-commutative products of expressions.

```
#include "ncmul.h"
#include "ex.h"
#include "add.h"
#include "mul.h"
#include "clifford.h"
#include "matrix.h"
#include "archive.h"
#include "indexed.h"
#include "utils.h"
#include <algorithm>
#include <iostream>
#include <stdexcept>
```

## Namespaces

- [GiNaC](#)

## Typedefs

- typedef std::vector< std::size\_t > [GiNaC::uintvector](#)
- typedef std::vector< unsigned > [GiNaC::unsignedvector](#)
- typedef std::vector< exvector > [GiNaC::exvectorvector](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (ncmul, exprseq, print\_func< print\_context >(&ncmul::do\_print). print\_func< print\_tree >(&ncmul::do\_print\_tree). print\_func< print\_csrc >(&ncmul::do\_print\_csrc). print\_func< print\_python\_repr >(&ncmul::do\_print\_csrc)) ncmul
- ex [GiNaC::reeval\\_ncmul](#) (const exvector &v)
- ex [GiNaC::hold\\_ncmul](#) (const exvector &v)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (ncmul)

### 7.60.1 Detailed Description

Implementation of [GiNaC](#)'s non-commutative products of expressions.

## 7.61 ncmul.h File Reference

Interface to [GiNaC](#)'s non-commutative products of expressions.

```
#include "exprseq.h"
#include "archive.h"
```

### Classes

- class [GiNaC::ncmul](#)  
*Non-commutative product of expressions.*

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (ncmul)
- ex [GiNaC::reeval\\_ncmul](#) (const exvector &v)
- ex [GiNaC::hold\\_ncmul](#) (const exvector &v)

#### 7.61.1 Detailed Description

Interface to [GiNaC](#)'s non-commutative products of expressions.

## 7.62 normal.cpp File Reference

This file implements several functions that work on univariate and multivariate polynomials and rational functions.

```
#include "normal.h"
#include "basic.h"
#include "ex.h"
#include "add.h"
#include "constant.h"
#include "expairseq.h"
#include "fail.h"
#include "inifcns.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "matrix.h"
#include "pseries.h"
#include "symbol.h"
#include "utils.h"
#include "polynomial/chinrem_gcd.h"
#include <algorithm>
#include <map>
```

## Classes

- struct [GiNaC::sym\\_desc](#)  
*This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".*
- class [GiNaC::gcdheu\\_failed](#)  
*Exception thrown by [heur\\_gcd\(\)](#) to signal failure.*
- struct [GiNaC::normal\\_map\\_function](#)  
*Function object to be applied by [basic::normal\(\)](#).*

## Namespaces

- [GiNaC](#)

## Macros

- `#define` [FAST\\_COMPARE](#) 1
- `#define` [USE\\_REMEMBER](#) 0
- `#define` [USE\\_TRIAL\\_DIVISION](#) 0
- `#define` [STATISTICS](#) 0

## Typedefs

- `typedef std::vector< sym_desc >` [GiNaC::sym\\_desc\\_vec](#)

## Functions

- static bool [GiNaC::get\\_first\\_symbol](#) (const ex &e, ex &x)  
*Return pointer to first symbol found in expression.*
- static void [GiNaC::add\\_symbol](#) (const ex &s, sym\_desc\_vec &v)
- static void [GiNaC::collect\\_symbols](#) (const ex &e, sym\_desc\_vec &v)
- static void [GiNaC::get\\_symbol\\_stats](#) (const ex &a, const ex &b, sym\_desc\_vec &v)  
*Collect statistical information about symbols in polynomials.*
- static numeric [GiNaC::lcmcoeff](#) (const ex &e, const numeric &l)
- static numeric [GiNaC::lcm\\_of\\_coefficients\\_denominators](#) (const ex &e)  
*Compute LCM of denominators of coefficients of a polynomial.*
- static ex [GiNaC::multiply\\_lcm](#) (const ex &e, const numeric &lcm)  
*Bring polynomial from  $Q[X]$  to  $Z[X]$  by multiplying in the previously determined LCM of the coefficient's denominators.*
- ex [GiNaC::quo](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::rem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::decomp\\_rational](#) (const ex &a, const ex &x)  
*Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .*
- ex [GiNaC::prem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::sprem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- bool [GiNaC::divide](#) (const ex &a, const ex &b, ex &q, bool check\_args)

- Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Q[X]$ .*
  - static bool [GiNaC::divide\\_in\\_z](#) (const ex &a, const ex &b, ex &q, sym\_desc\_vec::const\_iterator var)
- Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Z[X]$ .*
  - static ex [GiNaC::sr\\_gcd](#) (const ex &a, const ex &b, sym\_desc\_vec::const\_iterator var)
- Compute GCD of multivariate polynomials using the subresultant PRS algorithm.*
  - static ex [GiNaC::interpolate](#) (const ex &gamma, const numeric &xi, const ex &x, int degree\_hint=1)
- $\xi$ -adic polynomial interpolation*
  - static bool [GiNaC::heur\\_gcd\\_z](#) (ex &res, const ex &a, const ex &b, ex \*ca, ex \*cb, sym\_desc\_vec::const\_iterator var)
- Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
  - static bool [GiNaC::heur\\_gcd](#) (ex &res, const ex &a, const ex &b, ex \*ca, ex \*cb, sym\_desc\_vec::const\_iterator var)
- Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
  - static ex [GiNaC::gcd\\_pf\\_pow](#) (const ex &a, const ex &b, ex \*ca, ex \*cb)
  - static ex [GiNaC::gcd\\_pf\\_mul](#) (const ex &a, const ex &b, ex \*ca, ex \*cb)
  - ex [GiNaC::gcd](#) (const ex &a, const ex &b, ex \*ca, ex \*cb, bool check\_args, unsigned options)
- Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $Z[X]$ .*
  - static ex [GiNaC::gcd\\_pf\\_pow\\_pow](#) (const ex &a, const ex &b, ex \*ca, ex \*cb)
  - ex [GiNaC::lcm](#) (const ex &a, const ex &b, bool check\_args)
- Compute LCM (Least Common Multiple) of multivariate polynomials in  $Z[X]$ .*
  - static epvector [GiNaC::sqrfree\\_yun](#) (const ex &a, const symbol &x)
- Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.*
  - ex [GiNaC::sqrfree](#) (const ex &a, const lst &l)
- Compute a square-free factorization of a multivariate polynomial in  $Q[X]$ .*
  - ex [GiNaC::sqrfree\\_parfrac](#) (const ex &a, const symbol &x)
- Compute square-free partial fraction decomposition of rational function  $a(x)$ .*
  - static ex [GiNaC::replace\\_with\\_symbol](#) (const ex &e, exmap &repl, exmap &rev\_lookup, lst &modifier)
- Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
  - static ex [GiNaC::replace\\_with\\_symbol](#) (const ex &e, exmap &repl)
- Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
  - static ex [GiNaC::frac\\_cancel](#) (const ex &n, const ex &d)
- Fraction cancellation.*
  - static ex [GiNaC::find\\_common\\_factor](#) (const ex &e, ex &factor, exmap &repl)
- Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).*
  - ex [GiNaC::collect\\_common\\_factors](#) (const ex &e)
- Collect common factors in sums.*
  - ex [GiNaC::resultant](#) (const ex &e1, const ex &e2, const ex &s)
- Resultant of two expressions  $e_1, e_2$  with respect to symbol  $s$ .*

### 7.62.1 Detailed Description

This file implements several functions that work on univariate and multivariate polynomials and rational functions.

These functions include polynomial quotient and remainder, GCD and LCM computation, square-free factorization and rational function normalization.

### 7.62.2 Macro Definition Documentation

#### 7.62.2.1 FAST\_COMPARE

```
#define FAST_COMPARE 1
```

#### 7.62.2.2 USE\_REMEMBER

```
#define USE_REMEMBER 0
```

#### 7.62.2.3 USE\_TRIAL\_DIVISION

```
#define USE_TRIAL_DIVISION 0
```

#### 7.62.2.4 STATISTICS

```
#define STATISTICS 0
```

### 7.63 normal.h File Reference

This file defines several functions that work on univariate and multivariate polynomials and rational functions.

```
#include "lst.h"
```

#### Classes

- struct [GiNaC::gcd\\_options](#)  
*Flags to control the behavior of [gcd\(\)](#) and friends.*

#### Namespaces

- [GiNaC](#)



## Functions

- ex [GiNaC::quo](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::rem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::decomp\\_rational](#) (const ex &a, const ex &x)  
*Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .*
- ex [GiNaC::prem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::sprem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- bool [GiNaC::divide](#) (const ex &a, const ex &b, ex &q, bool check\_args)  
*Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Q[X]$ .*
- ex [GiNaC::gcd](#) (const ex &a, const ex &b, ex \*ca, ex \*cb, bool check\_args, unsigned options)  
*Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $Z[X]$ .*
- ex [GiNaC::lcm](#) (const ex &a, const ex &b, bool check\_args)  
*Compute LCM (Least Common Multiple) of multivariate polynomials in  $Z[X]$ .*
- ex [GiNaC::sqrfree](#) (const ex &a, const lst &l)  
*Compute a square-free factorization of a multivariate polynomial in  $Q[X]$ .*
- ex [GiNaC::sqrfree\\_parfrac](#) (const ex &a, const symbol &x)  
*Compute square-free partial fraction decomposition of rational function  $a(x)$ .*
- ex [GiNaC::collect\\_common\\_factors](#) (const ex &e)  
*Collect common factors in sums.*
- ex [GiNaC::resultant](#) (const ex &e1, const ex &e2, const ex &s)  
*Resultant of two expressions  $e1, e2$  with respect to symbol  $s$ .*

### 7.63.1 Detailed Description

This file defines several functions that work on univariate and multivariate polynomials and rational functions.

These functions include polynomial quotient and remainder, GCD and LCM computation, square-free factorization and rational function normalization.

## 7.64 numeric.cpp File Reference

This file contains the interface to the underlying bignum package.

```
#include "numeric.h"
#include "ex.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include <limits>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
#include <cln/output.h>
#include <cln/integer_io.h>
```

```
#include <cln/integer_ring.h>
#include <cln/rational_io.h>
#include <cln/rational_ring.h>
#include <cln/lfloat_class.h>
#include <cln/lfloat_io.h>
#include <cln/real_io.h>
#include <cln/real_ring.h>
#include <cln/complex_io.h>
#include <cln/complex_ring.h>
#include <cln/numtheory.h>
```

## Classes

- class [GiNaC::lanczos\\_coeffs](#)

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (numeric, basic, print\_func< print\_context >(&numeric::do\_print). print\_func< print\_latex >(&numeric::do\_print\_latex). print\_func< print\_csrc >(&numeric::do\_print\_csrc). print\_func< print\_csrc\_cl\_N >(&numeric::do\_print\_csrc\_cl\_N). print\_func< print\_tree >(&numeric::do\_print\_tree). print\_func< print\_python\_repr >(&numeric::do\_print\_python\_repr))  
numeric  
*default ctor.*
- static const cln::cl\_F [GiNaC::make\\_real\\_float](#) (const cln::cl\_idecoded\_float &dec)  
*Construct a floating point number from sign, mantissa, and exponent.*
- static const cln::cl\_F [GiNaC::read\\_real\\_float](#) (std::istream &s)  
*Read serialized floating point number.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (numeric)
- static void [GiNaC::write\\_real\\_float](#) (std::ostream &s, const cln::cl\_R &n)
- static void [GiNaC::print\\_real\\_number](#) (const print\_context &c, const cln::cl\_R &x)  
*Helper function to print a real number in a nicer way than is CLN's default.*
- static void [GiNaC::print\\_integer\\_csrc](#) (const print\_context &c, const cln::cl\_I &x)  
*Helper function to print integer number in C++ source format.*
- static void [GiNaC::print\\_real\\_csrc](#) (const print\_context &c, const cln::cl\_R &x)  
*Helper function to print real number in C++ source format.*
- template<typename T1 , typename T2 >  
static bool [GiNaC::coerce](#) (T1 &dst, const T2 &arg)
- template<>  
bool [GiNaC::coerce< int, cln::cl\\_I >](#) (int &dst, const cln::cl\_I &arg)  
*Check if CLN integer can be converted into int.*
- template<>  
bool [GiNaC::coerce< unsigned int, cln::cl\\_I >](#) (unsigned int &dst, const cln::cl\_I &arg)
- static void [GiNaC::print\\_real\\_cl\\_N](#) (const print\_context &c, const cln::cl\_R &x)  
*Helper function to print real number in C++ source format using cl\_N types.*
- const numeric [GiNaC::exp](#) (const numeric &x)  
*Exponential function.*

- const numeric [GiNaC::log](#) (const numeric &x)  
*Natural logarithm.*
- const numeric [GiNaC::sin](#) (const numeric &x)  
*Numeric sine (trigonometric function).*
- const numeric [GiNaC::cos](#) (const numeric &x)  
*Numeric cosine (trigonometric function).*
- const numeric [GiNaC::tan](#) (const numeric &x)  
*Numeric tangent (trigonometric function).*
- const numeric [GiNaC::asin](#) (const numeric &x)  
*Numeric inverse sine (trigonometric function).*
- const numeric [GiNaC::acos](#) (const numeric &x)  
*Numeric inverse cosine (trigonometric function).*
- const numeric [GiNaC::atan](#) (const numeric &x)  
*Numeric arcustangent.*
- const numeric [GiNaC::atan](#) (const numeric &y, const numeric &x)  
*Numeric arcustangent of two arguments, analytically continued in a suitable way.*
- const numeric [GiNaC::sinh](#) (const numeric &x)  
*Numeric hyperbolic sine (trigonometric function).*
- const numeric [GiNaC::cosh](#) (const numeric &x)  
*Numeric hyperbolic cosine (trigonometric function).*
- const numeric [GiNaC::tanh](#) (const numeric &x)  
*Numeric hyperbolic tangent (trigonometric function).*
- const numeric [GiNaC::asinh](#) (const numeric &x)  
*Numeric inverse hyperbolic sine (trigonometric function).*
- const numeric [GiNaC::acosh](#) (const numeric &x)  
*Numeric inverse hyperbolic cosine (trigonometric function).*
- const numeric [GiNaC::atanh](#) (const numeric &x)  
*Numeric inverse hyperbolic tangent (trigonometric function).*
- static cln::cl\_N [GiNaC::Li2\\_series](#) (const cln::cl\_N &x, const cln::float\_format\_t &prec)  
*Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.*
- static cln::cl\_N [GiNaC::Li2\\_projection](#) (const cln::cl\_N &x, const cln::float\_format\_t &prec)  
*Folds Li2's argument inside a small rectangle to enhance convergence.*
- const cln::cl\_N [GiNaC::Li2\\_](#) (const cln::cl\_N &value)  
*Numeric evaluation of Dilogarithm.*
- const numeric [GiNaC::Li2](#) (const numeric &x)
- const numeric [GiNaC::zeta](#) (const numeric &x)  
*Numeric evaluation of Riemann's Zeta function.*
- static cln::float\_format\_t [GiNaC::guess\\_precision](#) (const cln::cl\_N &x)
- const cln::cl\_N [GiNaC::lgamma](#) (const cln::cl\_N &x)  
*The Gamma function.*
- const numeric [GiNaC::lgamma](#) (const numeric &x)
- const cln::cl\_N [GiNaC::tgamma](#) (const cln::cl\_N &x)
- const numeric [GiNaC::tgamma](#) (const numeric &x)
- const numeric [GiNaC::psi](#) (const numeric &x)  
*The psi function (aka polygamma function).*
- const numeric [GiNaC::psi](#) (const numeric &n, const numeric &x)  
*The psi functions (aka polygamma functions).*
- const numeric [GiNaC::factorial](#) (const numeric &n)  
*Factorial combinatorial function.*
- const numeric [GiNaC::doublefactorial](#) (const numeric &n)  
*The double factorial combinatorial function.*

- const numeric [GiNaC::binomial](#) (const numeric &n, const numeric &k)  
*The Binomial coefficients.*
- const numeric [GiNaC::bernoulli](#) (const numeric &nn)  
*Bernoulli number.*
- const numeric [GiNaC::fibonacci](#) (const numeric &n)  
*Fibonacci number.*
- const numeric [GiNaC::abs](#) (const numeric &x)  
*Absolute value.*
- const numeric [GiNaC::mod](#) (const numeric &a, const numeric &b)  
*Modulus (in positive representation).*
- const numeric [GiNaC::smod](#) (const numeric &a\_, const numeric &b\_)  
*Modulus (in symmetric representation).*
- const numeric [GiNaC::irem](#) (const numeric &a, const numeric &b)  
*Numeric integer remainder.*
- const numeric [GiNaC::irem](#) (const numeric &a, const numeric &b, numeric &q)  
*Numeric integer remainder.*
- const numeric [GiNaC::iquo](#) (const numeric &a, const numeric &b)  
*Numeric integer quotient.*
- const numeric [GiNaC::iquo](#) (const numeric &a, const numeric &b, numeric &r)  
*Numeric integer quotient.*
- const numeric [GiNaC::gcd](#) (const numeric &a, const numeric &b)  
*Greatest Common Divisor.*
- const numeric [GiNaC::lcm](#) (const numeric &a, const numeric &b)  
*Least Common Multiple.*
- const numeric [GiNaC::sqrt](#) (const numeric &x)  
*Numeric square root.*
- const numeric [GiNaC::isqrt](#) (const numeric &x)  
*Integer numeric square root.*
- ex [GiNaC::PiEvalf](#) ()  
*Floating point evaluation of Archimedes' constant Pi.*
- ex [GiNaC::EulerEvalf](#) ()  
*Floating point evaluation of Euler's constant gamma.*
- ex [GiNaC::CatalanEvalf](#) ()  
*Floating point evaluation of Catalan's constant.*
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const \_numeric\_digits &e)

## Variables

- const numeric [GiNaC::I](#) = numeric(cln::complex(cln::cl\_I(0),cln::cl\_I(1)))  
*Imaginary unit.*
- \_numeric\_digits [GiNaC::Digits](#)  
*Accuracy in decimal digits.*

### 7.64.1 Detailed Description

This file contains the interface to the underlying bignum package.

Its most important design principle is to completely hide the inner working of that other package from the user of [GiNaC](#). It must either provide implementation of arithmetic operators and numerical evaluation of special functions or implement the interface to the bignum package.

## 7.65 numeric.h File Reference

Makes the interface to the underlying bignum package available.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include <cln/complex.h>
#include <stdexcept>
#include <vector>
```

### Classes

- class [GiNaC::\\_numeric\\_digits](#)  
*This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.*
- class [GiNaC::pole\\_error](#)  
*Exception class thrown when a singularity is encountered.*
- class [GiNaC::numeric](#)  
*This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.*

### Namespaces

- [GiNaC](#)

### Typedefs

- typedef void(\* [GiNaC::digits\\_changed\\_callback](#)) (long)  
*Function pointer to implement callbacks in the case 'Digits' gets changed.*

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (numeric)
- const numeric [GiNaC::exp](#) (const numeric &x)  
*Exponential function.*
- const numeric [GiNaC::log](#) (const numeric &x)  
*Natural logarithm.*
- const numeric [GiNaC::sin](#) (const numeric &x)  
*Numeric sine (trigonometric function).*
- const numeric [GiNaC::cos](#) (const numeric &x)  
*Numeric cosine (trigonometric function).*
- const numeric [GiNaC::tan](#) (const numeric &x)  
*Numeric tangent (trigonometric function).*
- const numeric [GiNaC::asin](#) (const numeric &x)  
*Numeric inverse sine (trigonometric function).*
- const numeric [GiNaC::acos](#) (const numeric &x)  
*Numeric inverse cosine (trigonometric function).*
- const numeric [GiNaC::atan](#) (const numeric &x)  
*Numeric arcustangent.*

- const numeric [GiNaC::atan](#) (const numeric &y, const numeric &x)  
*Numeric arcustangent of two arguments, analytically continued in a suitable way.*
- const numeric [GiNaC::sinh](#) (const numeric &x)  
*Numeric hyperbolic sine (trigonometric function).*
- const numeric [GiNaC::cosh](#) (const numeric &x)  
*Numeric hyperbolic cosine (trigonometric function).*
- const numeric [GiNaC::tanh](#) (const numeric &x)  
*Numeric hyperbolic tangent (trigonometric function).*
- const numeric [GiNaC::asinh](#) (const numeric &x)  
*Numeric inverse hyperbolic sine (trigonometric function).*
- const numeric [GiNaC::acosh](#) (const numeric &x)  
*Numeric inverse hyperbolic cosine (trigonometric function).*
- const numeric [GiNaC::atanh](#) (const numeric &x)  
*Numeric inverse hyperbolic tangent (trigonometric function).*
- const numeric [GiNaC::Li2](#) (const numeric &x)
- const numeric [GiNaC::zeta](#) (const numeric &x)  
*Numeric evaluation of Riemann's Zeta function.*
- const numeric [GiNaC::lgamma](#) (const numeric &x)
- const numeric [GiNaC::tgamma](#) (const numeric &x)
- const numeric [GiNaC::psi](#) (const numeric &x)  
*The psi function (aka polygamma function).*
- const numeric [GiNaC::psi](#) (const numeric &n, const numeric &x)  
*The psi functions (aka polygamma functions).*
- const numeric [GiNaC::factorial](#) (const numeric &n)  
*Factorial combinatorial function.*
- const numeric [GiNaC::doublefactorial](#) (const numeric &n)  
*The double factorial combinatorial function.*
- const numeric [GiNaC::binomial](#) (const numeric &n, const numeric &k)  
*The Binomial coefficients.*
- const numeric [GiNaC::bernoulli](#) (const numeric &nn)  
*Bernoulli number.*
- const numeric [GiNaC::fibonacci](#) (const numeric &n)  
*Fibonacci number.*
- const numeric [GiNaC::isqrt](#) (const numeric &x)  
*Integer numeric square root.*
- const numeric [GiNaC::sqrt](#) (const numeric &x)  
*Numeric square root.*
- const numeric [GiNaC::abs](#) (const numeric &x)  
*Absolute value.*
- const numeric [GiNaC::mod](#) (const numeric &a, const numeric &b)  
*Modulus (in positive representation).*
- const numeric [GiNaC::smod](#) (const numeric &a\_, const numeric &b\_)  
*Modulus (in symmetric representation).*
- const numeric [GiNaC::irem](#) (const numeric &a, const numeric &b)  
*Numeric integer remainder.*
- const numeric [GiNaC::irem](#) (const numeric &a, const numeric &b, numeric &q)  
*Numeric integer remainder.*
- const numeric [GiNaC::iquo](#) (const numeric &a, const numeric &b)  
*Numeric integer quotient.*
- const numeric [GiNaC::iquo](#) (const numeric &a, const numeric &b, numeric &r)  
*Numeric integer quotient.*

- const numeric [GiNaC::gcd](#) (const numeric &a, const numeric &b)  
*Greatest Common Divisor.*
- const numeric [GiNaC::lcm](#) (const numeric &a, const numeric &b)  
*Least Common Multiple.*
- const numeric [GiNaC::pow](#) (const numeric &x, const numeric &y)
- const numeric [GiNaC::inverse](#) (const numeric &x)
- numeric [GiNaC::step](#) (const numeric &x)
- int [GiNaC::csgn](#) (const numeric &x)
- bool [GiNaC::is\\_zero](#) (const numeric &x)
- bool [GiNaC::is\\_positive](#) (const numeric &x)
- bool [GiNaC::is\\_negative](#) (const numeric &x)
- bool [GiNaC::is\\_integer](#) (const numeric &x)
- bool [GiNaC::is\\_pos\\_integer](#) (const numeric &x)
- bool [GiNaC::is\\_nonneg\\_integer](#) (const numeric &x)
- bool [GiNaC::is\\_even](#) (const numeric &x)
- bool [GiNaC::is\\_odd](#) (const numeric &x)
- bool [GiNaC::is\\_prime](#) (const numeric &x)
- bool [GiNaC::is\\_rational](#) (const numeric &x)
- bool [GiNaC::is\\_real](#) (const numeric &x)
- bool [GiNaC::is\\_cinteger](#) (const numeric &x)
- bool [GiNaC::is\\_crational](#) (const numeric &x)
- int [GiNaC::to\\_int](#) (const numeric &x)
- long [GiNaC::to\\_long](#) (const numeric &x)
- double [GiNaC::to\\_double](#) (const numeric &x)
- const numeric [GiNaC::real](#) (const numeric &x)
- const numeric [GiNaC::imag](#) (const numeric &x)
- const numeric [GiNaC::numer](#) (const numeric &x)
- const numeric [GiNaC::denom](#) (const numeric &x)
- ex [GiNaC::PiEvalf](#) ()  
*Floating point evaluation of Archimedes' constant Pi.*
- ex [GiNaC::EulerEvalf](#) ()  
*Floating point evaluation of Euler's constant gamma.*
- ex [GiNaC::CatalanEvalf](#) ()  
*Floating point evaluation of Catalan's constant.*

### 7.65.1 Detailed Description

Makes the interface to the underlying bignum package available.

## 7.66 operators.cpp File Reference

Implementation of [GiNaC](#)'s overloaded operators.

```
#include "operators.h"
#include "numeric.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "ncmul.h"
#include "relational.h"
#include "print.h"
#include "utils.h"
#include <iostream>
```

## Namespaces

- [GiNaC](#)

## Enumerations

- enum { [GiNaC::callback\\_registered](#) = 1 }

## Functions

- static const ex [GiNaC::exadd](#) (const ex &lh, const ex &rh)  
*Used internally by [operator+\(\)](#) to add two ex objects.*
- static const ex [GiNaC::exmul](#) (const ex &lh, const ex &rh)  
*Used internally by [operator\\*\(\)](#) to multiply two ex objects.*
- static const ex [GiNaC::exminus](#) (const ex &lh)  
*Used internally by [operator-\(\)](#) and friends to change the sign of an argument.*
- const ex [GiNaC::operator+](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator-](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator\\*](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator/](#) (const ex &lh, const ex &rh)
- const numeric [GiNaC::operator+](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator-](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator\\*](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator/](#) (const numeric &lh, const numeric &rh)
- ex & [GiNaC::operator+=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator-=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator\\*=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator/=](#) (ex &lh, const ex &rh)
- numeric & [GiNaC::operator+=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator-=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator\\*=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator/=](#) (numeric &lh, const numeric &rh)
- const ex [GiNaC::operator+](#) (const ex &lh)
- const ex [GiNaC::operator-](#) (const ex &lh)
- const numeric [GiNaC::operator+](#) (const numeric &lh)
- const numeric [GiNaC::operator-](#) (const numeric &lh)
- ex & [GiNaC::operator++](#) (ex &rh)  
*Expression prefix increment.*
- ex & [GiNaC::operator--](#) (ex &rh)  
*Expression prefix decrement.*
- const ex [GiNaC::operator++](#) (ex &lh, int)  
*Expression postfix increment.*
- const ex [GiNaC::operator--](#) (ex &lh, int)  
*Expression postfix decrement.*
- numeric & [GiNaC::operator++](#) (numeric &rh)  
*Numeric prefix increment.*
- numeric & [GiNaC::operator--](#) (numeric &rh)  
*Numeric prefix decrement.*
- const numeric [GiNaC::operator++](#) (numeric &lh, int)  
*Numeric postfix increment.*
- const numeric [GiNaC::operator--](#) (numeric &lh, int)



*Numeric postfix decrement.*

- const relational [GiNaC::operator==](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator!=](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator<](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator<=](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator>](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator>=](#) (const ex &lh, const ex &rh)
- static int [GiNaC::my\\_ios\\_index](#) ()
- static void [GiNaC::my\\_ios\\_callback](#) (std::ios\_base::event ev, std::ios\_base &s, int i)
- static print\_context \* [GiNaC::get\\_print\\_context](#) (std::ios\_base &s)
- static void [GiNaC::set\\_print\\_context](#) (std::ios\_base &s, const print\_context &c)
- static unsigned [GiNaC::get\\_print\\_options](#) (std::ios\_base &s)
- static void [GiNaC::set\\_print\\_options](#) (std::ostream &s, unsigned options)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const ex &e)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const exvector &e)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const exset &e)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const exmap &e)
- std::istream & [GiNaC::operator>>](#) (std::istream &is, ex &e)
- std::ostream & [GiNaC::dflt](#) (std::ostream &os)
- std::ostream & [GiNaC::latex](#) (std::ostream &os)
- std::ostream & [GiNaC::python](#) (std::ostream &os)
- std::ostream & [GiNaC::python\\_repr](#) (std::ostream &os)
- std::ostream & [GiNaC::tree](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_float](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_double](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_cl\\_N](#) (std::ostream &os)
- std::ostream & [GiNaC::index\\_dimensions](#) (std::ostream &os)
- std::ostream & [GiNaC::no\\_index\\_dimensions](#) (std::ostream &os)

### 7.66.1 Detailed Description

Implementation of [GiNaC](#)'s overloaded operators.

## 7.67 operators.h File Reference

Interface to [GiNaC](#)'s overloaded operators.

```
#include <iosfwd>
```

### Namespaces

- [GiNaC](#)

## Functions

- const ex [GiNaC::operator+](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator-](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator\\*](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator/](#) (const ex &lh, const ex &rh)
- const numeric [GiNaC::operator+](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator-](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator\\*](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator/](#) (const numeric &lh, const numeric &rh)
- ex & [GiNaC::operator+=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator-=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator\\*=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator/=](#) (ex &lh, const ex &rh)
- numeric & [GiNaC::operator+=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator-=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator\\*=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator/=](#) (numeric &lh, const numeric &rh)
- const ex [GiNaC::operator+](#) (const ex &lh)
- const ex [GiNaC::operator-](#) (const ex &lh)
- const numeric [GiNaC::operator+](#) (const numeric &lh)
- const numeric [GiNaC::operator-](#) (const numeric &lh)
- ex & [GiNaC::operator++](#) (ex &rh)
- Expression prefix increment.*
- ex & [GiNaC::operator--](#) (ex &rh)
- Expression prefix decrement.*
- const ex [GiNaC::operator++](#) (ex &lh, int)
- Expression postfix increment.*
- const ex [GiNaC::operator--](#) (ex &lh, int)
- Expression postfix decrement.*
- numeric & [GiNaC::operator++](#) (numeric &rh)
- Numeric prefix increment.*
- numeric & [GiNaC::operator--](#) (numeric &rh)
- Numeric prefix decrement.*
- const numeric [GiNaC::operator++](#) (numeric &lh, int)
- Numeric postfix increment.*
- const numeric [GiNaC::operator--](#) (numeric &lh, int)
- Numeric postfix decrement.*
- const relational [GiNaC::operator==](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator!=](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator<](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator<=](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator>](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator>=](#) (const ex &lh, const ex &rh)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const ex &e)
- std::istream & [GiNaC::operator>>](#) (std::istream &is, ex &e)
- std::ostream & [GiNaC::dflt](#) (std::ostream &os)
- std::ostream & [GiNaC::latex](#) (std::ostream &os)
- std::ostream & [GiNaC::python](#) (std::ostream &os)
- std::ostream & [GiNaC::python\\_repr](#) (std::ostream &os)
- std::ostream & [GiNaC::tree](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_float](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_double](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_cl\\_N](#) (std::ostream &os)
- std::ostream & [GiNaC::index\\_dimensions](#) (std::ostream &os)
- std::ostream & [GiNaC::no\\_index\\_dimensions](#) (std::ostream &os)

### 7.67.1 Detailed Description

Interface to [GiNaC](#)'s overloaded operators.

## 7.68 power.cpp File Reference

Implementation of [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

```
#include "power.h"
#include "expairseq.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "numeric.h"
#include "constant.h"
#include "operators.h"
#include "inifcns.h"
#include "matrix.h"
#include "indexed.h"
#include "symbol.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "relational.h"
#include "compiler.h"
#include <iostream>
#include <limits>
#include <stdexcept>
#include <vector>
#include <algorithm>
```

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (power, basic, print\_func< print\_dflt >(&power::do\_print\_dflt). print\_func< print\_latex >(&power::do\_print\_latex). print\_func< print\_csrc >(&power::do\_print\_csrc). print\_func< print\_python >(&power::do\_print\_python). print\_func< print\_python\_repr >(&power::do\_print\_python\_repr). print\_func< print\_csrc\_cl\_N >(&power::do\_print\_csrc\_cl\_N)) power
- static void [GiNaC::print\\_sym\\_pow](#) (const print\_context &c, const symbol &x, int exp)
- bool [GiNaC::tryfactsubs](#) (const ex &origfactor, const ex &patternfactor, int &nummatches, exmap &repls)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (power)

### 7.68.1 Detailed Description

Implementation of [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

## 7.69 power.h File Reference

Interface to [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

### Classes

- class [GiNaC::power](#)

*This class holds a two-component object, a basis and an exponent representing exponentiation.*

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (power)
- ex [GiNaC::pow](#) (const ex &b, const ex &e)  
*Symbolic exponentiation.*
- template<typename T1, typename T2 >  
ex [GiNaC::pow](#) (const T1 &b, const T2 &e)
- ex [GiNaC::sqrt](#) (const ex &a)

*Square root expression.*

#### 7.69.1 Detailed Description

Interface to [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

## 7.70 print.cpp File Reference

Implementation of helper classes for expression output.

```
#include "print.h"
#include <iostream>
```

### Namespaces

- [GiNaC](#)

## Variables

- unsigned [GiNaC::next\\_print\\_context\\_id](#) = 0  
Next free ID for [print\\_context](#) types.

### 7.70.1 Detailed Description

Implementation of helper classes for expression output.

## 7.71 print.h File Reference

Definition of helper classes for expression output.

```
#include "class_info.h"
#include <iosfwd>
#include <memory>
#include <string>
```

## Classes

- class [GiNaC::print\\_context\\_options](#)  
*This class stores information about a registered [print\\_context](#) class.*
- class [GiNaC::print\\_options](#)  
*Flags to control the behavior of a [print\\_context](#).*
- class [GiNaC::print\\_context](#)  
*Base class for [print\\_contexts](#).*
- class [GiNaC::print\\_dflt](#)  
*Context for default (ginsh-parsable) output.*
- class [GiNaC::print\\_latex](#)  
*Context for latex-parsable output.*
- class [GiNaC::print\\_python](#)  
*Context for python pretty-print output.*
- class [GiNaC::print\\_python\\_repr](#)  
*Context for python-parsable output.*
- class [GiNaC::print\\_tree](#)  
*Context for tree-like output for debugging.*
- class [GiNaC::print\\_csrc](#)  
*Base context for C source output.*
- class [GiNaC::print\\_csrc\\_float](#)  
*Context for C source output using float precision.*
- class [GiNaC::print\\_csrc\\_double](#)  
*Context for C source output using double precision.*
- class [GiNaC::print\\_csrc\\_cl\\_N](#)  
*Context for C source output using CLN numbers.*
- class [GiNaC::print\\_functor\\_impl](#)  
*Base class for [print\\_functor](#) handlers.*
- class [GiNaC::print\\_ptrfun\\_handler< T, C >](#)  
*[print\\_functor](#) handler for pointer-to-functions of class *T*, context type *C**
- class [GiNaC::print\\_memfun\\_handler< T, C >](#)  
*[print\\_functor](#) handler for member functions of class *T*, context type *C**
- class [GiNaC::print\\_functor](#)  
*This class represents a print method for a certain algebraic class and [print\\_context](#) type.*

## Namespaces

- [GiNaC](#)

## Macros

- `#define GINAC_DECLARE_PRINT_CONTEXT_COMMON(classname)`  
*Common part of GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE and GINAC\_DECLARE\_PRINT\_CONTEXT\_DERIVED.*
- `#define GINAC_DECLARE_PRINT_CONTEXT_BASE(classname)`
- `#define GINAC_DECLARE_PRINT_CONTEXT(classname, supname)`  
*Macro for inclusion in the declaration of a print\_context class.*
- `#define GINAC_IMPLEMENT_PRINT_CONTEXT(classname, supname)`  
*Macro for inclusion in the implementation of each print\_context class.*

## Typedefs

- `typedef class_info< print_context_options > GiNaC::print\_context\_class\_info`

## Functions

- `template<class T >`  
`bool GiNaC::is\_a (const print_context &obj)`  
*Check if obj is a T, including base classes.*

### 7.71.1 Detailed Description

Definition of helper classes for expression output.

### 7.71.2 Macro Definition Documentation

#### 7.71.2.1 GINAC\_DECLARE\_PRINT\_CONTEXT\_COMMON

```
#define GINAC_DECLARE_PRINT_CONTEXT_COMMON(  
    classname )
```

#### Value:

```
public: \  
    friend class function_options; \  
    friend class registered_class_options; \  
    static const GiNaC::print\_context\_class\_info &get_class_info_static(); \  
    classname();
```

Common part of GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE and GINAC\_DECLARE\_PRINT\_CONTEXT\_DERIVED.

## 7.71.2.2 GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE

```
#define GINAC_DECLARE_PRINT_CONTEXT_BASE(  
    classname )
```

**Value:**

```
GINAC_DECLARE_PRINT_CONTEXT_COMMON(classname) \
    virtual const GiNaC::print_context_class_info &get_class_info() const {  
        return classname::get_class_info_static(); } \
    virtual const char *class_name() const { return classname::get_class_info_static().options.get_name();  
    } \
    virtual classname * duplicate() const { return new classname(*this); } \
private:
```

## 7.71.2.3 GINAC\_DECLARE\_PRINT\_CONTEXT

```
#define GINAC_DECLARE_PRINT_CONTEXT(  
    classname,  
    supername )
```

**Value:**

```
GINAC_DECLARE_PRINT_CONTEXT_COMMON(classname) \
    typedef supername inherited; \
    const GiNaC::print_context_class_info &get_class_info() const override {  
        return classname::get_class_info_static(); } \
    const char *class_name() const override { return classname::get_class_info_static().options.get_name();  
    } \
    classname * duplicate() const override { return new classname(*this); } \
private:
```

Macro for inclusion in the declaration of a `print_context` class.

It declares some functions that are common to all classes derived from 'print\_context' as well as all required stuff for the [GiNaC](#) registry.

## 7.71.2.4 GINAC\_IMPLEMENT\_PRINT\_CONTEXT

```
#define GINAC_IMPLEMENT_PRINT_CONTEXT(  
    classname,  
    supername )
```

**Value:**

```
const GiNaC::print_context_class_info &classname::get_class_info_static() \
{ \
    static GiNaC::print_context_class_info reg_info =  
        GiNaC::print_context_class_info(  
            GiNaC::print_context_options(#classname, #supername,  
            GiNaC::next_print_context_id++)); \
    return reg_info; \
}
```

Macro for inclusion in the implementation of each `print_context` class.

## 7.72 pseries.cpp File Reference

Implementation of class for extended truncated power series and methods for series expansion.

```
#include "pseries.h"
#include "add.h"
#include "inifcns.h"
#include "lst.h"
#include "mul.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "symbol.h"
#include "integral.h"
#include "archive.h"
#include "utils.h"
#include <limits>
#include <numeric>
#include <stdexcept>
```

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (pseries, basic, print\_func< print\_context >(&pseries::do\_print). print\_func< print\_latex >(&pseries::do\_print\_latex). print\_func< print\_tree >(&pseries::do\_print\_tree). print\_func< print\_python >(&pseries::do\_print\_python). print\_func< print\_python\_repr >(&pseries::do\_print\_python\_repr)) pseries
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (pseries)

### 7.72.1 Detailed Description

Implementation of class for extended truncated power series and methods for series expansion.

## 7.73 pseries.h File Reference

Interface to class for extended truncated power series.

