

## CPSC 4660 Compiler

Generated by Doxygen 1.8.5

Sat Feb 1 2020 19:36:57



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	Administration Class Reference . . . . .	5
3.1.1	Constructor & Destructor Documentation . . . . .	6
3.1.1.1	Administration . . . . .	6
3.1.2	Member Function Documentation . . . . .	7
3.1.2.1	checkError . . . . .	7
3.1.2.2	error . . . . .	7
3.1.2.3	getToken . . . . .	7
3.1.2.4	newLine . . . . .	7
3.1.2.5	scan . . . . .	7
3.1.2.6	syntaxError . . . . .	7
3.1.3	Member Data Documentation . . . . .	7
3.1.3.1	correctLine . . . . .	7
3.1.3.2	errorCount . . . . .	8
3.1.3.3	fout . . . . .	8
3.1.3.4	lineNum . . . . .	8
3.1.3.5	scanner . . . . .	8
3.2	Parser Class Reference . . . . .	8
3.2.1	Constructor & Destructor Documentation . . . . .	9
3.2.1.1	Parser . . . . .	9
3.2.2	Member Function Documentation . . . . .	9
3.2.2.1	addOp . . . . .	9
3.2.2.2	block . . . . .	9
3.2.2.3	boolSym . . . . .	9
3.2.2.4	exprList . . . . .	9
3.2.2.5	factor . . . . .	9

3.2.2.6	match	9
3.2.2.7	multOp	9
3.2.2.8	parse	9
3.2.2.9	program	9
3.2.2.10	simpleExpr	9
3.2.2.11	term	9
3.2.3	Member Data Documentation	9
3.2.3.1	admin	9
3.2.3.2	look	9
3.3	Scanner Class Reference	10
3.3.1	Constructor & Destructor Documentation	10
3.3.1.1	Scanner	10
3.3.1.2	~Scanner	11
3.3.2	Member Function Documentation	11
3.3.2.1	getToken	11
3.3.2.2	isSpecial	11
3.3.2.3	isWhitespace	11
3.3.2.4	recognizeName	11
3.3.2.5	recognizeNumeral	11
3.3.2.6	recognizeSpecial	11
3.3.3	Member Data Documentation	11
3.3.3.1	fin	11
3.3.3.2	line	11
3.3.3.3	pos	12
3.3.3.4	symmap	12
3.3.3.5	symtable	12
3.4	SymbolTable Class Reference	12
3.4.1	Constructor & Destructor Documentation	13
3.4.1.1	SymbolTable	13
3.4.2	Member Function Documentation	13
3.4.2.1	full	13
3.4.2.2	getLoad	13
3.4.2.3	hash	13
3.4.2.4	insert	13
3.4.2.5	loadKeywords	13
3.4.2.6	probe	13
3.4.2.7	search	13
3.4.2.8	toString	13
3.4.3	Member Data Documentation	13
3.4.3.1	keywords	13

3.4.3.2	load	14
3.4.3.3	table	14
3.5	Token Class Reference	14
3.5.1	Constructor & Destructor Documentation	14
3.5.1.1	Token	14
3.5.1.2	Token	14
3.5.2	Member Function Documentation	14
3.5.2.1	getLexeme	14
3.5.2.2	getSymbol	15
3.5.2.3	getVal	15
3.5.2.4	setLexeme	15
3.5.2.5	setSymbol	15
3.5.2.6	setVal	15
3.5.2.7	toString	15
3.5.3	Member Data Documentation	15
3.5.3.1	lexeme	15
3.5.3.2	sname	15
3.5.3.3	val	15
<b>4</b>	<b>File Documentation</b>	<b>17</b>
4.1	Administration.h File Reference	17
4.1.1	Variable Documentation	17
4.1.1.1	MAX_ERRORS	17
4.2	Parser.h File Reference	17
4.3	Scanner.h File Reference	17
4.4	Symbol.h File Reference	18
4.4.1	Enumeration Type Documentation	18
4.4.1.1	Symbol	18
4.4.2	Variable Documentation	19
4.4.2.1	SymbolToString	19
4.5	SymbolTable.h File Reference	19
4.5.1	Variable Documentation	20
4.5.1.1	ID_MAX_CHARS	20
4.5.1.2	MOD	20
4.5.1.3	PRIME	20
4.6	Token.h File Reference	20
	<b>Index</b>	<b>21</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Administration</a>	5
<a href="#">Parser</a>	8
<a href="#">Scanner</a>	10
<a href="#">SymbolTable</a>	12
<a href="#">Token</a>	14





## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Administration.h</a>	17
<a href="#">Parser.h</a>	17
<a href="#">Scanner.h</a>	17
<a href="#">Symbol.h</a>	18
<a href="#">SymbolTable.h</a>	19
<a href="#">Token.h</a>	20



## Chapter 3

# Class Documentation

### 3.1 Administration Class Reference

```
#include <Administration.h>
```

#### Public Member Functions

- [Administration](#) (std::ostream &fout, [Scanner](#) &sc)  
*Creates a new [Administration](#) object.*
- [Token](#) [getToken](#) ()
- void [newLine](#) ()  
*Adds line number and resets correctLine.*
- void [syntaxError](#) ([Symbol](#) expected, [Symbol](#) actual)  
*Deals with a syntax error.*
- void [error](#) (std::string text)  
*Display text for an error.*
- int [scan](#) ()  
*Scan the whole file and output all tokens to fout.*
- void [checkError](#) ([Token](#) ntoken)  
*Checks if current token is an error token.*

#### Public Attributes

- std::ostream & [fout](#)  
*File to print all tokens to.*
- [Scanner](#) & [scanner](#)  
*The scanner to use on the input.*
- int [lineNum](#)  
*The current line number.*
- bool [correctLine](#)  
*True if the line has no errors so far.*
- int [errorCount](#)  
*The total number of errors so far.*

### 3.1.1 Constructor & Destructor Documentation

#### 3.1.1.1 Administration::Administration ( std::ostream & *fout*, Scanner & *sc* )

Creates a new [Administration](#) object.

## Parameters

<i>fout</i>	The output file stream.
<i>sc</i>	The scanner beign used by administration.

### 3.1.2 Member Function Documentation

#### 3.1.2.1 void Administration::checkError ( Token *ntoken* )

Checks if current token is an error token.

## Parameters

<i>ntoken</i>	The current token.
---------------	--------------------

#### 3.1.2.2 void Administration::error ( std::string *text* )

Display text for an error.

## Parameters

<i>text</i>	The error message.
-------------	--------------------

#### 3.1.2.3 Token Administration::getToken ( )

#### 3.1.2.4 void Administration::newLine ( )

Adds line number and resets correctLine.

#### 3.1.2.5 int Administration::scan ( )

Scan the whole file and output all tokens to fout.

Returns the number of tokens.

#### 3.1.2.6 void Administration::syntaxError ( Symbol *expected*, Symbol *actual* )

Deals with a syntax error.

*expected* The token type we expected to see.

## Parameters

<i>actual</i>	The token type we got.
---------------	------------------------

## Returns

false when the max number of errors is reached.

### 3.1.3 Member Data Documentation

#### 3.1.3.1 bool Administration::correctLine

True if the line has no errors so far.

### 3.1.3.2 int Administration::errorCount

The total number of errors so far.

### 3.1.3.3 std::ostream& Administration::fout

File to print all tokens to.

### 3.1.3.4 int Administration::lineNum

The current line number.

### 3.1.3.5 Scanner& Administration::scanner

The scanner to use on the input.

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

- [Administration.h](#)

## 3.2 Parser Class Reference

```
#include <Parser.h>
```

### Public Member Functions

- [Parser](#) ([Administration](#) &[admin](#))  
*Creates a new [Parser](#) object.*
- void [parse](#) ()  
*Parses a PL program.*

### Private Member Functions

- void [match](#) ([Symbol](#) symbol)  
*Match a [Token](#) and move to the next one.*
- void [program](#) ()
- void [block](#) ()
- void [exprList](#) ()
- void [simpleExpr](#) ()
- void [term](#) ()
- void [factor](#) ()
- void [addOp](#) ()
- void [multOp](#) ()
- void [boolSym](#) ()

### Private Attributes

- [Administration](#) & [admin](#)  
*The administration object for errors and holding the scanner and symbol table.*
- [Token](#) [look](#)  
*The look ahead token.*

### 3.2.1 Constructor & Destructor Documentation

#### 3.2.1.1 Parser::Parser ( Administration & *admin* )

Creates a new [Parser](#) object.

