Lab7 - Introduction to flex

Started: Oct 18 at 10:08pm

Quiz Instructions

Question 1 — 100 pts

flex is a gnu implementation of lex, a program that allows you to process text searching for patterns and an process matches with corresponding handlers. Patterns to match are expressed as regular expressions, and the handlers are implemented in C. The output of flex is a C file that is a complete implementation of the pattern matcher.

Although flex can be used for any scanner, it was designed as a generator of lexical analyzers that you implemented by hand a couple of weeks ago.

Please study the lex tutorial before starting the implementation of this lab:

http://epaperpress.com/lexandyacc/ **(http://epaperpress.com/lexandyacc/)**

(here is a local **PDF version**

- in case the tutorial Web site is down).

Focus on the "Lex" menu item and the "Practice" in this lab. We will study the other parts in the following weeks.

Starting with the project seed in **lab7_flex_t1.zip** implement a lex-based C program that scans for all the tokens of the language (keywords, identifiers, numbers, operators, and so on) built over the following grammar:

```
<program> ::= <statement> | <program> <statement>
<statement> ::= <assignStmt> | <ifStmt> | <whileStmt> |
<printStmt>
<assignStmt> ::= <id> = <expr> ;
<ifStmt> ::= if ( <expr> ) then <stmt>
<whileStmt> ::= while ( <expr> ) do <stmt>
<printStmt> ::= print <expr> ;
<expr> ::= <term> | <expr> <addOp> <term>
<term> ::= <factor> | <term> <multOp> <factor>
<factor> ::= <id> | <number> | - <factor> | ( <expr> )
<id> ::= <letter> | <id> [ <letter> | <digit> | _ ]
```

```
<letter> ::= a | b | c | d | e | f | g | h | i | j
| k | l | m | n | o | p | r | s | t
| u | v | w | x | y | z
<number> ::= <digit>+ [ . <digit>+ ]
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<addOp> ::= + | -
<multOp> ::= * | / | %
```

The `main()` should include the code for printing all tokens returned from the scanner following the sample output provided in the package.

To test the scanner you may either enter the test program in the console, or you can put the test program in a file (as done with `flex_t1in.txt` in the distributed CLion project), and then either copy and paste the text into the console, or add the name of the input file to the CLion configuration for the executable at the "Program arguments" label. For any text added through the console, you need to provide the end of file character to finish the program. That is done with the Command-d key combination on macOS.

In this lab, please ignore the file `dummy.y` that is an empty yacc/bison file necessary for inducing creation of the scanner by lex/flex.

Please submit a zip archive of your complete CLion project directory.

Upload Choose a File

Saving... Submit Quiz