

## CPSC 4660 Compiler

Generated by Doxygen 1.8.5

Sat Feb 1 2020 09:29:46



# 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	newLine . . . . .	7
3.1.2.4	scan . . . . .	7
3.1.3	Member Data Documentation . . . . .	7
3.1.3.1	correctLine . . . . .	7
3.1.3.2	errorCount . . . . .	7
3.1.3.3	fout . . . . .	7
3.1.3.4	lineNum . . . . .	7
3.1.3.5	scanner . . . . .	8
3.2	Scanner Class Reference . . . . .	8
3.2.1	Constructor & Destructor Documentation . . . . .	9
3.2.1.1	Scanner . . . . .	9
3.2.1.2	~Scanner . . . . .	10
3.2.2	Member Function Documentation . . . . .	10
3.2.2.1	getToken . . . . .	10
3.2.2.2	isSpecial . . . . .	10
3.2.2.3	isWhitespace . . . . .	10
3.2.2.4	recognizeName . . . . .	10
3.2.2.5	recognizeNumeral . . . . .	10
3.2.2.6	recognizeSpecial . . . . .	10

3.2.3	Member Data Documentation . . . . .	10
3.2.3.1	fin . . . . .	10
3.2.3.2	line . . . . .	11
3.2.3.3	pos . . . . .	11
3.2.3.4	symmap . . . . .	11
3.2.3.5	symtable . . . . .	11
3.3	SymbolTable Class Reference . . . . .	11
3.3.1	Constructor & Destructor Documentation . . . . .	12
3.3.1.1	SymbolTable . . . . .	12
3.3.2	Member Function Documentation . . . . .	12
3.3.2.1	full . . . . .	12
3.3.2.2	getLoad . . . . .	12
3.3.2.3	hash . . . . .	12
3.3.2.4	insert . . . . .	12
3.3.2.5	loadKeywords . . . . .	12
3.3.2.6	probe . . . . .	12
3.3.2.7	search . . . . .	12
3.3.2.8	toString . . . . .	12
3.3.3	Member Data Documentation . . . . .	13
3.3.3.1	keywords . . . . .	13
3.3.3.2	load . . . . .	13
3.3.3.3	table . . . . .	13
3.4	Token Class Reference . . . . .	13
3.4.1	Constructor & Destructor Documentation . . . . .	14
3.4.1.1	Token . . . . .	14
3.4.1.2	Token . . . . .	14
3.4.2	Member Function Documentation . . . . .	14
3.4.2.1	getLexeme . . . . .	14
3.4.2.2	getSymbol . . . . .	14
3.4.2.3	getVal . . . . .	14
3.4.2.4	setLexeme . . . . .	14
3.4.2.5	setSymbol . . . . .	14
3.4.2.6	setVal . . . . .	14
3.4.2.7	toString . . . . .	14
3.4.3	Member Data Documentation . . . . .	14
3.4.3.1	lexeme . . . . .	14
3.4.3.2	sname . . . . .	14
3.4.3.3	val . . . . .	14

4.1	Administration.h File Reference . . . . .	15
4.1.1	Variable Documentation . . . . .	15
4.1.1.1	MAX_ERRORS . . . . .	15
4.2	Scanner.h File Reference . . . . .	15
4.2.1	Macro Definition Documentation . . . . .	16
4.2.1.1	SCANNER_h . . . . .	16
4.3	Symbol.h File Reference . . . . .	16
4.3.1	Enumeration Type Documentation . . . . .	16
4.3.1.1	Symbol . . . . .	16
4.3.2	Variable Documentation . . . . .	17
4.3.2.1	SymbolToString . . . . .	17
4.4	SymbolTable.h File Reference . . . . .	17
4.4.1	Variable Documentation . . . . .	18
4.4.1.1	ID_MAX_CHARS . . . . .	18
4.4.1.2	MOD . . . . .	18
4.4.1.3	PRIME . . . . .	18
4.5	Token.h File Reference . . . . .	18



# 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="#">Scanner</a>	8
<a href="#">SymbolTable</a>	11
<a href="#">Token</a>	13





## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Administration.h</a>	15
<a href="#">Scanner.h</a>	15
<a href="#">Symbol.h</a>	16
<a href="#">SymbolTable.h</a>	17
<a href="#">Token.h</a>	18



## 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.*
- `void newLine ()`  
*Adds line number and resets `correctLine`.*
- `void error (std::string text)`  
*Display text for an error.*
- `int scan ()`  
*Scan the whole file and output all tokens to `fout`.*

#### Private Member Functions

- `void checkError (Token ntoken)`  
*Checks if current token is an error token.*

#### Private 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* ) [private]

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 void Administration::newLine ( )

Adds line number and resets correctLine.

