

iteration 3

Generated by Doxygen 1.8.11

Contents

1	INDEX PAGE	1
1.1	Introduction	1
1.1.1	Team Name	1
1.1.2	TA Name	1
1.1.3	Discussion Section	1
1.1.4	Details	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9

6	Namespace Documentation	11
6.1	fcsl Namespace Reference	11
6.1.1	Detailed Description	11
6.2	fcsl::ast Namespace Reference	11
6.2.1	Enumeration Type Documentation	12
6.2.1.1	decType	12
6.3	fcsl::parser Namespace Reference	12
6.4	fcsl::scanner Namespace Reference	13
6.4.1	Typedef Documentation	14
6.4.1.1	TokenType	14
6.4.2	Enumeration Type Documentation	14
6.4.2.1	kTokenEnumType	14
6.4.3	Function Documentation	15
6.4.3.1	make_regex(const char *pattern)	15
6.4.3.2	match_regex(regex_t *re, const char *text)	16
6.4.3.3	ReadInput(int argc, char **argv)	16
6.4.3.4	ReadInputFromFile(const char *filename)	17
6.4.4	Variable Documentation	17
6.4.4.1	kRegexNSub	17
7	Class Documentation	19
7.1	fcsl::ast::binaryExpr Class Reference	19
7.1.1	Detailed Description	21
7.1.2	Constructor & Destructor Documentation	21
7.1.2.1	binaryExpr(std::string binaryOp, Expr *expr1, Expr *expr2)	21
7.1.3	Member Function Documentation	21
7.1.3.1	UnParse(void)	21
7.1.4	Member Data Documentation	21
7.1.4.1	binaryOp_	21
7.1.4.2	expr1_	22
7.1.4.3	expr2_	22

7.2	fcal::ast::boolExpr Class Reference	22
7.2.1	Detailed Description	24
7.2.2	Constructor & Destructor Documentation	24
7.2.2.1	boolExpr(std::string boolean, Expr *expr1, Expr *expr2)	24
7.2.3	Member Function Documentation	24
7.2.3.1	UnParse(void)	24
7.2.4	Member Data Documentation	24
7.2.4.1	boolean_	24
7.2.4.2	expr1_	25
7.2.4.3	expr2_	25
7.3	fcal::ast::bracketStmt Class Reference	25
7.3.1	Detailed Description	26
7.3.2	Constructor & Destructor Documentation	26
7.3.2.1	bracketStmt(Stmts *stmts)	26
7.3.3	Member Function Documentation	27
7.3.3.1	UnParse(void)	27
7.3.4	Member Data Documentation	27
7.3.4.1	stmts_	27
7.4	fcal::scanner::CharConstToken Class Reference	27
7.4.1	Detailed Description	30
7.4.2	Constructor & Destructor Documentation	30
7.4.2.1	CharConstToken(parser::Parser *p, Token *t)	30
7.4.3	Member Function Documentation	30
7.4.3.1	description()	30
7.4.3.2	nud()	30
7.5	fcal::ast::constantExpr Class Reference	31
7.5.1	Detailed Description	32
7.5.2	Constructor & Destructor Documentation	32
7.5.2.1	constantExpr(std::string const_)	32
7.5.3	Member Function Documentation	33

7.5.3.1	UnParse(void)	33
7.5.4	Member Data Documentation	33
7.5.4.1	constant_	33
7.6	fcal::scanner::DashToken Class Reference	33
7.6.1	Detailed Description	36
7.6.2	Constructor & Destructor Documentation	36
7.6.2.1	DashToken(parser::Parser *p, Token *t)	36
7.6.3	Member Function Documentation	36
7.6.3.1	description()	36
7.6.3.2	lbp()	36
7.6.3.3	led(parser::ParseResult left)	36
7.7	fcal::ast::Decl Class Reference	37
7.7.1	Detailed Description	38
7.8	fcal::ast::declStmt Class Reference	38
7.8.1	Detailed Description	40
7.8.2	Constructor & Destructor Documentation	40
7.8.2.1	declStmt(Decl *decl)	40
7.8.3	Member Function Documentation	41
7.8.3.1	UnParse(void)	41
7.8.4	Member Data Documentation	41
7.8.4.1	decl_	41
7.9	fcal::ast::emptyStmts Class Reference	41
7.9.1	Detailed Description	43
7.9.2	Constructor & Destructor Documentation	43
7.9.2.1	emptyStmts(void)	43
7.9.3	Member Function Documentation	44
7.9.3.1	UnParse(void)	44
7.10	fcal::scanner::EndOfFileToken Class Reference	44
7.10.1	Detailed Description	47
7.10.2	Constructor & Destructor Documentation	47

7.10.2.1	EndOfFileToken(parser::Parser *p, Token *t)	47
7.10.3	Member Function Documentation	47
7.10.3.1	description()	47
7.11	fcal::ast::equalsStmt Class Reference	47
7.11.1	Detailed Description	50
7.11.2	Constructor & Destructor Documentation	50
7.11.2.1	equalsStmt(varName *var, Expr *expr1, bool isMatrix)	50
7.11.2.2	equalsStmt(varName *var, Expr *expr1, Expr *expr2, Expr *expr3, bool isMatrix)	50
7.11.3	Member Function Documentation	50
7.11.3.1	UnParse(void)	51
7.11.4	Member Data Documentation	51
7.11.4.1	expr1_	51
7.11.4.2	expr2_	51
7.11.4.3	expr3_	51
7.11.4.4	isMatrix_	51
7.11.4.5	var_	51
7.12	fcal::ast::Expr Class Reference	52
7.12.1	Detailed Description	52
7.13	fcal::scanner::ExtToken Class Reference	53
7.13.1	Constructor & Destructor Documentation	55
7.13.1.1	ExtToken(parser::Parser *p, Token *t)	55
7.13.1.2	ExtToken(parser::Parser *p, Token *t, std::string d)	55
7.13.1.3	~ExtToken()	55
7.13.1.4	ExtToken(void)	55
7.13.2	Member Function Documentation	56
7.13.2.1	description()	56
7.13.2.2	ExtendToken(parser::Parser *p, Token *tokens)	56
7.13.2.3	ExtendTokenList(parser::Parser *p, Token *tokens)	57
7.13.2.4	lbp()	58
7.13.2.5	led(parser::ParseResult left)	58

7.13.2.6	lexeme(void) const	59
7.13.2.7	next(void) const	60
7.13.2.8	nud(void)	60
7.13.2.9	parser(void)	61
7.13.2.10	terminal(void) const	62
7.13.3	Member Data Documentation	62
7.13.3.1	desc_str_	62
7.13.3.2	lexeme_	62
7.13.3.3	next_	62
7.13.3.4	parser_	62
7.13.3.5	terminal_	62
7.14	fcal::scanner::FalseKwdToken Class Reference	63
7.14.1	Detailed Description	65
7.14.2	Constructor & Destructor Documentation	65
7.14.2.1	FalseKwdToken(parser::Parser *p, Token *t)	65
7.14.3	Member Function Documentation	65
7.14.3.1	description()	65
7.14.3.2	nud()	65
7.15	fcal::scanner::FloatConstToken Class Reference	65
7.15.1	Detailed Description	68
7.15.2	Constructor & Destructor Documentation	68
7.15.2.1	FloatConstToken(parser::Parser *p, Token *t)	68
7.15.3	Member Function Documentation	68
7.15.3.1	description()	68
7.15.3.2	nud()	68
7.16	fcal::scanner::ForwardSlashToken Class Reference	68
7.16.1	Detailed Description	71
7.16.2	Constructor & Destructor Documentation	71
7.16.2.1	ForwardSlashToken(parser::Parser *p, Token *t)	71
7.16.3	Member Function Documentation	71

7.16.3.1	description()	71
7.16.3.2	lbp()	71
7.16.3.3	led(parser::ParseResult left)	71
7.17	fcal::ast::ifElseStmt Class Reference	72
7.17.1	Detailed Description	73
7.17.2	Constructor & Destructor Documentation	74
7.17.2.1	ifElseStmt(Expr *expr, Stmt *stmt1, Stmt *stmt2)	74
7.17.3	Member Function Documentation	74
7.17.3.1	UnParse(void)	74
7.17.4	Member Data Documentation	74
7.17.4.1	expr_	74
7.17.4.2	stmt1_	74
7.17.4.3	stmt2_	75
7.18	fcal::ast::ifExpr Class Reference	75
7.18.1	Detailed Description	76
7.18.2	Constructor & Destructor Documentation	77
7.18.2.1	ifExpr(Expr *expr1, Expr *expr2, Expr *expr3)	77
7.18.3	Member Function Documentation	77
7.18.3.1	UnParse(void)	77
7.18.4	Member Data Documentation	77
7.18.4.1	expr1_	77
7.18.4.2	expr2_	77
7.18.4.3	expr3_	77
7.19	fcal::ast::ifStmt Class Reference	78
7.19.1	Detailed Description	79
7.19.2	Constructor & Destructor Documentation	80
7.19.2.1	ifStmt(Expr *expr, Stmt *stmt)	80
7.19.3	Member Function Documentation	80
7.19.3.1	UnParse(void)	80
7.19.4	Member Data Documentation	80

7.19.4.1	<code>expr_</code>	80
7.19.4.2	<code>stmt_</code>	80
7.20	<code>fcal::scanner::IfToken</code> Class Reference	81
7.20.1	Constructor & Destructor Documentation	83
7.20.1.1	<code>IfToken(parser::Parser *p, Token *t)</code>	83
7.20.2	Member Function Documentation	83
7.20.2.1	<code>description()</code>	83
7.20.2.2	<code>lbp()</code>	83
7.20.2.3	<code>nud()</code>	83
7.21	<code>fcal::scanner::IntConstToken</code> Class Reference	83
7.21.1	Detailed Description	86
7.21.2	Constructor & Destructor Documentation	86
7.21.2.1	<code>IntConstToken(parser::Parser *p, Token *t)</code>	86
7.21.3	Member Function Documentation	86
7.21.3.1	<code>description()</code>	86
7.21.3.2	<code>nud()</code>	86
7.22	<code>fcal::scanner::LeftParenToken</code> Class Reference	86
7.22.1	Detailed Description	89
7.22.2	Constructor & Destructor Documentation	89
7.22.2.1	<code>LeftParenToken(parser::Parser *p, Token *t)</code>	89
7.22.3	Member Function Documentation	89
7.22.3.1	<code>description()</code>	89
7.22.3.2	<code>lbp()</code>	89
7.22.3.3	<code>nud()</code>	89
7.23	<code>fcal::ast::letExpr</code> Class Reference	90
7.23.1	Detailed Description	91
7.23.2	Constructor & Destructor Documentation	92
7.23.2.1	<code>letExpr(Stmts *stmts, Expr *expr)</code>	92
7.23.3	Member Function Documentation	93
7.23.3.1	<code>UnParse(void)</code>	93

7.23.4	Member Data Documentation	93
7.23.4.1	expr_	93
7.23.4.2	stmts_	93
7.24	fcal::scanner::LetToken Class Reference	94
7.24.1	Constructor & Destructor Documentation	96
7.24.1.1	LetToken(parser::Parser *p, Token *t)	96
7.24.2	Member Function Documentation	96
7.24.2.1	description()	96
7.24.2.2	lbp()	96
7.24.2.3	nud()	96
7.25	fcal::ast::matrixDecl Class Reference	97
7.25.1	Detailed Description	99
7.25.2	Constructor & Destructor Documentation	99
7.25.2.1	matrixDecl(varName *var1, Expr *expr1, bool simpleMatrix)	99
7.25.2.2	matrixDecl(varName *var1, varName *var2, varName *var3, Expr *expr1, Expr *expr2, Expr *expr3, bool simpleMatrix)	99
7.25.3	Member Function Documentation	100
7.25.3.1	UnParse(void)	100
7.25.4	Member Data Documentation	100
7.25.4.1	expr1_	100
7.25.4.2	expr2_	100
7.25.4.3	expr3_	100
7.25.4.4	simpleMatrix_	100
7.25.4.5	var1_	101
7.25.4.6	var2_	101
7.25.4.7	var3_	101
7.26	fcal::ast::matrixExpr Class Reference	101
7.26.1	Detailed Description	102
7.26.2	Constructor & Destructor Documentation	103
7.26.2.1	matrixExpr(varName *var, Expr *expr1, Expr *expr2)	103
7.26.3	Member Function Documentation	103

7.26.3.1	UnParse(void)	103
7.26.4	Member Data Documentation	103
7.26.4.1	expr1_	103
7.26.4.2	expr2_	103
7.26.4.3	var_	103
7.27	fcgal::ast::nestedOrExpr Class Reference	104
7.27.1	Detailed Description	105
7.27.2	Constructor & Destructor Documentation	106
7.27.2.1	nestedOrExpr(varName *var, Expr *expr)	106
7.27.3	Member Function Documentation	106
7.27.3.1	UnParse(void)	106
7.27.4	Member Data Documentation	106
7.27.4.1	expr_	106
7.27.4.2	var_	106
7.28	fcgal::ast::Node Class Reference	107
7.28.1	Detailed Description	107
7.28.2	Constructor & Destructor Documentation	107
7.28.2.1	~Node(void)	107
7.28.3	Member Function Documentation	108
7.28.3.1	CppCode(void)	108
7.28.3.2	UnParse(void)	108
7.29	fcgal::ast::notExpr Class Reference	109
7.29.1	Detailed Description	111
7.29.2	Constructor & Destructor Documentation	112
7.29.2.1	notExpr(Expr *expr)	112
7.29.3	Member Function Documentation	113
7.29.3.1	UnParse(void)	113
7.29.4	Member Data Documentation	113
7.29.4.1	expr_	113
7.30	fcgal::scanner::NotOpToken Class Reference	113

7.30.1 Detailed Description	116
7.30.2 Constructor & Destructor Documentation	116
7.30.2.1 NotOpToken(parser::Parser *p, Token *t)	116
7.30.3 Member Function Documentation	116
7.30.3.1 description()	116
7.30.3.2 nud()	116
7.31 fcal::ast::parenthesisExpr Class Reference	117
7.31.1 Detailed Description	118
7.31.2 Constructor & Destructor Documentation	118
7.31.2.1 parenthesisExpr(Expr *expr)	118
7.31.3 Member Function Documentation	119
7.31.3.1 UnParse(void)	119
7.31.4 Member Data Documentation	119
7.31.4.1 expr_	119
7.32 fcal::parser::Parser Class Reference	119
7.32.1 Constructor & Destructor Documentation	122
7.32.1.1 Parser(void)	122
7.32.1.2 ~Parser(void)	122
7.32.2 Member Function Documentation	123
7.32.2.1 attempt_match(const scanner::TokenType &tt)	123
7.32.2.2 make_error_msg(const scanner::TokenType &terminal)	123
7.32.2.3 make_error_msg(const char *msg)	124
7.32.2.4 make_error_msg_expected(const scanner::TokenType &terminal)	124
7.32.2.5 match(const scanner::TokenType &tt)	125
7.32.2.6 next_is(const scanner::TokenType &tt)	126
7.32.2.7 next_token(void)	127
7.32.2.8 Parse(const char *text)	127
7.32.2.9 parse_addition(ParseResult left)	128
7.32.2.10 parse_char_const()	129
7.32.2.11 parse_decl()	129

7.32.2.12	parse_division(ParseResult left)	130
7.32.2.13	parse_expr(int rbp)	130
7.32.2.14	parse_false_kwd()	131
7.32.2.15	parse_float_const()	131
7.32.2.16	parse_if_expr()	132
7.32.2.17	parse_int_const()	133
7.32.2.18	parse_let_expr()	133
7.32.2.19	parse_matrix_decl()	134
7.32.2.20	parse_multiplication(ParseResult left)	135
7.32.2.21	parse_nested_expr()	135
7.32.2.22	parse_not_expr()	136
7.32.2.23	parse_relational_expr(ParseResult left)	136
7.32.2.24	parse_standard_decl()	137
7.32.2.25	parse_stmt()	138
7.32.2.26	parse_stmts()	140
7.32.2.27	parse_string_const()	140
7.32.2.28	parse_subtraction(ParseResult left)	141
7.32.2.29	parse_true_kwd()	142
7.32.2.30	parse_variable_name()	142
7.32.2.31	ParseProgram()	143
7.32.2.32	terminal_description(const scanner::TokenType &terminal)	144
7.32.3	Member Data Documentation	145
7.32.3.1	curr_token_	145
7.32.3.2	prev_token_	145
7.32.3.3	scanner_	145
7.32.3.4	tokens_	145
7.32.3.5	tokens_	145
7.33	fcal::parser::ParseResult Class Reference	145
7.33.1	Constructor & Destructor Documentation	147
7.33.1.1	ParseResult(void)	147

7.33.2	Member Function Documentation	147
7.33.2.1	ast(void)	147
7.33.2.2	ast(ast::Node *Node_ptr)	147
7.33.2.3	errors(void) const	147
7.33.2.4	errors(const std::string str_in)	148
7.33.2.5	ok(void) const	148
7.33.2.6	ok(bool result_in)	148
7.33.3	Member Data Documentation	148
7.33.3.1	ast_	148
7.33.3.2	errors_	148
7.33.3.3	ok_	148
7.34	fcal::scanner::PlusSignToken Class Reference	148
7.34.1	Detailed Description	151
7.34.2	Constructor & Destructor Documentation	151
7.34.2.1	PlusSignToken(parser::Parser *p, Token *t)	151
7.34.3	Member Function Documentation	151
7.34.3.1	description()	151
7.34.3.2	lbp()	151
7.34.3.3	led(parser::ParseResult left)	151
7.35	fcal::ast::printStmt Class Reference	152
7.35.1	Detailed Description	153
7.35.2	Constructor & Destructor Documentation	153
7.35.2.1	printStmt(Expr *expr)	153
7.35.3	Member Function Documentation	154
7.35.3.1	UnParse(void)	154
7.35.4	Member Data Documentation	154
7.35.4.1	expr_	154
7.36	fcal::scanner::RelationalOpToken Class Reference	154
7.36.1	Detailed Description	157
7.36.2	Constructor & Destructor Documentation	157

7.36.2.1	RelationalOpToken(parser::Parser *p, Token *t, std::string d)	157
7.36.3	Member Function Documentation	157
7.36.3.1	lbp()	157
7.36.3.2	led(parser::ParseResult left)	157
7.37	fcal::ast::repeatStmt Class Reference	158
7.37.1	Detailed Description	159
7.37.2	Constructor & Destructor Documentation	160
7.37.2.1	repeatStmt(varName *var, Expr *expr1, Expr *expr2, Stmt *stmt)	160
7.37.3	Member Function Documentation	160
7.37.3.1	UnParse(void)	160
7.37.4	Member Data Documentation	160
7.37.4.1	expr1_	160
7.37.4.2	expr2_	160
7.37.4.3	stmt_	161
7.37.4.4	var_	161
7.38	fcal::ast::Root Class Reference	161
7.38.1	Detailed Description	162
7.38.2	Constructor & Destructor Documentation	163
7.38.2.1	Root(varName *name, Stmts *stmts)	163
7.38.3	Member Function Documentation	164
7.38.3.1	UnParse(void)	164
7.38.4	Member Data Documentation	164
7.38.4.1	name_	164
7.38.4.2	stmts_	164
7.39	fcal::scanner::Scanner Class Reference	165
7.39.1	Detailed Description	166
7.39.2	Constructor & Destructor Documentation	166
7.39.2.1	Scanner()	166
7.39.2.2	~Scanner()	166
7.39.3	Member Function Documentation	166

7.39.3.1	consume_whitespace_and_comments(regex_t *, regex_t *, regex_t *, const char *)	166
7.39.3.2	Scan(const char *)	167
7.39.4	Member Data Documentation	168
7.39.4.1	comments	168
7.39.4.2	current_token	169
7.39.4.3	line_comment	169
7.39.4.4	previous_token	169
7.39.4.5	regex_strings	169
7.39.4.6	return_token	169
7.39.4.7	text	169
7.39.4.8	white_space	169
7.40	fcal::ast::semiColonStmt Class Reference	170
7.40.1	Detailed Description	171
7.40.2	Constructor & Destructor Documentation	171
7.40.2.1	semiColonStmt()	171
7.40.3	Member Function Documentation	172
7.40.3.1	UnParse(void)	172
7.41	fcal::ast::seqStmts Class Reference	172
7.41.1	Detailed Description	174
7.41.2	Constructor & Destructor Documentation	174
7.41.2.1	seqStmts Stmt *stmt, Stmt *stmts)	174
7.41.3	Member Function Documentation	175
7.41.3.1	UnParse(void)	175
7.41.4	Member Data Documentation	175
7.41.4.1	stmt_	175
7.41.4.2	stmts_	175
7.42	fcal::scanner::StarToken Class Reference	176
7.42.1	Detailed Description	178
7.42.2	Constructor & Destructor Documentation	178
7.42.2.1	StarToken(parser::Parser *p, Token *t)	178

7.42.3	Member Function Documentation	178
7.42.3.1	description()	178
7.42.3.2	lbp()	178
7.42.3.3	led(parser::ParseResult left)	178
7.43	fcal::ast::Stmt Class Reference	179
7.43.1	Detailed Description	179
7.44	fcal::ast::Stmts Class Reference	180
7.44.1	Detailed Description	181
7.45	fcal::scanner::StringConstToken Class Reference	181
7.45.1	Detailed Description	184
7.45.2	Constructor & Destructor Documentation	184
7.45.2.1	StringConstToken(parser::Parser *p, Token *t)	184
7.45.3	Member Function Documentation	184
7.45.3.1	description()	184
7.45.3.2	nud()	184
7.46	fcal::scanner::Token Class Reference	184
7.46.1	Detailed Description	185
7.46.2	Constructor & Destructor Documentation	186
7.46.2.1	Token()	186
7.46.2.2	Token(std::string, const TokenType, Token *)	186
7.46.2.3	~Token()	186
7.46.3	Member Function Documentation	186
7.46.3.1	lexeme()	186
7.46.3.2	next()	187
7.46.3.3	set_next(Token *current)	187
7.46.3.4	set_token(TokenType t, std::string lex)	187
7.46.3.5	terminal()	188
7.46.4	Member Data Documentation	188
7.46.4.1	lexeme_	188
7.46.4.2	next_	188

7.46.4.3	terminal_	188
7.47	fcsl::scanner::TrueKwdToken Class Reference	189
7.47.1	Detailed Description	191
7.47.2	Constructor & Destructor Documentation	191
7.47.2.1	TrueKwdToken(parser::Parser *p, Token *t)	191
7.47.3	Member Function Documentation	191
7.47.3.1	description()	191
7.47.3.2	nud()	191
7.48	fcsl::ast::varDecl Class Reference	191
7.48.1	Detailed Description	193
7.48.2	Constructor & Destructor Documentation	194
7.48.2.1	varDecl(decType type, varName *name)	194
7.48.3	Member Function Documentation	194
7.48.3.1	UnParse(void)	194
7.48.4	Member Data Documentation	194
7.48.4.1	name_	194
7.48.4.2	type_	194
7.49	fcsl::scanner::VariableNameToken Class Reference	195
7.49.1	Detailed Description	197
7.49.2	Constructor & Destructor Documentation	197
7.49.2.1	VariableNameToken(parser::Parser *p, Token *t)	197
7.49.3	Member Function Documentation	197
7.49.3.1	description()	197
7.49.3.2	nud()	197
7.50	fcsl::ast::varName Class Reference	198
7.50.1	Detailed Description	199
7.50.2	Constructor & Destructor Documentation	199
7.50.2.1	varName(std::string lexeme)	199
7.50.3	Member Function Documentation	199
7.50.3.1	UnParse(void)	199
7.50.4	Member Data Documentation	200
7.50.4.1	lexeme_	200
7.51	fcsl::ast::whileStmt Class Reference	200
7.51.1	Detailed Description	201
7.51.2	Constructor & Destructor Documentation	202
7.51.2.1	whileStmt(Expr *expr, Stmt *stmt)	202
7.51.3	Member Function Documentation	202
7.51.3.1	UnParse(void)	202
7.51.4	Member Data Documentation	202
7.51.4.1	expr_	202
7.51.4.2	stmt_	202

8 File Documentation	203
8.1 include/ast.h File Reference	203
8.2 include/ext_token.h File Reference	205
8.3 include/mainpage.h File Reference	207
8.4 include/parse_result.h File Reference	207
8.5 include/parser.h File Reference	208
8.6 include/read_input.h File Reference	210
8.7 include/regex.h File Reference	210
8.8 include/scanner.h File Reference	211
8.9 include/scanner_class.h File Reference	213
8.10 include/token_class.h File Reference	214
8.11 src/ast.cc File Reference	215
8.12 src/ext_token.cc File Reference	215
8.13 src/parser.cc File Reference	216
8.14 src/read_input.cc File Reference	217
8.15 src/regex.cc File Reference	218
8.16 src/scanner_class.cc File Reference	218
8.17 src/token_class.cc File Reference	219
 Index	 221

Chapter 1

INDEX PAGE

1.1 Introduction

1.1.1 Team Name

Nostromo

1.1.2 TA Name

John Harwell

1.1.3 Discussion Section

02

1.1.4 Details

This document contains the details of different classes in scanner, parser and ast namespaces under the fcal namespace. Each class in this project can be viewed in this documentation, also the relationship between the classes has been shown. The intention of this project is to compile a file written in FCAL language and convert it to C/C++ code. In the present iteration, the goal is to create an AST and since the project is yet to be complete the documentation covers everything upto the present iteration which is the creation of the AST. All methods of classes with their arguments and details of some important methods with their description is included. This documentation should provide the reader with brief information regarding the work done so far in this project.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

fcal	Namespaces	11
fcal::ast	11
fcal::parser	12
fcal::scanner	13

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

fcsl::scanner::ExtToken	53
fcsl::scanner::CharConstToken	27
fcsl::scanner::DashToken	33
fcsl::scanner::EndOfFileToken	44
fcsl::scanner::FalseKwdToken	63
fcsl::scanner::FloatConstToken	65
fcsl::scanner::ForwardSlashToken	68
fcsl::scanner::IfToken	81
fcsl::scanner::IntConstToken	83
fcsl::scanner::LeftParenToken	86
fcsl::scanner::LetToken	94
fcsl::scanner::NotOpToken	113
fcsl::scanner::PlusSignToken	148
fcsl::scanner::RelationalOpToken	154
fcsl::scanner::StarToken	176
fcsl::scanner::StringConstToken	181
fcsl::scanner::TrueKwdToken	189
fcsl::scanner::VariableNameToken	195
fcsl::ast::Node	107
fcsl::ast::Decl	37
fcsl::ast::matrixDecl	97
fcsl::ast::varDecl	191
fcsl::ast::Expr	52
fcsl::ast::binaryExpr	19
fcsl::ast::boolExpr	22
fcsl::ast::constantExpr	31
fcsl::ast::ifExpr	75
fcsl::ast::letExpr	90
fcsl::ast::matrixExpr	101
fcsl::ast::nestedOrExpr	104
fcsl::ast::notExpr	109
fcsl::ast::parenthesisExpr	117
fcsl::ast::Root	161
fcsl::ast::Stmt	179
fcsl::ast::bracketStmt	25

fcal::ast::declStmt	38
fcal::ast::equalsStmt	47
fcal::ast::ifElseStmt	72
fcal::ast::ifStmt	78
fcal::ast::printStmt	152
fcal::ast::repeatStmt	158
fcal::ast::semiColonStmt	170
fcal::ast::whileStmt	200
fcal::ast::Stmts	180
fcal::ast::emptyStmts	41
fcal::ast::seqStmts	172
fcal::ast::varName	198
fcal::parser::Parser	119
fcal::parser::ParseResult	145
fcal::scanner::Scanner	165
fcal::scanner::Token	184

Chapter 4

Class Index

4.1 Class List

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

fcal::ast::binaryExpr	19
fcal::ast::boolExpr	22
fcal::ast::bracketStmt	25
fcal::scanner::CharConstToken	
Char Const	27
fcal::ast::constantExpr	31
fcal::scanner::DashToken	
Dash	33
fcal::ast::Decl	37
fcal::ast::declStmt	38
fcal::ast::emptyStmts	41
fcal::scanner::EndOfFileToken	
End of File	44
fcal::ast::equalsStmt	47
fcal::ast::Expr	52
fcal::scanner::ExtToken	53
fcal::scanner::FalseKwdToken	
False Kwd	63
fcal::scanner::FloatConstToken	
Float Const	65
fcal::scanner::ForwardSlashToken	
ForwardSlash	68
fcal::ast::ifElseStmt	72
fcal::ast::ifExpr	75
fcal::ast::ifStmt	78
fcal::scanner::IfToken	81
fcal::scanner::IntConstToken	
Int Const	83
fcal::scanner::LeftParenToken	
Left Paren	86
fcal::ast::letExpr	90
fcal::scanner::LetToken	94
fcal::ast::matrixDecl	97
fcal::ast::matrixExpr	101
fcal::ast::nestedOrExpr	104

fcal::ast::Node	107
fcal::ast::notExpr	109
fcal::scanner::NotOpToken	113
fcal::ast::parenthesisExpr	117
fcal::parser::Parser	119
fcal::parser::ParseResult	145
fcal::scanner::PlusSignToken	
Plus Sign	148
fcal::ast::printStmt	152
fcal::scanner::RelationalOpToken	
Relational Op	154
fcal::ast::repeatStmt	158
fcal::ast::Root	161
fcal::scanner::Scanner	165
fcal::ast::semiColonStmt	170
fcal::ast::seqStmts	172
fcal::scanner::StarToken	
Star	176
fcal::ast::Stmt	179
fcal::ast::Stmts	180
fcal::scanner::StringConstToken	
String Const	181
fcal::scanner::Token	184
fcal::scanner::TrueKwdToken	
True Kwd	189
fcal::ast::varDecl	191
fcal::scanner::VariableNameToken	
Variable Name	195
fcal::ast::varName	
This class holds the lexeme details of a variable name	198
fcal::ast::whileStmt	200

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/ast.h	203
include/ext_token.h	205
include/mainpage.h	207
include/parse_result.h	207
include/parser.h	208
include/read_input.h	210
include/regex.h	210
include/scanner.h	211
include/scanner_class.h	213
include/token_class.h	214
src/ast.cc	215
src/ext_token.cc	215
src/parser.cc	216
src/read_input.cc	217
src/regex.cc	218
src/scanner_class.cc	218
src/token_class.cc	219

Chapter 6

Namespace Documentation

6.1 fcal Namespace Reference

Namespaces.

Namespaces

- [ast](#)
- [parser](#)
- [scanner](#)

6.1.1 Detailed Description

Namespaces.

6.2 fcal::ast Namespace Reference

Classes

- class [binaryExpr](#)
- class [boolExpr](#)
- class [bracketStmt](#)
- class [constantExpr](#)
- class [Decl](#)
- class [declStmt](#)
- class [emptyStmts](#)
- class [equalsStmt](#)
- class [Expr](#)
- class [ifElseStmt](#)
- class [ifExpr](#)
- class [ifStmt](#)
- class [letExpr](#)
- class [matrixDecl](#)

- class [matrixExpr](#)
- class [nestedOrExpr](#)
- class [Node](#)
- class [notExpr](#)
- class [parenthesisExpr](#)
- class [printStmt](#)
- class [repeatStmt](#)
- class [Root](#)
- class [semiColonStmt](#)
- class [seqStmts](#)
- class [Stmt](#)
- class [Stmts](#)
- class [varDecl](#)
- class [varName](#)

This class holds the lexeme details of a variable name.

- class [whileStmt](#)

Enumerations

- enum [decType](#) { [int_](#), [float_](#), [string_](#), [boolean_](#) }

6.2.1 Enumeration Type Documentation

6.2.1.1 enum `fcval::ast::decType`

`decType` has four possible values based on declaration types.

Enumerator

int_

float_

string_

boolean_

6.3 `fcval::parser` Namespace Reference

Classes

- class [Parser](#)
- class [ParseResult](#)

6.4 fcal::scanner Namespace Reference

Classes

- class [CharConstToken](#)
Char Const.
- class [DashToken](#)
Dash.
- class [EndOfFileToken](#)
End of File.
- class [ExtToken](#)
- class [FalseKwdToken](#)
False Kwd.
- class [FloatConstToken](#)
Float Const.
- class [ForwardSlashToken](#)
ForwardSlash.
- class [IfToken](#)
- class [IntConstToken](#)
Int Const.
- class [LeftParenToken](#)
Left Paren.
- class [LetToken](#)
- class [NotOpToken](#)
- class [PlusSignToken](#)
Plus Sign.
- class [RelationalOpToken](#)
Relational Op.
- class [Scanner](#)
- class [StarToken](#)
Star.
- class [StringConstToken](#)
String Const.
- class [Token](#)
- class [TrueKwdToken](#)
True Kwd.
- class [VariableNameToken](#)
Variable Name.

Typedefs

- typedef enum [kTokenEnumType](#) [TokenType](#)

Enumerations

- enum [kTokenEnumType](#) {
[kIntKwd](#), [kFloatKwd](#), [kBoolKwd](#), [kTrueKwd](#),
[kFalseKwd](#), [kStringKwd](#), [kMatrixKwd](#), [kLetKwd](#),
[kInKwd](#), [kEndKwd](#), [kIfKwd](#), [kThenKwd](#),
[kElseKwd](#), [kRepeatKwd](#), [kWhileKwd](#), [kPrintKwd](#),
[kToKwd](#), [kIntConst](#), [kFloatConst](#), [kStringConst](#),
[kVariableName](#), [kLeftParen](#), [kRightParen](#), [kLeftCurly](#),
[kRightCurly](#), [kLeftSquare](#), [kRightSquare](#), [kSemiColon](#),
[kColon](#), [kAssign](#), [kPlusSign](#), [kStar](#),
[kDash](#), [kForwardSlash](#), [kLessThan](#), [kLessThanEqual](#),
[kGreaterThan](#), [kGreaterThanEqual](#), [kEqualsEquals](#), [kNotEquals](#),
[kAndOp](#), [kOrOp](#), [kNotOp](#), [kEndOfFile](#),
[kLexicalError](#) }

Functions

- char * [ReadInputFromFile](#) (const char *filename)
- char * [ReadInput](#) (int argc, char **argv)
- regex_t * [make_regex](#) (const char *pattern)
- int [match_regex](#) (regex_t *re, const char *text)

Variables

- const int [kRegexNSub](#) = 1

6.4.1 Typedef Documentation

6.4.1.1 typedef enum kTokenEnumType fcal::scanner::TokenType

6.4.2 Enumeration Type Documentation

6.4.2.1 enum fcal::scanner::kTokenEnumType

Enumerator

kIntKwd
kFloatKwd
kBoolKwd
kTrueKwd
kFalseKwd
kStringKwd
kMatrixKwd
kLetKwd
kInKwd
kEndKwd
kIfKwd
kThenKwd

kElseKwd
kRepeatKwd
kWhileKwd
kPrintKwd
kToKwd
kIntConst
kFloatConst
kStringConst
kVariableName
kLeftParen
kRightParen
kLeftCurly
kRightCurly
kLeftSquare
kRightSquare
kSemiColon
kColon
kAssign
kPlusSign
kStar
kDash
kForwardSlash
kLessThan
kLessThanEqual
kGreaterThan
kGreaterThanEqual
kEqualsEquals
kNotEquals
kAndOp
kOrOp
kNotOp
kEndOfFile
kLexicalError

6.4.3 Function Documentation

6.4.3.1 `regex_t * fcal::scanner::make_regex (const char * pattern)`

"Compile" the regular expression. This sets up the regex to do the matching specified by the regular expression given as a character string.

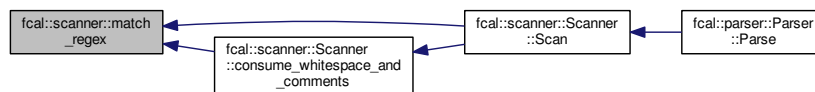
Here is the caller graph for this function:



6.4.3.2 `int fcal::scanner::match_regex (regex_t * re, const char * text)`

Execute the regular expression match against the text. If it matches, the beginning and ending of the matched text are stored in the first element of the matches array.

Here is the caller graph for this function:

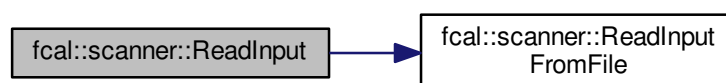


6.4.3.3 `char * fcal::scanner::ReadInput (int argc, char ** argv)`

[ReadInput\(\)](#) - Read a file into a char buffer. The calling function is responsible for disposing of the return memory.

RETURN: `char*` - The buffer, or NULL if an error occurred.

Here is the call graph for this function:

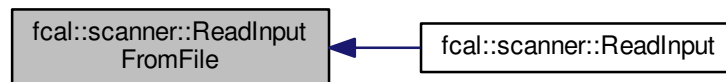


6.4.3.4 `char * fcal::scanner::ReadInputFromFile (const char * filename)`

[ReadInputFromFile\(\)](#) - Do the actual reading of the file into the buffer

RETURN: `char*` - The buffer, or NULL if an error occurred.

Here is the caller graph for this function:



6.4.4 Variable Documentation

6.4.4.1 `const int fcal::scanner::kRegexNSub = 1`

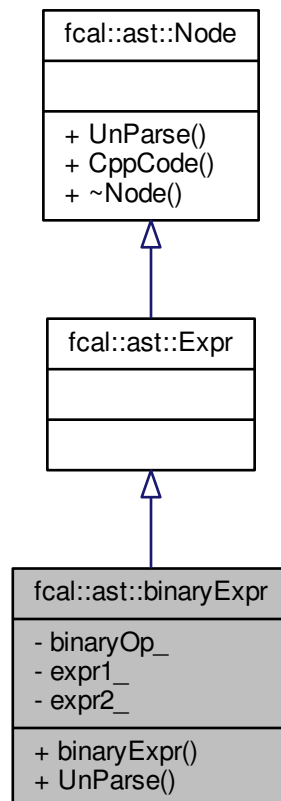
Chapter 7

Class Documentation

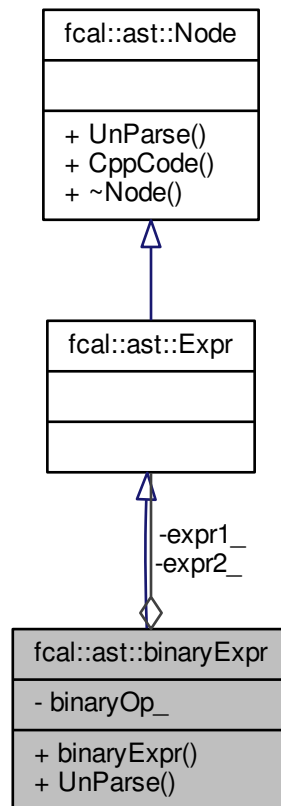
7.1 fcal::ast::binaryExpr Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::binaryExpr:



Collaboration diagram for `fcgal::ast::binaryExpr`:



Public Member Functions

- `binaryExpr` (`std::string` `binaryOp`, `Expr` *`expr1`, `Expr` *`expr2`)
- `std::string` `UnParse` (`void`)
Retruns the string : `expr1_ -> UnParse()` + `binaryOp_` + `expr2_ -> UnParse()`

Private Attributes

- `std::string` `binaryOp_`
*string representing +,/, -, **
- `Expr` * `expr1_`
`Expr` representing Left Operand in this Binary Operation.
- `Expr` * `expr2_`
`Expr` representing Right Operand in this Binary Operation.

7.1.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the productions,

`Expr ::= Expr '*' Expr`

`Expr ::= Expr '/' Expr`

`Expr ::= Expr '+' Expr`

`Expr ::= Expr '-' Expr`

7.1.2 Constructor & Destructor Documentation

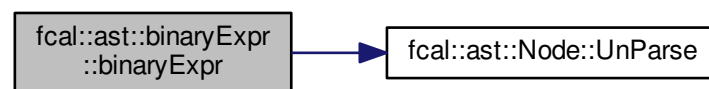
7.1.2.1 `fcal::ast::binaryExpr::binaryExpr (std::string binaryOp, Expr * expr1, Expr * expr2)` `[inline]`

This constructor takes three parameters

Parameters

<i>binaryOp</i>	string representing +,/,*,*
<i>expr1</i>	left Expr of binaryOp
<i>expr2</i>	Right Expr of binaryOp

Here is the call graph for this function:



7.1.3 Member Function Documentation

7.1.3.1 `std::string fcal::ast::binaryExpr::UnParse (void)` `[virtual]`

Retruns the string : `expr1_->UnParse() + binaryOp_ + expr2_->UnParse()`

Reimplemented from `fcal::ast::Node`.

7.1.4 Member Data Documentation

7.1.4.1 `std::string fcal::ast::binaryExpr::binaryOp_` `[private]`

string representing +,/,*,*

7.1.4.2 `Expr* fcal::ast::binaryExpr::expr1_` [private]

[Expr](#) representing Left Operand in this Binary Operation.

7.1.4.3 `Expr* fcal::ast::binaryExpr::expr2_` [private]

[Expr](#) representing Right Operand in this Binary Operation.

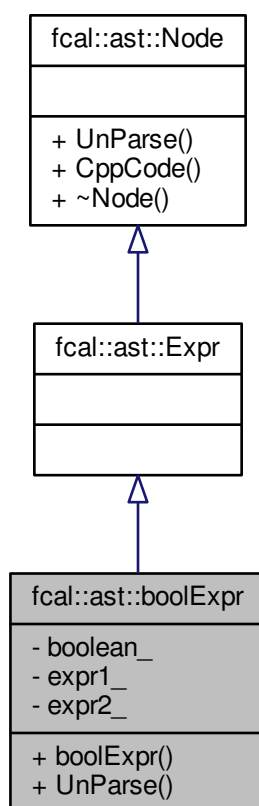
The documentation for this class was generated from the following files:

- [include/ast.h](#)
- [src/ast.cc](#)

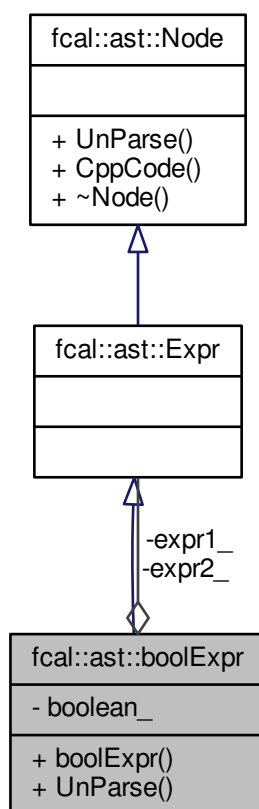
7.2 `fcal::ast::boolExpr` Class Reference

```
#include <ast.h>
```

Inheritance diagram for `fcal::ast::boolExpr`:



Collaboration diagram for fcal::ast::boolExpr:



Public Member Functions

- [boolExpr](#) (std::string boolean, [Expr](#) *expr1, [Expr](#) *expr2)
- std::string [UnParse](#) (void)
Returns the string : *expr1_ -> [UnParse\(\)](#) + boolean_ + expr2_ -> [UnParse\(\)](#)*

Private Attributes

- std::string [boolean_](#)
string representing relational operators
- [Expr](#) * [expr1_](#)
[Expr](#) representing Left Operand in this relational Operation.
- [Expr](#) * [expr2_](#)
[Expr](#) representing Right Operand in this relational Operation.

7.2.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the productions,

```
Expr ::= Expr '>' Expr
Expr ::= Expr '>=' Expr
Expr ::= Expr '<' Expr
Expr ::= Expr '<=' Expr
Expr ::= Expr '==' Expr
Expr ::= Expr '!=' Expr
Expr ::= Expr '&&' Expr
Expr ::= Expr '||' Expr
```

7.2.2 Constructor & Destructor Documentation

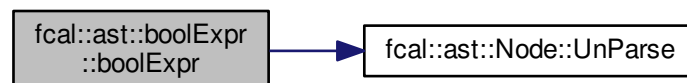
7.2.2.1 `fcgal::ast::boolExpr::boolExpr (std::string boolean, Expr * expr1, Expr * expr2)` `[inline]`

This constructor takes three parameters

Parameters

<i>boolean</i>	string representing relational operators
<i>expr1</i>	Expr representing Left Operand
<i>expr2</i>	Expr representing Right Operand

Here is the call graph for this function:



7.2.3 Member Function Documentation

7.2.3.1 `std::string fcgal::ast::boolExpr::UnParse (void)` `[virtual]`

Returns the string : `expr1_->UnParse() + boolean_ + expr2_->UnParse()`

Reimplemented from [fcgal::ast::Node](#).

7.2.4 Member Data Documentation

7.2.4.1 `std::string fcgal::ast::boolExpr::boolean_` `[private]`

string representing relational operators

7.2.4.2 `Expr* fcal::ast::boolExpr::expr1_ [private]`

[Expr](#) representing Left Operand in this relational Operation.

7.2.4.3 `Expr* fcal::ast::boolExpr::expr2_ [private]`

[Expr](#) representing Right Operand in this relational Operation.