```
#include "basic.h"
#include "expairseq.h"
```

### Classes

- class [GiNaC::pseries](#)

*This class holds a extended truncated power series (positive and negative integer powers).*



## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (pseries)
- ex [GiNaC::series\\_to\\_poly](#) (const ex &e)  
*Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.*
- bool [GiNaC::is\\_terminating](#) (const pseries &s)

### 7.73.1 Detailed Description

Interface to class for extended truncated power series.

## 7.74 ptr.h File Reference

Reference-counted pointer template.

```
#include "assertion.h"
#include <cstdint>
#include <functional>
#include <iosfwd>
```

## Classes

- class [GiNaC::refcounted](#)  
*Base class for reference-counted objects.*
- class [GiNaC::ptr< T >](#)  
*Class of (intrusively) reference-counted pointers that support copy-on-write semantics.*
- struct [std::less< GiNaC::ptr< T > >](#)  
*Specialization of std::less for ptr<T> to enable ordering of ptr<T> objects (e.g.*

## Namespaces

- [GiNaC](#)
- [std](#)

### 7.74.1 Detailed Description

Reference-counted pointer template.

## 7.75 registrar.cpp File Reference

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

```
#include "registrar.h"
#include <map>
#include <stdexcept>
#include <string>
```

### Namespaces

- [GiNaC](#)

#### 7.75.1 Detailed Description

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

## 7.76 registrar.h File Reference

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

```
#include "class_info.h"
#include "print.h"
#include <list>
#include <string>
#include <typeinfo>
#include <vector>
```

### Classes

- class [GiNaC::container< C >](#)  
*Wrapper template for making [GiNaC](#) classes out of STL containers.*
- struct [GiNaC::return\\_type\\_t](#)  
*To distinguish between different kinds of non-commutative objects.*
- class [GiNaC::registered\\_class\\_options](#)  
*This class stores information about a registered [GiNaC](#) class.*

### Namespaces

- [GiNaC](#)

## Macros

- `#define GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname)`  
*Common part of GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS and GINAC\_DECLARE\_REGISTERED\_CLASS.*
- `#define GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS(classname, supertype)`  
*Primary macro for inclusion in the declaration of each registered class.*
- `#define GINAC_DECLARE_REGISTERED_CLASS(classname, supertype)`  
*Macro for inclusion in the declaration of each registered class.*
- `#define GINAC_IMPLEMENT_REGISTERED_CLASS(classname, supertype) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC::registered_class_options(#classname, #supertype, typeid(classname)));`  
*Macro for inclusion in the implementation of each registered class.*
- `#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT(classname, supertype, options) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC::registered_class_options(#classname, #supertype, typeid(classname)).options);`  
*Macro for inclusion in the implementation of each registered class.*
- `#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T(classname, supertype, options) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC::registered_class_options(#classname, #supertype, typeid(classname)).options);`  
*Macro for inclusion in the implementation of each registered class.*

## Typedefs

- `typedef class_info< registered_class_options > GiNaC::registered_class_info`

## Functions

- `template<typename T> return_type_t GiNaC::make_return_type_t (const unsigned rl=0)`
- `template<class Alg, class Ctx, class T, class C> void GiNaC::set_print_func (void f(const T &, const C &c, unsigned))`  
*Add or replace a print method.*
- `template<class Alg, class Ctx, class T, class C> void GiNaC::set_print_func (void(T::*f)(const C &, unsigned))`  
*Add or replace a print method.*

### 7.76.1 Detailed Description

GiNaC's class registrar (for class basic and all classes derived from it).

### 7.76.2 Macro Definition Documentation

### 7.76.2.1 GINAC\_DECLARE\_REGISTERED\_CLASS\_COMMON

```
#define GINAC_DECLARE_REGISTERED_CLASS_COMMON(  
    classname )
```

**Value:**

```
private: \  
    static GiNaC::registered_class_info reg_info; \  
public: \  
    static GiNaC::registered_class_info &get_class_info_static() { return  
        reg_info; } \  
    class visitor { \  
    public: \  
        virtual void visit(const classname &) = 0; \  
        virtual ~visitor() {}; \  
};
```

Common part of GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS and GINAC\_DECLARE\_REGISTERED\_CLASS.

### 7.76.2.2 GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS

```
#define GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS(  
    classname,  
    supername )
```

**Value:**

```
GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname) \  
    typedef supername inherited; \  
    virtual const GiNaC::registered_class_info &get_class_info() const { return  
        classname::get_class_info_static(); } \  
    virtual GiNaC::registered_class_info &get_class_info() { return  
        classname::get_class_info_static(); } \  
    virtual const char *class_name() const { return classname::get_class_info_static().options.get_name();  
    } \  
private:
```

Primary macro for inclusion in the declaration of each registered class.

### 7.76.2.3 GINAC\_DECLARE\_REGISTERED\_CLASS

```
#define GINAC_DECLARE_REGISTERED_CLASS(  
    classname,  
    supername )
```

**Value:**

```
GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname) \
template<class B, typename... Args> friend B & dynallocate(Args &&... args); \
typedef typename inherited; \
classname(); \
classname * duplicate() const override { \
    classname * bp = new classname(*this); \
    bp->setflag(GiNaC::status_flags::dynallocated); \
    return bp; \
} \
void accept(GiNaC::visitor & v) const override \
{ \
    if (visitor *p = dynamic_cast<visitor *>(&v)) \
        p->visit(*this); \
    else \
        inherited::accept(v); \
} \
const GiNaC::registered_class_info &get_class_info() const override { \
    return classname::get_class_info_static(); } \
GiNaC::registered_class_info &get_class_info() override { return \
    classname::get_class_info_static(); } \
const char *class_name() const override { return classname::get_class_info_static().options.get_name(); \
} \
protected: \
    int compare_same_type(const GiNaC::basic & other) const override; \
private:
```

Macro for inclusion in the declaration of each registered class.

It declares some functions that are common to all classes derived from 'basic' as well as all required stuff for the [GiNaC](#) class registry (mainly needed for archiving).

#### 7.76.2.4 GINAC\_IMPLEMENT\_REGISTERED\_CLASS

```
#define GINAC_IMPLEMENT_REGISTERED_CLASS( \
    classname, \
    supertype ) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC:: \
    #supertype, typeid(classname));
```

Macro for inclusion in the implementation of each registered class.

#### 7.76.2.5 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT

```
#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT( \
    classname, \
    supertype, \
    options ) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC:: \
    #supertype, typeid(classname)).options;
```

Macro for inclusion in the implementation of each registered class.

Additional options can be specified.

#### 7.76.2.6 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T

```
#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T( \
    classname, \
    supertype, \
    options ) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC:: \
    #supertype, typeid(classname)).options;
```

Macro for inclusion in the implementation of each registered class.

Additional options can be specified.

## 7.77 relational.cpp File Reference

Implementation of relations between expressions.

```
#include "relational.h"
#include "operators.h"
#include "numeric.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <iostream>
#include <stdexcept>
```

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (relational, basic, print\_func< print\_context >(&relational::do\_print). print\_func< print\_tree >(&relational::do\_print\_tree). print\_func< print\_python\_repr >(&relational::do\_print\_python\_repr)) relational
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (relational)
- static void [GiNaC::print\\_operator](#) (const print\_context &c, relational::operators o)

### 7.77.1 Detailed Description

Implementation of relations between expressions.

## 7.78 relational.h File Reference

Interface to relations between expressions.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

### Classes

- class [GiNaC::relational](#)  
*This class holds a relation consisting of two expressions and a logical relation between them.*
- struct [GiNaC::relational::safe\\_bool\\_helper](#)

### Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (relational)

### 7.78.1 Detailed Description

Interface to relations between expressions.

## 7.79 remember.cpp File Reference

Implementation of helper classes for using the remember option in [GiNaC](#) functions.

```
#include "function.h"
#include "utils.h"
#include "remember.h"
#include <stdexcept>
```

## Namespaces

- [GiNaC](#)

### 7.79.1 Detailed Description

Implementation of helper classes for using the remember option in [GiNaC](#) functions.

## 7.80 remember.h File Reference

Interface to helper classes for using the remember option in [GiNaC](#) functions.

```
#include <iosfwd>
#include <list>
#include <vector>
```

## Classes

- class [GiNaC::remember\\_table\\_entry](#)  
*A single entry in the remember table of a function.*
- class [GiNaC::remember\\_table\\_list](#)  
*A list of entries in the remember table having some least significant bits of the hashvalue in common.*
- class [GiNaC::remember\\_table](#)  
*The remember table is organized like an n-fold associative cache in a microprocessor.*

## Namespaces

- [GiNaC](#)

### 7.80.1 Detailed Description

Interface to helper classes for using the remember option in [GiNaC](#) functions.

## 7.81 structure.h File Reference

Wrapper template for making [GiNaC](#) classes out of C++ structures.

```
#include "ex.h"
#include "ncmul.h"
#include "numeric.h"
#include "operators.h"
#include "print.h"
#include <functional>
```

## Classes

- class [GiNaC::compare\\_all\\_equal< T >](#)  
*Comparison policy: all structures of one type are equal.*
- class [GiNaC::compare\\_std\\_less< T >](#)  
*Comparison policy: use std::equal\_to/std::less (defaults to operators == and <) to compare structures.*
- class [GiNaC::compare\\_bitwise< T >](#)  
*Comparison policy: use bit-wise comparison to compare structures.*
- class [GiNaC::structure< T, ComparisonPolicy >](#)  
*Wrapper template for making [GiNaC](#) classes out of C++ structures.*
- class [GiNaC::structure< T, ComparisonPolicy >](#)  
*Wrapper template for making [GiNaC](#) classes out of C++ structures.*

## Namespaces

- [GiNaC](#)

### 7.81.1 Detailed Description

Wrapper template for making [GiNaC](#) classes out of C++ structures.



## 7.82 symbol.cpp File Reference

Implementation of [GiNaC](#)'s symbolic objects.

```
#include "symbol.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include "inifcns.h"
#include <map>
#include <stdexcept>
#include <string>
```

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (symbol, basic, print\_func< print\_context >(&symbol::do\_print). print\_func< print\_latex >(&symbol::do\_print\_latex). print\_func< print\_tree >(&symbol::do\_print\_tree). print\_func< print\_python\_repr >(&symbol::do\_print\_python\_repr)) symbol
- static const std::string & [GiNaC::get\\_default\\_TeX\\_name](#) (const std::string &name)  
*Return default TeX name for symbol.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (symbol)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (realsymbol)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (possymbol)

### 7.82.1 Detailed Description

Implementation of [GiNaC](#)'s symbolic objects.

## 7.83 symbol.h File Reference

Interface to [GiNaC](#)'s symbolic objects.

```
#include "basic.h"
#include "ex.h"
#include "ptr.h"
#include "archive.h"
#include <string>
#include <typeinfo>
```

## Classes

- class [GiNaC::symbol](#)  
*Basic CAS symbol.*
- class [GiNaC::realsymbol](#)  
*Specialization of symbol to real domain.*
- class [GiNaC::possymbol](#)  
*Specialization of symbol to real positive domain.*

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (symbol)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (realsymbol)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (possymbol)

### 7.83.1 Detailed Description

Interface to [GiNaC](#)'s symbolic objects.

## 7.84 symmetry.cpp File Reference

Implementation of [GiNaC](#)'s symmetry definitions.

```
#include "symmetry.h"
#include "lst.h"
#include "add.h"
#include "numeric.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <functional>
#include <iostream>
#include <limits>
#include <stdexcept>
```

## Classes

- class [GiNaC::sy\\_is\\_less](#)
- class [GiNaC::sy\\_swap](#)

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (symmetry, basic, print\_func< print\_context >(&symmetry::do\_print). print\_func< print\_tree >(&symmetry::do\_print\_tree)) symmetry
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (symmetry)
- static const symmetry & [GiNaC::index0](#) ()
- static const symmetry & [GiNaC::index1](#) ()
- static const symmetry & [GiNaC::index2](#) ()
- static const symmetry & [GiNaC::index3](#) ()
- const symmetry & [GiNaC::not\\_symmetric](#) ()
- const symmetry & [GiNaC::symmetric2](#) ()
- const symmetry & [GiNaC::symmetric3](#) ()
- const symmetry & [GiNaC::symmetric4](#) ()
- const symmetry & [GiNaC::antisymmetric2](#) ()
- const symmetry & [GiNaC::antisymmetric3](#) ()
- const symmetry & [GiNaC::antisymmetric4](#) ()
- int [GiNaC::canonicalize](#) (exvector::iterator v, const symmetry &symm)  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- static ex [GiNaC::symm](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator [last](#), bool asymmetric)
- ex [GiNaC::symmetrize](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Symmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::antisymmetrize](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::symmetrize\\_cyclic](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*

### 7.84.1 Detailed Description

Implementation of [GiNaC](#)'s symmetry definitions.

## 7.85 symmetry.h File Reference

Interface to [GiNaC](#)'s symmetry definitions.

```
#include "ex.h"
#include "archive.h"
#include <set>
```

## Classes

- class [GiNaC::symmetry](#)  
*This class describes the symmetry of a group of indices.*

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (symmetry)
- symmetry [GiNaC::sy\\_none](#) ()
- symmetry [GiNaC::sy\\_none](#) (const symmetry &c1, const symmetry &c2)
- symmetry [GiNaC::sy\\_none](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3)
- symmetry [GiNaC::sy\\_none](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3, const symmetry &c4)
- symmetry [GiNaC::sy\\_symm](#) ()
- symmetry [GiNaC::sy\\_symm](#) (const symmetry &c1, const symmetry &c2)
- symmetry [GiNaC::sy\\_symm](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3)
- symmetry [GiNaC::sy\\_symm](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3, const symmetry &c4)
- symmetry [GiNaC::sy\\_anti](#) ()
- symmetry [GiNaC::sy\\_anti](#) (const symmetry &c1, const symmetry &c2)
- symmetry [GiNaC::sy\\_anti](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3)
- symmetry [GiNaC::sy\\_anti](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3, const symmetry &c4)
- symmetry [GiNaC::sy\\_cycl](#) ()
- symmetry [GiNaC::sy\\_cycl](#) (const symmetry &c1, const symmetry &c2)
- symmetry [GiNaC::sy\\_cycl](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3)
- symmetry [GiNaC::sy\\_cycl](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3, const symmetry &c4)
- const symmetry & [GiNaC::not\\_symmetric](#) ()
- const symmetry & [GiNaC::symmetric2](#) ()
- const symmetry & [GiNaC::symmetric3](#) ()
- const symmetry & [GiNaC::symmetric4](#) ()
- const symmetry & [GiNaC::antisymmetric2](#) ()
- const symmetry & [GiNaC::antisymmetric3](#) ()
- const symmetry & [GiNaC::antisymmetric4](#) ()
- int [GiNaC::canonicalize](#) (exvector::iterator v, const symmetry &symm)  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- ex [GiNaC::symmetrize](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator last)  
*Symmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::symmetrize](#) (const ex &e, const exvector &v)  
*Symmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::antisymmetrize](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator last)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::antisymmetrize](#) (const ex &e, const exvector &v)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::symmetrize\\_cyclic](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator last)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*
- ex [GiNaC::symmetrize\\_cyclic](#) (const ex &e, const exvector &v)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*

### 7.85.1 Detailed Description

Interface to [GiNaC](#)'s symmetry definitions.

## 7.86 tensor.cpp File Reference

Implementation of [GiNaC](#)'s special tensors.

```
#include "tensor.h"
#include "idx.h"
#include "indexed.h"
#include "symmetry.h"
#include "relational.h"
#include "operators.h"
#include "lst.h"
#include "numeric.h"
#include "matrix.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
#include <stdexcept>
#include <vector>
```

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (tensdelta, tensor, print\_func< print\_dflt >(&tensdelta::do\_print). print\_func< print\_latex >(&tensdelta::do\_print\_latex)) [GINAC\\_IMPLEMENT\\_](#)↵  
[REGISTERED\\_CLASS\\_OPT](#)(tensmetric
- [GiNaC::print\\_func< print\\_dflt >](#) (&tensmetric::do\_print). print\_func< print\_latex >(&tensmetric
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (minkmetric)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (tensepsilon)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (tensdelta)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (tensmetric)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (spinmetric)
- ex [GiNaC::delta\\_tensor](#) (const ex &i1, const ex &i2)  
*Create a delta tensor with specified indices.*
- ex [GiNaC::metric\\_tensor](#) (const ex &i1, const ex &i2)  
*Create a symmetric metric tensor with specified indices.*
- ex [GiNaC::lorentz\\_g](#) (const ex &i1, const ex &i2, bool pos\_sig=false)  
*Create a Minkowski metric tensor with specified indices.*
- ex [GiNaC::spinor\\_metric](#) (const ex &i1, const ex &i2)  
*Create a spinor metric tensor with specified indices.*
- ex [GiNaC::epsilon\\_tensor](#) (const ex &i1, const ex &i2)  
*Create an epsilon tensor in a Euclidean space with two indices.*
- ex [GiNaC::epsilon\\_tensor](#) (const ex &i1, const ex &i2, const ex &i3)  
*Create an epsilon tensor in a Euclidean space with three indices.*
- ex [GiNaC::lorentz\\_eps](#) (const ex &i1, const ex &i2, const ex &i3, const ex &i4, bool pos\_sig=false)  
*Create an epsilon tensor in a Minkowski space with four indices.*

### 7.86.1 Detailed Description

Implementation of [GiNaC](#)'s special tensors.

## 7.87 tensor.h File Reference

Interface to [GiNaC](#)'s special tensors.

```
#include "ex.h"
#include "archive.h"
```

### Classes

- class [GiNaC::tensor](#)  
*This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.*
- class [GiNaC::tensdelta](#)  
*This class represents the delta tensor.*
- class [GiNaC::tensmetric](#)  
*This class represents a general metric tensor which can be used to raise/lower indices.*
- class [GiNaC::minkmetric](#)  
*This class represents a Minkowski metric tensor.*
- class [GiNaC::spinmetric](#)  
*This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.*
- class [GiNaC::tensepsilon](#)  
*This class represents the totally antisymmetric epsilon tensor.*

### Namespaces

- [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (tensdelta)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (tensmetric)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (minkmetric)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (spinmetric)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (tensepsilon)
- ex [GiNaC::delta\\_tensor](#) (const ex &i1, const ex &i2)  
*Create a delta tensor with specified indices.*
- ex [GiNaC::metric\\_tensor](#) (const ex &i1, const ex &i2)  
*Create a symmetric metric tensor with specified indices.*
- ex [GiNaC::lorentz\\_g](#) (const ex &i1, const ex &i2, bool pos\_sig=false)  
*Create a Minkowski metric tensor with specified indices.*
- ex [GiNaC::spinor\\_metric](#) (const ex &i1, const ex &i2)  
*Create a spinor metric tensor with specified indices.*
- ex [GiNaC::epsilon\\_tensor](#) (const ex &i1, const ex &i2)  
*Create an epsilon tensor in a Euclidean space with two indices.*
- ex [GiNaC::epsilon\\_tensor](#) (const ex &i1, const ex &i2, const ex &i3)  
*Create an epsilon tensor in a Euclidean space with three indices.*
- ex [GiNaC::lorentz\\_eps](#) (const ex &i1, const ex &i2, const ex &i3, const ex &i4, bool pos\_sig=false)  
*Create an epsilon tensor in a Minkowski space with four indices.*

### 7.87.1 Detailed Description

Interface to [GiNaC](#)'s special tensors.

## 7.88 utils.cpp File Reference

Implementation of several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

```
#include "ex.h"
#include "numeric.h"
#include "utils.h"
#include "version.h"
```

### Namespaces

- [GiNaC](#)

### Functions

- unsigned [GiNaC::log2](#) (unsigned n)  
*Integer binary logarithm.*
- const numeric [GiNaC::multinomial\\_coefficient](#) (const std::vector< unsigned > &p)  
*Compute the multinomial coefficient  $n!/(p1!*p2!*...*pk!)$  where  $n = p1+p2+...+pk$ , i.e.*

### Variables

- const int [GiNaC::version\\_major](#) = GINACLIB\_MAJOR\_VERSION
- const int [GiNaC::version\\_minor](#) = GINACLIB\_MINOR\_VERSION
- const int [GiNaC::version\\_micro](#) = GINACLIB\_MICRO\_VERSION
- const numeric \* [GiNaC::\\_num\\_120\\_p](#)
- const ex [GiNaC::\\_ex\\_120](#) = \_ex\_120
- const numeric \* [GiNaC::\\_num\\_60\\_p](#)
- const ex [GiNaC::\\_ex\\_60](#) = \_ex\_60
- const numeric \* [GiNaC::\\_num\\_48\\_p](#)
- const ex [GiNaC::\\_ex\\_48](#) = \_ex\_48
- const numeric \* [GiNaC::\\_num\\_30\\_p](#)
- const ex [GiNaC::\\_ex\\_30](#) = \_ex\_30
- const numeric \* [GiNaC::\\_num\\_25\\_p](#)
- const ex [GiNaC::\\_ex\\_25](#) = \_ex\_25
- const numeric \* [GiNaC::\\_num\\_24\\_p](#)
- const ex [GiNaC::\\_ex\\_24](#) = \_ex\_24
- const numeric \* [GiNaC::\\_num\\_20\\_p](#)
- const ex [GiNaC::\\_ex\\_20](#) = \_ex\_20
- const numeric \* [GiNaC::\\_num\\_18\\_p](#)
- const ex [GiNaC::\\_ex\\_18](#) = \_ex\_18
- const numeric \* [GiNaC::\\_num\\_15\\_p](#)
- const ex [GiNaC::\\_ex\\_15](#) = \_ex\_15

- const numeric \* [GiNaC::\\_num\\_12\\_p](#)
- const ex [GiNaC::\\_ex\\_12](#) = [\\_ex\\_12](#)
- const numeric \* [GiNaC::\\_num\\_11\\_p](#)
- const ex [GiNaC::\\_ex\\_11](#) = [\\_ex\\_11](#)
- const numeric \* [GiNaC::\\_num\\_10\\_p](#)
- const ex [GiNaC::\\_ex\\_10](#) = [\\_ex\\_10](#)
- const numeric \* [GiNaC::\\_num\\_9\\_p](#)
- const ex [GiNaC::\\_ex\\_9](#) = [\\_ex\\_9](#)
- const numeric \* [GiNaC::\\_num\\_8\\_p](#)
- const ex [GiNaC::\\_ex\\_8](#) = [\\_ex\\_8](#)
- const numeric \* [GiNaC::\\_num\\_7\\_p](#)
- const ex [GiNaC::\\_ex\\_7](#) = [\\_ex\\_7](#)
- const numeric \* [GiNaC::\\_num\\_6\\_p](#)
- const ex [GiNaC::\\_ex\\_6](#) = [\\_ex\\_6](#)
- const numeric \* [GiNaC::\\_num\\_5\\_p](#)
- const ex [GiNaC::\\_ex\\_5](#) = [\\_ex\\_5](#)
- const numeric \* [GiNaC::\\_num\\_4\\_p](#)
- const ex [GiNaC::\\_ex\\_4](#) = [\\_ex\\_4](#)
- const numeric \* [GiNaC::\\_num\\_3\\_p](#)
- const ex [GiNaC::\\_ex\\_3](#) = [\\_ex\\_3](#)
- const numeric \* [GiNaC::\\_num\\_2\\_p](#)
- const ex [GiNaC::\\_ex\\_2](#) = [\\_ex\\_2](#)
- const numeric \* [GiNaC::\\_num\\_1\\_p](#)
- const ex [GiNaC::\\_ex\\_1](#) = [\\_ex\\_1](#)
- const numeric \* [GiNaC::\\_num\\_1\\_2\\_p](#)
- const ex [GiNaC::\\_ex\\_1\\_2](#) = [\\_ex\\_1\\_2](#)
- const numeric \* [GiNaC::\\_num\\_1\\_3\\_p](#)
- const ex [GiNaC::\\_ex\\_1\\_3](#) = [\\_ex\\_1\\_3](#)
- const numeric \* [GiNaC::\\_num\\_1\\_4\\_p](#)
- const ex [GiNaC::\\_ex\\_1\\_4](#) = [\\_ex\\_1\\_4](#)
- const numeric \* [GiNaC::\\_num0\\_p](#)
- const ex [GiNaC::\\_ex0](#) = [\\_ex0](#)
- const numeric \* [GiNaC::\\_num1\\_4\\_p](#)
- const ex [GiNaC::\\_ex1\\_4](#) = [\\_ex1\\_4](#)
- const numeric \* [GiNaC::\\_num1\\_3\\_p](#)
- const ex [GiNaC::\\_ex1\\_3](#) = [\\_ex1\\_3](#)
- const numeric \* [GiNaC::\\_num1\\_2\\_p](#)
- const ex [GiNaC::\\_ex1\\_2](#) = [\\_ex1\\_2](#)
- const numeric \* [GiNaC::\\_num1\\_p](#)
- const ex [GiNaC::\\_ex1](#) = [\\_ex1](#)
- const numeric \* [GiNaC::\\_num2\\_p](#)
- const ex [GiNaC::\\_ex2](#) = [\\_ex2](#)
- const numeric \* [GiNaC::\\_num3\\_p](#)
- const ex [GiNaC::\\_ex3](#) = [\\_ex3](#)
- const numeric \* [GiNaC::\\_num4\\_p](#)
- const ex [GiNaC::\\_ex4](#) = [\\_ex4](#)
- const numeric \* [GiNaC::\\_num5\\_p](#)
- const ex [GiNaC::\\_ex5](#) = [\\_ex5](#)
- const numeric \* [GiNaC::\\_num6\\_p](#)
- const ex [GiNaC::\\_ex6](#) = [\\_ex6](#)
- const numeric \* [GiNaC::\\_num7\\_p](#)
- const ex [GiNaC::\\_ex7](#) = [\\_ex7](#)
- const numeric \* [GiNaC::\\_num8\\_p](#)
- const ex [GiNaC::\\_ex8](#) = [\\_ex8](#)
- const numeric \* [GiNaC::\\_num9\\_p](#)



- const ex [GiNaC::\\_ex9](#) = \_ex9
- const numeric \* [GiNaC::\\_num10\\_p](#)
- const ex [GiNaC::\\_ex10](#) = \_ex10
- const numeric \* [GiNaC::\\_num11\\_p](#)
- const ex [GiNaC::\\_ex11](#) = \_ex11
- const numeric \* [GiNaC::\\_num12\\_p](#)
- const ex [GiNaC::\\_ex12](#) = \_ex12
- const numeric \* [GiNaC::\\_num15\\_p](#)
- const ex [GiNaC::\\_ex15](#) = \_ex15
- const numeric \* [GiNaC::\\_num18\\_p](#)
- const ex [GiNaC::\\_ex18](#) = \_ex18
- const numeric \* [GiNaC::\\_num20\\_p](#)
- const ex [GiNaC::\\_ex20](#) = \_ex20
- const numeric \* [GiNaC::\\_num24\\_p](#)
- const ex [GiNaC::\\_ex24](#) = \_ex24
- const numeric \* [GiNaC::\\_num25\\_p](#)
- const ex [GiNaC::\\_ex25](#) = \_ex25
- const numeric \* [GiNaC::\\_num30\\_p](#)
- const ex [GiNaC::\\_ex30](#) = \_ex30
- const numeric \* [GiNaC::\\_num48\\_p](#)
- const ex [GiNaC::\\_ex48](#) = \_ex48
- const numeric \* [GiNaC::\\_num60\\_p](#)
- const ex [GiNaC::\\_ex60](#) = \_ex60
- const numeric \* [GiNaC::\\_num120\\_p](#)
- const ex [GiNaC::\\_ex120](#) = \_ex120

### 7.88.1 Detailed Description

Implementation of several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

## 7.89 utils.h File Reference

Interface to several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

```
#include "assertion.h"
#include <functional>
#include <cstdlib>
#include <string>
```

## Classes

- class [GiNaC::dunno](#)  
*Exception class thrown by functions to signal unimplemented functionality so the expression may just be .hold()*
- class [GiNaC::basic\\_partition\\_generator](#)  
*Base class for generating all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts in non-decreasing order.*
- struct [GiNaC::basic\\_partition\\_generator::mpartition2](#)
- class [GiNaC::partition\\_with\\_zero\\_parts\\_generator](#)  
*Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (including zero parts) in non-decreasing order.*
- class [GiNaC::partition\\_generator](#)  
*Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (not including zero parts) in non-decreasing order.*
- class [GiNaC::composition\\_generator](#)  
*Generate all compositions of a partition of an integer  $n$ , starting with the compositions which has non-decreasing order.*
- struct [GiNaC::composition\\_generator::coolmulti](#)
- struct [GiNaC::composition\\_generator::coolmulti::element](#)

## Namespaces

- [GiNaC](#)

## Macros

- #define [DEFAULT\\_CTOR](#)(classname) classname::classname() { setflag(status\_flags::evaluated | status\_↔ flags::expanded); }
- #define [DEFAULT\\_COMPARE](#)(classname)
- #define [DEFAULT\\_PRINT](#)(classname, text)
- #define [DEFAULT\\_PRINT\\_LATEX](#)(classname, text, latex)

## Functions

- unsigned [GiNaC::log2](#) (unsigned  $n$ )  
*Integer binary logarithm.*
- unsigned [GiNaC::rotate\\_left](#) (unsigned  $n$ )  
*Rotate bits of unsigned value by one bit to the left.*
- template<class T >  
int [GiNaC::compare\\_pointers](#) (const T \*a, const T \*b)  
*Compare two pointers (just to establish some sort of canonical order).*
- unsigned [GiNaC::golden\\_ratio\\_hash](#) (uintptr\_t  $n$ )  
*Truncated multiplication with golden ratio, for computing hash values.*
- template<class It >  
int [GiNaC::permutation\\_sign](#) (It first, It last)
- template<class It, class Cmp, class Swap >  
int [GiNaC::permutation\\_sign](#) (It first, It last, Cmp comp, Swap swapit)
- template<class It, class Cmp, class Swap >  
void [GiNaC::shaker\\_sort](#) (It first, It last, Cmp comp, Swap swapit)
- template<class It, class Swap >  
void [GiNaC::cyclic\\_permutation](#) (It first, It last, It new\_first, Swap swapit)
- const numeric [GiNaC::multinomial\\_coefficient](#) (const std::vector< unsigned > &p)  
*Compute the multinomial coefficient  $n!/(p_1! * p_2! * \dots * p_k!)$  where  $n = p_1 + p_2 + \dots + p_k$ , i.e.*

## 7.89.1 Detailed Description

Interface to several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

## 7.89.2 Macro Definition Documentation

### 7.89.2.1 DEFAULT\_CTOR

```
#define DEFAULT_CTOR(  
    classname ) classname::classname() { setflag(status_flags::evaluated | status_←  
flags::expanded); }
```

### 7.89.2.2 DEFAULT\_COMPARE

```
#define DEFAULT_COMPARE(  
    classname )
```

**Value:**

```
int classname::compare_same_type(const basic & other) const \  
{ \  
    /* by default, the objects are always identical */ \  
    return 0; \  
}
```

### 7.89.2.3 DEFAULT\_PRINT

```
#define DEFAULT_PRINT(  
    classname,  
    text )
```

**Value:**

```
void classname::do_print(const print_context & c, unsigned level) const \  
{ \  
    c.s << text; \  
}
```

## 7.89.2.4 DEFAULT\_PRINT\_LATEX

```
#define DEFAULT_PRINT_LATEX(
    classname,
    text,
    latex )
```

## Value:

```
DEFAULT_PRINT(classname, text) \
void classname::do_print_latex(const print_latex & c, unsigned level) const \
{ \
    c.s << latex; \
}
```

## 7.90 utils\_multi\_iterator.h File Reference

Utilities for summing over multiple indices.

```
#include <cstddef>
#include <vector>
#include <ostream>
#include <iterator>
```

## Classes

- class `GiNaC::has_distance< T >`  
*SFINAE test for distance.*
- class `GiNaC::basic_multi_iterator< T >`  
*basic\_multi\_iterator is a base class.*
- class `GiNaC::multi_iterator_ordered< T >`  
*The class multi\_iterator\_ordered defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that*

$$B \leq i_j < N$$
*and*

$$i_j < i_{j+1}.$$
*It is assumed that  $k > 0$  and  $N - B \geq k$ .*
- class `GiNaC::multi_iterator_ordered_eq< T >`  
*The class multi\_iterator\_ordered\_eq defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that*

$$B \leq i_j < N$$
*and*

$$i_j \leq i_{j+1}.$$
*It is assumed that  $k > 0$ .*
- class `GiNaC::multi_iterator_ordered_eq_indv< T >`  
*The class multi\_iterator\_ordered\_eq\_indv defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that*

$$B \leq i_j < N_j$$
*and*

$$i_j \leq i_{j+1}.$$
*.*

- class [GiNaC::multi\\_iterator\\_counter< T >](#)

The class [multi\\_iterator\\_counter](#) defines a [multi\\_iterator](#)  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N$$

- class [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#)

The class [multi\\_iterator\\_counter\\_indv](#) defines a [multi\\_iterator](#)  $(i_1, i_2, \dots, i_k)$ , such that

$$B \leq i_j < N_j$$

- class [GiNaC::multi\\_iterator\\_permutation< T >](#)

The class [multi\\_iterator\\_permutation](#) defines a [multi\\_iterator](#)  $(i_1, i_2, \dots, i_k)$ , for which

$$B \leq i_j < N$$

and

$$i_i \neq i_j$$

In particular, if  $N - B = k$ , [multi\\_iterator\\_permutation](#) loops over all permutations of  $k$  elements.

- class [GiNaC::multi\\_iterator\\_shuffle< T >](#)

The class [multi\\_iterator\\_shuffle](#) defines a [multi\\_iterator](#), which runs over all shuffles of  $a$  and  $b$ .

- class [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#)

The class [multi\\_iterator\\_shuffle\\_prime](#) defines a [multi\\_iterator](#), which runs over all shuffles of  $a$  and  $b$ , excluding the first one  $(a,b)$ .

## Namespaces

- [GiNaC](#)

## Functions

- `template<typename T >`  
`std::enable_if< has_distance< T >::value, typename std::iterator_traits< T >::difference_type >::type`  
[GiNaC::format\\_index\\_value](#) (const T &a, const T &b)  
*For printing a multi-index: If the templates are used, where  $T$  is an iterator, printing the address where the iterator points to is not meaningful.*
- `template<typename T >`  
`std::enable_if< !has_distance< T >::value, T >::type` [GiNaC::format\\_index\\_value](#) (const T &a, const T &b)  
*For all other cases we simply print the value.*
- `template<class T >`  
`std::ostream & GiNaC::operator<<` (std::ostream &os, const basic\_multi\_iterator< T > &v)  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<<` (std::ostream &os, const multi\_iterator\_ordered< T > &v)  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<<` (std::ostream &os, const multi\_iterator\_ordered\_eq< T > &v)  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<<` (std::ostream &os, const multi\_iterator\_ordered\_eq\_indv< T > &v)  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<<` (std::ostream &os, const multi\_iterator\_counter< T > &v)  
*Output operator.*

- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_counter_indv< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_permutation< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_shuffle< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_shuffle_prime< T > &v)`  
*Output operator.*

### 7.90.1 Detailed Description

Utilities for summing over multiple indices.

## 7.91 version.h File Reference

[GiNaC](#) library version information.

### Namespaces

- [GiNaC](#)

### Macros

- `#define GINACLIB_MAJOR_VERSION 1`
- `#define GINACLIB_MINOR_VERSION 8`
- `#define GINACLIB_MICRO_VERSION 0`
- `#define GINAC_LT_CURRENT 11`
- `#define GINAC_LT_REVISION 0`
- `#define GINAC_LT_AGE 0`
- `#define GINACLIB_ARCHIVE_VERSION 3`
- `#define GINACLIB_ARCHIVE_AGE 3`
- `#define GINACLIB_STR_HELPER(x) #x`
- `#define GINACLIB_STR(x) GINACLIB_STR_HELPER(x)`
- `#define GINACLIB_VERSION`

### 7.91.1 Detailed Description

[GiNaC](#) library version information.

### 7.91.2 Macro Definition Documentation

#### 7.91.2.1 GINACLIB\_MAJOR\_VERSION

```
#define GINACLIB_MAJOR_VERSION 1
```

#### 7.91.2.2 GINACLIB\_MINOR\_VERSION

```
#define GINACLIB_MINOR_VERSION 8
```

#### 7.91.2.3 GINACLIB\_MICRO\_VERSION

```
#define GINACLIB_MICRO_VERSION 0
```

#### 7.91.2.4 GINAC\_LT\_CURRENT

```
#define GINAC_LT_CURRENT 11
```

#### 7.91.2.5 GINAC\_LT\_REVISION

```
#define GINAC_LT_REVISION 0
```

#### 7.91.2.6 GINAC\_LT\_AGE

```
#define GINAC_LT_AGE 0
```

#### 7.91.2.7 GINACLIB\_ARCHIVE\_VERSION

```
#define GINACLIB_ARCHIVE_VERSION 3
```

Referenced by `GiNaC::operator<<()`, and `GiNaC::operator>>()`.

#### 7.91.2.8 GINACLIB\_ARCHIVE\_AGE

```
#define GINACLIB_ARCHIVE_AGE 3
```

Referenced by `GiNaC::operator>>()`.

#### 7.91.2.9 GINACLIB\_STR\_HELPER

```
#define GINACLIB_STR_HELPER(  
    x ) #x
```

#### 7.91.2.10 GINACLIB\_STR

```
#define GINACLIB_STR(  
    x ) GINACLIB_STR_HELPER(x)
```

#### 7.91.2.11 GINACLIB\_VERSION

```
#define GINACLIB_VERSION
```

**Value:**

```
GINACLIB_STR(GINACLIB_MAJOR_VERSION) "." \\  
GINACLIB_STR(GINACLIB_MINOR_VERSION) "." \\  
GINACLIB_STR(GINACLIB_MICRO_VERSION)
```

## 7.92 wildcard.cpp File Reference

Implementation of [GiNaC](#)'s wildcard objects.

```
#include "wildcard.h"  
#include "archive.h"  
#include "utils.h"  
#include "hash_seed.h"  
#include <iostream>
```

### Namespaces

- [GiNaC](#)



## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (wildcard, basic, print\_func< print\_context >(&wildcard::do\_print). print\_func< print\_tree >(&wildcard::do\_print\_tree). print\_func< print\_python\_repr >(&wildcard::do\_print\_python\_repr)) wildcard
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (wildcard)
- bool [GiNaC::haswild](#) (const ex &x)

*Check whether x has a wildcard anywhere as a subexpression.*

### 7.92.1 Detailed Description

Implementation of [GiNaC](#)'s wildcard objects.

## 7.93 wildcard.h File Reference

Interface to [GiNaC](#)'s wildcard objects.