#### 3.1.2.4 int Administration::scan ( )

Scan the whole file and output all tokens to fout.

Returns the number of tokens.

### 3.1.3 Member Data Documentation

#### 3.1.3.1 bool Administration::correctLine [private]

True if the line has no errors so far.

#### 3.1.3.2 int Administration::errorCount [private]

The total number of errors so far.

#### 3.1.3.3 std::ostream& Administration::fout [private]

File to print all tokens to.

#### 3.1.3.4 int Administration::lineNum [private]

The current line number.

### 3.1.3.5 Scanner& Administration::scanner [private]

The scanner to use on the input.

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

- [Administration.h](#)

## 3.2 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 rthe 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 inputed 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 postion of the char the scanner is reading.*
- std::map< std::string, [Symbol](#) > [symmap](#)  
*The map containing the symbols.*

### 3.2.1 Constructor & Destructor Documentation

#### 3.2.1.1 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.

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

Destructor of the scanner.

## 3.2.2 Member Function Documentation

## 3.2.2.1 Token Scanner::getToken( )

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

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

Checks the inputted char against all possible symbols.

## Parameters

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

## 3.2.2.3 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
---------------	--------------------------------

## 3.2.2.4 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.

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

Read and generate a token for any number/digit.

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

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

## 3.2.3 Member Data Documentation

## 3.2.3.1 std::istream&amp; Scanner::fin [private]

The file stream.



**3.2.3.2** `std::string Scanner::line` `[private]`

The current line the scanner is reading.

**3.2.3.3** `std::size_t Scanner::pos` `[private]`

The position of the char the scanner is reading.

**3.2.3.4** `std::map<std::string, Symbol> Scanner::symmap` `[private]`

The map containing the symbols.

**3.2.3.5** `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](#)

## 3.3 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`

## 3.3.1 Constructor & Destructor Documentation

### 3.3.1.1 `SymbolTable::SymbolTable ( )`

## 3.3.2 Member Function Documentation

### 3.3.2.1 `bool SymbolTable::full ( )`

Returns true if the table is full.

### 3.3.2.2 `int SymbolTable::getLoad ( )`

Returns the number items in the table.

### 3.3.2.3 `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.

### 3.3.2.4 `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.

### 3.3.2.5 `void SymbolTable::loadKeywords ( )` [private]

Loads all reserved keywords into the symbol table.

### 3.3.2.6 `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.

### 3.3.2.7 `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.

### 3.3.2.8 `std::string SymbolTable::toString ( )`

Returns a string representation of the table.

### 3.3.3 Member Data Documentation

3.3.3.1 `const std::vector<std::string> SymbolTable::keywords` [private]

Initial value:

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

3.3.3.2 `int SymbolTable::load` [private]

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

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

- [SymbolTable.h](#)

## 3.4 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](#)

### 3.4.1 Constructor & Destructor Documentation

3.4.1.1 `Token::Token ( )`

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

### 3.4.2 Member Function Documentation

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

Returns the lexeme.

3.4.2.2 `Symbol Token::getSymbol ( )`

Returns the symbol.

3.4.2.3 `int Token::getVal ( )`

Returns the value.

3.4.2.4 `void Token::setLexeme ( std::string lexeme )`

Sets the lexeme.

3.4.2.5 `void Token::setSymbol ( Symbol sym )`

Sets the symbol.

3.4.2.6 `void Token::setVal ( int val )`

Sets the value.