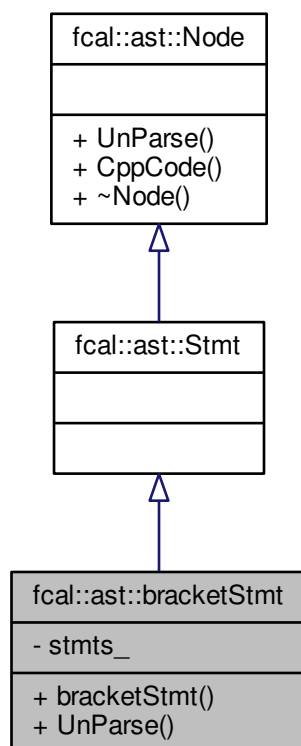
The documentation for this class was generated from the following files:

- [include/ast.h](#)
- [src/ast.cc](#)

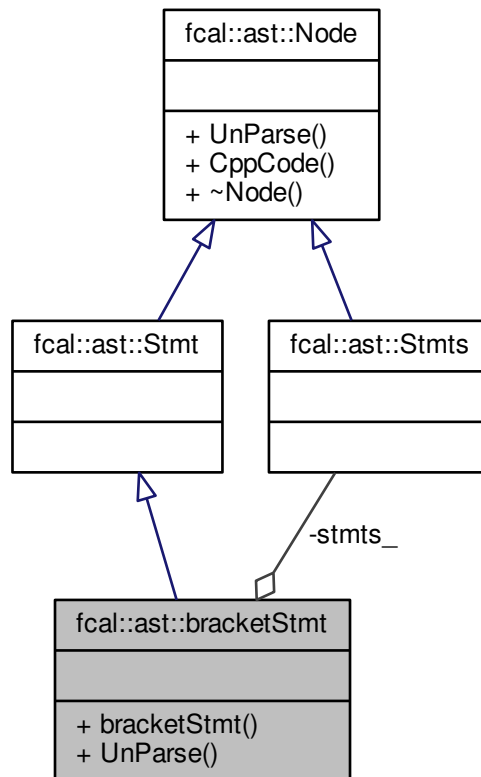
7.3 fcal::ast::bracketStmt Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::bracketStmt:



Collaboration diagram for `fcal::ast::bracketStmt`:



Public Member Functions

- `bracketStmt (Stmts *stmts)`
- `std::string UnParse (void)`
Returns the string : `"{" + stmts_ -> UnParse() + "}"`.

Private Attributes

- `Stmts * stmts_`

7.3.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
`Stmt ::= '{' Stmts '}'`

7.3.2 Constructor & Destructor Documentation

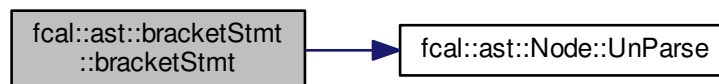
7.3.2.1 `fcal::ast::bracketStmt::bracketStmt (Stmts * stmts)` `[inline],[explicit]`

This constructor takes only one parameter

Parameters

<i>stmts</i>	a Stmt* holding the address which contains details of Stmts under brackets '{' Stmts '}'
--------------	--

Here is the call graph for this function:



7.3.3 Member Function Documentation

7.3.3.1 std::string fcal::ast::bracketStmt::UnParse (void) [virtual]

Returns the string : " {" + stmts_ -> [UnParse\(\)](#) + "}".

Reimplemented from [fcal::ast::Node](#).

7.3.4 Member Data Documentation

7.3.4.1 Stmt* fcal::ast::bracketStmt::stmts_ [private]

stmts_ holds the address containing details of [Stmts](#) under brackets '{' [Stmts](#) '}'

The documentation for this class was generated from the following files:

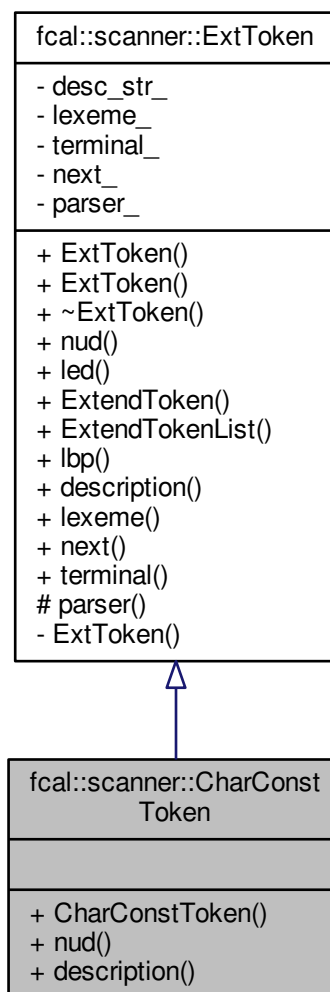
- include/[ast.h](#)
- src/[ast.cc](#)

7.4 fcal::scanner::CharConstToken Class Reference

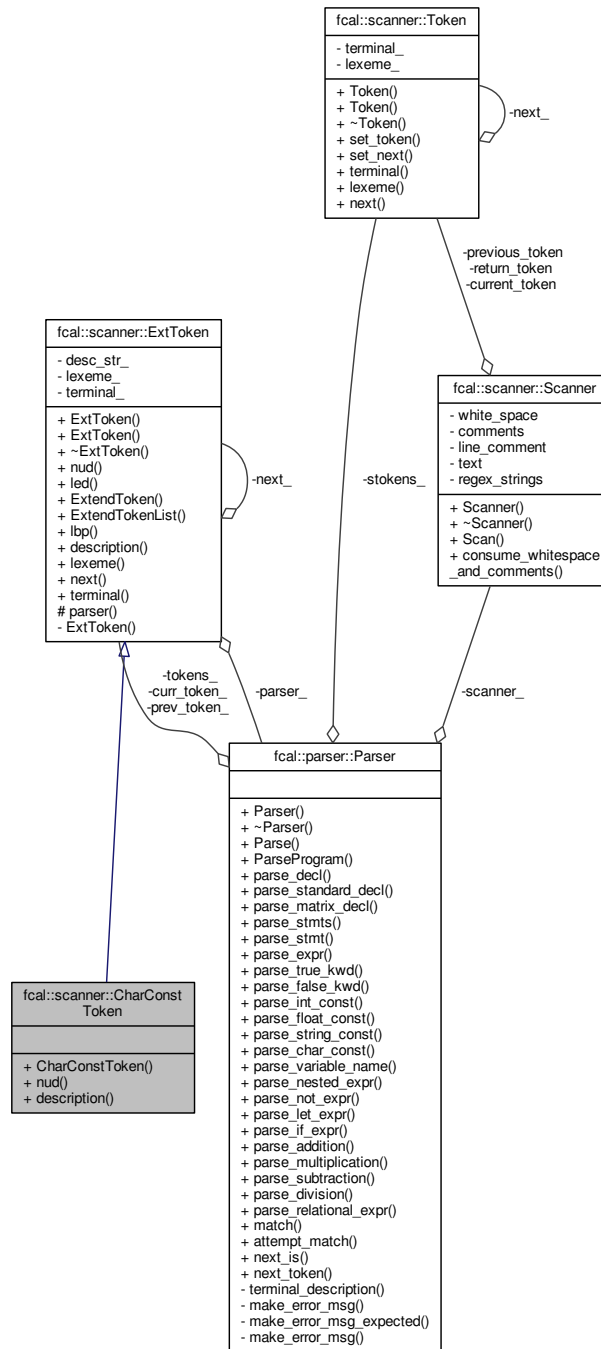
Char Const.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::CharConstToken:



Collaboration diagram for fcal::scanner::CharConstToken:



Public Member Functions

- [CharConstToken](#) ([parser::Parser](#) *p, [Token](#) *t)
- [parser::ParseResult nud](#) ()
- [std::string description](#) ()

Additional Inherited Members

7.4.1 Detailed Description

Char Const.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 `fcal::scanner::CharConstToken::CharConstToken (parser::Parser * p, Token * t)` `[inline]`

7.4.3 Member Function Documentation

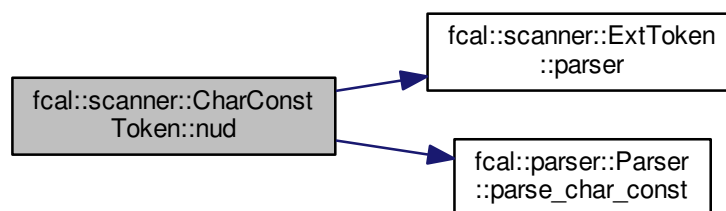
7.4.3.1 `std::string fcal::scanner::CharConstToken::description ()` `[inline]`, `[virtual]`

Reimplemented from [fcal::scanner::ExtToken](#).

7.4.3.2 `parser::ParseResult fcal::scanner::CharConstToken::nud (void)` `[inline]`, `[virtual]`

Reimplemented from [fcal::scanner::ExtToken](#).

Here is the call graph for this function:



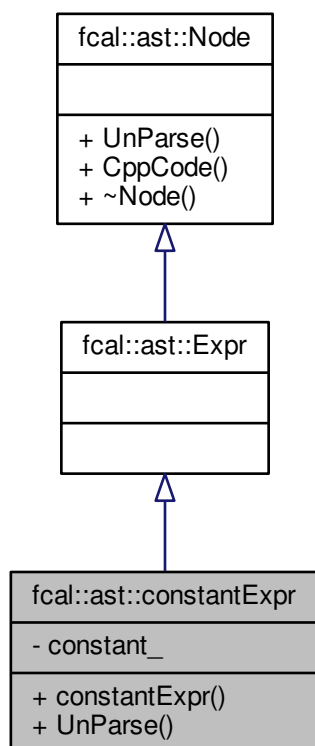
The documentation for this class was generated from the following file:

- [include/ext_token.h](#)

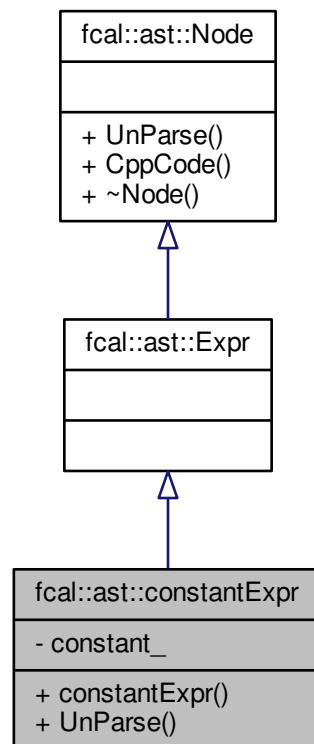
7.5 fcal::ast::constantExpr Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::constantExpr:



Collaboration diagram for `fcgal::ast::constantExpr`:



Public Member Functions

- `constantExpr` (`std::string const_`)
- `std::string UnParse` (`void`)
Returns the string : `constant_`.

Private Attributes

- `std::string constant_`

7.5.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the productions,

`Expr ::= varName`

`Expr ::= integerConst | floatConst | stringConst`

`Expr ::= 'True' | 'False'`

7.5.2 Constructor & Destructor Documentation

7.5.2.1 `fcgal::ast::constantExpr::constantExpr (std::string const_) [inline],[explicit]`

This constructor takes one parameter

Parameters

<i>const</i> ↔	takes a string constant representing a varName , integer, float , string , True or False
—	

Here is the call graph for this function:



7.5.3 Member Function Documentation

7.5.3.1 std::string fcal::ast::constantExpr::UnParse (void) [virtual]

Returns the string : constant_.

Reimplemented from [fcal::ast::Node](#).

7.5.4 Member Data Documentation

7.5.4.1 std::string fcal::ast::constantExpr::constant_ [private]

a string constant representing either one of the following : [varName](#) , integer, float , string , True , False

The documentation for this class was generated from the following files:

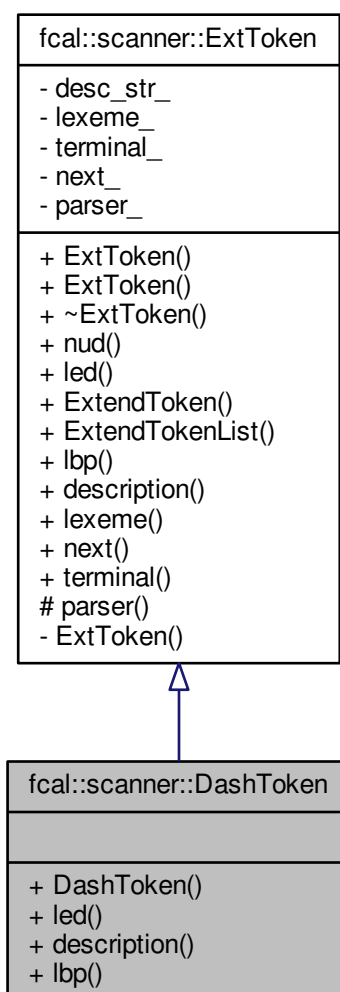
- include/[ast.h](#)
- src/[ast.cc](#)

7.6 fcal::scanner::DashToken Class Reference

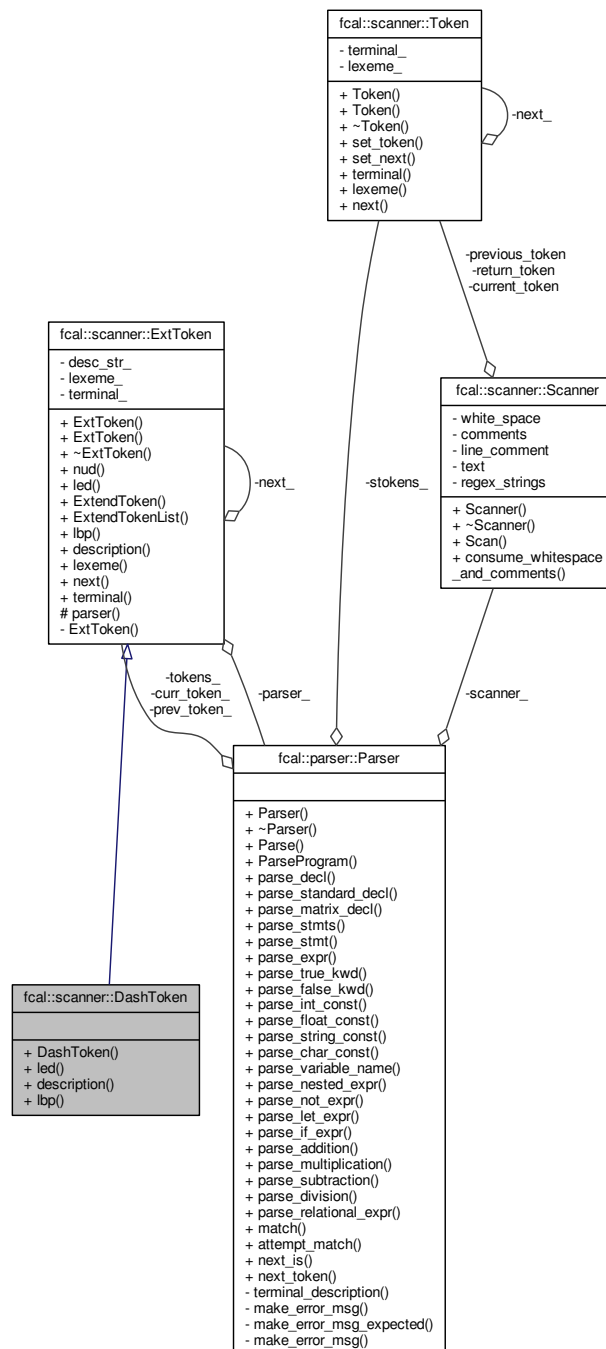
Dash.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::DashToken:



Collaboration diagram for fcal::scanner::DashToken:



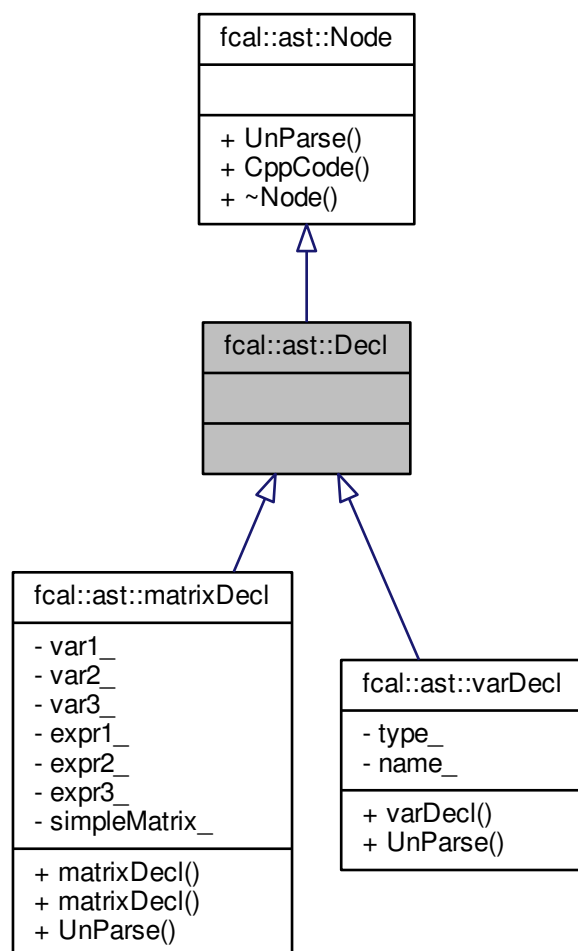
Public Member Functions

- [DashToken](#) ([parser::Parser *p](#), [Token *t](#))
- [parser::ParseResult led](#) ([parser::ParseResult left](#))
- `std::string description ()`
- `int lbp ()`

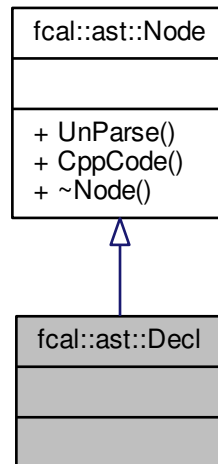
7.7 fcal::ast::Decl Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::Decl:



Collaboration diagram for `fcgal::ast::Decl`:



Additional Inherited Members

7.7.1 Detailed Description

This is the abstract class which inherits from [Node](#). It has no implementation for any functions derived from [Node](#). It is the base class for classes which implement the productions which are derived from the nonterminal '[Decl](#)' in the FCAL grammar.

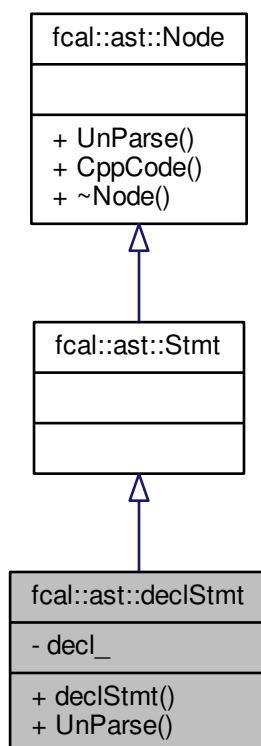
The documentation for this class was generated from the following file:

- [include/ast.h](#)

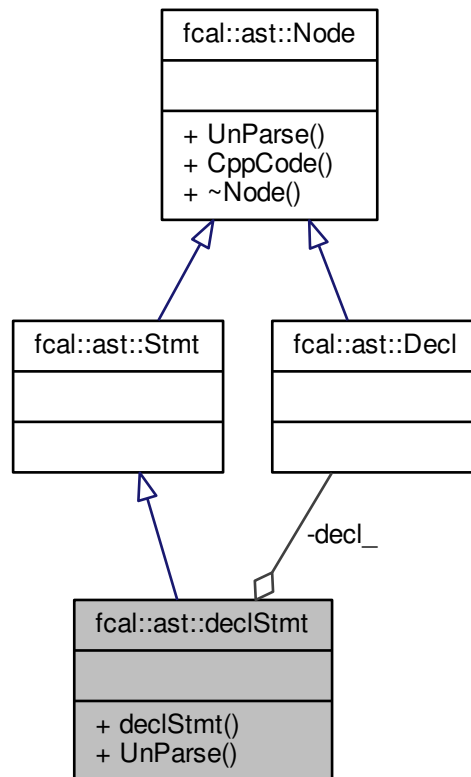
7.8 `fcgal::ast::declStmt` Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::declStmt:



Collaboration diagram for `fcal::ast::declStmt`:



Public Member Functions

- [declStmt](#) ([Decl](#) *decl)
- `std::string` [UnParse](#) (void)
Returns the string : `decl_->UnParse()`

Private Attributes

- [Decl](#) * [decl_](#)

7.8.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production, [Stmts](#) ::= [Stmt](#) [Stmts](#)

7.8.2 Constructor & Destructor Documentation

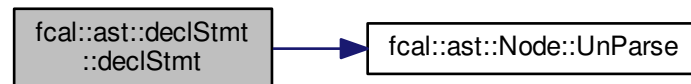
7.8.2.1 `fcal::ast::declStmt::declStmt (Decl * decl) [inline], [explicit]`

This constructor takes only one parameter

Parameters

<i>decl</i>	a Decl* holding the address which contains information reagarding a declaration
-------------	---

Here is the call graph for this function:



7.8.3 Member Function Documentation

7.8.3.1 `std::string fcal::ast::declStmt::UnParse (void)` [virtual]

Returns the string : `decl_->UnParse()`

Reimplemented from [fcal::ast::Node](#).

7.8.4 Member Data Documentation

7.8.4.1 `Decl* fcal::ast::declStmt::decl_` [private]

`decl_` holds the address which contains details regarding the type of Declaration

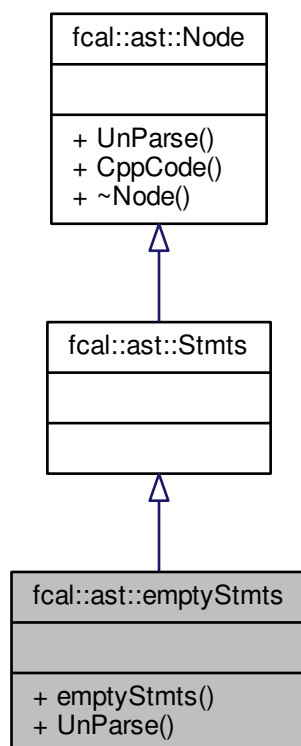
The documentation for this class was generated from the following files:

- [include/ast.h](#)
- [src/ast.cc](#)

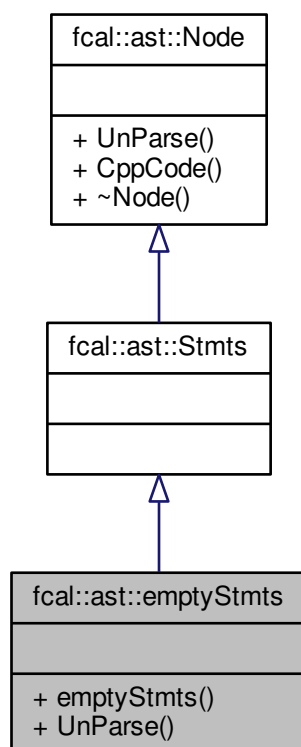
7.9 fcal::ast::emptyStmts Class Reference

```
#include <ast.h>
```

Inheritance diagram for `fcal::ast::emptyStmts`:



Collaboration diagram for fcal::ast::emptyStmts:



Public Member Functions

- [emptyStmts](#) (void)
Constructor for [emptyStmts](#), it does nothing.
- std::string [UnParse](#) (void)
Returns an empty String.

7.9.1 Detailed Description

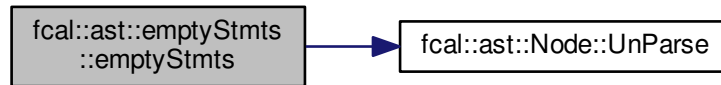
This is a concrete class in the ast class heirarchy. It implements the production, [Stmts](#) ::= <<empty>>

7.9.2 Constructor & Destructor Documentation

7.9.2.1 fcal::ast::emptyStmts::emptyStmts (void) [inline]

Constructor for [emptyStmts](#), it does nothing.

Here is the call graph for this function:



7.9.3 Member Function Documentation

7.9.3.1 `std::string fcal::ast::emptyStmts::UnParse (void)` [virtual]

Returns an empty String.

Returns the string : "".

Reimplemented from [fcal::ast::Node](#).

The documentation for this class was generated from the following files:

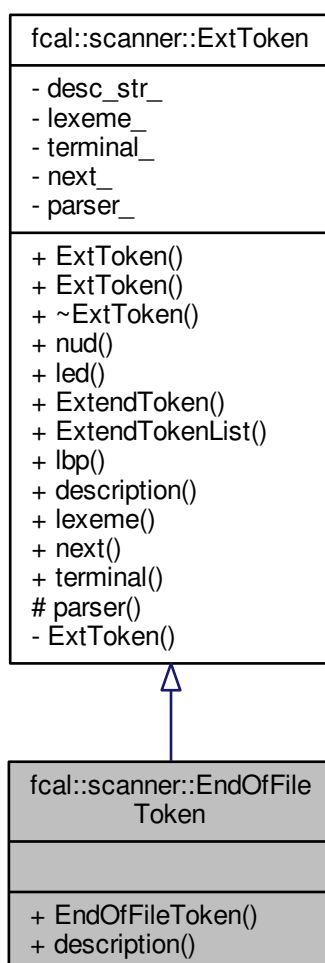
- [include/ast.h](#)
- [src/ast.cc](#)

7.10 `fcal::scanner::EndOfFileToken` Class Reference

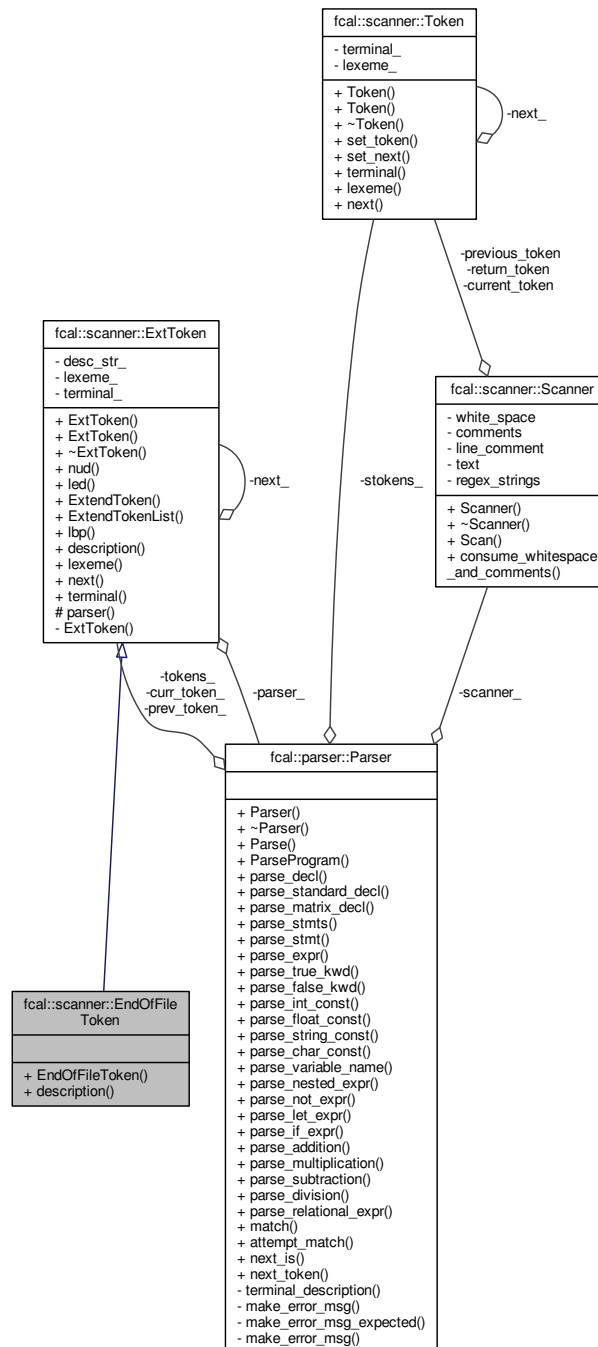
End of File.

```
#include <ext_token.h>
```


Inheritance diagram for fcal::scanner::EndOfFileToken:



Collaboration diagram for `fcsl::scanner::EndOfFileToken`:



Public Member Functions

- [EndOfFileToken](#) ([parser::Parser *p](#), [Token *t](#))
- `std::string` [description](#) ()

Additional Inherited Members

7.10.1 Detailed Description

End of File.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 `fcal::scanner::EndOfFileToken::EndOfFileToken (parser::Parser * p, Token * t)` `[inline]`

7.10.3 Member Function Documentation

7.10.3.1 `std::string fcal::scanner::EndOfFileToken::description ()` `[inline]`, `[virtual]`

Reimplemented from [fcal::scanner::ExtToken](#).

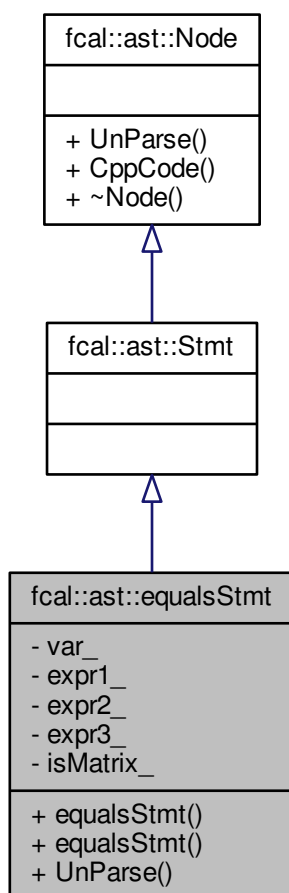
The documentation for this class was generated from the following file:

- [include/ext_token.h](#)

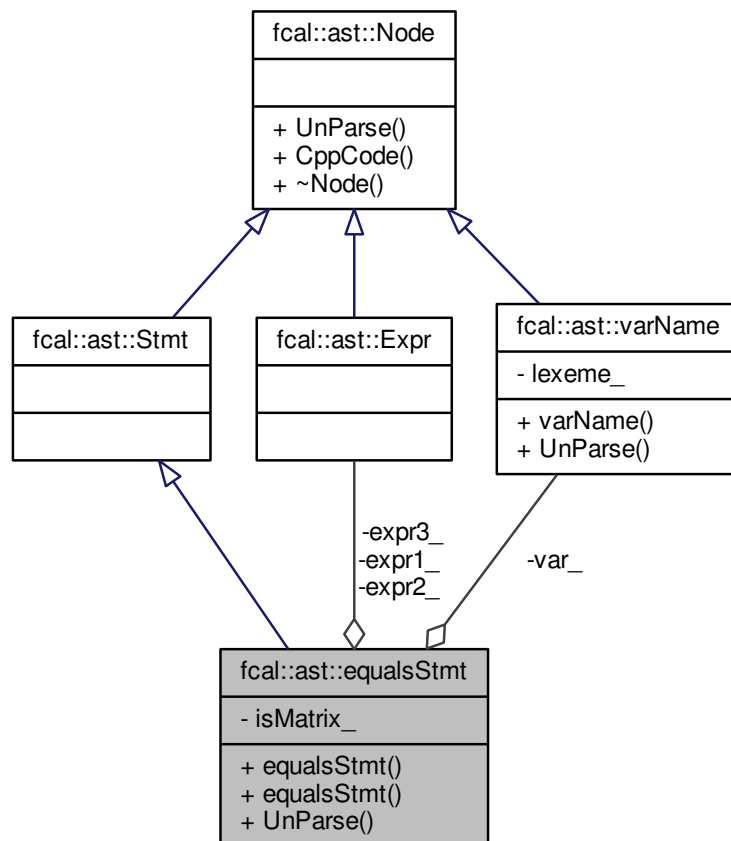
7.11 fcal::ast::equalsStmt Class Reference

```
#include <ast.h>
```

Inheritance diagram for `fcal::ast::equalsStmt`:



Collaboration diagram for fcal::ast::equalsStmt:



Public Member Functions

- `equalsStmt (varName *var, Expr *expr1, bool isMatrix)`
- `equalsStmt (varName *var, Expr *expr1, Expr *expr2, Expr *expr3, bool isMatrix)`
- `std::string UnParse (void)`

Private Attributes

- `varName * var_`
variable name or that of a matrix
- `Expr * expr1_`
- `Expr * expr2_`
Expr representing second element of Matrix if isMatrix is True.
- `Expr * expr3_`
- `bool isMatrix_`
isMatrix distinguishes between regular variable or matrix constructor

7.11.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
`Stmt ::= varName '=' Expr ';' | varName '[' Expr ':' Expr ']' '=' Expr ';' ;`

7.11.2 Constructor & Destructor Documentation

7.11.2.1 `fcal::ast::equalsStmt::equalsStmt (varName * var, Expr * expr1, bool isMatrix) [inline]`

This constructor takes three parameters

Parameters

<i>var</i>	a varName* which holds the address containing the information of varName representing an Expr
<i>expr1</i>	a Expr* holding the address which contains the information of an Expr whose value will be assigned to a varName
<i>isMatrix</i>	bool value which is True if assignment is a Martix and False otherwise

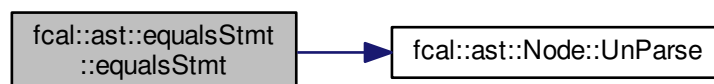
7.11.2.2 `fcal::ast::equalsStmt::equalsStmt (varName * var, Expr * expr1, Expr * expr2, Expr * expr3, bool isMatrix) [inline]`

This constructor takes five parameters

Parameters

<i>var</i>	varName of a matrix
<i>expr1</i>	Expr representing rows of matrix
<i>expr2</i>	Expr representing columns of matrix
<i>expr3</i>	Expr whise value will be assigned to a particular index in the matrix
<i>isMatrix</i>	bool value which is True if assignment is a Martix and False otherwise

Here is the call graph for this function:



7.11.3 Member Function Documentation

7.11.3.1 std::string fcal::ast::equalsStmt::UnParse (void) [virtual]

Returns the string : `var_->UnParse() + " = " + expr1_->UnParse() + ";\n"` , if `isMatrix_` is False.

Returns the string : `var_->UnParse() + " [" + expr1_->UnParse() + ":"`

- `expr2_->UnParse() + + "]" + expr3_->UnParse() + ";\n"` , if `isMatrix_` is True.

Reimplemented from [fcal::ast::Node](#).

7.11.4 Member Data Documentation

7.11.4.1 Expr* fcal::ast::equalsStmt::expr1_ [private]

[Expr](#) representing first element of Matrix if `isMatrix` is True else it represents an [Expr](#) to be assigned to a variable

7.11.4.2 Expr* fcal::ast::equalsStmt::expr2_ [private]

[Expr](#) representing second element of Matrix if `isMatrix` is True.

7.11.4.3 Expr* fcal::ast::equalsStmt::expr3_ [private]

[Expr](#) which will b assigned to a position in the Matrix if `isMatrix` is True

7.11.4.4 bool fcal::ast::equalsStmt::isMatrix_ [private]

`isMatrix` distinguishes between regular variable or matrix constructor

7.11.4.5 varName* fcal::ast::equalsStmt::var_ [private]

variable name or that of a matrix

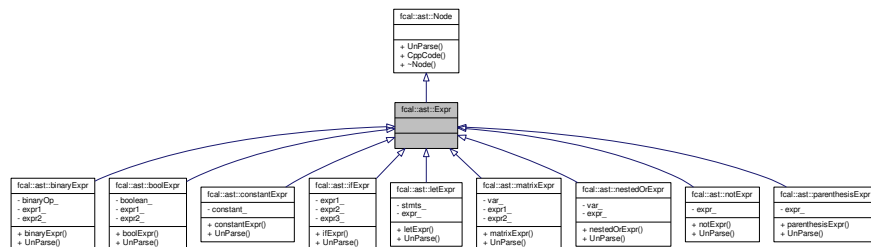
The documentation for this class was generated from the following files:

- [include/ast.h](#)
- [src/ast.cc](#)

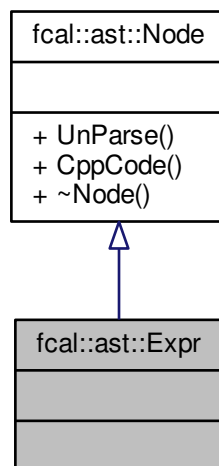
7.12 fcal::ast::Expr Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::Expr:



Collaboration diagram for fcal::ast::Expr:



Additional Inherited Members

7.12.1 Detailed Description

This is the abstract class which inherits from [Node](#). It has no implementation for any functions derived from [Node](#). It is the base class for classes which implement the productions which are derived from the nonterminal '[Expr](#)' in the FCAL grammar.

The documentation for this class was generated from the following file:

- [include/ast.h](#)

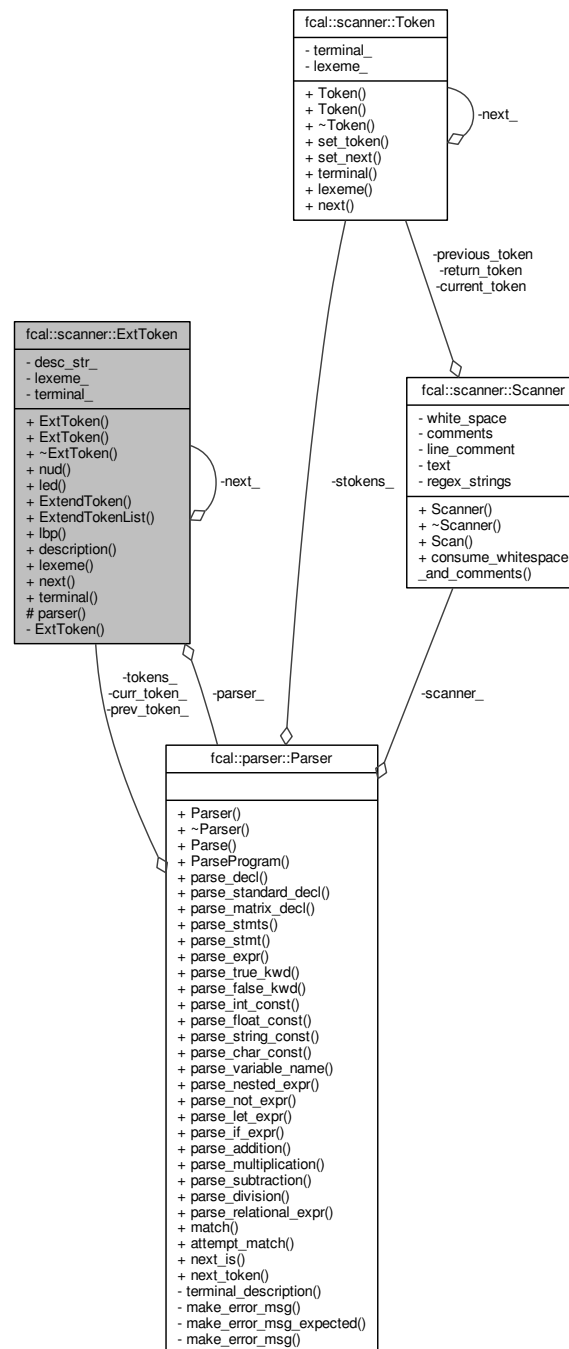
7.13 fcal::scanner::ExtToken Class Reference

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::ExtToken:



Collaboration diagram for `fcsl::scanner::ExtToken`:



Public Member Functions

- `ExtToken (parser::Parser *p, Token *t)`
- `ExtToken (parser::Parser *p, Token *t, std::string d)`
- `virtual ~ExtToken ()`
- `virtual parser::ParseResult nud (void)`
- `virtual parser::ParseResult led (parser::ParseResult left)`

- [ExtToken](#) * [ExtendToken](#) ([parser::Parser](#) **p*, [Token](#) **tokens*)
- [ExtToken](#) * [ExtendTokenList](#) ([parser::Parser](#) **p*, [Token](#) **tokens*)
- virtual int [lbp](#) ()
- virtual std::string [description](#) ()
- std::string [lexeme](#) (void) const
- [ExtToken](#) * [next](#) (void) const
- [scanner::TokenType](#) [terminal](#) (void) const

Protected Member Functions

- [parser::Parser](#) * [parser](#) (void)

Private Member Functions

- [ExtToken](#) (void)

Private Attributes

- std::string [desc_str_](#)
- std::string [lexeme_](#)
- [scanner::TokenType](#) [terminal_](#)
- [ExtToken](#) * [next_](#)
- [parser::Parser](#) * [parser_](#)

7.13.1 Constructor & Destructor Documentation

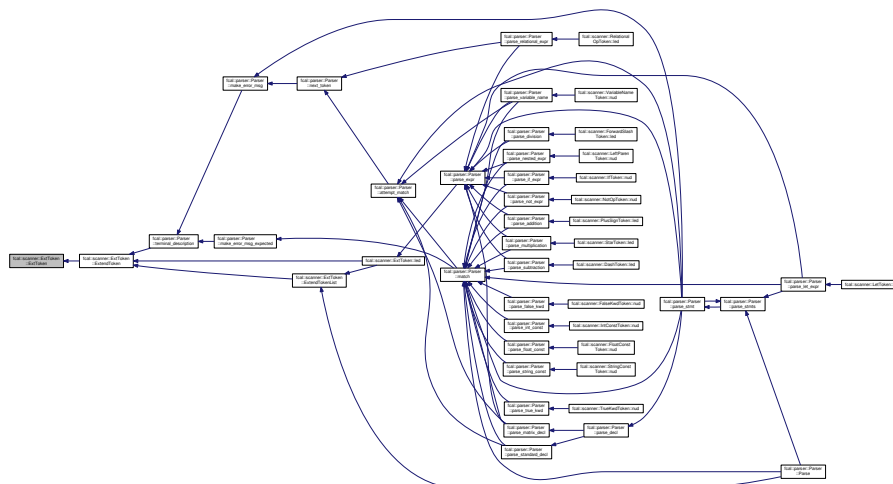
7.13.1.1 `fcal::scanner::ExtToken::ExtToken (parser::Parser * p, Token * t)` `[inline]`

7.13.1.2 `fcal::scanner::ExtToken::ExtToken (parser::Parser * p, Token * t, std::string d)` `[inline]`

7.13.1.3 `virtual fcal::scanner::ExtToken::~~ExtToken ()` `[inline]`, `[virtual]`

7.13.1.4 `fcal::scanner::ExtToken::ExtToken (void)` `[inline]`, `[private]`

Here is the caller graph for this function:

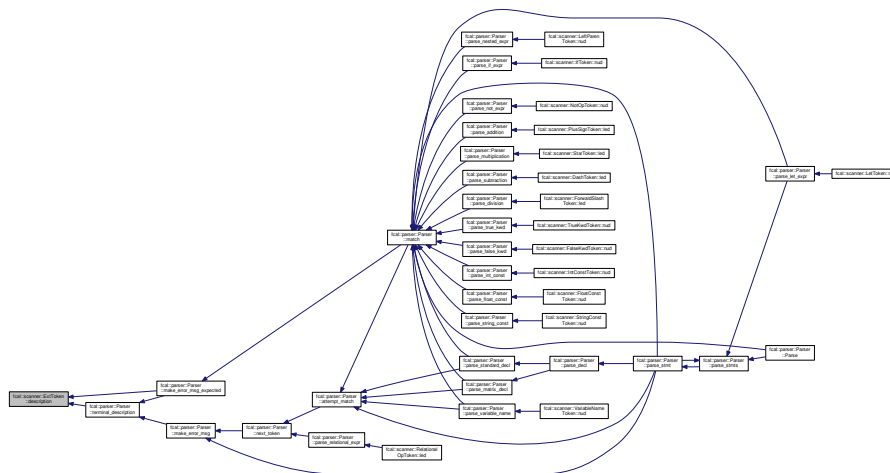


7.13.2 Member Function Documentation

7.13.2.1 `virtual std::string fcal::scanner::ExtToken::description () [inline],[virtual]`

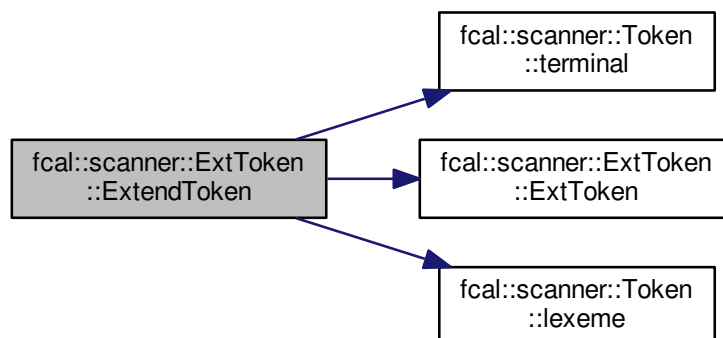
Reimplemented in `fcal::scanner::EndOfFileToken`, `fcal::scanner::ForwardSlashToken`, `fcal::scanner::DashToken`, `fcal::scanner::StarToken`, `fcal::scanner::PlusSignToken`, `fcal::scanner::LeftParenToken`, `fcal::scanner::LetToken`, `fcal::scanner::IfToken`, `fcal::scanner::VariableNameToken`, `fcal::scanner::CharConstToken`, `fcal::scanner::StringConstToken`, `fcal::scanner::FloatConstToken`, `fcal::scanner::IntConstToken`, `fcal::scanner::FalseKwdToken`, `fcal::scanner::TrueKwdToken`, and `fcal::scanner::NotOpToken`.

