CPSC 4660 Compiler

Generated by Doxygen 1.8.5

Sat Feb 1 2020 19:36:57

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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

fout	The output file stream.
sc	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

ntoken	The current token.

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

Display text for an error.

Parameters

text	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

actual	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

admin	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 varaibles 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.

3.3.1 Constructor & Destructor Documentation

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

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

Parameters

ifs	The file stream.
symboltable	The Symbol Table used throughout the scan being updated.

3.3.1.2 Scanner::∼Scanner ( ) [inline]

Destructor of rthe 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 inputed char against all possible symbols.

Parameters

inchar	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

inchar	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 postion 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