Parameters

<i>admin</i>	An administration object for handling errors and holding our scanner etc. for now.
--------------	--

### 3.2.2 Member Function Documentation

#### 3.2.2.1 void Parser::addOp ( ) [private]

#### 3.2.2.2 void Parser::block ( ) [private]

#### 3.2.2.3 void Parser::boolSym ( ) [private]

#### 3.2.2.4 void Parser::exprList ( ) [private]

#### 3.2.2.5 void Parser::factor ( ) [private]

#### 3.2.2.6 void Parser::match ( Symbol *symbol* ) [private]

Match a [Token](#) and move to the next one.

#### 3.2.2.7 void Parser::multOp ( ) [private]

#### 3.2.2.8 void Parser::parse ( )

Parses a PL program.

#### 3.2.2.9 void Parser::program ( ) [private]

#### 3.2.2.10 void Parser::simpleExpr ( ) [private]

#### 3.2.2.11 void Parser::term ( ) [private]

### 3.2.3 Member Data Documentation

#### 3.2.3.1 Administration& Parser::admin [private]

The administration object for errors and holding the scanner and symbol table.

#### 3.2.3.2 Token Parser::look [private]

The look ahead token.

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

- [Parser.h](#)

### 3.3 Scanner Class Reference

```
#include <Scanner.h>
```

#### Public Member Functions

- [Scanner](#) (std::istream &ifs, [SymbolTable](#) &symboltable)  
*Constructor for the scanner, initializes the private variables to appropriate values.*
- [~Scanner](#) ()  
*Destructor of the scanner.*
- [Token getToken](#) ()  
*Get the next [Token](#) in the line.*

#### Private Member Functions

- bool [isWhitespace](#) (char inchar)  
*Check input symbol against Whitespace whether tab or space.*
- bool [isSpecial](#) (char inchar)  
*Checks the inputted char against all possible symbols.*
- [Token recognizeName](#) ()  
*Read and generate tokens for keywords and ID's, also checks for invalid characters and returns a `CHAR_ERR` token and checks the symbol table is filled then return a `FULL_TAB` error token.*
- [Token recognizeSpecial](#) ()  
*Read and generate a token for any of the special symbols.*
- [Token recognizeNumeral](#) ()  
*Read and generate a token for any number/digit.*

#### Private Attributes

- std::istream & [fin](#)  
*The file stream.*
- [SymbolTable](#) & [symtable](#)  
*The Symbol Table being checked and filled with tokens.*
- std::string [line](#)  
*The current line the scanner is reading.*
- std::size\_t [pos](#)  
*The position of the char the scanner is reading.*
- std::map< std::string, [Symbol](#) > [symmap](#)  
*The map containing the symbols.*

#### 3.3.1 Constructor & Destructor Documentation

##### 3.3.1.1 Scanner::Scanner ( std::istream & ifs, SymbolTable & symboltable )

Constructor for the scanner, initializes the private variables to appropriate values.

##### Parameters

---



<i>ifs</i>	The file stream.
<i>symboltable</i>	The Symbol Table used throughout the scan being updated.

### 3.3.1.2 Scanner::~Scanner ( ) [inline]

Destructor of the scanner.

## 3.3.2 Member Function Documentation

### 3.3.2.1 Token Scanner::getToken ( )

Get the next [Token](#) in the line.

### 3.3.2.2 bool Scanner::isSpecial ( char *inchar* ) [private]

Checks the inputted char against all possible symbols.

Parameters

<i>inchar</i>	The current char being read in
---------------	--------------------------------

### 3.3.2.3 bool Scanner::isWhitespace ( char *inchar* ) [private]

Check input symbol against Whitespace whether tab or space.

Parameters

<i>inchar</i>	The current char being read in
---------------	--------------------------------

### 3.3.2.4 Token Scanner::recognizeName ( ) [private]

Read and generate tokens for keywords and ID's, also checks for invalid characters and returns a CHAR\_ERR token and checks the symbol table is filled then return a FULL\_TAB error token.

### 3.3.2.5 Token Scanner::recognizeNumeral ( ) [private]

Read and generate a token for any number/digit.

### 3.3.2.6 Token Scanner::recognizeSpecial ( ) [private]

Read and generate a token for any of the special symbols.

## 3.3.3 Member Data Documentation

### 3.3.3.1 std::istream& Scanner::fin [private]

The file stream.

### 3.3.3.2 std::string Scanner::line [private]

The current line the scanner is reading.