Here is the caller graph for this function:

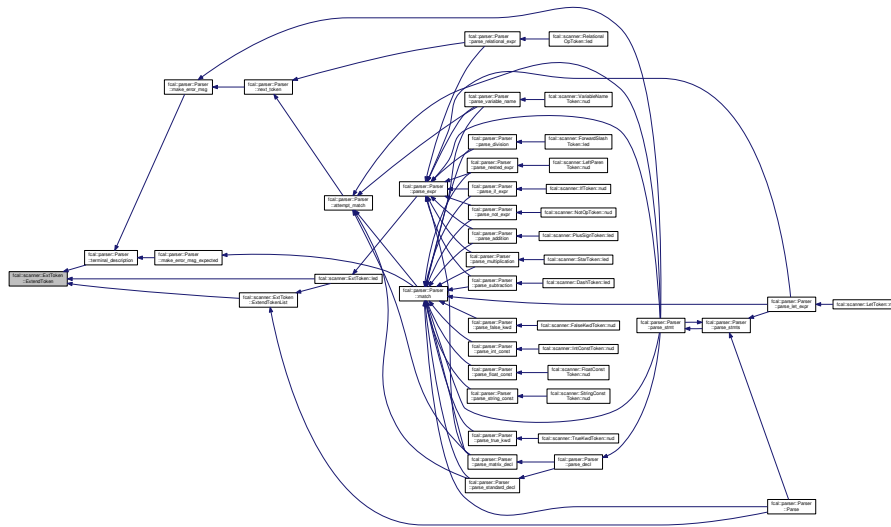


7.13.2.2 `ExtToken * fcal::scanner::ExtToken::ExtendToken (parser::Parser * p, Token * tokens)`

Here is the call graph for this function:

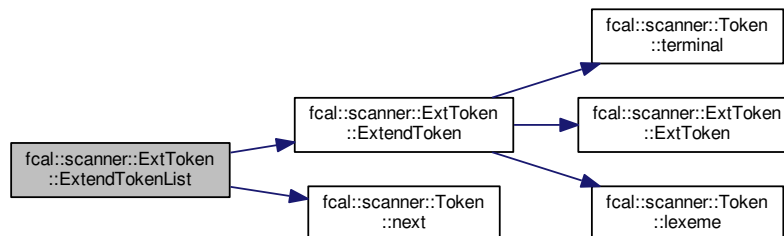


Here is the caller graph for this function:

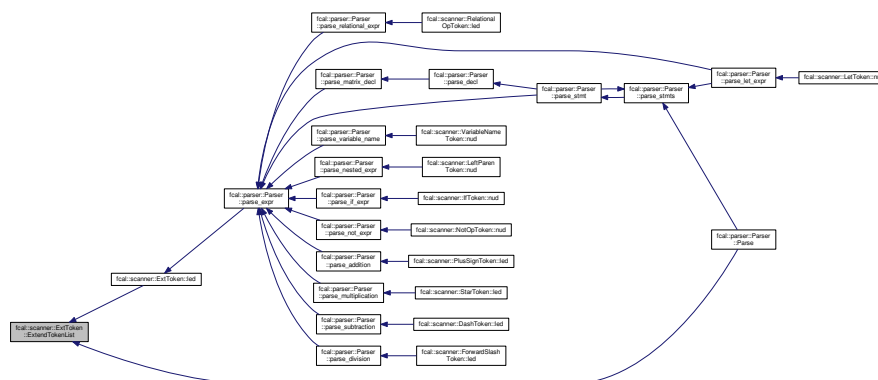


7.13.2.3 ExtToken * fcal::scanner::ExtToken::ExtendTokenList (parser::Parser * p, Token * tokens)

Here is the call graph for this function:



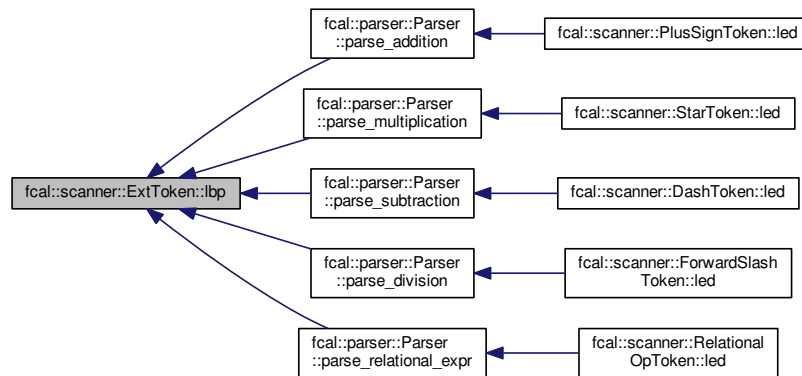
Here is the caller graph for this function:



7.13.2.4 `virtual int fcal::scanner::ExtToken::lbp () [inline],[virtual]`

Reimplemented in [fcal::scanner::RelationalOpToken](#), [fcal::scanner::ForwardSlashToken](#), [fcal::scanner::DashToken](#), [fcal::scanner::StarToken](#), [fcal::scanner::PlusSignToken](#), [fcal::scanner::LeftParenToken](#), [fcal::scanner::LetToken](#), and [fcal::scanner::IfToken](#).

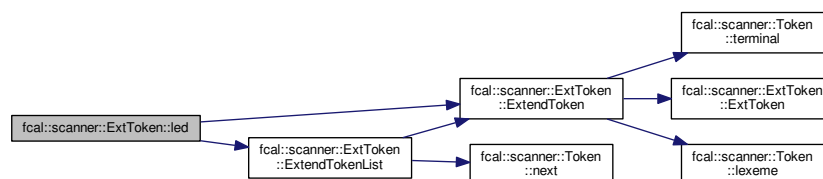
Here is the caller graph for this function:

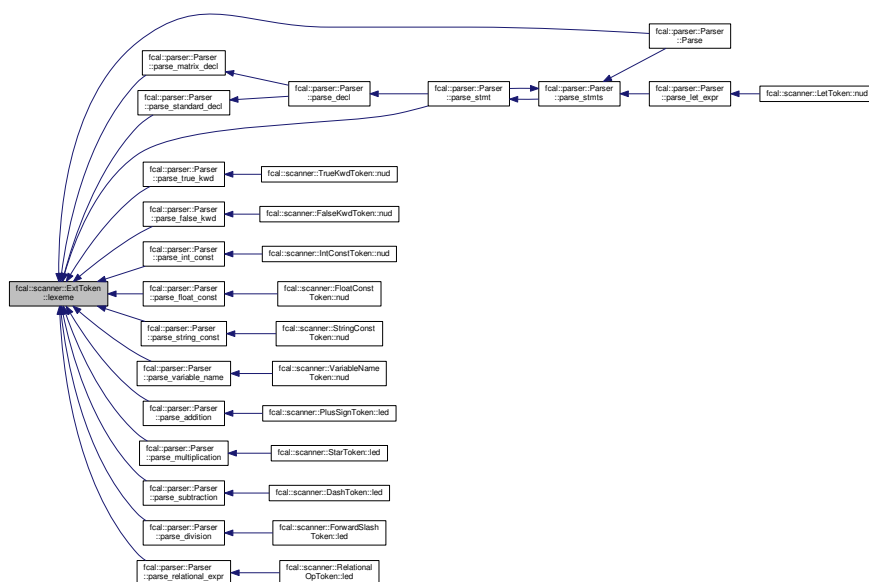
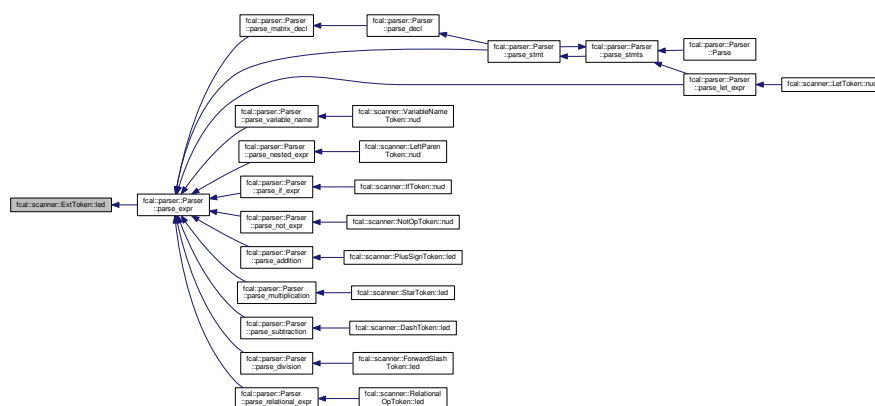


7.13.2.5 `virtual parser::ParseResult fcal::scanner::ExtToken::led (parser::ParseResult left) [inline],[virtual]`

Reimplemented in [fcal::scanner::RelationalOpToken](#), [fcal::scanner::ForwardSlashToken](#), [fcal::scanner::DashToken](#), [fcal::scanner::StarToken](#), and [fcal::scanner::PlusSignToken](#).

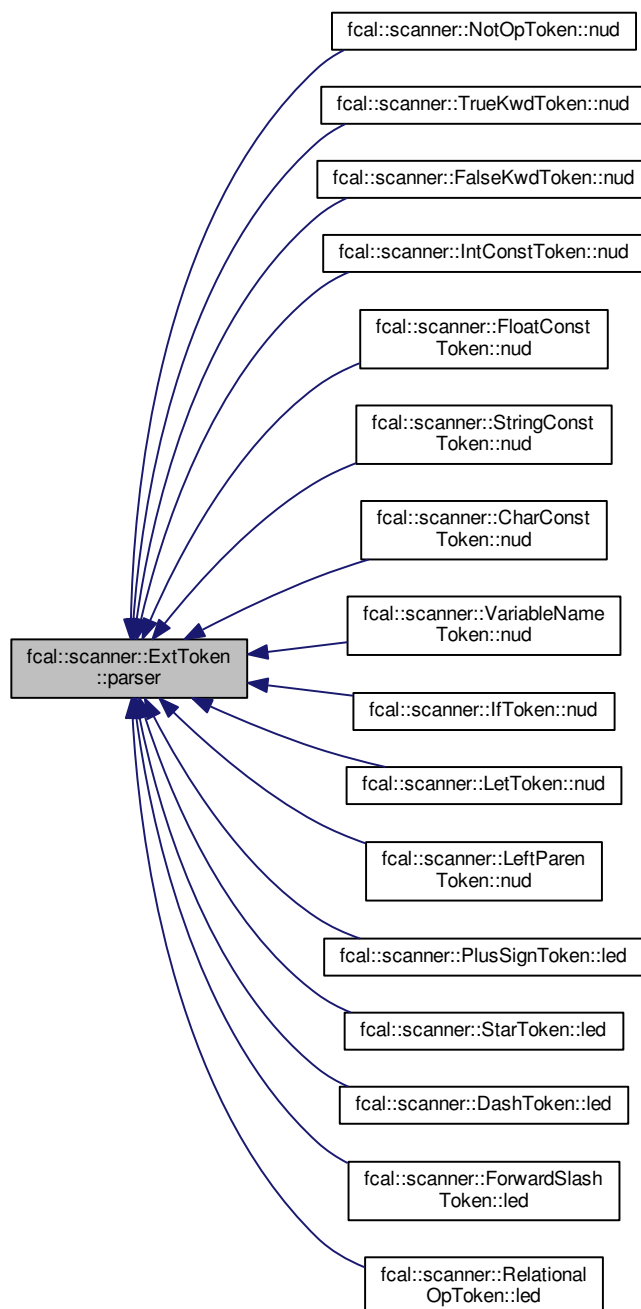
Here is the call graph for this function:





7.13.2.9 `parser::Parser* fcal::scanner::ExtToken::parser (void) [inline],[protected]`

Here is the caller graph for this function:

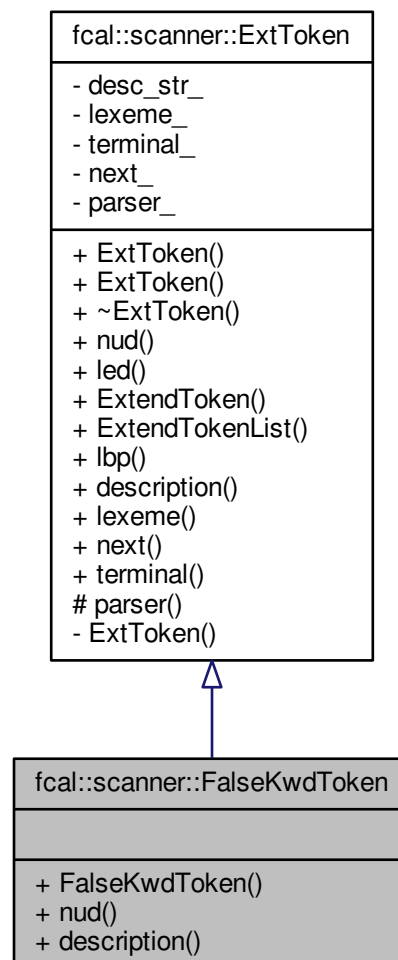


7.14 fcal::scanner::FalseKwdToken Class Reference

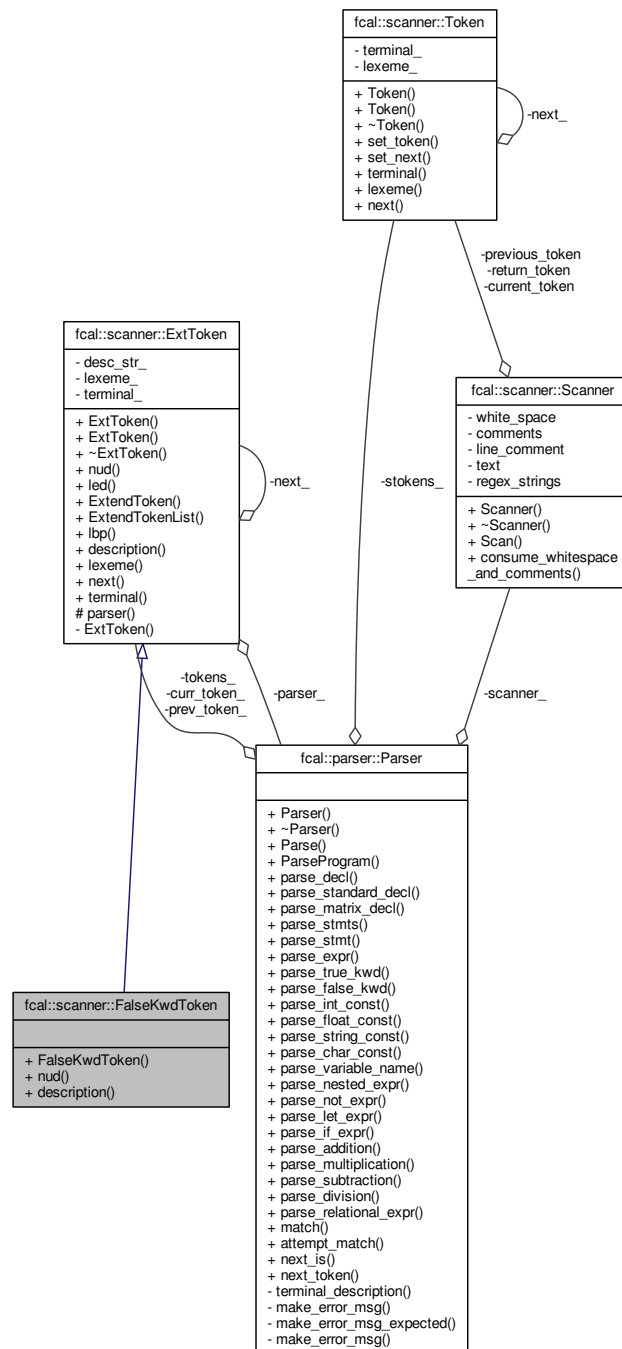
False Kwd.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::FalseKwdToken:



Collaboration diagram for `fcsl::scanner::FalseKwdToken`:



Public Member Functions

- `FalseKwdToken (parser::Parser *p, Token *t)`
- `parser::ParseResult nud ()`
- `std::string description ()`

Additional Inherited Members

7.14.1 Detailed Description

False Kwd.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 `fcal::scanner::FalseKwdToken::FalseKwdToken (parser::Parser * p, Token * t)` `[inline]`

7.14.3 Member Function Documentation

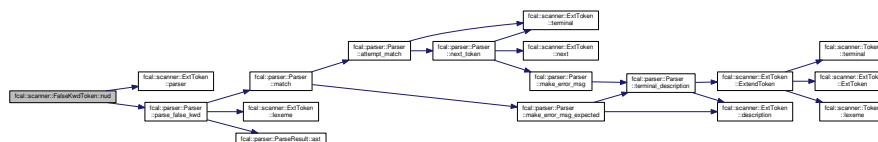
7.14.3.1 `std::string fcal::scanner::FalseKwdToken::description ()` `[inline]`, `[virtual]`

Reimplemented from [fcal::scanner::ExtToken](#).

7.14.3.2 `parser::ParseResult fcal::scanner::FalseKwdToken::nud (void)` `[inline]`, `[virtual]`

Reimplemented from [fcal::scanner::ExtToken](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

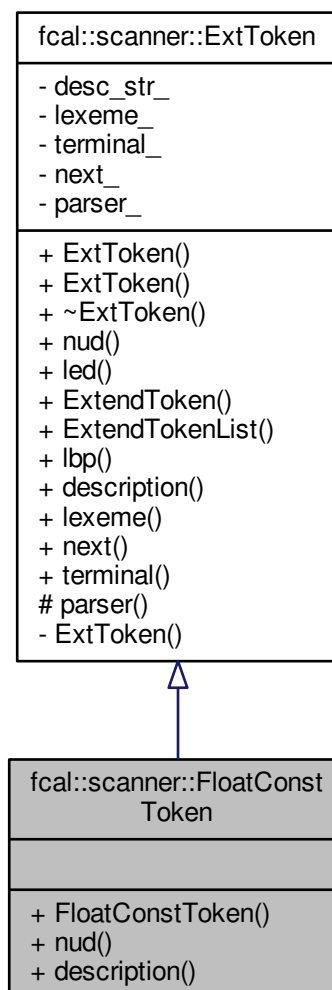
- [include/ext_token.h](#)

7.15 fcal::scanner::FloatConstToken Class Reference

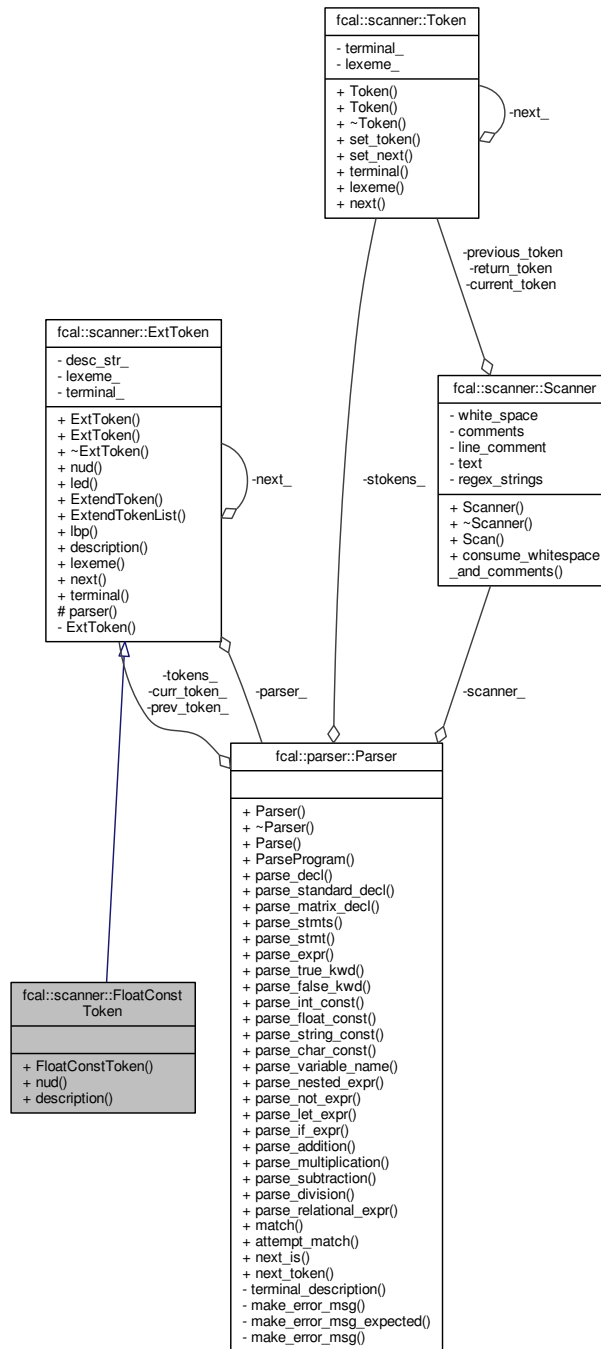
Float Const.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::FloatConstToken:



Collaboration diagram for fcal::scanner::FloatConstToken:



Public Member Functions

- [FloatConstToken](#) ([parser::Parser](#) *p, [Token](#) *t)
- [parser::ParseResult nud](#) ()
- [std::string description](#) ()

Additional Inherited Members

7.15.1 Detailed Description

Float Const.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 `fcal::scanner::FloatConstToken::FloatConstToken (parser::Parser * p, Token * t)` `[inline]`

7.15.3 Member Function Documentation

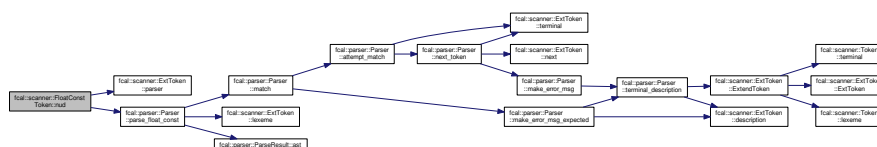
7.15.3.1 `std::string fcal::scanner::FloatConstToken::description ()` `[inline]`, `[virtual]`

Reimplemented from [fcal::scanner::ExtToken](#).

7.15.3.2 `parser::ParseResult fcal::scanner::FloatConstToken::nud (void)` `[inline]`, `[virtual]`

Reimplemented from [fcal::scanner::ExtToken](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

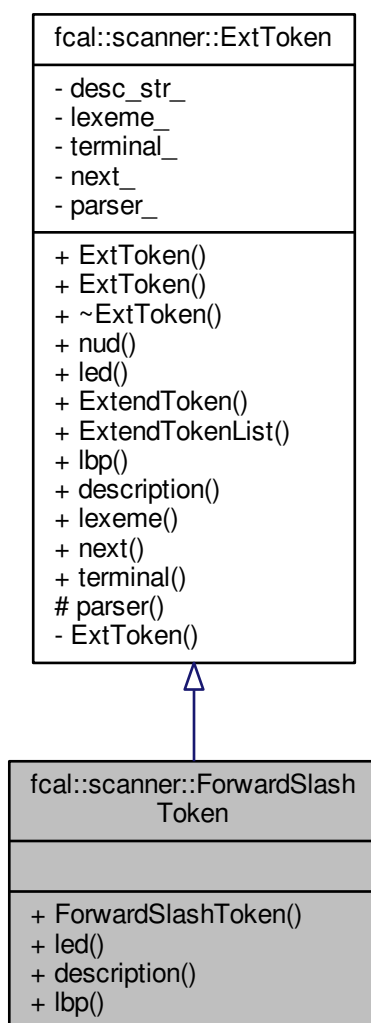
- [include/ext_token.h](#)

7.16 fcal::scanner::ForwardSlashToken Class Reference

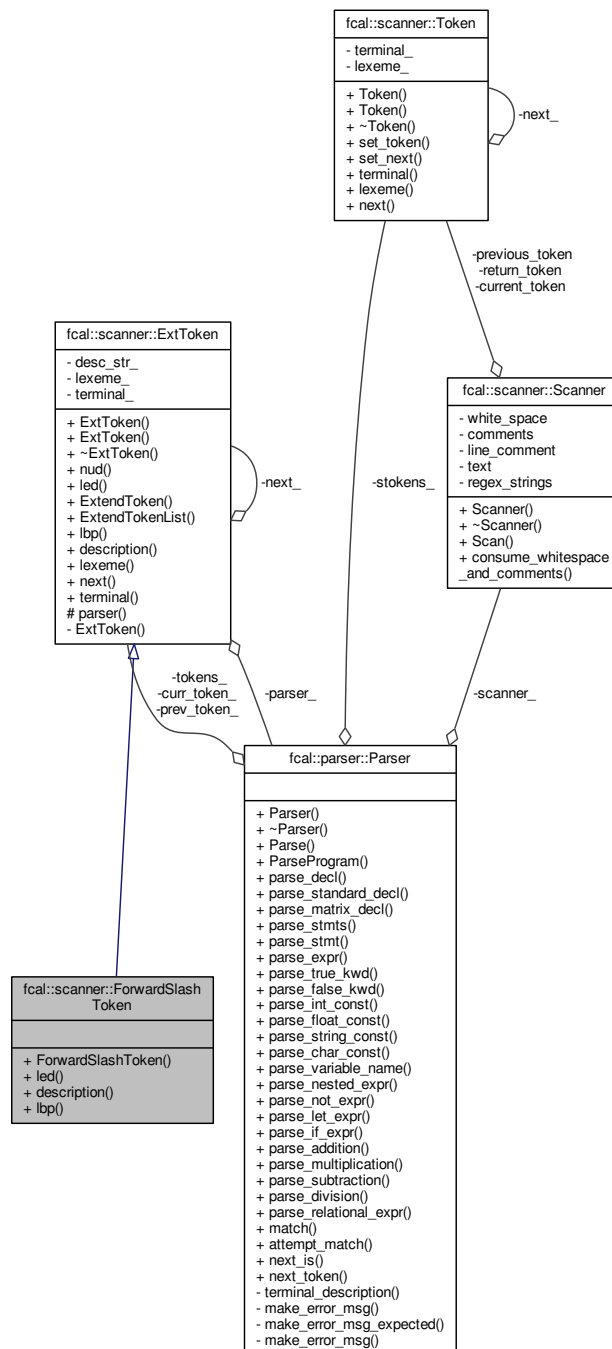
ForwardSlash.

```
#include <ext_token.h>
```


Inheritance diagram for fcal::scanner::ForwardSlashToken:



Collaboration diagram for `fcsl::scanner::ForwardSlashToken`:



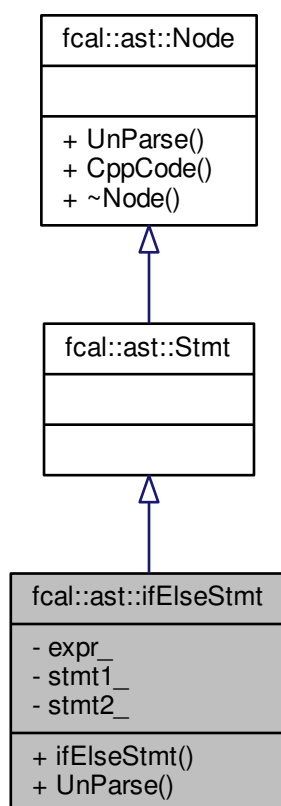
Public Member Functions

- [ForwardSlashToken](#) ([parser::Parser *p](#), [Token *t](#))
- [parser::ParseResult led](#) ([parser::ParseResult left](#))
- [std::string description](#) ()
- [int lbp](#) ()

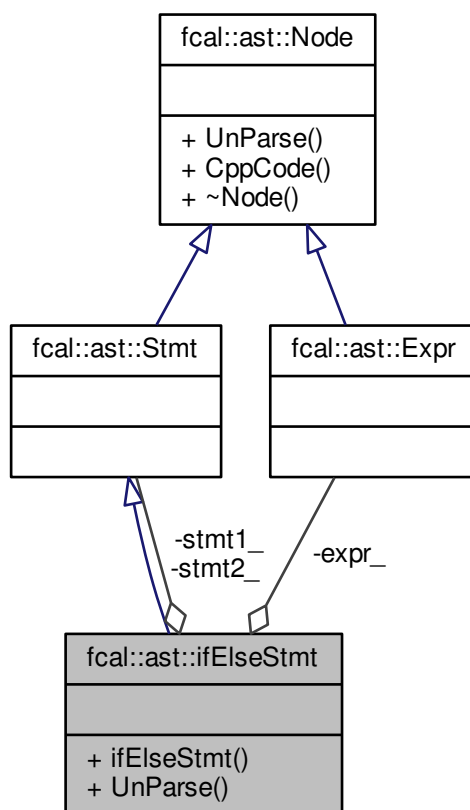
7.17 fcal::ast::ifElseStmt Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::ifElseStmt:



Collaboration diagram for fcal::ast::ifElseStmt:



Public Member Functions

- `ifElseStmt (Expr *expr, Stmt *stmt1, Stmt *stmt2)`
- `std::string UnParse (void)`

Private Attributes

- `Expr * expr_`
Expr to be evaluated as a condition for if-else.
- `Stmt * stmt1_`
Stmt to be executed if expr_ is True.
- `Stmt * stmt2_`
Stmt to be executed if expr_ is False.

7.17.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
`Stmt ::= 'if' '(' Expr ')' Stmt 'else' Stmt`

7.17.2 Constructor & Destructor Documentation

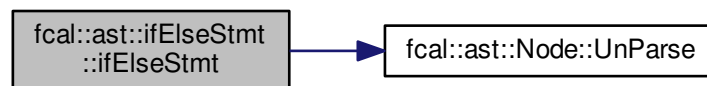
7.17.2.1 `fcal::ast::ifElseStmt::ifElseStmt (Expr * expr, Stmt * stmt1, Stmt * stmt2)` `[inline]`

This constructor takes two parameters

Parameters

<i>expr</i>	a Expr* holding the address which contains information regarding an Expr to be evaluated as condition for if-else Stmt
<i>stmt1</i>	a Stmt* holding the address which contains information regarding a Stmt to be executed if <i>expr</i> is True
<i>stmt2</i>	a Stmt* holding the address which contains the information regarding a Stmt to be executed if <i>expr</i> is False

Here is the call graph for this function:



7.17.3 Member Function Documentation

7.17.3.1 `std::string fcal::ast::ifElseStmt::UnParse (void)` `[virtual]`

Returns the string : "if (" + *expr_*->[UnParse\(\)](#) + ")\n"

- *stmt1_*->[UnParse\(\)](#) + "else " + *stmt2_*->[UnParse\(\)](#)

Reimplemented from [fcal::ast::Node](#).

7.17.4 Member Data Documentation

7.17.4.1 `Expr* fcal::ast::ifElseStmt::expr_` `[private]`

[Expr](#) to be evaluated as a condition for if-else.

7.17.4.2 `Stmt* fcal::ast::ifElseStmt::stmt1_` `[private]`

[Stmt](#) to be executed if *expr_* is True.

7.17.4.3 Stmt* fcal::ast::ifElseStmt::stmt2_ [private]

[Stmt](#) to be executed if `expr_` is False.

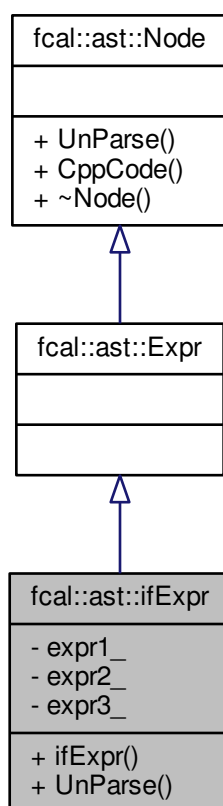
The documentation for this class was generated from the following files:

- include/[ast.h](#)
- src/[ast.cc](#)

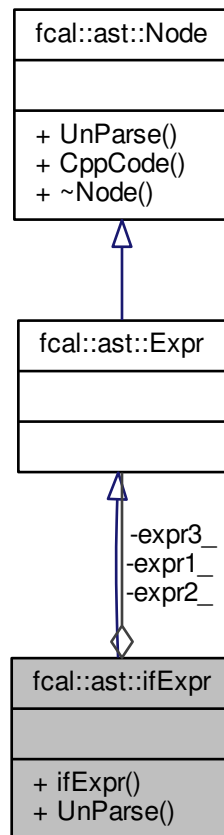
7.18 fcal::ast::ifExpr Class Reference

```
#include <ast.h>
```

Inheritance diagram for `fcal::ast::ifExpr`:



Collaboration diagram for `fcgal::ast::ifExpr`:



Public Member Functions

- `ifExpr` (`Expr *expr1`, `Expr *expr2`, `Expr *expr3`)
- `std::string UnParse` (`void`)

Private Attributes

- `Expr * expr1_`
Expr to be evaluated as a condition.
- `Expr * expr2_`
Expr to be evaluated if expr1_ is True.
- `Expr * expr3_`
Expr to be evaluated if expr1_ is False.

7.18.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
`Expr ::= 'if' Expr 'then' Expr 'else' Expr`

7.18.2 Constructor & Destructor Documentation

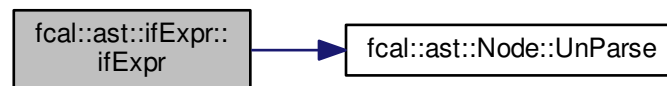
7.18.2.1 fcal::ast::ifExpr::ifExpr (Expr * *expr1*, Expr * *expr2*, Expr * *expr3*) [inline]

This constructor takes three parameters

Parameters

<i>expr1</i>	Expr to be evaluated as a condition
<i>expr2</i>	Expr to be evaluated if <i>expr1</i> is True
<i>expr3</i>	Expr to be evaluated if <i>expr1</i> is False

Here is the call graph for this function:



7.18.3 Member Function Documentation

7.18.3.1 std::string fcal::ast::ifExpr::UnParse (void) [virtual]

Returns the string : "if " + *expr1_*->[UnParse\(\)](#) + " then " + *expr2_*->[UnParse\(\)](#) + " else " + *expr3_*->[UnParse\(\)](#);

Reimplemented from [fcal::ast::Node](#).

7.18.4 Member Data Documentation

7.18.4.1 Expr* fcal::ast::ifExpr::expr1_ [private]

[Expr](#) to be evaluated as a condition.

7.18.4.2 Expr* fcal::ast::ifExpr::expr2_ [private]

[Expr](#) to be evaluated if *expr1_* is True.

7.18.4.3 Expr* fcal::ast::ifExpr::expr3_ [private]

[Expr](#) to be evaluated if *expr1_* is False.

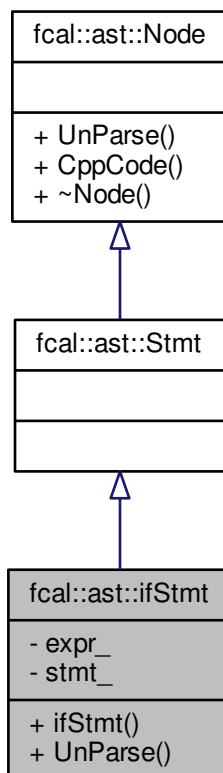
The documentation for this class was generated from the following files:

- include/[ast.h](#)
- src/[ast.cc](#)

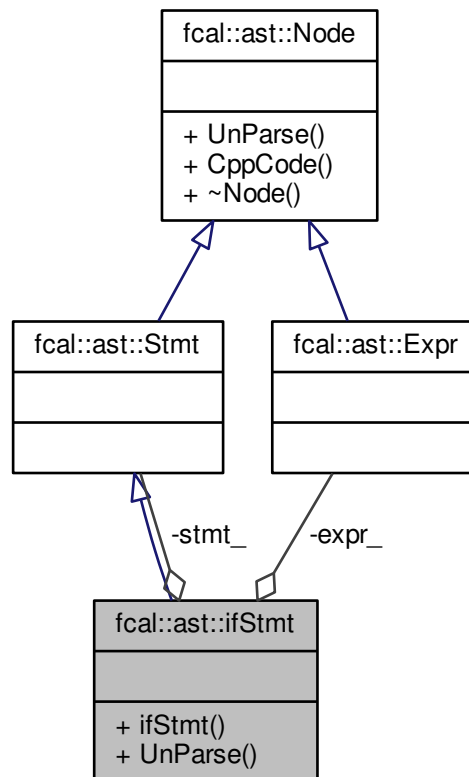
7.19 fcal::ast::ifStmt Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::ifStmt:



Collaboration diagram for fcal::ast::ifStmt:



Public Member Functions

- [ifStmt](#) ([Expr](#) *expr, [Stmt](#) *stmt)
- [std::string UnParse](#) (void)
Returns the string : "if (" + [expr_](#) -> [UnParse\(\)](#) + ")" + [stmt_](#) -> [UnParse\(\)](#)

Private Attributes

- [Expr](#) * [expr_](#)
[Expr](#) to be evaluated inside the if condition.
- [Stmt](#) * [stmt_](#)
[Stmt](#) to be executed if [expr_](#) is True.

7.19.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
[Stmt](#) ::= 'if' '(' [Expr](#) ')' [Stmt](#)

7.19.2 Constructor & Destructor Documentation

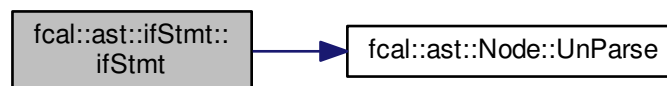
7.19.2.1 `fcal::ast::ifStmt::ifStmt (Expr * expr, Stmt * stmt)` `[inline]`

This constructor takes two parameters

Parameters

<i>expr</i>	a Expr* holding address which contains details of an Expr inside an if condition
<i>stmt</i>	a Stmt* holding address which contains details of a Stmt which is executed if expr is True

Here is the call graph for this function:



7.19.3 Member Function Documentation

7.19.3.1 `std::string fcal::ast::ifStmt::UnParse (void)` `[virtual]`

Returns the string : "if (" + `expr_ -> UnParse()` + ")" + `stmt_ -> UnParse()`

Reimplemented from [fcal::ast::Node](#).

7.19.4 Member Data Documentation

7.19.4.1 `Expr* fcal::ast::ifStmt::expr_` `[private]`

[Expr](#) to be evaluated inside the if condition.

7.19.4.2 `Stmt* fcal::ast::ifStmt::stmt_` `[private]`

[Stmt](#) to be executed if `expr_` is True.

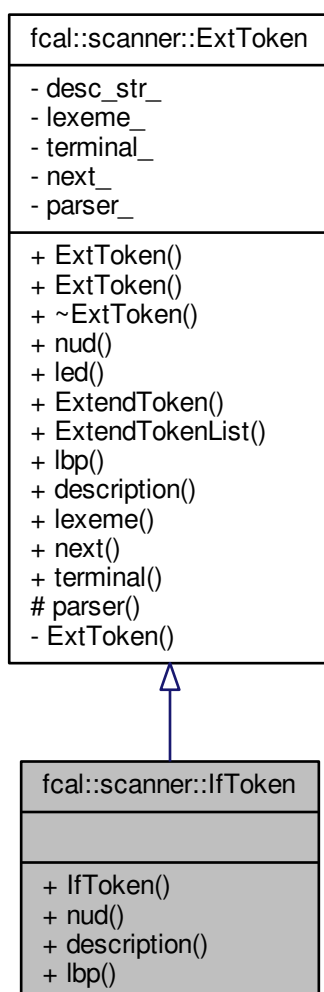
The documentation for this class was generated from the following files:

- [include/ast.h](#)
- [src/ast.cc](#)

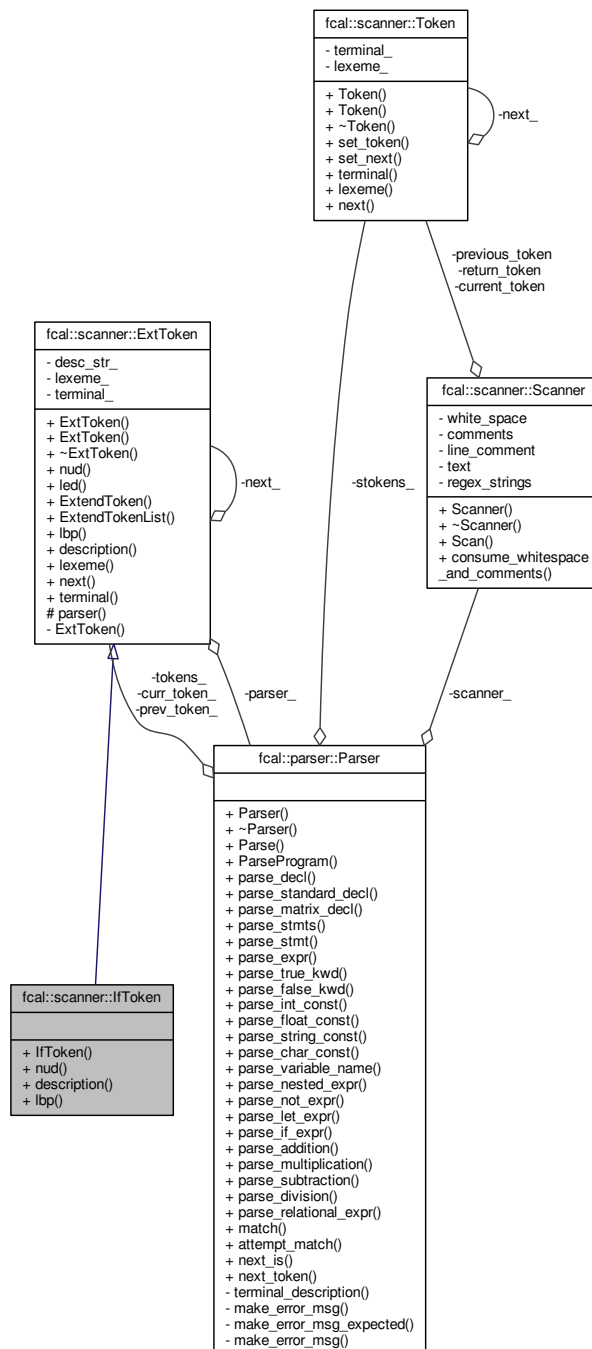
7.20 fcal::scanner::IfToken Class Reference

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::IfToken:



Collaboration diagram for `fcsl::scanner::IfToken`:



Public Member Functions

- `IfToken (parser::Parser *p, Token *t)`
- `parser::ParseResult nud ()`
- `std::string description ()`
- `int lbp ()`

7.20.1 Constructor & Destructor Documentation

7.20.2 Member Function Documentation

Reimplemented from `fcal::scanner::ExtToken`.

Reimplemented from `fcml::scanner::ExtToken`.

Reimplemented from `fcml::scanner::ExtToken`.

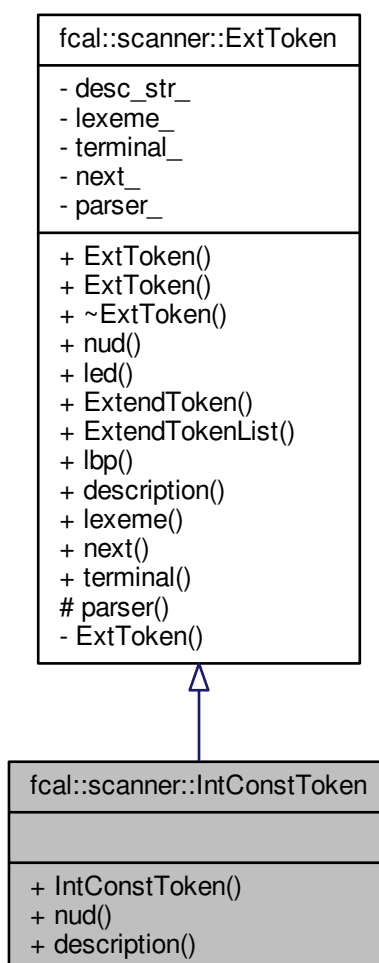
[illegible]

- `include/ext_token.h`

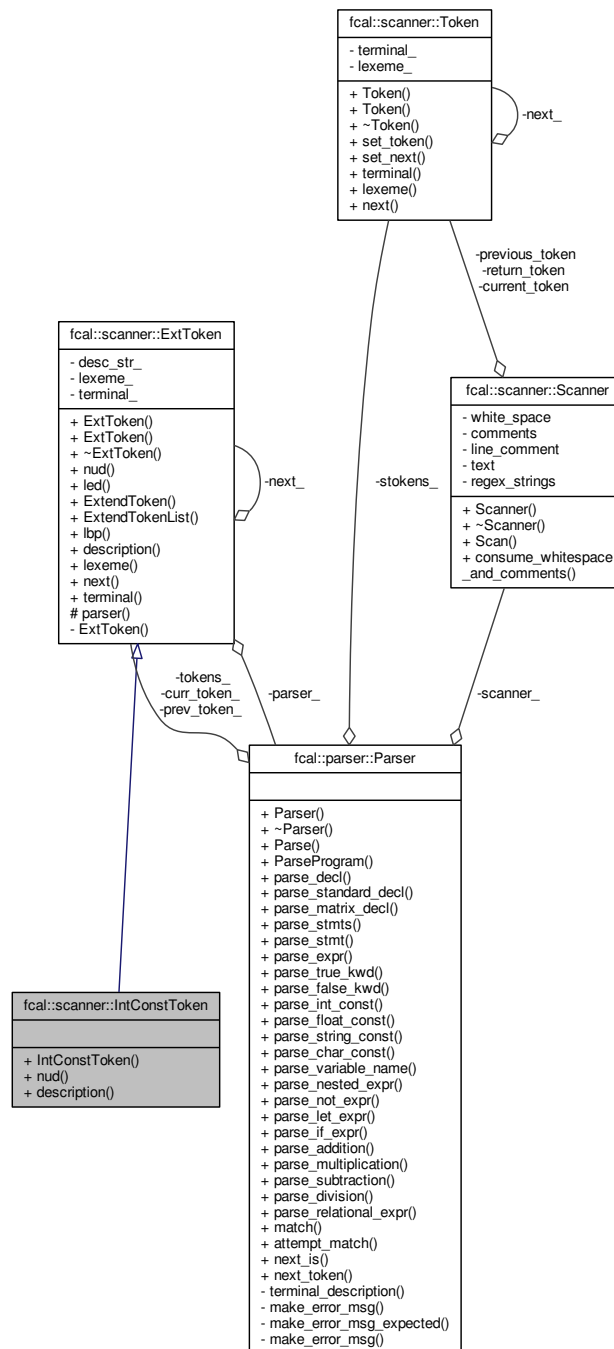
Int Const.

