Clayton Green (230089109)
kgreen1@unbc.ca
September 20, 2013

CPSC 425 - Compiler Project Phase I

Introduction

This phase consists of the Scanner. The scanner is able to recognize tokens from the c13 language source files. Also included is an Administrator module that runs the Scanner and will run the Parser in the future. A messaging class is also included for printing out debug messages when the command line switch -t is present.

Participation

Clayton Green - All

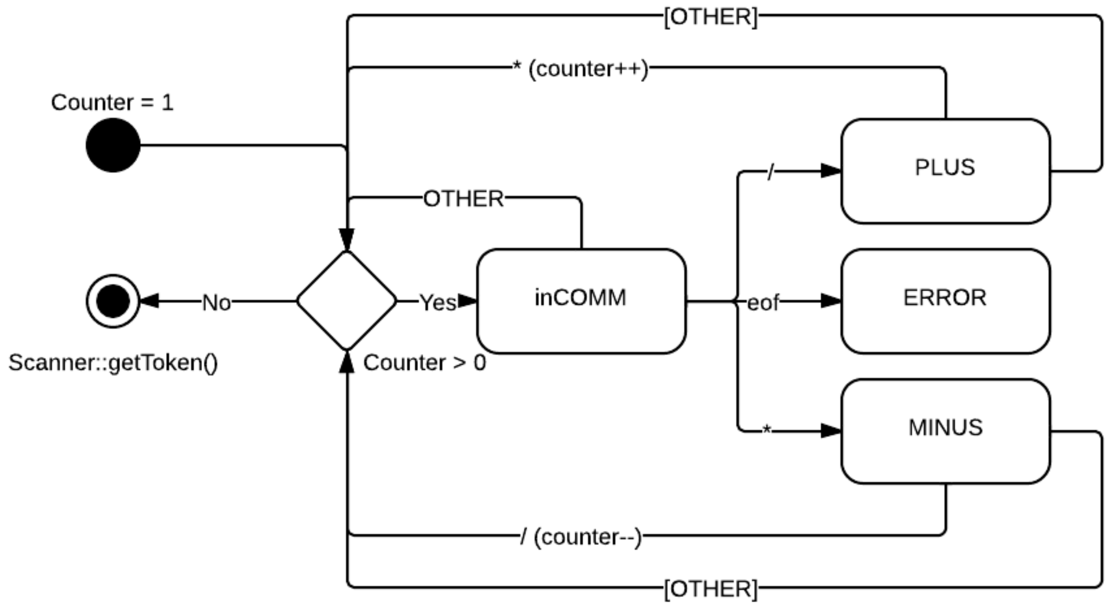
Project Status

Scanner - Complete
Others
+ Fix --err/-e appending to file rather than overwriting
+ Add functionality for --out/-o

Architecture and Design

Scanner State Diagram: refer to *“Scanner State Diagram.png”*

Comment State Diagram:



Implementation

All of the classes have simple usage examples in their comments. Basically the compiler works by creating an Administrator object and passing it a source file. When the administrator is constructed so are the required lists and maps (word table, spelling table, etc.). Calling the compile method will, currently, cause the Scanner to continually getToken() until and ENDFILE token has been read in. If tracing has been enabled the compiler will output each line and under the line the tokens found on that line. Regardless if tracing is enabled the list, if any, of errors encountered are printed to the screen. One limitation, which can be easily change, is that the debug output for printing the lines are only 128 characters long (char array is only 128).

Building and Use

Building:
Using the following will compile the program using g++ on the Ubuntu computers at the university. (Note: c+11 features are used)

```
$ g++ -std=c++0x main.cpp Administrator.cpp Messenger.cpp Scanner.cpp Token.cpp -
o Compiler
```

Usage:
The compiler has one option to display the scanner trace using the switch *-t*. A filename is mandatory and is chosen using *-f filename.cs13* (If the file is not of type .cs13 the program will exit).

```
$ ./Compiler -t -f test.cs13     # will show scanner trace and errors
$ ./Compiler -f test.cs13         # will only show errors
```

Code

Administrator Class (Administrator.h/Administrator.cpp)

The administrator class that uses the Scanner to find Tokens in a c13 source file and prints messages if debugging is activated. The main public method is compile(), which gets the scanner to getToken() until an ENDFILE token is received.

Messenger Class (Messenger.h/Messenger.cpp)

The messenger class is completely static that can print debug messages if debuging is active and keeps a list of error messages to print when scanning completes.

Scanner Class (Scanner.h/Scanner.cpp)

The scanner class looks through a given source file and finds and constructs tokens from its contents. The main public method is getToken() which returns the next token the scanner finds.

Token Class (Token.h/Token.cpp)

Tokens are defined in one class with an enum differentiating the type of token (STD_TOKEN, WORD_TOKEN, NUM_TOKEN) not inheritance. The token has two basic methods used for printing: one overriding the << operator and the other toString() for instances when << is not appropriate.

main.cpp

Has methods for parsing command line arguments and creates and runs the administrator.

**All classes have small usage examples*

Tests and Observations

To run tests you can use the test.sh file if on unix or just run the compiler with tracing on and load the file test.cs13. Comments in the test file say what is being tested and what to look for. With trace on the tokens will be printed out and if invalid will print an ERROR token. The test checks to see if all of the reserved words are initialized correctly, if identifiers are correctly identified (include identifiers with $ and _ in their names), as well as numbers and boolean literals. Pseudo tokens are checked. Simple errors like invalid tokens are checked (:, &, |, $, _). Control codes (Invisible characters) are checked. Finally both types of comments are checked and whitespace.