```
#include "ex.h"
#include "archive.h"
```

## Classes

- class [GiNaC::wildcard](#)  
*This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).*

## Namespaces

- [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (wildcard)
- ex [GiNaC::wild](#) (unsigned label=0)  
*Create a wildcard object with the specified label.*
- bool [GiNaC::haswild](#) (const ex &x)  
*Check whether x has a wildcard anywhere as a subexpression.*

### 7.93.1 Detailed Description

Interface to [GiNaC](#)'s wildcard objects.



# Index

[\\_ex0](#)  
    [GiNaC, 299](#)

[\\_ex1](#)  
    [GiNaC, 301](#)

[\\_ex10](#)  
    [GiNaC, 304](#)

[\\_ex11](#)  
    [GiNaC, 305](#)

[\\_ex12](#)  
    [GiNaC, 305](#)

[\\_ex120](#)  
    [GiNaC, 308](#)

[\\_ex15](#)  
    [GiNaC, 305](#)

[\\_ex18](#)  
    [GiNaC, 306](#)

[\\_ex1\\_2](#)  
    [GiNaC, 300](#)

[\\_ex1\\_3](#)  
    [GiNaC, 300](#)

[\\_ex1\\_4](#)  
    [GiNaC, 300](#)

[\\_ex2](#)  
    [GiNaC, 301](#)

[\\_ex20](#)  
    [GiNaC, 306](#)

[\\_ex24](#)  
    [GiNaC, 306](#)

[\\_ex25](#)  
    [GiNaC, 307](#)

[\\_ex3](#)  
    [GiNaC, 302](#)

[\\_ex30](#)  
    [GiNaC, 307](#)

[\\_ex4](#)  
    [GiNaC, 302](#)

[\\_ex48](#)  
    [GiNaC, 307](#)

[\\_ex5](#)  
    [GiNaC, 303](#)

[\\_ex6](#)  
    [GiNaC, 303](#)

[\\_ex60](#)  
    [GiNaC, 308](#)

[\\_ex7](#)  
    [GiNaC, 303](#)

[\\_ex8](#)  
    [GiNaC, 304](#)

[\\_ex9](#)  
    [GiNaC, 304](#)

[\\_ex\\_1](#)  
    [GiNaC, 298](#)

[\\_ex\\_10](#)  
    [GiNaC, 295](#)

[\\_ex\\_11](#)  
    [GiNaC, 294](#)

[\\_ex\\_12](#)  
    [GiNaC, 294](#)

[\\_ex\\_120](#)  
    [GiNaC, 291](#)

[\\_ex\\_15](#)  
    [GiNaC, 294](#)

[\\_ex\\_18](#)  
    [GiNaC, 293](#)

[\\_ex\\_1\\_2](#)  
    [GiNaC, 298](#)

[\\_ex\\_1\\_3](#)  
    [GiNaC, 298](#)

[\\_ex\\_1\\_4](#)  
    [GiNaC, 299](#)

[\\_ex\\_2](#)  
    [GiNaC, 297](#)

[\\_ex\\_20](#)  
    [GiNaC, 293](#)

[\\_ex\\_24](#)  
    [GiNaC, 293](#)

[\\_ex\\_25](#)  
    [GiNaC, 292](#)

[\\_ex\\_3](#)  
    [GiNaC, 297](#)

[\\_ex\\_30](#)  
    [GiNaC, 292](#)

[\\_ex\\_4](#)  
    [GiNaC, 297](#)

[\\_ex\\_48](#)  
    [GiNaC, 292](#)

[\\_ex\\_5](#)  
    [GiNaC, 296](#)

[\\_ex\\_6](#)  
    [GiNaC, 296](#)

[\\_ex\\_60](#)  
    [GiNaC, 291](#)

[\\_ex\\_7](#)  
    [GiNaC, 296](#)

[\\_ex\\_8](#)  
    [GiNaC, 295](#)

[\\_ex\\_9](#)  
    [GiNaC, 295](#)

- `_iter_rep`
  - `GiNaC::internal::_iter_rep`, 311
- `_num0_bp`
  - `GiNaC`, 289
- `_num0_p`
  - `GiNaC`, 299
- `_num10_p`
  - `GiNaC`, 304
- `_num11_p`
  - `GiNaC`, 304
- `_num120_p`
  - `GiNaC`, 308
- `_num12_p`
  - `GiNaC`, 305
- `_num15_p`
  - `GiNaC`, 305
- `_num18_p`
  - `GiNaC`, 305
- `_num1_2_p`
  - `GiNaC`, 300
- `_num1_3_p`
  - `GiNaC`, 300
- `_num1_4_p`
  - `GiNaC`, 299
- `_num1_p`
  - `GiNaC`, 300
- `_num20_p`
  - `GiNaC`, 306
- `_num24_p`
  - `GiNaC`, 306
- `_num25_p`
  - `GiNaC`, 306
- `_num2_p`
  - `GiNaC`, 301
- `_num30_p`
  - `GiNaC`, 307
- `_num3_p`
  - `GiNaC`, 302
- `_num48_p`
  - `GiNaC`, 307
- `_num4_p`
  - `GiNaC`, 302
- `_num5_p`
  - `GiNaC`, 302
- `_num60_p`
  - `GiNaC`, 307
- `_num6_p`
  - `GiNaC`, 303
- `_num7_p`
  - `GiNaC`, 303
- `_num8_p`
  - `GiNaC`, 303
- `_num9_p`
  - `GiNaC`, 304
- `_num_10_p`
  - `GiNaC`, 294
- `_num_11_p`
  - `GiNaC`, 294
- `_num_120_p`
  - `GiNaC`, 291
- `_num_12_p`
  - `GiNaC`, 294
- `_num_15_p`
  - `GiNaC`, 293
- `_num_18_p`
  - `GiNaC`, 293
- `_num_1_2_p`
  - `GiNaC`, 298
- `_num_1_3_p`
  - `GiNaC`, 298
- `_num_1_4_p`
  - `GiNaC`, 299
- `_num_1_p`
  - `GiNaC`, 297
- `_num_20_p`
  - `GiNaC`, 293
- `_num_24_p`
  - `GiNaC`, 292
- `_num_25_p`
  - `GiNaC`, 292
- `_num_2_p`
  - `GiNaC`, 297
- `_num_30_p`
  - `GiNaC`, 292
- `_num_3_p`
  - `GiNaC`, 297
- `_num_48_p`
  - `GiNaC`, 291
- `_num_4_p`
  - `GiNaC`, 296
- `_num_5_p`
  - `GiNaC`, 296
- `_num_60_p`
  - `GiNaC`, 291
- `_num_6_p`
  - `GiNaC`, 296
- `_num_7_p`
  - `GiNaC`, 295
- `_num_8_p`
  - `GiNaC`, 295
- `_num_9_p`
  - `GiNaC`, 295
- `_numeric_digits`
  - `GiNaC::_numeric_digits`, 313
- `~archive`
  - `GiNaC::archive`, 329
- `~basic`
  - `GiNaC::basic`, 351
- `~basic_multi_iterator`
  - `GiNaC::basic_multi_iterator`, 378
- `~compare_all_equal`
  - `GiNaC::compare_all_equal`, 402
- `~compare_bitwise`
  - `GiNaC::compare_bitwise`, 403
- `~compare_std_less`
  - `GiNaC::compare_std_less`, 404

- ~container\_storage
  - GiNaC::container\_storage, 442
- ~coolmulti
  - GiNaC::composition\_generator::coolmulti, 444
- ~element
  - GiNaC::composition\_generator::coolmulti::element, 471
- ~function\_options
  - GiNaC::function\_options, 568
- ~library\_init
  - GiNaC::library\_init, 669
- ~map\_function
  - GiNaC::map\_function, 674
- ~print\_context
  - GiNaC::print\_context, 823
- ~print\_functor\_impl
  - GiNaC::print\_functor\_impl, 834
- ~ptr
  - GiNaC::ptr, 865
- ~unarchive\_table\_t
  - GiNaC::unarchive\_table\_t, 976
- ~visitor
  - GiNaC::visitor, 986
- a
  - GiNaC::Eisenstein\_kernel, 470
  - GiNaC::archive\_node, 344
  - GiNaC::integral, 648
- abs
  - GiNaC, 240
- abs\_conjugate
  - GiNaC, 143
- abs\_eval
  - GiNaC, 142
- abs\_evalf
  - GiNaC, 142
- abs\_expand
  - GiNaC, 143
- abs\_expl\_derivative
  - GiNaC, 143
- abs\_imag\_part
  - GiNaC, 144
- abs\_info
  - GiNaC, 144
- abs\_power
  - GiNaC, 144
- abs\_print\_csrc\_float
  - GiNaC, 143
- abs\_print\_latex
  - GiNaC, 143
- abs\_real\_part
  - GiNaC, 144
- accept
  - GiNaC::basic, 359
  - GiNaC::ex, 495
- access\_counter
  - GiNaC::remember\_table\_entry, 891
- acos
  - GiNaC, 231
- acos\_conjugate
  - GiNaC, 184
- acos\_deriv
  - GiNaC, 184
- acos\_eval
  - GiNaC, 184
- acos\_evalf
  - GiNaC, 184
- acosh
  - GiNaC, 234
- acosh\_conjugate
  - GiNaC, 193
- acosh\_deriv
  - GiNaC, 193
- acosh\_eval
  - GiNaC, 192
- acosh\_evalf
  - GiNaC, 192
- adaptivesimpson
  - GiNaC, 195
- add
  - GiNaC::add, 317, 318
  - GiNaC::matrix, 683
  - GiNaC::mul, 720
  - GiNaC::numeric, 770
  - GiNaC::scalar\_products, 897
  - GiNaC::symmetry, 956
- add.cpp, 993
- add.h, 994
- add\_bool
  - GiNaC::archive\_node, 339
- add\_callback
  - GiNaC::\_numeric\_digits, 314
- add\_child
  - GiNaC::class\_info::tree\_node, 975
- add\_dyn
  - GiNaC::numeric, 772
- add\_entry
  - GiNaC::remember\_table, 887
  - GiNaC::remember\_table\_list, 892
- add\_ex
  - GiNaC::archive\_node, 339
- add\_indexed
  - GiNaC::basic, 364
  - GiNaC::matrix, 680
  - GiNaC::structure, 924
- add\_node
  - GiNaC::archive, 332
- add\_reference
  - GiNaC::refcounted, 872
- add\_series
  - GiNaC::pseries, 857
- add\_string
  - GiNaC::archive\_node, 339
- add\_symbol
  - GiNaC, 211
- add\_unsigned
  - GiNaC::archive\_node, 339

- add\_vectors
  - GiNaC::scalar\_products, 898
- after\_i
  - GiNaC::composition\_generator::coolmulti, 445
- algebraic\_match\_mul\_with\_mul
  - GiNaC, 209
- algebraic\_subs\_mul
  - GiNaC::mul, 718
- all\_index\_values\_are
  - GiNaC::indexed, 635
- antisymmetric2
  - GiNaC, 270
- antisymmetric3
  - GiNaC, 270
- antisymmetric4
  - GiNaC, 270
- antisymmetrize
  - GiNaC::ex, 508, 509
  - GiNaC, 119, 271, 275
- append
  - GiNaC::container, 436
- append\_factors
  - GiNaC::ncmul, 754
- archive
  - GiNaC::archive, 329, 330
  - GiNaC::basic, 368
  - GiNaC::clifford, 389
  - GiNaC::color, 399
  - GiNaC::constant, 422
  - GiNaC::container, 433
  - GiNaC::expairseq, 527
  - GiNaC::fderivative, 543
  - GiNaC::function, 557
  - GiNaC::idx, 618
  - GiNaC::indexed, 633
  - GiNaC::integral, 646
  - GiNaC::matrix, 681
  - GiNaC::minkmetric, 697
  - GiNaC::numeric, 769
  - GiNaC::power, 817
  - GiNaC::pseries, 854
  - GiNaC::relational, 880
  - GiNaC::spinidx, 903
  - GiNaC::symbol, 946
  - GiNaC::symmetry, 955
  - GiNaC::tensepsilon, 967
  - GiNaC::varidx, 983
  - GiNaC::wildcard, 987
- archive.cpp, 994
- archive.h, 995
  - GINAC\_BIND\_UNARCHIVER, 997
  - GINAC\_DECLARE\_UNARCHIVER, 996
- archive\_atom
  - GiNaC, 58
- archive\_ex
  - GiNaC::archive, 330
- archive\_node
  - GiNaC::archive\_node, 338
- GiNaC::ex, 513
- archive\_node\_cit
  - GiNaC::archive\_node, 337
- archive\_node\_id
  - GiNaC, 58
- archived\_ex
  - GiNaC::archive::archived\_ex, 346, 347
- are\_ex\_trivially\_equal
  - GiNaC::ex, 513
  - GiNaC, 111
- arg1
  - GiNaC::pointer\_to\_map\_function\_1arg, 793
  - GiNaC::pointer\_to\_map\_function\_2args, 794
  - GiNaC::pointer\_to\_map\_function\_3args, 796
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
1arg, 800
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
2args, 802
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
3args, 804
- arg2
  - GiNaC::pointer\_to\_map\_function\_2args, 794
  - GiNaC::pointer\_to\_map\_function\_3args, 796
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
2args, 802
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
3args, 804
- arg3
  - GiNaC::pointer\_to\_map\_function\_3args, 796
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
3args, 804
- argument\_type
  - GiNaC::map\_function, 673
- asin
  - GiNaC, 231
- asin\_conjugate
  - GiNaC, 183
- asin\_deriv
  - GiNaC, 183
- asin\_eval
  - GiNaC, 183
- asin\_evalf
  - GiNaC, 183
- asinh
  - GiNaC, 234
- asinh\_conjugate
  - GiNaC, 192
- asinh\_deriv
  - GiNaC, 192
- asinh\_eval
  - GiNaC, 191
- asinh\_evalf
  - GiNaC, 191
- assertion.h, 998
  - GINAC\_ASSERT, 998
- atan
  - GiNaC, 232
- atan2\_deriv

- GiNaC, [186](#)
- atan2\_eval
  - GiNaC, [186](#)
- atan2\_evalf
  - GiNaC, [186](#)
- atan\_conjugate
  - GiNaC, [186](#)
- atan\_deriv
  - GiNaC, [185](#)
- atan\_eval
  - GiNaC, [185](#)
- atan\_evalf
  - GiNaC, [185](#)
- atan\_series
  - GiNaC, [185](#)
- atanh
  - GiNaC, [234](#)
- atanh\_conjugate
  - GiNaC, [194](#)
- atanh\_deriv
  - GiNaC, [194](#)
- atanh\_eval
  - GiNaC, [193](#)
- atanh\_evalf
  - GiNaC, [193](#)
- atanh\_series
  - GiNaC, [194](#)
- atend
  - GiNaC::composition\_generator, [407](#)
- atomize
  - GiNaC::archive, [333](#)
- atoms
  - GiNaC::archive, [335](#)
- attribute\_deprecated
  - compiler.h, [1010](#)
- B
  - GiNaC::basic\_multi\_iterator, [380](#)
- b
  - GiNaC::Eisenstein\_kernel, [470](#)
  - GiNaC::integral, [649](#)
- base\_and\_index
  - GiNaC, [94](#)
- basic
  - GiNaC::basic, [351](#), [352](#)
- basic.cpp, [999](#)
- basic.h, [1000](#)
- basic\_multi\_iterator
  - GiNaC::basic\_multi\_iterator, [377](#)
- basic\_partition\_generator
  - GiNaC::basic\_partition\_generator, [382](#)
- basis
  - GiNaC::power, [822](#)
- begin
  - GiNaC::archive\_node::archive\_node\_cit\_range, [345](#)
  - GiNaC::container, [437](#)
  - GiNaC::ex, [486](#)
- bernoulli
  - GiNaC, [239](#)
- Bernoulli\_polynomial
  - GiNaC, [197](#)
- beta\_deriv
  - GiNaC, [166](#)
- beta\_eval
  - GiNaC, [166](#)
- beta\_evalf
  - GiNaC, [165](#)
- beta\_series
  - GiNaC, [166](#)
- binomial
  - GiNaC, [239](#)
- binomial\_conjugate
  - GiNaC, [153](#)
- binomial\_eval
  - GiNaC, [153](#)
- binomial\_evalf
  - GiNaC, [152](#)
- binomial\_imag\_part
  - GiNaC, [153](#)
- binomial\_real\_part
  - GiNaC, [153](#)
- binomial\_sym
  - GiNaC, [153](#)
- bp
  - GiNaC::ex, [514](#)
- c
  - factor.cpp, [1023](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function, [798](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↵
    - 1arg, [800](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↵
    - 2args, [802](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↵
    - 3args, [804](#)
- C\_norm
  - GiNaC::Eisenstein\_h\_kernel, [465](#)
  - GiNaC::Eisenstein\_kernel, [470](#)
  - GiNaC::Kronecker\_dtau\_kernel, [661](#)
  - GiNaC::Kronecker\_dz\_kernel, [665](#)
  - GiNaC::modular\_form\_kernel, [702](#)
- CMPINDICES
  - color.cpp, [1008](#)
- cache
  - factor.cpp, [1025](#)
- cache\_step\_size
  - GiNaC::integration\_kernel, [654](#)
- cache\_vec
  - integration\_kernel.cpp, [1067](#)
- calc\_lanczos\_A
  - GiNaC::lanczos\_coeffs, [666](#)
- calchash
  - GiNaC::basic, [367](#)
  - GiNaC::constant, [423](#)
  - GiNaC::expairseq, [529](#)
  - GiNaC::function, [555](#)
  - GiNaC::idx, [619](#)

- GiNaC::numeric, 770
  - GiNaC::relational, 882
  - GiNaC::structure, 926
  - GiNaC::symbol, 947
  - GiNaC::symmetry, 955
  - GiNaC::wildcard, 988
- callbacklist
  - GiNaC::\_numeric\_digits, 315
- can\_be\_further\_expanded
  - GiNaC::mul, 719
- can\_make\_flat
  - GiNaC::expairseq, 532
  - GiNaC::mul, 717
- canonicalize
  - GiNaC::expairseq, 533
  - GiNaC::symmetry, 958
  - GiNaC, 270
- canonicalize\_clifford
  - GiNaC, 99
- Catalan
  - GiNaC, 288
- CatalanEvalf
  - GiNaC, 245
- charpoly
  - GiNaC::matrix, 687
  - GiNaC, 207
- children
  - GiNaC::class\_info::tree\_node, 975
  - GiNaC::symmetry, 959
- class\_info
  - GiNaC::class\_info, 383
- class\_info.h, 1001
- clear
  - GiNaC::archive, 332
  - GiNaC::scalar\_products, 898
- clear\_all\_entries
  - GiNaC::remember\_table, 887
- clearflag
  - GiNaC::basic, 372
- clifford
  - GiNaC::clifford, 387, 388
- clifford.cpp, 1002
- clifford.h, 1004
- clifford\_bar
  - GiNaC, 104
- clifford\_inverse
  - GiNaC, 100
- clifford\_max\_label
  - GiNaC, 100
- clifford\_moebius\_map
  - GiNaC, 102
- clifford\_norm
  - GiNaC, 100
- clifford\_prime
  - GiNaC, 99
- clifford\_star
  - GiNaC, 104
- clifford\_star\_bar
  - GiNaC, 99
- clifford\_to\_lst
  - GiNaC, 101
- clifford\_unit
  - GiNaC, 95
- cmgen
  - GiNaC::composition\_generator, 406
- coeff
  - GiNaC::add, 320
  - GiNaC::basic, 360
  - GiNaC::ex, 497
  - GiNaC::expair, 519
  - GiNaC::mul, 710
  - GiNaC::ncmul, 751
  - GiNaC::numeric, 764
  - GiNaC::power, 812
  - GiNaC::pseries, 851
  - GiNaC::structure, 920
  - GiNaC::symminfo, 960
  - GiNaC, 115
- coefficient\_a0
  - GiNaC::Eisenstein\_h\_kernel, 463
- coefficient\_an
  - GiNaC::Eisenstein\_h\_kernel, 463
- coeffop
  - GiNaC::pseries, 857
- coeffs
  - GiNaC::lanczos\_coeffs, 667
- coerce
  - GiNaC, 228
- coerce< int, cln::cl\_I >
  - GiNaC, 228
- coerce< unsigned int, cln::cl\_I >
  - GiNaC, 229
- col
  - GiNaC::matrix, 694
- collect
  - GiNaC::basic, 360
  - GiNaC::ex, 498
  - GiNaC::pseries, 851
  - GiNaC::structure, 920
  - GiNaC, 117
- collect\_common\_factors
  - GiNaC, 225
- collect\_symbols
  - GiNaC, 211
- color
  - GiNaC::color, 397, 398
- color.cpp, 1006
  - CMPINDICES, 1008
  - TEST\_PERMUTATION, 1007
- color.h, 1008
- color\_ONE
  - GiNaC, 106
- color\_d
  - GiNaC, 107
- color\_f
  - GiNaC, 107



- color\_h
  - GiNaC, [108](#)
- color\_T
  - GiNaC, [107](#)
- color\_trace
  - GiNaC, [109](#)
- cols
  - GiNaC::matrix, [683](#)
  - GiNaC, [206](#)
- combine\_ex\_with\_coeff\_to\_pair
  - GiNaC::add, [325](#)
  - GiNaC::expairseq, [530](#)
  - GiNaC::mul, [715](#)
- combine\_indices
  - GiNaC::make\_flat\_inserter, [671](#)
- combine\_overall\_coeff
  - GiNaC::expairseq, [531](#)
  - GiNaC::mul, [717](#)
- combine\_pair\_with\_coeff\_to\_pair
  - GiNaC::add, [325](#)
  - GiNaC::expairseq, [530](#)
  - GiNaC::mul, [716](#)
- combine\_same\_terms\_sorted\_seq
  - GiNaC::expairseq, [534](#)
- commutator\_sign
  - GiNaC::clifford, [394](#)
- compare
  - GiNaC::basic, [370](#)
  - GiNaC::ex, [507](#)
  - GiNaC::expair, [518](#)
  - GiNaC::numeric, [775](#)
- compare\_pointers
  - GiNaC, [282](#)
- compare\_same\_type
  - GiNaC::basic, [366](#)
- compile\_ex
  - GiNaC, [123](#), [124](#)
- compiler.h, [1009](#)
  - attribute\_deprecated, [1010](#)
  - likely, [1010](#)
  - unlikely, [1010](#)
- composition
  - GiNaC::composition\_generator, [407](#)
- composition\_generator
  - GiNaC::composition\_generator, [406](#)
- conjugate
  - GiNaC::add, [322](#)
  - GiNaC::basic, [366](#)
  - GiNaC::constant, [422](#)
  - GiNaC::container, [433](#)
  - GiNaC::diracgamma5, [450](#)
  - GiNaC::diracgammaL, [451](#)
  - GiNaC::diracgammaR, [453](#)
  - GiNaC::ex, [492](#)
  - GiNaC::expair, [519](#)
  - GiNaC::expairseq, [527](#)
  - GiNaC::function, [556](#)
  - GiNaC::integral, [646](#)
  - GiNaC::matrix, [680](#)
  - GiNaC::mul, [714](#)
  - GiNaC::ncmul, [752](#)
  - GiNaC::numeric, [768](#)
  - GiNaC::power, [816](#)
  - GiNaC::pseries, [853](#)
  - GiNaC::realsymbol, [869](#)
  - GiNaC::spinidx, [903](#)
  - GiNaC::symbol, [945](#)
  - GiNaC, [113](#)
- conjugate\_conjugate
  - GiNaC, [138](#)
- conjugate\_eval
  - GiNaC, [138](#)
- conjugate\_evalf
  - GiNaC, [138](#)
- conjugate\_expl\_derivative
  - GiNaC, [138](#)
- conjugate\_f
  - GiNaC::function\_options, [603](#)
- conjugate\_func
  - GiNaC::function\_options, [574–576](#), [595](#)
- conjugate\_funcp
  - GiNaC, [61](#)
- conjugate\_funcp\_1
  - GiNaC, [62](#)
- conjugate\_funcp\_10
  - GiNaC, [77](#)
- conjugate\_funcp\_11
  - GiNaC, [79](#)
- conjugate\_funcp\_12
  - GiNaC, [80](#)
- conjugate\_funcp\_13
  - GiNaC, [82](#)
- conjugate\_funcp\_14
  - GiNaC, [84](#)
- conjugate\_funcp\_2
  - GiNaC, [64](#)
- conjugate\_funcp\_3
  - GiNaC, [65](#)
- conjugate\_funcp\_4
  - GiNaC, [67](#)
- conjugate\_funcp\_5
  - GiNaC, [68](#)
- conjugate\_funcp\_6
  - GiNaC, [70](#)
- conjugate\_funcp\_7
  - GiNaC, [72](#)
- conjugate\_funcp\_8
  - GiNaC, [73](#)
- conjugate\_funcp\_9
  - GiNaC, [75](#)
- conjugate\_funcp\_exvector
  - GiNaC, [86](#)
- conjugate\_imag\_part
  - GiNaC, [139](#)
- conjugate\_info
  - GiNaC, [139](#)

- conjugate\_print\_latex
  - GiNaC, [138](#)
- conjugate\_real\_part
  - GiNaC, [138](#)
- conjugate\_use\_exvector\_args
  - GiNaC::function\_options, [606](#)
- conjugateepvector
  - GiNaC, [126](#)
- const\_iterator
  - GiNaC::const\_iterator, [409](#)
  - GiNaC::container, [429](#)
- const\_postorder\_iterator
  - GiNaC::const\_iterator, [412](#)
  - GiNaC::const\_postorder\_iterator, [414](#)
- const\_preorder\_iterator
  - GiNaC::const\_iterator, [412](#)
  - GiNaC::const\_preorder\_iterator, [417](#)
- const\_reverse\_iterator
  - GiNaC::container, [429](#)
- constant
  - GiNaC::constant, [420](#)
- constant.cpp, [1010](#)
- constant.h, [1011](#)
- construct\_from\_2\_ex
  - GiNaC::expairseq, [532](#)
- construct\_from\_2\_expairseq
  - GiNaC::expairseq, [532](#)
- construct\_from\_basic
  - GiNaC::ex, [510](#)
- construct\_from\_double
  - GiNaC::ex, [512](#)
- construct\_from\_epvector
  - GiNaC::expairseq, [533](#)
- construct\_from\_expairseq\_ex
  - GiNaC::expairseq, [532](#)
- construct\_from\_exvector
  - GiNaC::expairseq, [533](#)
- construct\_from\_int
  - GiNaC::ex, [510](#)
- construct\_from\_long
  - GiNaC::ex, [511](#)
- construct\_from\_longlong
  - GiNaC::ex, [511](#)
- construct\_from\_string\_and\_lst
  - GiNaC::ex, [512](#)
- construct\_from\_uint
  - GiNaC::ex, [511](#)
- construct\_from\_ulong
  - GiNaC::ex, [511](#)
- construct\_from\_ulonglong
  - GiNaC::ex, [512](#)
- container
  - GiNaC::container, [429](#), [430](#)
- container.h, [1012](#)
- container\_storage
  - GiNaC::container\_storage, [441](#)
- content
  - GiNaC::ex, [502](#)
- contract\_with
  - GiNaC::basic, [365](#)
  - GiNaC::cliffordunit, [396](#)
  - GiNaC::diracgamma, [448](#)
  - GiNaC::matrix, [680](#)
  - GiNaC::spinmetric, [908](#)
  - GiNaC::structure, [925](#)
  - GiNaC::su3d, [929](#)
  - GiNaC::su3f, [931](#)
  - GiNaC::su3t, [934](#)
  - GiNaC::tensdelta, [963](#)
  - GiNaC::tensepsilon, [966](#)
  - GiNaC::tensmetric, [970](#)
- convert\_H\_to\_Li
  - GiNaC, [159](#)
- convert\_to\_poly
  - GiNaC::pseries, [856](#)
- coolmulti
  - GiNaC::composition\_generator::coolmulti, [444](#)
- cos
  - GiNaC, [230](#)
- cos\_conjugate
  - GiNaC, [181](#)
- cos\_deriv
  - GiNaC, [180](#)
- cos\_eval
  - GiNaC, [180](#)
- cos\_evalf
  - GiNaC, [180](#)
- cos\_imag\_part
  - GiNaC, [181](#)
- cos\_real\_part
  - GiNaC, [180](#)
- cosh
  - GiNaC, [233](#)
- cosh\_conjugate
  - GiNaC, [189](#)
- cosh\_deriv
  - GiNaC, [189](#)
- cosh\_eval
  - GiNaC, [188](#)
- cosh\_evalf
  - GiNaC, [188](#)
- cosh\_imag\_part
  - GiNaC, [189](#)
- cosh\_real\_part
  - GiNaC, [189](#)
- count
  - GiNaC::archive\_node::property\_info, [846](#)
  - GiNaC::library\_init, [670](#)
- count\_dummy\_indices
  - GiNaC, [132](#)
- count\_factors
  - GiNaC::ncmul, [754](#)
- count\_free\_indices
  - GiNaC, [132](#)
- covariant
  - GiNaC::varidx, [985](#)

- crc32
  - GiNaC, [111](#)
- crc32.h, [1012](#)
- crctab
  - GiNaC, [289](#)
- csgn
  - GiNaC::numeric, [774](#)
  - GiNaC, [246](#)
- csgn\_conjugate
  - GiNaC, [147](#)
- csgn\_eval
  - GiNaC, [146](#)
- csgn\_evalf
  - GiNaC, [146](#)
- csgn\_imag\_part
  - GiNaC, [147](#)
- csgn\_power
  - GiNaC, [147](#)
- csgn\_real\_part
  - GiNaC, [147](#)
- csgn\_series
  - GiNaC, [146](#)
- csrc
  - GiNaC, [261](#)
- csrc\_cl\_N
  - GiNaC, [262](#)
- csrc\_double
  - GiNaC, [261](#)
- csrc\_float
  - GiNaC, [261](#)
- current\_serial
  - GiNaC::function, [562](#)
- current\_updated
  - GiNaC::composition\_generator, [407](#)
  - GiNaC::partition\_generator, [788](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [790](#)
- current\_vector
  - GiNaC::lanczos\_coeffs, [667](#)
- cyclic\_permutation
  - GiNaC, [283](#)
- DCOUT2
  - factor.cpp, [1022](#)
- DCOUTVAR
  - factor.cpp, [1022](#)
- DCOUT
  - factor.cpp, [1022](#)
- DECLARE\_FUNCTION\_10P
  - function.h, [1043](#)
- DECLARE\_FUNCTION\_11P
  - function.h, [1043](#)
- DECLARE\_FUNCTION\_12P
  - function.h, [1043](#)
- DECLARE\_FUNCTION\_13P
  - function.h, [1044](#)
- DECLARE\_FUNCTION\_14P
  - function.h, [1044](#)
- DECLARE\_FUNCTION\_1P
  - function.h, [1040](#)
- DECLARE\_FUNCTION\_2P
  - function.h, [1040](#)
- DECLARE\_FUNCTION\_3P
  - function.h, [1040](#)
- DECLARE\_FUNCTION\_4P
  - function.h, [1041](#)
- DECLARE\_FUNCTION\_5P
  - function.h, [1041](#)
- DECLARE\_FUNCTION\_6P
  - function.h, [1041](#)
- DECLARE\_FUNCTION\_7P
  - function.h, [1042](#)
- DECLARE\_FUNCTION\_8P
  - function.h, [1042](#)
- DECLARE\_FUNCTION\_9P
  - function.h, [1042](#)
- DEFAULT\_COMPARE
  - utils.h, [1113](#)
- DEFAULT\_CTOR
  - utils.h, [1113](#)
- DEFAULT\_PRINT\_LATEX
  - utils.h, [1113](#)
- DEFAULT\_PRINT
  - utils.h, [1113](#)
- dbgprint
  - GiNaC::basic, [354](#)
  - GiNaC::ex, [489](#)
- dbgprinttree
  - GiNaC::basic, [355](#)
  - GiNaC::ex, [489](#)
- debugprint
  - GiNaC::scalar\_products, [899](#)
  - GiNaC::spmapkey, [910](#)
- decomp\_rational
  - GiNaC, [214](#)
- default\_overall\_coeff
  - GiNaC::expairseq, [531](#)
  - GiNaC::mul, [716](#)
- deg
  - GiNaC::pole\_error, [806](#)
- deg\_a
  - GiNaC::sym\_desc, [939](#)
- deg\_b
  - GiNaC::sym\_desc, [939](#)
- degree
  - GiNaC::add, [319](#)
  - GiNaC::basic, [359](#)
  - GiNaC::ex, [497](#)
  - GiNaC::integral, [643](#)
  - GiNaC::mul, [709](#)
  - GiNaC::ncmul, [750](#)
  - GiNaC::numeric, [763](#)
  - GiNaC::pole\_error, [805](#)
  - GiNaC::power, [812](#)
  - GiNaC::pseries, [850](#)
  - GiNaC::structure, [919](#)
  - GiNaC, [114](#)
- delta\_indent

- GiNaC::print\_tree, 843
- delta\_tensor
  - GiNaC, 277
- denom
  - GiNaC::ex, 501
  - GiNaC::numeric, 783
  - GiNaC, 115, 250
- derivative
  - GiNaC::add, 323
  - GiNaC::basic, 361
  - GiNaC::constant, 423
  - GiNaC::fderivative, 543
  - GiNaC::function, 558
  - GiNaC::idx, 619
  - GiNaC::indexed, 634
  - GiNaC::integral, 647
  - GiNaC::mul, 714
  - GiNaC::ncmul, 753
  - GiNaC::numeric, 769
  - GiNaC::power, 817
  - GiNaC::pseries, 855
  - GiNaC::structure, 921
  - GiNaC::symbol, 947
- derivative\_f
  - GiNaC::function\_options, 603
- derivative\_func
  - GiNaC::function\_options, 583–585, 596
- derivative\_funcp
  - GiNaC, 61
- derivative\_funcp\_1
  - GiNaC, 63
- derivative\_funcp\_10
  - GiNaC, 77
- derivative\_funcp\_11
  - GiNaC, 79
- derivative\_funcp\_12
  - GiNaC, 81
- derivative\_funcp\_13
  - GiNaC, 83
- derivative\_funcp\_14
  - GiNaC, 85
- derivative\_funcp\_2
  - GiNaC, 64
- derivative\_funcp\_3
  - GiNaC, 66
- derivative\_funcp\_4
  - GiNaC, 67
- derivative\_funcp\_5
  - GiNaC, 69
- derivative\_funcp\_6
  - GiNaC, 71
- derivative\_funcp\_7
  - GiNaC, 72
- derivative\_funcp\_8
  - GiNaC, 74
- derivative\_funcp\_9
  - GiNaC, 76
- derivative\_funcp\_exvector
  - GiNaC, 87
- derivative\_map\_function
  - GiNaC::derivative\_map\_function, 446
- derivative\_use\_exvector\_args
  - GiNaC::function\_options, 607
- derivatives
  - GiNaC::fderivative, 544
- descend
  - GiNaC::const\_postorder\_iterator, 415
- determinant
  - GiNaC::matrix, 686
  - GiNaC, 207
- determinant\_minor
  - GiNaC::matrix, 690
- dflt
  - GiNaC, 260
- diag\_matrix
  - GiNaC, 204
- diff
  - GiNaC::basic, 369
  - GiNaC::ex, 498
  - GiNaC, 118
- Digits
  - GiNaC, 290
- digits
  - GiNaC::\_numeric\_digits, 315
- digits\_changed\_callback
  - GiNaC, 89
- dim
  - GiNaC::idx, 623
  - GiNaC::spmapkey, 910
- dirac\_ONE
  - GiNaC, 94
- dirac\_gamma
  - GiNaC, 95
- dirac\_gamma5
  - GiNaC, 96
- dirac\_gammaL
  - GiNaC, 96
- dirac\_gammaR
  - GiNaC, 96
- dirac\_slash
  - GiNaC, 97
- dirac\_trace
  - GiNaC, 97, 98
- dirichlet\_character
  - GiNaC, 197
- div
  - GiNaC::numeric, 771
- div\_dyn
  - GiNaC::numeric, 772
- divide
  - GiNaC, 215
- divide\_in\_z
  - GiNaC, 216
- division\_free\_elimination
  - GiNaC::matrix, 691
- do\_not\_evalf\_params

- GiNaC::function\_options, 600
- do\_print
  - GiNaC::ELi\_kernel, 474
  - GiNaC::Ebar\_kernel, 458
  - GiNaC::Eisenstein\_h\_kernel, 464
  - GiNaC::Eisenstein\_kernel, 469
  - GiNaC::Kronecker\_dtau\_kernel, 660
  - GiNaC::Kronecker\_dz\_kernel, 664
  - GiNaC::add, 326
  - GiNaC::basic, 372
  - GiNaC::basic\_log\_kernel, 375
  - GiNaC::cliffordunit, 396
  - GiNaC::constant, 424
  - GiNaC::container, 438
  - GiNaC::diracgamma, 449
  - GiNaC::diracgamma5, 450
  - GiNaC::diracgammaL, 451
  - GiNaC::diracgammaR, 453
  - GiNaC::diracone, 454
  - GiNaC::expairseq, 532
  - GiNaC::fail, 539
  - GiNaC::fderivative, 545
  - GiNaC::idx, 622
  - GiNaC::indexed, 637
  - GiNaC::integral, 647
  - GiNaC::integration\_kernel, 654
  - GiNaC::matrix, 693
  - GiNaC::minkmetric, 697
  - GiNaC::modular\_form\_kernel, 702
  - GiNaC::mul, 718
  - GiNaC::multiple\_polylog\_kernel, 746
  - GiNaC::ncmul, 754
  - GiNaC::numeric, 784
  - GiNaC::pseries, 859
  - GiNaC::relational, 882
  - GiNaC::spinidx, 905
  - GiNaC::spinmetric, 908
  - GiNaC::su3d, 929
  - GiNaC::su3f, 931
  - GiNaC::su3one, 932
  - GiNaC::su3t, 934
  - GiNaC::symbol, 949
  - GiNaC::symmetry, 957
  - GiNaC::tensdelta, 964
  - GiNaC::tensepsilon, 967
  - GiNaC::tensmetric, 971
  - GiNaC::user\_defined\_kernel, 980
  - GiNaC::varidx, 984
  - GiNaC::wildcard, 988
- do\_print\_csrc
  - GiNaC::add, 327
  - GiNaC::fderivative, 545
  - GiNaC::idx, 622
  - GiNaC::mul, 719
  - GiNaC::ncmul, 754
  - GiNaC::numeric, 784
  - GiNaC::power, 819
- do\_print\_csrc\_cl\_N
  - GiNaC::numeric, 784
  - GiNaC::power, 820
- do\_print\_dflt
  - GiNaC::clifford, 393
  - GiNaC::power, 819
- do\_print\_latex
  - GiNaC::add, 326
  - GiNaC::clifford, 393
  - GiNaC::cliffordunit, 396
  - GiNaC::constant, 424
  - GiNaC::diracgamma, 449
  - GiNaC::diracgamma5, 450
  - GiNaC::diracgammaL, 452
  - GiNaC::diracgammaR, 453
  - GiNaC::diracone, 454
  - GiNaC::fderivative, 545
  - GiNaC::idx, 623
  - GiNaC::indexed, 637
  - GiNaC::integral, 648
  - GiNaC::matrix, 693
  - GiNaC::minkmetric, 698
  - GiNaC::mul, 719
  - GiNaC::numeric, 784
  - GiNaC::power, 819
  - GiNaC::pseries, 860
  - GiNaC::spinidx, 905
  - GiNaC::spinmetric, 908
  - GiNaC::su3d, 929
  - GiNaC::su3f, 931
  - GiNaC::su3one, 933
  - GiNaC::su3t, 934
  - GiNaC::symbol, 949
  - GiNaC::tensdelta, 964
  - GiNaC::tensepsilon, 968
- do\_print\_python
  - GiNaC::container, 439
  - GiNaC::power, 819
  - GiNaC::pseries, 860
- do\_print\_python\_repr
  - GiNaC::add, 327
  - GiNaC::basic, 373
  - GiNaC::constant, 424
  - GiNaC::container, 439
  - GiNaC::matrix, 694
  - GiNaC::mul, 719
  - GiNaC::numeric, 785
  - GiNaC::power, 820
  - GiNaC::pseries, 860
  - GiNaC::relational, 882
  - GiNaC::symbol, 949
  - GiNaC::wildcard, 989
- do\_print\_tree
  - GiNaC::basic, 373
  - GiNaC::clifford, 394
  - GiNaC::constant, 424
  - GiNaC::container, 439
  - GiNaC::expairseq, 532
  - GiNaC::fderivative, 545

- GiNaC::idx, [623](#)
  - GiNaC::indexed, [637](#)
  - GiNaC::numeric, [785](#)
  - GiNaC::pseries, [860](#)
  - GiNaC::spinidx, [905](#)
  - GiNaC::symbol, [949](#)