### 3.3.3.3 `std::size_t Scanner::pos` [private]

The position of the char the scanner is reading.

### 3.3.3.4 `std::map<std::string, Symbol> Scanner::symmap` [private]

The map containing the symbols.

### 3.3.3.5 `SymbolTable& Scanner::symtable` [private]

The Symbol Table being checked and filled with tokens.

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

- [Scanner.h](#)

## 3.4 SymbolTable Class Reference

```
#include <SymbolTable.h>
```

### Public Member Functions

- [SymbolTable](#) ()
- [Token search](#) (const std::string &str)  
*Searches for a lexeme in the symbol table and returns its position.*
- [Token insert](#) (const std::string &str)  
*Insert a new lexeme into the symbol table.*
- int [hash](#) (const std::string &str)  
*Computes a rolling hash for a given string using the MOD constant.*
- bool [full](#) ()  
*Returns true if the table is full.*
- int [getLoad](#) ()  
*Returns the number items in the table.*
- std::string [toString](#) ()  
*Returns a string representation of the table.*

### Private Member Functions

- std::pair< int, [Token](#) > [probe](#) (int idx, std::string lexeme)  
*Given a position linear probe until the token with the given lexeme is found or an empty token is found.*
- void [loadKeywords](#) ()  
*Loads all reserved keywords into the symbol table.*

### Private Attributes

- std::vector< [Token](#) > [table](#)
- int [load](#)
- const std::vector< std::string > [keywords](#)

### 3.4.1 Constructor & Destructor Documentation

#### 3.4.1.1 SymbolTable::SymbolTable ( )

### 3.4.2 Member Function Documentation

#### 3.4.2.1 bool SymbolTable::full ( )

Returns true if the table is full.

#### 3.4.2.2 int SymbolTable::getLoad ( )

Returns the number items in the table.

#### 3.4.2.3 int SymbolTable::hash ( const std::string & *str* )

Computes a rolling hash for a given string using the MOD constant.

Only looks at a max of 10 characters from the string. Returns the integer hash of the string.

#### 3.4.2.4 Token SymbolTable::insert ( const std::string & *str* )

Insert a new lexeme into the symbol table.

Creates a new ID token for the lexeme as once the reserve words are loaded the only thing loaded should be IDs. Returns the ERROR token if the table is full.

#### 3.4.2.5 void SymbolTable::loadKeywords ( ) [private]

Loads all reserved keywords into the symbol table.

#### 3.4.2.6 std::pair<int, Token> SymbolTable::probe ( int *idx*, std::string *lexeme* ) [private]

Given a position linear probe until the token with the given lexeme is found or an empty token is found.

Returns a pair with the position of the token and the lexeme.

#### 3.4.2.7 Token SymbolTable::search ( const std::string & *str* )

Searches for a lexeme in the symbol table and returns its position.

Returns the EMPTY token if the table is full.

#### 3.4.2.8 std::string SymbolTable::toString ( )

Returns a string representation of the table.

### 3.4.3 Member Data Documentation

#### 3.4.3.1 const std::vector<std::string> SymbolTable::keywords [private]

**Initial value:**

```
{
    "const", "array", "proc", "skip", "read", "write",
    "call", "if", "fi", "do", "od", "integer", "Boolean", "true", "false"
}
```

3.4.3.2 `int SymbolTable::load` `[private]`

3.4.3.3 `std::vector<Token> SymbolTable::table` `[private]`

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

- [SymbolTable.h](#)

## 3.5 Token Class Reference

```
#include <Token.h>
```

### Public Member Functions

- [Token](#) ()
- [Token](#) ([Symbol](#) sym, `std::string lexeme=""`, `int val=-1`)
- [Symbol](#) [getSymbol](#) ()  
*Returns the symbol.*
- `std::string` [getLexeme](#) ()  
*Returns the lexeme.*
- `int` [getVal](#) ()  
*Returns the value.*
- `void` [setSymbol](#) ([Symbol](#) sym)  
*Sets the symbol.*
- `void` [setLexeme](#) (`std::string lexeme`)  
*Sets the lexeme.*
- `void` [setVal](#) (`int val`)  
*Sets the value.*
- `void` [toString](#) (`std::ostream &out`) `const`  
*returns a string representation of the [Token](#).*

### Private Attributes

- [Symbol](#) `sname`
- `std::string` `lexeme`
- `int` `val`

### 3.5.1 Constructor & Destructor Documentation

3.5.1.1 `Token::Token ( )`

3.5.1.2 `Token::Token ( Symbol sym, std::string lexeme = "", int val = -1 )`

### 3.5.2 Member Function Documentation

3.5.2.1 `std::string` `Token::getLexeme ( )`

Returns the lexeme.