Generated by Doxygen

Inheritance diagram for fcal::scanner::IntConstToken:



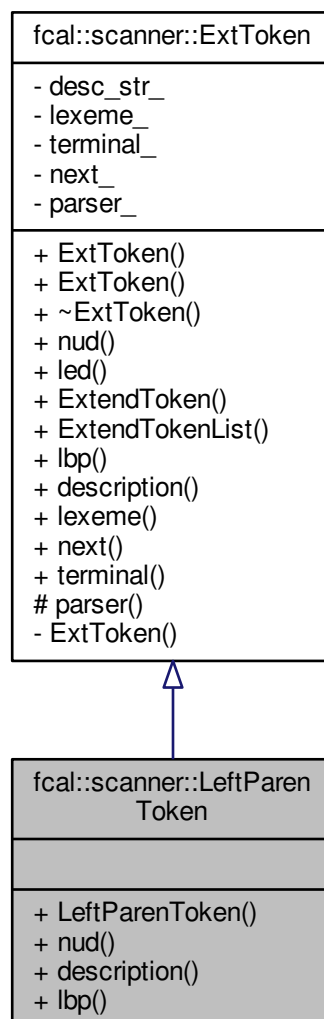
Collaboration diagram for fcal::scanner::IntConstToken:



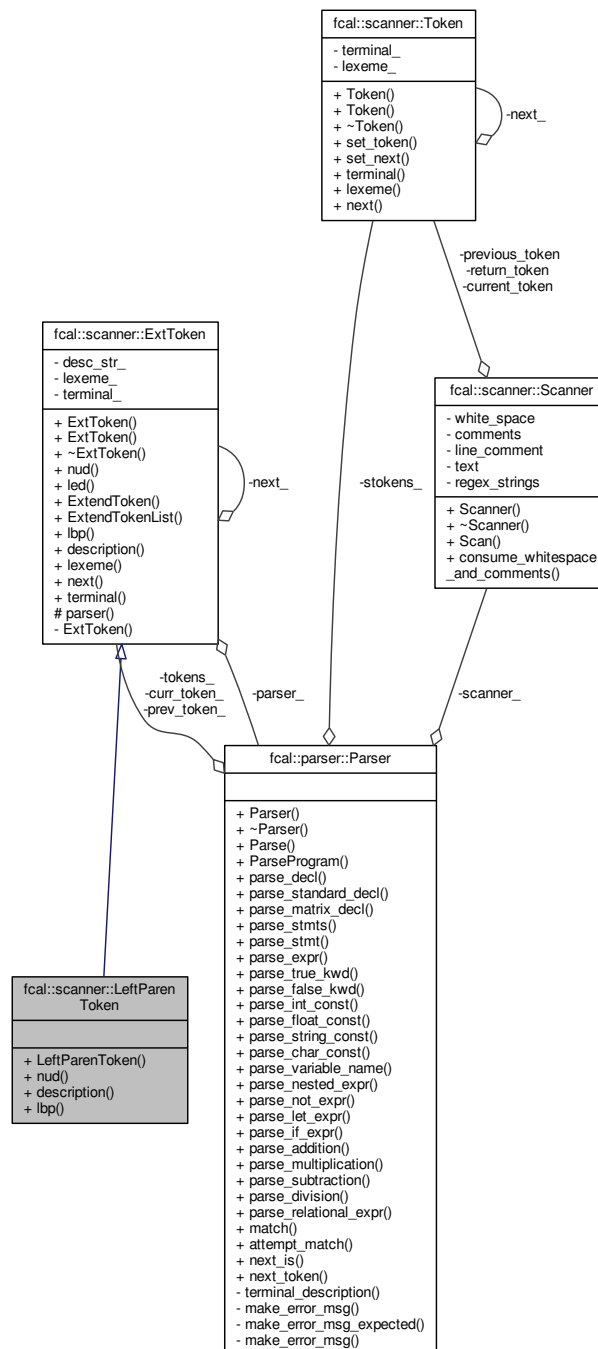
Public Member Functions

- [IntConstToken](#) ([parser::Parser *p](#), [Token *t](#))
- [parser::ParseResult nud](#) ()
- [std::string description](#) ()

Inheritance diagram for fcal::scanner::LeftParenToken:



Collaboration diagram for `fcsl::scanner::LeftParenToken`:



Public Member Functions

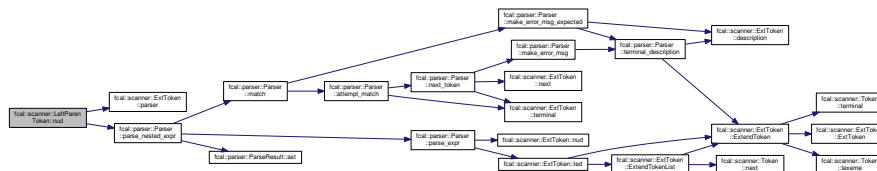
- [LeftParenToken](#) ([parser::Parser](#) *p, [Token](#) *t)
- [parser::ParseResult](#) nud ()
- [std::string](#) description ()
- [int](#) lbp ()

7.22.1 Detailed Description

7.22.2 Constructor & Destructor Documentation

7.22.3 Member Function Documentation

Here is the call graph for this function:

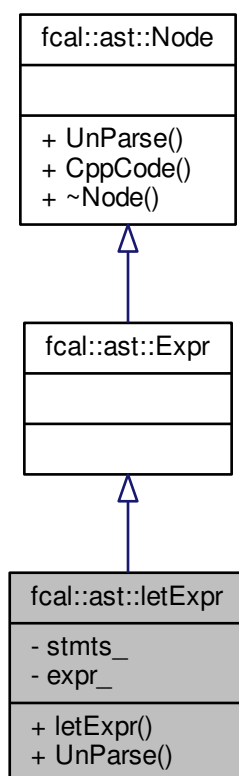


- include/ext_token.h

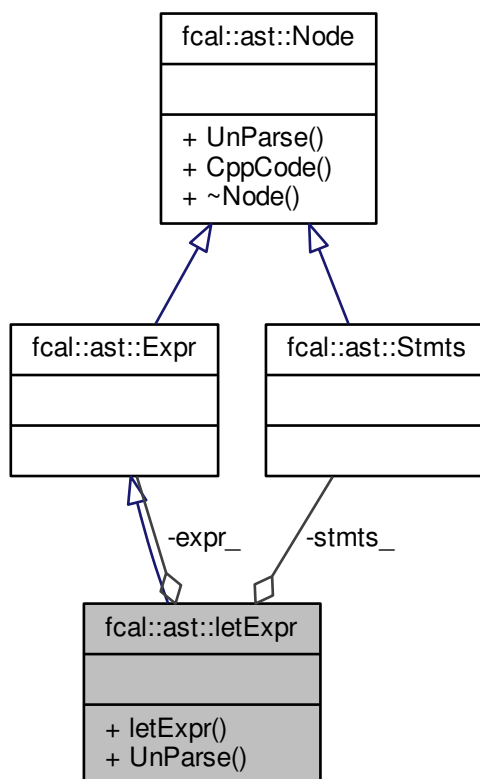
7.23 fcal::ast::letExpr Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::letExpr:



Collaboration diagram for fcal::ast::letExpr:



Public Member Functions

- [letExpr](#) ([Stmts](#) *stmts, [Expr](#) *expr)
- std::string [UnParse](#) (void)

Private Attributes

- [Stmts](#) * [stmts_](#)
Stmts in a Let.
- [Expr](#) * [expr_](#)
Expr in a let.

7.23.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
[Expr](#) ::= 'let' [Stmts](#) 'in' [Expr](#) 'end'

7.23.2 Constructor & Destructor Documentation

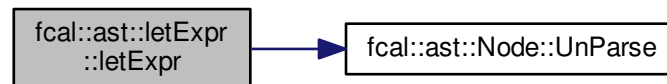
7.23.2.1 `fcal::ast::letExpr::letExpr (Stmt * stmts, Expr * expr) [inline]`

This constructor takes two parameters

Parameters

<i>stmts</i>	Stmts in a Let
<i>expr</i>	Expr in a Let

Here is the call graph for this function:



7.23.3 Member Function Documentation

7.23.3.1 `std::string fcal::ast::letExpr::UnParse (void)` [virtual]

Returns the string : "let " + stmts_ -> [UnParse\(\)](#) + " in " + expr_ -> [UnParse\(\)](#) + " end"

Reimplemented from [fcal::ast::Node](#).

7.23.4 Member Data Documentation

7.23.4.1 `Expr* fcal::ast::letExpr::expr_` [private]

[Expr](#) in a let.

7.23.4.2 `Stmts* fcal::ast::letExpr::stmts_` [private]

[Stmts](#) in a Let.

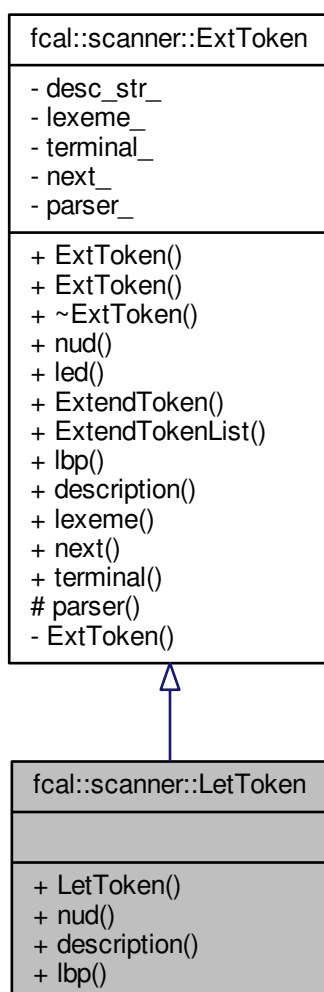
The documentation for this class was generated from the following files:

- [include/ast.h](#)
- [src/ast.cc](#)

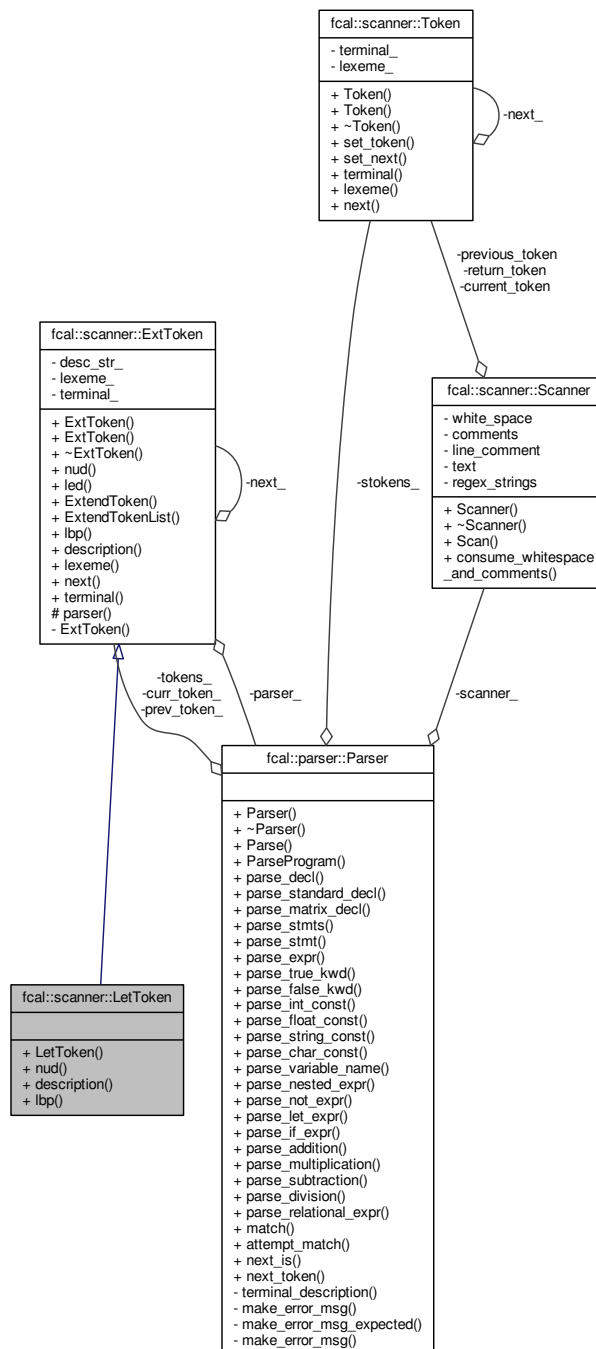
7.24 fcal::scanner::LetToken Class Reference

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::LetToken:



Collaboration diagram for fcal::scanner::LetToken:



Public Member Functions

- `LetToken` (`parser::Parser *p`, `Token *t`)
- `parser::ParseResult nud` ()
- `std::string description` ()
- `int lbp` ()

Additional Inherited Members

7.24.1 Constructor & Destructor Documentation

7.24.1.1 `fcal::scanner::LetToken::LetToken (parser::Parser * p, Token * t)` `[inline]`

7.24.2 Member Function Documentation

7.24.2.1 `std::string fcal::scanner::LetToken::description ()` `[inline],[virtual]`

Reimplemented from [fcal::scanner::ExtToken](#).

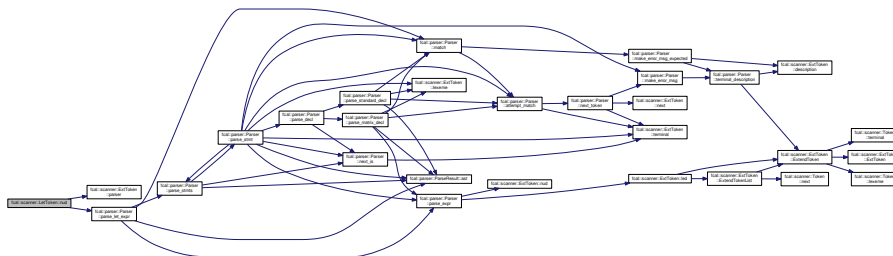
7.24.2.2 `int fcal::scanner::LetToken::lbp ()` `[inline],[virtual]`

Reimplemented from [fcal::scanner::ExtToken](#).

7.24.2.3 `parser::ParseResult fcal::scanner::LetToken::nud (void)` `[inline],[virtual]`

Reimplemented from [fcal::scanner::ExtToken](#).

Here is the call graph for this function:



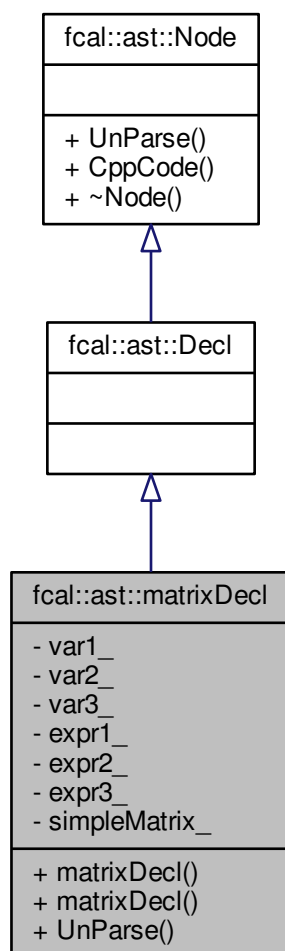
The documentation for this class was generated from the following file:

- [include/ext_token.h](#)

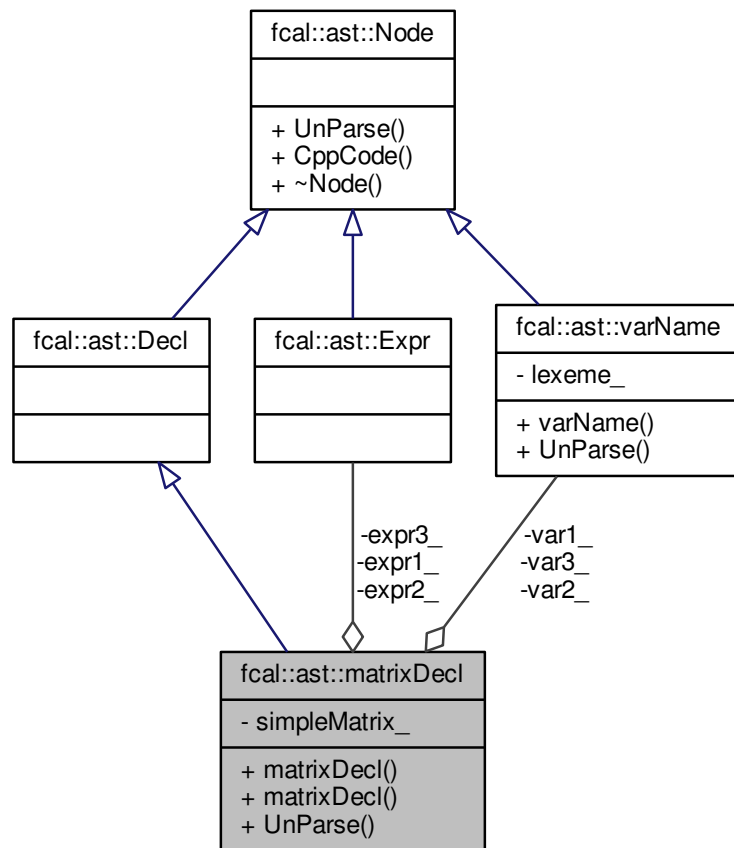
7.25 fcal::ast::matrixDecl Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::matrixDecl:



Collaboration diagram for `fcf::ast::matrixDecl`:



Public Member Functions

- `matrixDecl` (`varName` *var1, `Expr` *expr1, bool simpleMatrix)
- `matrixDecl` (`varName` *var1, `varName` *var2, `varName` *var3, `Expr` *expr1, `Expr` *expr2, `Expr` *expr3, bool simpleMatrix)
- `std::string UnParse` (void)

Private Attributes

- `varName` * var1_
varName representing Matrix name
- `varName` * var2_
varName representing Matrix row if `simpleMatrix_` is False
- `varName` * var3_
varName representing Matrix column if `simpleMatrix_` is False
- `Expr` * expr1_
- `Expr` * expr2_

- expr assigned to column of Matrix if simpleMatrix_ is False*
- [Expr](#) * [expr3_](#)
expr assigned to matrix if simpleMatrix_ is False
- bool [simpleMatrix_](#)
simpleMatrix distinguishes which constructor is used

7.25.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,

`Decl ::= 'matrix' varName '[' Expr ':' Expr ']' varName '=' Expr ;'`

`Decl ::= 'matrix' varName '=' Expr ;'`

7.25.2 Constructor & Destructor Documentation

7.25.2.1 `fcal::ast::matrixDecl::matrixDecl (varName * var1, Expr * expr1, bool simpleMatrix)` `[inline]`

This constructor takes three parameters

Parameters

<i>var1</i>	a varName repersenting the matrix
<i>expr1</i>	expression assigned to a matrix
<i>simpleMatrix</i>	bool which is True if Matrix Declaration is <code>Decl ::= 'matrix' varName '=' Expr ;'</code>

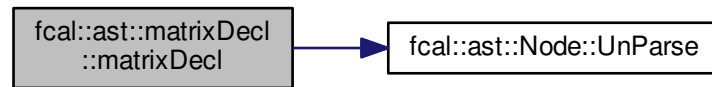
7.25.2.2 `fcal::ast::matrixDecl::matrixDecl (varName * var1, varName * var2, varName * var3, Expr * expr1, Expr * expr2, Expr * expr3, bool simpleMatrix)` `[inline]`

This constructor takes seven parameters

Parameters

<i>var1</i>	a varName repersenting the matrix
<i>var2</i>	a varName repersenting row of matrix
<i>var3</i>	a varName repersenting column of matrix
<i>expr1</i>	expression for row of Matrix
<i>expr2</i>	expression for column of Matrix
<i>expr3</i>	expression assigned to the matrix
<i>simpleMatrix</i>	bool which is False if Matrix Declaration is <code>Decl ::= 'matrix' varName '[' Expr ':' Expr ']' varName '=' Expr ;'</code>

Here is the call graph for this function:



7.25.3 Member Function Documentation

7.25.3.1 `std::string fcal::ast::matrixDecl::UnParse (void) [virtual]`

If `simpleMatrix_` is True then return string : "matrix " + `var1_->UnParse()` + " = " + `expr1_->UnParse()` + " ;"
 Else return : "matrix " + `var1_->UnParse()` + "[" + `expr1_->UnParse()` + ":"

- `expr2_->UnParse()` + "]" + `var2_->UnParse()` + ":" + `var3_->UnParse()` + " = "
- `expr3_->UnParse()` + " ;"

Reimplemented from `fcal::ast::Node`.

7.25.4 Member Data Documentation

7.25.4.1 `Expr* fcal::ast::matrixDecl::expr1_ [private]`

`expr` assigned to matrix if `simpleMatrix_` is True `expr` assigned to row of Matrix if `simpleMatrix_` is False

7.25.4.2 `Expr* fcal::ast::matrixDecl::expr2_ [private]`

`expr` assigned to column of Matrix if `simpleMatrix_` is False

7.25.4.3 `Expr* fcal::ast::matrixDecl::expr3_ [private]`

`expr` assigned to matrix if `simpleMatrix_` is False

7.25.4.4 `bool fcal::ast::matrixDecl::simpleMatrix_ [private]`

`simpleMatrix` distinguishes which constructor is used

7.25.4.5 **varName*** fcal::ast::matrixDecl::var1_ [private]

varName representing Matrix name

7.25.4.6 **varName*** fcal::ast::matrixDecl::var2_ [private]

varName representing Matrix row if simpleMatrix_ is False

7.25.4.7 **varName*** fcal::ast::matrixDecl::var3_ [private]

varName representing Matrix column if simpleMatrix_ is False

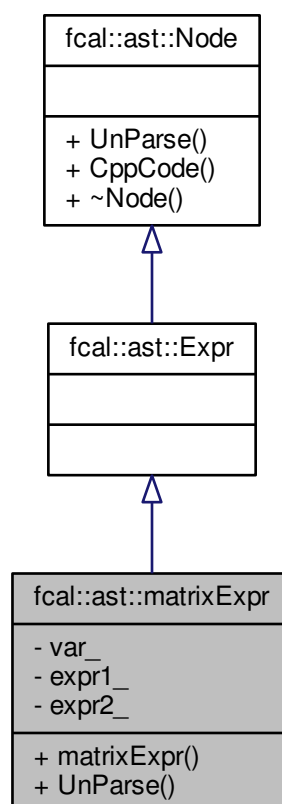
The documentation for this class was generated from the following files:

- include/[ast.h](#)
- src/[ast.cc](#)

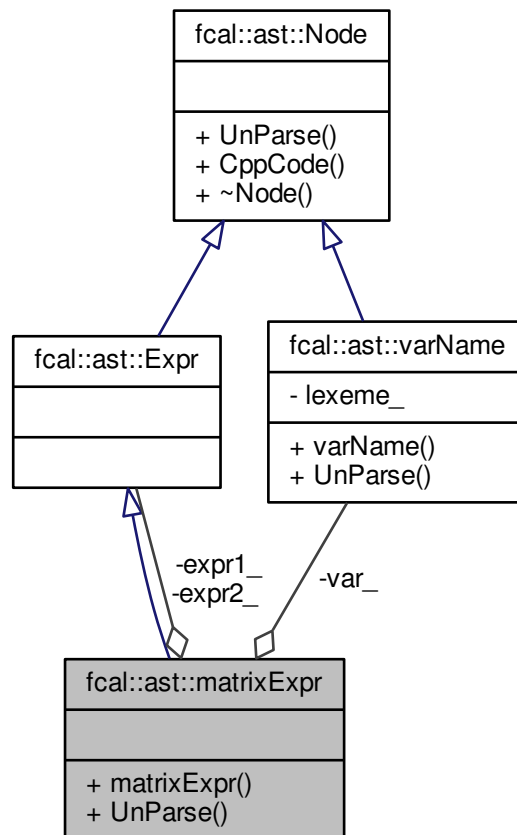
7.26 fcal::ast::matrixExpr Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::matrixExpr:



Collaboration diagram for `fcal::ast::matrixExpr`:



Public Member Functions

- `matrixExpr` (`varName` *var, `Expr` *expr1, `Expr` *expr2)
- `std::string UnParse` (void)

Private Attributes

- `varName` * `var_`
varName representing matrix name
- `Expr` * `expr1_`
Expr which represents row of Matrix.
- `Expr` * `expr2_`
Expr which represents column of Matrix.

7.26.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,

`Expr ::= varName '[' Expr ':' Expr ']'`

7.26.2 Constructor & Destructor Documentation

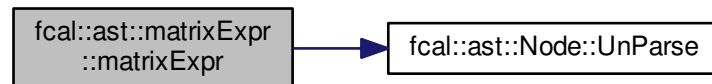
7.26.2.1 `fcal::ast::matrixExpr (varName * var, Expr * expr1, Expr * expr2)` `[inline]`

This constructor takes three parameters

Parameters

<i>var</i>	varName representing matrix name
<i>expr1</i>	Expr which represents row of Matrix
<i>expr2</i>	Expr which represents column of Matrix

Here is the call graph for this function:



7.26.3 Member Function Documentation

7.26.3.1 `std::string fcal::ast::matrixExpr::UnParse (void)` `[virtual]`

Returns the string : `var_->UnParse() + "[" + expr1_->UnParse() + ":" + expr2_->UnParse() + "]"`

Reimplemented from [fcal::ast::Node](#).

7.26.4 Member Data Documentation

7.26.4.1 `Expr* fcal::ast::matrixExpr::expr1_` `[private]`

[Expr](#) which represents row of Matrix.

7.26.4.2 `Expr* fcal::ast::matrixExpr::expr2_` `[private]`

[Expr](#) which represents column of Matrix.

7.26.4.3 `varName* fcal::ast::matrixExpr::var_` `[private]`

[varName](#) representing matrix name

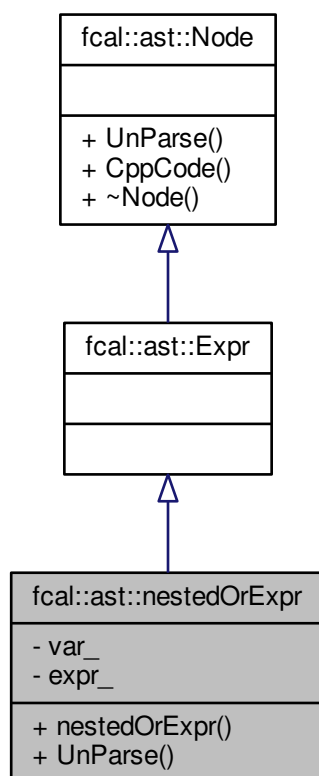
The documentation for this class was generated from the following files:

- [include/ast.h](#)
- [src/ast.cc](#)

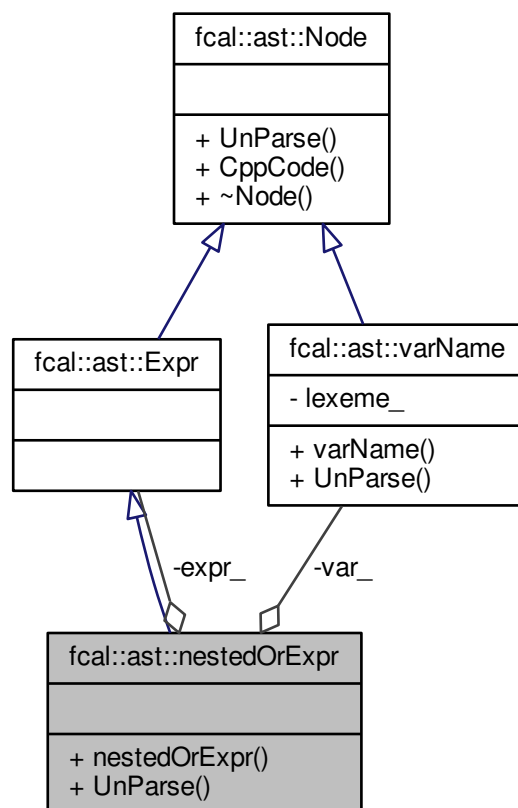
7.27 fcal::ast::nestedOrExpr Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::nestedOrExpr:



Collaboration diagram for fcal::ast::nestedOrExpr:



Public Member Functions

- [nestedOrExpr](#) ([varName](#) *var, [Expr](#) *expr)
- [std::string UnParse](#) (void)
Returns the string : `var_ -> UnParse\(\) + " (" + expr_ -> UnParse\(\) + ")"`.

Private Attributes

- [varName](#) * [var_](#)
[varName](#) before [Expr](#) enclosed in parenthesis
- [Expr](#) * [expr_](#)
[Expr](#) contained in parenthesis.

7.27.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
[Expr](#) ::= [varName](#) '(' [Expr](#) ')'

7.27.2 Constructor & Destructor Documentation

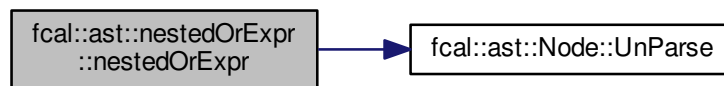
7.27.2.1 `fcgal::ast::nestedOrExpr::nestedOrExpr (varName * var, Expr * expr)` `[inline]`

This constructor takes 2 parameters

Parameters

<i>var</i>	<code>varName</code> before <code>Expr</code> enclosed in parenthesis
<i>expr</i>	<code>Expr</code> contained in parenthesis

Here is the call graph for this function:



7.27.3 Member Function Documentation

7.27.3.1 `std::string fcal::ast::nestedOrExpr::UnParse (void)` `[virtual]`

Returns the string : `var_->UnParse() + " (" + expr_->UnParse() + ")"`.

Reimplemented from `fcgal::ast::Node`.

7.27.4 Member Data Documentation

7.27.4.1 `Expr* fcal::ast::nestedOrExpr::expr_` `[private]`

`Expr` contained in parenthesis.

7.27.4.2 `varName* fcal::ast::nestedOrExpr::var_` `[private]`

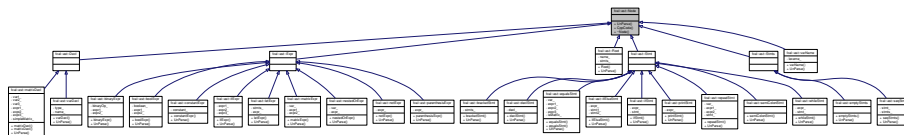
`varName` before `Expr` enclosed in parenthesis

The documentation for this class was generated from the following files:

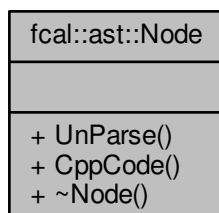
- `include/ast.h`
- `src/ast.cc`

```
#include <ast.h>
```

Inheritance diagram for `fcal::ast::Node`:



Collaboration diagram for `fcal::ast::Node`:



- virtual std::string UnParse (void)
- virtual std::string CppCode (void)
- virtual ~Node (void)

virtual destructor for polymorphism

This is the abstract class for ast heirarchy. `Root`, `Stmts`, `Stmt`, `Decl`, `Expr` and `varName` are derived from this class.

7.28.2.1 virtual fcal::ast::Node::~~Node (void) [inline], [virtual]

virtual destructor for polymorphism

7.28.3 Member Function Documentation

7.28.3.1 `virtual std::string fcal::ast::Node::CppCode (void) [inline],[virtual]`

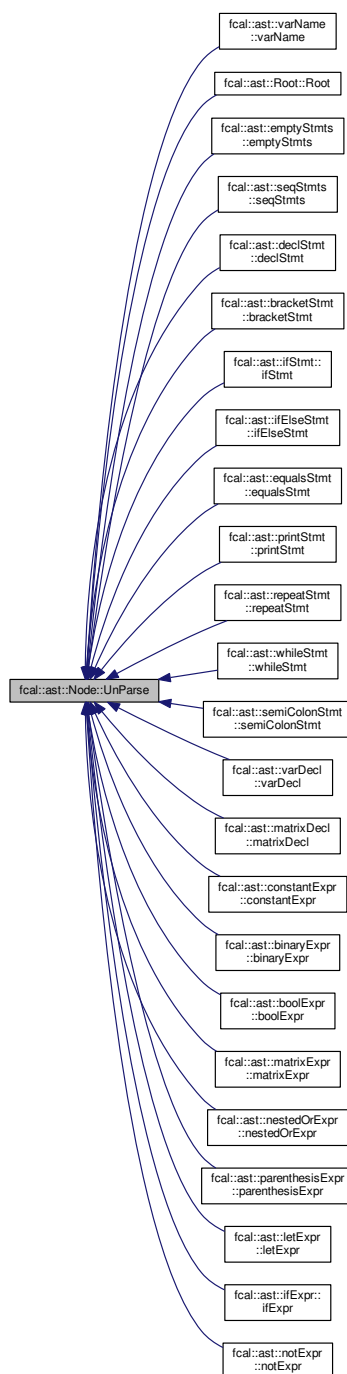
CppCode is a polymorphic function which is implemented by concrete classes in the ast.

7.28.3.2 `virtual std::string fcal::ast::Node::UnParse (void) [inline],[virtual]`

UnParse is a polymorphic function which is implemented by concrete classes in the ast.

Reimplemented in [fcal::ast::notExpr](#), [fcal::ast::ifExpr](#), [fcal::ast::letExpr](#), [fcal::ast::parenthesisExpr](#), [fcal::ast::nestedOrExpr](#), [fcal::ast::matrixExpr](#), [fcal::ast::boolExpr](#), [fcal::ast::binaryExpr](#), [fcal::ast::constantExpr](#), [fcal::ast::matrixDecl](#), [fcal::ast::varDecl](#), [fcal::ast::semiColonStmt](#), [fcal::ast::whileStmt](#), [fcal::ast::repeatStmt](#), [fcal::ast::printStmt](#), [fcal::ast::equalsStmt](#), [fcal::ast::ifElseStmt](#), [fcal::ast::ifStmt](#), [fcal::ast::bracketStmt](#), [fcal::ast::declStmt](#), [fcal::ast::seqStmts](#), [fcal::ast::emptyStmts](#), [fcal::ast::Root](#), and [fcal::ast::varName](#).

Here is the caller graph for this function:



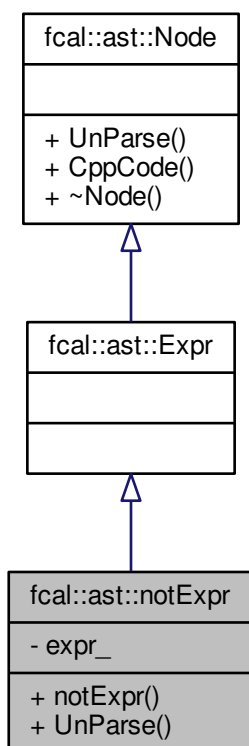
The documentation for this class was generated from the following file:

- [include/ast.h](#)

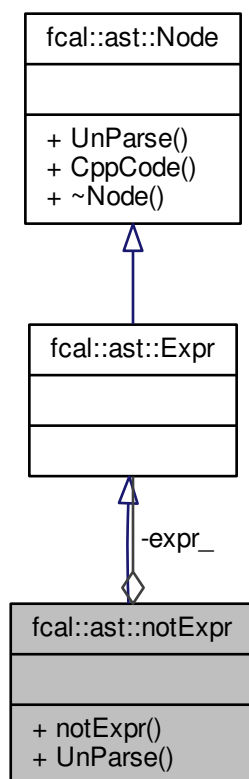
7.29 fcal::ast::notExpr Class Reference

```
#include <ast.h>
```

Inheritance diagram for `fcgal::ast::notExpr`:



Collaboration diagram for fcal::ast::notExpr:



Public Member Functions

- [notExpr](#) ([Expr](#) *expr)
- `std::string` [UnParse](#) (void)
Returns the string : `"!" + expr_->UnParse();`.

Private Attributes

- [Expr](#) * [expr_](#)
[Expr](#) whose boolean result will be negated.

7.29.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,

[Expr](#) ::= `!' Expr`

7.29.2 Constructor & Destructor Documentation

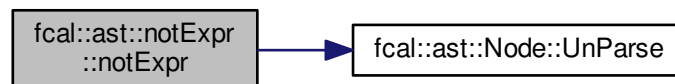
7.29.2.1 `fcf::ast::notExpr::notExpr (Expr * expr)` `[inline]`, `[explicit]`

This constructor takes one parameter

Parameters

<i>expr</i>	Expr whose boolean result will be negated
-------------	---

Here is the call graph for this function:



7.29.3 Member Function Documentation

7.29.3.1 `std::string fcal::ast::notExpr::UnParse (void)` `[virtual]`

Returns the string : `"!" + expr_->UnParse();`.

Reimplemented from [fcal::ast::Node](#).

7.29.4 Member Data Documentation

7.29.4.1 `Expr* fcal::ast::notExpr::expr_` `[private]`

[Expr](#) whose boolean result will be negated.

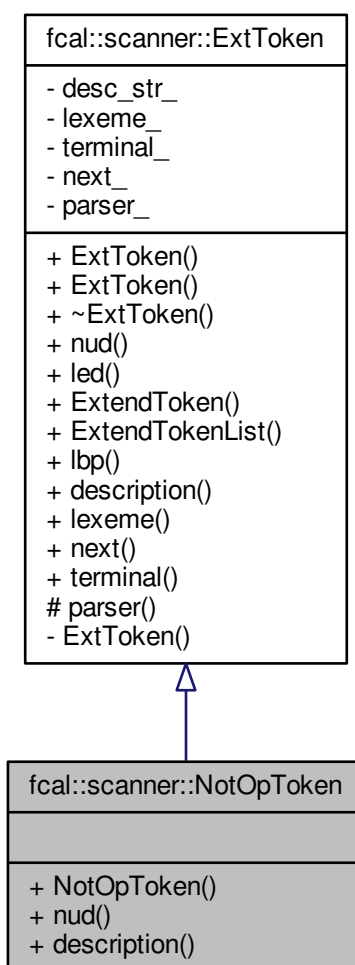
The documentation for this class was generated from the following files:

- `include/ast.h`
- `src/ast.cc`

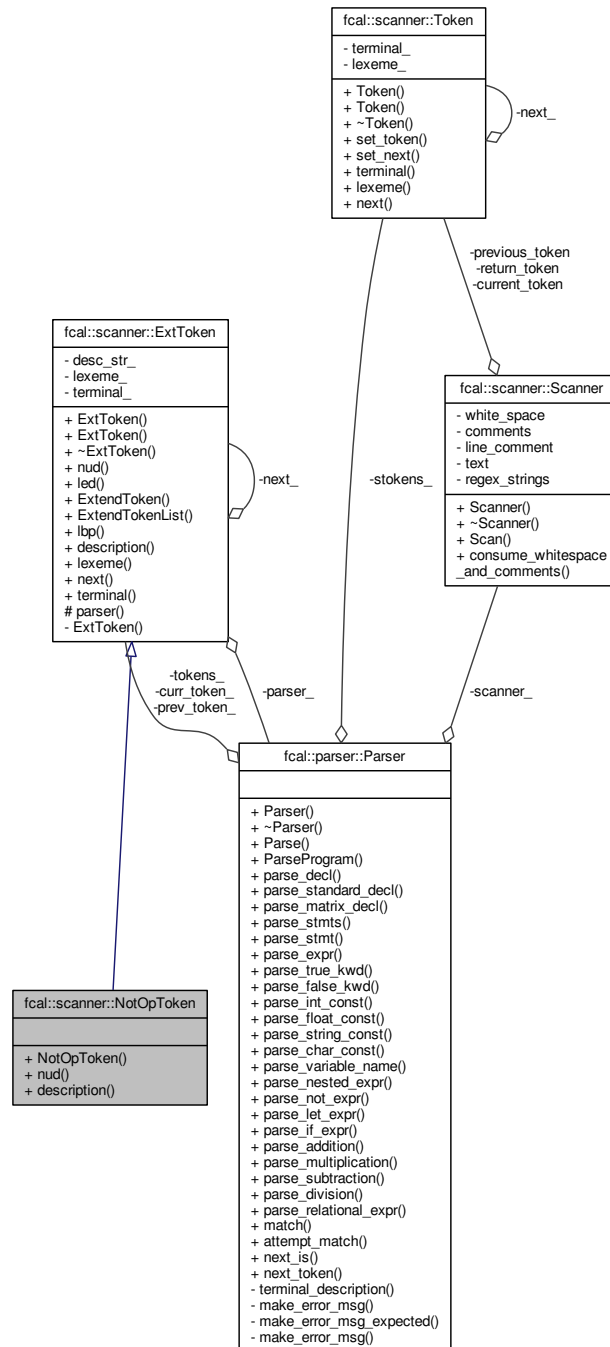
7.30 fcal::scanner::NotOpToken Class Reference

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::NotOpToken:



Collaboration diagram for fcal::scanner::NotOpToken:



Public Member Functions

- [NotOpToken](#) ([parser::Parser *p](#), [Token *t](#))
- [parser::ParseResult nud](#) ()
- [std::string description](#) ()

Additional Inherited Members

7.30.1 Detailed Description

For each terminal symbol that will play some unique role in the semantic analysis of the program, we need a unique subclass of [ExtToken](#).

7.30.2 Constructor & Destructor Documentation

7.30.2.1 `fcsl::scanner::NotOpToken::NotOpToken (parser::Parser * p, Token * t)` `[inline]`

7.30.3 Member Function Documentation

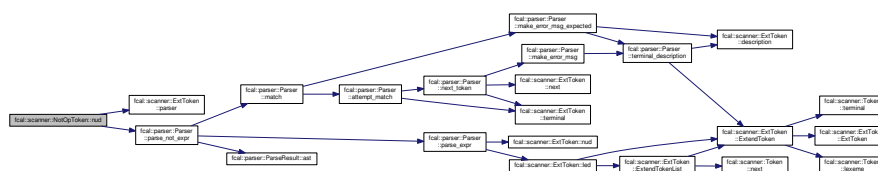
7.30.3.1 `std::string fcsl::scanner::NotOpToken::description ()` `[inline]`, `[virtual]`

Reimplemented from [fcsl::scanner::ExtToken](#).

7.30.3.2 `parser::ParseResult fcsl::scanner::NotOpToken::nud (void)` `[inline]`, `[virtual]`

Reimplemented from [fcsl::scanner::ExtToken](#).

Here is the call graph for this function:



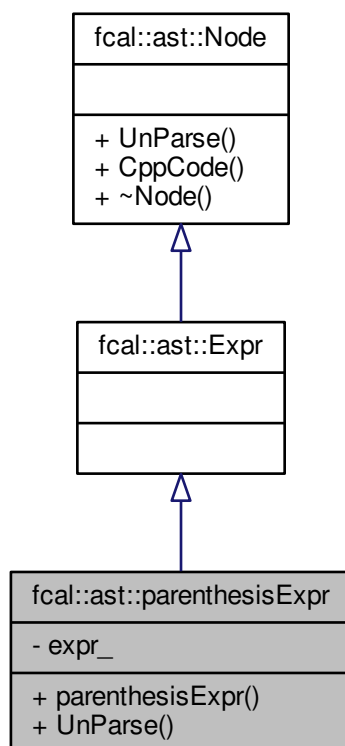
The documentation for this class was generated from the following file:

- [include/ext_token.h](#)

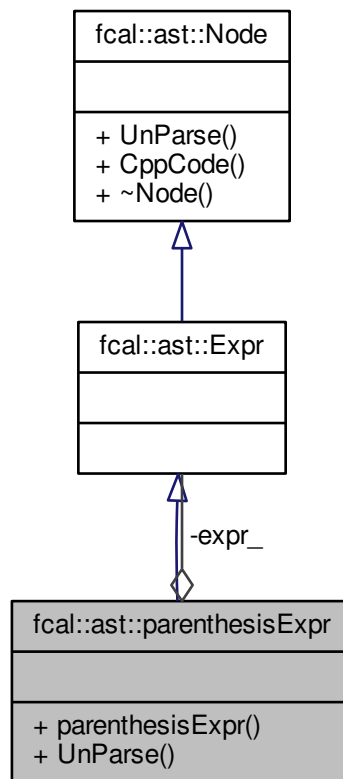
7.31 fcal::ast::parenthesisExpr Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::parenthesisExpr:



Collaboration diagram for `fcgal::ast::parenthesisExpr`:



Public Member Functions

- [parenthesisExpr](#) ([Expr](#) *expr)
- `std::string` [UnParse](#) (void)
Returns the string : "(" + expr_->[UnParse\(\)](#) + ")".

Private Attributes

- [Expr](#) * [expr_](#)
an [Expr](#) enclosed in parenthesis

7.31.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
[Expr](#) ::= '(' [Expr](#) ')'

7.31.2 Constructor & Destructor Documentation

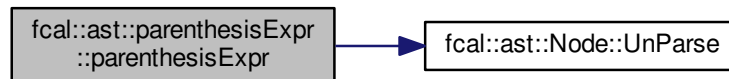
7.31.2.1 `fcgal::ast::parenthesisExpr::parenthesisExpr (Expr * expr) [inline], [explicit]`

This constructor takes one paramters

Parameters

<i>expr</i>	an Expr enclosed in parenthesis
-------------	---

Here is the call graph for this function:



7.31.3 Member Function Documentation

7.31.3.1 `std::string fcal::ast::parenthesisExpr::UnParse (void)` `[virtual]`

Returns the string : "(" + `expr_`->`UnParse()` + ")".