  - GiNaC::symmetry, [957](#)
  - GiNaC::varidx, [985](#)
  - GiNaC::wildcard, [989](#)
- do\_renaming
  - GiNaC::make\_flat\_inserter, [672](#)
- domain
  - GiNaC::constant, [426](#)
- dotted
  - GiNaC::spinidx, [906](#)
- doublefactorial
  - GiNaC, [238](#)
- dummy
  - GiNaC::function\_options, [568](#)
- dump\_hierarchy
  - GiNaC::class\_info, [384](#)
- dump\_tree
  - GiNaC::class\_info, [384](#)
- duplicate
  - GiNaC::basic, [352](#)
  - GiNaC::possymbol, [807](#)
  - GiNaC::print\_functor\_impl, [834](#)
  - GiNaC::print\_memfun\_handler, [836](#)
  - GiNaC::print\_ptrfun\_handler, [839](#)
  - GiNaC::realsymbol, [870](#)
- dynallocate
  - GiNaC, [92](#)
- e
  - GiNaC::archive\_node, [345](#)
  - GiNaC::const\_iterator, [413](#)
  - GiNaC::internal::\_iter\_rep, [312](#)
- ELi\_kernel
  - GiNaC::ELi\_kernel, [473](#)
- Ebar\_kernel
  - GiNaC::Ebar\_kernel, [457](#)
- echelon\_form
  - GiNaC::matrix, [690](#)
- ef
  - GiNaC::constant, [425](#)
- Eisenstein\_h\_kernel
  - GiNaC::Eisenstein\_h\_kernel, [461](#)
- Eisenstein\_kernel
  - GiNaC::Eisenstein\_kernel, [467](#)
- element
  - GiNaC::composition\_generator::coolmulti::element, [471](#)
- EllipticE\_deriv
  - GiNaC, [161](#)
- EllipticE\_eval
  - GiNaC, [160](#)
- EllipticE\_evalf
  - GiNaC, [160](#)
- EllipticE\_print\_latex
  - GiNaC, [161](#)
- EllipticE\_series
  - GiNaC, [161](#)
- EllipticK\_deriv
  - GiNaC, [159](#)
- EllipticK\_eval
  - GiNaC, [159](#)
- EllipticK\_evalf
  - GiNaC, [159](#)
- EllipticK\_print\_latex
  - GiNaC, [160](#)
- EllipticK\_series
  - GiNaC, [160](#)
- end
  - GiNaC::archive\_node::archive\_node\_cit\_range, [346](#)
  - GiNaC::container, [438](#)
  - GiNaC::ex, [486](#)
- ensure\_if\_modifiable
  - GiNaC::basic, [372](#)
- epp
  - GiNaC, [60](#)
- epsilon\_tensor
  - GiNaC, [279](#)
- epvector
  - GiNaC, [60](#)
- error
  - GiNaC::error\_and\_integral, [477](#)
- error\_and\_integral
  - GiNaC::error\_and\_integral, [477](#)
- eta\_conjugate
  - GiNaC, [148](#)
- eta\_eval
  - GiNaC, [148](#)
- eta\_evalf
  - GiNaC, [148](#)
- eta\_imag\_part
  - GiNaC, [149](#)
- eta\_real\_part
  - GiNaC, [148](#)
- eta\_series
  - GiNaC, [148](#)
- Euler
  - GiNaC, [288](#)
- EulerEvalf
  - GiNaC, [245](#)
- eval
  - GiNaC::add, [320](#)
  - GiNaC::basic, [352](#)
  - GiNaC::ex, [487](#)
  - GiNaC::expairseq, [526](#)
  - GiNaC::fderivative, [542](#)
  - GiNaC::function, [554](#)
  - GiNaC::indexed, [632](#)
  - GiNaC::integral, [643](#)
  - GiNaC::mul, [710](#)
  - GiNaC::ncmul, [751](#)
  - GiNaC::numeric, [764](#)

- GiNaC::power, [813](#)
  - GiNaC::pseries, [852](#)
  - GiNaC::structure, [914](#)
  - GiNaC::symbol, [943](#)
  - GiNaC, [117](#)
- eval\_f
  - GiNaC::function\_options, [602](#)
- eval\_func
  - GiNaC::function\_options, [569–571](#), [595](#)
- eval\_funcp
  - GiNaC, [60](#)
- eval\_funcp\_1
  - GiNaC, [62](#)
- eval\_funcp\_10
  - GiNaC, [77](#)
- eval\_funcp\_11
  - GiNaC, [78](#)
- eval\_funcp\_12
  - GiNaC, [80](#)
- eval\_funcp\_13
  - GiNaC, [82](#)
- eval\_funcp\_14
  - GiNaC, [84](#)
- eval\_funcp\_2
  - GiNaC, [63](#)
- eval\_funcp\_3
  - GiNaC, [65](#)
- eval\_funcp\_4
  - GiNaC, [66](#)
- eval\_funcp\_5
  - GiNaC, [68](#)
- eval\_funcp\_6
  - GiNaC, [70](#)
- eval\_funcp\_7
  - GiNaC, [71](#)
- eval\_funcp\_8
  - GiNaC, [73](#)
- eval\_funcp\_9
  - GiNaC, [75](#)
- eval\_funcp\_exvector
  - GiNaC, [86](#)
- eval\_indexed
  - GiNaC::basic, [354](#)
  - GiNaC::matrix, [679](#)
  - GiNaC::minkmetric, [696](#)
  - GiNaC::spinmetric, [907](#)
  - GiNaC::structure, [915](#)
  - GiNaC::su3d, [928](#)
  - GiNaC::su3f, [931](#)
  - GiNaC::tensdelta, [963](#)
  - GiNaC::tensepsilon, [966](#)
  - GiNaC::tensmetric, [970](#)
- eval\_integ
  - GiNaC::basic, [353](#)
  - GiNaC::ex, [488](#)
  - GiNaC::integral, [646](#)
  - GiNaC::pseries, [854](#)
  - GiNaC, [117](#)
- eval\_ncmul
  - GiNaC::add, [323](#)
  - GiNaC::basic, [353](#)
  - GiNaC::clifford, [390](#)
  - GiNaC::color, [399](#)
  - GiNaC::ex, [488](#)
  - GiNaC::function, [555](#)
  - GiNaC::integral, [644](#)
  - GiNaC::mul, [714](#)
  - GiNaC::power, [817](#)
  - GiNaC::relational, [881](#)
  - GiNaC::structure, [915](#)
- eval\_use\_exvector\_args
  - GiNaC::function\_options, [606](#)
- evalchildren
  - GiNaC::expairseq, [534](#)
- evalf
  - GiNaC::basic, [353](#)
  - GiNaC::constant, [421](#)
  - GiNaC::ex, [487](#)
  - GiNaC::function, [555](#)
  - GiNaC::idx, [617](#)
  - GiNaC::integral, [643](#)
  - GiNaC::mul, [711](#)
  - GiNaC::numeric, [765](#)
  - GiNaC::power, [813](#)
  - GiNaC::pseries, [852](#)
  - GiNaC::symbol, [943](#)
  - GiNaC, [117](#), [206](#)
- evalf\_f
  - GiNaC::function\_options, [603](#)
- evalf\_func
  - GiNaC::function\_options, [571–573](#), [595](#)
- evalf\_funcp
  - GiNaC, [60](#)
- evalf\_funcp\_1
  - GiNaC, [62](#)
- evalf\_funcp\_10
  - GiNaC, [77](#)
- evalf\_funcp\_11
  - GiNaC, [78](#)
- evalf\_funcp\_12
  - GiNaC, [80](#)
- evalf\_funcp\_13
  - GiNaC, [82](#)
- evalf\_funcp\_14
  - GiNaC, [84](#)
- evalf\_funcp\_2
  - GiNaC, [63](#)
- evalf\_funcp\_3
  - GiNaC, [65](#)
- evalf\_funcp\_4
  - GiNaC, [67](#)
- evalf\_funcp\_5
  - GiNaC, [68](#)
- evalf\_funcp\_6
  - GiNaC, [70](#)
- evalf\_funcp\_7

- GiNaC, 72
- evalf\_funcp\_8
  - GiNaC, 73
- evalf\_funcp\_9
  - GiNaC, 75
- evalf\_funcp\_exvector
  - GiNaC, 86
- evalf\_params\_first
  - GiNaC::function\_options, 604
- evalf\_use\_exvector\_args
  - GiNaC::function\_options, 606
- evalffunctype
  - GiNaC, 59
- evalm
  - GiNaC::add, 320
  - GiNaC::basic, 353
  - GiNaC::ex, 488
  - GiNaC::matrix, 679
  - GiNaC::mul, 711
  - GiNaC::ncmul, 751
  - GiNaC::power, 814
  - GiNaC::pseries, 854
  - GiNaC::structure, 915
  - GiNaC, 117
- evalpoint
  - factor.cpp, 1028
- evaluate
  - GiNaC::scalar\_products, 898
- ex
  - GiNaC::basic, 373
  - GiNaC::const\_iterator, 412
  - GiNaC::ex, 484–486
- ex.cpp, 1013
- ex.h, 1013
- ex\_to
  - GiNaC::ex, 513
  - GiNaC, 122
- exadd
  - GiNaC, 251
- excompiler.cpp, 1016
- excompiler.h, 1016
- exhashmap
  - GiNaC, 87
- exmap
  - GiNaC, 59
- exminus
  - GiNaC, 251
- exmul
  - GiNaC, 251
- exp
  - GiNaC, 229
- exp\_conjugate
  - GiNaC, 176
- exp\_deriv
  - GiNaC, 175
- exp\_eval
  - GiNaC, 175
- exp\_evalf
  - GiNaC, 175
- exp\_expand
  - GiNaC, 175
- exp\_imag\_part
  - GiNaC, 176
- exp\_power
  - GiNaC, 176
- exp\_real\_part
  - GiNaC, 176
- expair
  - GiNaC::expair, 517
- expair.cpp, 1017
- expair.h, 1018
- expair\_needs\_further\_processing
  - GiNaC::expairseq, 531
  - GiNaC::mul, 716
- expairseq
  - GiNaC::expairseq, 524
- expairseq.cpp, 1019
- expairseq.h, 1019
- expand
  - GiNaC::add, 326
  - GiNaC::basic, 360
  - GiNaC::ex, 498
  - GiNaC::expairseq, 529
  - GiNaC::function, 554
  - GiNaC::indexed, 635
  - GiNaC::integral, 645
  - GiNaC::mul, 717
  - GiNaC::ncmul, 750
  - GiNaC::power, 818
  - GiNaC::pseries, 853
  - GiNaC::structure, 920
  - GiNaC, 113, 206
- expand\_add
  - GiNaC::power, 820
- expand\_add\_2
  - GiNaC::power, 820
- expand\_dummy\_sum
  - GiNaC, 137
- expand\_f
  - GiNaC::function\_options, 603
- expand\_func
  - GiNaC::function\_options, 581–583, 595
- expand\_funcp
  - GiNaC, 61
- expand\_funcp\_1
  - GiNaC, 62
- expand\_funcp\_10
  - GiNaC, 77
- expand\_funcp\_11
  - GiNaC, 79
- expand\_funcp\_12
  - GiNaC, 81
- expand\_funcp\_13
  - GiNaC, 83
- expand\_funcp\_14
  - GiNaC, 85



- expand\_funcp\_2
  - GiNaC, [64](#)
- expand\_funcp\_3
  - GiNaC, [65](#)
- expand\_funcp\_4
  - GiNaC, [67](#)
- expand\_funcp\_5
  - GiNaC, [69](#)
- expand\_funcp\_6
  - GiNaC, [70](#)
- expand\_funcp\_7
  - GiNaC, [72](#)
- expand\_funcp\_8
  - GiNaC, [74](#)
- expand\_funcp\_9
  - GiNaC, [76](#)
- expand\_funcp\_exvector
  - GiNaC, [87](#)
- expand\_map\_function
  - GiNaC::expand\_map\_function, [536](#)
- expand\_mul
  - GiNaC::power, [821](#)
- expand\_use\_exvector\_args
  - GiNaC::function\_options, [607](#)
- expandchildren
  - GiNaC::expairseq, [534](#)
  - GiNaC::mul, [719](#)
  - GiNaC::ncmul, [754](#)
- expl\_derivative
  - GiNaC::function, [559](#)
- expl\_derivative\_f
  - GiNaC::function\_options, [604](#)
- expl\_derivative\_func
  - GiNaC::function\_options, [585–587](#), [596](#)
- expl\_derivative\_funcp
  - GiNaC, [61](#)
- expl\_derivative\_funcp\_1
  - GiNaC, [63](#)
- expl\_derivative\_funcp\_10
  - GiNaC, [78](#)
- expl\_derivative\_funcp\_11
  - GiNaC, [79](#)
- expl\_derivative\_funcp\_12
  - GiNaC, [81](#)
- expl\_derivative\_funcp\_13
  - GiNaC, [83](#)
- expl\_derivative\_funcp\_14
  - GiNaC, [85](#)
- expl\_derivative\_funcp\_2
  - GiNaC, [64](#)
- expl\_derivative\_funcp\_3
  - GiNaC, [66](#)
- expl\_derivative\_funcp\_4
  - GiNaC, [67](#)
- expl\_derivative\_funcp\_5
  - GiNaC, [69](#)
- expl\_derivative\_funcp\_6
  - GiNaC, [71](#)
- expl\_derivative\_funcp\_7
  - GiNaC, [72](#)
- expl\_derivative\_funcp\_8
  - GiNaC, [74](#)
- expl\_derivative\_funcp\_9
  - GiNaC, [76](#)
- expl\_derivative\_funcp\_exvector
  - GiNaC, [87](#)
- expl\_derivative\_use\_exvector\_args
  - GiNaC::function\_options, [607](#)
- exponent
  - GiNaC::power, [822](#)
- exponop
  - GiNaC::pseries, [857](#)
- exprs
  - GiNaC::archive, [334](#)
- exprseq
  - GiNaC, [60](#)
- exprseq.cpp, [1020](#)
- exprseq.h, [1020](#)
- exprtable
  - GiNaC::archive, [335](#)
- exset
  - GiNaC, [59](#)
- exvector
  - GiNaC, [59](#)
- exvectorvector
  - GiNaC, [88](#)
- F
  - GiNaC::print\_memfun\_handler, [836](#)
  - GiNaC::print\_ptrfun\_handler, [839](#)
- f
  - GiNaC::integral, [649](#)
  - GiNaC::print\_memfun\_handler, [837](#)
  - GiNaC::print\_ptrfun\_handler, [840](#)
  - GiNaC::user\_defined\_kernel, [980](#)
- FAST\_COMPARE
  - normal.cpp, [1077](#)
- FUNCP\_1P
  - GiNaC, [59](#)
- FUNCP\_2P
  - GiNaC, [59](#)
- FUNCP\_CUBA
  - GiNaC, [60](#)
- factor
  - GiNaC, [126](#)
- factor.cpp, [1021](#)
  - c, [1023](#)
  - cache, [1025](#)
  - DCOUT2, [1022](#)
  - DCOUTVAR, [1022](#)
  - DCOUT, [1022](#)
  - evalpoint, [1028](#)
  - factors, [1025](#)
  - k, [1026](#)
  - last, [1026](#)
  - len, [1026](#)
  - lr, [1025](#)

- m, [1024](#)
  - n, [1025](#)
  - one, [1025](#)
  - options, [1028](#)
  - poly, [1027](#)
  - R, [1028](#)
  - r, [1023](#)
  - syms, [1028](#)
  - USE\_SAME\_DEGREE\_FACTOR, [1022](#)
  - value, [1023](#)
  - x, [1027](#)
- factor.h, [1029](#)
- factorial
  - GiNaC, [238](#)
- factorial\_conjugate
  - GiNaC, [152](#)
- factorial\_eval
  - GiNaC, [151](#)
- factorial\_evalf
  - GiNaC, [151](#)
- factorial\_imag\_part
  - GiNaC, [152](#)
- factorial\_print\_dflt\_latex
  - GiNaC, [152](#)
- factorial\_real\_part
  - GiNaC, [152](#)
- factors
  - factor.cpp, [1025](#)
- fail.cpp, [1029](#)
- fail.h, [1030](#)
- fderivative
  - GiNaC::fderivative, [541](#)
  - GiNaC::function\_options, [602](#)
- fderivative.cpp, [1031](#)
- fderivative.h, [1031](#)
- fibonacci
  - GiNaC, [240](#)
- find
  - GiNaC::class\_info, [384](#)
  - GiNaC::ex, [493](#)
  - GiNaC::unarchive\_table\_t, [976](#)
  - GiNaC, [114](#)
- find\_bool
  - GiNaC::archive\_node, [340](#)
- find\_common\_factor
  - GiNaC, [225](#)
- find\_dummy\_indices
  - GiNaC, [131](#)
- find\_ex
  - GiNaC::archive\_node, [341](#)
- find\_ex\_by\_loc
  - GiNaC::archive\_node, [342](#)
- find\_ex\_node
  - GiNaC::archive\_node, [342](#)
- find\_factory\_fcn
  - GiNaC, [91](#)
- find\_first
  - GiNaC::archive\_node, [341](#)
- find\_free\_and\_dummy
  - GiNaC, [130](#), [131](#)
- find\_function
  - GiNaC::function, [561](#)
- find\_last
  - GiNaC::archive\_node, [341](#)
- find\_property\_range
  - GiNaC::archive\_node, [341](#)
- find\_real\_imag
  - GiNaC::mul, [718](#)
- find\_string
  - GiNaC::archive\_node, [340](#)
- find\_unsigned
  - GiNaC::archive\_node, [340](#)
- find\_variant\_indices
  - GiNaC, [133](#)
- finished
  - GiNaC::composition\_generator::coolmulti, [444](#)
- first
  - GiNaC::class\_info, [385](#)
- flag\_overflow
  - GiNaC::basic\_multi\_iterator, [381](#)
- flags
  - GiNaC::basic, [373](#)
- flags.h, [1032](#)
- force\_include\_tgamma
  - GiNaC, [289](#)
- force\_include\_zeta1
  - GiNaC, [289](#)
- forget
  - GiNaC::archive, [333](#)
  - GiNaC::archive\_node, [343](#)
- format\_index\_value
  - GiNaC, [283](#), [284](#)
- frac\_cancel
  - GiNaC, [224](#)
- fraction\_free\_elimination
  - GiNaC::matrix, [692](#)
- fsolve
  - GiNaC, [155](#)
- func\_arg\_info
  - GiNaC, [139](#)
- function
  - GiNaC::function, [549–553](#)
  - GiNaC::function\_options, [602](#)
- function.cpp, [1033](#)
- function.h, [1033](#)
  - DECLARE\_FUNCTION\_10P, [1043](#)
  - DECLARE\_FUNCTION\_11P, [1043](#)
  - DECLARE\_FUNCTION\_12P, [1043](#)
  - DECLARE\_FUNCTION\_13P, [1044](#)
  - DECLARE\_FUNCTION\_14P, [1044](#)
  - DECLARE\_FUNCTION\_1P, [1040](#)
  - DECLARE\_FUNCTION\_2P, [1040](#)
  - DECLARE\_FUNCTION\_3P, [1040](#)
  - DECLARE\_FUNCTION\_4P, [1041](#)
  - DECLARE\_FUNCTION\_5P, [1041](#)
  - DECLARE\_FUNCTION\_6P, [1041](#)

- DECLARE\_FUNCTION\_7P, [1042](#)
- DECLARE\_FUNCTION\_8P, [1042](#)
- DECLARE\_FUNCTION\_9P, [1042](#)
- is\_ex\_the\_function, [1045](#)
- REGISTER\_FUNCTION, [1044](#)
- function\_options
  - GiNaC::function\_options, [567](#), [568](#)
- functions\_with\_same\_name
  - GiNaC::function\_options, [608](#)
- G
  - GiNaC, [157](#)
- G2\_eval
  - GiNaC, [169](#)
- G2\_evalf
  - GiNaC, [168](#)
- G3\_eval
  - GiNaC, [169](#)
- G3\_evalf
  - GiNaC, [169](#)
- GINAC\_ASSERT
  - assertion.h, [998](#)
- GINAC\_BIND\_UNARCHIVER
  - archive.h, [997](#)
  - GiNaC, [90](#), [93](#), [94](#), [105](#), [111](#), [127](#), [129](#), [132](#), [195](#), [198–201](#), [203](#), [209](#), [210](#), [227](#), [263](#), [264](#), [266–268](#), [276](#), [277](#), [286](#)
- GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE
  - print.h, [1092](#)
- GINAC\_DECLARE\_PRINT\_CONTEXT\_COMMON
  - print.h, [1092](#)
- GINAC\_DECLARE\_PRINT\_CONTEXT
  - print.h, [1093](#)
- GINAC\_DECLARE\_REGISTERED\_CLASS\_COMMON
  - registrar.h, [1097](#)
- GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CT↔ORS
  - registrar.h, [1098](#)
- GINAC\_DECLARE\_REGISTERED\_CLASS
  - registrar.h, [1098](#)
- GINAC\_DECLARE\_UNARCHIVER
  - archive.h, [996](#)
  - GiNaC, [90](#), [103](#), [104](#), [110](#), [111](#), [127](#), [130](#), [131](#), [137](#), [195](#), [201–203](#), [205](#), [209](#), [210](#), [245](#), [263](#), [264](#), [266–268](#), [272](#), [280](#), [281](#), [287](#)
- GINAC\_IMPLEMENT\_PRINT\_CONTEXT
  - print.h, [1093](#)
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T
  - registrar.h, [1099](#)
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT
  - GiNaC, [89](#), [91](#), [93](#), [104](#), [111](#), [127](#), [128](#), [132](#), [194](#), [198–201](#), [203](#), [208](#), [210](#), [226](#), [262](#), [264](#), [266](#), [268](#), [276](#), [286](#)
  - registrar.h, [1099](#)
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS
  - registrar.h, [1099](#)
- GINAC\_LT\_AGE
  - version.h, [1117](#)
- GINAC\_LT\_CURRENT
  - version.h, [1117](#)
- GINAC\_LT\_REVISION
  - version.h, [1117](#)
- GINACLIB\_ARCHIVE\_AGE
  - version.h, [1117](#)
- GINACLIB\_ARCHIVE\_VERSION
  - version.h, [1117](#)
- GINACLIB\_MAJOR\_VERSION
  - version.h, [1116](#)
- GINACLIB\_MICRO\_VERSION
  - version.h, [1117](#)
- GINACLIB\_MINOR\_VERSION
  - version.h, [1117](#)
- GINACLIB\_STR\_HELPER
  - version.h, [1118](#)
- GINACLIB\_STR
  - version.h, [1118](#)
- GINACLIB\_VERSION
  - version.h, [1118](#)
- gauss\_elimination
  - GiNaC::matrix, [691](#)
- gcd
  - GiNaC, [220](#), [243](#)
- gcd\_pf\_mul
  - GiNaC, [220](#)
- gcd\_pf\_pow
  - GiNaC, [219](#)
- gcd\_pf\_pow\_pow
  - GiNaC, [221](#)
- generalised\_Bernoulli\_number
  - GiNaC, [197](#)
- get
  - GiNaC::composition\_generator, [406](#)
  - GiNaC::partition\_generator, [787](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [789](#)
- get\_TeX\_name
  - GiNaC::symbol, [949](#)
- get\_all\_dummy\_indices
  - GiNaC, [136](#)
- get\_all\_dummy\_indices\_safely
  - GiNaC, [135](#)
- get\_cache\_size
  - GiNaC::integration\_kernel, [653](#)
- get\_class\_name
  - GiNaC::structure, [914](#)
- get\_clifford\_comp
  - GiNaC, [101](#)
- get\_close\_delim
  - GiNaC::container, [430](#)
- get\_commutator\_sign
  - GiNaC::clifford, [392](#)
- get\_default\_TeX\_name
  - GiNaC, [267](#)
- get\_default\_flags
  - GiNaC::container, [430](#)
- get\_dim
  - GiNaC::idx, [621](#)
- get\_dim\_uint

- GiNaC, 95
- get\_domain
  - GiNaC::possymbol, 807
  - GiNaC::realsymbol, 869
  - GiNaC::symbol, 948
- get\_dummy\_indices
  - GiNaC::indexed, 636
- get\_ex
  - GiNaC::archive\_node, 343
- get\_factors
  - GiNaC::ncmul, 755
- get\_first\_symbol
  - GiNaC, 210
- get\_free\_indices
  - GiNaC::add, 323
  - GiNaC::basic, 363
  - GiNaC::ex, 505
  - GiNaC::indexed, 633
  - GiNaC::integral, 645
  - GiNaC::mul, 713
  - GiNaC::ncmul, 752
  - GiNaC::structure, 923
- get\_id
  - GiNaC::print\_context\_options, 825
  - GiNaC::registered\_class\_options, 874
- get\_indices
  - GiNaC::indexed, 635
- get\_label
  - GiNaC::wildcard, 988
- get\_last\_access
  - GiNaC::remember\_table\_entry, 890
- get\_metric
  - GiNaC::clifford, 391
- get\_name
  - GiNaC::function, 561
  - GiNaC::function\_options, 600
  - GiNaC::print\_context\_options, 825
  - GiNaC::registered\_class\_options, 874
  - GiNaC::symbol, 948
- get\_node
  - GiNaC::archive, 332
- get\_nparams
  - GiNaC::function\_options, 601
- get\_numerical\_value
  - GiNaC::ELi\_kernel, 474
  - GiNaC::Ebar\_kernel, 458
  - GiNaC::Eisenstein\_h\_kernel, 463
  - GiNaC::Eisenstein\_kernel, 468
  - GiNaC::Kronecker\_dtau\_kernel, 659
  - GiNaC::Kronecker\_dz\_kernel, 663
  - GiNaC::integration\_kernel, 652
  - GiNaC::modular\_form\_kernel, 701
- get\_numerical\_value\_impl
  - GiNaC::integration\_kernel, 654
- get\_open\_delim
  - GiNaC::container, 430
- get\_order
  - GiNaC::lanczos\_coeffs, 666
- get\_parent
  - GiNaC::class\_info, 383
- get\_parent\_name
  - GiNaC::print\_context\_options, 825
  - GiNaC::registered\_class\_options, 874
- get\_point
  - GiNaC::pseries, 856
- get\_pointer
  - GiNaC::ptr, 866
- get\_print\_context
  - GiNaC, 259
- get\_print\_dispatch\_table
  - GiNaC::registered\_class\_options, 874
- get\_print\_options
  - GiNaC, 259
- get\_properties
  - GiNaC::archive\_node, 342
- get\_refcount
  - GiNaC::refcounted, 872
- get\_registered\_functions
  - GiNaC::function, 561
- get\_representation\_label
  - GiNaC::clifford, 391
  - GiNaC::color, 401
  - GiNaC, 97, 108
- get\_result
  - GiNaC::remember\_table\_entry, 890
- get\_serial
  - GiNaC::function, 561
- get\_series\_coeff
  - GiNaC::integration\_kernel, 653
- get\_sign
  - GiNaC::multi\_iterator\_permutation, 737
- get\_struct
  - GiNaC::structure, 927
- get\_successful\_hits
  - GiNaC::remember\_table\_entry, 890
- get\_symbol\_stats
  - GiNaC, 211
- get\_symmetry
  - GiNaC::indexed, 636
- get\_top\_node
  - GiNaC::archive, 332
- get\_type
  - GiNaC::symmetry, 956
- get\_value
  - GiNaC::idx, 620
- get\_var
  - GiNaC::pseries, 855
- get\_vector
  - GiNaC::basic\_multi\_iterator, 378
- gethash
  - GiNaC::basic, 371
  - GiNaC::ex, 510
- GiNaC::\_numeric\_digits, 313
  - \_numeric\_digits, 313
  - add\_callback, 314
  - callbacklist, 315

- digits, 315
- operator long, 314
- operator=, 314
- print, 314
- too\_late, 315
- GiNaC::ELi\_kernel, 472
  - do\_print, 474
  - ELi\_kernel, 473
  - get\_numerical\_value, 474
  - is\_numeric, 474
  - let\_op, 473
  - m, 475
  - n, 475
  - nops, 473
  - op, 473
  - series\_coeff\_impl, 474
  - x, 475
  - y, 475
- GiNaC::Ebar\_kernel, 456
  - do\_print, 458
  - Ebar\_kernel, 457
  - get\_numerical\_value, 458
  - is\_numeric, 458
  - let\_op, 457
  - m, 459
  - n, 459
  - nops, 457
  - op, 457
  - series\_coeff\_impl, 458
  - x, 459
  - y, 459
- GiNaC::Eisenstein\_h\_kernel, 460
  - C\_norm, 465
  - coefficient\_a0, 463
  - coefficient\_an, 463
  - do\_print, 464
  - Eisenstein\_h\_kernel, 461
  - get\_numerical\_value, 463
  - is\_numeric, 462
  - k, 464
  - Laurent\_series, 462
  - let\_op, 462
  - N, 464
  - nops, 461
  - op, 462
  - q\_expansion\_modular\_form, 464
  - r, 465
  - s, 465
  - series, 461
  - uses\_Laurent\_series, 463
- GiNaC::Eisenstein\_kernel, 465
  - a, 470
  - b, 470
  - C\_norm, 470
  - do\_print, 469
  - Eisenstein\_kernel, 467
  - get\_numerical\_value, 468
  - is\_numeric, 468
  - K, 470
  - k, 469
  - Laurent\_series, 468
  - let\_op, 468
  - N, 470
  - nops, 467
  - op, 467
  - q\_expansion\_modular\_form, 469
  - series, 467
  - uses\_Laurent\_series, 469
- GiNaC::G2\_SERIAL, 608
  - serial, 609
- GiNaC::G3\_SERIAL, 609
  - serial, 609
- GiNaC::Kronecker\_dtau\_kernel, 657
  - C\_norm, 661
  - do\_print, 660
  - get\_numerical\_value, 659
  - is\_numeric, 659
  - K, 661
  - Kronecker\_dtau\_kernel, 658
  - let\_op, 659
  - n, 660
  - nops, 659
  - op, 659
  - series\_coeff\_impl, 660
  - z, 660
- GiNaC::Kronecker\_dz\_kernel, 661
  - C\_norm, 665
  - do\_print, 664
  - get\_numerical\_value, 663
  - is\_numeric, 663
  - K, 665
  - Kronecker\_dz\_kernel, 662
  - let\_op, 663
  - n, 664
  - nops, 663
  - op, 663
  - series\_coeff\_impl, 664
  - tau, 665
  - z\_j, 664
- GiNaC::add, 315
  - add, 317, 318
  - coeff, 320
  - combine\_ex\_with\_coeff\_to\_pair, 325
  - combine\_pair\_with\_coeff\_to\_pair, 325
  - conjugate, 322
  - degree, 319
  - derivative, 323
  - do\_print, 326
  - do\_print\_csrc, 327
  - do\_print\_latex, 326
  - do\_print\_python\_repr, 327
  - eval, 320
  - eval\_ncmul, 323
  - evalm, 320
  - expand, 326
  - get\_free\_indices, 323

- imag\_part, 323
- info, 319
- integer\_content, 321
- is\_polynomial, 319
- ldegree, 320
- max\_coefficient, 322
- mul, 327
- normal, 321
- power, 327
- precedence, 318
- print\_add, 326
- real\_part, 323
- recombine\_pair\_to\_ex, 325
- return\_type, 324
- return\_type\_tinfo, 324
- series, 321
- smod, 322
- split\_ex\_to\_pair, 325
- thisexpairseq, 324
- GiNaC::archive, 328
  - ~archive, 329
  - add\_node, 332
  - archive, 329, 330
  - archive\_ex, 330
  - atomize, 333
  - atoms, 335
  - clear, 332
  - exprs, 334
  - exprtable, 335
  - forget, 333
  - get\_node, 332
  - get\_top\_node, 332
  - inv\_at\_cit, 329
  - inverse\_atoms, 335
  - nodes, 334
  - num\_expressions, 332
  - operator<<, 334
  - operator>>, 334
  - printraw, 333
  - unarchive\_ex, 330, 331
  - unatomize, 333
- GiNaC::archive::archived\_ex, 346
  - archived\_ex, 346, 347
  - name, 347
  - root, 347
- GiNaC::archive\_node, 335
  - a, 344
  - add\_bool, 339
  - add\_ex, 339
  - add\_string, 339
  - add\_unsigned, 339
  - archive\_node, 338
  - archive\_node\_cit, 337
  - e, 345
  - find\_bool, 340
  - find\_ex, 341
  - find\_ex\_by\_loc, 342
  - find\_ex\_node, 342
  - find\_first, 341
  - find\_last, 341
  - find\_property\_range, 341
  - find\_string, 340
  - find\_unsigned, 340
  - forget, 343
  - get\_ex, 343
  - get\_properties, 342
  - has\_ex, 343
  - has\_expression, 345
  - has\_same\_ex\_as, 343
  - operator<<, 344
  - operator>>, 344
  - operator=, 339
  - printraw, 343
  - property\_type, 338
  - propinfovector, 337
  - props, 344
  - unarchive, 342
- GiNaC::archive\_node::archive\_node\_cit\_range, 345
  - begin, 345
  - end, 346
- GiNaC::archive\_node::property, 843
  - name, 844
  - property, 844
  - type, 844
  - value, 844
- GiNaC::archive\_node::property\_info, 845
  - count, 846
  - name, 846
  - property\_info, 845, 846
  - type, 846
- GiNaC::basic, 348
  - ~basic, 351
  - accept, 359
  - add\_indexed, 364
  - archive, 368
  - basic, 351, 352
  - calchash, 367
  - clearflag, 372
  - coeff, 360
  - collect, 360
  - compare, 370
  - compare\_same\_type, 366
  - conjugate, 366
  - contract\_with, 365
  - dbgprint, 354
  - dbgprinttree, 355
  - degree, 359
  - derivative, 361
  - diff, 369
  - do\_print, 372
  - do\_print\_python\_repr, 373
  - do\_print\_tree, 373
  - duplicate, 352
  - ensure\_if\_modifiable, 372
  - eval, 352
  - eval\_indexed, 354

- eval\_integ, 353
- eval\_ncmul, 353
- evalf, 353
- evalm, 353
- ex, 373
- expand, 360
- flags, 373
- get\_free\_indices, 363
- gethash, 371
- has, 357
- hashvalue, 374
- hold, 370
- imag\_part, 366
- info, 355
- integer\_content, 362
- is\_equal, 370
- is\_equal\_same\_type, 367
- is\_polynomial, 359
- ldegree, 359
- let\_op, 356
- map, 358
- match, 357
- match\_same\_type, 358
- max\_coefficient, 363
- nops, 355
- normal, 361
- op, 356
- operator=, 352
- operator[], 356, 357
- precedence, 355
- print, 354
- print\_dispatch, 367, 368
- read\_archive, 368
- real\_part, 366
- return\_type, 365
- return\_type\_tinfo, 366
- scalar\_mul\_indexed, 364
- series, 361
- setflag, 371
- smod, 363
- subs, 358
- subs\_one\_level, 369
- to\_polynomial, 362
- to\_rational, 362
- GiNaC::basic\_log\_kernel, 374
  - do\_print, 375
  - series\_coeff\_impl, 375
- GiNaC::basic\_multi\_iterator
  - ~basic\_multi\_iterator, 378
  - B, 380
  - basic\_multi\_iterator, 377
  - flag\_overflow, 381
  - get\_vector, 378
  - init, 379
  - N, 380
  - operator<<, 380
  - operator(), 379
  - operator++, 379
  - operator[], 378, 379
  - overflow, 378
  - size, 378
  - v, 380
- GiNaC::basic\_multi\_iterator< T >, 376
- GiNaC::basic\_partition\_generator, 381
  - basic\_partition\_generator, 382
  - mpgen, 382
- GiNaC::basic\_partition\_generator::mpartition2, 703
  - m, 704
  - mpartition2, 703
  - n, 704
  - next\_partition, 703
  - x, 704
- GiNaC::class\_info
  - class\_info, 383
  - dump\_hierarchy, 384
  - dump\_tree, 384
  - find, 384
  - first, 385
  - get\_parent, 383
  - identify\_parents, 384
  - next, 385
  - options, 385
  - parent, 385
  - parents\_identified, 385
- GiNaC::class\_info< OPT >, 382
- GiNaC::class\_info< OPT >::tree\_node, 974
- GiNaC::class\_info::tree\_node
  - add\_child, 975
  - children, 975
  - info, 975
  - tree\_node, 975
- GiNaC::clifford, 386
  - archive, 389
  - clifford, 387, 388
  - commutator\_sign, 394
  - do\_print\_dflt, 393
  - do\_print\_latex, 393
  - do\_print\_tree, 394
  - eval\_ncmul, 390
  - get\_commutator\_sign, 392
  - get\_metric, 391
  - get\_representation\_label, 391
  - let\_op, 393
  - match\_same\_type, 390
  - metric, 394
  - nops, 392
  - op, 392
  - precedence, 389
  - read\_archive, 389
  - representation\_label, 394
  - return\_type, 391
  - return\_type\_tinfo, 391
  - same\_metric, 392
  - subs, 393
  - thiscontainer, 390, 391
- GiNaC::cliffordunit, 395

- contract\_with, 396
- do\_print, 396
- do\_print\_latex, 396
- GiNaC::color, 396
  - archive, 399
  - color, 397, 398
  - eval\_ncmul, 399
  - get\_representation\_label, 401
  - match\_same\_type, 399
  - read\_archive, 399
  - representation\_label, 401
  - return\_type, 400
  - return\_type\_tinfo, 400
  - thiscontainer, 400
- GiNaC::compare\_all\_equal
  - ~compare\_all\_equal, 402
  - struct\_compare, 402
  - struct\_is\_equal, 402
- GiNaC::compare\_all\_equal< T >, 401
- GiNaC::compare\_bitwise
  - ~compare\_bitwise, 403
  - struct\_compare, 403
  - struct\_is\_equal, 403
- GiNaC::compare\_bitwise< T >, 403
- GiNaC::compare\_std\_less
  - ~compare\_std\_less, 404
  - struct\_compare, 405
  - struct\_is\_equal, 404
- GiNaC::compare\_std\_less< T >, 404
- GiNaC::composition\_generator, 405
  - atend, 407
  - cmgen, 406
  - composition, 407
  - composition\_generator, 406
  - current\_updated, 407
  - get, 406
  - next, 406
  - trivial, 407
- GiNaC::composition\_generator::coolmulti, 443
  - ~coolmulti, 444
  - after\_i, 445
  - coolmulti, 444
  - finished, 444
  - head, 445
  - i, 445
  - next\_permutation, 444
- GiNaC::composition\_generator::coolmulti::element, 471
  - ~element, 471
  - element, 471
  - next, 471
  - value, 471
- GiNaC::const\_iterator, 408
  - const\_iterator, 409
  - const\_postorder\_iterator, 412
  - const\_preorder\_iterator, 412
  - e, 413
  - ex, 412
  - i, 413
  - operator!=, 411
  - operator<, 411
  - operator<=, 411
  - operator>, 411
  - operator>=, 412
  - operator\*, 409
  - operator+, 410, 412