### 3.5.2.2 Symbol Token::getSymbol ( )

Returns the symbol.

### 3.5.2.3 int Token::getVal ( )

Returns the value.

### 3.5.2.4 void Token::setLexeme ( std::string *lexeme* )

Sets the lexeme.

### 3.5.2.5 void Token::setSymbol ( Symbol *sym* )

Sets the symbol.

### 3.5.2.6 void Token::setVal ( int *val* )

Sets the value.

### 3.5.2.7 void Token::toString ( std::ostream & *out* ) const

returns a string representation of the [Token](#).

## 3.5.3 Member Data Documentation

### 3.5.3.1 std::string Token::lexeme [private]

### 3.5.3.2 Symbol Token::sname [private]

### 3.5.3.3 int Token::val [private]

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

- [Token.h](#)



## Chapter 4

# File Documentation

### 4.1 Administration.h File Reference

```
#include <iostream>
#include "Token.h"
#include "Scanner.h"
```

#### Classes

- class [Administration](#)

#### Variables

- const int [MAX\\_ERRORS](#) = 10

#### 4.1.1 Variable Documentation

4.1.1.1 const int [MAX\\_ERRORS](#) = 10

### 4.2 Parser.h File Reference

```
#include <iostream>
#include "Token.h"
#include "Administration.h"
```

#### Classes

- class [Parser](#)

### 4.3 Scanner.h File Reference

```
#include "SymbolTable.h"
#include "Token.h"
#include <map>
#include <iostream>
```

## Classes

- class [Scanner](#)

## 4.4 Symbol.h File Reference

```
#include <map>
```

## Enumerations

- enum [Symbol](#) {  
    [DOT](#) = 256, [COMMA](#), [SEMI](#), [LHSQR](#),  
    [RHSQR](#), [AMP](#), [BAR](#), [TILD](#),  
    [LESS](#), [EQUAL](#), [GREAT](#), [PLUS](#),  
    [MINUS](#), [TIMES](#), [FSLASH](#), [BSLASH](#),  
    [LHRND](#), [RHRND](#), [INIT](#), [GUARD](#),  
    [ARROW](#), [DOLLAR](#), [INT](#), [BOOL](#),  
    [FALSE](#), [TRUE](#), [BEGIN](#), [END](#),  
    [ID](#), [KEY](#), [ENDFILE](#), [EMPTY](#),  
    [NEWLINE](#), [NUM](#), [NAME\\_ERR](#), [NUM\\_ERR](#),  
    [CHAR\\_ERR](#), [FULL\\_TAB](#) }

*Enum containing all possible Symbols.*

## Variables

- const std::map< [Symbol](#),  
    std::string > [SymbolToString](#)

*Map mapping all the symbols to string versions of themselves for printing.*

### 4.4.1 Enumeration Type Documentation

#### 4.4.1.1 enum [Symbol](#)

Enum containing all possible Symbols.

#### Enumerator

***[DOT](#)***  
***[COMMA](#)***  
***[SEMI](#)***  
***[LHSQR](#)***  
***[RHSQR](#)***  
***[AMP](#)***  
***[BAR](#)***  
***[TILD](#)***  
***[LESS](#)***  
***[EQUAL](#)***



***GREAT***  
***PLUS***  
***MINUS***  
***TIMES***  
***FSLASH***  
***BSLASH***  
***LHRND***  
***RHRND***  
***INIT***  
***GUARD***  
***ARROW***  
***DOLLAR***  
***INT***  
***BOOL***  
***FALSE***  
***TRUE***  
***BEGIN***  
***END***  
***ID***  
***KEY***  
***ENDFILE***  
***EMPTY***  
***NEWLINE***  
***NUM***  
***NAME\_ERR***  
***NUM\_ERR***  
***CHAR\_ERR***  
***FULL\_TAB***

#### 4.4.2 Variable Documentation

##### 4.4.2.1 `const std::map<Symbol, std::string> SymbolToString`

Map mapping all the symbols to string versions of themselves for printing.

