Compilers

Programming Assignment II

Objectives

1. To become familiar with flex, a popular program used to generate lexical analyzers.
2. To implement a lexical analyzer for programs written in DJ.
3. To understand which strings constitute valid tokens in DJ.
4. To practice writing regular expressions by specifying valid DJ tokens.

Due Date: Sunday, February 9, 2025 (at 11:59pm).

Machine Details

Complete this assignment by yourself on the cselx##.cse.usf.edu computers. You are responsible for ensuring that your programs compile and execute properly on these machines.

Assignment Description

This assignment asks you to implement the lexical-analysis phase of our DJ compiler. You will use flex (the current version of lex) to generate a lexical analyzer for programs written in DJ.

Please begin by downloading the auxiliary file *dj.y* from http://www.cse.usf.edu/~ligatti/compilers/25/a2/. In *dj.y* you will find an enumeration of all possible DJ tokens.

Then, in a new file called *dj.l* (that is the letter "el", which stands for "lex", after the period), write a flex input file that recognizes the token types enumerated in *dj.y*. Whenever your lexer recognizes a DJ token, it should print the token to the screen (example executions are shown on the next page).

Big Hint

You will probably find it helpful to study the *dism.l* file (available at http://www.cse.usf.edu/~ligatti/compilers/25/a1/dism/sim-dism/) and use *dism.l* as a reference for your own *dj.l* file. Your *dj.l* will probably be quite similar to *dism.l*, except that *dj.l* will have the DEBUG flag set to 1 instead of 0, and *dj.l* will have different tokens and regular expressions to match those tokens.

Behavior on Source-program Errors

For all assignments in this course, your program may exit immediately after detecting and reporting the first error, with an accurate line number given for the error. Of course, such behavior may be undesirable, from a user's perspective. Hence, although not required, you might try to implement all phases of compilation such that as many errors as possible get reported (always with accurate line numbers) before exiting the compiler.

Compilation of the Lexer

Compile your lexer with the following commands (where ">" is a command prompt).

```
> flex dj.l
> bison dj.y
> gcc dj.tab.c -o lexdj
```

For full credit, your lexer must compile without warnings or errors.

Example Executions

The *good1.dj* program is:

```
class C extends Object { }
main {
 0;
}
```

This *good1.dj* file is correctly tokenized as follows.

```
> ./lexdj good1.dj
CLASS ID(C) EXTENDS ID(Object) LBRACE RBRACE MAIN LBRACE NATLITERAL(0)
SEMICOLON RBRACE ENDOFFILE
>
```

As another example, the *good3.dj* program is:

```
//prints 4
main {
  nat x;
  x=0;
  x=x+1;
  x=x+1;
  x=x+1;
  x=x+1;
  printNat(x);
}
```

This *good3.dj* file is correctly tokenized as follows.

```
> ./lexdj good3.dj
MAIN LBRACE NATTYPE ID(x) SEMICOLON ID(x) ASSIGN NATLITERAL(0) SEMICOLON ID(x)
ASSIGN ID(x) PLUS NATLITERAL(1) SEMICOLON ID(x) ASSIGN ID(x) PLUS NATLITERAL(1)
SEMICOLON ID(x) ASSIGN ID(x) PLUS NATLITERAL(1) SEMICOLON ID(x) ASSIGN ID(x)
PLUS NATLITERAL(1) SEMICOLON PRINTNAT LPAREN ID(x) RPAREN SEMICOLON RBRACE
ENDOFFILE
>
```

Please note that we'll test your lexer on DJ files that haven't been distributed to the class.

Extra Credit

For up to +10% extra credit, also implement a DJ lexer in C "by hand" (that is, without using lex/flex and without using any C libraries besides *stdlib* and *stdio*).

Formatting Issues

If you ever save your *dj.l* file on a Windows machine, be sure to run the command *dos2unix dj.l* on a C4 machine before using flex. This command will remove extra whitespace symbols (e.g., ^M) inserted by Windows, which may affect flex.

Submission Notes

- Upload and submit your *dj.l* file in Canvas. You may submit your assignment in Canvas as many times as you like; we will grade your latest submission.
- For every day that your assignment is late, up to 2 days, your grade reduces 10%.
- As always, submissions will be graded on correctness and style, so ensure that your *dj.l* isn't significantly more complicated than necessary, uses spaces rather than tabs, avoids overly long lines of code, etc.