Reimplemented from [fcal::ast::Node](#).

7.31.4 Member Data Documentation

7.31.4.1 `Expr* fcal::ast::parenthesisExpr::expr_` `[private]`

an [Expr](#) enclosed in parenthesis

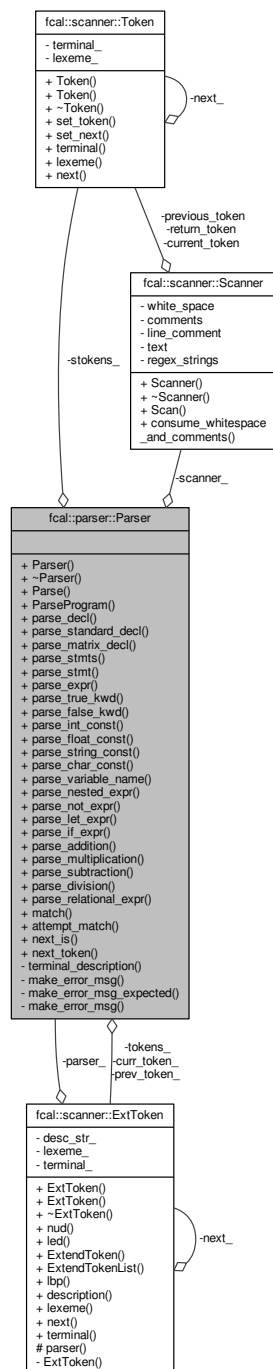
The documentation for this class was generated from the following files:

- include/[ast.h](#)
- src/[ast.cc](#)

7.32 fcal::parser::Parser Class Reference

```
#include <parser.h>
```

Collaboration diagram for `fcsl::parser::Parser`:



Public Member Functions

- [Parser](#) (void)
- [~Parser](#) (void)
- *Parser constructors/destructors.*
- [ParseResult Parse](#) (const char *text)
- [ParseResult ParseProgram](#) ()

Parser methods for the nonterminals:

- [ParseResult parse_decl \(\)](#)
Parser::standardDecl.
- [ParseResult parse_standard_decl \(\)](#)
Parser::matrixDecl.
- [ParseResult parse_matrix_decl \(\)](#)
Parser::ParseProgram()
- [ParseResult parse_stmts \(\)](#)
Parse general statements.
- [ParseResult parse_stmt \(\)](#)
Parse a general statement.
- [ParseResult parse_expr \(int rbp\)](#)
Parser::parse_stmt.
- [ParseResult parse_true_kwd \(\)](#)
methods for parsing productions for Expr
- [ParseResult parse_false_kwd \(\)](#)
Expr ::= falseKwd.
- [ParseResult parse_int_const \(\)](#)
Expr ::= intConst.
- [ParseResult parse_float_const \(\)](#)
Expr ::= floatConst.
- [ParseResult parse_string_const \(\)](#)
Expr ::= stringConst.
- [ParseResult parse_char_const \(\)](#)
- [ParseResult parse_variable_name \(\)](#)
Expr ::= variableName
- [ParseResult parse_nested_expr \(\)](#)
Expr ::= leftParen Expr rightParen.
- [ParseResult parse_not_expr \(\)](#)
Expr ::= '!' Expr.
- [ParseResult parse_let_expr \(\)](#)
Expr ::= 'let' Stmt 'in' Expr 'end'.
- [ParseResult parse_if_expr \(\)](#)
Expr ::= 'if' Expr 'then' Expr 'else' Expr.
- [ParseResult parse_addition \(ParseResult left\)](#)
Expr ::= Expr plusSign Expr.
- [ParseResult parse_multiplication \(ParseResult left\)](#)
Expr ::= Expr star Expr.
- [ParseResult parse_subtraction \(ParseResult left\)](#)
- [ParseResult parse_division \(ParseResult left\)](#)
Expr ::= Expr forwardSlash Expr.
- [ParseResult parse_relational_expr \(ParseResult left\)](#)
- void [match](#) (const [scanner::TokenType](#) &tt)
Helper function used by the parser.
- bool [attempt_match](#) (const [scanner::TokenType](#) &tt)
- bool [next_is](#) (const [scanner::TokenType](#) &tt)
- void [next_token](#) (void)

Private Member Functions

- `std::string terminal_description` (const `scanner::TokenType` &terminal)
- `std::string make_error_msg` (const `scanner::TokenType` &terminal)
- `std::string make_error_msg_expected` (const `scanner::TokenType` &terminal)
- `std::string make_error_msg` (const char *msg)

Private Attributes

- `scanner::ExtToken * tokens_`
- `scanner::ExtToken * curr_token_`
- `scanner::ExtToken * prev_token_`
- `scanner::Token * stokens_`
- `scanner::Scanner * scanner_`

7.32.1 Constructor & Destructor Documentation

7.32.1.1 `fcsl::parser::Parser::Parser (void)` `[inline]`

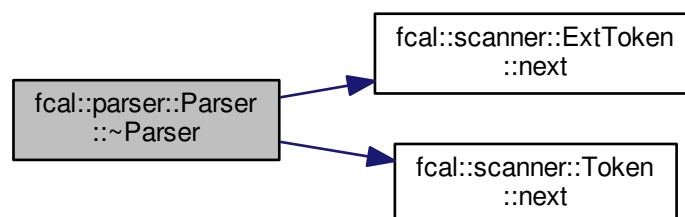
7.32.1.2 `fcsl::parser::Parser::~~Parser (void)`

`Parser` constructors/destructors.

`while()`

`while()`

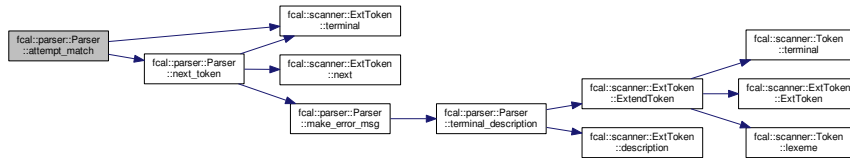
Here is the call graph for this function:



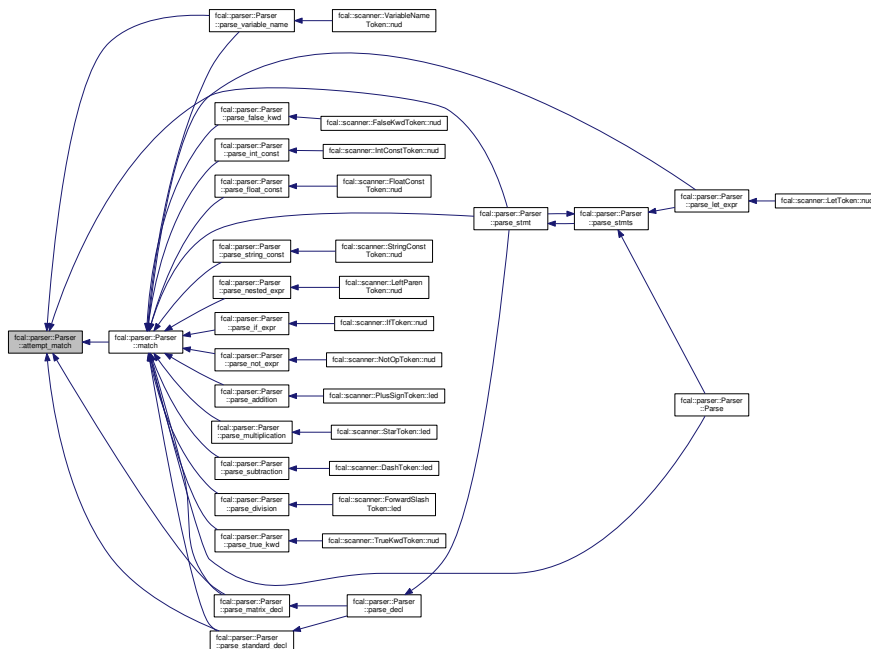
7.32.2 Member Function Documentation

7.32.2.1 bool fcal::parser::Parser::attempt_match (const scanner::TokenType & tt)

Here is the call graph for this function:

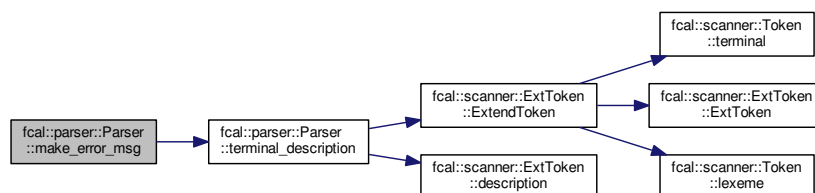


Here is the caller graph for this function:

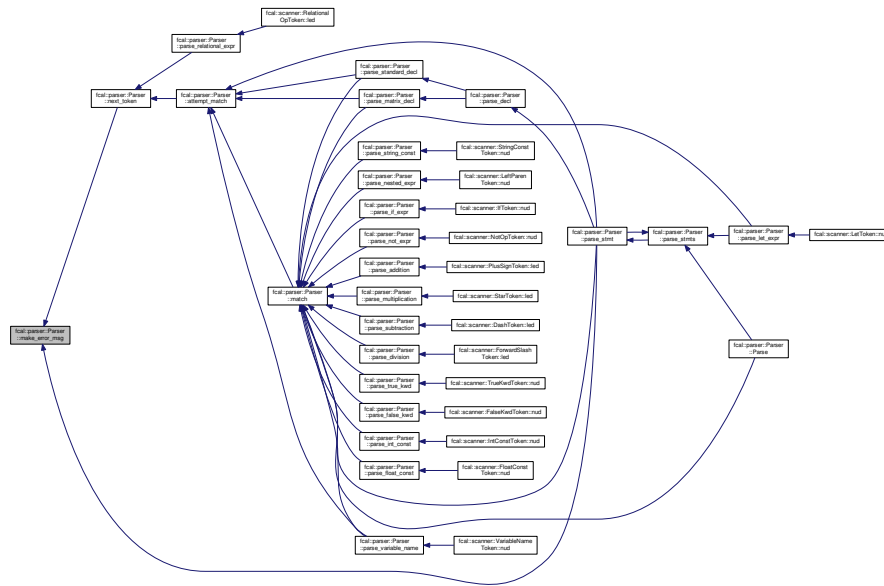


7.32.2.2 std::string fcal::parser::Parser::make_error_msg (const scanner::TokenType & terminal) [private]

Here is the call graph for this function:



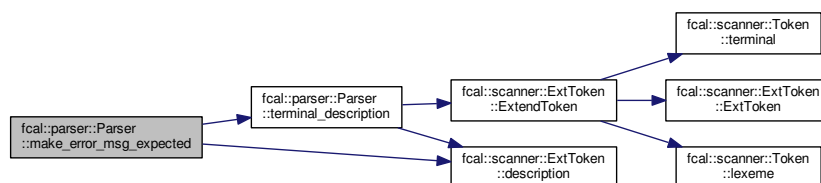
Here is the caller graph for this function:



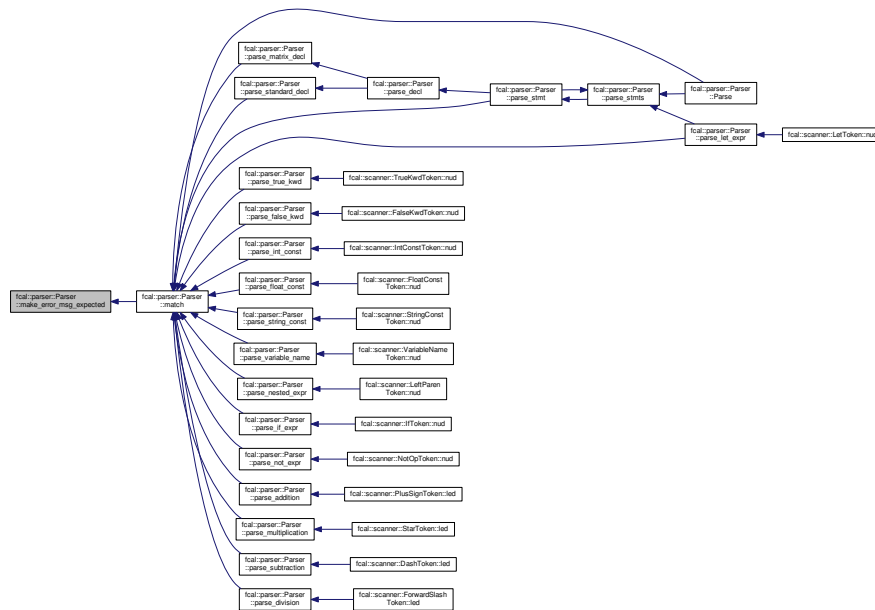
7.32.2.3 `std::string fcal::parser::Parser::make_error_msg (const char * msg)` [private]

7.32.2.4 `std::string fcal::parser::Parser::make_error_msg_expected (const scanner::TokenType & terminal)` [private]

Here is the call graph for this function:



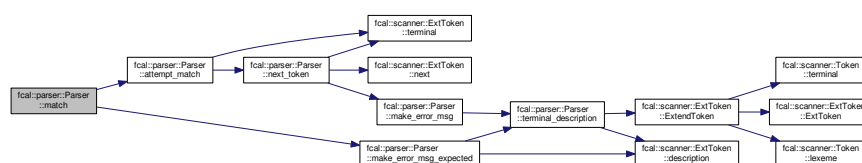
Here is the caller graph for this function:



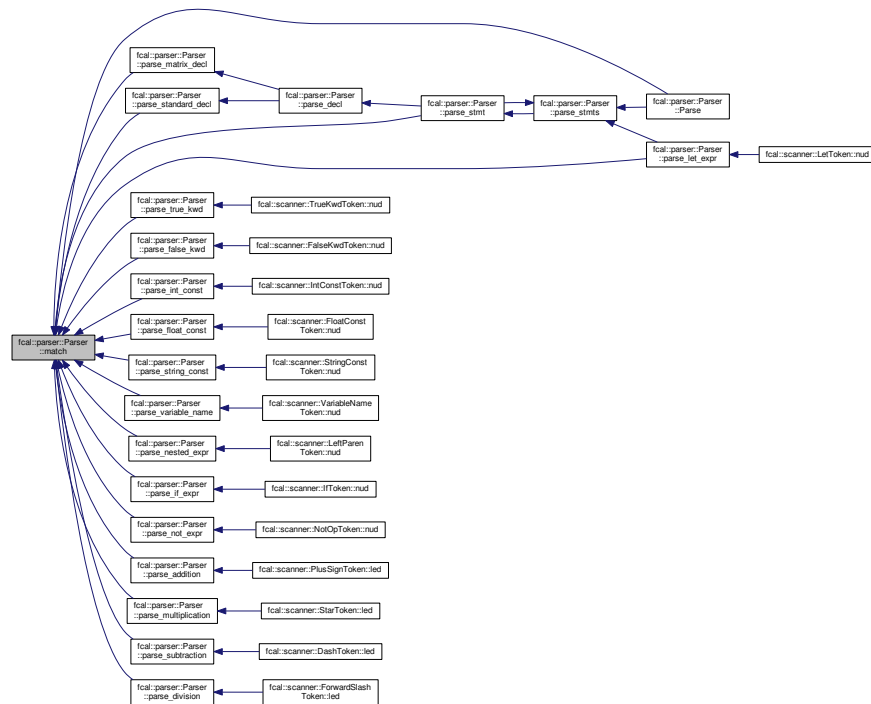
7.32.2.5 void fcal::parser::Parser::match (const scanner::TokenType & tt)

Helper function used by the parser.

Here is the call graph for this function:

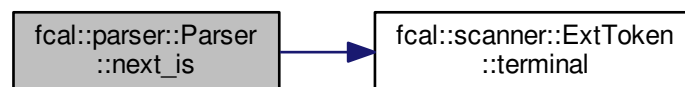


Here is the caller graph for this function:

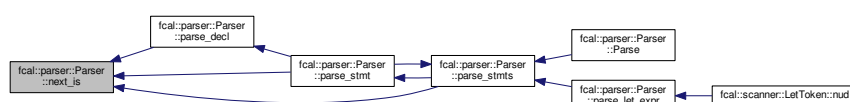


7.32.2.6 bool fcal::parser::Parser::next_is (const scanner::TokenType & tt)

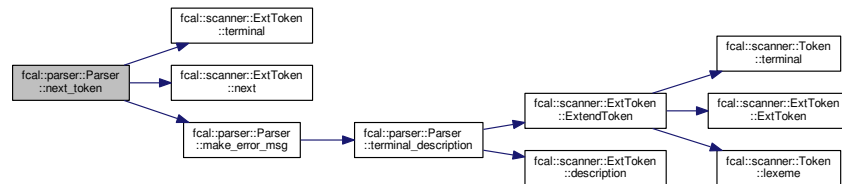
Here is the call graph for this function:



Here is the caller graph for this function:

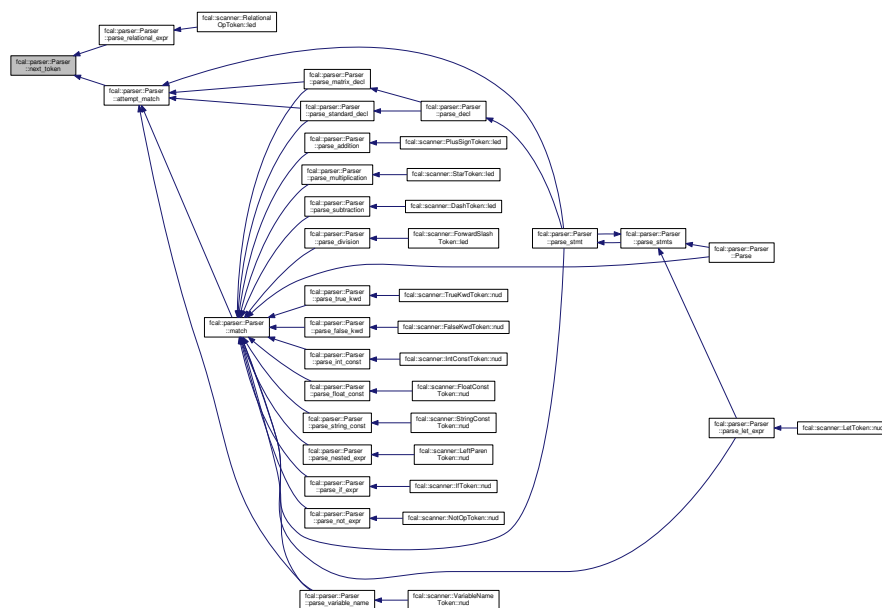


Here is the call graph for this function:



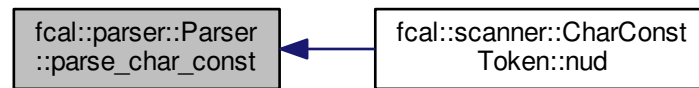
```

    fcat::parser::Parser
      : parse relational_expr
    fcat::scanner::Relational
      OpToken: fed
  
```



```
Parser::~~Parser()
```


Here is the caller graph for this function:

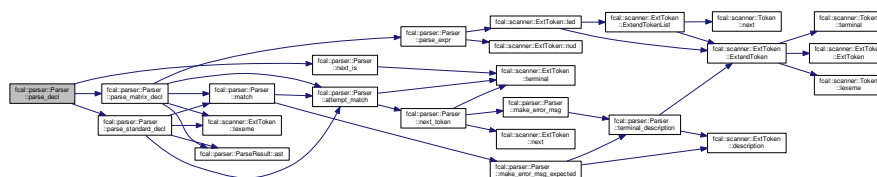


```
Parser::standardDecl.
```

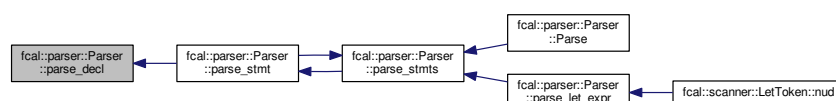
Determines whether declaration is a matrix or standard variable declaration. Decl :: matrix variableName ...

$$\text{Decl} ::= \text{Type variableName semiColon}$$

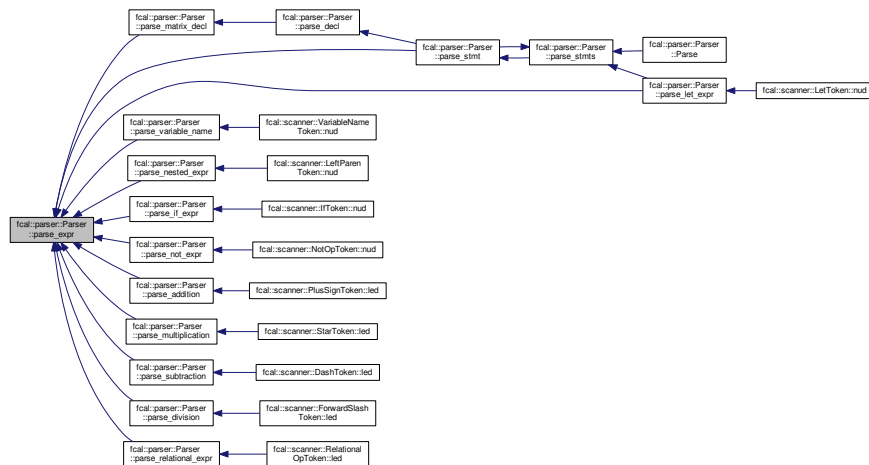
Here is the call graph for this function:



Here is the caller graph for this function:



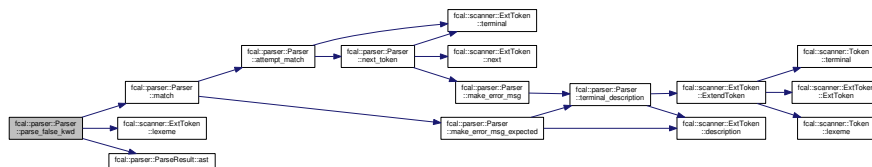
Here is the caller graph for this function:



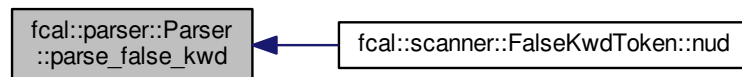
7.32.2.14 ParseResult fcal::parser::Parser::parse_false_kwd ()

Expr ::= falseKwd.

Here is the call graph for this function:



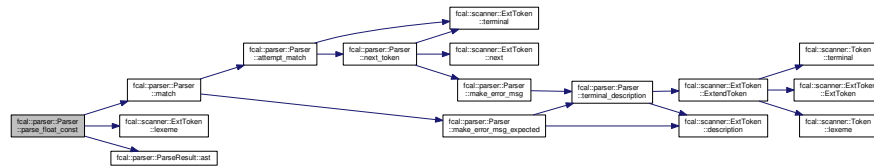
Here is the caller graph for this function:



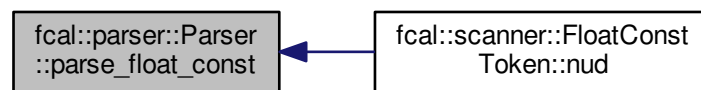
7.32.2.15 ParseResult fcal::parser::Parser::parse_float_const ()

Expr ::= floatConst.

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.16 ParseResult fcal::parser::Parser::parse_if_expr ()

Expr ::= 'if' Expr 'then' Expr 'else' Expr.

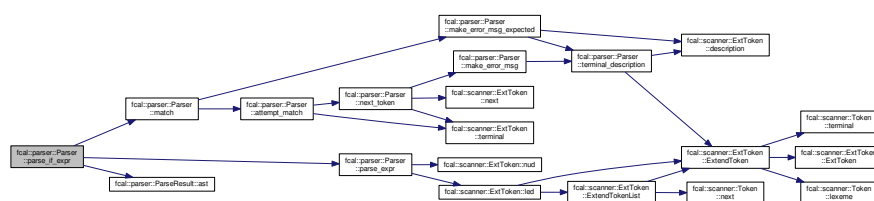
expr1

expr2

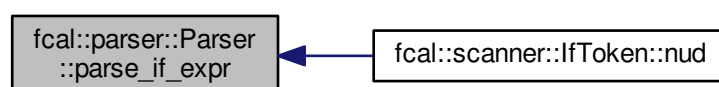
expr3

Create ifExpr node

Here is the call graph for this function:



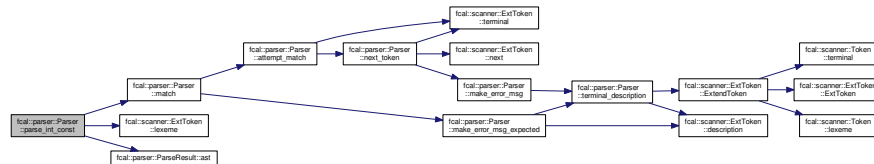
Here is the caller graph for this function:



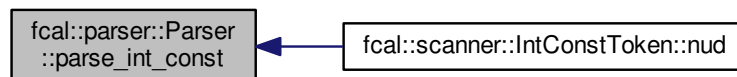
7.32.2.17 ParseResult fcal::parser::Parser::parse_int_const ()

Expr ::= intConst.

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.18 ParseResult fcal::parser::Parser::parse_let_expr ()

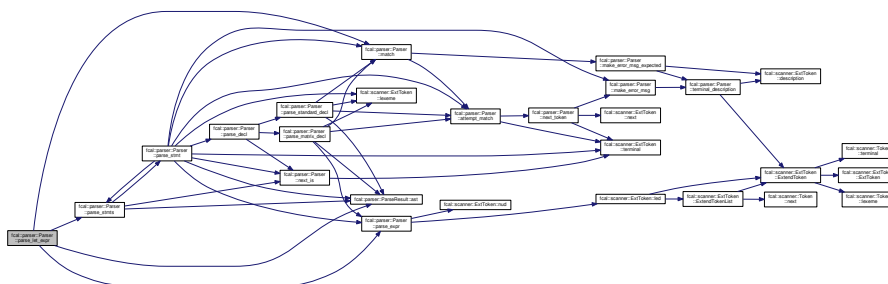
Expr ::= 'let' Stmts 'in' Expr 'end'.

stmts

expr

Create letExpr node

Here is the call graph for this function:



$$\text{Expr} ::= \text{Expr star Expr.}$$

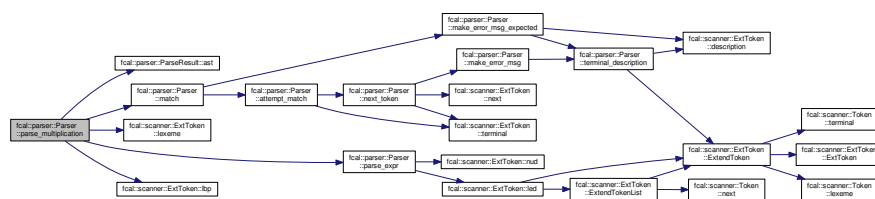
expr1

string *

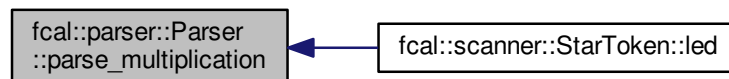
expr1

Create binaryExpr node

Here is the call graph for this function:

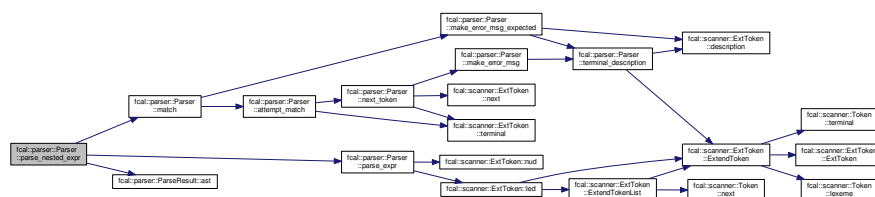


Here is the caller graph for this function:

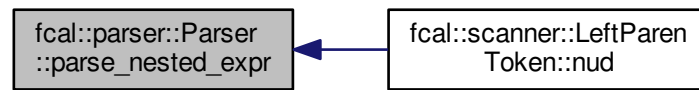


Expr ::= leftParen Expr rightParen.

Here is the call graph for this function:



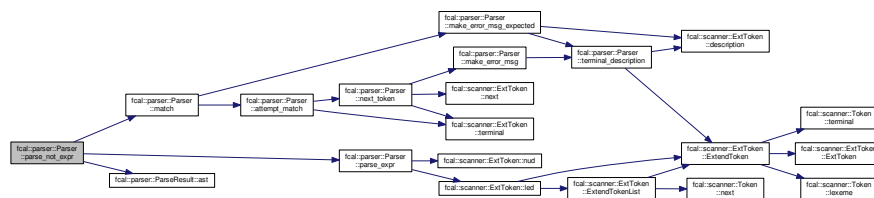
Here is the caller graph for this function:



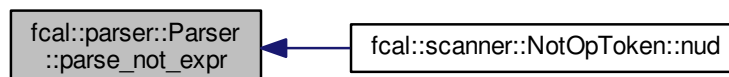
7.32.2.22 ParseResult fcal::parser::Parser::parse_not_expr ()

Expr ::= '!' Expr.

Here is the call graph for this function:



Here is the caller graph for this function:



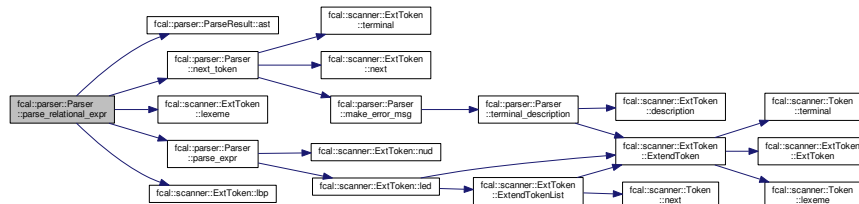
7.32.2.23 ParseResult fcal::parser::Parser::parse_relational_expr (ParseResult prLeft)

Expr ::= Expr equalEquals Expr Expr ::= Expr lessThanEquals Expr Expr ::= Expr greaterThanEquals Expr Expr ::= Expr notEquals Expr Expr ::= Expr leftAngle Expr Expr ::= Expr rightAngle Expr Notice that for relational operators we use just one parse function. This shows another possible means for implementing expressions, as opposed to the method used for arithmetic expressions in which each operation has its own parse method. It will depend on what we do in iteration 3 in building an abstract syntax tree to decide which method is better. parser has already matched left expression

expr1

expr2

Here is the call graph for this function:

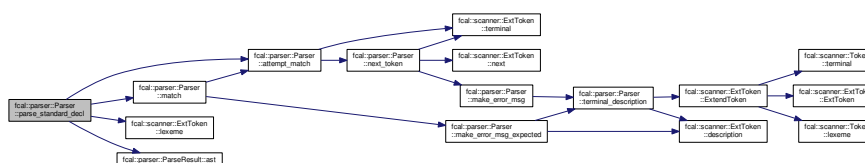


```
graph LR
    A["fcal::parser::Parser  
::parse_relational_expr"]
    B["fcal::scanner::Relational  
OpToken::led"]
    B --> A
```

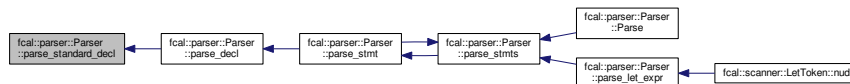
Parser::matrixDecl.

varName

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.25 ParseResult fcal::parser::Parser::parse_stmt ()

Parse a general statement.

If we have a declaration statement: Stmt ::= Decl

decl

Create new declStmt node

Stmt ::= '{' Stmt '}'

stmts

Create bracketStmt node

Stmt ::= 'if' '(' Expr ')' Stmt Stmt ::= 'if' '(' Expr ')' Stmt 'else' Stmt

expr

stmt

Create ifStmt node

It is if-else statment

elseStmt

Create ifElseStmt node

We have an assignment statement Stmt ::= varName '=' Expr ';' | varName '[' Expr ':' Expr ']' '=' Expr ';' |

isMatrix is used to show matrix or standard assginment

varName

We have a matrix assignment

expr2

expr3

expr1

Create equalsStmt node for matrix or standard declaration

We have a print statement Stmt ::= 'print' '(' Expr ')' ';' |

Create printStmt node

varName

expr1

expr2

stmt

Create repeatStmt node

We have a while statement $\text{Stmt} ::= \text{'while' ' (' Expr ') ' Stmt}$

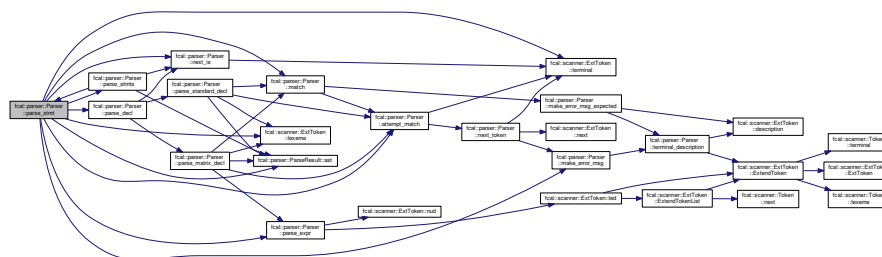
expr

stmt

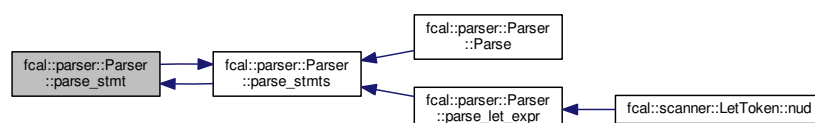
Create whileStmt node

Stmt ::= ';

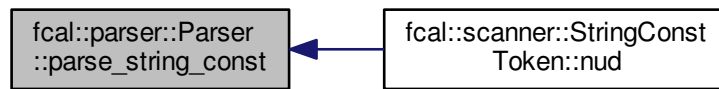
Here is the call graph for this function:



Here is the caller graph for this function:



Here is the caller graph for this function:



7.32.2.28 ParseResult fcal::parser::Parser::parse_subtraction (ParseResult left)

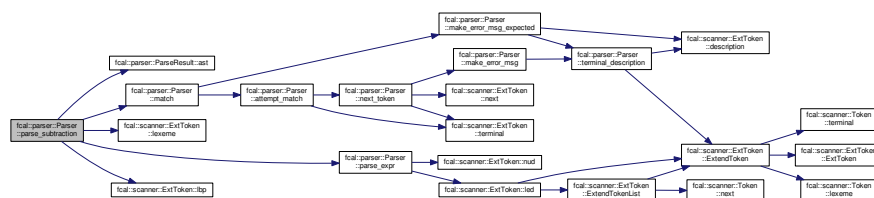
expr1

string -

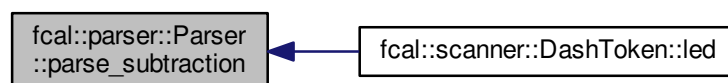
expr2

Create binaryExpr node

Here is the call graph for this function:



Here is the caller graph for this function:



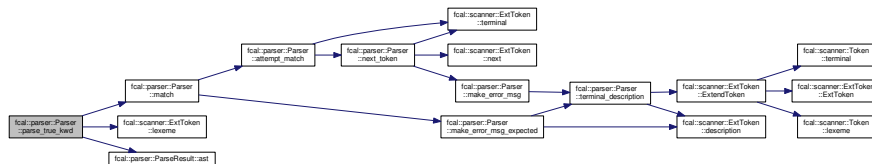
7.32.2.29 ParseResult fcal::parser::Parser::parse_true_kwd ()

methods for parsing productions for Expr

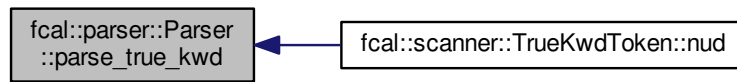
[Parser::parse_expr.](#)

Parse each specific expression and create the proper node in the AST. Expr ::= trueKwd

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.30 ParseResult fcal::parser::Parser::parse_variable_name ()

Expr ::= variableName

Expr ::= matrix [Expr:Expr]

expr1

expr2

Create matrixExpr node

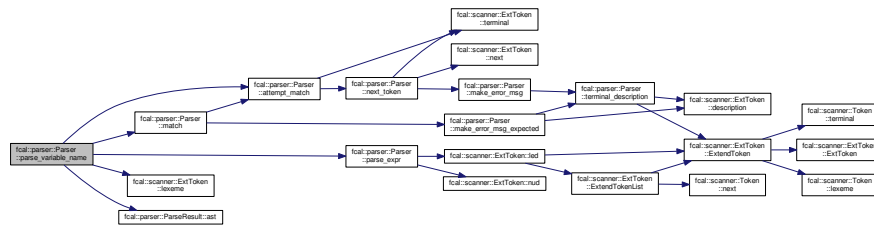
Expr ::= variableName '(' Expr ')'

expr

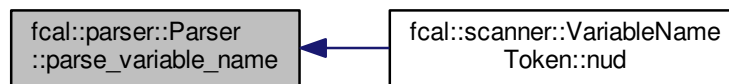
Create nestedExpr node

We have right hand side variable expression

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.31 ParseResult fcal::parser::Parser::ParseProgram ()

[Parser](#) methods for the nonterminals:

Parser::parse()

parse methods for non-terminal symbols

Program – This is the root node where the entire program will be placed once everything has been parsed and placed in the proper node of the abstract syntax tree(AST). This function's comments describe the general structure of the overall program, and specifically how each function parses the program according to the grammar of that function and how it then creates the AST from that grammar. Each function will follow this same general structure. **root** – the following line shows the grammar that this function is intended to parse to create the specific node needed for the AST.

```
Program ::= varName '(' ')' '{' Stmtns '}'
```

String name holds varName in grammar after finding kVariableName, setting varName class with name. This same form will continue throughout program for any function requiring a variable name in the grammar.

pr_stmts will hold all statements after recursively calling appropriate grammar. [ParseResult](#) pr_stmts, pr_expr, pr_<_decl, or similar variations will follow the same form throughout the program calling each function recursively when a declaration, statement, or expression is needed according to the specific grammar that the current function requires. [ParseResult](#) pr_stmts = [parse_stmts](#)();

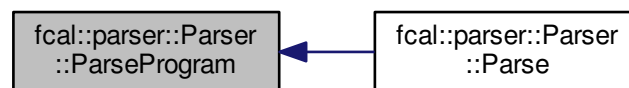
[ast::Stmnts](#) stmts will hold the Node *ast value from pr_stmts. This same form will continue throughout the program to set the current Expr, Stmt, or Decl class *ast value. These *ast values are what will be used to set the values for the subclasses that will be used to create the nodes for the AST.

The following checks that `pr_stmts ast` value isn't null, and that `stmts` was properly assigned the value from [ParseResult](#)'s Node class `*ast`.

Set the Root node with the entire program, `varName {statements}`. This same form will be used throughout the program to set the respective variables, statements, expressions, and declarations to the proper node of the AST according to the grammar for the current function.

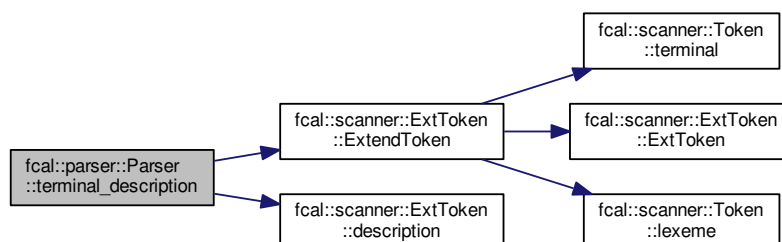
Set the ParseResults Node `*ast` with the final program once everything is parsed. This same form will continue throughout the program, `pr.ast(node)` sets the primary [ParseResult](#) object with the node that holds the values acquired during the current function according to its grammar.