  - operator++, 409, 410
  - operator+=, 410
  - operator-, 411, 412
  - operator->, 409
  - operator--, 410
  - operator=, 410
  - operator==, 411
  - operator[], 409
- GiNaC::const\_postorder\_iterator, 413
  - const\_postorder\_iterator, 414
  - descend, 415
  - increment, 415
  - operator!=, 415
  - operator\*, 414
  - operator++, 414, 415
  - operator->, 414
  - operator==, 415
  - s, 416
- GiNaC::const\_preorder\_iterator, 416
  - const\_preorder\_iterator, 417
  - increment, 418
  - operator!=, 418
  - operator\*, 417
  - operator++, 417
  - operator->, 417
  - operator==, 418
  - s, 418
- GiNaC::constant, 419
  - archive, 422
  - calchash, 423
  - conjugate, 422
  - constant, 420
  - derivative, 423
  - do\_print, 424
  - do\_print\_latex, 424
  - do\_print\_python\_repr, 424
  - do\_print\_tree, 424
  - domain, 426
  - ef, 425
  - evalf, 421
  - imag\_part, 422
  - info, 421
  - is\_equal\_same\_type, 423
  - is\_polynomial, 421
  - name, 425
  - next\_serial, 425
  - number, 425
  - read\_archive, 422
  - real\_part, 422
  - serial, 425
  - TeX\_name, 425



- GiNaC::container
  - append, 436
  - archive, 433
  - begin, 437
  - conjugate, 433
  - const\_iterator, 429
  - const\_reverse\_iterator, 429
  - container, 429, 430
  - do\_print, 438
  - do\_print\_python, 439
  - do\_print\_python\_repr, 439
  - do\_print\_tree, 439
  - end, 438
  - get\_close\_delim, 430
  - get\_default\_flags, 430
  - get\_open\_delim, 430
  - imag\_part, 434
  - info, 430
  - is\_equal\_same\_type, 434
  - let\_op, 432
  - nops, 431
  - op, 431
  - precedence, 431
  - prepend, 436
  - printseq, 435
  - rbegin, 438
  - read\_archive, 432
  - real\_part, 433
  - remove\_all, 437
  - remove\_first, 436
  - remove\_last, 437
  - rend, 438
  - STLT, 429
  - sort, 437
  - sort\_, 435, 436
  - subs, 432
  - subchildren, 439
  - thiscontainer, 434, 435
  - unique, 437
  - unique\_, 436, 439
- GiNaC::container< C >, 426
- GiNaC::container\_storage
  - ~container\_storage, 442
  - container\_storage, 441
  - reserve, 442
  - STLT, 441
  - seq, 443
- GiNaC::container\_storage< C >, 440
- GiNaC::derivative\_map\_function, 445
  - derivative\_map\_function, 446
  - operator(), 446
  - s, 446
- GiNaC::determinant\_algo, 447
- GiNaC::diracgamma, 448
  - contract\_with, 448
  - do\_print, 449
  - do\_print\_latex, 449
- GiNaC::diracgamma5, 449
  - conjugate, 450
  - do\_print, 450
  - do\_print\_latex, 450
- GiNaC::diracgammaL, 451
  - conjugate, 451
  - do\_print, 451
  - do\_print\_latex, 452
- GiNaC::diracgammaR, 452
  - conjugate, 453
  - do\_print, 453
  - do\_print\_latex, 453
- GiNaC::diracone, 453
  - do\_print, 454
  - do\_print\_latex, 454
- GiNaC::do\_taylor, 454
- GiNaC::domain, 455
- GiNaC::dunno, 455
- GiNaC::error\_and\_integral, 476
  - error, 477
  - error\_and\_integral, 477
  - integral, 477
- GiNaC::error\_and\_integral\_is\_less, 477
  - operator(), 477
- GiNaC::eval\_integ\_map\_function, 478
  - operator(), 478
- GiNaC::evalf\_map\_function, 479
  - operator(), 479
- GiNaC::evalm\_map\_function, 479
  - operator(), 480
- GiNaC::ex, 480
  - accept, 495
  - antisymmetrize, 508, 509
  - archive\_node, 513
  - are\_ex\_trivially\_equal, 513
  - begin, 486
  - bp, 514
  - coeff, 497
  - collect, 498
  - compare, 507
  - conjugate, 492
  - construct\_from\_basic, 510
  - construct\_from\_double, 512
  - construct\_from\_int, 510
  - construct\_from\_long, 511
  - construct\_from\_longlong, 511
  - construct\_from\_string\_and\_lst, 512
  - construct\_from\_uint, 511
  - construct\_from\_ulong, 511
  - construct\_from\_ulonglong, 512
  - content, 502
  - dbgprint, 489
  - dbgprinttree, 489
  - degree, 497
  - denom, 501
  - diff, 498
  - end, 486
  - eval, 487
  - eval\_integ, 488

- eval\_ncmul, 488
- evalf, 487
- evalm, 488
- ex, 484–486
- ex\_to, 513
- expand, 498
- find, 493
- get\_free\_indices, 505
- gethash, 510
- has, 493
- imag\_part, 493
- info, 489
- integer\_content, 503
- is\_a, 514
- is\_equal, 507
- is\_exactly\_a, 514
- is\_polynomial, 496
- is\_zero, 507
- is\_zero\_matrix, 508
- lcoeff, 497
- ldegree, 497
- let\_op, 491
- lhs, 492
- makewritable, 512
- map, 495
- match, 493, 494
- max\_coefficient, 505
- nops, 490
- normal, 499
- numer, 501
- numer\_denom, 501
- op, 490
- operator[], 491, 492
- postorder\_begin, 487
- postorder\_end, 487
- preorder\_begin, 487
- preorder\_end, 487
- primpart, 503, 504
- print, 488
- real\_part, 492
- return\_type, 509
- return\_type\_tinfo, 510
- rhs, 492
- series, 499
- share, 512
- simplify\_indexed, 506
- smod, 505
- subs, 494, 495
- swap, 486
- symmetrize, 508
- symmetrize\_cyclic, 509
- tcoeff, 498
- to\_polynomial, 500
- to\_rational, 500
- traverse, 496
- traverse\_postorder, 496
- traverse\_preorder, 496
- unit, 502
- unitcontprim, 504
- GiNaC::ex\_base\_is\_less, 515
  - operator(), 515
- GiNaC::ex\_is\_equal, 515
  - operator(), 515
- GiNaC::ex\_is\_less, 516
  - operator(), 516
- GiNaC::ex\_swap, 516
  - operator(), 516
- GiNaC::expair, 517
  - coeff, 519
  - compare, 518
  - conjugate, 519
  - expair, 517
  - is\_canonical\_numeric, 519
  - is\_equal, 518
  - is\_less, 518
  - print, 518
  - rest, 519
  - swap, 519
- GiNaC::expair\_is\_less, 520
  - operator(), 520
- GiNaC::expair\_rest\_is\_less, 521
  - operator(), 521
- GiNaC::expair\_swap, 521
  - operator(), 521
- GiNaC::expairseq, 522
  - archive, 527
  - calchash, 529
  - can\_make\_flat, 532
  - canonicalize, 533
  - combine\_ex\_with\_coeff\_to\_pair, 530
  - combine\_overall\_coeff, 531
  - combine\_pair\_with\_coeff\_to\_pair, 530
  - combine\_same\_terms\_sorted\_seq, 534
  - conjugate, 527
  - construct\_from\_2\_ex, 532
  - construct\_from\_2\_expairseq, 532
  - construct\_from\_epvector, 533
  - construct\_from\_expairseq\_ex, 532
  - construct\_from\_exvector, 533
  - default\_overall\_coeff, 531
  - do\_print, 532
  - do\_print\_tree, 532
  - eval, 526
  - evalchildren, 534
  - expair\_needs\_further\_processing, 531
  - expairseq, 524
  - expand, 529
  - expandchildren, 534
  - info, 525
  - is\_canonical, 534
  - is\_equal\_same\_type, 528
  - make\_flat, 533
  - map, 526
  - match, 527
  - nops, 525
  - op, 525

- overall\_coeff, 535
- precedence, 525
- printpair, 530
- printseq, 530
- read\_archive, 528
- recombine\_pair\_to\_ex, 531
- return\_type, 528
- seq, 534
- split\_ex\_to\_pair, 530
- subs, 527
- subchildren, 534
- thisexpairseq, 529
- to\_polynomial, 526
- to\_rational, 526
- GiNaC::expand\_map\_function, 535
  - expand\_map\_function, 536
  - operator(), 536
  - options, 536
- GiNaC::expand\_options, 537
- GiNaC::factor\_options, 537
- GiNaC::fail, 538
  - do\_print, 539
  - return\_type, 538
- GiNaC::fderivative, 539
  - archive, 543
  - derivative, 543
  - derivatives, 544
  - do\_print, 545
  - do\_print\_csrc, 545
  - do\_print\_latex, 545
  - do\_print\_tree, 545
  - eval, 542
  - fderivative, 541
  - is\_equal\_same\_type, 544
  - match\_same\_type, 544
  - parameter\_set, 546
  - print, 542
  - read\_archive, 543
  - series, 542
  - thiscontainer, 542, 543
- GiNaC::function, 546
  - archive, 557
  - calchash, 555
  - conjugate, 556
  - current\_serial, 562
  - derivative, 558
  - eval, 554
  - eval\_ncmul, 555
  - evalf, 555
  - expand, 554
  - expl\_derivative, 559
  - find\_function, 561
  - function, 549–553
  - get\_name, 561
  - get\_registered\_functions, 561
  - get\_serial, 561
  - imag\_part, 557
  - info, 557
  - is\_equal\_same\_type, 558
  - lookup\_remember\_table, 560
  - match\_same\_type, 558
  - pderivative, 559
  - power, 560
  - precedence, 554
  - print, 553
  - read\_archive, 557
  - real\_part, 556
  - register\_new, 560
  - registered\_functions, 560
  - remember\_table\_entry, 562
  - return\_type, 559
  - return\_type\_tinfo, 559
  - serial, 562
  - series, 555
  - store\_remember\_table, 560
  - thiscontainer, 556
- GiNaC::function\_options, 562
  - ~function\_options, 568
  - conjugate\_f, 603
  - conjugate\_func, 574–576, 595
  - conjugate\_use\_exvector\_args, 606
  - derivative\_f, 603
  - derivative\_func, 583–585, 596
  - derivative\_use\_exvector\_args, 607
  - do\_not\_evalf\_params, 600
  - dummy, 568
  - eval\_f, 602
  - eval\_func, 569–571, 595
  - eval\_use\_exvector\_args, 606
  - evalf\_f, 603
  - evalf\_func, 571–573, 595
  - evalf\_params\_first, 604
  - evalf\_use\_exvector\_args, 606
  - expand\_f, 603
  - expand\_func, 581–583, 595
  - expand\_use\_exvector\_args, 607
  - expl\_derivative\_f, 604
  - expl\_derivative\_func, 585–587, 596
  - expl\_derivative\_use\_exvector\_args, 607
  - fderivative, 602
  - function, 602
  - function\_options, 567, 568
  - functions\_with\_same\_name, 608
  - get\_name, 600
  - get\_nparams, 601
  - has\_derivative, 601
  - has\_power, 601
  - imag\_part\_f, 603
  - imag\_part\_func, 578–580, 595
  - imag\_part\_use\_exvector\_args, 606
  - info\_f, 604
  - info\_func, 592–594, 596
  - info\_use\_exvector\_args, 608
  - initialize, 568
  - latex\_name, 569
  - name, 602

- nparams, 602
- overloaded, 600
- power\_f, 604
- power\_func, 588–590, 596
- power\_use\_exvector\_args, 607
- print\_dispatch\_table, 604
- print\_func, 596–599
- print\_use\_exvector\_args, 607
- real\_part\_f, 603
- real\_part\_func, 576–578, 595
- real\_part\_use\_exvector\_args, 606
- remember, 600
- remember\_assoc\_size, 605
- remember\_size, 605
- remember\_strategy, 606
- return\_type, 605
- return\_type\_tinfo, 605
- series\_f, 604
- series\_func, 590–592, 596
- series\_use\_exvector\_args, 607
- set\_name, 569
- set\_print\_func, 601
- set\_return\_type, 599
- set\_symmetry, 600
- symtree, 608
- TeX\_name, 602
- test\_and\_set\_nparams, 601
- use\_remember, 605
- use\_return\_type, 605
- GiNaC::gcd\_options, 610
- GiNaC::gcdheu\_failed, 611
- GiNaC::has\_distance
  - no\_type, 612
  - test, 612
  - yes\_type, 612
- GiNaC::has\_distance < T >, 611
- GiNaC::has\_options, 613
- GiNaC::idx, 614
  - archive, 618
  - calchash, 619
  - derivative, 619
  - dim, 623
  - do\_print, 622
  - do\_print\_csrc, 622
  - do\_print\_latex, 623
  - do\_print\_tree, 623
  - evalf, 617
  - get\_dim, 621
  - get\_value, 620
  - idx, 616
  - info, 616
  - is\_dim\_numeric, 621
  - is\_dim\_symbolic, 621
  - is\_dummy\_pair\_same\_type, 620
  - is\_numeric, 620
  - is\_symbolic, 620
  - map, 617
  - match\_same\_type, 619
  - minimal\_dim, 622
  - nops, 617
  - op, 617
  - print\_index, 622
  - read\_archive, 618
  - replace\_dim, 621
  - subs, 618
  - value, 623
- GiNaC::idx\_is\_equal\_ignore\_dim, 624
  - operator(), 624
- GiNaC::indexed, 624
  - all\_index\_values\_are, 635
  - archive, 633
  - derivative, 634
  - do\_print, 637
  - do\_print\_latex, 637
  - do\_print\_tree, 637
  - eval, 632
  - expand, 635
  - get\_dummy\_indices, 636
  - get\_free\_indices, 633
  - get\_indices, 635
  - get\_symmetry, 636
  - has\_dummy\_index\_for, 636
  - imag\_part, 633
  - indexed, 626–631
  - info, 632
  - precedence, 632
  - print\_indexed, 637
  - printindices, 636
  - read\_archive, 633
  - real\_part, 632
  - reposition\_dummy\_indices, 638
  - return\_type, 634
  - return\_type\_tinfo, 635
  - simplify\_indexed, 638
  - simplify\_indexed\_product, 638
  - symtree, 639
  - thiscontainer, 634
  - validate, 637
- GiNaC::info\_flags, 639
- GiNaC::integral, 641
  - a, 648
  - archive, 646
  - b, 649
  - conjugate, 646
  - degree, 643
  - derivative, 647
  - do\_print, 647
  - do\_print\_latex, 648
  - eval, 643
  - eval\_integ, 646
  - eval\_ncmul, 644
  - evalf, 643
  - expand, 645
  - f, 649
  - get\_free\_indices, 645
  - integral, 642

- ldegree, 643
- let\_op, 644
- max\_integration\_level, 648
- nops, 644
- op, 644
- precedence, 642
- read\_archive, 646
- relative\_integration\_error, 648
- return\_type, 645
- return\_type\_tinfo, 645
- series, 647
- x, 648
- GiNaC::integration\_kernel, 649
  - cache\_step\_size, 654
  - do\_print, 654
  - get\_cache\_size, 653
  - get\_numerical\_value, 652
  - get\_numerical\_value\_impl, 654
  - get\_series\_coeff, 653
  - has\_trailing\_zero, 651
  - is\_numeric, 652
  - Laurent\_series, 652
  - series, 651
  - series\_coeff, 653
  - series\_coeff\_impl, 653
  - series\_vec, 654
  - set\_cache\_step, 653
  - uses\_Laurent\_series, 652
- GiNaC::internal, 308
- GiNaC::internal::\_iter\_rep, 311
  - \_iter\_rep, 311
  - e, 312
  - i, 312
  - i\_end, 312
  - operator!=, 312
  - operator==, 311
- GiNaC::is\_not\_a\_clifford, 655
  - operator(), 655
- GiNaC::is\_summation\_idx, 655
  - operator(), 655
- GiNaC::iterated\_integral2\_SERIAL, 656
  - serial, 656
- GiNaC::iterated\_integral3\_SERIAL, 657
  - serial, 657
- GiNaC::lanczos\_coeffs, 665
  - calc\_lanczos\_A, 666
  - coeffs, 667
  - current\_vector, 667
  - get\_order, 666
  - lanczos\_coeffs, 666
  - sufficiently\_accurate, 666
- GiNaC::library\_init, 668
  - ~library\_init, 669
  - count, 670
  - init\_unarchivers, 669
  - library\_init, 669
- GiNaC::make\_flat\_inserter, 670
  - combine\_indices, 671
  - do\_renaming, 672
  - handle\_factor, 671
  - make\_flat\_inserter, 671
  - used\_indices, 672
- GiNaC::map\_function, 672
  - ~map\_function, 674
  - argument\_type, 673
  - operator(), 674
  - result\_type, 673
- GiNaC::matrix, 674
  - add, 683
  - add\_indexed, 680
  - archive, 681
  - charpoly, 687
  - col, 694
  - cols, 683
  - conjugate, 680
  - contract\_with, 680
  - determinant, 686
  - determinant\_minor, 690
  - division\_free\_elimination, 691
  - do\_print, 693
  - do\_print\_latex, 693
  - do\_print\_python\_repr, 694
  - echelon\_form, 690
  - eval\_indexed, 679
  - evalm, 679
  - fraction\_free\_elimination, 692
  - gauss\_elimination, 691
  - imag\_part, 681
  - inverse, 688
  - is\_zero\_matrix, 690
  - let\_op, 679
  - m, 694
  - markowitz\_elimination, 692
  - match\_same\_type, 682
  - matrix, 677, 678
  - mul, 684
  - mul\_scalar, 684
  - nops, 678
  - op, 678
  - operator(), 685
  - pivot, 692
  - pow, 684
  - print\_elements, 693
  - rank, 689, 690
  - read\_archive, 681
  - real\_part, 681
  - return\_type, 682
  - row, 694
  - rows, 682
  - scalar\_mul\_indexed, 680
  - set, 686
  - solve, 689
  - sub, 683
  - subs, 679
  - trace, 687
  - transpose, 686

- GiNaC::minkmetric, 695
  - archive, 697
  - do\_print, 697
  - do\_print\_latex, 698
  - eval\_indexed, 696
  - info, 696
  - minkmetric, 696
  - pos\_sig, 698
  - read\_archive, 697
  - return\_type, 697
- GiNaC::modular\_form\_kernel, 698
  - C\_norm, 702
  - do\_print, 702
  - get\_numerical\_value, 701
  - is\_numeric, 701
  - k, 702
  - Laurent\_series, 701
  - let\_op, 700
  - modular\_form\_kernel, 699
  - nops, 700
  - op, 700
  - P, 702
  - q\_expansion\_modular\_form, 702
  - series, 700
  - uses\_Laurent\_series, 701
- GiNaC::mul, 705
  - add, 720
  - algebraic\_subs\_mul, 718
  - can\_be\_further\_expanded, 719
  - can\_make\_flat, 717
  - coeff, 710
  - combine\_ex\_with\_coeff\_to\_pair, 715
  - combine\_overall\_coeff, 717
  - combine\_pair\_with\_coeff\_to\_pair, 716
  - conjugate, 714
  - default\_overall\_coeff, 716
  - degree, 709
  - derivative, 714
  - do\_print, 718
  - do\_print\_csrc, 719
  - do\_print\_latex, 719
  - do\_print\_python\_repr, 719
  - eval, 710
  - eval\_ncmul, 714
  - evalf, 711
  - evalm, 711
  - expair\_needs\_further\_processing, 716
  - expand, 717
  - expandchildren, 719
  - find\_real\_imag, 718
  - get\_free\_indices, 713
  - has, 710
  - imag\_part, 711
  - info, 709
  - integer\_content, 712
  - is\_polynomial, 709
  - ldegree, 709
  - max\_coefficient, 713
  - mul, 707, 708
  - ncmul, 720
  - normal, 712
  - power, 720
  - precedence, 708
  - print\_overall\_coeff, 718
  - real\_part, 711
  - recombine\_pair\_to\_ex, 716
  - return\_type, 714
  - return\_type\_tinfo, 715
  - series, 712
  - smod, 713
  - split\_ex\_to\_pair, 715
  - thisexpairseq, 715
- GiNaC::multi\_iterator\_counter
  - init, 722
  - multi\_iterator\_counter, 722
  - operator<<, 723
  - operator++, 722
- GiNaC::multi\_iterator\_counter< T >, 721
- GiNaC::multi\_iterator\_counter\_indv
  - init, 725
  - multi\_iterator\_counter\_indv, 724, 725
  - Nv, 726
  - operator<<, 726
  - operator++, 725
- GiNaC::multi\_iterator\_counter\_indv< T >, 723
- GiNaC::multi\_iterator\_ordered
  - init, 728
  - multi\_iterator\_ordered, 727, 728
  - operator<<, 729
  - operator++, 728
- GiNaC::multi\_iterator\_ordered< T >, 726
- GiNaC::multi\_iterator\_ordered\_eq
  - init, 731
  - multi\_iterator\_ordered\_eq, 730
  - operator<<, 731
  - operator++, 731
- GiNaC::multi\_iterator\_ordered\_eq< T >, 729
- GiNaC::multi\_iterator\_ordered\_eq\_indv
  - init, 734
  - multi\_iterator\_ordered\_eq\_indv, 733
  - Nv, 734
  - operator<<, 734
  - operator++, 734
- GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 732
- GiNaC::multi\_iterator\_permutation
  - get\_sign, 737
  - init, 737
  - multi\_iterator\_permutation, 736
  - operator<<, 738
  - operator++, 737
- GiNaC::multi\_iterator\_permutation< T >, 735
- GiNaC::multi\_iterator\_shuffle
  - init, 740
  - multi\_iterator\_shuffle, 739
  - N\_internal, 741
  - operator<<, 740

- operator++, 740
- v\_internal, 741
- v\_orig, 741
- GiNaC::multi\_iterator\_shuffle< T >, 738
- GiNaC::multi\_iterator\_shuffle\_prime
  - init, 743
  - multi\_iterator\_shuffle\_prime, 742, 743
  - operator<<, 743
- GiNaC::multi\_iterator\_shuffle\_prime< T >, 742
- GiNaC::multiple\_polylog\_kernel, 744
  - do\_print, 746
  - is\_numeric, 745
  - let\_op, 745
  - multiple\_polylog\_kernel, 745
  - nops, 745
  - op, 745
  - series\_coeff\_impl, 746
  - z, 746
- GiNaC::ncmul, 747
  - append\_factors, 754
  - coeff, 751
  - conjugate, 752
  - count\_factors, 754
  - degree, 750
  - derivative, 753
  - do\_print, 754
  - do\_print\_csrc, 754
  - eval, 751
  - evalm, 751
  - expand, 750
  - expandchildren, 754
  - get\_factors, 755
  - get\_free\_indices, 752
  - hold\_ncmul, 755
  - imag\_part, 753
  - info, 750
  - ldegree, 750
  - ncmul, 748, 749
  - power, 755
  - precedence, 749
  - real\_part, 752
  - reeval\_ncmul, 755
  - return\_type, 753
  - return\_type\_tinfo, 753
  - thiscontainer, 752
- GiNaC::normal\_map\_function, 756
  - operator(), 756
- GiNaC::numeric, 757
  - add, 770
  - add\_dyn, 772
  - archive, 769
  - calchash, 770
  - coeff, 764
  - compare, 775
  - conjugate, 768
  - csgn, 774
  - degree, 763
  - denom, 783
  - derivative, 769
  - div, 771
  - div\_dyn, 772
  - do\_print, 784
  - do\_print\_csrc, 784
  - do\_print\_csrc\_cl\_N, 784
  - do\_print\_latex, 784
  - do\_print\_python\_repr, 785
  - do\_print\_tree, 785
  - eval, 764
  - evalf, 765
  - has, 764
  - imag, 782
  - imag\_part, 768
  - info, 763
  - int\_length, 783
  - integer\_content, 766
  - inverse, 774
  - is\_cinteger, 778
  - is\_crational, 778
  - is\_equal, 775
  - is\_equal\_same\_type, 769
  - is\_even, 777
  - is\_integer, 776
  - is\_negative, 776
  - is\_nonneg\_integer, 777
  - is\_odd, 777
  - is\_polynomial, 763
  - is\_pos\_integer, 776
  - is\_positive, 776
  - is\_prime, 777
  - is\_rational, 778
  - is\_real, 778
  - is\_zero, 775
  - ldegree, 764
  - max\_coefficient, 768
  - mul, 771
  - mul\_dyn, 772
  - normal, 765
  - numer, 783
  - numeric, 760–762
  - operator!=, 779
  - operator<, 779
  - operator<=, 779
  - operator>, 781
  - operator>=, 781
  - operator=, 773, 774
  - operator==, 779
  - power, 771
  - power\_dyn, 773
  - precedence, 762
  - print\_numeric, 783
  - read\_archive, 769
  - real, 782
  - real\_part, 768
  - smod, 767
  - step, 774
  - sub, 770

- sub\_dyn, 772
- subs, 765
- to\_cl\_N, 782
- to\_double, 782
- to\_int, 781
- to\_long, 781
- to\_polynomial, 766
- to\_rational, 766
- value, 785
- GiNaC::op0\_is\_equal, 786
- operator(), 786
- GiNaC::partition\_generator, 786
  - current\_updated, 788
  - get, 787
  - next, 787
  - partition, 787
  - partition\_generator, 787
- GiNaC::partition\_with\_zero\_parts\_generator, 788
  - current\_updated, 790
  - get, 789
  - m, 789
  - next, 789
  - partition, 790
  - partition\_with\_zero\_parts\_generator, 789
- GiNaC::pointer\_to\_map\_function, 790
  - operator(), 791
  - pointer\_to\_map\_function, 791
  - ptr, 791
- GiNaC::pointer\_to\_map\_function\_1arg
  - arg1, 793
  - operator(), 792
  - pointer\_to\_map\_function\_1arg, 792
  - ptr, 792
- GiNaC::pointer\_to\_map\_function\_1arg< T1 >, 791
- GiNaC::pointer\_to\_map\_function\_2args
  - arg1, 794
  - arg2, 794
  - operator(), 794
  - pointer\_to\_map\_function\_2args, 793
  - ptr, 794
- GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, 793
- GiNaC::pointer\_to\_map\_function\_3args
  - arg1, 796
  - arg2, 796
  - arg3, 796
  - operator(), 796
  - pointer\_to\_map\_function\_3args, 795
  - ptr, 796
- GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, 795
- GiNaC::pointer\_to\_member\_to\_map\_function
  - c, 798
  - operator(), 798
  - pointer\_to\_member\_to\_map\_function, 797
  - ptr, 798
- GiNaC::pointer\_to\_member\_to\_map\_function< C >, 797
- GiNaC::pointer\_to\_member\_to\_map\_function\_1arg
  - arg1, 800
  - c, 800
  - operator(), 799
  - pointer\_to\_member\_to\_map\_function\_1arg, 799
  - ptr, 799
- GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, 798
- GiNaC::pointer\_to\_member\_to\_map\_function\_2args
  - arg1, 802
  - arg2, 802
  - c, 802
  - operator(), 801
  - pointer\_to\_member\_to\_map\_function\_2args, 801
  - ptr, 801
- GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, 800
- GiNaC::pointer\_to\_member\_to\_map\_function\_3args
  - arg1, 804
  - arg2, 804
  - arg3, 804
  - c, 804
  - operator(), 803
  - pointer\_to\_member\_to\_map\_function\_3args, 803
  - ptr, 804
- GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, 802
- GiNaC::pole\_error, 805
  - deg, 806
  - degree, 805
  - pole\_error, 805
- GiNaC::possymbol, 806
  - duplicate, 807
  - get\_domain, 807
  - possymbol, 807
- GiNaC::power, 808
  - archive, 817
  - basis, 822
  - coeff, 812
  - conjugate, 816
  - degree, 812
  - derivative, 817
  - do\_print\_csrc, 819
  - do\_print\_csrc\_cl\_N, 820
  - do\_print\_dflt, 819
  - do\_print\_latex, 819
  - do\_print\_python, 819
  - do\_print\_python\_repr, 820
  - eval, 813
  - eval\_ncmul, 817
  - evalf, 813
  - evalm, 814
  - expand, 818
  - expand\_add, 820
  - expand\_add\_2, 820
  - expand\_mul, 821
  - exponent, 822
  - has, 815



- imag\_part, 816
- info, 811
- is\_polynomial, 812
- ldegree, 812
- map, 811
- mul, 821
- nops, 811
- normal, 815
- op, 811
- power, 810
- precedence, 810
- print\_power, 818
- read\_archive, 817
- real\_part, 816
- return\_type, 818
- return\_type\_tinfo, 818
- series, 814
- subs, 814
- to\_polynomial, 816
- to\_rational, 815
- GiNaC::print\_context, 822
  - ~print\_context, 823
  - options, 823
  - print\_context, 823
  - s, 823
- GiNaC::print\_context\_options, 824
  - get\_id, 825
  - get\_name, 825
  - get\_parent\_name, 825
  - id, 826
  - name, 825
  - parent\_name, 825
  - print\_context\_options, 824
- GiNaC::print\_csrc, 826
  - print\_csrc, 827
- GiNaC::print\_csrc\_cl\_N, 827
  - print\_csrc\_cl\_N, 827
- GiNaC::print\_csrc\_double, 828
  - print\_csrc\_double, 828
- GiNaC::print\_csrc\_float, 829
  - print\_csrc\_float, 829
- GiNaC::print\_dflt, 830
  - print\_dflt, 830
- GiNaC::print\_functor, 831
  - impl, 833
  - is\_valid, 832
  - operator(), 832
  - operator=, 832
  - print\_functor, 831, 832
- GiNaC::print\_functor\_impl, 833
  - ~print\_functor\_impl, 834
  - duplicate, 834
  - operator(), 834
- GiNaC::print\_latex, 834
  - print\_latex, 835
- GiNaC::print\_memfun\_handler
  - duplicate, 836
  - F, 836
  - f, 837
  - operator(), 837
  - print\_memfun\_handler, 836
- GiNaC::print\_memfun\_handler< T, C >, 835
- GiNaC::print\_options, 837
- GiNaC::print\_ptrfun\_handler
  - duplicate, 839
  - F, 839
  - f, 840
  - operator(), 839
  - print\_ptrfun\_handler, 839
- GiNaC::print\_ptrfun\_handler< T, C >, 838
- GiNaC::print\_python, 840
  - print\_python, 841
- GiNaC::print\_python\_repr, 841
  - print\_python\_repr, 842
- GiNaC::print\_tree, 842
  - delta\_indent, 843
  - print\_tree, 842, 843
- GiNaC::pseries, 847
  - add\_series, 857
  - archive, 854
  - coeff, 851
  - coeffop, 857
  - collect, 851
  - conjugate, 853
  - convert\_to\_poly, 856
  - degree, 850
  - derivative, 855
  - do\_print, 859
  - do\_print\_latex, 860
  - do\_print\_python, 860
  - do\_print\_python\_repr, 860
  - do\_print\_tree, 860
  - eval, 852
  - eval\_integ, 854
  - evalf, 852
  - evalm, 854
  - expand, 853
  - exponop, 857
  - get\_point, 856
  - get\_var, 855
  - imag\_part, 854
  - is\_compatible\_to, 856
  - is\_terminating, 857
  - is\_zero, 856
  - ldegree, 851
  - mul\_const, 858
  - mul\_series, 858
  - nops, 850
  - normal, 853
  - op, 850
  - point, 861
  - power\_const, 859
  - precedence, 850
  - print\_series, 859
  - pseries, 849
  - read\_archive, 855

- real\_part, 854
- seq, 861
- series, 852
- shift\_exponents, 859
- subs, 852
- var, 861
- GiNaC::psi1\_SERIAL, 861
  - serial, 862
- GiNaC::psi2\_SERIAL, 862
  - serial, 863
- GiNaC::ptr
  - ~ptr, 865
  - get\_pointer, 866
  - makewritable, 865
  - operator!=, 866, 867
  - operator<<, 867
  - operator\*, 865
  - operator->, 865
  - operator=, 865
  - operator==, 866, 867
  - p, 867
  - ptr, 864
  - std::less< ptr< T > >, 866
  - swap, 865
- GiNaC::ptr< T >, 863
- GiNaC::realsymbol, 868
  - conjugate, 869
  - duplicate, 870
  - get\_domain, 869
  - imag\_part, 870
  - real\_part, 869
  - realsymbol, 869
- GiNaC::refcounted, 871
  - add\_reference, 872
  - get\_refcount, 872
  - refcount, 873
  - refcounted, 872
  - remove\_reference, 872
  - set\_refcount, 872
- GiNaC::registered\_class\_options, 873
  - get\_id, 874
  - get\_name, 874
  - get\_parent\_name, 874
  - get\_print\_dispatch\_table, 874
  - name, 876
  - parent\_name, 876
  - print\_dispatch\_table, 876
  - print\_func, 875
  - registered\_class\_options, 874
  - set\_print\_func, 875
  - tinfo\_key, 876
- GiNaC::relational, 877
  - archive, 880
  - calchash, 882
  - do\_print, 882
  - do\_print\_python\_repr, 882
  - eval\_ncmul, 881
  - info, 879
  - lh, 884
  - lhs, 883
  - make\_safe\_bool, 883
  - map, 880
  - match\_same\_type, 881
  - nops, 879
  - o, 884
  - op, 880
  - operator safe\_bool, 883
  - operator!, 883
  - operators, 878
  - precedence, 879
  - read\_archive, 881
  - relational, 879
  - return\_type, 881
  - return\_type\_tinfo, 882
  - rh, 884
  - rhs, 883
  - safe\_bool, 878
  - subs, 880
- GiNaC::relational::safe\_bool\_helper, 896
  - nonnull, 896
- GiNaC::remember\_strategies, 884
- GiNaC::remember\_table, 885
  - add\_entry, 887
  - clear\_all\_entries, 887
  - init\_table, 888
  - lookup\_entry, 887
  - max\_assoc\_size, 888
  - remember\_strategy, 888
  - remember\_table, 886
  - remember\_tables, 887
  - show\_statistics, 887
  - table\_size, 888
- GiNaC::remember\_table\_entry, 889
  - access\_counter, 891
  - get\_last\_access, 890
  - get\_result, 890
  - get\_successful\_hits, 890
  - hashvalue, 890
  - is\_equal, 890
  - last\_access, 891
  - remember\_table\_entry, 889
  - result, 891
  - seq, 890
  - successful\_hits, 891
- GiNaC::remember\_table\_list, 892
  - add\_entry, 892
  - lookup\_entry, 893
  - max\_assoc\_size, 893
  - remember\_strategy, 893
  - remember\_table\_list, 892
- GiNaC::return\_type\_t, 893
  - operator!=, 894
  - operator<, 894
  - operator==, 894
  - rl, 895
  - tinfo, 895

- GiNaC::return\_types, 895
- GiNaC::scalar\_products, 896
  - add, 897
  - add\_vectors, 898
  - clear, 898
  - debugprint, 899
  - evaluate, 898
  - is\_defined, 898
  - spm, 899
- GiNaC::series\_options, 899
- GiNaC::solve\_algo, 900
- GiNaC::spinidx, 901
  - archive, 903
  - conjugate, 903
  - do\_print, 905
  - do\_print\_latex, 905
  - do\_print\_tree, 905
  - dotted, 906
  - is\_dotted, 904
  - is\_dummy\_pair\_same\_type, 903
  - is\_undotted, 904
  - match\_same\_type, 904
  - read\_archive, 903
  - spinidx, 902