## 4.5 SymbolTable.h File Reference

```
#include "Token.h"  
#include <vector>  
#include <string>
```

### Classes

- class [SymbolTable](#)

## Variables

- const int MOD = 307
- const int PRIME = 67
- const int ID\_MAX\_CHARS = 10

### 4.5.1 Variable Documentation

4.5.1.1 const int ID\_MAX\_CHARS = 10

4.5.1.2 const int MOD = 307

4.5.1.3 const int PRIME = 67

## 4.6 Token.h File Reference

```
#include "Symbol.h"  
#include <iostream>  
#include <string>
```

## Classes

- class Token

# Index

- ~Scanner
  - Scanner, [11](#)
- AMP
  - Symbol.h, [18](#)
- ARROW
  - Symbol.h, [19](#)
- addOp
  - Parser, [9](#)
- admin
  - Parser, [9](#)
- Administration, [5](#)
  - Administration, [6](#)
  - checkError, [7](#)
  - correctLine, [7](#)
  - error, [7](#)
  - errorCount, [7](#)
  - fout, [8](#)
  - getToken, [7](#)
  - lineNum, [8](#)
  - newLine, [7](#)
  - scan, [7](#)
  - scanner, [8](#)
  - syntaxError, [7](#)
- Administration.h, [17](#)
  - MAX\_ERRORS, [17](#)
- BAR
  - Symbol.h, [18](#)
- BEGIN
  - Symbol.h, [19](#)
- BOOL
  - Symbol.h, [19](#)
- BSLASH
  - Symbol.h, [19](#)
- block
  - Parser, [9](#)
- boolSym
  - Parser, [9](#)
- CHAR\_ERR
  - Symbol.h, [19](#)
- COMMA
  - Symbol.h, [18](#)
- checkError
  - Administration, [7](#)
- correctLine
  - Administration, [7](#)
- DOLLAR
  - Symbol.h, [19](#)
- DOT
  - Symbol.h, [18](#)
- EMPTY
  - Symbol.h, [19](#)
- END
  - Symbol.h, [19](#)
- ENDFILE
  - Symbol.h, [19](#)
- EQUAL
  - Symbol.h, [18](#)
- error
  - Administration, [7](#)
- errorCount
  - Administration, [7](#)
- exprList
  - Parser, [9](#)
- FALSE
  - Symbol.h, [19](#)
- FSLASH
  - Symbol.h, [19](#)
- FULL\_TAB
  - Symbol.h, [19](#)
- factor
  - Parser, [9](#)
- fin
  - Scanner, [11](#)
- fout
  - Administration, [8](#)
- full
  - SymbolTable, [13](#)
- GREAT
  - Symbol.h, [18](#)
- GUARD
  - Symbol.h, [19](#)
- getLexeme
  - Token, [14](#)
- getLoad
  - SymbolTable, [13](#)
- getSymbol
  - Token, [14](#)
- getToken
  - Administration, [7](#)
  - Scanner, [11](#)
- getVal
  - Token, [15](#)
- hash