Here is the caller graph for this function:

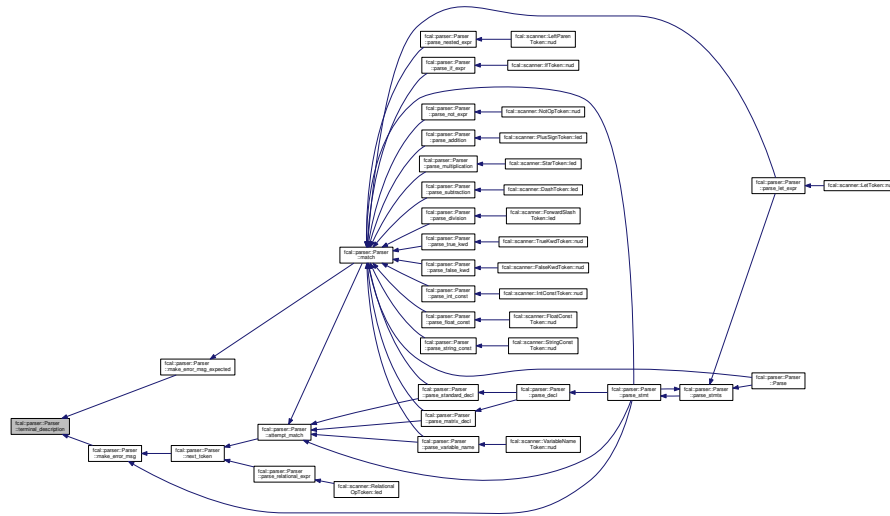


7.32.2.32 `std::string fcal::parser::Parser::terminal_description (const scanner::TokenType & terminal)` `[private]`

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.3 Member Data Documentation

7.32.3.1 `scanner::ExtToken* fcal::parser::Parser::curr_token_` [private]

7.32.3.2 `scanner::ExtToken* fcal::parser::Parser::prev_token_` [private]

7.32.3.3 `scanner::Scanner* fcal::parser::Parser::scanner_` [private]

7.32.3.4 `scanner::Token* fcal::parser::Parser::tokens_` [private]

7.32.3.5 `scanner::ExtToken* fcal::parser::Parser::tokens_` [private]

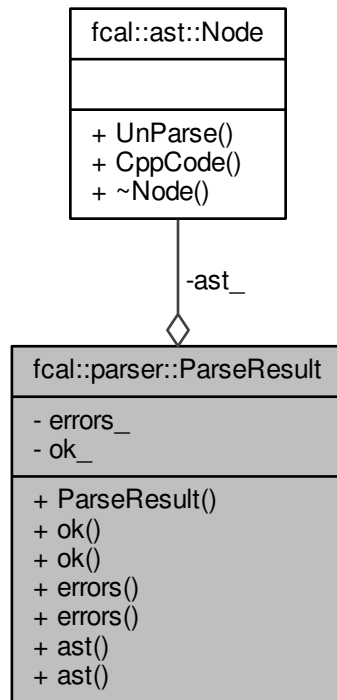
The documentation for this class was generated from the following files:

- [include/parser.h](#)
- [src/parser.cc](#)

7.33 fcal::parser::ParseResult Class Reference

```
#include <parse_result.h>
```

Collaboration diagram for `fcgal::parser::ParseResult`:



Public Member Functions

- [ParseResult](#) (void)
- bool [ok](#) (void) const
- void [ok](#) (bool result_in)
- std::string [errors](#) (void) const
- void [errors](#) (const std::string str_in)
- [ast::Node](#) * [ast](#) (void)
- void [ast](#) ([ast::Node](#) *Node_ptr)

Private Attributes

- std::string [errors_](#)
- [ast::Node](#) * [ast_](#)
- bool [ok_](#)

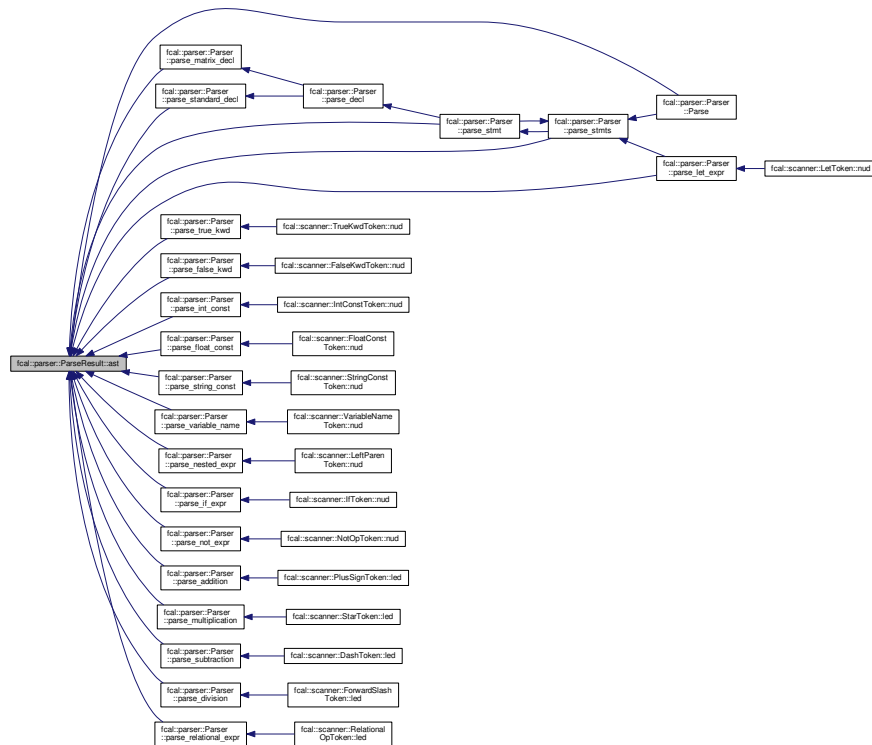
7.33.1 Constructor & Destructor Documentation

7.33.1.1 `fcal::parser::ParseResult::ParseResult (void)` `[inline]`

7.33.2 Member Function Documentation

7.33.2.1 `ast::Node* fcal::parser::ParseResult::ast (void)` `[inline]`

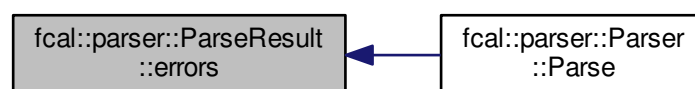
Here is the caller graph for this function:



7.33.2.2 `void fcal::parser::ParseResult::ast (ast::Node * Node_ptr)` `[inline]`

7.33.2.3 `std::string fcal::parser::ParseResult::errors (void) const` `[inline]`

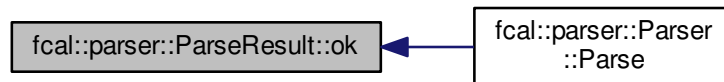
Here is the caller graph for this function:



7.33.2.4 `void fcal::parser::ParseResult::errors (const std::string str_in)` `[inline]`

7.33.2.5 `bool fcal::parser::ParseResult::ok (void) const` `[inline]`

Here is the caller graph for this function:



7.33.2.6 `void fcal::parser::ParseResult::ok (bool result_in)` `[inline]`

7.33.3 Member Data Documentation

7.33.3.1 `ast::Node* fcal::parser::ParseResult::ast_` `[private]`

7.33.3.2 `std::string fcal::parser::ParseResult::errors_` `[private]`

7.33.3.3 `bool fcal::parser::ParseResult::ok_` `[private]`

The documentation for this class was generated from the following file:

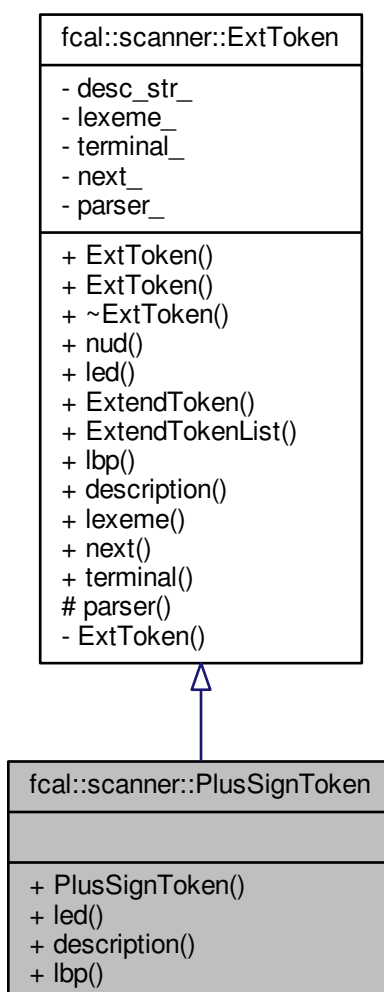
- [include/parse_result.h](#)

7.34 fcal::scanner::PlusSignToken Class Reference

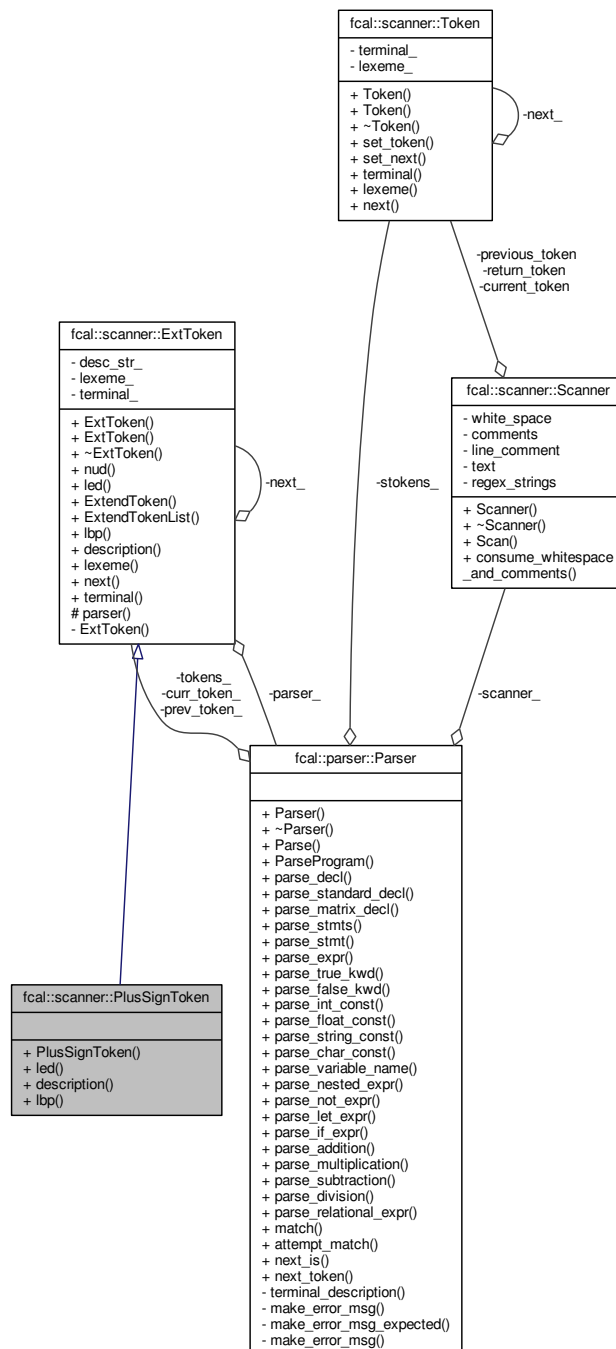
Plus Sign.

```
#include <ext_token.h>
```


Inheritance diagram for fcal::scanner::PlusSignToken:



Collaboration diagram for `fcsl::scanner::PlusSignToken`:



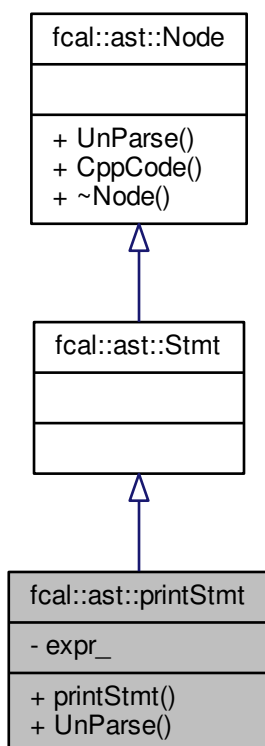
Public Member Functions

- [PlusSignToken](#) ([parser::Parser](#) *p, [Token](#) *t)
- [parser::ParseResult](#) led ([parser::ParseResult](#) left)
- [std::string](#) description ()
- [int](#) lbp ()

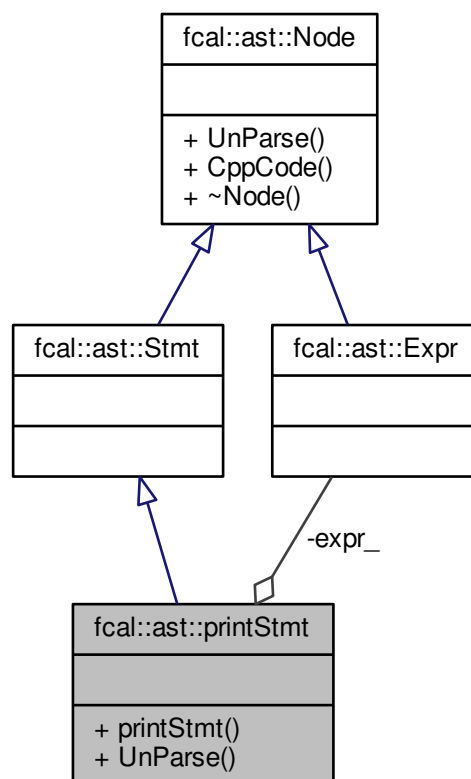
7.35 fcal::ast::printStmt Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::printStmt:



Collaboration diagram for fcal::ast::printStmt:



Public Member Functions

- [printStmt](#) ([Expr](#) *expr)
- std::string [UnParse](#) (void)
Returns the string: "print(" + expr_->[UnParse](#)() + ");\n".

Private Attributes

- [Expr](#) * expr_
An [Expr](#) which will be printed.

7.35.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production, [Stmt](#) ::= 'print' '(' [Expr](#) ')';

7.35.2 Constructor & Destructor Documentation

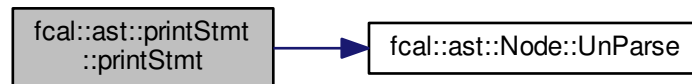
7.35.2.1 fcal::ast::printStmt::printStmt ([Expr](#) * expr) [inline], [explicit]

This constructor takes one arguement

Parameters

<i>expr</i>	an expression which should be printed
-------------	---------------------------------------

Here is the call graph for this function:



7.35.3 Member Function Documentation

7.35.3.1 `std::string fcal::ast::printStmt::UnParse (void)` [virtual]

Returns the string: "print(" + `expr_`->`UnParse()` + ");\n".

Reimplemented from [fcal::ast::Node](#).

7.35.4 Member Data Documentation

7.35.4.1 `Expr* fcal::ast::printStmt::expr_` [private]

An [Expr](#) which will be printed.

The documentation for this class was generated from the following files:

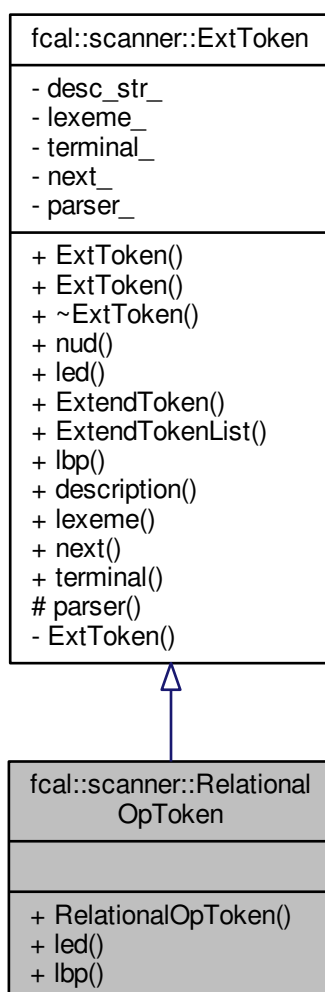
- [include/ast.h](#)
- [src/ast.cc](#)

7.36 `fcal::scanner::RelationalOpToken` Class Reference

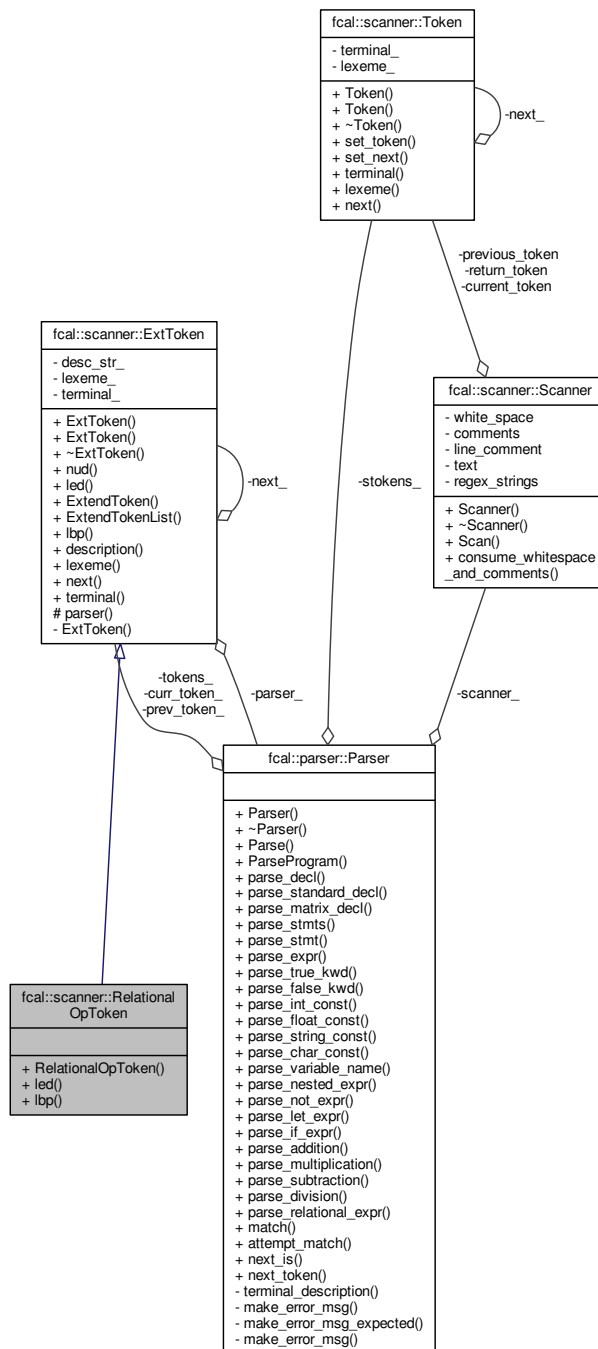
Relational Op.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::RelationalOpToken:



Collaboration diagram for `fcf::scanner::RelationalOpToken`:



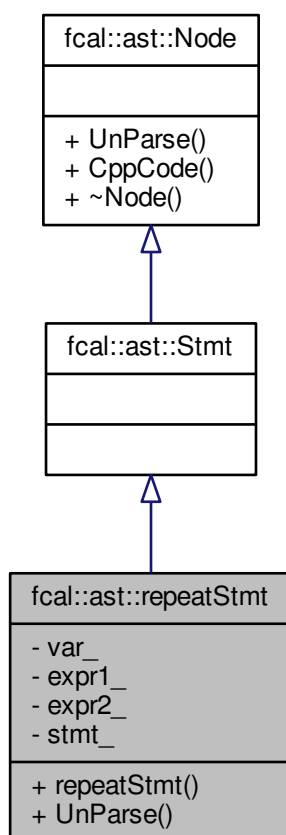
Public Member Functions

- [RelationalOpToken](#) (`parser::Parser *p`, `Token *t`, `std::string d`)
- [parser::ParseResult led](#) (`parser::ParseResult left`)
- [int lbp](#) ()

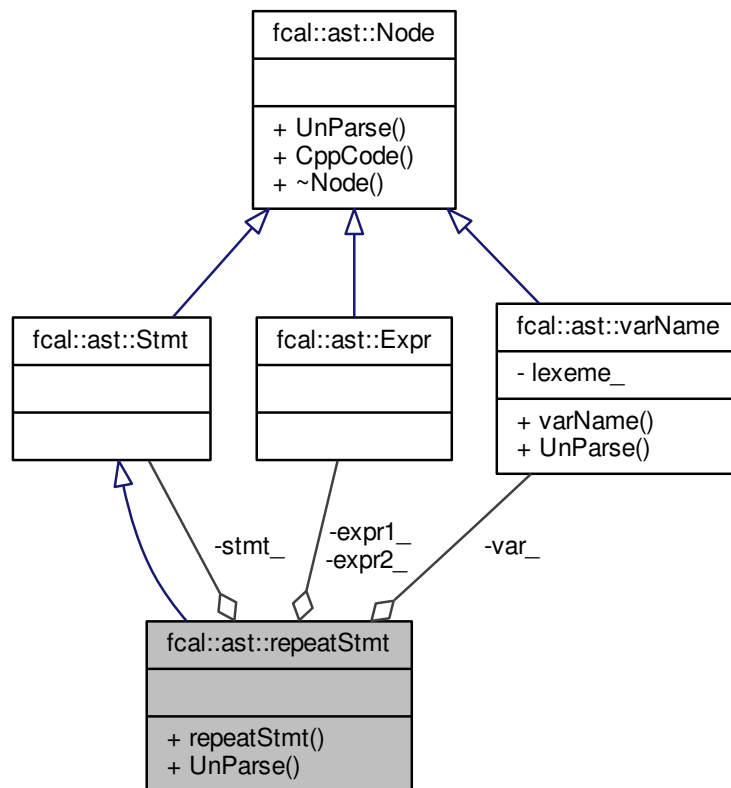
7.37 fcal::ast::repeatStmt Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::repeatStmt:



Collaboration diagram for fcal::ast::repeatStmt:



Public Member Functions

- `repeatStmt (varName *var, Expr *expr1, Expr *expr2, Stmt *stmt)`
- `std::string UnParse (void)`

Private Attributes

- `varName * var_`
a variable representing the index
- `Expr * expr1_`
an Expr which is the lower bound of repeat loop
- `Expr * expr2_`
an Expr which is the upper bound of repeat loop
- `Stmt * stmt_`

7.37.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
`Stmt ::= 'repeat' '(' varName '=' Expr 'to' Expr ')' Stmt`

7.37.2 Constructor & Destructor Documentation

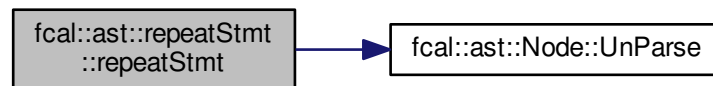
7.37.2.1 `fcal::ast::repeatStmt::repeatStmt (varName * var, Expr * expr1, Expr * expr2, Stmt * stmt)` `[inline]`

This constructor takes four parameters

Parameters

<i>var</i>	a variable representing the index
<i>expr1</i>	an Expr which is the lower bound of repeat loop
<i>expr2</i>	an Expr which is the upper bound of repeat loop
<i>stmt</i>	a Stmt which is executed while the index is within the limits specified by the <i>expr1</i> and <i>expr2</i>

Here is the call graph for this function:



7.37.3 Member Function Documentation

7.37.3.1 `std::string fcal::ast::repeatStmt::UnParse (void)` `[virtual]`

Returns the string : "repeat (" + `var_`->`UnParse()` + " = "

- `expr1_`->`UnParse()` + " to " + `expr2_`->`UnParse()` + ")\n" + `stmt_`->`UnParse()`

Reimplemented from [fcal::ast::Node](#).

7.37.4 Member Data Documentation

7.37.4.1 `Expr* fcal::ast::repeatStmt::expr1_` `[private]`

an [Expr](#) which is the lower bound of repeat loop

7.37.4.2 `Expr* fcal::ast::repeatStmt::expr2_` `[private]`

an [Expr](#) which is the upper bound of repeat loop

7.37.4.3 Stmt* fcal::ast::repeatStmt::stmt_ [private]

a [Stmt](#) which is executed while the index is within the limits specified by the expr1 and expr2

7.37.4.4 varName* fcal::ast::repeatStmt::var_ [private]

a variable representing the index

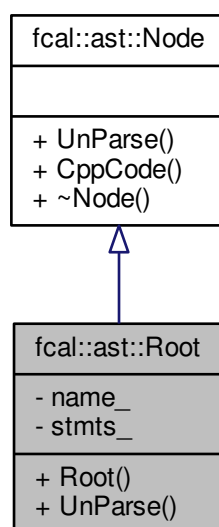
The documentation for this class was generated from the following files:

- include/[ast.h](#)
- src/[ast.cc](#)

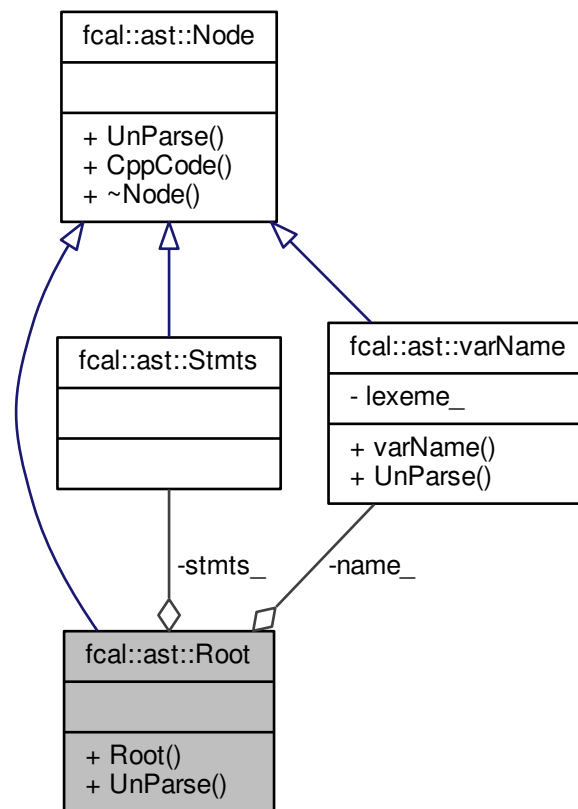
7.38 fcal::ast::Root Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::Root:



Collaboration diagram for `fcal::ast::Root`:



Public Member Functions

- `Root` (`varName` *name, `Stmts` *stmts)
- `std::string UnParse` (void)

Private Attributes

- `varName` * name_
name_ holds the address containing details of program name
- `Stmts` * stmts_

7.38.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production, `Program ::= varName '(' ')' '{' Stmts '}'` This class is the root of the ast tree created by the parser.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 fcal::ast::Root::Root (varName * *name*, Stmts * *stmts*) [inline]

Constructor for the [Root](#) class takes parameters

Parameters

<i>name</i>	a varName* holding a reference to the varName representing the program
<i>stmts</i>	a Stmts* holding a reference to the Stmts in the program

Here is the call graph for this function:



7.38.3 Member Function Documentation

7.38.3.1 `std::string fcal::ast::Root::UnParse (void)` `[virtual]`

Returns the string : `name_->UnParse() + " () " + "{\n" + stmts_->UnParse() + "\n}\n"`

Reimplemented from [fcal::ast::Node](#).

7.38.4 Member Data Documentation

7.38.4.1 `varName* fcal::ast::Root::name_` `[private]`

`name_` holds the address containing details of program name

7.38.4.2 `Stmts* fcal::ast::Root::stmts_` `[private]`

`stmts_` holds the address containing details of [Stmts](#) in the program

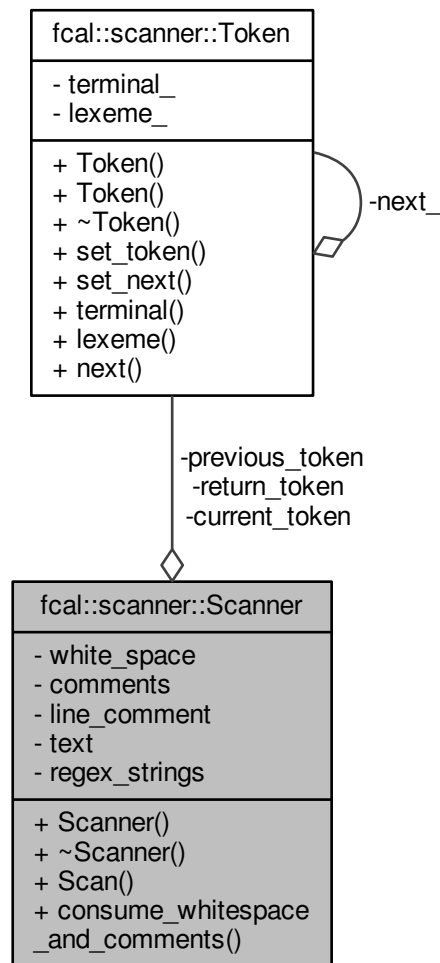
The documentation for this class was generated from the following files:

- [include/ast.h](#)
- [src/ast.cc](#)

7.39 fcal::scanner::Scanner Class Reference

```
#include <scanner_class.h>
```

Collaboration diagram for fcal::scanner::Scanner:



Public Member Functions

- [Scanner](#) ()
Default constructor for [Scanner](#).
- [~Scanner](#) ()
- [Token](#) * [Scan](#) (const char *)
- int [consume_whitespace_and_comments](#) (regex_t *, regex_t *, regex_t *, const char *)
end scan

Private Attributes

- `Token * return_token`
Token representing front of the list.*
- `Token * previous_token`
Token representing previous `Token` in the list.*
- `Token * current_token`
Token representing current `Token` in the list.*
- `regex_t * white_space`
A regex which matches white space.
- `regex_t * comments`
A regex which matches comments.
- `regex_t * line_comment`
A regex which matches line comments.
- `const char * text`
String representing contents of the File.
- `regex_t * regex_strings [1+static_cast< int >(kNotOp)]`
A regex array created for matching lexeme of Tokens.

7.39.1 Detailed Description

This is the `Scanner` class which implements the scanner of the FCAL program. `Scanner` has the method `Scan` which returns the front of the List of Tokens generated from scanning a file. It implements the first stage of the parsing process.

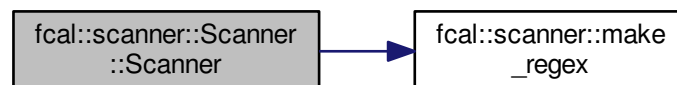
7.39.2 Constructor & Destructor Documentation

7.39.2.1 `fcal::scanner::Scanner::Scanner ()`

Default constructor for `Scanner`.

Initialize `token_strings`, `white_space`, `comments`, `line_comment` and `regex_strings`

Here is the call graph for this function:



7.39.2.2 `fcal::scanner::Scanner::~~Scanner ()`

7.39.3 Member Function Documentation

7.39.3.1 `int fcal::scanner::Scanner::consume_whitespace_and_comments (regex_t * white_space, regex_t * block_comment, regex_t * line_comment, const char * text)`

end scan

Consumes white spaces and comments

Parameters

<i>white_space</i>	regex for matching white space
<i>block_comment</i>	regex for matching block comment
<i>line_comment</i>	regex for matching line comment
<i>text</i>	A string representing File input

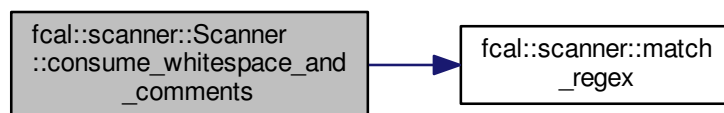
exit loop if not reset by a match

Try to match white space

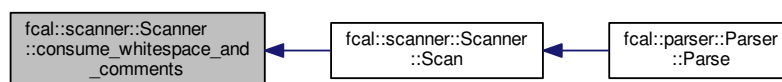
Try to match block comments

Try to match line comment comments

Here is the call graph for this function:



Here is the caller graph for this function:



7.39.3.2 Token * fcal::scanner::Scanner::Scan (const char * s)

Scans the input string and return's the front of the list of [Token](#)'s.

Parameters

<i>s</i>	input string representing the contents of a file to be scanned
----------	--

[Token](#) pointers to create list of tokens

consume `white_space` and comments

Check the text for qualified tokens or lexical errors

current [Token](#) is where we store token information

Check if next text string matches a token, if so add terminal and lexeme null pointer. If a token isn't found add lexeme and lexical error.

If no token match, set lexical error

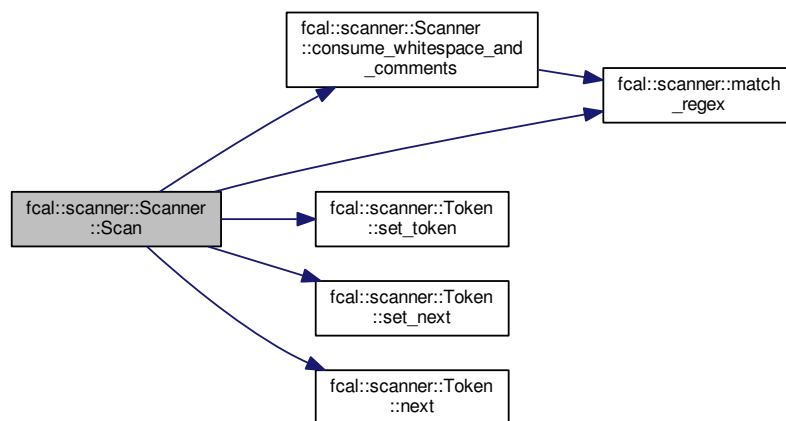
Find the current lexeme using the number of matched characters

Set the current token values

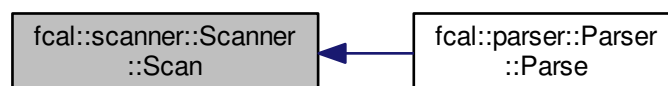
Set previous token to point to current

Add end of file token once all matches are found

Here is the call graph for this function:



Here is the caller graph for this function:



7.39.4 Member Data Documentation

7.39.4.1 `regex_t* fcal::scanner::Scanner::comments` `[private]`

A regex which matches comments.

7.39.4.2 **Token*** fcal::scanner::Scanner::current_token [private]

Token* representing current [Token](#) in the list.

7.39.4.3 **regex_t*** fcal::scanner::Scanner::line_comment [private]

A regex which matches line comments.

7.39.4.4 **Token*** fcal::scanner::Scanner::previous_token [private]

Token* representing previous [Token](#) in the list.

7.39.4.5 **regex_t*** fcal::scanner::Scanner::regex_strings[1+static_cast<int>(kNotOp)] [private]

A regex array created for matching lexeme of Tokens.

7.39.4.6 **Token*** fcal::scanner::Scanner::return_token [private]

Token* representing front of the list.

7.39.4.7 **const char*** fcal::scanner::Scanner::text [private]

String representing contents of the File.

7.39.4.8 **regex_t*** fcal::scanner::Scanner::white_space [private]

A regex which matches white space.

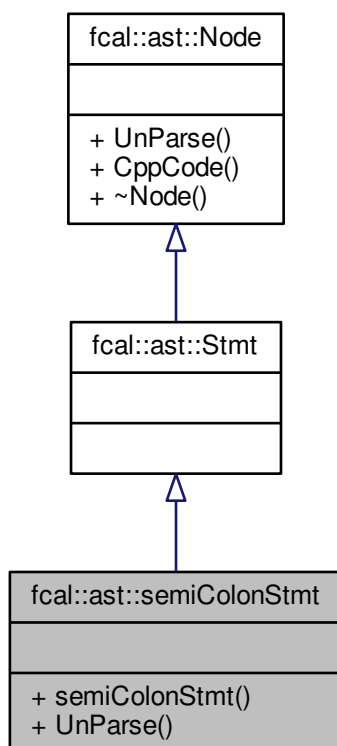
The documentation for this class was generated from the following files:

- [include/scanner_class.h](#)
- [src/scanner_class.cc](#)

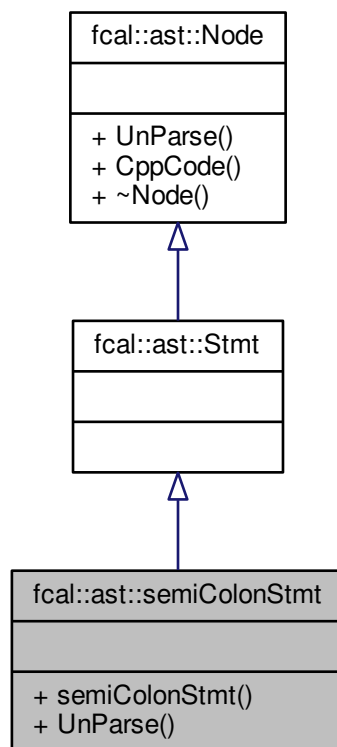
7.40 fcal::ast::semiColonStmt Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::semiColonStmt:



Collaboration diagram for fcal::ast::semiColonStmt:



Public Member Functions

- [semiColonStmt](#) ()
This is a default constructor and does nothing.
- std::string [UnParse](#) (void)
Returns the string : ";".

7.40.1 Detailed Description

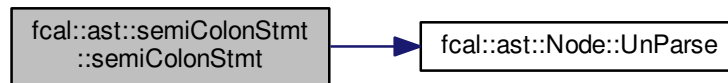
This is a concrete class in the ast class heirarchy. It implements the production, [Stmt](#) ::= ';'.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 fcal::ast::semiColonStmt::semiColonStmt () [inline]

This is a default constructor and does nothing.

Here is the call graph for this function:



7.40.3 Member Function Documentation

7.40.3.1 `std::string fcal::ast::semiColonStmt::UnParse (void)` [virtual]

Returns the string : ";".

Reimplemented from [fcal::ast::Node](#).

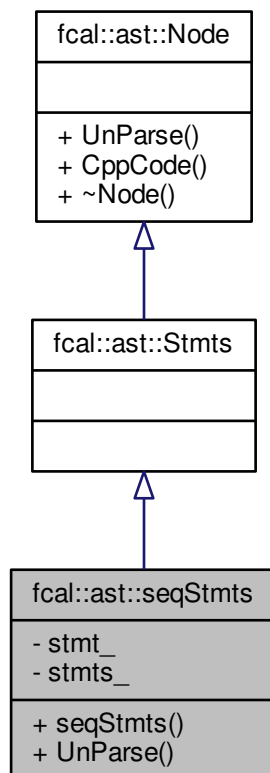
The documentation for this class was generated from the following files:

- [include/ast.h](#)
- [src/ast.cc](#)

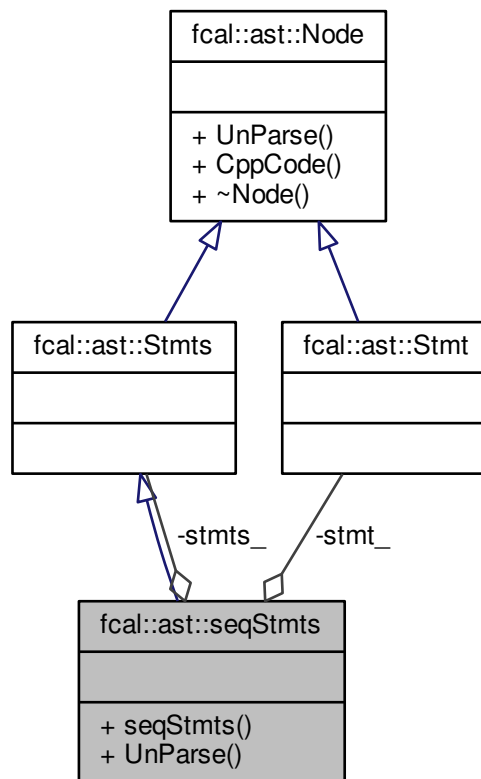
7.41 `fcal::ast::seqStmts` Class Reference

```
#include <ast.h>
```


Inheritance diagram for fcal::ast::seqStmts:



Collaboration diagram for `fcal::ast::seqStmts`:



Public Member Functions

- [seqStmts](#) ([Stmt](#) *stmt, [Stmts](#) *stmts)
- `std::string` [UnParse](#) (void)
Returns a string representing a sequence of [Stmts](#).

Private Attributes

- [Stmt](#) * [stmt_](#)
- [Stmts](#) * [stmts_](#)

7.41.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production, [Stmts](#) ::= [Stmt](#) [Stmts](#)

7.41.2 Constructor & Destructor Documentation

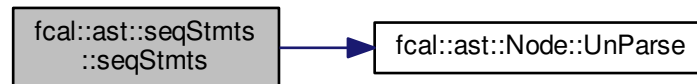
7.41.2.1 `fcal::ast::seqStmts::seqStmts (Stmt * stmt, Stmts * stmts)` `[inline]`

Constructor which takes parameters

Parameters

<i>stmt</i>	a Stmt* which holds the address which contains details of first Stmt in a sequence of Stmt
<i>stmts</i>	a Stmt* which holds the address which contains details of rest of the Stmt after the first Stmt

Here is the call graph for this function:



7.41.3 Member Function Documentation

7.41.3.1 `std::string fcal::ast::seqStmts::UnParse (void)` [virtual]

Returns a string representing a sequence of Stmt.

Returns the string : `stmt_->UnParse() + stmts_->UnParse()`

Reimplemented from `fcal::ast::Node`.

7.41.4 Member Data Documentation

7.41.4.1 `Stmt* fcal::ast::seqStmts::stmt_` [private]

`stmt_` holds the address which contains details of first Stmt in a sequence of Stmt

7.41.4.2 `Stmt* fcal::ast::seqStmts::stmts_` [private]

`stmts_` holds the address which contains details of rest of Stmt after the first Stmt

The documentation for this class was generated from the following files:

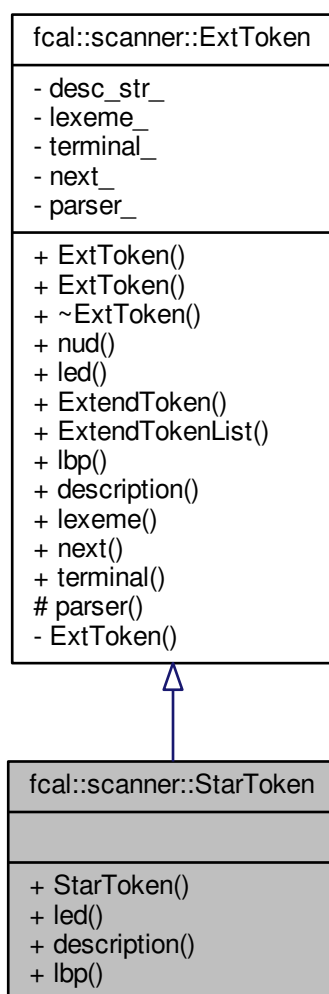
- `include/ast.h`
- `src/ast.cc`

7.42 fcal::scanner::StarToken Class Reference

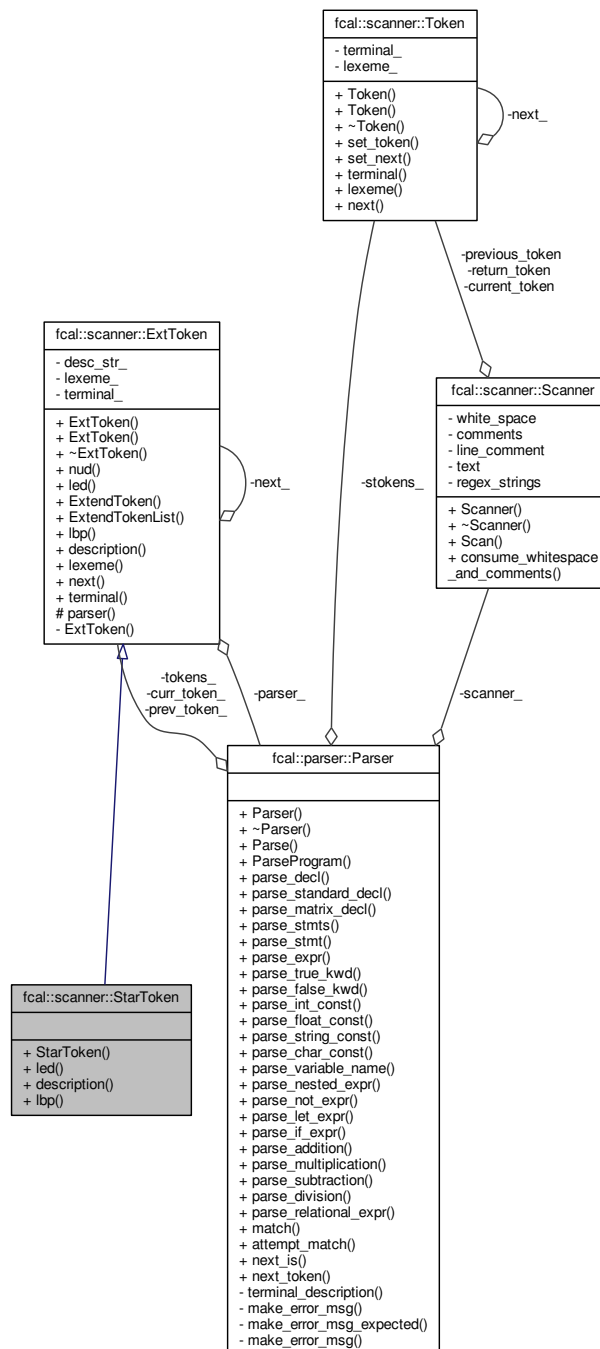
Star.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::StarToken:



Collaboration diagram for fcal::scanner::StarToken:



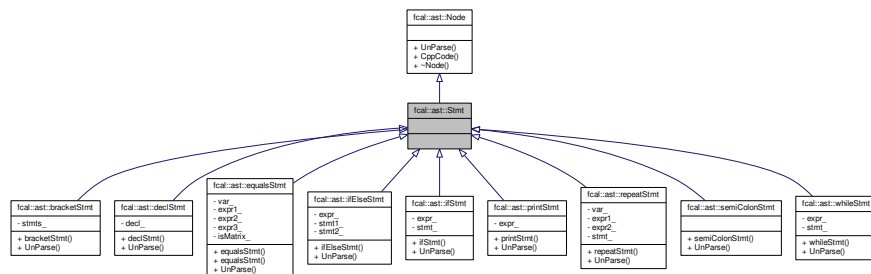
Public Member Functions

- [StarToken](#) ([parser::Parser](#) *p, [Token](#) *t)
- [parser::ParseResult](#) led ([parser::ParseResult](#) left)
- [std::string](#) description ()
- [int](#) lbp ()

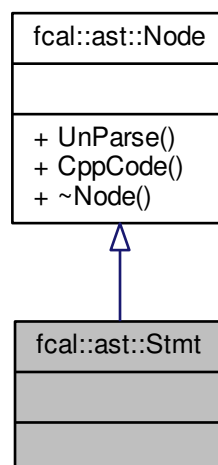
7.43 fcal::ast::Stmt Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::Stmt:



Collaboration diagram for fcal::ast::Stmt:



Additional Inherited Members

7.43.1 Detailed Description

This is the abstract class which inherits from [Node](#). It has no implementation for any functions derived from [Node](#). It is the base class for classes which implement the productions which are derived from the nonterminal 'Stmt' in the FCAL grammar.

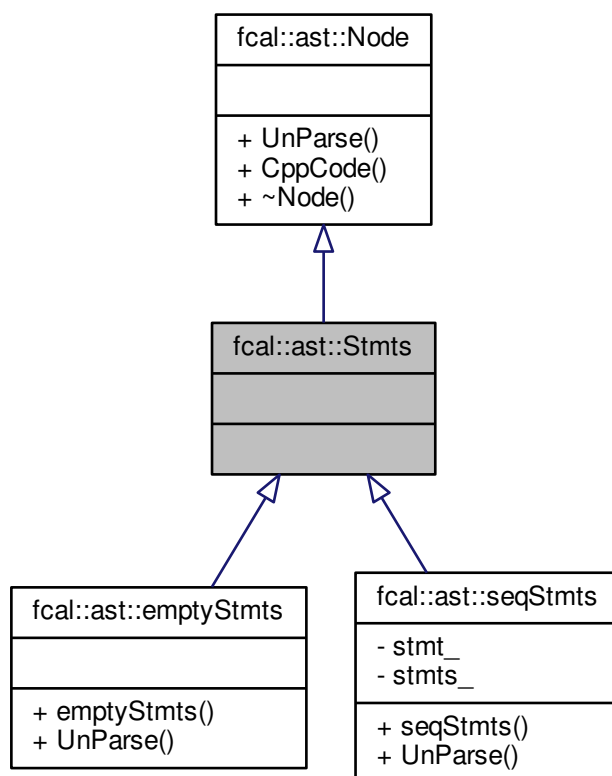
The documentation for this class was generated from the following file:

- [include/ast.h](#)

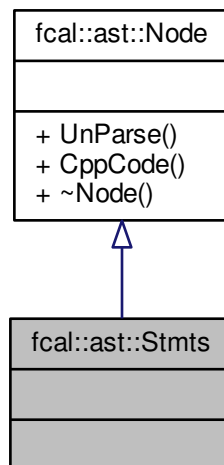
7.44 fcal::ast::Stmts Class Reference

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::Stmts:



Collaboration diagram for fcal::ast::Stmts:



Additional Inherited Members

7.44.1 Detailed Description

This is the abstract class which inherits from [Node](#). It has no implementation for any functions derived from [Node](#). It is the base class for classes which implement the productions which are derived from the nonterminal '[Stmts](#)' in the FCAL grammar.