3.4.2.7 `void Token::toString ( std::ostream & out ) const`

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

### 3.4.3 Member Data Documentation

3.4.3.1 `std::string Token::lexeme` `[private]`

3.4.3.2 `Symbol Token::sname` `[private]`

3.4.3.3 `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"
```

#### 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 Scanner.h File Reference

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

#### Classes

- class [Scanner](#)

#### Macros

- #define [SCANNER\\_h](#)

## 4.2.1 Macro Definition Documentation

### 4.2.1.1 #define SCANNER\_h

## 4.3 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, 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.*

## 4.3.1 Enumeration Type Documentation

### 4.3.1.1 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***  
***ID***  
***KEY***  
***ENDFILE***  
***EMPTY***  
***NEWLINE***  
***NUM***  
***NAME\_ERR***  
***NUM\_ERR***  
***CHAR\_ERR***  
***FULL\_TAB***

#### 4.3.2 Variable Documentation

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

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

## 4.4 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

#### 4.4.1 Variable Documentation

4.4.1.1 `const int ID_MAX_CHARS = 10`

4.4.1.2 `const int MOD = 307`

4.4.1.3 `const int PRIME = 67`

### 4.5 Token.h File Reference

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

#### Classes

- class [Token](#)



# Index

- ~Scanner
  - Scanner, [10](#)
- AMP
  - Symbol.h, [16](#)
- ARROW
  - Symbol.h, [17](#)
- Administration, [5](#)
  - Administration, [6](#)
  - checkError, [7](#)
  - correctLine, [7](#)
  - error, [7](#)
  - errorCount, [7](#)
  - fout, [7](#)
  - lineNum, [7](#)
  - newLine, [7](#)
  - scan, [7](#)
  - scanner, [7](#)
- Administration.h, [15](#)
  - MAX\_ERRORS, [15](#)
- BAR
  - Symbol.h, [16](#)
- BOOL
  - Symbol.h, [17](#)
- BSLASH
  - Symbol.h, [17](#)
- CHAR\_ERR
  - Symbol.h, [17](#)
- COMMA
  - Symbol.h, [16](#)
- checkError
  - Administration, [7](#)
- correctLine
  - Administration, [7](#)
- DOLLAR
  - Symbol.h, [17](#)
- DOT
  - Symbol.h, [16](#)
- EMPTY
  - Symbol.h, [17](#)
- ENDFILE
  - Symbol.h, [17](#)
- EQUAL
  - Symbol.h, [16](#)
- error
  - Administration, [7](#)
- errorCount
  - Administration, [7](#)
- FALSE
  - Symbol.h, [17](#)
- FSLASH
  - Symbol.h, [16](#)
- FULL\_TAB
  - Symbol.h, [17](#)
- fin
  - Scanner, [10](#)
- fout
  - Administration, [7](#)
- full
  - SymbolTable, [12](#)
- GREAT
  - Symbol.h, [16](#)
- GUARD
  - Symbol.h, [17](#)
- getLexeme
  - Token, [14](#)
- getLoad
  - SymbolTable, [12](#)
- getSymbol
  - Token, [14](#)
- getToken
  - Scanner, [10](#)
- getVal
  - Token, [14](#)
- hash
  - SymbolTable, [12](#)
- ID
  - Symbol.h, [17](#)
- INIT
  - Symbol.h, [17](#)
- INT
  - Symbol.h, [17](#)
- ID\_MAX\_CHARS
  - SymbolTable.h, [18](#)
- insert
  - SymbolTable, [12](#)
- isSpecial
  - Scanner, [10](#)
- isWhitespace
  - Scanner, [10](#)
- KEY
  - Symbol.h, [17](#)
- keywords

- SymbolTable, [13](#)
- LESS
  - Symbol.h, [16](#)
- LHRND
  - Symbol.h, [17](#)
- LHSQR
  - Symbol.h, [16](#)