- SymbolTable, [13](#)
- ID
  - Symbol.h, [19](#)
- INIT
  - Symbol.h, [19](#)
- INT
  - Symbol.h, [19](#)
- ID\_MAX\_CHARS
  - SymbolTable.h, [20](#)
- insert
  - SymbolTable, [13](#)
- isSpecial
  - Scanner, [11](#)
- isWhitespace
  - Scanner, [11](#)
- KEY
  - Symbol.h, [19](#)
- keywords
  - SymbolTable, [13](#)
- LESS
  - Symbol.h, [18](#)
- LHRND
  - Symbol.h, [19](#)
- LHSQR
  - Symbol.h, [18](#)
- lexeme
  - Token, [15](#)
- line
  - Scanner, [11](#)
- lineNum
  - Administration, [8](#)
- load
  - SymbolTable, [14](#)
- loadKeywords
  - SymbolTable, [13](#)
- look
  - Parser, [9](#)
- MINUS
  - Symbol.h, [19](#)
- MAX\_ERRORS
  - Administration.h, [17](#)
- MOD
  - SymbolTable.h, [20](#)
- match
  - Parser, [9](#)
- multOp
  - Parser, [9](#)
- NAME\_ERR
  - Symbol.h, [19](#)
- NEWLINE
  - Symbol.h, [19](#)
- NUM
  - Symbol.h, [19](#)
- NUM\_ERR
  - Symbol.h, [19](#)
- newLine
  - Administration, [7](#)
- PLUS
  - Symbol.h, [19](#)
- PRIME
  - SymbolTable.h, [20](#)
- parse
  - Parser, [9](#)
- Parser, [8](#)
  - addOp, [9](#)
  - admin, [9](#)
  - block, [9](#)
  - boolSym, [9](#)
  - exprList, [9](#)
  - factor, [9](#)
  - look, [9](#)
  - match, [9](#)
  - multOp, [9](#)
  - parse, [9](#)
  - Parser, [9](#)
  - program, [9](#)
  - simpleExpr, [9](#)
  - term, [9](#)
- Parser.h, [17](#)
- pos
  - Scanner, [11](#)
- probe
  - SymbolTable, [13](#)
- program
  - Parser, [9](#)
- RHRND
  - Symbol.h, [19](#)
- RHSQR
  - Symbol.h, [18](#)
- recognizeName
  - Scanner, [11](#)
- recognizeNumeral
  - Scanner, [11](#)
- recognizeSpecial
  - Scanner, [11](#)
- SEMI
  - Symbol.h, [18](#)
- scan
  - Administration, [7](#)
- Scanner, [10](#)
  - ~Scanner, [11](#)
  - fin, [11](#)
  - getToken, [11](#)
  - isSpecial, [11](#)
  - isWhitespace, [11](#)
  - line, [11](#)
  - pos, [11](#)
  - recognizeName, [11](#)
  - recognizeNumeral, [11](#)
  - recognizeSpecial, [11](#)

- Scanner, 10
- symmap, 12
- symtable, 12
- scanner
  - Administration, 8
- Scanner.h, 17
- search
  - SymbolTable, 13
- setLexeme
  - Token, 15
- setSymbol
  - Token, 15
- setVal
  - Token, 15
- simpleExpr
  - Parser, 9
- sname
  - Token, 15
- Symbol
  - Symbol.h, 18
- Symbol.h
  - AMP, 18
  - ARROW, 19
  - BAR, 18
  - BEGIN, 19
  - BOOL, 19
  - BSLASH, 19
  - CHAR\_ERR, 19
  - COMMA, 18
  - DOLLAR, 19
  - DOT, 18
  - EMPTY, 19
  - END, 19
  - ENDFILE, 19
  - EQUAL, 18
  - FALSE, 19
  - FSLASH, 19
  - FULL\_TAB, 19
  - GREAT, 18
  - GUARD, 19
  - ID, 19
  - INIT, 19
  - INT, 19
  - KEY, 19
  - LESS, 18
  - LHRND, 19
  - LHSQR, 18
  - MINUS, 19
  - NAME\_ERR, 19
  - NEWLINE, 19
  - NUM, 19
  - NUM\_ERR, 19
  - PLUS, 19
  - RHRND, 19
  - RHSQR, 18
  - SEMI, 18
  - TILD, 18
  - TIMES, 19
  - TRUE, 19
  - Symbol.h, 18
  - Symbol, 18
  - SymbolToString, 19
  - SymbolTable, 12
    - full, 13
    - getLoad, 13
    - hash, 13
    - insert, 13
    - keywords, 13
    - load, 14
    - loadKeywords, 13
    - probe, 13
    - search, 13
    - SymbolTable, 13
    - SymbolTable, 13
    - table, 14
    - toString, 13
  - SymbolTable.h, 19
    - ID\_MAX\_CHARS, 20
    - MOD, 20
    - PRIME, 20
  - SymbolToString
    - Symbol.h, 19
  - symmap
    - Scanner, 12
  - symtable
    - Scanner, 12
  - syntaxError
    - Administration, 7
  - TILD
    - Symbol.h, 18
  - TIMES
    - Symbol.h, 19
  - TRUE
    - Symbol.h, 19
  - table
    - SymbolTable, 14
  - term
    - Parser, 9
  - toString
    - SymbolTable, 13
    - Token, 15
  - Token, 14
    - getLexeme, 14
    - getSymbol, 14
    - getVal, 15
    - lexeme, 15
    - setLexeme, 15
    - setSymbol, 15
    - setVal, 15
    - sname, 15
    - toString, 15
    - Token, 14
    - val, 15
  - Token.h, 20
  - val

Token, [15](#)