The documentation for this class was generated from the following file:

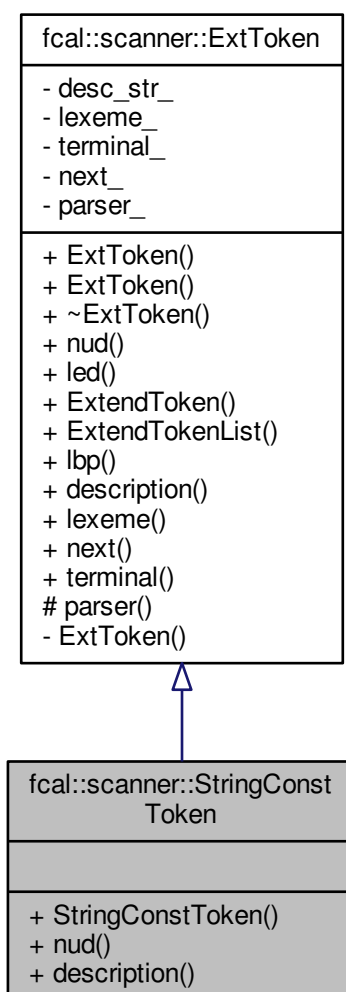
- [include/ast.h](#)

7.45 fcal::scanner::StringConstToken Class Reference

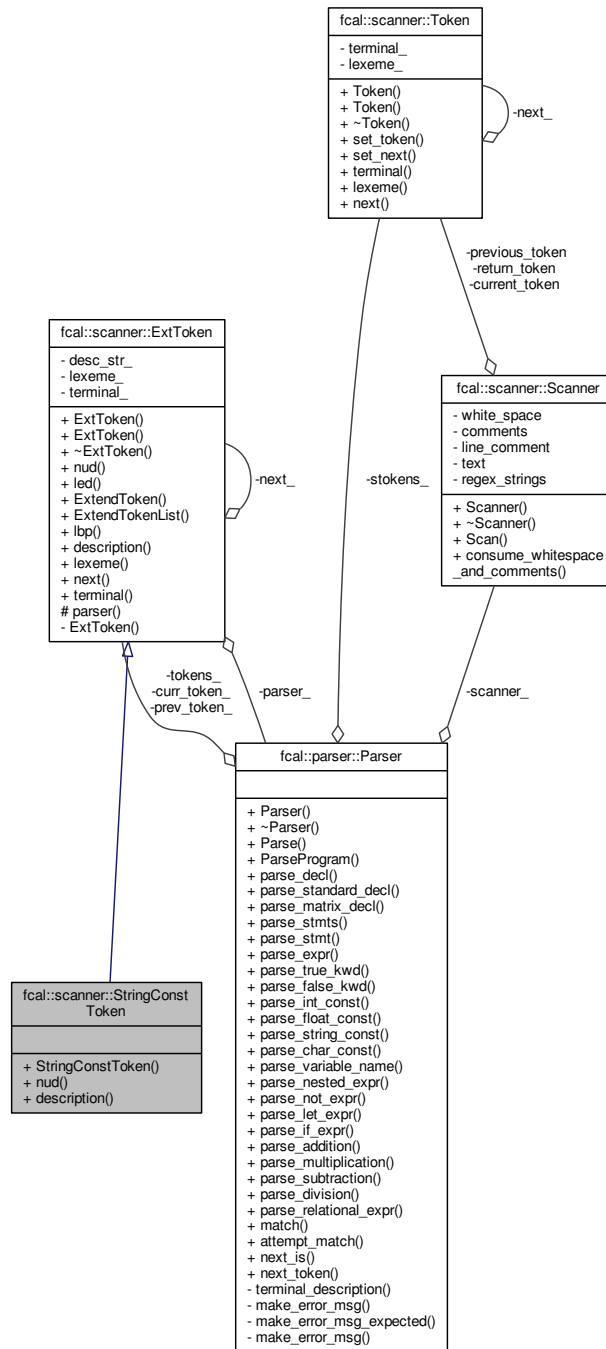
String Const.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::StringConstToken:



Collaboration diagram for fcal::scanner::StringConstToken:



Public Member Functions

- [StringConstToken](#) ([parser::Parser *p](#), [Token *t](#))
- [parser::ParseResult nud](#) ()
- [std::string description](#) ()

Additional Inherited Members

7.45.1 Detailed Description

String Const.

7.45.2 Constructor & Destructor Documentation

7.45.2.1 `fcsl::scanner::StringConstToken::StringConstToken (parser::Parser * p, Token * t)` `[inline]`

7.45.3 Member Function Documentation

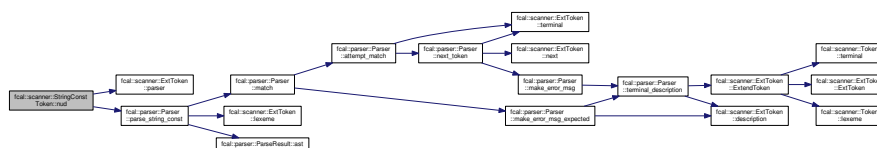
7.45.3.1 `std::string fcsl::scanner::StringConstToken::description ()` `[inline]`, `[virtual]`

Reimplemented from [fcsl::scanner::ExtToken](#).

7.45.3.2 `parser::ParseResult fcsl::scanner::StringConstToken::nud (void)` `[inline]`, `[virtual]`

Reimplemented from [fcsl::scanner::ExtToken](#).

Here is the call graph for this function:



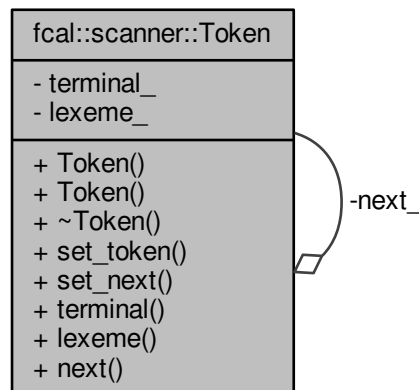
The documentation for this class was generated from the following file:

- [include/ext_token.h](#)

7.46 fcsl::scanner::Token Class Reference

```
#include <token_class.h>
```

Collaboration diagram for fcal::scanner::Token:



Public Member Functions

- `Token ()`
Default constructor for `Token`.
- `Token (std::string, const TokenType, Token *)`
Constructor which initializes all attributes of `Token`.
- `~Token ()`
- `void set_token (TokenType t, std::string lex)`
Set the values of the current token.
- `void set_next (Token *current)`
end set_token
- `TokenType terminal ()`
end set_next
- `std::string lexeme ()`
Get the value of lexeme.
- `Token * next ()`
Get the value of next.

Private Attributes

- `TokenType terminal_`
Type of the `Token`.
- `std::string lexeme_`
lexeme of the `Token`
- `Token * next_`
Next `Token` in the `Token` Array returned by Scan of `Scanner`.

7.46.1 Detailed Description

This class holds information about a `Token`.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 fcal::scanner::Token::Token ()

Default constructor for [Token](#).

7.46.2.2 fcal::scanner::Token::Token (std::string a, const TokenType b, Token * c)

Constructor which initializes all attributes of [Token](#).

Parameters

<i>a</i>	string representing the lexeme of the Token
<i>b</i>	TokenType of the Token
<i>c</i>	Token* which holds address of the next Token in the Token list returned by Scan method of Scanner

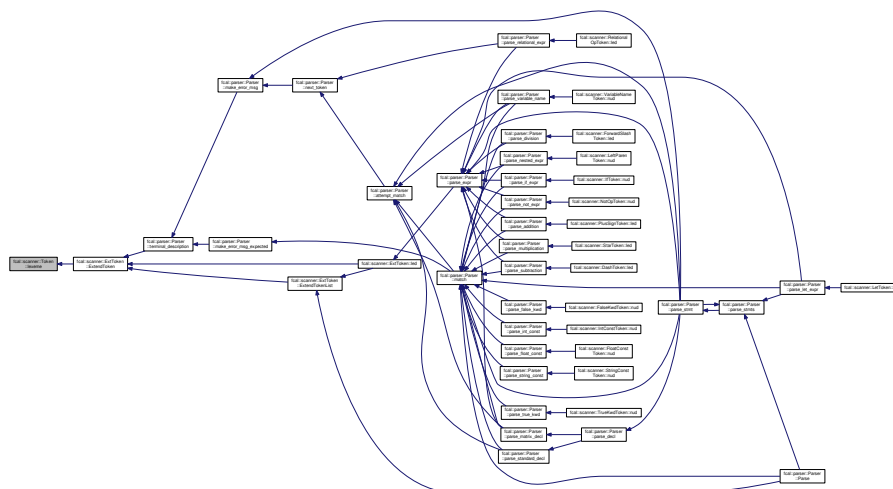
7.46.2.3 fcal::scanner::Token::~~Token ()

7.46.3 Member Function Documentation

7.46.3.1 std::string fcal::scanner::Token::lexeme (void)

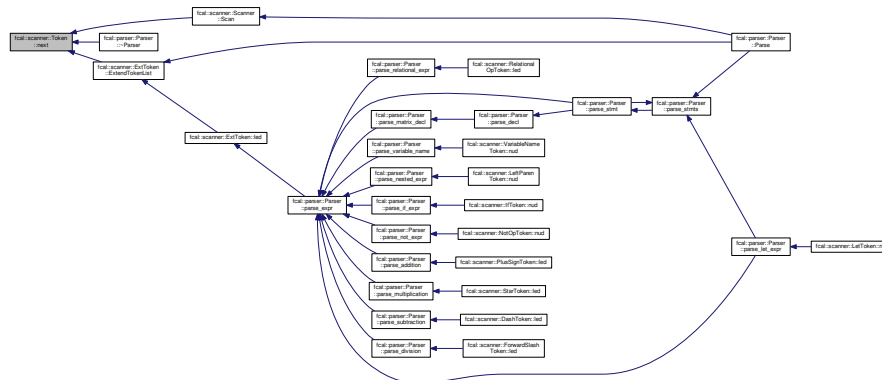
Get the value of lexeme.

Here is the caller graph for this function:



Get the value of next.

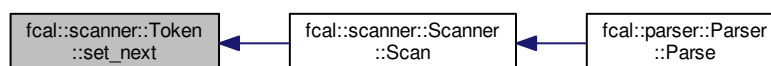
Here is the caller graph for this function:



end set_token

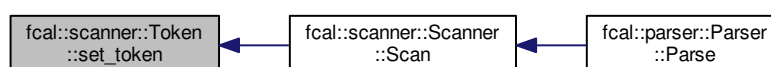
Set the value of the previous token to point to the current token

Here is the caller graph for this function:



Set the values of the current token.

Here is the caller graph for this function:

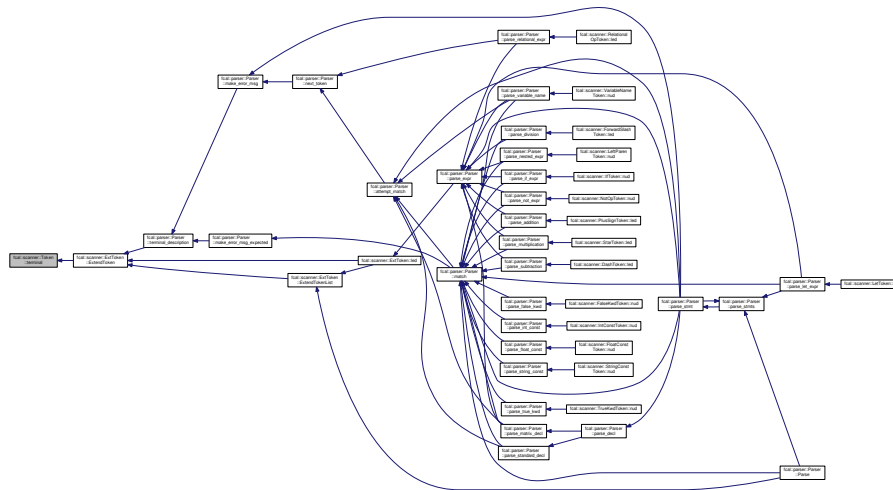


7.46.3.5 TokenType fcal::scanner::Token::terminal (void)

end set_next

Get the value of terminal

Here is the caller graph for this function:



7.46.4 Member Data Documentation

7.46.4.1 std::string fcal::scanner::Token::lexeme_ [private]

lexeme of the [Token](#)

7.46.4.2 Token* fcal::scanner::Token::next_ [private]

Next [Token](#) in the [Token](#) Array returned by Scan of [Scanner](#).

7.46.4.3 TokenType fcal::scanner::Token::terminal_ [private]

Type of the [Token](#).

The documentation for this class was generated from the following files:

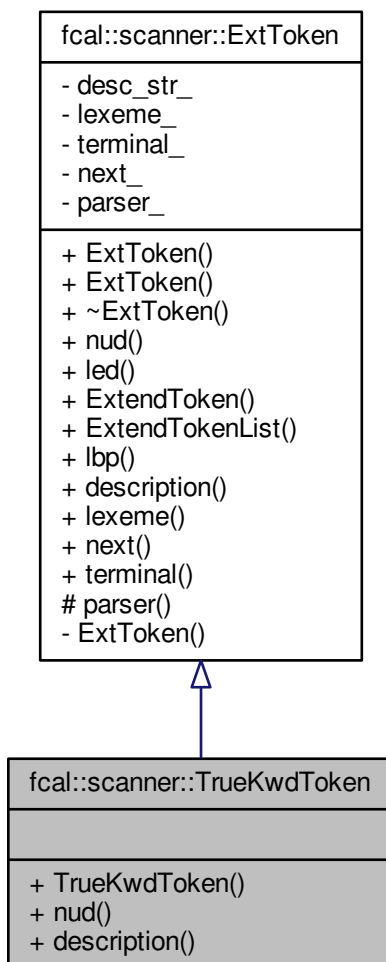
- [include/token_class.h](#)
- [src/token_class.cc](#)

7.47 fcal::scanner::TrueKwdToken Class Reference

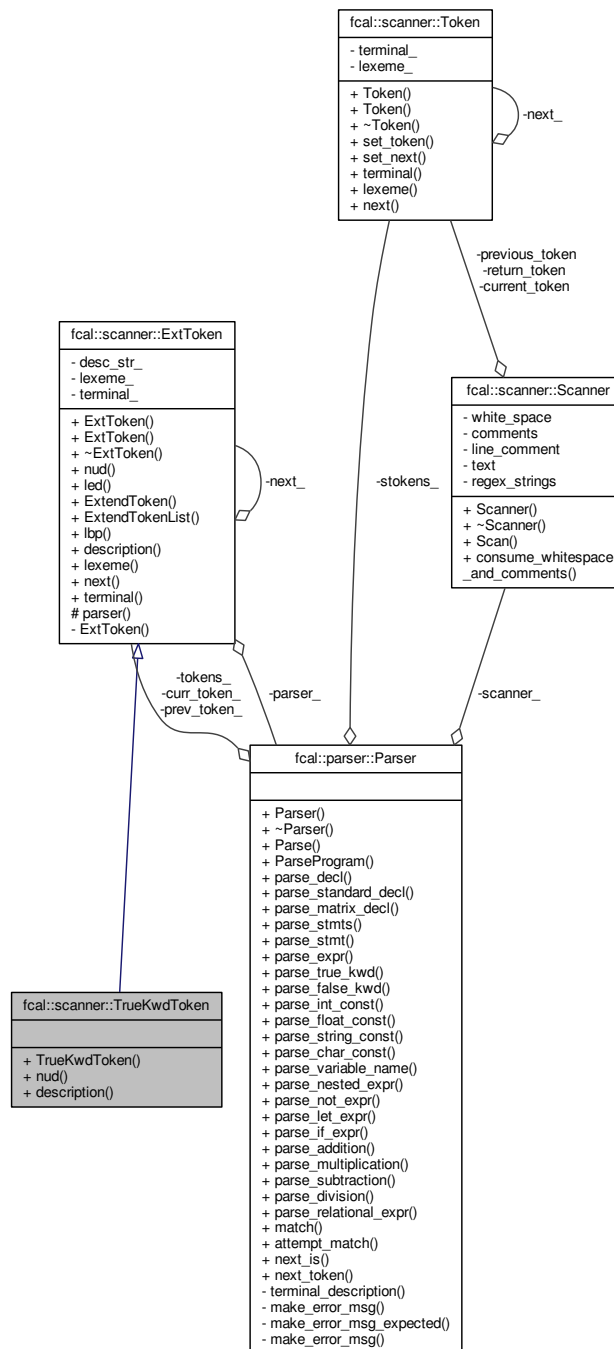
True Kwd.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::TrueKwdToken:



Collaboration diagram for `fcsl::scanner::TrueKwdToken`:



Public Member Functions

- [TrueKwdToken](#) ([parser::Parser *p](#), [Token *t](#))
- [parser::ParseResult nud](#) ()
- [std::string description](#) ()

7.47.1 Detailed Description

7.47.2 Constructor & Destructor Documentation

7.47.3 Member Function Documentation

Reimplemented from `fcsl::scanner::ExtToken`.

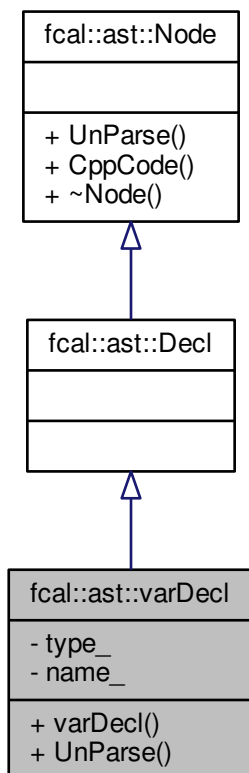
Reimplemented from `fcsl::scanner::ExtToken`.

[illegible]

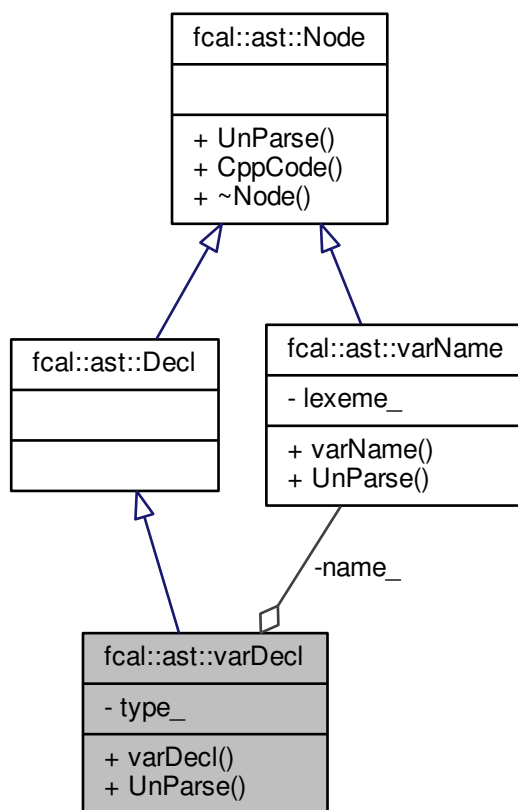
- `include/ext token.h`

```
#include <ast.h>
```

Inheritance diagram for `fcab::ast::varDecl`:



Collaboration diagram for fcal::ast::varDecl:



Public Member Functions

- `varDecl` (`decType` type, `varName` *name)
- `std::string UnParse` (void)

Private Attributes

- `decType type_`
type of declaration
- `varName * name_`
varName declared as a type of type_

7.48.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the productions,

`Decl ::= 'int' varName ';' ;`

`Decl ::= 'float' varName ';' ;`

`Decl ::= 'string' varName ';' ;`

`Decl ::= 'boolean' varName ';' ;`

7.48.2 Constructor & Destructor Documentation

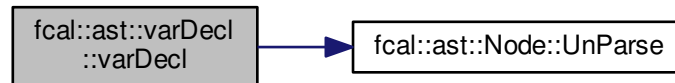
7.48.2.1 `fcal::ast::varDecl::varDecl (decType type, varName * name) [inline]`

This constructor takes two parameters

Parameters

<i>type</i>	a decType which tells the type of declaration
<i>name</i>	a varName used as a declaration

Here is the call graph for this function:



7.48.3 Member Function Documentation

7.48.3.1 `std::string fcal::ast::varDecl::UnParse (void) [virtual]`

Returns a string depending on the value of `type_`.

If `type_ = int_` then return string : "int " + `name_->UnParse()` + ";"

If `type_ = float_` then return string : "float " + `name_->UnParse()` + ";"

If `type_ = string_` then return string : "string " + `name_->UnParse()` + ";"

If `type_ = boolean_` then return string : "boolean " + `name_->UnParse()` + ";"

Reimplemented from [fcal::ast::Node](#).

7.48.4 Member Data Documentation

7.48.4.1 `varName* fcal::ast::varDecl::name_ [private]`

[varName](#) declared as a type of `type_`

7.48.4.2 `decType fcal::ast::varDecl::type_ [private]`

type of declaration

The documentation for this class was generated from the following files:

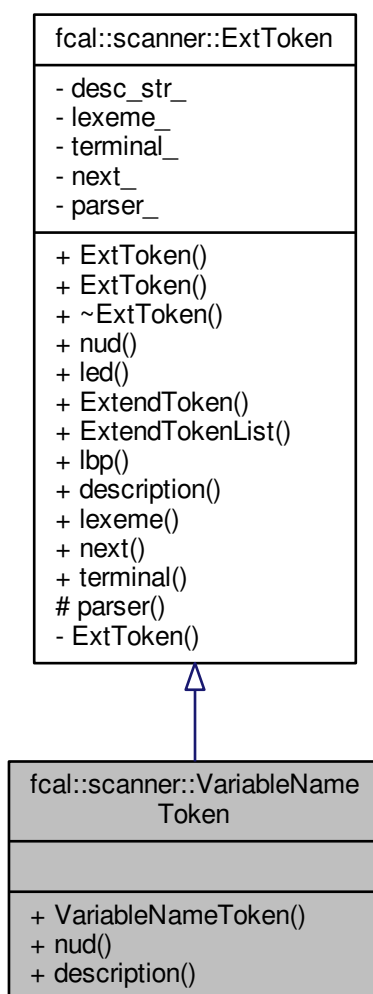
- [include/ast.h](#)
- [src/ast.cc](#)

7.49 fcal::scanner::VariableNameToken Class Reference

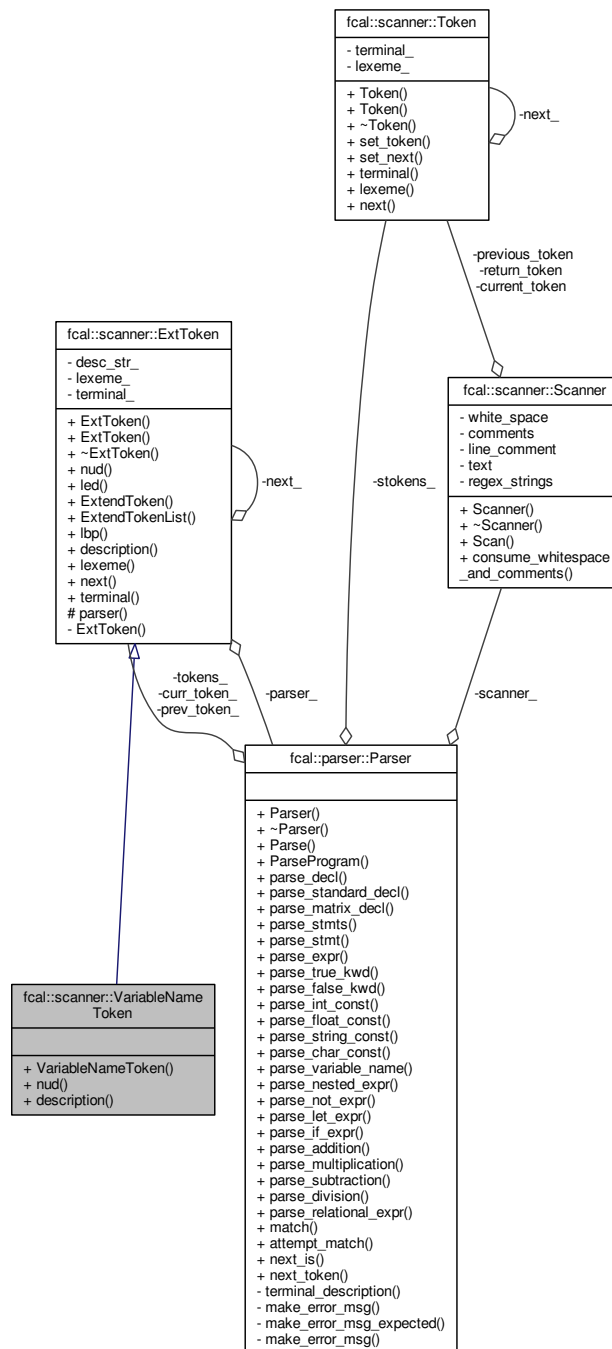
Variable Name.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::VariableNameToken:



Collaboration diagram for `fcsl::scanner::VariableNameToken`:



Public Member Functions

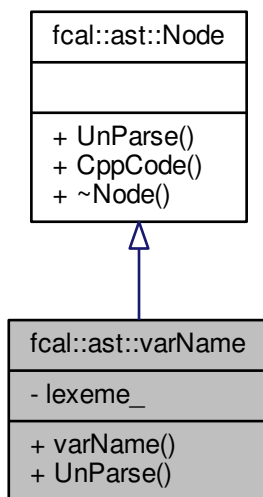
- `VariableNameToken` (`parser::Parser *p`, `Token *t`)
- `parser::ParseResult nud` ()
- `std::string description` ()

7.50 fcal::ast::varName Class Reference

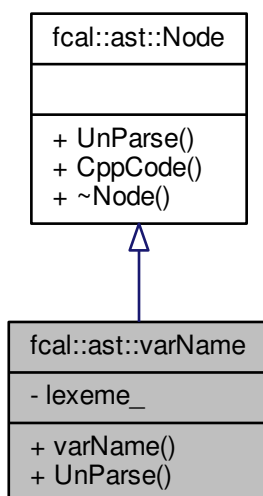
This class holds the lexeme details of a variable name.

```
#include <ast.h>
```

Inheritance diagram for fcal::ast::varName:



Collaboration diagram for fcal::ast::varName:



Public Member Functions

- [varName](#) (std::string lexeme)
- std::string [UnParse](#) (void)

This function returns the variable name's lexeme.

Private Attributes

- std::string [lexeme_](#)

A string holding the lexeme of [varName](#).

7.50.1 Detailed Description

This class holds the lexeme details of a variable name.

7.50.2 Constructor & Destructor Documentation

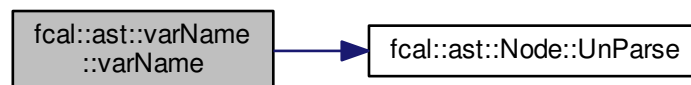
7.50.2.1 fcal::ast::varName::varName (std::string *lexeme*) [inline], [explicit]

This constructor requires one parameter: lexeme

Parameters

<i>lexeme</i>	a string which is the variable name
---------------	-------------------------------------

Here is the call graph for this function:



7.50.3 Member Function Documentation

7.50.3.1 std::string fcal::ast::varName::UnParse (void) [virtual]

This function returns the variable name's lexeme.

returns the string : lexeme_

Reimplemented from [fcal::ast::Node](#).

7.50.4 Member Data Documentation

7.50.4.1 `std::string fcal::ast::varName::lexeme_` [private]

A string holding the lexeme of [varName](#).

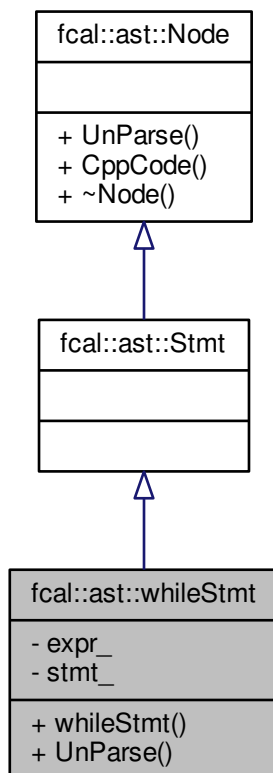
The documentation for this class was generated from the following files:

- [include/ast.h](#)
- [src/ast.cc](#)

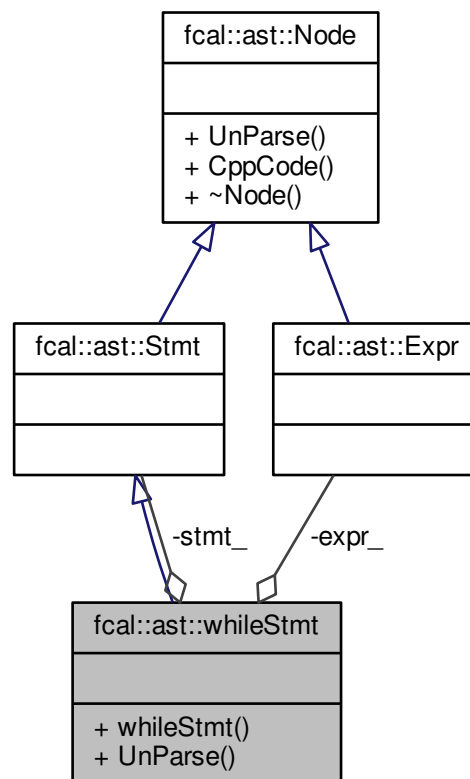
7.51 `fcal::ast::whileStmt` Class Reference

```
#include <ast.h>
```

Inheritance diagram for `fcal::ast::whileStmt`:



Collaboration diagram for fcal::ast::whileStmt:



Public Member Functions

- [whileStmt](#) ([Expr](#) *expr, [Stmt](#) *stmt)
- [std::string UnParse](#) (void)
Returns the string : "while (" + expr_ -> [UnParse\(\)](#) + ")ln" + stmt_ -> [UnParse\(\)](#)

Private Attributes

- [Expr](#) * [expr_](#)
an [Expr](#) evaluated as a condition in while
- [Stmt](#) * [stmt_](#)
stmt_ executed while expr_ is still True

7.51.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
[Stmt](#) ::= 'while' '(' [Expr](#) ')' [Stmt](#)

7.51.2 Constructor & Destructor Documentation

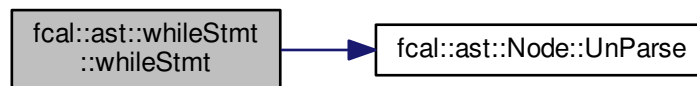
7.51.2.1 `fcal::ast::whileStmt::whileStmt (Expr * expr, Stmt * stmt)` `[inline]`

The constructor takes two arguments

Parameters

<i>expr</i>	an Expr evaluated as a condition in while
<i>stmt</i>	a Stmt executed while the Expr is still True

Here is the call graph for this function:



7.51.3 Member Function Documentation

7.51.3.1 `std::string fcal::ast::whileStmt::UnParse (void)` `[virtual]`

Returns the string : "while (" + `expr_>UnParse()` + ")\n" + `stmt_>UnParse()`

Reimplemented from [fcal::ast::Node](#).

7.51.4 Member Data Documentation

7.51.4.1 `Expr* fcal::ast::whileStmt::expr_` `[private]`

an [Expr](#) evaluated as a condition in while

7.51.4.2 `Stmt* fcal::ast::whileStmt::stmt_` `[private]`

`stmt_` executed while `expr_` is still True

The documentation for this class was generated from the following files:

- `include/ast.h`
- `src/ast.cc`

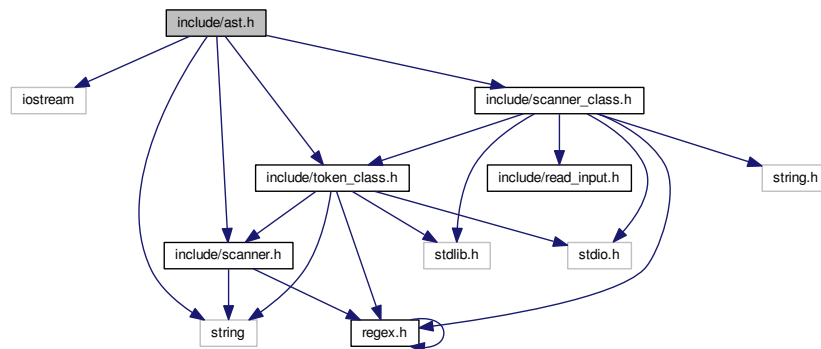
Chapter 8

File Documentation

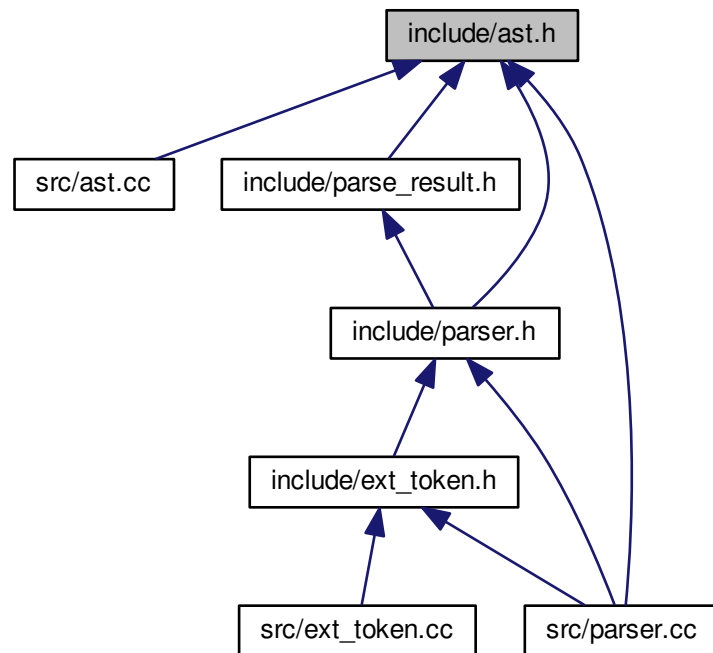
8.1 include/ast.h File Reference

```
#include <iostream>
#include <string>
#include "include/scanner.h"
#include "include/scanner_class.h"
#include "include/token_class.h"
```

Include dependency graph for ast.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [fcal::ast::Node](#)
- class [fcal::ast::varName](#)
 - This class holds the lexeme details of a variable name.*
- class [fcal::ast::Stmts](#)
- class [fcal::ast::Stmt](#)
- class [fcal::ast::Decl](#)
- class [fcal::ast::Expr](#)
- class [fcal::ast::Root](#)
- class [fcal::ast::emptyStmts](#)
- class [fcal::ast::seqStmts](#)
- class [fcal::ast::declStmt](#)
- class [fcal::ast::bracketStmt](#)
- class [fcal::ast::ifStmt](#)
- class [fcal::ast::ifElseStmt](#)
- class [fcal::ast::equalsStmt](#)
- class [fcal::ast::printStmt](#)
- class [fcal::ast::repeatStmt](#)
- class [fcal::ast::whileStmt](#)
- class [fcal::ast::semiColonStmt](#)
- class [fcal::ast::varDecl](#)
- class [fcal::ast::matrixDecl](#)
- class [fcal::ast::constantExpr](#)

- class `fcal::ast::binaryExpr`
- class `fcal::ast::boolExpr`
- class `fcal::ast::matrixExpr`
- class `fcal::ast::nestedOrExpr`
- class `fcal::ast::parenthesisExpr`
- class `fcal::ast::letExpr`
- class `fcal::ast::ifExpr`
- class `fcal::ast::notExpr`

Namespaces

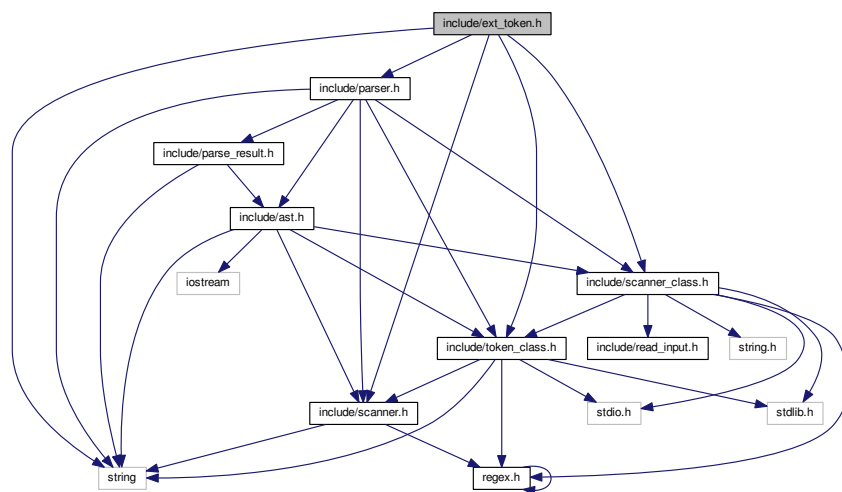
- `fcal`
Namespaces.
- `fcal::ast`

Enumerations

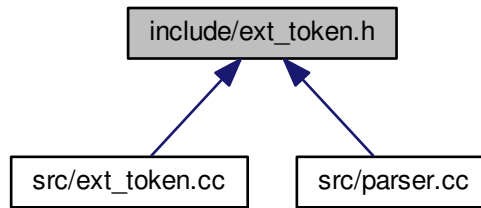
- enum `fcal::ast::decType` { `fcal::ast::int_`, `fcal::ast::float_`, `fcal::ast::string_`, `fcal::ast::boolean_` }

8.2 include/ext_token.h File Reference

```
#include <string>
#include "include/parser.h"
#include "include/scanner.h"
#include "include/token_class.h"
#include "include/scanner_class.h"
Include dependency graph for ext_token.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [fcal::scanner::ExtToken](#)
- class [fcal::scanner::NotOpToken](#)
- class [fcal::scanner::TrueKwdToken](#)
True Kwd.
- class [fcal::scanner::FalseKwdToken](#)
False Kwd.
- class [fcal::scanner::IntConstToken](#)
Int Const.
- class [fcal::scanner::FloatConstToken](#)
Float Const.
- class [fcal::scanner::StringConstToken](#)
String Const.
- class [fcal::scanner::CharConstToken](#)
Char Const.
- class [fcal::scanner::VariableNameToken](#)
Variable Name.
- class [fcal::scanner::IfToken](#)
- class [fcal::scanner::LetToken](#)
- class [fcal::scanner::LeftParenToken](#)
Left Paren.
- class [fcal::scanner::PlusSignToken](#)
Plus Sign.
- class [fcal::scanner::StarToken](#)
Star.
- class [fcal::scanner::DashToken](#)
Dash.
- class [fcal::scanner::ForwardSlashToken](#)
ForwardSlash.
- class [fcal::scanner::RelationalOpToken](#)
Relational Op.
- class [fcal::scanner::EndOfFileToken](#)
End of File.

Namespaces

- [fcal](#)

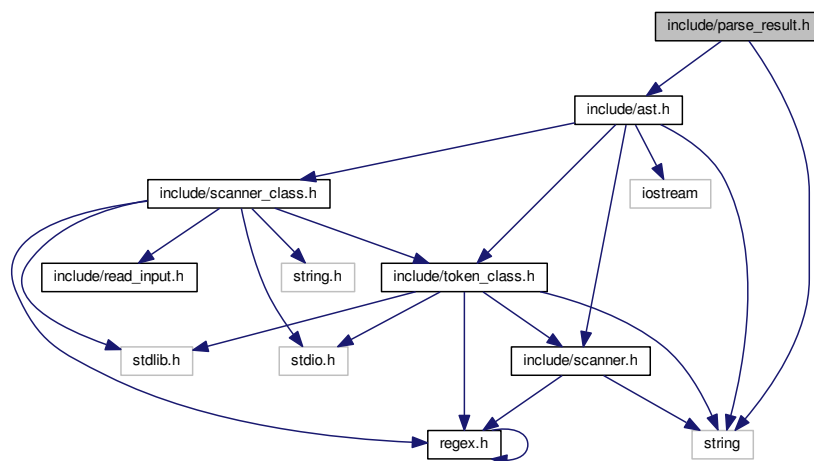
Namespaces.

- [fcal::scanner](#)

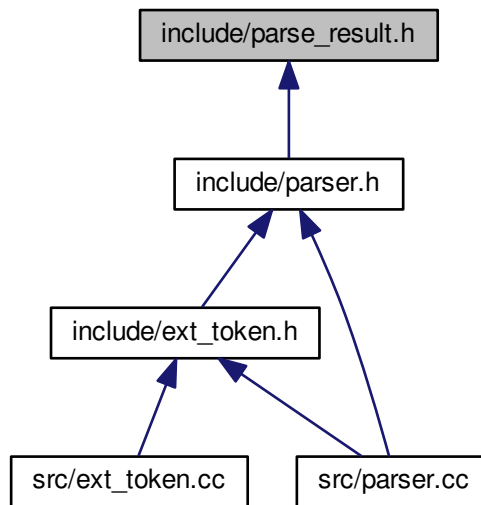
8.3 include/mainpage.h File Reference

8.4 include/parse_result.h File Reference

```
#include <string>
#include "include/ast.h"
Include dependency graph for parse_result.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [fcal::parser::ParseResult](#)

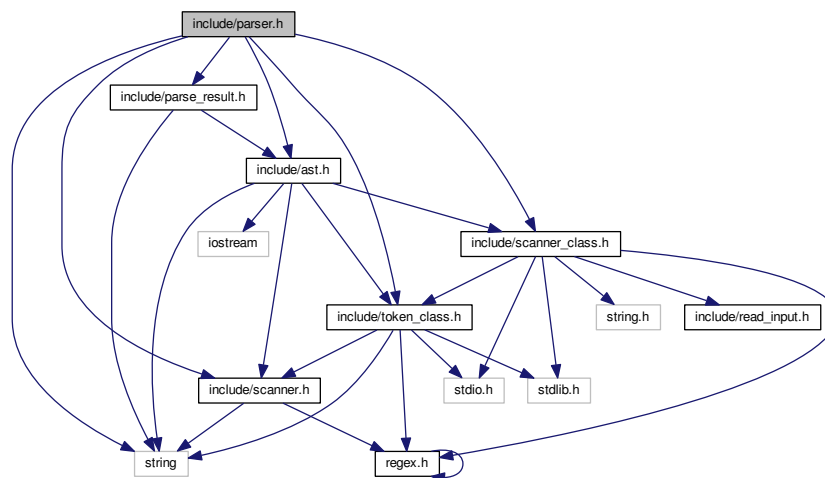
Namespaces

- [fcal](#)
 - Namespaces.*
- [fcal::parser](#)

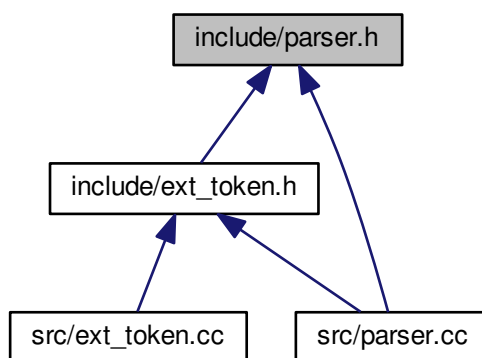
8.5 include/parser.h File Reference

```
#include <string>
#include "include/parse_result.h"
#include "include/scanner.h"
#include "include/scanner_class.h"
#include "include/token_class.h"
#include "include/ast.h"
```

Include dependency graph for parser.h:



This graph shows which files directly or indirectly include this file:



Classes

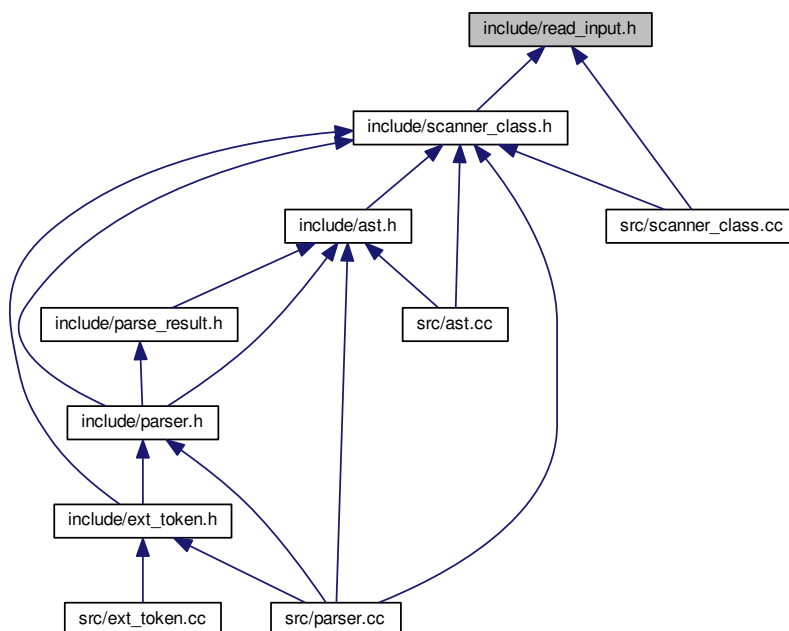
- class [fcal::parser::Parser](#)

Namespaces

- [fcal](#)
Namespaces.
- [fcal::scanner](#)
- [fcal::parser](#)

8.6 include/read_input.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- [fcal](#)
Namespaces.
- [fcal::scanner](#)

Functions

- `char * fcal::scanner::ReadInput (int argc, char **argv)`
- `char * fcal::scanner::ReadInputFromFile (const char *filename)`

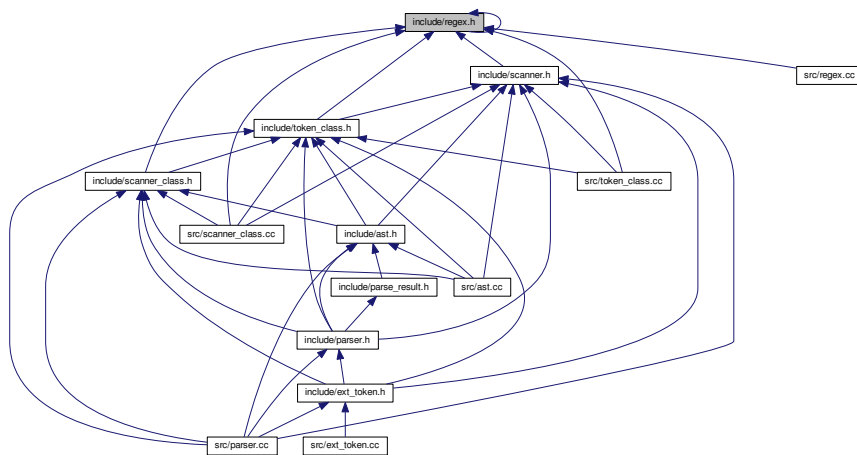
8.7 include/regex.h File Reference

```
#include <regex.h>
```

Include dependency graph for `regex.h`:



This graph shows which files directly or indirectly include this file:



Namespaces

- `fcal`
Namespaces.
- `fcal::scanner`

Functions

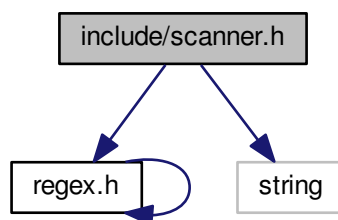
- `regex_t * fcal::scanner::make_regex (const char *pattern)`
- `int fcal::scanner::match_regex (regex_t *re, const char *text)`

Variables

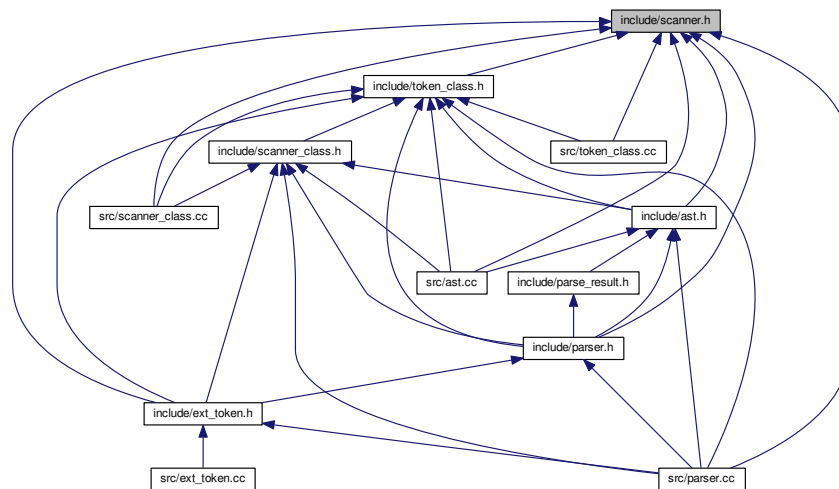
- `const int fcal::scanner::kRegexNSub = 1`

8.8 include/scanner.h File Reference

```
#include <regex.h>
#include <string>
Include dependency graph for scanner.h:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- [fcal](#)
Namespaces.
- [fcal::scanner](#)

Typedefs

- typedef enum kTokenEnumType [fcal::scanner::TokenType](#)

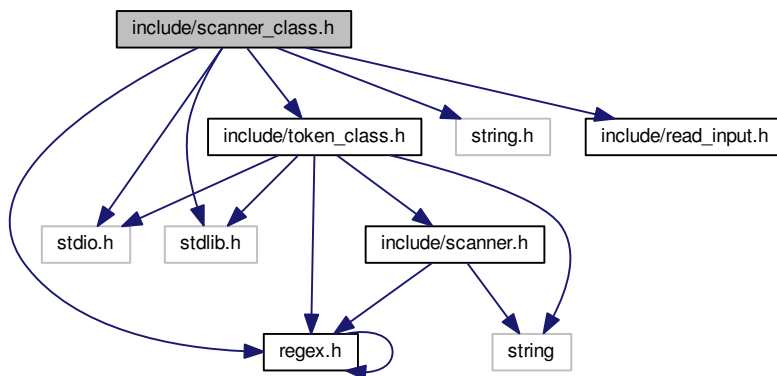
Enumerations

- enum [fcal::scanner::kTokenEnumType](#) {
[fcal::scanner::kIntKwd](#), [fcal::scanner::kFloatKwd](#), [fcal::scanner::kBoolKwd](#), [fcal::scanner::kTrueKwd](#),
[fcal::scanner::kFalseKwd](#), [fcal::scanner::kStringKwd](#), [fcal::scanner::kMatrixKwd](#), [fcal::scanner::kLetKwd](#),
[fcal::scanner::kInKwd](#), [fcal::scanner::kEndKwd](#), [fcal::scanner::kIfKwd](#), [fcal::scanner::kThenKwd](#),
[fcal::scanner::kElseKwd](#), [fcal::scanner::kRepeatKwd](#), [fcal::scanner::kWhileKwd](#), [fcal::scanner::kPrintKwd](#),
[fcal::scanner::kToKwd](#), [fcal::scanner::kIntConst](#), [fcal::scanner::kFloatConst](#), [fcal::scanner::kStringConst](#),
[fcal::scanner::kVariableName](#), [fcal::scanner::kLeftParen](#), [fcal::scanner::kRightParen](#), [fcal::scanner::kLeftCurly](#),
[fcal::scanner::kRightCurly](#), [fcal::scanner::kLeftSquare](#), [fcal::scanner::kRightSquare](#), [fcal::scanner::kSemiColon](#),
[fcal::scanner::kColon](#), [fcal::scanner::kAssign](#), [fcal::scanner::kPlusSign](#), [fcal::scanner::kStar](#),
[fcal::scanner::kDash](#), [fcal::scanner::kForwardSlash](#), [fcal::scanner::kLessThan](#), [fcal::scanner::kLessThanEqual](#),
[fcal::scanner::kGreaterThan](#), [fcal::scanner::kGreaterThanEqual](#), [fcal::scanner::kEqualsEquals](#), [fcal::scanner::kNotEquals](#),
[fcal::scanner::kAndOp](#), [fcal::scanner::kOrOp](#), [fcal::scanner::kNotOp](#), [fcal::scanner::kEndOfFile](#),
[fcal::scanner::kLexicalError](#) }

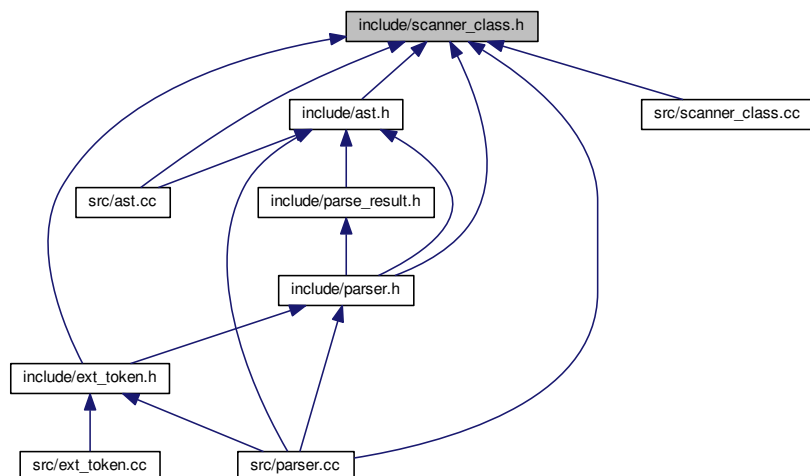
8.9 include/scanner_class.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "include/token_class.h"
#include "include/read_input.h"
#include "include/regex.h"
```

Include dependency graph for scanner_class.h:



This graph shows which files directly or indirectly include this file:



Classes

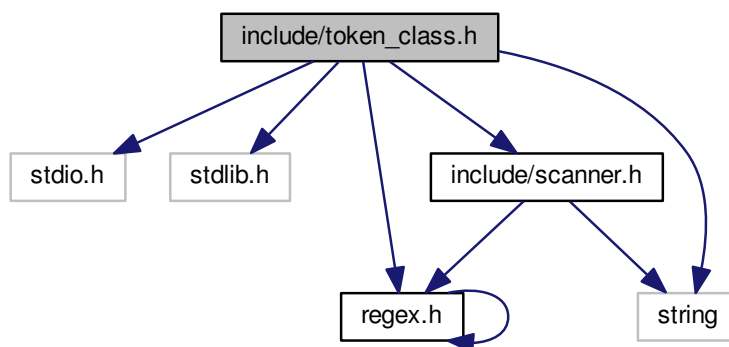
- class `fcal::scanner::Scanner`

Namespaces

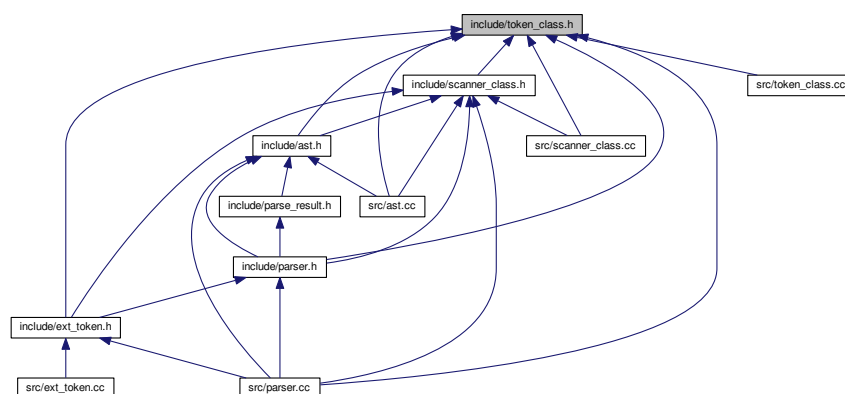
- [fcal](#)
Namespaces.
- [fcal::scanner](#)

8.10 include/token_class.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <regex.h>
#include <string>
#include "include/scanner.h"
Include dependency graph for token_class.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [fcal::scanner::Token](#)

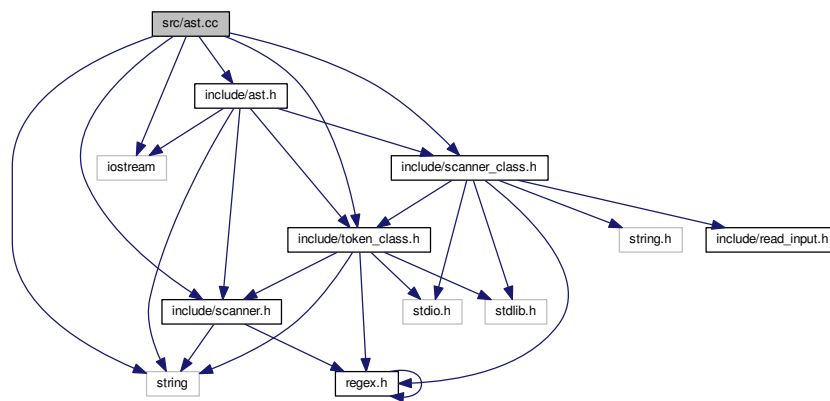
Namespaces

- [fcal](#)
- Namespaces.
- [fcal::scanner](#)

8.11 src/ast.cc File Reference

```
#include <iostream>
#include <string>
#include "include/scanner.h"
#include "include/scanner_class.h"
#include "include/token_class.h"
#include "include/ast.h"
```

Include dependency graph for ast.cc:



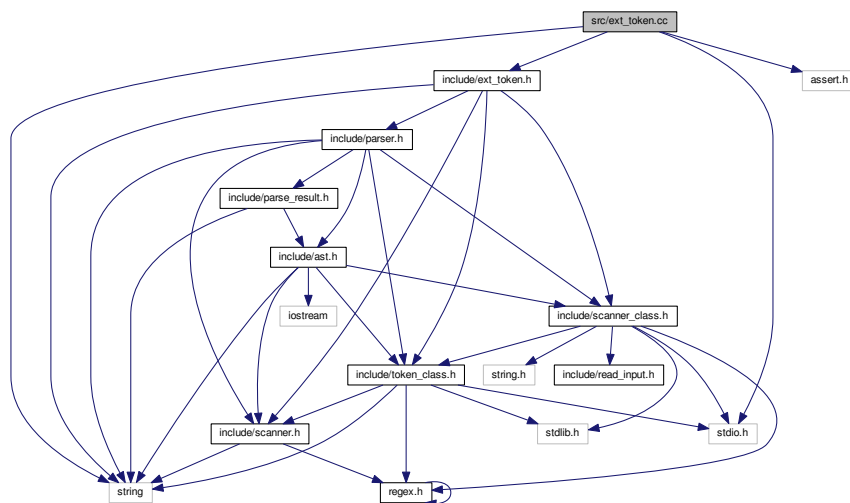
Namespaces

- [fcal](#)
- Namespaces.
- [fcal::ast](#)

8.12 src/ext_token.cc File Reference

```
#include "include/ext_token.h"
#include <assert.h>
#include <stdio.h>
#include <string>
```

Include dependency graph for ext_token.cc:



Namespaces

- [fcal](#)

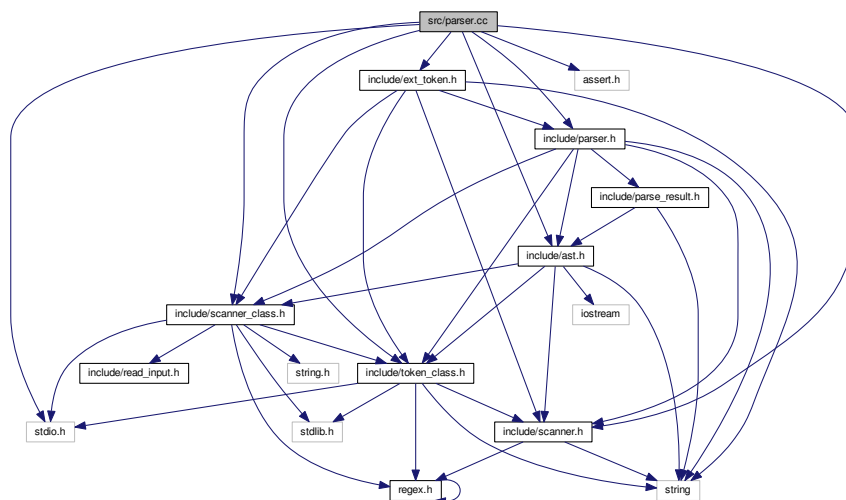
Namespaces.

- [fcal::scanner](#)

8.13 src/parser.cc File Reference

```
#include "include/parser.h"
#include <assert.h>
#include <stdio.h>
#include "include/ext_token.h"
#include "include/scanner.h"
#include "include/scanner_class.h"
#include "include/token_class.h"
#include "include/ast.h"
```

Include dependency graph for parser.cc:



Namespaces

- [fcal](#)

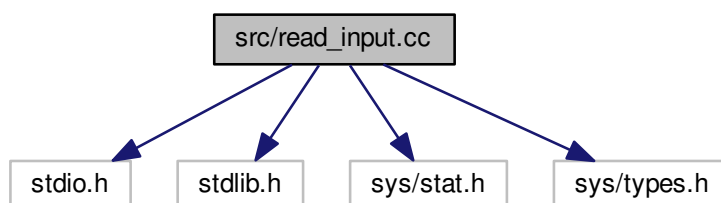
Namespaces.

- [fcal::parser](#)

8.14 src/read_input.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
```

Include dependency graph for read_input.cc:



Namespaces

- [fcal](#)

Namespaces.

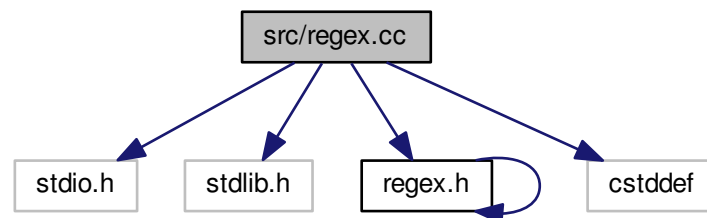
- [fcal::scanner](#)

Functions

- char * [fcal::scanner::ReadInputFromFile](#) (const char *filename)
- char * [fcal::scanner::ReadInput](#) (int argc, char **argv)

8.15 src/regex.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <regex.h>
#include <cstdint>
Include dependency graph for regex.cc:
```



Namespaces

- [fcal](#)
Namespaces.
- [fcal::scanner](#)

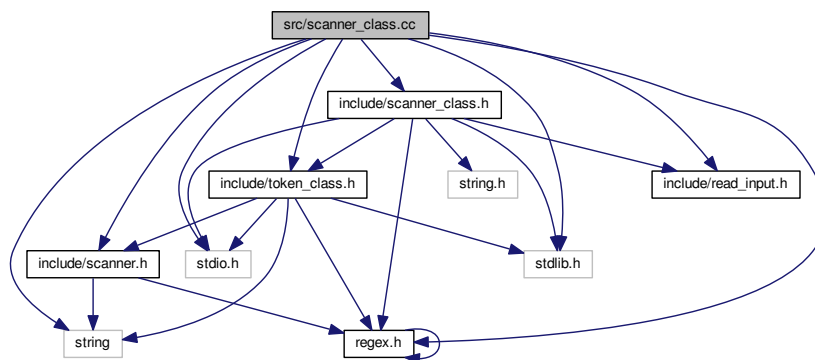
Functions

- regex_t * [fcal::scanner::make_regex](#) (const char *pattern)
- int [fcal::scanner::match_regex](#) (regex_t *re, const char *text)

8.16 src/scanner_class.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include "include/scanner.h"
#include "include/scanner_class.h"
#include "include/token_class.h"
#include "include/regex.h"
#include "include/read_input.h"
```

Include dependency graph for scanner_class.cc:



Namespaces

- [fcal](#)
Namespaces.
- [fcal::scanner](#)

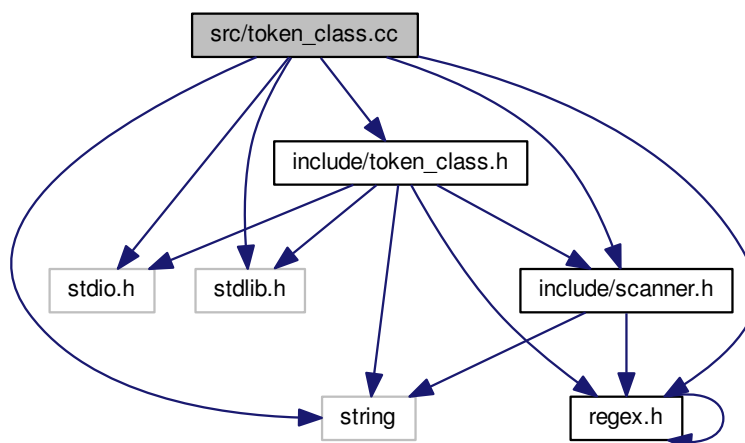
8.17 src/token_class.cc File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <regex.h>
#include <string>
#include "include/token_class.h"
#include "include/scanner.h"

```

Include dependency graph for token_class.cc:



Namespaces

- [fcal](#)
Namespaces.
- [fcal::scanner](#)

Index

- ~ExtToken
 - fcgal::scanner::ExtToken, [55](#)
- ~Node
 - fcgal::ast::Node, [107](#)
- ~Parser
 - fcgal::parser::Parser, [122](#)
- ~Scanner
 - fcgal::scanner::Scanner, [166](#)
- ~Token
 - fcgal::scanner::Token, [186](#)
- ast
 - fcgal::parser::ParseResult, [147](#)
- ast_
 - fcgal::parser::ParseResult, [148](#)
- attempt_match
 - fcgal::parser::Parser, [123](#)
- binaryExpr
 - fcgal::ast::binaryExpr, [21](#)
- binaryOp_
 - fcgal::ast::binaryExpr, [21](#)
- boolExpr
 - fcgal::ast::boolExpr, [24](#)
- boolean_
 - fcgal::ast, [12](#)
 - fcgal::ast::boolExpr, [24](#)
- bracketStmt
 - fcgal::ast::bracketStmt, [26](#)
- CharConstToken
 - fcgal::scanner::CharConstToken, [30](#)
- comments
 - fcgal::scanner::Scanner, [168](#)
- constant_
 - fcgal::ast::constantExpr, [33](#)
- constantExpr
 - fcgal::ast::constantExpr, [32](#)
- consume_whitespace_and_comments
 - fcgal::scanner::Scanner, [166](#)
- CppCode
 - fcgal::ast::Node, [108](#)
- curr_token_
 - fcgal::parser::Parser, [145](#)
- current_token
 - fcgal::scanner::Scanner, [168](#)
- DashToken
 - fcgal::scanner::DashToken, [36](#)
- decType
 - fcgal::ast, [12](#)
- decl_
 - fcgal::ast::declStmt, [41](#)
- declStmt
 - fcgal::ast::declStmt, [40](#)
- desc_str_
 - fcgal::scanner::ExtToken, [62](#)
- description
 - fcgal::scanner::CharConstToken, [30](#)
 - fcgal::scanner::DashToken, [36](#)
 - fcgal::scanner::EndOfFileToken, [47](#)
 - fcgal::scanner::ExtToken, [56](#)
 - fcgal::scanner::FalseKwdToken, [65](#)
 - fcgal::scanner::FloatConstToken, [68](#)
 - fcgal::scanner::ForwardSlashToken, [71](#)
 - fcgal::scanner::IfToken, [83](#)
 - fcgal::scanner::IntConstToken, [86](#)
 - fcgal::scanner::LeftParenToken, [89](#)
 - fcgal::scanner::LetToken, [96](#)
 - fcgal::scanner::NotOpToken, [116](#)
 - fcgal::scanner::PlusSignToken, [151](#)
 - fcgal::scanner::StarToken, [178](#)
 - fcgal::scanner::StringConstToken, [184](#)
 - fcgal::scanner::TrueKwdToken, [191](#)
 - fcgal::scanner::VariableNameToken, [197](#)
- emptyStmts
 - fcgal::ast::emptyStmts, [43](#)
- EndOfFileToken
 - fcgal::scanner::EndOfFileToken, [47](#)
- equalsStmt
 - fcgal::ast::equalsStmt, [50](#)
- errors
 - fcgal::parser::ParseResult, [147](#)
- errors_
 - fcgal::parser::ParseResult, [148](#)
- expr1_
 - fcgal::ast::binaryExpr, [21](#)
 - fcgal::ast::boolExpr, [24](#)
 - fcgal::ast::equalsStmt, [51](#)
 - fcgal::ast::ifExpr, [77](#)
 - fcgal::ast::matrixDecl, [100](#)
 - fcgal::ast::matrixExpr, [103](#)
 - fcgal::ast::repeatStmt, [160](#)
- expr2_
 - fcgal::ast::binaryExpr, [22](#)
 - fcgal::ast::boolExpr, [25](#)
 - fcgal::ast::equalsStmt, [51](#)
 - fcgal::ast::ifExpr, [77](#)
 - fcgal::ast::matrixDecl, [100](#)

- fcval::ast::matrixExpr, 103
 - fcval::ast::repeatStmt, 160
- expr3_
 - fcval::ast::equalsStmt, 51
 - fcval::ast::ifExpr, 77
 - fcval::ast::matrixDecl, 100
- expr_
 - fcval::ast::ifElseStmt, 74
 - fcval::ast::ifStmt, 80
 - fcval::ast::letExpr, 93
 - fcval::ast::nestedOrExpr, 106
 - fcval::ast::notExpr, 113
 - fcval::ast::parenthesisExpr, 119
 - fcval::ast::printStmt, 154
 - fcval::ast::whileStmt, 202
- ExtToken
 - fcval::scanner::ExtToken, 55
- ExtendToken
 - fcval::scanner::ExtToken, 56
- ExtendTokenList
 - fcval::scanner::ExtToken, 57
- FalseKwdToken
 - fcval::scanner::FalseKwdToken, 65
- fcval, 11
- fcval::ast, 11
 - boolean_, 12
 - decType, 12
 - float_, 12
 - int_, 12
 - string_, 12
- fcval::ast::Decl, 37
- fcval::ast::Expr, 52
- fcval::ast::Node, 107
 - ~Node, 107
 - CppCode, 108
 - UnParse, 108
- fcval::ast::Root, 161
 - name_, 164
 - Root, 163
 - stmts_, 164
 - UnParse, 164
- fcval::ast::Stmt, 179
- fcval::ast::Stmts, 180
- fcval::ast::binaryExpr, 19
 - binaryExpr, 21
 - binaryOp_, 21
 - expr1_, 21
 - expr2_, 22
 - UnParse, 21
- fcval::ast::boolExpr, 22
 - boolExpr, 24
 - boolean_, 24
 - expr1_, 24
 - expr2_, 25
 - UnParse, 24
- fcval::ast::bracketStmt, 25
 - bracketStmt, 26
 - stmts_, 27
- UnParse, 27
- fcval::ast::constantExpr, 31
 - constant_, 33
 - constantExpr, 32
 - UnParse, 33
- fcval::ast::declStmt, 38
 - decl_, 41
 - declStmt, 40
 - UnParse, 41
- fcval::ast::emptyStmts, 41
 - emptyStmts, 43
 - UnParse, 44
- fcval::ast::equalsStmt, 47
 - equalsStmt, 50
 - expr1_, 51
 - expr2_, 51
 - expr3_, 51
 - isMatrix_, 51
 - UnParse, 50
 - var_, 51
- fcval::ast::ifElseStmt, 72
 - expr_, 74
 - ifElseStmt, 74
 - stmt1_, 74
 - stmt2_, 74
 - UnParse, 74
- fcval::ast::ifExpr, 75
 - expr1_, 77
 - expr2_, 77
 - expr3_, 77
 - ifExpr, 77
 - UnParse, 77
- fcval::ast::ifStmt, 78
 - expr_, 80
 - ifStmt, 80
 - stmt_, 80
 - UnParse, 80
- fcval::ast::letExpr, 90
 - expr_, 93
 - letExpr, 92
 - stmts_, 93
 - UnParse, 93
- fcval::ast::matrixDecl, 97
 - expr1_, 100
 - expr2_, 100
 - expr3_, 100
 - matrixDecl, 99
 - simpleMatrix_, 100
 - UnParse, 100
 - var1_, 100
 - var2_, 101
 - var3_, 101
- fcval::ast::matrixExpr, 101
 - expr1_, 103
 - expr2_, 103
 - matrixExpr, 103
 - UnParse, 103
 - var_, 103

- fcsl::ast::nestedOrExpr, 104
 - expr_, 106
 - nestedOrExpr, 106
 - UnParse, 106
 - var_, 106
- fcsl::ast::notExpr, 109
 - expr_, 113
 - notExpr, 112
 - UnParse, 113
- fcsl::ast::parenthesisExpr, 117
 - expr_, 119
 - parenthesisExpr, 118
 - UnParse, 119
- fcsl::ast::printStmt, 152
 - expr_, 154
 - printStmt, 153
 - UnParse, 154
- fcsl::ast::repeatStmt, 158
 - expr1_, 160
 - expr2_, 160
 - repeatStmt, 160
 - stmt_, 160
 - UnParse, 160
 - var_, 161
- fcsl::ast::semiColonStmt, 170
 - semiColonStmt, 171
 - UnParse, 172
- fcsl::ast::seqStmts, 172
 - seqStmts, 174
 - stmt_, 175
 - stmts_, 175
 - UnParse, 175
- fcsl::ast::varDecl, 191
 - name_, 194
 - type_, 194
 - UnParse, 194
 - varDecl, 194
- fcsl::ast::varName, 198
 - lexeme_, 200
 - UnParse, 199
 - varName, 199
- fcsl::ast::whileStmt, 200
 - expr_, 202
 - stmt_, 202
 - UnParse, 202
 - whileStmt, 202
- fcsl::parser, 12
- fcsl::parser::ParseResult, 145
 - ast, 147
 - ast_, 148
 - errors, 147
 - errors_, 148
 - ok, 148
 - ok_, 148
 - ParseResult, 147
- fcsl::parser::Parser, 119
 - ~Parser, 122
 - attempt_match, 123
 - curr_token_, 145
 - make_error_msg, 123, 124
 - make_error_msg_expected, 124
 - match, 125
 - next_is, 126
 - next_token, 126
 - Parse, 127
 - parse_addition, 128
 - parse_char_const, 128
 - parse_decl, 129
 - parse_division, 129
 - parse_expr, 130
 - parse_false_kwd, 131
 - parse_float_const, 131
 - parse_if_expr, 132
 - parse_int_const, 133
 - parse_let_expr, 133
 - parse_matrix_decl, 134
 - parse_multiplication, 134
 - parse_nested_expr, 135
 - parse_not_expr, 136
 - parse_relational_expr, 136
 - parse_standard_decl, 137
 - parse_stmt, 138
 - parse_stmts, 139
 - parse_string_const, 140
 - parse_subtraction, 141
 - parse_true_kwd, 141
 - parse_variable_name, 142
 - ParseProgram, 143
 - Parser, 122
 - prev_token_, 145
 - scanner_, 145
 - tokens_, 145
 - terminal_description, 144
 - tokens_, 145
- fcsl::scanner, 13
 - kAndOp, 15
 - kAssign, 15
 - kBoolKwd, 14
 - kColon, 15
 - kDash, 15
 - kElseKwd, 14
 - kEndKwd, 14
 - kEndOfFile, 15
 - kEqualsEquals, 15
 - kFalseKwd, 14
 - kFloatConst, 15
 - kFloatKwd, 14
 - kForwardSlash, 15
 - kGreaterThan, 15
 - kGreaterThanEqual, 15
 - kIfKwd, 14
 - kInKwd, 14
 - kIntConst, 15
 - kIntKwd, 14
 - kLeftCurly, 15
 - kLeftParen, 15

- kLeftSquare, 15
- kLessThan, 15
- kLessThanEqual, 15
- kLetKwd, 14
- kLexicalError, 15
- kMatrixKwd, 14
- kNotEquals, 15
- kNotOp, 15
- kOrOp, 15
- kPlusSign, 15
- kPrintKwd, 15
- kRegexNSub, 17
- kRepeatKwd, 15
- kRightCurly, 15
- kRightParen, 15
- kRightSquare, 15
- kSemiColon, 15
- kStar, 15
- kStringConst, 15
- kStringKwd, 14
- kThenKwd, 14
- kToKwd, 15
- kTokenEnumType, 14
- kTrueKwd, 14
- kVariableName, 15
- kWhileKwd, 15
- make_regex, 15
- match_regex, 16
- ReadInput, 16
- ReadInputFromFile, 16
- TokenType, 14
- fcsl::scanner::CharConstToken, 27
 - CharConstToken, 30
 - description, 30
 - nud, 30
- fcsl::scanner::DashToken, 33
 - DashToken, 36
 - description, 36
 - lbp, 36
 - led, 36
- fcsl::scanner::EndOfFileToken, 44
 - description, 47
 - EndOfFileToken, 47
- fcsl::scanner::ExtToken, 53
 - ~ExtToken, 55
 - desc_str_, 62
 - description, 56
 - ExtToken, 55
 - ExtendToken, 56
 - ExtendTokenList, 57
 - lbp, 57
 - led, 58
 - lexeme, 59
 - lexeme_, 62
 - next, 59
 - next_, 62
 - nud, 60
 - parser, 60
- parser_, 62
- terminal, 61
- terminal_, 62
- fcsl::scanner::FalseKwdToken, 63
 - description, 65
 - FalseKwdToken, 65
 - nud, 65
- fcsl::scanner::FloatConstToken, 65
 - description, 68
 - FloatConstToken, 68
 - nud, 68
- fcsl::scanner::ForwardSlashToken, 68
 - description, 71
 - ForwardSlashToken, 71
 - lbp, 71
 - led, 71
- fcsl::scanner::IfToken, 81
 - description, 83
 - IfToken, 83
 - lbp, 83
 - nud, 83
- fcsl::scanner::IntConstToken, 83
 - description, 86
 - IntConstToken, 86
 - nud, 86
- fcsl::scanner::LeftParenToken, 86
 - description, 89
 - lbp, 89
 - LeftParenToken, 89
 - nud, 89
- fcsl::scanner::LetToken, 94
 - description, 96
 - lbp, 96
 - LetToken, 96
 - nud, 96
- fcsl::scanner::NotOpToken, 113
 - description, 116
 - NotOpToken, 116
 - nud, 116
- fcsl::scanner::PlusSignToken, 148
 - description, 151
 - lbp, 151
 - led, 151
 - PlusSignToken, 151
- fcsl::scanner::RelationalOpToken, 154
 - lbp, 157
 - led, 157
 - RelationalOpToken, 157
- fcsl::scanner::Scanner, 165
 - ~Scanner, 166
 - comments, 168
 - consume_whitespace_and_comments, 166
 - current_token, 168
 - line_comment, 169
 - previous_token, 169
 - regex_strings, 169
 - return_token, 169
 - Scan, 167

- Scanner, 166
 - text, 169
 - white_space, 169
- fcgal::scanner::StarToken, 176
 - description, 178
 - lbp, 178
 - led, 178
 - StarToken, 178
- fcgal::scanner::StringConstToken, 181
 - description, 184
 - nud, 184
 - StringConstToken, 184
- fcgal::scanner::Token, 184
 - ~Token, 186
 - lexeme, 186
 - lexeme_, 188
 - next, 186
 - next_, 188
 - set_next, 187
 - set_token, 187
 - terminal, 187
 - terminal_, 188
 - Token, 186
- fcgal::scanner::TrueKwdToken, 189
 - description, 191
 - nud, 191
 - TrueKwdToken, 191
- fcgal::scanner::VariableNameToken, 195
 - description, 197
 - nud, 197
 - VariableNameToken, 197
- float_
 - fcgal::ast, 12
- FloatConstToken
 - fcgal::scanner::FloatConstToken, 68
- ForwardSlashToken
 - fcgal::scanner::ForwardSlashToken, 71
- ifElseStmt
 - fcgal::ast::ifElseStmt, 74
- ifExpr
 - fcgal::ast::ifExpr, 77
- ifStmt
 - fcgal::ast::ifStmt, 80
- IfToken
 - fcgal::scanner::IfToken, 83
- include/ast.h, 203
- include/ext_token.h, 205
- include/mainpage.h, 207
- include/parse_result.h, 207
- include/parser.h, 208
- include/read_input.h, 210
- include/regex.h, 210
- include/scanner.h, 211
- include/scanner_class.h, 213
- include/token_class.h, 214
- int_
 - fcgal::ast, 12
- IntConstToken
 - fcgal::scanner::IntConstToken, 86
- isMatrix_
 - fcgal::ast::equalsStmt, 51
- kAndOp
 - fcgal::scanner, 15
- kAssign
 - fcgal::scanner, 15
- kBoolKwd
 - fcgal::scanner, 14
- kColon
 - fcgal::scanner, 15
- kDash
 - fcgal::scanner, 15
- kElseKwd
 - fcgal::scanner, 14
- kEndKwd
 - fcgal::scanner, 14
- kEndOfFile
 - fcgal::scanner, 15
- kEqualsEquals
 - fcgal::scanner, 15
- kFalseKwd
 - fcgal::scanner, 14
- kFloatConst
 - fcgal::scanner, 15
- kFloatKwd
 - fcgal::scanner, 14
- kForwardSlash
 - fcgal::scanner, 15
- kGreaterThan
 - fcgal::scanner, 15
- kGreaterThanEqual
 - fcgal::scanner, 15
- kIfKwd
 - fcgal::scanner, 14
- kInKwd
 - fcgal::scanner, 14
- kIntConst
 - fcgal::scanner, 15
- kIntKwd
 - fcgal::scanner, 14
- kLeftCurly
 - fcgal::scanner, 15
- kLeftParen
 - fcgal::scanner, 15
- kLeftSquare
 - fcgal::scanner, 15
- kLessThan
 - fcgal::scanner, 15
- kLessThanEqual
 - fcgal::scanner, 15
- kLetKwd
 - fcgal::scanner, 14
- kLexicalError
 - fcgal::scanner, 15
- kMatrixKwd
 - fcgal::scanner, 14
- kNotEquals

- fcgal::scanner, 15
- kNotOp
 - fcgal::scanner, 15
- kOrOp
 - fcgal::scanner, 15
- kPlusSign
 - fcgal::scanner, 15
- kPrintKwd
 - fcgal::scanner, 15
- kRegexNSub
 - fcgal::scanner, 17
- kRepeatKwd
 - fcgal::scanner, 15
- kRightCurly
 - fcgal::scanner, 15
- kRightParen
 - fcgal::scanner, 15
- kRightSquare
 - fcgal::scanner, 15
- kSemiColon
 - fcgal::scanner, 15
- kStar
 - fcgal::scanner, 15
- kStringConst
 - fcgal::scanner, 15
- kStringKwd
 - fcgal::scanner, 14
- kThenKwd
 - fcgal::scanner, 14
- kToKwd
 - fcgal::scanner, 15
- kTokenEnumType
 - fcgal::scanner, 14
- kTrueKwd
 - fcgal::scanner, 14
- kVariableName
 - fcgal::scanner, 15
- kWhileKwd
 - fcgal::scanner, 15
- lbp
 - fcgal::scanner::DashToken, 36
 - fcgal::scanner::ExtToken, 57
 - fcgal::scanner::ForwardSlashToken, 71
 - fcgal::scanner::IfToken, 83
 - fcgal::scanner::LeftParenToken, 89
 - fcgal::scanner::LetToken, 96
 - fcgal::scanner::PlusSignToken, 151
 - fcgal::scanner::RelationalOpToken, 157
 - fcgal::scanner::StarToken, 178
- led
 - fcgal::scanner::DashToken, 36
 - fcgal::scanner::ExtToken, 58
 - fcgal::scanner::ForwardSlashToken, 71
 - fcgal::scanner::PlusSignToken, 151
 - fcgal::scanner::RelationalOpToken, 157
 - fcgal::scanner::StarToken, 178
- LeftParenToken
 - fcgal::scanner::LeftParenToken, 89
- letExpr
 - fcgal::ast::letExpr, 92
- LetToken
 - fcgal::scanner::LetToken, 96
- lexeme
 - fcgal::scanner::ExtToken, 59
 - fcgal::scanner::Token, 186
- lexeme_
 - fcgal::ast::varName, 200
 - fcgal::scanner::ExtToken, 62
 - fcgal::scanner::Token, 188
- line_comment
 - fcgal::scanner::Scanner, 169
- make_error_msg
 - fcgal::parser::Parser, 123, 124
- make_error_msg_expected
 - fcgal::parser::Parser, 124
- make_regex
 - fcgal::scanner, 15
- match
 - fcgal::parser::Parser, 125
- match_regex
 - fcgal::scanner, 16
- matrixDecl
 - fcgal::ast::matrixDecl, 99
- matrixExpr
 - fcgal::ast::matrixExpr, 103
- name_
 - fcgal::ast::Root, 164
 - fcgal::ast::varDecl, 194
- nestedOrExpr
 - fcgal::ast::nestedOrExpr, 106
- next
 - fcgal::scanner::ExtToken, 59
 - fcgal::scanner::Token, 186
- next_
 - fcgal::scanner::ExtToken, 62
 - fcgal::scanner::Token, 188
- next_is
 - fcgal::parser::Parser, 126
- next_token
 - fcgal::parser::Parser, 126
- notExpr
 - fcgal::ast::notExpr, 112
- NotOpToken
 - fcgal::scanner::NotOpToken, 116
- nud
 - fcgal::scanner::CharConstToken, 30
 - fcgal::scanner::ExtToken, 60
 - fcgal::scanner::FalseKwdToken, 65
 - fcgal::scanner::FloatConstToken, 68
 - fcgal::scanner::IfToken, 83
 - fcgal::scanner::IntConstToken, 86
 - fcgal::scanner::LeftParenToken, 89
 - fcgal::scanner::LetToken, 96
 - fcgal::scanner::NotOpToken, 116
 - fcgal::scanner::StringConstToken, 184

- fcgal::scanner::TrueKwdToken, 191
 - fcgal::scanner::VariableNameToken, 197
- ok
 - fcgal::parser::ParseResult, 148
- ok_
 - fcgal::parser::ParseResult, 148
- parenthesisExpr
 - fcgal::ast::parenthesisExpr, 118
- Parse
 - fcgal::parser::Parser, 127
- parse_addition
 - fcgal::parser::Parser, 128
- parse_char_const
 - fcgal::parser::Parser, 128
- parse_decl
 - fcgal::parser::Parser, 129
- parse_division
 - fcgal::parser::Parser, 129
- parse_expr
 - fcgal::parser::Parser, 130
- parse_false_kwd
 - fcgal::parser::Parser, 131
- parse_float_const
 - fcgal::parser::Parser, 131
- parse_if_expr
 - fcgal::parser::Parser, 132
- parse_int_const
 - fcgal::parser::Parser, 133
- parse_let_expr
 - fcgal::parser::Parser, 133
- parse_matrix_decl
 - fcgal::parser::Parser, 134
- parse_multiplication
 - fcgal::parser::Parser, 134
- parse_nested_expr
 - fcgal::parser::Parser, 135
- parse_not_expr
 - fcgal::parser::Parser, 136
- parse_relational_expr
 - fcgal::parser::Parser, 136
- parse_standard_decl
 - fcgal::parser::Parser, 137
- parse_stmt
 - fcgal::parser::Parser, 138
- parse_stmts
 - fcgal::parser::Parser, 139
- parse_string_const
 - fcgal::parser::Parser, 140
- parse_subtraction
 - fcgal::parser::Parser, 141
- parse_true_kwd
 - fcgal::parser::Parser, 141
- parse_variable_name
 - fcgal::parser::Parser, 142
- ParseProgram
 - fcgal::parser::Parser, 143
- ParseResult
 - fcgal::parser::ParseResult, 147
- Parser
 - fcgal::parser::Parser, 122
- parser
 - fcgal::scanner::ExtToken, 60
- parser_
 - fcgal::scanner::ExtToken, 62
- PlusSignToken
 - fcgal::scanner::PlusSignToken, 151
- prev_token_
 - fcgal::parser::Parser, 145
- previous_token
 - fcgal::scanner::Scanner, 169
- printStmt
 - fcgal::ast::printStmt, 153
- ReadInput
 - fcgal::scanner, 16
- ReadInputFromFile
 - fcgal::scanner, 16
- regex_strings
 - fcgal::scanner::Scanner, 169
- RelationalOpToken
 - fcgal::scanner::RelationalOpToken, 157
- repeatStmt
 - fcgal::ast::repeatStmt, 160
- return_token
 - fcgal::scanner::Scanner, 169
- Root
 - fcgal::ast::Root, 163
- Scan
 - fcgal::scanner::Scanner, 167
- Scanner
 - fcgal::scanner::Scanner, 166
- scanner_
 - fcgal::parser::Parser, 145
- semiColonStmt
 - fcgal::ast::semiColonStmt, 171
- seqStmts
 - fcgal::ast::seqStmts, 174
- set_next
 - fcgal::scanner::Token, 187
- set_token
 - fcgal::scanner::Token, 187
- simpleMatrix_
 - fcgal::ast::matrixDecl, 100
- src/ast.cc, 215
- src/ext_token.cc, 215
- src/parser.cc, 216
- src/read_input.cc, 217
- src/regex.cc, 218
- src/scanner_class.cc, 218
- src/token_class.cc, 219
- StarToken
 - fcgal::scanner::StarToken, 178
- stmt1_
 - fcgal::ast::ifElseStmt, 74
- stmt2_

- fc`al::ast::ifElseStmt`, 74
- stmt_
 - fc`al::ast::ifStmt`, 80
 - fc`al::ast::repeatStmt`, 160
 - fc`al::ast::seqStmts`, 175
 - fc`al::ast::whileStmt`, 202
- stmts_
 - fc`al::ast::Root`, 164
 - fc`al::ast::bracketStmt`, 27
 - fc`al::ast::letExpr`, 93
 - fc`al::ast::seqStmts`, 175
- tokens_
 - fc`al::parser::Parser`, 145
- string_
 - fc`al::ast`, 12
- StringConstToken
 - fc`al::scanner::StringConstToken`, 184
- terminal
 - fc`al::scanner::ExtToken`, 61
 - fc`al::scanner::Token`, 187
- terminal_
 - fc`al::scanner::ExtToken`, 62
 - fc`al::scanner::Token`, 188
- terminal_description
 - fc`al::parser::Parser`, 144
- text
 - fc`al::scanner::Scanner`, 169
- Token
 - fc`al::scanner::Token`, 186
- TokenType
 - fc`al::scanner`, 14
- tokens_
 - fc`al::parser::Parser`, 145
- TrueKwdToken
 - fc`al::scanner::TrueKwdToken`, 191
- type_
 - fc`al::ast::varDecl`, 194
- UnParse
 - fc`al::ast::Node`, 108
 - fc`al::ast::Root`, 164
 - fc`al::ast::binaryExpr`, 21
 - fc`al::ast::boolExpr`, 24
 - fc`al::ast::bracketStmt`, 27
 - fc`al::ast::constantExpr`, 33
 - fc`al::ast::declStmt`, 41
 - fc`al::ast::emptyStmts`, 44
 - fc`al::ast::equalsStmt`, 50
 - fc`al::ast::ifElseStmt`, 74
 - fc`al::ast::ifExpr`, 77
 - fc`al::ast::ifStmt`, 80
 - fc`al::ast::letExpr`, 93
 - fc`al::ast::matrixDecl`, 100
 - fc`al::ast::matrixExpr`, 103
 - fc`al::ast::nestedOrExpr`, 106
 - fc`al::ast::notExpr`, 113
 - fc`al::ast::parenthesisExpr`, 119
 - fc`al::ast::printStmt`, 154
 - fc`al::ast::repeatStmt`, 160
 - fc`al::ast::semiColonStmt`, 172
 - fc`al::ast::seqStmts`, 175
 - fc`al::ast::varDecl`, 194
 - fc`al::ast::varName`, 199
 - fc`al::ast::whileStmt`, 202
- var1_
 - fc`al::ast::matrixDecl`, 100
- var2_
 - fc`al::ast::matrixDecl`, 101
- var3_
 - fc`al::ast::matrixDecl`, 101
- var_
 - fc`al::ast::equalsStmt`, 51
 - fc`al::ast::matrixExpr`, 103
 - fc`al::ast::nestedOrExpr`, 106
 - fc`al::ast::repeatStmt`, 161
- varDecl
 - fc`al::ast::varDecl`, 194
- varName
 - fc`al::ast::varName`, 199
- VariableNameToken
 - fc`al::scanner::VariableNameToken`, 197
- whileStmt
 - fc`al::ast::whileStmt`, 202
- white_space
 - fc`al::scanner::Scanner`, 169