CSCI 2400 – Models of Computation

Project 1: Lex

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...9. Examples:
  129, 000234, 1789000

- *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.

- *Comments*

  A comment is any string that starts with the characters "//" and ends at the end of a line. Example:
  // this is a comment

  A comment can include spaces and tabs.

2 Input and Output

The input of your program is any file with text.

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. For each kind of string use a different line in your lex code. Example:

  ```
  "+"    {printf("operator: + \n");}
  "-"    {printf("operator: - \n");}
  "*"    {printf("operator: * \n");}
  "/"    {printf("operator: / \n");}
  ```

- *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. Otherwise, if there is a match with some identifier in the symbol table then the identifier has appeared before and you should 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, and in order to achieve this you may use the following operations:

  ```
  strncpy(str, &(yytext[1]), strlen(yytext)-2);
  str[strlen(yytext)-2] = (char) 0;
  ```

  where `str` will hold the quoted string.

- *Comments*

  You should ignore comments. Your program won't print anything when you identify comments.

- *Wrong Input*

  When you encounter a string that doesn't match any of the above cases then you should print an error message indicating the line number in which the error occurs, and then your program should abort execution.

3 Compiling and Running your Lex Code

The discussion below is for the Unix environment.

After you write your lex program and save it in a file, e.g. `scan.l`, you can compile the program by executing the following commands:

```
lex scan.l
cc lex.yy.c -ll -o scan
```

where `scan` is the resulting executable file. To run your program just type `scan`.

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

The computer monitor is the standard output of your program. To save the output of your program to a file, e.g. `output.txt`, you can use the redirection for output: `scan > output.txt`.

4 Hand-in

We will provide you with submission instructions when the due date comes close by.

5 Sample Input and Output

Here is a sample input file and the respective output file generated by the lex program.

Input

```
+
-
*
/
=
>
>=
<
<=
==
!=
(
)
;
if
then
else
endif
while
do
endwhile
print
newline
000000000000345
id1
id2
id1
"hello, how are you?"
```

Output

```
operator: +
operator: -
operator: *
operator: /
operator: =
operator: >
operator: >=
operator: <
operator: <=
operator: ==
operator: !=
parenthesis: (
parenthesis: )
```

```
semicolumn: ;
keyword: if
keyword: then
keyword: else
keyword: endif
keyword: while
keyword: do
keyword: endwhile
keyword: print
keyword: newline
integer: 345
New Identifier "id1"
New Identifier "id2"
Identifier "id1" already in symbol table
quoted string: hello, how are you?
```