  - toggle\_dot, 905
  - toggle\_variance\_dot, 905
- GiNaC::spinmetric, 906
  - contract\_with, 908
  - do\_print, 908
  - do\_print\_latex, 908
  - eval\_indexed, 907
  - info, 907
- GiNaC::spmapkey, 909
  - debugprint, 910
  - dim, 910
  - operator<, 910
  - operator==, 909
  - spmapkey, 909
  - v1, 910
  - v2, 910
- GiNaC::status\_flags, 911
- GiNaC::structure
  - add\_indexed, 924
  - calchash, 926
  - coeff, 920
  - collect, 920
  - contract\_with, 925
  - degree, 919
  - derivative, 921
  - eval, 914
  - eval\_indexed, 915
  - eval\_ncmul, 915
  - evalm, 915
  - expand, 920
  - get\_class\_name, 914
  - get\_free\_indices, 923
  - get\_struct, 927
  - has, 918
  - info, 916
  - integer\_content, 922
  - is\_equal\_same\_type, 926
  - ldegree, 920
  - let\_op, 917
  - map, 919
  - match, 918
  - match\_same\_type, 918
  - max\_coefficient, 923
  - nops, 916
  - normal, 921
  - obj, 927
  - op, 916
  - operator->, 926
  - operator[], 917
  - precedence, 916
  - print, 915
  - return\_type, 925
  - return\_type\_tinfo, 926
  - scalar\_mul\_indexed, 924
  - series, 921
  - smod, 923
  - structure, 914
  - subs, 919
  - to\_polynomial, 922
  - to\_rational, 922
- GiNaC::structure< T, ComparisonPolicy >, 912
- GiNaC::su3d, 928
  - contract\_with, 929
  - do\_print, 929
  - do\_print\_latex, 929
  - eval\_indexed, 928
  - return\_type, 929
- GiNaC::su3f, 930
  - contract\_with, 931
  - do\_print, 931
  - do\_print\_latex, 931
  - eval\_indexed, 931
  - return\_type, 931
- GiNaC::su3one, 932
  - do\_print, 932
  - do\_print\_latex, 933
- GiNaC::su3t, 933
  - contract\_with, 934
  - do\_print, 934
  - do\_print\_latex, 934
- GiNaC::subs\_options, 934
- GiNaC::sy\_is\_less, 935
  - operator(), 936
  - sy\_is\_less, 936
  - v, 936
- GiNaC::sy\_swap, 936
  - operator(), 937
  - swapped, 937
  - sy\_swap, 937
  - v, 937
- GiNaC::sym\_desc, 938
  - deg\_a, 939

- deg\_b, 939
- lddeg\_a, 940
- lddeg\_b, 940
- max\_deg, 940
- max\_lcnops, 940
- operator<, 939
- sym, 939
- sym\_desc, 939
- GiNaC::symbol, 941
  - archive, 946
  - calchash, 947
  - conjugate, 945
  - derivative, 947
  - do\_print, 949
  - do\_print\_latex, 949
  - do\_print\_python\_repr, 949
  - do\_print\_tree, 949
  - eval, 943
  - evalf, 943
  - get\_TeX\_name, 949
  - get\_domain, 948
  - get\_name, 948
  - imag\_part, 946
  - info, 943
  - is\_equal\_same\_type, 947
  - is\_polynomial, 946
  - name, 950
  - next\_serial, 950
  - normal, 944
  - read\_archive, 946
  - real\_part, 945
  - serial, 950
  - series, 944
  - set\_TeX\_name, 948
  - set\_name, 948
  - subs, 944
  - symbol, 942
  - TeX\_name, 950
  - to\_polynomial, 945
  - to\_rational, 945
- GiNaC::symbolset, 951
  - has, 952
  - insert\_symbols, 951
  - s, 952
  - symbolset, 951
- GiNaC::symmetry, 952
  - add, 956
  - archive, 955
  - calchash, 955
  - canonicalize, 958
  - children, 959
  - do\_print, 957
  - do\_print\_tree, 957
  - get\_type, 956
  - has\_cyclic, 957
  - has\_nonsymmetric, 957
  - has\_symmetry, 957
  - indices, 959
  - read\_archive, 955
  - set\_type, 956
  - sy\_is\_less, 958
  - sy\_swap, 958
  - symmetry, 954
  - symmetry\_type, 954
  - type, 958
  - validate, 956
- GiNaC::symminfo, 959
  - coeff, 960
  - num, 961
  - orig, 961
  - symminfo, 960
  - symmterm, 960
- GiNaC::symminfo\_is\_less\_by\_orig, 961
  - operator(), 961
- GiNaC::symminfo\_is\_less\_by\_symmterm, 962
  - operator(), 962
- GiNaC::tensdelta, 962
  - contract\_with, 963
  - do\_print, 964
  - do\_print\_latex, 964
  - eval\_indexed, 963
  - info, 963
  - return\_type, 964
- GiNaC::tensepsilon, 965
  - archive, 967
  - contract\_with, 966
  - do\_print, 967
  - do\_print\_latex, 968
  - eval\_indexed, 966
  - info, 966
  - minkowski, 968
  - pos\_sig, 968
  - read\_archive, 967
  - return\_type, 967
  - tensepsilon, 966
- GiNaC::tensmetric, 969
  - contract\_with, 970
  - do\_print, 971
  - eval\_indexed, 970
  - info, 969
  - return\_type, 970
- GiNaC::tensor, 971
  - replace\_contr\_index, 972
  - return\_type, 972
- GiNaC::terminfo, 973
  - orig, 973
  - symm, 973
  - terminfo, 973
- GiNaC::terminfo\_is\_less, 974
  - operator(), 974
- GiNaC::unarchive\_table\_t, 976
  - ~unarchive\_table\_t, 976
  - find, 976
  - insert, 976
  - unarch\_map, 977
  - unarchive\_table\_t, 976

- usecount, 977
- GiNaC::user\_defined\_kernel, 977
  - do\_print, 980
  - f, 980
  - is\_numeric, 979
  - Laurent\_series, 979
  - let\_op, 979
  - nops, 978
  - op, 979
  - user\_defined\_kernel, 978
  - uses\_Laurent\_series, 980
  - x, 980
- GiNaC::varidx, 981
  - archive, 983
  - covariant, 985
  - do\_print, 984
  - do\_print\_tree, 985
  - is\_contravariant, 984
  - is\_covariant, 984
  - is\_dummy\_pair\_same\_type, 982
  - match\_same\_type, 983
  - read\_archive, 983
  - toggle\_variance, 984
  - varidx, 982
- GiNaC::visitor, 985
  - ~visitor, 986
- GiNaC::wildcard, 986
  - archive, 987
  - calchash, 988
  - do\_print, 988
  - do\_print\_python\_repr, 989
  - do\_print\_tree, 989
  - get\_label, 988
  - label, 989
  - match, 987
  - read\_archive, 988
  - wildcard, 987
- GiNaC::zeta1\_SERIAL, 990
  - serial, 990
- GiNaC::zeta2\_SERIAL, 990
  - serial, 991
- GiNaC, 19
  - \_ex0, 299
  - \_ex1, 301
  - \_ex10, 304
  - \_ex11, 305
  - \_ex12, 305
  - \_ex120, 308
  - \_ex15, 305
  - \_ex18, 306
  - \_ex1\_2, 300
  - \_ex1\_3, 300
  - \_ex1\_4, 300
  - \_ex2, 301
  - \_ex20, 306
  - \_ex24, 306
  - \_ex25, 307
  - \_ex3, 302
  - \_ex30, 307
  - \_ex4, 302
  - \_ex48, 307
  - \_ex5, 303
  - \_ex6, 303
  - \_ex60, 308
  - \_ex7, 303
  - \_ex8, 304
  - \_ex9, 304
  - \_ex\_1, 298
  - \_ex\_10, 295
  - \_ex\_11, 294
  - \_ex\_12, 294
  - \_ex\_120, 291
  - \_ex\_15, 294
  - \_ex\_18, 293
  - \_ex\_1\_2, 298
  - \_ex\_1\_3, 298
  - \_ex\_1\_4, 299
  - \_ex\_2, 297
  - \_ex\_20, 293
  - \_ex\_24, 293
  - \_ex\_25, 292
  - \_ex\_3, 297
  - \_ex\_30, 292
  - \_ex\_4, 297
  - \_ex\_48, 292
  - \_ex\_5, 296
  - \_ex\_6, 296
  - \_ex\_60, 291
  - \_ex\_7, 296
  - \_ex\_8, 295
  - \_ex\_9, 295
  - \_num0\_bp, 289
  - \_num0\_p, 299
  - \_num10\_p, 304
  - \_num11\_p, 304
  - \_num120\_p, 308
  - \_num12\_p, 305
  - \_num15\_p, 305
  - \_num18\_p, 305
  - \_num1\_2\_p, 300
  - \_num1\_3\_p, 300
  - \_num1\_4\_p, 299
  - \_num1\_p, 300
  - \_num20\_p, 306
  - \_num24\_p, 306
  - \_num25\_p, 306
  - \_num2\_p, 301
  - \_num30\_p, 307
  - \_num3\_p, 302
  - \_num48\_p, 307
  - \_num4\_p, 302
  - \_num5\_p, 302
  - \_num60\_p, 307
  - \_num6\_p, 303
  - \_num7\_p, 303
  - \_num8\_p, 303

- [\\_num9\\_p](#), 304
- [\\_num\\_10\\_p](#), 294
- [\\_num\\_11\\_p](#), 294
- [\\_num\\_120\\_p](#), 291
- [\\_num\\_12\\_p](#), 294
- [\\_num\\_15\\_p](#), 293
- [\\_num\\_18\\_p](#), 293
- [\\_num\\_1\\_2\\_p](#), 298
- [\\_num\\_1\\_3\\_p](#), 298
- [\\_num\\_1\\_4\\_p](#), 299
- [\\_num\\_1\\_p](#), 297
- [\\_num\\_20\\_p](#), 293
- [\\_num\\_24\\_p](#), 292
- [\\_num\\_25\\_p](#), 292
- [\\_num\\_2\\_p](#), 297
- [\\_num\\_30\\_p](#), 292
- [\\_num\\_3\\_p](#), 297
- [\\_num\\_48\\_p](#), 291
- [\\_num\\_4\\_p](#), 296
- [\\_num\\_5\\_p](#), 296
- [\\_num\\_60\\_p](#), 291
- [\\_num\\_6\\_p](#), 296
- [\\_num\\_7\\_p](#), 295
- [\\_num\\_8\\_p](#), 295
- [\\_num\\_9\\_p](#), 295
- [abs](#), 240
- [abs\\_conjugate](#), 143
- [abs\\_eval](#), 142
- [abs\\_evalf](#), 142
- [abs\\_expand](#), 143
- [abs\\_expl\\_derivative](#), 143
- [abs\\_imag\\_part](#), 144
- [abs\\_info](#), 144
- [abs\\_power](#), 144
- [abs\\_print\\_csrc\\_float](#), 143
- [abs\\_print\\_latex](#), 143
- [abs\\_real\\_part](#), 144
- [acos](#), 231
- [acos\\_conjugate](#), 184
- [acos\\_deriv](#), 184
- [acos\\_eval](#), 184
- [acos\\_evalf](#), 184
- [acosh](#), 234
- [acosh\\_conjugate](#), 193
- [acosh\\_deriv](#), 193
- [acosh\\_eval](#), 192
- [acosh\\_evalf](#), 192
- [adaptivesimpson](#), 195
- [add\\_symbol](#), 211
- [algebraic\\_match\\_mul\\_with\\_mul](#), 209
- [antisymmetric2](#), 270
- [antisymmetric3](#), 270
- [antisymmetric4](#), 270
- [antisymmetrize](#), 119, 271, 275
- [archive\\_atom](#), 58
- [archive\\_node\\_id](#), 58
- [are\\_ex\\_trivially\\_equal](#), 111
- [asin](#), 231
- [asin\\_conjugate](#), 183
- [asin\\_deriv](#), 183
- [asin\\_eval](#), 183
- [asin\\_evalf](#), 183
- [asinh](#), 234
- [asinh\\_conjugate](#), 192
- [asinh\\_deriv](#), 192
- [asinh\\_eval](#), 191
- [asinh\\_evalf](#), 191
- [atan](#), 232
- [atan2\\_deriv](#), 186
- [atan2\\_eval](#), 186
- [atan2\\_evalf](#), 186
- [atan\\_conjugate](#), 186
- [atan\\_deriv](#), 185
- [atan\\_eval](#), 185
- [atan\\_evalf](#), 185
- [atan\\_series](#), 185
- [atanh](#), 234
- [atanh\\_conjugate](#), 194
- [atanh\\_deriv](#), 194
- [atanh\\_eval](#), 193
- [atanh\\_evalf](#), 193
- [atanh\\_series](#), 194
- [base\\_and\\_index](#), 94
- [bernoulli](#), 239
- [Bernoulli\\_polynomial](#), 197
- [beta\\_deriv](#), 166
- [beta\\_eval](#), 166
- [beta\\_evalf](#), 165
- [beta\\_series](#), 166
- [binomial](#), 239
- [binomial\\_conjugate](#), 153
- [binomial\\_eval](#), 153
- [binomial\\_evalf](#), 152
- [binomial\\_imag\\_part](#), 153
- [binomial\\_real\\_part](#), 153
- [binomial\\_sym](#), 153
- [canonicalize](#), 270
- [canonicalize\\_clifford](#), 99
- [Catalan](#), 288
- [CatalanEvalf](#), 245
- [charpoly](#), 207
- [clifford\\_bar](#), 104
- [clifford\\_inverse](#), 100
- [clifford\\_max\\_label](#), 100
- [clifford\\_moebius\\_map](#), 102
- [clifford\\_norm](#), 100
- [clifford\\_prime](#), 99
- [clifford\\_star](#), 104
- [clifford\\_star\\_bar](#), 99
- [clifford\\_to\\_lst](#), 101
- [clifford\\_unit](#), 95
- [coeff](#), 115
- [coerce](#), 228
- [coerce< int, cln::cl\\_I >](#), 228
- [coerce< unsigned int, cln::cl\\_I >](#), 229
- [collect](#), 117

collect\_common\_factors, 225  
collect\_symbols, 211  
color\_ONE, 106  
color\_d, 107  
color\_f, 107  
color\_h, 108  
color\_T, 107  
color\_trace, 109  
cols, 206  
compare\_pointers, 282  
compile\_ex, 123, 124  
conjugate, 113  
conjugate\_conjugate, 138  
conjugate\_eval, 138  
conjugate\_evalf, 138  
conjugate\_expl\_derivative, 138  
conjugate\_funcp, 61  
conjugate\_funcp\_1, 62  
conjugate\_funcp\_10, 77  
conjugate\_funcp\_11, 79  
conjugate\_funcp\_12, 80  
conjugate\_funcp\_13, 82  
conjugate\_funcp\_14, 84  
conjugate\_funcp\_2, 64  
conjugate\_funcp\_3, 65  
conjugate\_funcp\_4, 67  
conjugate\_funcp\_5, 68  
conjugate\_funcp\_6, 70  
conjugate\_funcp\_7, 72  
conjugate\_funcp\_8, 73  
conjugate\_funcp\_9, 75  
conjugate\_funcp\_exvector, 86  
conjugate\_imag\_part, 139  
conjugate\_info, 139  
conjugate\_print\_latex, 138  
conjugate\_real\_part, 138  
conjugateepvector, 126  
convert\_H\_to\_Li, 159  
cos, 230  
cos\_conjugate, 181  
cos\_deriv, 180  
cos\_eval, 180  
cos\_evalf, 180  
cos\_imag\_part, 181  
cos\_real\_part, 180  
cosh, 233  
cosh\_conjugate, 189  
cosh\_deriv, 189  
cosh\_eval, 188  
cosh\_evalf, 188  
cosh\_imag\_part, 189  
cosh\_real\_part, 189  
count\_dummy\_indices, 132  
count\_free\_indices, 132  
crc32, 111  
crctab, 289  
csgn, 246  
csgn\_conjugate, 147  
csgn\_eval, 146  
csgn\_evalf, 146  
csgn\_imag\_part, 147  
csgn\_power, 147  
csgn\_real\_part, 147  
csgn\_series, 146  
csrc, 261  
csrc\_cl\_N, 262  
csrc\_double, 261  
csrc\_float, 261  
cyclic\_permutation, 283  
decomp\_rational, 214  
degree, 114  
delta\_tensor, 277  
denom, 115, 250  
derivative\_funcp, 61  
derivative\_funcp\_1, 63  
derivative\_funcp\_10, 77  
derivative\_funcp\_11, 79  
derivative\_funcp\_12, 81  
derivative\_funcp\_13, 83  
derivative\_funcp\_14, 85  
derivative\_funcp\_2, 64  
derivative\_funcp\_3, 66  
derivative\_funcp\_4, 67  
derivative\_funcp\_5, 69  
derivative\_funcp\_6, 71  
derivative\_funcp\_7, 72  
derivative\_funcp\_8, 74  
derivative\_funcp\_9, 76  
derivative\_funcp\_exvector, 87  
determinant, 207  
dflt, 260  
diag\_matrix, 204  
diff, 118  
Digits, 290  
digits\_changed\_callback, 89  
dirac\_ONE, 94  
dirac\_gamma, 95  
dirac\_gamma5, 96  
dirac\_gammaL, 96  
dirac\_gammaR, 96  
dirac\_slash, 97  
dirac\_trace, 97, 98  
dirichlet\_character, 197  
divide, 215  
divide\_in\_z, 216  
doublefactorial, 238  
dynallocate, 92  
EllipticE\_deriv, 161  
EllipticE\_eval, 160  
EllipticE\_evalf, 160  
EllipticE\_print\_latex, 161  
EllipticE\_series, 161  
EllipticK\_deriv, 159  
EllipticK\_eval, 159  
EllipticK\_evalf, 159  
EllipticK\_print\_latex, 160

EllipticK\_series, 160  
 epp, 60  
 epsilon\_tensor, 279  
 epvector, 60  
 eta\_conjugate, 148  
 eta\_eval, 148  
 eta\_evalf, 148  
 eta\_imag\_part, 149  
 eta\_real\_part, 148  
 eta\_series, 148  
 Euler, 288  
 EulerEvalf, 245  
 eval, 117  
 eval\_funcp, 60  
 eval\_funcp\_1, 62  
 eval\_funcp\_10, 77  
 eval\_funcp\_11, 78  
 eval\_funcp\_12, 80  
 eval\_funcp\_13, 82  
 eval\_funcp\_14, 84  
 eval\_funcp\_2, 63  
 eval\_funcp\_3, 65  
 eval\_funcp\_4, 66  
 eval\_funcp\_5, 68  
 eval\_funcp\_6, 70  
 eval\_funcp\_7, 71  
 eval\_funcp\_8, 73  
 eval\_funcp\_9, 75  
 eval\_funcp\_exvector, 86  
 eval\_integ, 117  
 evalf, 117, 206  
 evalf\_funcp, 60  
 evalf\_funcp\_1, 62  
 evalf\_funcp\_10, 77  
 evalf\_funcp\_11, 78  
 evalf\_funcp\_12, 80  
 evalf\_funcp\_13, 82  
 evalf\_funcp\_14, 84  
 evalf\_funcp\_2, 63  
 evalf\_funcp\_3, 65  
 evalf\_funcp\_4, 67  
 evalf\_funcp\_5, 68  
 evalf\_funcp\_6, 70  
 evalf\_funcp\_7, 72  
 evalf\_funcp\_8, 73  
 evalf\_funcp\_9, 75  
 evalf\_funcp\_exvector, 86  
 evalffuncdtype, 59  
 evalm, 117  
 ex\_to, 122  
 exadd, 251  
 exhashmap, 87  
 exmap, 59  
 exminus, 251  
 exmul, 251  
 exp, 229  
 exp\_conjugate, 176  
 exp\_deriv, 175  
 exp\_eval, 175  
 exp\_evalf, 175  
 exp\_expand, 175  
 exp\_imag\_part, 176  
 exp\_power, 176  
 exp\_real\_part, 176  
 expand, 113, 206  
 expand\_dummy\_sum, 137  
 expand\_funcp, 61  
 expand\_funcp\_1, 62  
 expand\_funcp\_10, 77  
 expand\_funcp\_11, 79  
 expand\_funcp\_12, 81  
 expand\_funcp\_13, 83  
 expand\_funcp\_14, 85  
 expand\_funcp\_2, 64  
 expand\_funcp\_3, 65  
 expand\_funcp\_4, 67  
 expand\_funcp\_5, 69  
 expand\_funcp\_6, 70  
 expand\_funcp\_7, 72  
 expand\_funcp\_8, 74  
 expand\_funcp\_9, 76  
 expand\_funcp\_exvector, 87  
 expl\_derivative\_funcp, 61  
 expl\_derivative\_funcp\_1, 63  
 expl\_derivative\_funcp\_10, 78  
 expl\_derivative\_funcp\_11, 79  
 expl\_derivative\_funcp\_12, 81  
 expl\_derivative\_funcp\_13, 83  
 expl\_derivative\_funcp\_14, 85  
 expl\_derivative\_funcp\_2, 64  
 expl\_derivative\_funcp\_3, 66  
 expl\_derivative\_funcp\_4, 67  
 expl\_derivative\_funcp\_5, 69  
 expl\_derivative\_funcp\_6, 71  
 expl\_derivative\_funcp\_7, 72  
 expl\_derivative\_funcp\_8, 74  
 expl\_derivative\_funcp\_9, 76  
 expl\_derivative\_funcp\_exvector, 87  
 exprseq, 60  
 exset, 59  
 exvector, 59  
 exvectorvector, 88  
 FUNCNP\_1P, 59  
 FUNCNP\_2P, 59  
 FUNCNP\_CUBA, 60  
 factor, 126  
 factorial, 238  
 factorial\_conjugate, 152  
 factorial\_eval, 151  
 factorial\_evalf, 151  
 factorial\_imag\_part, 152  
 factorial\_print\_dflt\_latex, 152  
 factorial\_real\_part, 152  
 fibonacci, 240  
 find, 114  
 find\_common\_factor, 225



- find\_dummy\_indices, 131
- find\_factory\_fcn, 91
- find\_free\_and\_dummy, 130, 131
- find\_variant\_indices, 133
- force\_include\_tgamma, 289
- force\_include\_zeta1, 289
- format\_index\_value, 283, 284
- frac\_cancel, 224
- fsolve, 155
- func\_arg\_info, 139
- G, 157
- G2\_eval, 169
- G2\_evalf, 168
- G3\_eval, 169
- G3\_evalf, 169
- GINAC\_BIND\_UNARCHIVER, 90, 93, 94, 105, 111, 127, 129, 132, 195, 198–201, 203, 209, 210, 227, 263, 264, 266–268, 276, 277, 286
- GINAC\_DECLARE\_UNARCHIVER, 90, 103, 104, 110, 111, 127, 130, 131, 137, 195, 201–203, 205, 209, 210, 245, 263, 264, 266–268, 272, 280, 281, 287
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_←  
OPT, 89, 91, 93, 104, 111, 127, 128, 132, 194, 198–201, 203, 208, 210, 226, 262, 264, 266, 268, 276, 286
- gcd, 220, 243
- gcd\_pf\_mul, 220
- gcd\_pf\_pow, 219
- gcd\_pf\_pow\_pow, 221
- generalised\_Bernoulli\_number, 197
- get\_all\_dummy\_indices, 136
- get\_all\_dummy\_indices\_safely, 135
- get\_clifford\_comp, 101
- get\_default\_TeX\_name, 267
- get\_dim\_uint, 95
- get\_first\_symbol, 210
- get\_print\_context, 259
- get\_print\_options, 259
- get\_representation\_label, 97, 108
- get\_symbol\_stats, 211
- golden\_ratio\_hash, 282
- guess\_precision, 236
- H\_deriv, 173
- H\_eval, 172
- H\_evalf, 172
- H\_print\_latex, 173
- H\_series, 172
- has, 114
- hasindex, 135
- haswild, 287
- heur\_gcd, 219
- heur\_gcd\_z, 218
- hold\_ncmul, 210
- I, 290
- idx, 289
- idx\_symmetrization, 134
- ifactor, 196
- imag, 250
- imag\_part, 113
- imag\_part\_conjugate, 141
- imag\_part\_eval, 141
- imag\_part\_evalf, 141
- imag\_part\_expl\_derivative, 142
- imag\_part\_funcp, 61
- imag\_part\_funcp\_1, 62
- imag\_part\_funcp\_10, 77
- imag\_part\_funcp\_11, 79
- imag\_part\_funcp\_12, 81
- imag\_part\_funcp\_13, 83
- imag\_part\_funcp\_14, 85
- imag\_part\_funcp\_2, 64
- imag\_part\_funcp\_3, 65
- imag\_part\_funcp\_4, 67
- imag\_part\_funcp\_5, 69
- imag\_part\_funcp\_6, 70
- imag\_part\_funcp\_7, 72
- imag\_part\_funcp\_8, 74
- imag\_part\_funcp\_9, 75
- imag\_part\_funcp\_exvector, 86
- imag\_part\_imag\_part, 142
- imag\_part\_print\_latex, 141
- imag\_part\_real\_part, 142
- index0, 268
- index1, 268
- index2, 269
- index3, 269
- index\_dimensions, 262
- indices\_consistent, 132
- info\_funcp, 62
- info\_funcp\_1, 63
- info\_funcp\_10, 78
- info\_funcp\_11, 80
- info\_funcp\_12, 82
- info\_funcp\_13, 84
- info\_funcp\_14, 86
- info\_funcp\_2, 65
- info\_funcp\_3, 66
- info\_funcp\_4, 68
- info\_funcp\_5, 70
- info\_funcp\_6, 71
- info\_funcp\_7, 73
- info\_funcp\_8, 75
- info\_funcp\_9, 76
- info\_funcp\_exvector, 87
- interpolate, 217
- inverse, 207, 246
- iquo, 242, 243
- irem, 241, 242
- is\_a, 92, 122, 264
- is\_cinteger, 249
- is\_clifford\_tinfo, 104
- is\_color\_tinfo, 108
- is\_crational, 249
- is\_dirac\_slash, 94
- is\_discriminant\_of\_quadratic\_number\_field, 196

- is\_dummy\_pair, 129
- is\_even, 248
- is\_exactly\_a, 92, 122
- is\_integer, 247
- is\_negative, 247
- is\_nonneg\_integer, 247
- is\_odd, 248
- is\_order\_function, 158
- is\_polynomial, 114
- is\_pos\_integer, 247
- is\_positive, 247
- is\_prime, 248
- is\_rational, 248
- is\_real, 248
- is\_terminating, 265
- is\_the\_function, 128
- is\_the\_function< G\_SERIAL >, 157
- is\_the\_function< iterated\_integral\_SERIAL >, 158
- is\_the\_function< psi\_SERIAL >, 158
- is\_the\_function< zeta\_SERIAL >, 156
- is\_zero, 121, 246
- isqrt, 244
- iterated\_integral, 158
- iterated\_integral2\_eval, 162
- iterated\_integral2\_evalf, 162
- iterated\_integral3\_eval, 162
- iterated\_integral3\_evalf, 162
- iterated\_integral\_evalf\_impl, 161
- kronecker\_symbol, 196
- latex, 260
- lcm, 221, 243
- lcm\_of\_coefficients\_denominators, 212
- lcmcoeff, 212
- ldegree, 115
- lgamma, 236, 237
- lgamma\_conjugate, 164
- lgamma\_deriv, 163
- lgamma\_eval, 163
- lgamma\_evalf, 163
- lgamma\_series, 163
- lhs, 120
- Li2, 236
- Li2\_, 235
- Li2\_conjugate, 150
- Li2\_deriv, 149
- Li2\_eval, 149
- Li2\_evalf, 149
- Li2\_projection, 235
- Li2\_series, 150, 235
- Li3\_eval, 150
- Li\_deriv, 170
- Li\_eval, 170
- Li\_evalf, 169
- Li\_print\_latex, 170
- Li\_series, 170
- library\_initializer, 289
- link\_ex, 124, 125
- log, 229
- log2, 281
- log\_conjugate, 178
- log\_deriv, 177
- log\_eval, 177
- log\_evalf, 177
- log\_expand, 178
- log\_imag\_part, 178
- log\_real\_part, 177
- log\_series, 177
- lookup\_map, 88
- lorentz\_eps, 280
- lorentz\_g, 278
- lsolve, 155
- lst, 88
- lst\_to\_clifford, 100, 101
- lst\_to\_matrix, 204
- make\_hash\_seed, 128
- make\_real\_float, 226
- make\_return\_type\_t, 265
- map\_eval\_integ, 288
- map\_evalm, 287
- match, 118
- metric\_tensor, 277
- minimal\_dim, 130
- mod, 240
- multinomial\_coefficient, 281
- multiply\_lcm, 212
- my\_ios\_callback, 259
- my\_ios\_index, 258
- next\_print\_context\_id, 290
- no\_index\_dimensions, 262
- nops, 112, 205
- normal, 116
- not\_symmetric, 269
- number\_of\_type, 133
- numer, 115, 250
- numer\_denom, 116
- op, 120
- operator!=, 257
- operator<, 258
- operator<<, 90, 112, 245, 260, 284–286
- operator<=, 258
- operator>, 258
- operator>>, 91, 260
- operator>=, 258
- operator\*, 252, 253
- operator\*=, 253, 254
- operator+, 251, 252, 255
- operator++, 255–257
- operator+=, 253, 254
- operator-, 252, 255
- operator--, 255–257
- operator-=, 253, 254
- operator/, 252, 253
- operator/=: 254
- operator==, 257
- Order\_conjugate, 154
- Order\_eval, 154

- Order\_expl\_derivative, 155
- Order\_imag\_part, 155
- Order\_real\_part, 154
- Order\_series, 154
- paramset, 60
- permutation\_sign, 282, 283
- permute\_free\_index\_to\_front, 106
- Pi, 288
- PiEvalf, 244
- pow, 245, 263
- power\_funcp, 61
- power\_funcp\_1, 63
- power\_funcp\_10, 78
- power\_funcp\_11, 79
- power\_funcp\_12, 81
- power\_funcp\_13, 83
- power\_funcp\_14, 85
- power\_funcp\_2, 64
- power\_funcp\_3, 66
- power\_funcp\_4, 67
- power\_funcp\_5, 69
- power\_funcp\_6, 71
- power\_funcp\_7, 73
- power\_funcp\_8, 74
- power\_funcp\_9, 76
- power\_funcp\_exvector, 87
- prem, 214
- primitive\_dirichlet\_character, 196
- print\_context\_class\_info, 89
- print\_func< print\_context >, 128
- print\_func< print\_dfft >, 93, 105, 276
- print\_funcp, 62
- print\_funcp\_1, 63
- print\_funcp\_10, 78
- print\_funcp\_11, 80
- print\_funcp\_12, 82
- print\_funcp\_13, 84
- print\_funcp\_14, 86
- print\_funcp\_2, 65
- print\_funcp\_3, 66
- print\_funcp\_4, 68
- print\_funcp\_5, 69
- print\_funcp\_6, 71
- print\_funcp\_7, 73
- print\_funcp\_8, 75
- print\_funcp\_9, 76
- print\_funcp\_exvector, 87
- print\_integer\_csrc, 227
- print\_operator, 266
- print\_real\_cl\_N, 229
- print\_real\_csrc, 228
- print\_real\_number, 227
- print\_sym\_pow, 262
- product\_to\_exvector, 134
- psi, 157, 237, 238
- psi1\_deriv, 167
- psi1\_eval, 167
- psi1\_evalf, 166
- psi1\_series, 167
- psi2\_deriv, 168
- psi2\_eval, 168
- psi2\_evalf, 167
- psi2\_series, 168
- python, 260
- python\_repr, 261
- quo, 213
- REGISTER\_FUNCTION, 139, 141, 142, 144, 146, 147, 149–152, 154, 155, 160, 161, 164–166, 171–173, 176, 178, 180, 181, 183–189, 191–194
- rank, 208
- read\_real\_float, 226
- read\_unsigned, 90
- real, 250
- real\_part, 113
- real\_part\_conjugate, 140
- real\_part\_eval, 140
- real\_part\_evalf, 139
- real\_part\_expl\_derivative, 140
- real\_part\_funcp, 61
- real\_part\_funcp\_1, 62
- real\_part\_funcp\_10, 77
- real\_part\_funcp\_11, 79
- real\_part\_funcp\_12, 80
- real\_part\_funcp\_13, 82
- real\_part\_funcp\_14, 84
- real\_part\_funcp\_2, 64
- real\_part\_funcp\_3, 65
- real\_part\_funcp\_4, 67
- real\_part\_funcp\_5, 68
- real\_part\_funcp\_6, 70
- real\_part\_funcp\_7, 72
- real\_part\_funcp\_8, 74
- real\_part\_funcp\_9, 75
- real\_part\_funcp\_exvector, 86
- real\_part\_imag\_part, 140
- real\_part\_print\_latex, 140
- real\_part\_real\_part, 140
- reduced\_matrix, 205
- reeval\_ncmul, 210
- registered\_class\_info, 89
- rem, 213
- remove\_dirac\_ONE, 99
- rename\_dummy\_indices, 133
- rename\_dummy\_indices\_uniquely, 136, 137
- replace\_with\_symbol, 223, 224
- reposition\_dummy\_indices, 134
- resultant, 225
- rhs, 120
- rotate\_left, 281
- rows, 206
- S\_deriv, 171
- S\_eval, 171
- S\_evalf, 171
- S\_print\_latex, 172
- S\_series, 171

series, 118  
series\_funcp, 61  
series\_funcp\_1, 63  
series\_funcp\_10, 78  
series\_funcp\_11, 80  
series\_funcp\_12, 81  
series\_funcp\_13, 83  
series\_funcp\_14, 85  
series\_funcp\_2, 64  
series\_funcp\_3, 66  
series\_funcp\_4, 68  
series\_funcp\_5, 69  
series\_funcp\_6, 71  
series\_funcp\_7, 73  
series\_funcp\_8, 74  
series\_funcp\_9, 76  
series\_funcp\_exvector, 87  
series\_to\_poly, 264  
set\_print\_context, 259  
set\_print\_func, 265  
set\_print\_options, 259  
shaker\_sort, 283  
simplify\_indexed, 118, 119, 134  
simplify\_indexed\_product, 135  
sin, 230  
sin\_conjugate, 179  
sin\_deriv, 179  
sin\_eval, 179  
sin\_evalf, 178  
sin\_imag\_part, 179  
sin\_real\_part, 179  
sinh, 233  
sinh\_conjugate, 188  
sinh\_deriv, 187  
sinh\_eval, 187  
sinh\_evalf, 187  
sinh\_imag\_part, 188  
sinh\_real\_part, 187  
smod, 241  
spinor\_metric, 278  
spmap, 88  
sprem, 215  
sqrfree, 222  
sqrfree\_parfrac, 223  
sqrfree\_yun, 221  
sqrt, 244, 264  
sr\_gcd, 217  
step, 246  
step\_conjugate, 145  
step\_eval, 145  
step\_evalf, 145  
step\_imag\_part, 146  
step\_real\_part, 145  
step\_series, 145  
sub\_matrix, 205  
subs, 121, 122  
subsvalue, 195  
swap, 121, 125  
sy\_anti, 273, 274  
sy\_cycl, 274, 275  
sy\_none, 272  
sy\_symm, 273  
sym\_desc\_vec, 88  
symbolic\_matrix, 204, 208  
symm, 271  
symmetric2, 269  
symmetric3, 269  
symmetric4, 269  
symmetrize, 119, 271, 275  
symmetrize\_cyclic, 120, 271, 275  
synthesize\_func, 58  
tan, 231  
tan\_conjugate, 182  
tan\_deriv, 182  
tan\_eval, 181  
tan\_evalf, 181  
tan\_imag\_part, 182  
tan\_real\_part, 182  
tan\_series, 182  
tanh, 233  
tanh\_conjugate, 191  
tanh\_deriv, 190  
tanh\_eval, 190  
tanh\_evalf, 190  
tanh\_imag\_part, 191  
tanh\_real\_part, 190  
tanh\_series, 190  
tensor, 288  
tgamma, 237  
tgamma\_conjugate, 165  
tgamma\_deriv, 165  
tgamma\_eval, 164  
tgamma\_evalf, 164  
tgamma\_series, 165  
to\_double, 250  
to\_int, 249  
to\_long, 249  
to\_polynomial, 116  
to\_rational, 116  
trace, 207  
trace\_string, 97  
transpose, 206  
tree, 261  
tryfactsubs, 209  
uintvector, 88  
unarch\_table\_instance, 287  
unarchive\_map\_t, 59  
unit\_matrix, 204, 208  
unlink\_ex, 125  
unsignedvector, 88  
version\_major, 290  
version\_micro, 291  
version\_minor, 290  
wild, 287  
write\_real\_float, 227  
write\_unsigned, 90

- zeta, [156](#), [236](#)
- zeta1\_deriv, [174](#)
- zeta1\_eval, [173](#)
- zeta1\_evalf, [173](#)
- zeta1\_print\_latex, [174](#)
- zeta2\_deriv, [174](#)
- zeta2\_eval, [174](#)
- zeta2\_evalf, [174](#)
- zeta2\_print\_latex, [175](#)
- zetaderiv\_deriv, [151](#)
- zetaderiv\_eval, [151](#)
- ginac.h, [1045](#)
- golden\_ratio\_hash
  - GiNaC, [282](#)
- guess\_precision
  - GiNaC, [236](#)
- H\_deriv
  - GiNaC, [173](#)
- H\_eval
  - GiNaC, [172](#)
- H\_evalf
  - GiNaC, [172](#)
- H\_print\_latex
  - GiNaC, [173](#)
- H\_series
  - GiNaC, [172](#)
- handle\_factor
  - GiNaC::make\_flat\_inserter, [671](#)
- has
  - GiNaC::basic, [357](#)
  - GiNaC::ex, [493](#)
  - GiNaC::mul, [710](#)
  - GiNaC::numeric, [764](#)
  - GiNaC::power, [815](#)
  - GiNaC::structure, [918](#)
  - GiNaC::symbolset, [952](#)
  - GiNaC, [114](#)
- has\_cyclic
  - GiNaC::symmetry, [957](#)
- has\_derivative
  - GiNaC::function\_options, [601](#)
- has\_dummy\_index\_for
  - GiNaC::indexed, [636](#)
- has\_ex
  - GiNaC::archive\_node, [343](#)
- has\_expression
  - GiNaC::archive\_node, [345](#)
- has\_nonsymmetric
  - GiNaC::symmetry, [957](#)
- has\_power
  - GiNaC::function\_options, [601](#)
- has\_same\_ex\_as
  - GiNaC::archive\_node, [343](#)
- has\_symmetry
  - GiNaC::symmetry, [957](#)
- has\_trailing\_zero
  - GiNaC::integration\_kernel, [651](#)
- hash\_map.h, [1046](#)
- hash\_seed.h, [1046](#)
- hashvalue
  - GiNaC::basic, [374](#)
  - GiNaC::remember\_table\_entry, [890](#)
- hasindex
  - GiNaC, [135](#)
- haswild
  - GiNaC, [287](#)
- head
  - GiNaC::composition\_generator::coolmulti, [445](#)
- heur\_gcd
  - GiNaC, [219](#)
- heur\_gcd\_z
  - GiNaC, [218](#)
- hold
  - GiNaC::basic, [370](#)
- hold\_ncmul
  - GiNaC::ncmul, [755](#)
  - GiNaC, [210](#)
- I
  - GiNaC, [290](#)
- i
  - GiNaC::composition\_generator::coolmulti, [445](#)
  - GiNaC::const\_iterator, [413](#)
  - GiNaC::internal::\_iter\_rep, [312](#)
- i\_end
  - GiNaC::internal::\_iter\_rep, [312](#)
- id
  - GiNaC::print\_context\_options, [826](#)
- identify\_parents
  - GiNaC::class\_info, [384](#)
- idx
  - GiNaC::idx, [616](#)
  - GiNaC, [289](#)
- idx.cpp, [1047](#)
- idx.h, [1048](#)