- lexeme
  - Token, [14](#)
- line
  - Scanner, [10](#)
- lineNum
  - Administration, [7](#)
- load
  - SymbolTable, [13](#)
- loadKeywords
  - SymbolTable, [12](#)
- MINUS
  - Symbol.h, [16](#)
- MAX\_ERRORS
  - Administration.h, [15](#)
- MOD
  - SymbolTable.h, [18](#)
- NAME\_ERR
  - Symbol.h, [17](#)
- NEWLINE
  - Symbol.h, [17](#)
- NUM
  - Symbol.h, [17](#)
- NUM\_ERR
  - Symbol.h, [17](#)
- newLine
  - Administration, [7](#)
- PLUS
  - Symbol.h, [16](#)
- PRIME
  - SymbolTable.h, [18](#)
- pos
  - Scanner, [11](#)
- probe
  - SymbolTable, [12](#)
- RHRND
  - Symbol.h, [17](#)
- RHSQR
  - Symbol.h, [16](#)
- recognizeName
  - Scanner, [10](#)
- recognizeNumeral
  - Scanner, [10](#)
- recognizeSpecial
  - Scanner, [10](#)
- SEMI
  - Symbol.h, [16](#)
- SCANNER\_h
  - Scanner.h, [16](#)
- scan
  - Administration, [7](#)
- Scanner, [8](#)
  - ~Scanner, [10](#)
  - fin, [10](#)
  - getToken, [10](#)
  - isSpecial, [10](#)
  - isWhitespace, [10](#)
  - line, [10](#)
  - pos, [11](#)
  - recognizeName, [10](#)
  - recognizeNumeral, [10](#)
  - recognizeSpecial, [10](#)
  - Scanner, [9](#)
  - symmap, [11](#)
  - symtable, [11](#)
- scanner
  - Administration, [7](#)
- Scanner.h, [15](#)
  - SCANNER\_h, [16](#)
- search
  - SymbolTable, [12](#)
- setLexeme
  - Token, [14](#)
- setSymbol
  - Token, [14](#)
- setVal
  - Token, [14](#)
- sname
  - Token, [14](#)
- Symbol
  - Symbol.h, [16](#)
- Symbol.h
  - AMP, [16](#)
  - ARROW, [17](#)
  - BAR, [16](#)
  - BOOL, [17](#)
  - BSLASH, [17](#)
  - CHAR\_ERR, [17](#)
  - COMMA, [16](#)
  - DOLLAR, [17](#)
  - DOT, [16](#)
  - EMPTY, [17](#)
  - ENDFILE, [17](#)
  - EQUAL, [16](#)
  - FALSE, [17](#)
  - FSLASH, [16](#)
  - FULL\_TAB, [17](#)
  - GREAT, [16](#)
  - GUARD, [17](#)
  - ID, [17](#)
  - INIT, [17](#)
  - INT, [17](#)
  - KEY, [17](#)
  - LESS, [16](#)
  - LHRND, [17](#)
  - LHSQR, [16](#)

- MINUS, [16](#)
- NAME\_ERR, [17](#)
- NEWLINE, [17](#)
- NUM, [17](#)
- NUM\_ERR, [17](#)
- PLUS, [16](#)
- RHRND, [17](#)
- RHSQR, [16](#)
- SEMI, [16](#)
- TILD, [16](#)
- TIMES, [16](#)
- TRUE, [17](#)
- Symbol.h, [16](#)
  - Symbol, [16](#)
  - SymbolToString, [17](#)
- SymbolTable, [11](#)
  - full, [12](#)
  - getLoad, [12](#)
  - hash, [12](#)
  - insert, [12](#)
  - keywords, [13](#)
  - load, [13](#)
  - loadKeywords, [12](#)
  - probe, [12](#)
  - search, [12](#)
  - SymbolTable, [12](#)
  - SymbolTable, [12](#)
  - table, [13](#)
  - toString, [12](#)
- SymbolTable.h, [17](#)
  - ID\_MAX\_CHARS, [18](#)
  - MOD, [18](#)
  - PRIME, [18](#)
- SymbolToString
  - Symbol.h, [17](#)
- symmap
  - Scanner, [11](#)
- symtable
  - Scanner, [11](#)
- TILD
  - Symbol.h, [16](#)
- TIMES
  - Symbol.h, [16](#)
- TRUE
  - Symbol.h, [17](#)
- table
  - SymbolTable, [13](#)
- toString
  - SymbolTable, [12](#)
  - Token, [14](#)
- Token, [13](#)
  - getLexeme, [14](#)
  - getSymbol, [14](#)
  - getVal, [14](#)
  - lexeme, [14](#)
  - setLexeme, [14](#)
  - setSymbol, [14](#)
  - setVal, [14](#)
  - sname, [14](#)
  - toString, [14](#)
  - Token, [14](#)
  - val, [14](#)
- Token.h, [18](#)
- val
  - Token, [14](#)