CS 280: Program #2

Spring 2016

Due March 7

It's sometimes useful to make a little language for a simple problem. We are making a language to let us play with strings and numbers, and for this assignment we are building a lexical analyzer for this simple language.

Here are the lexical rules for the language:

1. The language has identifiers. An identifier starts with a letter and is followed by zero or more letters.
2. The language has string constants. A string constant is a sequence of characters, all on one line, enclosed in double quotes. You do not have to support special handling for escape characters.
3. The language has integer constants, defined as a sequence of digits.
4. The language has 4 operators: +, -, *, and =
5. The language has 3 keywords: "print", "int" and "string".
6. Statements in the language end in a semicolon.
7. The language supports parentheses.
8. White space is used to separate tokens and lines for readability.
9. A comment begins with two slashes (//) and ends at a newline.

The lexical analyzer is to be implemented in a C++ function. The function will be passed a pointer to an input stream to read from. It will return a Token representing the token and lexeme that has been recognized.

The definitions for the Token class and unique values for each of the tokens (TokenType) that you must recognize is provided in the header file p2lex.h, which is on the course website.

The lexical analyzer must ignore white space and comments, using them only to note separation between tokens. The program should maintain an external integer named "linenum", which should be initialized to 1 and incremented whenever a newline is seen by the lexer.

A lexical error should cause the token ERR to be returned. An end of file should cause DONE to be returned.

You **MUST** use the p2lex.h header file for your assignment. **You may not change it.** You **do not** need to hand in p2lex.h, but it doesn't matter if you do. I will compile and test your program against the distributed p2lex.h header file

You must produce and submit three files:

1. **p2lex.cpp**
   This file should #include p2lex.h and should provide an implementation for the copy constructor, the assignment operator, and the << operator to print out a Token.

   The printed version of the token must be a string representation of the TokenType. We define the string representation of a TokenType to be the symbol for the TokenType in lowercase letters. For example, the string representation of the TokenType value PRINTKW should be "printkw".

   For the VAR, SCONST, ICONST and ERR tokens, the string representation must be followed by the lexeme for the token. The lexeme should be printed in parentheses.

2. **getToken.cpp**
   This file should #include p2lex.h and provide an implementation of the getToken function. The function should read from the stream pointed to by the first argument and return the token that it recognizes.

3. **project2.cpp**
   This file should #include p2lex.h and provide a main program to test the lexical analyzer
   The specifications for the main program are as follows:
   1. The program should accept at most one command line argument. If present, this one command line argument is the name of a file to open and read for input. If not present, the program should read from the standard input.
   2. There may also, optionally, be an initial command line argument equal to the string “-v”. If it is, then your program should be in “verbose mode” and should print each token that it recognizes in addition to the remaining items.
   3. The program should loop repeatedly, calling getToken(), until getToken returns an ERR or DONE token
   4. If the loop stopped because of an ERR token, it should print out the token
   5. The program should print out a count of the number of times each type of token was seen.
   6. The program should print out the lexeme for any string, integer or variable that appears more than once