- idx\_symmetrization
  - GiNaC, [134](#)
- ifactor
  - GiNaC, [196](#)
- imag
  - GiNaC::numeric, [782](#)
  - GiNaC, [250](#)
- imag\_part
  - GiNaC::add, [323](#)
  - GiNaC::basic, [366](#)
  - GiNaC::constant, [422](#)
  - GiNaC::container, [434](#)
  - GiNaC::ex, [493](#)
  - GiNaC::function, [557](#)
  - GiNaC::indexed, [633](#)
  - GiNaC::matrix, [681](#)
  - GiNaC::mul, [711](#)
  - GiNaC::ncmul, [753](#)
  - GiNaC::numeric, [768](#)
  - GiNaC::power, [816](#)
  - GiNaC::pseries, [854](#)
  - GiNaC::realsymbol, [870](#)

- GiNaC::symbol, 946
  - GiNaC, 113
- imag\_part\_conjugate
  - GiNaC, 141
- imag\_part\_eval
  - GiNaC, 141
- imag\_part\_evalf
  - GiNaC, 141
- imag\_part\_expl\_derivative
  - GiNaC, 142
- imag\_part\_f
  - GiNaC::function\_options, 603
- imag\_part\_func
  - GiNaC::function\_options, 578–580, 595
- imag\_part\_funcp
  - GiNaC, 61
- imag\_part\_funcp\_1
  - GiNaC, 62
- imag\_part\_funcp\_10
  - GiNaC, 77
- imag\_part\_funcp\_11
  - GiNaC, 79
- imag\_part\_funcp\_12
  - GiNaC, 81
- imag\_part\_funcp\_13
  - GiNaC, 83
- imag\_part\_funcp\_14
  - GiNaC, 85
- imag\_part\_funcp\_2
  - GiNaC, 64
- imag\_part\_funcp\_3
  - GiNaC, 65
- imag\_part\_funcp\_4
  - GiNaC, 67
- imag\_part\_funcp\_5
  - GiNaC, 69
- imag\_part\_funcp\_6
  - GiNaC, 70
- imag\_part\_funcp\_7
  - GiNaC, 72
- imag\_part\_funcp\_8
  - GiNaC, 74
- imag\_part\_funcp\_9
  - GiNaC, 75
- imag\_part\_funcp\_exvector
  - GiNaC, 86
- imag\_part\_imag\_part
  - GiNaC, 142
- imag\_part\_print\_latex
  - GiNaC, 141
- imag\_part\_real\_part
  - GiNaC, 142
- imag\_part\_use\_exvector\_args
  - GiNaC::function\_options, 606
- impl
  - GiNaC::print\_functor, 833
- increment
  - GiNaC::const\_postorder\_iterator, 415
- GiNaC::const\_preorder\_iterator, 418
- index0
  - GiNaC, 268
- index1
  - GiNaC, 268
- index2
  - GiNaC, 269
- index3
  - GiNaC, 269
- index\_dimensions
  - GiNaC, 262
- indexed
  - GiNaC::indexed, 626–631
- indexed.cpp, 1049
- indexed.h, 1051
- indices
  - GiNaC::symmetry, 959
- indices\_consistent
  - GiNaC, 132
- info
  - GiNaC::add, 319
  - GiNaC::basic, 355
  - GiNaC::class\_info::tree\_node, 975
  - GiNaC::constant, 421
  - GiNaC::container, 430
  - GiNaC::ex, 489
  - GiNaC::expairseq, 525
  - GiNaC::function, 557
  - GiNaC::idx, 616
  - GiNaC::indexed, 632
  - GiNaC::minkmetric, 696
  - GiNaC::mul, 709
  - GiNaC::ncmul, 750
  - GiNaC::numeric, 763
  - GiNaC::power, 811
  - GiNaC::relational, 879
  - GiNaC::spinmetric, 907
  - GiNaC::structure, 916
  - GiNaC::symbol, 943
  - GiNaC::tensdelta, 963
  - GiNaC::tensepsilon, 966
  - GiNaC::tensmetric, 969
- info\_f
  - GiNaC::function\_options, 604
- info\_func
  - GiNaC::function\_options, 592–594, 596
- info\_funcp
  - GiNaC, 62
- info\_funcp\_1
  - GiNaC, 63
- info\_funcp\_10
  - GiNaC, 78
- info\_funcp\_11
  - GiNaC, 80
- info\_funcp\_12
  - GiNaC, 82
- info\_funcp\_13
  - GiNaC, 84

- info\_funcp\_14
  - GiNaC, [86](#)
- info\_funcp\_2
  - GiNaC, [65](#)
- info\_funcp\_3
  - GiNaC, [66](#)
- info\_funcp\_4
  - GiNaC, [68](#)
- info\_funcp\_5
  - GiNaC, [70](#)
- info\_funcp\_6
  - GiNaC, [71](#)
- info\_funcp\_7
  - GiNaC, [73](#)
- info\_funcp\_8
  - GiNaC, [75](#)
- info\_funcp\_9
  - GiNaC, [76](#)
- info\_funcp\_exvector
  - GiNaC, [87](#)
- info\_use\_exvector\_args
  - GiNaC::function\_options, [608](#)
- inifcns.cpp, [1052](#)
- inifcns.h, [1055](#)
- inifcns\_elliptic.cpp, [1056](#)
- inifcns\_gamma.cpp, [1058](#)
- inifcns\_nstdsums.cpp, [1059](#)
- inifcns\_trans.cpp, [1061](#)
- init
  - GiNaC::basic\_multi\_iterator, [379](#)
  - GiNaC::multi\_iterator\_counter, [722](#)
  - GiNaC::multi\_iterator\_counter\_indv, [725](#)
  - GiNaC::multi\_iterator\_ordered, [728](#)
  - GiNaC::multi\_iterator\_ordered\_eq, [731](#)
  - GiNaC::multi\_iterator\_ordered\_eq\_indv, [734](#)
  - GiNaC::multi\_iterator\_permutation, [737](#)
  - GiNaC::multi\_iterator\_shuffle, [740](#)
  - GiNaC::multi\_iterator\_shuffle\_prime, [743](#)
- init\_table
  - GiNaC::remember\_table, [888](#)
- init\_unarchivers
  - GiNaC::library\_init, [669](#)
- initialize
  - GiNaC::function\_options, [568](#)
- insert
  - GiNaC::unarchive\_table\_t, [976](#)
- insert\_symbols
  - GiNaC::symbolset, [951](#)
- int\_length
  - GiNaC::numeric, [783](#)
- integer\_content
  - GiNaC::add, [321](#)
  - GiNaC::basic, [362](#)
  - GiNaC::ex, [503](#)
  - GiNaC::mul, [712](#)
  - GiNaC::numeric, [766](#)
  - GiNaC::structure, [922](#)
- integral
  - GiNaC::error\_and\_integral, [477](#)
  - GiNaC::integral, [642](#)
- integral.cpp, [1064](#)
- integral.h, [1065](#)
- integration\_kernel.cpp, [1065](#)
  - cache\_vec, [1067](#)
  - order, [1067](#)
  - qbar, [1067](#)
  - x, [1067](#)
- integration\_kernel.h, [1068](#)
- interpolate
  - GiNaC, [217](#)
- inv\_at\_cit
  - GiNaC::archive, [329](#)
- inverse
  - GiNaC::matrix, [688](#)
  - GiNaC::numeric, [774](#)
  - GiNaC, [207](#), [246](#)
- inverse\_atoms
  - GiNaC::archive, [335](#)
- iquo
  - GiNaC, [242](#), [243](#)
- irem
  - GiNaC, [241](#), [242](#)
- is\_a
  - GiNaC::ex, [514](#)
  - GiNaC, [92](#), [122](#), [264](#)
- is\_canonical
  - GiNaC::expairseq, [534](#)
- is\_canonical\_numeric
  - GiNaC::expair, [519](#)
- is\_cinteger
  - GiNaC::numeric, [778](#)
  - GiNaC, [249](#)
- is\_clifford\_tinfo
  - GiNaC, [104](#)
- is\_color\_tinfo
  - GiNaC, [108](#)
- is\_compatible\_to
  - GiNaC::pseries, [856](#)
- is\_contravariant
  - GiNaC::varidx, [984](#)
- is\_covariant
  - GiNaC::varidx, [984](#)
- is\_crational
  - GiNaC::numeric, [778](#)
  - GiNaC, [249](#)
- is\_defined
  - GiNaC::scalar\_products, [898](#)
- is\_dim\_numeric
  - GiNaC::idx, [621](#)
- is\_dim\_symbolic
  - GiNaC::idx, [621](#)
- is\_dirac\_slash
  - GiNaC, [94](#)
- is\_discriminant\_of\_quadratic\_number\_field
  - GiNaC, [196](#)
- is\_dotted

- GiNaC::spinidx, 904
- is\_dummy\_pair
  - GiNaC, 129
- is\_dummy\_pair\_same\_type
  - GiNaC::idx, 620
  - GiNaC::spinidx, 903
  - GiNaC::varidx, 982
- is\_equal
  - GiNaC::basic, 370
  - GiNaC::ex, 507
  - GiNaC::expair, 518
  - GiNaC::numeric, 775
  - GiNaC::remember\_table\_entry, 890
- is\_equal\_same\_type
  - GiNaC::basic, 367
  - GiNaC::constant, 423
  - GiNaC::container, 434
  - GiNaC::expairseq, 528
  - GiNaC::fderivative, 544
  - GiNaC::function, 558
  - GiNaC::numeric, 769
  - GiNaC::structure, 926
  - GiNaC::symbol, 947
- is\_even
  - GiNaC::numeric, 777
  - GiNaC, 248
- is\_ex\_the\_function
  - function.h, 1045
- is\_exactly\_a
  - GiNaC::ex, 514
  - GiNaC, 92, 122
- is\_integer
  - GiNaC::numeric, 776
  - GiNaC, 247
- is\_less
  - GiNaC::expair, 518
- is\_negative
  - GiNaC::numeric, 776
  - GiNaC, 247
- is\_nonneg\_integer
  - GiNaC::numeric, 777
  - GiNaC, 247
- is\_numeric
  - GiNaC::ELi\_kernel, 474
  - GiNaC::Ebar\_kernel, 458
  - GiNaC::Eisenstein\_h\_kernel, 462
  - GiNaC::Eisenstein\_kernel, 468
  - GiNaC::Kronecker\_dtau\_kernel, 659
  - GiNaC::Kronecker\_dz\_kernel, 663
  - GiNaC::idx, 620
  - GiNaC::integration\_kernel, 652
  - GiNaC::modular\_form\_kernel, 701
  - GiNaC::multiple\_polylog\_kernel, 745
  - GiNaC::user\_defined\_kernel, 979
- is\_odd
  - GiNaC::numeric, 777
  - GiNaC, 248
- is\_order\_function
  - GiNaC, 158
- is\_polynomial
  - GiNaC::add, 319
  - GiNaC::basic, 359
  - GiNaC::constant, 421
  - GiNaC::ex, 496
  - GiNaC::mul, 709
  - GiNaC::numeric, 763
  - GiNaC::power, 812
  - GiNaC::symbol, 946
  - GiNaC, 114
- is\_pos\_integer
  - GiNaC::numeric, 776
  - GiNaC, 247
- is\_positive
  - GiNaC::numeric, 776
  - GiNaC, 247
- is\_prime
  - GiNaC::numeric, 777
  - GiNaC, 248
- is\_rational
  - GiNaC::numeric, 778
  - GiNaC, 248
- is\_real
  - GiNaC::numeric, 778
  - GiNaC, 248
- is\_symbolic
  - GiNaC::idx, 620
- is\_terminating
  - GiNaC::pseries, 857
  - GiNaC, 265
- is\_the\_function
  - GiNaC, 128
- is\_the\_function< G\_SERIAL >
  - GiNaC, 157
- is\_the\_function< iterated\_integral\_SERIAL >
  - GiNaC, 158
- is\_the\_function< psi\_SERIAL >
  - GiNaC, 158
- is\_the\_function< zeta\_SERIAL >
  - GiNaC, 156
- is\_undotted
  - GiNaC::spinidx, 904
- is\_valid
  - GiNaC::print\_funcutor, 832
- is\_zero
  - GiNaC::ex, 507
  - GiNaC::numeric, 775
  - GiNaC::pseries, 856
  - GiNaC, 121, 246
- is\_zero\_matrix
  - GiNaC::ex, 508
  - GiNaC::matrix, 690
- isqrt
  - GiNaC, 244
- iterated\_integral
  - GiNaC, 158
- iterated\_integral2\_eval



- GiNaC, [162](#)
- iterated\_integral2\_evalf
  - GiNaC, [162](#)
- iterated\_integral3\_eval
  - GiNaC, [162](#)
- iterated\_integral3\_evalf
  - GiNaC, [162](#)
- iterated\_integral\_evalf\_impl
  - GiNaC, [161](#)
- K
  - GiNaC::Eisenstein\_kernel, [470](#)
  - GiNaC::Kronecker\_dtau\_kernel, [661](#)
  - GiNaC::Kronecker\_dz\_kernel, [665](#)
- k
  - factor.cpp, [1026](#)
  - GiNaC::Eisenstein\_h\_kernel, [464](#)
  - GiNaC::Eisenstein\_kernel, [469](#)
  - GiNaC::modular\_form\_kernel, [702](#)
- Kronecker\_dtau\_kernel
  - GiNaC::Kronecker\_dtau\_kernel, [658](#)
- Kronecker\_dz\_kernel
  - GiNaC::Kronecker\_dz\_kernel, [662](#)
- kronecker\_symbol
  - GiNaC, [196](#)
- label
  - GiNaC::wildcard, [989](#)
- lanczos\_coeffs
  - GiNaC::lanczos\_coeffs, [666](#)
- last
  - factor.cpp, [1026](#)
- last\_access
  - GiNaC::remember\_table\_entry, [891](#)
- latex
  - GiNaC, [260](#)
- latex\_name
  - GiNaC::function\_options, [569](#)
- Laurent\_series
  - GiNaC::Eisenstein\_h\_kernel, [462](#)
  - GiNaC::Eisenstein\_kernel, [468](#)
  - GiNaC::integration\_kernel, [652](#)
  - GiNaC::modular\_form\_kernel, [701](#)
  - GiNaC::user\_defined\_kernel, [979](#)
- lcm
  - GiNaC, [221](#), [243](#)
- lcm\_of\_coefficients\_denominators
  - GiNaC, [212](#)
- lcmcoeff
  - GiNaC, [212](#)
- lcoeff
  - GiNaC::ex, [497](#)
- ldeg\_a
  - GiNaC::sym\_desc, [940](#)
- ldeg\_b
  - GiNaC::sym\_desc, [940](#)
- ldegree
  - GiNaC::add, [320](#)
  - GiNaC::basic, [359](#)
- GiNaC::ex, [497](#)
  - GiNaC::integral, [643](#)
  - GiNaC::mul, [709](#)
  - GiNaC::ncmul, [750](#)
  - GiNaC::numeric, [764](#)
  - GiNaC::power, [812](#)
  - GiNaC::pseries, [851](#)
  - GiNaC::structure, [920](#)
  - GiNaC, [115](#)
- len
  - factor.cpp, [1026](#)
- let\_op
  - GiNaC::ELi\_kernel, [473](#)
  - GiNaC::Ebar\_kernel, [457](#)
  - GiNaC::Eisenstein\_h\_kernel, [462](#)
  - GiNaC::Eisenstein\_kernel, [468](#)
  - GiNaC::Kronecker\_dtau\_kernel, [659](#)
  - GiNaC::Kronecker\_dz\_kernel, [663](#)
  - GiNaC::basic, [356](#)
  - GiNaC::clifford, [393](#)
  - GiNaC::container, [432](#)
  - GiNaC::ex, [491](#)
  - GiNaC::integral, [644](#)
  - GiNaC::matrix, [679](#)
  - GiNaC::modular\_form\_kernel, [700](#)
  - GiNaC::multiple\_polylog\_kernel, [745](#)
  - GiNaC::structure, [917](#)
  - GiNaC::user\_defined\_kernel, [979](#)
- lgamma
  - GiNaC, [236](#), [237](#)
- lgamma\_conjugate
  - GiNaC, [164](#)
- lgamma\_deriv
  - GiNaC, [163](#)
- lgamma\_eval
  - GiNaC, [163](#)
- lgamma\_evalf
  - GiNaC, [163](#)
- lgamma\_series
  - GiNaC, [163](#)
- lh
  - GiNaC::relational, [884](#)
- lhs
  - GiNaC::ex, [492](#)
  - GiNaC::relational, [883](#)
  - GiNaC, [120](#)
- Li2
  - GiNaC, [236](#)
- Li2\_
  - GiNaC, [235](#)
- Li2\_conjugate
  - GiNaC, [150](#)
- Li2\_deriv
  - GiNaC, [149](#)
- Li2\_eval
  - GiNaC, [149](#)
- Li2\_evalf
  - GiNaC, [149](#)

- Li2\_projection
  - GiNaC, 235
- Li2\_series
  - GiNaC, 150, 235
- Li3\_eval
  - GiNaC, 150
- Li\_deriv
  - GiNaC, 170
- Li\_eval
  - GiNaC, 170
- Li\_evalf
  - GiNaC, 169
- Li\_print\_latex
  - GiNaC, 170
- Li\_series
  - GiNaC, 170
- library\_init
  - GiNaC::library\_init, 669
- library\_initializer
  - GiNaC, 289
- likely
  - compiler.h, 1010
- link\_ex
  - GiNaC, 124, 125
- log
  - GiNaC, 229
- log2
  - GiNaC, 281
- log\_conjugate
  - GiNaC, 178
- log\_deriv
  - GiNaC, 177
- log\_eval
  - GiNaC, 177
- log\_evalf
  - GiNaC, 177
- log\_expand
  - GiNaC, 178
- log\_imag\_part
  - GiNaC, 178
- log\_real\_part
  - GiNaC, 177
- log\_series
  - GiNaC, 177
- lookup\_entry
  - GiNaC::remember\_table, 887
  - GiNaC::remember\_table\_list, 893
- lookup\_map
  - GiNaC, 88
- lookup\_remember\_table
  - GiNaC::function, 560
- lorentz\_eps
  - GiNaC, 280
- lorentz\_g
  - GiNaC, 278
- lr
  - factor.cpp, 1025
- lsolve
  - GiNaC, 155
- lst
  - GiNaC, 88
- lst.cpp, 1069
- lst.h, 1069
- lst\_to\_clifford
  - GiNaC, 100, 101
- lst\_to\_matrix
  - GiNaC, 204
- m
  - factor.cpp, 1024
  - GiNaC::ELi\_kernel, 475
  - GiNaC::Ebar\_kernel, 459
  - GiNaC::basic\_partition\_generator::mpartition2, 704
  - GiNaC::matrix, 694
  - GiNaC::partition\_with\_zero\_parts\_generator, 789
- make\_flat
  - GiNaC::expairseq, 533
- make\_flat\_inserter
  - GiNaC::make\_flat\_inserter, 671
- make\_hash\_seed
  - GiNaC, 128
- make\_real\_float
  - GiNaC, 226
- make\_return\_type\_t
  - GiNaC, 265
- make\_safe\_bool
  - GiNaC::relational, 883
- makewritable
  - GiNaC::ptr, 865
- makewriteable
  - GiNaC::ex, 512
- map
  - GiNaC::basic, 358
  - GiNaC::ex, 495
  - GiNaC::expairseq, 526
  - GiNaC::idx, 617
  - GiNaC::power, 811
  - GiNaC::relational, 880
  - GiNaC::structure, 919
- map\_eval\_integ
  - GiNaC, 288
- map\_evalm
  - GiNaC, 287
- markowitz\_elimination
  - GiNaC::matrix, 692
- match
  - GiNaC::basic, 357
  - GiNaC::ex, 493, 494
  - GiNaC::expairseq, 527
  - GiNaC::structure, 918
  - GiNaC::wildcard, 987
  - GiNaC, 118
- match\_same\_type
  - GiNaC::basic, 358
  - GiNaC::clifford, 390
  - GiNaC::color, 399

- GiNaC::fderivative, [544](#)
  - GiNaC::function, [558](#)
  - GiNaC::idx, [619](#)
  - GiNaC::matrix, [682](#)
  - GiNaC::relational, [881](#)
  - GiNaC::spinidx, [904](#)
  - GiNaC::structure, [918](#)
  - GiNaC::varidx, [983](#)
- matrix
  - GiNaC::matrix, [677](#), [678](#)
- matrix.cpp, [1070](#)
- matrix.h, [1071](#)
- max\_assoc\_size
  - GiNaC::remember\_table, [888](#)
  - GiNaC::remember\_table\_list, [893](#)
- max\_coefficient
  - GiNaC::add, [322](#)
  - GiNaC::basic, [363](#)
  - GiNaC::ex, [505](#)
  - GiNaC::mul, [713](#)
  - GiNaC::numeric, [768](#)
  - GiNaC::structure, [923](#)
- max\_deg
  - GiNaC::sym\_desc, [940](#)
- max\_integration\_level
  - GiNaC::integral, [648](#)
- max\_lcnops
  - GiNaC::sym\_desc, [940](#)
- metric
  - GiNaC::clifford, [394](#)
- metric\_tensor
  - GiNaC, [277](#)
- minimal\_dim
  - GiNaC::idx, [622](#)
  - GiNaC, [130](#)
- minkmetric
  - GiNaC::minkmetric, [696](#)
- minkowski
  - GiNaC::tensepsilon, [968](#)
- mod
  - GiNaC, [240](#)
- modular\_form\_kernel
  - GiNaC::modular\_form\_kernel, [699](#)
- mpartition2
  - GiNaC::basic\_partition\_generator::mpartition2, [703](#)
- mpgen
  - GiNaC::basic\_partition\_generator, [382](#)
- mul
  - GiNaC::add, [327](#)
  - GiNaC::matrix, [684](#)
  - GiNaC::mul, [707](#), [708](#)
  - GiNaC::numeric, [771](#)
  - GiNaC::power, [821](#)
- mul.cpp, [1072](#)
- mul.h, [1073](#)
- mul\_const
  - GiNaC::pseries, [858](#)
- mul\_dyn
  - GiNaC::numeric, [772](#)
- mul\_scalar
  - GiNaC::matrix, [684](#)
- mul\_series
  - GiNaC::pseries, [858](#)
- multi\_iterator\_counter
  - GiNaC::multi\_iterator\_counter, [722](#)
- multi\_iterator\_counter\_indv
  - GiNaC::multi\_iterator\_counter\_indv, [724](#), [725](#)
- multi\_iterator\_ordered
  - GiNaC::multi\_iterator\_ordered, [727](#), [728](#)
- multi\_iterator\_ordered\_eq
  - GiNaC::multi\_iterator\_ordered\_eq, [730](#)
- multi\_iterator\_ordered\_eq\_indv
  - GiNaC::multi\_iterator\_ordered\_eq\_indv, [733](#)
- multi\_iterator\_permutation
  - GiNaC::multi\_iterator\_permutation, [736](#)
- multi\_iterator\_shuffle
  - GiNaC::multi\_iterator\_shuffle, [739](#)
- multi\_iterator\_shuffle\_prime
  - GiNaC::multi\_iterator\_shuffle\_prime, [742](#), [743](#)
- multinomial\_coefficient
  - GiNaC, [281](#)
- multiple\_polylog\_kernel
  - GiNaC::multiple\_polylog\_kernel, [745](#)
- multiply\_lcm
  - GiNaC, [212](#)
- my\_ios\_callback
  - GiNaC, [259](#)
- my\_ios\_index
  - GiNaC, [258](#)
- N
  - GiNaC::Eisenstein\_h\_kernel, [464](#)
  - GiNaC::Eisenstein\_kernel, [470](#)
  - GiNaC::basic\_multi\_iterator, [380](#)
- n
  - factor.cpp, [1025](#)
  - GiNaC::ELi\_kernel, [475](#)
  - GiNaC::Ebar\_kernel, [459](#)
  - GiNaC::Kronecker\_dtau\_kernel, [660](#)
  - GiNaC::Kronecker\_dz\_kernel, [664](#)
  - GiNaC::basic\_partition\_generator::mpartition2, [704](#)
- N\_internal
  - GiNaC::multi\_iterator\_shuffle, [741](#)
- name
  - GiNaC::archive::archived\_ex, [347](#)
  - GiNaC::archive\_node::property, [844](#)
  - GiNaC::archive\_node::property\_info, [846](#)
  - GiNaC::constant, [425](#)
  - GiNaC::function\_options, [602](#)
  - GiNaC::print\_context\_options, [825](#)
  - GiNaC::registered\_class\_options, [876](#)
  - GiNaC::symbol, [950](#)
- ncmul
  - GiNaC::mul, [720](#)
  - GiNaC::ncmul, [748](#), [749](#)

- ncmul.cpp, 1074
- ncmul.h, 1075
- next
  - GiNaC::class\_info, 385
  - GiNaC::composition\_generator, 406
  - GiNaC::composition\_generator::coolmulti::element, 471
  - GiNaC::partition\_generator, 787
  - GiNaC::partition\_with\_zero\_parts\_generator, 789
- next\_partition
  - GiNaC::basic\_partition\_generator::mpartition2, 703
- next\_permutation
  - GiNaC::composition\_generator::coolmulti, 444
- next\_print\_context\_id
  - GiNaC, 290
- next\_serial
  - GiNaC::constant, 425
  - GiNaC::symbol, 950
- no\_index\_dimensions
  - GiNaC, 262
- no\_type
  - GiNaC::has\_distance, 612
- nodes
  - GiNaC::archive, 334
- nonnull
  - GiNaC::relational::safe\_bool\_helper, 896
- nops
  - GiNaC::ELi\_kernel, 473
  - GiNaC::Ebar\_kernel, 457
  - GiNaC::Eisenstein\_h\_kernel, 461
  - GiNaC::Eisenstein\_kernel, 467
  - GiNaC::Kronecker\_dtau\_kernel, 659
  - GiNaC::Kronecker\_dz\_kernel, 663
  - GiNaC::basic, 355
  - GiNaC::clifford, 392
  - GiNaC::container, 431
  - GiNaC::ex, 490
  - GiNaC::expairseq, 525
  - GiNaC::idx, 617
  - GiNaC::integral, 644
  - GiNaC::matrix, 678
  - GiNaC::modular\_form\_kernel, 700
  - GiNaC::multiple\_polylog\_kernel, 745
  - GiNaC::power, 811
  - GiNaC::pseries, 850
  - GiNaC::relational, 879
  - GiNaC::structure, 916
  - GiNaC::user\_defined\_kernel, 978
  - GiNaC, 112, 205
- normal
  - GiNaC::add, 321
  - GiNaC::basic, 361
  - GiNaC::ex, 499
  - GiNaC::mul, 712
  - GiNaC::numeric, 765
  - GiNaC::power, 815
  - GiNaC::pseries, 853
  - GiNaC::structure, 921
  - GiNaC::symbol, 944
  - GiNaC, 116
- normal.cpp, 1075
  - FAST\_COMPARE, 1077
  - STATISTICS, 1078
  - USE\_REMEMBER, 1078
  - USE\_TRIAL\_DIVISION, 1078
- normal.h, 1078
- not\_symmetric
  - GiNaC, 269
- nparams
  - GiNaC::function\_options, 602
- num
  - GiNaC::symminfo, 961
- num\_expressions
  - GiNaC::archive, 332
- number
  - GiNaC::constant, 425
- number\_of\_type
  - GiNaC, 133
- numer
  - GiNaC::ex, 501
  - GiNaC::numeric, 783
  - GiNaC, 115, 250
- numer\_denom
  - GiNaC::ex, 501
  - GiNaC, 116
- numeric
  - GiNaC::numeric, 760–762
- numeric.cpp, 1079
- numeric.h, 1083
- Nv
  - GiNaC::multi\_iterator\_counter\_indv, 726
  - GiNaC::multi\_iterator\_ordered\_eq\_indv, 734
- o
  - GiNaC::relational, 884
- obj
  - GiNaC::structure, 927
- one
  - factor.cpp, 1025
- op
  - GiNaC::ELi\_kernel, 473
  - GiNaC::Ebar\_kernel, 457
  - GiNaC::Eisenstein\_h\_kernel, 462
  - GiNaC::Eisenstein\_kernel, 467
  - GiNaC::Kronecker\_dtau\_kernel, 659
  - GiNaC::Kronecker\_dz\_kernel, 663
  - GiNaC::basic, 356
  - GiNaC::clifford, 392
  - GiNaC::container, 431
  - GiNaC::ex, 490
  - GiNaC::expairseq, 525
  - GiNaC::idx, 617
  - GiNaC::integral, 644
  - GiNaC::matrix, 678
  - GiNaC::modular\_form\_kernel, 700
  - GiNaC::multiple\_polylog\_kernel, 745

- GiNaC::power, 811
  - GiNaC::pseries, 850
  - GiNaC::relational, 880
  - GiNaC::structure, 916
  - GiNaC::user\_defined\_kernel, 979
  - GiNaC, 120
- operator long
  - GiNaC::\_numeric\_digits, 314
- operator safe\_bool
  - GiNaC::relational, 883
- operator!
  - GiNaC::relational, 883
- operator!=
  - GiNaC::const\_iterator, 411
  - GiNaC::const\_postorder\_iterator, 415
  - GiNaC::const\_preorder\_iterator, 418
  - GiNaC::internal::\_iter\_rep, 312
  - GiNaC::numeric, 779
  - GiNaC::ptr, 866, 867
  - GiNaC::return\_type\_t, 894
  - GiNaC, 257
- operator<
  - GiNaC::const\_iterator, 411
  - GiNaC::numeric, 779
  - GiNaC::return\_type\_t, 894
  - GiNaC::spmapkey, 910
  - GiNaC::sym\_desc, 939
  - GiNaC, 258
- operator<<
  - GiNaC::archive, 334
  - GiNaC::archive\_node, 344
  - GiNaC::basic\_multi\_iterator, 380
  - GiNaC::multi\_iterator\_counter, 723
  - GiNaC::multi\_iterator\_counter\_indv, 726
  - GiNaC::multi\_iterator\_ordered, 729
  - GiNaC::multi\_iterator\_ordered\_eq, 731
  - GiNaC::multi\_iterator\_ordered\_eq\_indv, 734
  - GiNaC::multi\_iterator\_permutation, 738
  - GiNaC::multi\_iterator\_shuffle, 740
  - GiNaC::multi\_iterator\_shuffle\_prime, 743
  - GiNaC::ptr, 867
  - GiNaC, 90, 112, 245, 260, 284–286
- operator<=
  - GiNaC::const\_iterator, 411
  - GiNaC::numeric, 779
  - GiNaC, 258
- operator>
  - GiNaC::const\_iterator, 411
  - GiNaC::numeric, 781
  - GiNaC, 258
- operator>>
  - GiNaC::archive, 334
  - GiNaC::archive\_node, 344
  - GiNaC, 91, 260
- operator>=
  - GiNaC::const\_iterator, 412
  - GiNaC::numeric, 781
  - GiNaC, 258
- operator\*
  - GiNaC::const\_iterator, 409
  - GiNaC::const\_postorder\_iterator, 414
  - GiNaC::const\_preorder\_iterator, 417
  - GiNaC::ptr, 865
  - GiNaC, 252, 253
- operator\*=
  - GiNaC, 253, 254
- operator()
  - GiNaC::basic\_multi\_iterator, 379
  - GiNaC::derivative\_map\_function, 446
  - GiNaC::error\_and\_integral\_is\_less, 477
  - GiNaC::eval\_integ\_map\_function, 478
  - GiNaC::evalf\_map\_function, 479
  - GiNaC::evalm\_map\_function, 480
  - GiNaC::ex\_base\_is\_less, 515
  - GiNaC::ex\_is\_equal, 515
  - GiNaC::ex\_is\_less, 516
  - GiNaC::ex\_swap, 516
  - GiNaC::expair\_is\_less, 520
  - GiNaC::expair\_rest\_is\_less, 521
  - GiNaC::expair\_swap, 521
  - GiNaC::expand\_map\_function, 536
  - GiNaC::idx\_is\_equal\_ignore\_dim, 624
  - GiNaC::is\_not\_a\_clifford, 655
  - GiNaC::is\_summation\_idx, 655
  - GiNaC::map\_function, 674
  - GiNaC::matrix, 685
  - GiNaC::normal\_map\_function, 756
  - GiNaC::op0\_is\_equal, 786
  - GiNaC::pointer\_to\_map\_function, 791
  - GiNaC::pointer\_to\_map\_function\_1arg, 792
  - GiNaC::pointer\_to\_map\_function\_2args, 794
  - GiNaC::pointer\_to\_map\_function\_3args, 796
  - GiNaC::pointer\_to\_member\_to\_map\_function, 798
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
1arg, 799
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
2args, 801
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
3args, 803
  - GiNaC::print\_functor, 832
  - GiNaC::print\_functor\_impl, 834
  - GiNaC::print\_memfun\_handler, 837
  - GiNaC::print\_ptrfun\_handler, 839
  - GiNaC::sy\_is\_less, 936
  - GiNaC::sy\_swap, 937
  - GiNaC::symminfo\_is\_less\_by\_orig, 961
  - GiNaC::symminfo\_is\_less\_by\_symmterm, 962
  - GiNaC::terminfo\_is\_less, 974
  - std::equal\_to< GiNaC::ex >, 476
  - std::hash< GiNaC::ex >, 614
  - std::less< GiNaC::ptr< T > >, 667
- operator+
  - GiNaC::const\_iterator, 410, 412
  - GiNaC, 251, 252, 255
- operator++
  - GiNaC::basic\_multi\_iterator, 379

- GiNaC::const\_iterator, [409](#), [410](#)
  - GiNaC::const\_postorder\_iterator, [414](#), [415](#)
  - GiNaC::const\_preorder\_iterator, [417](#)
  - GiNaC::multi\_iterator\_counter, [722](#)
  - GiNaC::multi\_iterator\_counter\_indv, [725](#)
  - GiNaC::multi\_iterator\_ordered, [728](#)
  - GiNaC::multi\_iterator\_ordered\_eq, [731](#)
  - GiNaC::multi\_iterator\_ordered\_eq\_indv, [734](#)
  - GiNaC::multi\_iterator\_permutation, [737](#)
  - GiNaC::multi\_iterator\_shuffle, [740](#)
  - GiNaC, [255–257](#)
- operator+=
  - GiNaC::const\_iterator, [410](#)
  - GiNaC, [253](#), [254](#)
- operator-
  - GiNaC::const\_iterator, [411](#), [412](#)
  - GiNaC, [252](#), [255](#)
- operator->
  - GiNaC::const\_iterator, [409](#)
  - GiNaC::const\_postorder\_iterator, [414](#)
  - GiNaC::const\_preorder\_iterator, [417](#)
  - GiNaC::ptr, [865](#)
  - GiNaC::structure, [926](#)
- operator--
  - GiNaC::const\_iterator, [410](#)
  - GiNaC, [255–257](#)
- operator-=
  - GiNaC::const\_iterator, [410](#)
  - GiNaC, [253](#), [254](#)
- operator/
  - GiNaC, [252](#), [253](#)
- operator/=
  - GiNaC, [254](#)
- operator=
  - GiNaC::\_numeric\_digits, [314](#)
  - GiNaC::archive\_node, [339](#)
  - GiNaC::basic, [352](#)
  - GiNaC::numeric, [773](#), [774](#)
  - GiNaC::print\_functor, [832](#)
  - GiNaC::ptr, [865](#)
- operator==
  - GiNaC::const\_iterator, [411](#)
  - GiNaC::const\_postorder\_iterator, [415](#)
  - GiNaC::const\_preorder\_iterator, [418](#)
  - GiNaC::internal::\_iter\_rep, [311](#)
  - GiNaC::numeric, [779](#)
  - GiNaC::ptr, [866](#), [867](#)
  - GiNaC::return\_type\_t, [894](#)
  - GiNaC::spmapkey, [909](#)
  - GiNaC, [257](#)
- operator[]
  - GiNaC::basic, [356](#), [357](#)
  - GiNaC::basic\_multi\_iterator, [378](#), [379](#)
  - GiNaC::const\_iterator, [409](#)
  - GiNaC::ex, [491](#), [492](#)
  - GiNaC::structure, [917](#)
- operators
  - GiNaC::relational, [878](#)
- operators.cpp, [1085](#)
- operators.h, [1087](#)
- options
  - factor.cpp, [1028](#)
  - GiNaC::class\_info, [385](#)
  - GiNaC::expand\_map\_function, [536](#)
  - GiNaC::print\_context, [823](#)
- order
  - integration\_kernel.cpp, [1067](#)
- Order\_conjugate
  - GiNaC, [154](#)
- Order\_eval
  - GiNaC, [154](#)
- Order\_expl\_derivative
  - GiNaC, [155](#)
- Order\_imag\_part
  - GiNaC, [155](#)
- Order\_real\_part
  - GiNaC, [154](#)
- Order\_series
  - GiNaC, [154](#)
- orig
  - GiNaC::symminfo, [961](#)
  - GiNaC::terminfo, [973](#)
- overall\_coeff
  - GiNaC::expairseq, [535](#)
- overflow
  - GiNaC::basic\_multi\_iterator, [378](#)
- overloaded
  - GiNaC::function\_options, [600](#)
- P
  - GiNaC::modular\_form\_kernel, [702](#)
- p
  - GiNaC::ptr, [867](#)
- parameter\_set
  - GiNaC::fderivative, [546](#)
- paramset
  - GiNaC, [60](#)
- parent
  - GiNaC::class\_info, [385](#)
- parent\_name
  - GiNaC::print\_context\_options, [825](#)
  - GiNaC::registered\_class\_options, [876](#)
- parents\_identified
  - GiNaC::class\_info, [385](#)
- partition
  - GiNaC::partition\_generator, [787](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [790](#)
- partition\_generator
  - GiNaC::partition\_generator, [787](#)
- partition\_with\_zero\_parts\_generator
  - GiNaC::partition\_with\_zero\_parts\_generator, [789](#)
- pderivative
  - GiNaC::function, [559](#)
- permutation\_sign
  - GiNaC, [282](#), [283](#)
- permute\_free\_index\_to\_front
  - GiNaC, [106](#)

- Pi
  - GiNaC, [288](#)
- PiEvalf
  - GiNaC, [244](#)
- pivot
  - GiNaC::matrix, [692](#)
- point
  - GiNaC::pseries, [861](#)
- pointer\_to\_map\_function
  - GiNaC::pointer\_to\_map\_function, [791](#)
- pointer\_to\_map\_function\_1arg
  - GiNaC::pointer\_to\_map\_function\_1arg, [792](#)
- pointer\_to\_map\_function\_2args
  - GiNaC::pointer\_to\_map\_function\_2args, [793](#)
- pointer\_to\_map\_function\_3args
  - GiNaC::pointer\_to\_map\_function\_3args, [795](#)
- pointer\_to\_member\_to\_map\_function
  - GiNaC::pointer\_to\_member\_to\_map\_function, [797](#)
- pointer\_to\_member\_to\_map\_function\_1arg
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
1arg, [799](#)
- pointer\_to\_member\_to\_map\_function\_2args
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
2args, [801](#)
- pointer\_to\_member\_to\_map\_function\_3args
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↔  
3args, [803](#)
- pole\_error
  - GiNaC::pole\_error, [805](#)
- poly
  - factor.cpp, [1027](#)
- pos\_sig
  - GiNaC::minkmetric, [698](#)
  - GiNaC::tensepsilon, [968](#)
- possymbol
  - GiNaC::possymbol, [807](#)
- postorder\_begin
  - GiNaC::ex, [487](#)
- postorder\_end
  - GiNaC::ex, [487](#)
- pow
  - GiNaC::matrix, [684](#)
  - GiNaC, [245](#), [263](#)
- power
  - GiNaC::add, [327](#)
  - GiNaC::function, [560](#)
  - GiNaC::mul, [720](#)
  - GiNaC::ncmul, [755](#)
  - GiNaC::numeric, [771](#)
  - GiNaC::power, [810](#)
- power.cpp, [1089](#)
