CSCI 2400 – Models of Computation, Section 3

Programming Assignment 1

Return Day: October 30.
Teams: You are encouraged to form teams of 2 persons.

1 Overview

In this assignment you will use the lexical analyzer **lex**, to write a small program that recognizes the followings strings:

- Operators:

  `+ - * / = > >= < <= ==`

- Parentheses and semicolumn:

  `( ) ;`

- Keywords:

  `if, then, else, endif, while, do, endwhile, print, newline`

- Integers:

  An integer is any string made of decimal digits (0,1,..,9) which can start with a sign (+ or -). Examples: 129, 000234, -1789.

- Identifiers:

  An identifier is any string that starts with a letter and then continues with any combination of letters and decimal digits.

  Examples: var1, x, y, z34r4.

- Quoted Strings:

  A quoted string is any string that appears between two quotes.

  Example:

  "*hello, I am a quoted string*" .

  A quoted string should not include the quotes and it should not occupy more than one lines. A quoted string can include spaces and tabs.

2 The Input

The input for your program is any file with text.

The standard input of your program is from the keyboard. When your input is from a file you can use the unix redirection for input as in: `scan < input.txt`.

3 The Output

The output of your program will be a description of the strings that the program recognizes while it reads the input file. For each kind of string the output is as follows.

- Operators, Parentheses, Semicolumn, and Keywords:

  You should print in the output the kind of string you have recognized.

- Integers:

  You should print the integer value of the integer string.

  To do this, you need to convert the integer string to an integer value. You can do this by using the following operation:

  ```
  sscanf(yytext, ‘‘%d’’, &int_value);
  ```

  where `yytext` contains the last string recognized by *lex*.

- Identifiers:

  You should print the string for the identifier. Also you should print whether the identifier is new or if it appeared earlier in the text.

  To detect whether an identifier is new or it appeared earlier you need to build a *symbol table*. A symbol table is an array that contains the various identifiers that appeared so far in the input. Each entry in the array contains the string of the corresponding identifier.

  Each time *lex* recognizes an identifier from the input file, you need to compare this identifier with all the identifiers in the symbol table. If the identifier is different from all the identifiers in the symbol table then you have found a new identifier and you need to insert it in the symbol table. If there is a match with some identifier in the symbol table then the identifier has appeared before and you should do not insert it in the symbol table.

  You are free to implement the symbol table as you like. (It can be implemented with a linear array or more efficiently with a hash table. Be as efficient as you want.)

- Quoted Strings:

  You should print the quoted string.

  For this you need to remove the quotes from the `yytext` string. Use the operation

  ```
  sscanf(yytext, ‘‘\"%s\"’’, str);
  ```

  where `str` will hold the quoted string.

Normally, the standard output of your program will be the screen. If you want to save the output of you program to a file you should use the unix redirection for output as in: `scan > output.txt`.

4 What to Handin

You should handin your *lex* and C source codes in paper. You should also handin in paper the output results of you program for input test files which will be provided to you. You will find these test files at the course web page.

5 Helpful Tips

For more infomation about *lex* (and *yacc*) look at:

`http://www.combo.org/lex_yacc_page/`

You will find sample *lex* program files and sample input and output files in the course web page.