- power.h, [1090](#)
- power\_const
  - GiNaC::pseries, [859](#)
- power\_dyn
  - GiNaC::numeric, [773](#)
- power\_f
  - GiNaC::function\_options, [604](#)
- power\_func
  - GiNaC::function\_options, [588–590](#), [596](#)
- power\_funcp
  - GiNaC, [61](#)
- power\_funcp\_1
  - GiNaC, [63](#)
- power\_funcp\_10
  - GiNaC, [78](#)
- power\_funcp\_11
  - GiNaC, [79](#)
- power\_funcp\_12
  - GiNaC, [81](#)
- power\_funcp\_13
  - GiNaC, [83](#)
- power\_funcp\_14
  - GiNaC, [85](#)
- power\_funcp\_2
  - GiNaC, [64](#)
- power\_funcp\_3
  - GiNaC, [66](#)
- power\_funcp\_4
  - GiNaC, [67](#)
- power\_funcp\_5
  - GiNaC, [69](#)
- power\_funcp\_6
  - GiNaC, [71](#)
- power\_funcp\_7
  - GiNaC, [73](#)
- power\_funcp\_8
  - GiNaC, [74](#)
- power\_funcp\_9
  - GiNaC, [76](#)
- power\_funcp\_exvector
  - GiNaC, [87](#)
- power\_use\_exvector\_args
  - GiNaC::function\_options, [607](#)
- precedence
  - GiNaC::add, [318](#)
  - GiNaC::basic, [355](#)
  - GiNaC::clifford, [389](#)
  - GiNaC::container, [431](#)
  - GiNaC::expairseq, [525](#)
  - GiNaC::function, [554](#)
  - GiNaC::indexed, [632](#)
  - GiNaC::integral, [642](#)
  - GiNaC::mul, [708](#)
  - GiNaC::ncmul, [749](#)
  - GiNaC::numeric, [762](#)
  - GiNaC::power, [810](#)
  - GiNaC::pseries, [850](#)
  - GiNaC::relational, [879](#)
  - GiNaC::structure, [916](#)
- prem
  - GiNaC, [214](#)
- preorder\_begin
  - GiNaC::ex, [487](#)
- preorder\_end
  - GiNaC::ex, [487](#)



- prepend
  - GiNaC::container, [436](#)
- primitive\_dirichlet\_character
  - GiNaC, [196](#)
- primpart
  - GiNaC::ex, [503](#), [504](#)
- print
  - GiNaC::\_numeric\_digits, [314](#)
  - GiNaC::basic, [354](#)
  - GiNaC::ex, [488](#)
  - GiNaC::expair, [518](#)
  - GiNaC::fderivative, [542](#)
  - GiNaC::function, [553](#)
  - GiNaC::structure, [915](#)
- print.cpp, [1090](#)
- print.h, [1091](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE, [1092](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT\_COMM↔ON, [1092](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT, [1093](#)
  - GINAC\_IMPLEMENT\_PRINT\_CONTEXT, [1093](#)
- print\_add
  - GiNaC::add, [326](#)
- print\_context
  - GiNaC::print\_context, [823](#)
- print\_context\_class\_info
  - GiNaC, [89](#)
- print\_context\_options
  - GiNaC::print\_context\_options, [824](#)
- print\_csrc
  - GiNaC::print\_csrc, [827](#)
- print\_csrc\_cl\_N
  - GiNaC::print\_csrc\_cl\_N, [827](#)
- print\_csrc\_double
  - GiNaC::print\_csrc\_double, [828](#)
- print\_csrc\_float
  - GiNaC::print\_csrc\_float, [829](#)
- print\_dflt
  - GiNaC::print\_dflt, [830](#)
- print\_dispatch
  - GiNaC::basic, [367](#), [368](#)
- print\_dispatch\_table
  - GiNaC::function\_options, [604](#)
  - GiNaC::registered\_class\_options, [876](#)
- print\_elements
  - GiNaC::matrix, [693](#)
- print\_func
  - GiNaC::function\_options, [596–599](#)
  - GiNaC::registered\_class\_options, [875](#)
- print\_func< print\_context >
  - GiNaC, [128](#)
- print\_func< print\_dflt >
  - GiNaC, [93](#), [105](#), [276](#)
- print\_funcp
  - GiNaC, [62](#)
- print\_funcp\_1
  - GiNaC, [63](#)
- print\_funcp\_10
  - GiNaC, [78](#)
- print\_funcp\_11
  - GiNaC, [80](#)
- print\_funcp\_12
  - GiNaC, [82](#)
- print\_funcp\_13
  - GiNaC, [84](#)
- print\_funcp\_14
  - GiNaC, [86](#)
- print\_funcp\_2
  - GiNaC, [65](#)
- print\_funcp\_3
  - GiNaC, [66](#)
- print\_funcp\_4
  - GiNaC, [68](#)
- print\_funcp\_5
  - GiNaC, [69](#)
- print\_funcp\_6
  - GiNaC, [71](#)
- print\_funcp\_7
  - GiNaC, [73](#)
- print\_funcp\_8
  - GiNaC, [75](#)
- print\_funcp\_9
  - GiNaC, [76](#)
- print\_funcp\_exvector
  - GiNaC, [87](#)
- print\_functor
  - GiNaC::print\_functor, [831](#), [832](#)
- print\_index
  - GiNaC::idx, [622](#)
- print\_indexed
  - GiNaC::indexed, [637](#)
- print\_integer\_csrc
  - GiNaC, [227](#)
- print\_latex
  - GiNaC::print\_latex, [835](#)
- print\_memfun\_handler
  - GiNaC::print\_memfun\_handler, [836](#)
- print\_numeric
  - GiNaC::numeric, [783](#)
- print\_operator
  - GiNaC, [266](#)
- print\_overall\_coeff
  - GiNaC::mul, [718](#)
- print\_power
  - GiNaC::power, [818](#)
- print\_ptrfun\_handler
  - GiNaC::print\_ptrfun\_handler, [839](#)
- print\_python
  - GiNaC::print\_python, [841](#)
- print\_python\_repr
  - GiNaC::print\_python\_repr, [842](#)
- print\_real\_cl\_N
  - GiNaC, [229](#)
- print\_real\_csrc
  - GiNaC, [228](#)



- print\_real\_number
  - GiNaC, [227](#)
- print\_series
  - GiNaC::pseries, [859](#)
- print\_sym\_pow
  - GiNaC, [262](#)
- print\_tree
  - GiNaC::print\_tree, [842](#), [843](#)
- print\_use\_exvector\_args
  - GiNaC::function\_options, [607](#)
- printindices
  - GiNaC::indexed, [636](#)
- printpair
  - GiNaC::expairseq, [530](#)
- printraw
  - GiNaC::archive, [333](#)
  - GiNaC::archive\_node, [343](#)
- printseq
  - GiNaC::container, [435](#)
  - GiNaC::expairseq, [530](#)
- product\_to\_exvector
  - GiNaC, [134](#)
- property
  - GiNaC::archive\_node::property, [844](#)
- property\_info
  - GiNaC::archive\_node::property\_info, [845](#), [846](#)
- property\_type
  - GiNaC::archive\_node, [338](#)
- propinfovector
  - GiNaC::archive\_node, [337](#)
- props
  - GiNaC::archive\_node, [344](#)
- pseries
  - GiNaC::pseries, [849](#)
- pseries.cpp, [1094](#)
- pseries.h, [1094](#)
- psi
  - GiNaC, [157](#), [237](#), [238](#)
- psi1\_deriv
  - GiNaC, [167](#)
- psi1\_eval
  - GiNaC, [167](#)
- psi1\_evalf
  - GiNaC, [166](#)
- psi1\_series
  - GiNaC, [167](#)
- psi2\_deriv
  - GiNaC, [168](#)
- psi2\_eval
  - GiNaC, [168](#)
- psi2\_evalf
  - GiNaC, [167](#)
- psi2\_series
  - GiNaC, [168](#)
- ptr
  - GiNaC::pointer\_to\_map\_function, [791](#)
  - GiNaC::pointer\_to\_map\_function\_1arg, [792](#)
  - GiNaC::pointer\_to\_map\_function\_2args, [794](#)
  - GiNaC::pointer\_to\_map\_function\_3args, [796](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function, [798](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↵  
1arg, [799](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↵  
2args, [801](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_↵  
3args, [804](#)
  - GiNaC::ptr, [864](#)
- ptr.h, [1095](#)
- python
  - GiNaC, [260](#)
- python\_repr
  - GiNaC, [261](#)
- q\_expansion\_modular\_form
  - GiNaC::Eisenstein\_h\_kernel, [464](#)
  - GiNaC::Eisenstein\_kernel, [469](#)
  - GiNaC::modular\_form\_kernel, [702](#)
- qbar
  - integration\_kernel.cpp, [1067](#)
- quo
  - GiNaC, [213](#)
- R
  - factor.cpp, [1028](#)
- r
  - factor.cpp, [1023](#)
  - GiNaC::Eisenstein\_h\_kernel, [465](#)
- REGISTER\_FUNCTION
  - function.h, [1044](#)
  - GiNaC, [139](#), [141](#), [142](#), [144](#), [146](#), [147](#), [149–152](#),  
[154](#), [155](#), [160](#), [161](#), [164–166](#), [171–173](#), [176](#),  
[178](#), [180](#), [181](#), [183–189](#), [191–194](#)
- rank
  - GiNaC::matrix, [689](#), [690](#)
  - GiNaC, [208](#)
- rbegin
  - GiNaC::container, [438](#)
- read\_archive
  - GiNaC::basic, [368](#)
  - GiNaC::clifford, [389](#)
  - GiNaC::color, [399](#)
  - GiNaC::constant, [422](#)
  - GiNaC::container, [432](#)
  - GiNaC::expairseq, [528](#)
  - GiNaC::fderivative, [543](#)
  - GiNaC::function, [557](#)
  - GiNaC::idx, [618](#)
  - GiNaC::indexed, [633](#)
  - GiNaC::integral, [646](#)
  - GiNaC::matrix, [681](#)
  - GiNaC::minkmetric, [697](#)
  - GiNaC::numeric, [769](#)
  - GiNaC::power, [817](#)
  - GiNaC::pseries, [855](#)
  - GiNaC::relational, [881](#)
  - GiNaC::spinidx, [903](#)
  - GiNaC::symbol, [946](#)

- GiNaC::symmetry, [955](#)
  - GiNaC::tensepsilon, [967](#)
  - GiNaC::varidx, [983](#)
  - GiNaC::wildcard, [988](#)
- read\_real\_float
  - GiNaC, [226](#)
- read\_unsigned
  - GiNaC, [90](#)
- real
  - GiNaC::numeric, [782](#)
  - GiNaC, [250](#)
- real\_part
  - GiNaC::add, [323](#)
  - GiNaC::basic, [366](#)
  - GiNaC::constant, [422](#)
  - GiNaC::container, [433](#)
  - GiNaC::ex, [492](#)
  - GiNaC::function, [556](#)
  - GiNaC::indexed, [632](#)
  - GiNaC::matrix, [681](#)
  - GiNaC::mul, [711](#)
  - GiNaC::ncmul, [752](#)
  - GiNaC::numeric, [768](#)
  - GiNaC::power, [816](#)
  - GiNaC::pseries, [854](#)
  - GiNaC::realsymbol, [869](#)
  - GiNaC::symbol, [945](#)
  - GiNaC, [113](#)
- real\_part\_conjugate
  - GiNaC, [140](#)
- real\_part\_eval
  - GiNaC, [140](#)
- real\_part\_evalf
  - GiNaC, [139](#)
- real\_part\_expl\_derivative
  - GiNaC, [140](#)
- real\_part\_f
  - GiNaC::function\_options, [603](#)
- real\_part\_func
  - GiNaC::function\_options, [576–578](#), [595](#)
- real\_part\_funcp
  - GiNaC, [61](#)
- real\_part\_funcp\_1
  - GiNaC, [62](#)
- real\_part\_funcp\_10
  - GiNaC, [77](#)
- real\_part\_funcp\_11
  - GiNaC, [79](#)
- real\_part\_funcp\_12
  - GiNaC, [80](#)
- real\_part\_funcp\_13
  - GiNaC, [82](#)
- real\_part\_funcp\_14
  - GiNaC, [84](#)
- real\_part\_funcp\_2
  - GiNaC, [64](#)
- real\_part\_funcp\_3
  - GiNaC, [65](#)
- real\_part\_funcp\_4
  - GiNaC, [67](#)
- real\_part\_funcp\_5
  - GiNaC, [68](#)
- real\_part\_funcp\_6
  - GiNaC, [70](#)
- real\_part\_funcp\_7
  - GiNaC, [72](#)
- real\_part\_funcp\_8
  - GiNaC, [74](#)
- real\_part\_funcp\_9
  - GiNaC, [75](#)
- real\_part\_funcp\_exvector
  - GiNaC, [86](#)
- real\_part\_imag\_part
  - GiNaC, [140](#)
- real\_part\_print\_latex
  - GiNaC, [140](#)
- real\_part\_real\_part
  - GiNaC, [140](#)
- real\_part\_use\_exvector\_args
  - GiNaC::function\_options, [606](#)
- realsymbol
  - GiNaC::realsymbol, [869](#)
- recombine\_pair\_to\_ex
  - GiNaC::add, [325](#)
  - GiNaC::expairseq, [531](#)
  - GiNaC::mul, [716](#)
- reduced\_matrix
  - GiNaC, [205](#)
- reeval\_ncmul
  - GiNaC::ncmul, [755](#)
  - GiNaC, [210](#)
- refcount
  - GiNaC::refcounted, [873](#)
- refcounted
  - GiNaC::refcounted, [872](#)
- register\_new
  - GiNaC::function, [560](#)
- registered\_class\_info
  - GiNaC, [89](#)
- registered\_class\_options
  - GiNaC::registered\_class\_options, [874](#)
- registered\_functions
  - GiNaC::function, [560](#)
- registrar.cpp, [1096](#)
- registrar.h, [1096](#)
  - GINAC\_DECLARE\_REGISTERED\_CLASS\_CO↔  
MMON, [1097](#)
  - GINAC\_DECLARE\_REGISTERED\_CLASS\_NO↔  
\_CTORS, [1098](#)
  - GINAC\_DECLARE\_REGISTERED\_CLASS, [1098](#)
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS↔  
OPT\_T, [1099](#)
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS↔  
OPT, [1099](#)
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS,  
[1099](#)

- relational
  - GiNaC::relational, [879](#)
- relational.cpp, [1100](#)
- relational.h, [1100](#)
- relative\_integration\_error
  - GiNaC::integral, [648](#)
- rem
  - GiNaC, [213](#)
- remember
  - GiNaC::function\_options, [600](#)
- remember.cpp, [1101](#)
- remember.h, [1101](#)
- remember\_assoc\_size
  - GiNaC::function\_options, [605](#)
- remember\_size
  - GiNaC::function\_options, [605](#)
- remember\_strategy
  - GiNaC::function\_options, [606](#)
  - GiNaC::remember\_table, [888](#)
  - GiNaC::remember\_table\_list, [893](#)
- remember\_table
  - GiNaC::remember\_table, [886](#)
- remember\_table\_entry
  - GiNaC::function, [562](#)
  - GiNaC::remember\_table\_entry, [889](#)
- remember\_table\_list
  - GiNaC::remember\_table\_list, [892](#)
- remember\_tables
  - GiNaC::remember\_table, [887](#)
- remove\_all
  - GiNaC::container, [437](#)
- remove\_dirac\_ONE
  - GiNaC, [99](#)
- remove\_first
  - GiNaC::container, [436](#)
- remove\_last
  - GiNaC::container, [437](#)
- remove\_reference
  - GiNaC::refcounted, [872](#)
- rename\_dummy\_indices
  - GiNaC, [133](#)
- rename\_dummy\_indices\_uniquely
  - GiNaC, [136](#), [137](#)
- rend
  - GiNaC::container, [438](#)
- replace\_contr\_index
  - GiNaC::tensor, [972](#)
- replace\_dim
  - GiNaC::idx, [621](#)
- replace\_with\_symbol
  - GiNaC, [223](#), [224](#)
- reposition\_dummy\_indices
  - GiNaC::indexed, [638](#)
  - GiNaC, [134](#)
- representation\_label
  - GiNaC::clifford, [394](#)
  - GiNaC::color, [401](#)
- reserve
  - GiNaC::container\_storage, [442](#)
- rest
  - GiNaC::expair, [519](#)
- result
  - GiNaC::remember\_table\_entry, [891](#)
- result\_type
  - GiNaC::map\_function, [673](#)
- resultant
  - GiNaC, [225](#)
- return\_type
  - GiNaC::add, [324](#)
  - GiNaC::basic, [365](#)
  - GiNaC::clifford, [391](#)
  - GiNaC::color, [400](#)
  - GiNaC::ex, [509](#)
  - GiNaC::expairseq, [528](#)
  - GiNaC::fail, [538](#)
  - GiNaC::function, [559](#)
  - GiNaC::function\_options, [605](#)
  - GiNaC::indexed, [634](#)
  - GiNaC::integral, [645](#)
  - GiNaC::matrix, [682](#)
  - GiNaC::minkmetric, [697](#)
  - GiNaC::mul, [714](#)
  - GiNaC::ncmul, [753](#)
  - GiNaC::power, [818](#)
  - GiNaC::relational, [881](#)
  - GiNaC::structure, [925](#)
  - GiNaC::su3d, [929](#)
  - GiNaC::su3f, [931](#)
  - GiNaC::tensdelta, [964](#)
  - GiNaC::tensepsilon, [967](#)
  - GiNaC::tensmetric, [970](#)
  - GiNaC::tensor, [972](#)
- return\_type\_tinfo
  - GiNaC::add, [324](#)
  - GiNaC::basic, [366](#)
  - GiNaC::clifford, [391](#)
  - GiNaC::color, [400](#)
  - GiNaC::ex, [510](#)
  - GiNaC::function, [559](#)
  - GiNaC::function\_options, [605](#)
  - GiNaC::indexed, [635](#)
  - GiNaC::integral, [645](#)
  - GiNaC::mul, [715](#)
  - GiNaC::ncmul, [753](#)
  - GiNaC::power, [818](#)
  - GiNaC::relational, [882](#)
  - GiNaC::structure, [926](#)
- rh
  - GiNaC::relational, [884](#)
- rhs
  - GiNaC::ex, [492](#)
  - GiNaC::relational, [883](#)
  - GiNaC, [120](#)
- rl
  - GiNaC::return\_type\_t, [895](#)
- root

- GiNaC::archive::archived\_ex, 347
- rotate\_left
  - GiNaC, 281
- row
  - GiNaC::matrix, 694
- rows
  - GiNaC::matrix, 682
  - GiNaC, 206
- s
  - GiNaC::Eisenstein\_h\_kernel, 465
  - GiNaC::const\_postorder\_iterator, 416
  - GiNaC::const\_preorder\_iterator, 418
  - GiNaC::derivative\_map\_function, 446
  - GiNaC::print\_context, 823
  - GiNaC::symbolset, 952
- S\_deriv
  - GiNaC, 171
- S\_eval
  - GiNaC, 171
- S\_evalf
  - GiNaC, 171
- S\_print\_latex
  - GiNaC, 172
- S\_series
  - GiNaC, 171
- STATISTICS
  - normal.cpp, 1078
- STLT
  - GiNaC::container, 429
  - GiNaC::container\_storage, 441
- safe\_bool
  - GiNaC::relational, 878
- same\_metric
  - GiNaC::clifford, 392
- scalar\_mul\_indexed
  - GiNaC::basic, 364
  - GiNaC::matrix, 680
  - GiNaC::structure, 924
- seq
  - GiNaC::container\_storage, 443
  - GiNaC::expairseq, 534
  - GiNaC::pseries, 861
  - GiNaC::remember\_table\_entry, 890
- serial
  - GiNaC::G2\_SERIAL, 609
  - GiNaC::G3\_SERIAL, 609
  - GiNaC::constant, 425
  - GiNaC::function, 562
  - GiNaC::iterated\_integral2\_SERIAL, 656
  - GiNaC::iterated\_integral3\_SERIAL, 657
  - GiNaC::psi1\_SERIAL, 862
  - GiNaC::psi2\_SERIAL, 863
  - GiNaC::symbol, 950
  - GiNaC::zeta1\_SERIAL, 990
  - GiNaC::zeta2\_SERIAL, 991
- series
  - GiNaC::Eisenstein\_h\_kernel, 461
  - GiNaC::Eisenstein\_kernel, 467
  - GiNaC::add, 321
  - GiNaC::basic, 361
  - GiNaC::ex, 499
  - GiNaC::fderivative, 542
  - GiNaC::function, 555
  - GiNaC::integral, 647
  - GiNaC::integration\_kernel, 651
  - GiNaC::modular\_form\_kernel, 700
  - GiNaC::mul, 712
  - GiNaC::power, 814
  - GiNaC::pseries, 852
  - GiNaC::structure, 921
  - GiNaC::symbol, 944
  - GiNaC, 118
- series\_coeff
  - GiNaC::integration\_kernel, 653
- series\_coeff\_impl
  - GiNaC::ELi\_kernel, 474
  - GiNaC::Ebar\_kernel, 458
  - GiNaC::Kronecker\_dtau\_kernel, 660
  - GiNaC::Kronecker\_dz\_kernel, 664
  - GiNaC::basic\_log\_kernel, 375
  - GiNaC::integration\_kernel, 653
  - GiNaC::multiple\_polylog\_kernel, 746
- series\_f
  - GiNaC::function\_options, 604
- series\_func
  - GiNaC::function\_options, 590–592, 596
- series\_funcp
  - GiNaC, 61
- series\_funcp\_1
  - GiNaC, 63
- series\_funcp\_10
  - GiNaC, 78
- series\_funcp\_11
  - GiNaC, 80
- series\_funcp\_12
  - GiNaC, 81
- series\_funcp\_13
  - GiNaC, 83
- series\_funcp\_14
  - GiNaC, 85
- series\_funcp\_2
  - GiNaC, 64
- series\_funcp\_3
  - GiNaC, 66
- series\_funcp\_4
  - GiNaC, 68
- series\_funcp\_5
  - GiNaC, 69
- series\_funcp\_6
  - GiNaC, 71
- series\_funcp\_7
  - GiNaC, 73
- series\_funcp\_8
  - GiNaC, 74
- series\_funcp\_9
  - GiNaC, 76

- series\_funcp\_exvector
  - GiNaC, [87](#)
- series\_to\_poly
  - GiNaC, [264](#)
- series\_use\_exvector\_args
  - GiNaC::function\_options, [607](#)
- series\_vec
  - GiNaC::integration\_kernel, [654](#)
- set
  - GiNaC::matrix, [686](#)
- set\_TeX\_name
  - GiNaC::symbol, [948](#)
- set\_cache\_step
  - GiNaC::integration\_kernel, [653](#)
- set\_name
  - GiNaC::function\_options, [569](#)
  - GiNaC::symbol, [948](#)
- set\_print\_context
  - GiNaC, [259](#)
- set\_print\_func
  - GiNaC::function\_options, [601](#)
  - GiNaC::registered\_class\_options, [875](#)
  - GiNaC, [265](#)
- set\_print\_options
  - GiNaC, [259](#)
- set\_refcount
  - GiNaC::refcounted, [872](#)
- set\_return\_type
  - GiNaC::function\_options, [599](#)
- set\_symmetry
  - GiNaC::function\_options, [600](#)
- set\_type
  - GiNaC::symmetry, [956](#)
- setflag
  - GiNaC::basic, [371](#)
- shaker\_sort
  - GiNaC, [283](#)
- share
  - GiNaC::ex, [512](#)
- shift\_exponents
  - GiNaC::pseries, [859](#)
- show\_statistics
  - GiNaC::remember\_table, [887](#)
- simplify\_indexed
  - GiNaC::ex, [506](#)
  - GiNaC::indexed, [638](#)
  - GiNaC, [118](#), [119](#), [134](#)
- simplify\_indexed\_product
  - GiNaC::indexed, [638](#)
  - GiNaC, [135](#)
- sin
  - GiNaC, [230](#)
- sin\_conjugate
  - GiNaC, [179](#)
- sin\_deriv
  - GiNaC, [179](#)
- sin\_eval
  - GiNaC, [179](#)
- sin\_evalf
  - GiNaC, [178](#)
- sin\_imag\_part
  - GiNaC, [179](#)
- sin\_real\_part
  - GiNaC, [179](#)
- sinh
  - GiNaC, [233](#)
- sinh\_conjugate
  - GiNaC, [188](#)
- sinh\_deriv
  - GiNaC, [187](#)
- sinh\_eval
  - GiNaC, [187](#)
- sinh\_evalf
  - GiNaC, [187](#)
- sinh\_imag\_part
  - GiNaC, [188](#)
- sinh\_real\_part
  - GiNaC, [187](#)
- size
  - GiNaC::basic\_multi\_iterator, [378](#)
- smod
  - GiNaC::add, [322](#)
  - GiNaC::basic, [363](#)
  - GiNaC::ex, [505](#)
  - GiNaC::mul, [713](#)
  - GiNaC::numeric, [767](#)
  - GiNaC::structure, [923](#)
  - GiNaC, [241](#)
- solve
  - GiNaC::matrix, [689](#)
- sort
  - GiNaC::container, [437](#)
- sort\_
  - GiNaC::container, [435](#), [436](#)
- spinidx
  - GiNaC::spinidx, [902](#)
- spinor\_metric
  - GiNaC, [278](#)
- split\_ex\_to\_pair
  - GiNaC::add, [325](#)
  - GiNaC::expairseq, [530](#)
  - GiNaC::mul, [715](#)
- spm
  - GiNaC::scalar\_products, [899](#)
- spmap
  - GiNaC, [88](#)
- spmapkey
  - GiNaC::spmapkey, [909](#)
- sprem
  - GiNaC, [215](#)
- sqrfree
  - GiNaC, [222](#)
- sqrfree\_parfrac
  - GiNaC, [223](#)
- sqrfree\_yun
  - GiNaC, [221](#)

- sqrt
  - GiNaC, 244, 264
- sr\_gcd
  - GiNaC, 217
- std, 308
  - swap, 309
- std::equal\_to< GiNaC::ex >, 476
  - operator(), 476
- std::hash< GiNaC::ex >, 613
  - operator(), 614
- std::less< GiNaC::ptr< T > >, 667
  - operator(), 667
- std::less< ptr< T > >
  - GiNaC::ptr, 866
- step
  - GiNaC::numeric, 774
  - GiNaC, 246
- step\_conjugate
  - GiNaC, 145
- step\_eval
  - GiNaC, 145
- step\_evalf
  - GiNaC, 145
- step\_imag\_part
  - GiNaC, 146
- step\_real\_part
  - GiNaC, 145
- step\_series
  - GiNaC, 145
- store\_remember\_table
  - GiNaC::function, 560
- struct\_compare
  - GiNaC::compare\_all\_equal, 402
  - GiNaC::compare\_bitwise, 403
  - GiNaC::compare\_std\_less, 405
- struct\_is\_equal
  - GiNaC::compare\_all\_equal, 402
  - GiNaC::compare\_bitwise, 403
  - GiNaC::compare\_std\_less, 404
- structure
  - GiNaC::structure, 914
- structure.h, 1102
- sub
  - GiNaC::matrix, 683
  - GiNaC::numeric, 770
- sub\_dyn
  - GiNaC::numeric, 772
- sub\_matrix
  - GiNaC, 205
- subs
  - GiNaC::basic, 358
  - GiNaC::clifford, 393
  - GiNaC::container, 432
  - GiNaC::ex, 494, 495
  - GiNaC::expairseq, 527
  - GiNaC::idx, 618
  - GiNaC::matrix, 679
  - GiNaC::numeric, 765
  - GiNaC::power, 814
  - GiNaC::pseries, 852
  - GiNaC::relational, 880
  - GiNaC::structure, 919
  - GiNaC::symbol, 944
  - GiNaC, 121, 122
- subs\_one\_level
  - GiNaC::basic, 369
- subschildren
  - GiNaC::container, 439
  - GiNaC::expairseq, 534
- subvalue
  - GiNaC, 195
- successful\_hits
  - GiNaC::remember\_table\_entry, 891
- sufficiently\_accurate
  - GiNaC::lanczos\_coeffs, 666
- swap
  - GiNaC::ex, 486
  - GiNaC::expair, 519
  - GiNaC::ptr, 865
  - GiNaC, 121, 125
  - std, 309
- swapped
  - GiNaC::sy\_swap, 937
- sy\_anti
  - GiNaC, 273, 274
- sy\_cycl
  - GiNaC, 274, 275
- sy\_is\_less
  - GiNaC::sy\_is\_less, 936
  - GiNaC::symmetry, 958
- sy\_none
  - GiNaC, 272
- sy\_swap
  - GiNaC::sy\_swap, 937
  - GiNaC::symmetry, 958
- sy\_symm
  - GiNaC, 273
- sym
  - GiNaC::sym\_desc, 939
- sym\_desc
  - GiNaC::sym\_desc, 939
- sym\_desc\_vec
  - GiNaC, 88
- symbol
  - GiNaC::symbol, 942
- symbol.cpp, 1103
- symbol.h, 1103
- symbolic\_matrix
  - GiNaC, 204, 208
- symbolset
  - GiNaC::symbolset, 951
- symm
  - GiNaC::terminfo, 973
  - GiNaC, 271
- symmetric2
  - GiNaC, 269

- symmetric3
  - GiNaC, 269
- symmetric4
  - GiNaC, 269
- symmetrize
  - GiNaC::ex, 508
  - GiNaC, 119, 271, 275
- symmetrize\_cyclic
  - GiNaC::ex, 509
  - GiNaC, 120, 271, 275
- symmetry
  - GiNaC::symmetry, 954
- symmetry.cpp, 1104
- symmetry.h, 1105
- symmetry\_type
  - GiNaC::symmetry, 954
- symminfo
  - GiNaC::symminfo, 960
- symmterm
  - GiNaC::symminfo, 960
- syms
  - factor.cpp, 1028
- symtree
  - GiNaC::function\_options, 608
  - GiNaC::indexed, 639
- synthesize\_func
  - GiNaC, 58
- TEST\_PERMUTATION
  - color.cpp, 1007
- table\_size
  - GiNaC::remember\_table, 888
- tan
  - GiNaC, 231
- tan\_conjugate
  - GiNaC, 182
- tan\_deriv
  - GiNaC, 182
- tan\_eval
  - GiNaC, 181
- tan\_evalf
  - GiNaC, 181
- tan\_imag\_part
  - GiNaC, 182
- tan\_real\_part
  - GiNaC, 182
- tan\_series
  - GiNaC, 182
- tanh
  - GiNaC, 233
- tanh\_conjugate
  - GiNaC, 191
- tanh\_deriv
  - GiNaC, 190
- tanh\_eval
  - GiNaC, 190
- tanh\_evalf
  - GiNaC, 190
- tanh\_imag\_part
  - GiNaC, 191
- tanh\_real\_part
  - GiNaC, 190
- tanh\_series
  - GiNaC, 190
- tau
  - GiNaC::Kronecker\_dz\_kernel, 665
- tcoeff
  - GiNaC::ex, 498
- TeX\_name
  - GiNaC::constant, 425
  - GiNaC::function\_options, 602
  - GiNaC::symbol, 950
- tensepsilon
  - GiNaC::tensepsilon, 966
- tensor
  - GiNaC, 288
- tensor.cpp, 1107
- tensor.h, 1108
- terminfo
  - GiNaC::terminfo, 973
- test
  - GiNaC::has\_distance, 612
- test\_and\_set\_nparams
  - GiNaC::function\_options, 601
- tgamma
  - GiNaC, 237
- tgamma\_conjugate
  - GiNaC, 165
- tgamma\_deriv
  - GiNaC, 165
- tgamma\_eval
  - GiNaC, 164
- tgamma\_evalf
  - GiNaC, 164
- tgamma\_series
  - GiNaC, 165
- thiscontainer
  - GiNaC::clifford, 390, 391
  - GiNaC::color, 400
  - GiNaC::container, 434, 435
  - GiNaC::fderivative, 542, 543
  - GiNaC::function, 556
  - GiNaC::indexed, 634
  - GiNaC::ncmul, 752
- thisexpairseq
  - GiNaC::add, 324
  - GiNaC::expairseq, 529
  - GiNaC::mul, 715
- tinfo
  - GiNaC::return\_type\_t, 895
- tinfo\_key
  - GiNaC::registered\_class\_options, 876
- to\_cl\_N
  - GiNaC::numeric, 782
- to\_double
  - GiNaC::numeric, 782
- to\_double
  - GiNaC, 250

- to\_int
  - GiNaC::numeric, 781
  - GiNaC, 249
- to\_long
  - GiNaC::numeric, 781
  - GiNaC, 249
- to\_polynomial
  - GiNaC::basic, 362
  - GiNaC::ex, 500
  - GiNaC::expairseq, 526
  - GiNaC::numeric, 766
  - GiNaC::power, 816
  - GiNaC::structure, 922
  - GiNaC::symbol, 945
  - GiNaC, 116
- to\_rational
  - GiNaC::basic, 362
  - GiNaC::ex, 500
  - GiNaC::expairseq, 526
  - GiNaC::numeric, 766
  - GiNaC::power, 815
  - GiNaC::structure, 922
  - GiNaC::symbol, 945
  - GiNaC, 116
- toggle\_dot
  - GiNaC::spinidx, 905
- toggle\_variance
  - GiNaC::varidx, 984
- toggle\_variance\_dot
  - GiNaC::spinidx, 905
- too\_late
  - GiNaC::\_numeric\_digits, 315
- trace
  - GiNaC::matrix, 687
  - GiNaC, 207
- trace\_string
  - GiNaC, 97
- transpose
  - GiNaC::matrix, 686
  - GiNaC, 206
- traverse
  - GiNaC::ex, 496
- traverse\_postorder
  - GiNaC::ex, 496
- traverse\_preorder
  - GiNaC::ex, 496
- tree
  - GiNaC, 261
- tree\_node
  - GiNaC::class\_info::tree\_node, 975
- trivial
  - GiNaC::composition\_generator, 407
- tryfactsubs
  - GiNaC, 209
- type
  - GiNaC::archive\_node::property, 844
  - GiNaC::archive\_node::property\_info, 846
  - GiNaC::symmetry, 958
- USE\_REMEMBER
  - normal.cpp, 1078
- USE\_SAME\_DEGREE\_FACTOR
  - factor.cpp, 1022
- USE\_TRIAL\_DIVISION
  - normal.cpp, 1078
- uintvector
  - GiNaC, 88
- unarch\_map
  - GiNaC::unarchive\_table\_t, 977
- unarch\_table\_instance
  - GiNaC, 287
- unarchive
  - GiNaC::archive\_node, 342
- unarchive\_ex
  - GiNaC::archive, 330, 331
- unarchive\_map\_t
  - GiNaC, 59
- unarchive\_table\_t
  - GiNaC::unarchive\_table\_t, 976
- unatomize
  - GiNaC::archive, 333
- unique
  - GiNaC::container, 437
- unique\_
  - GiNaC::container, 436, 439
- unit
  - GiNaC::ex, 502
- unit\_matrix
  - GiNaC, 204, 208
- unitcontprim
  - GiNaC::ex, 504
- unlikely
  - compiler.h, 1010
- unlink\_ex
  - GiNaC, 125
- unsignedvector
  - GiNaC, 88
- use\_remember
  - GiNaC::function\_options, 605
- use\_return\_type
  - GiNaC::function\_options, 605
- usecount
  - GiNaC::unarchive\_table\_t, 977
- used\_indices
  - GiNaC::make\_flat\_inserter, 672
- user\_defined\_kernel
  - GiNaC::user\_defined\_kernel, 978
- uses\_Laurent\_series
  - GiNaC::Eisenstein\_h\_kernel, 463
  - GiNaC::Eisenstein\_kernel, 469
  - GiNaC::integration\_kernel, 652
  - GiNaC::modular\_form\_kernel, 701
  - GiNaC::user\_defined\_kernel, 980
- utils.cpp, 1109
- utils.h, 1111
  - DEFAULT\_COMPARE, 1113
  - DEFAULT\_CTOR, 1113



- DEFAULT\_PRINT\_LATEX, [1113](#)
  - DEFAULT\_PRINT, [1113](#)
- utils\_multi\_iterator.h, [1114](#)
- v
  - GiNaC::basic\_multi\_iterator, [380](#)
  - GiNaC::sy\_is\_less, [936](#)
  - GiNaC::sy\_swap, [937](#)
- v1
  - GiNaC::spmapkey, [910](#)
- v2
  - GiNaC::spmapkey, [910](#)
- v\_internal
  - GiNaC::multi\_iterator\_shuffle, [741](#)
- v\_orig
  - GiNaC::multi\_iterator\_shuffle, [741](#)
- validate
  - GiNaC::indexed, [637](#)
  - GiNaC::symmetry, [956](#)
- value
  - factor.cpp, [1023](#)
  - GiNaC::archive\_node::property, [844](#)
  - GiNaC::composition\_generator::coolmulti::element, [471](#)
  - GiNaC::idx, [623](#)
  - GiNaC::numeric, [785](#)
- var
  - GiNaC::pseries, [861](#)
- varidx
  - GiNaC::varidx, [982](#)
- version.h, [1116](#)
  - GINAC\_LT\_AGE, [1117](#)
  - GINAC\_LT\_CURRENT, [1117](#)
  - GINAC\_LT\_REVISION, [1117](#)
  - GINACLIB\_ARCHIVE\_AGE, [1117](#)
  - GINACLIB\_ARCHIVE\_VERSION, [1117](#)
  - GINACLIB\_MAJOR\_VERSION, [1116](#)
  - GINACLIB\_MICRO\_VERSION, [1117](#)
  - GINACLIB\_MINOR\_VERSION, [1117](#)
  - GINACLIB\_STR\_HELPER, [1118](#)
  - GINACLIB\_STR, [1118](#)
  - GINACLIB\_VERSION, [1118](#)
- version\_major
  - GiNaC, [290](#)
- version\_micro
  - GiNaC, [291](#)
- version\_minor
  - GiNaC, [290](#)
- wild
  - GiNaC, [287](#)
- wildcard
  - GiNaC::wildcard, [987](#)
- wildcard.cpp, [1118](#)
- wildcard.h, [1119](#)
- write\_real\_float
  - GiNaC, [227](#)
- write\_unsigned
  - GiNaC, [90](#)
- x
  - factor.cpp, [1027](#)
  - GiNaC::ELi\_kernel, [475](#)
  - GiNaC::Ebar\_kernel, [459](#)
  - GiNaC::basic\_partition\_generator::mpartition2, [704](#)
  - GiNaC::integral, [648](#)
  - GiNaC::user\_defined\_kernel, [980](#)
  - integration\_kernel.cpp, [1067](#)
- y
  - GiNaC::ELi\_kernel, [475](#)
  - GiNaC::Ebar\_kernel, [459](#)
- yes\_type
  - GiNaC::has\_distance, [612](#)
- z
  - GiNaC::Kronecker\_dtau\_kernel, [660](#)
  - GiNaC::multiple\_polylog\_kernel, [746](#)
- z\_j
  - GiNaC::Kronecker\_dz\_kernel, [664](#)
- zeta
  - GiNaC, [156](#), [236](#)
- zeta1\_deriv
  - GiNaC, [174](#)
- zeta1\_eval
  - GiNaC, [173](#)
- zeta1\_evalf
  - GiNaC, [173](#)
- zeta1\_print\_latex
  - GiNaC, [174](#)
- zeta2\_deriv
  - GiNaC, [174](#)
- zeta2\_eval
  - GiNaC, [174](#)
- zeta2\_evalf
  - GiNaC, [174](#)
- zeta2\_print\_latex
  - GiNaC, [175](#)
- zetaderiv\_deriv
  - GiNaC, [151](#)
- zetaderiv\_eval
  - GiNaC, [151](#)