Compilers Design: Lexical Analyzer

Spring 2018

Programming Assignment 1

Due Date: June 19, 2018

You are going to write a lexical analyzer for a toy programming language whose lexical structure is defined below. You have to design your lexical analyzer as a subroutine which can be called by a parser and returns a quadraple *(tokentype,attributevalue,linenumber,lineposition)*. The first item in the quadraple is the token type, the second one is its attribute value, the third one is the number of the line in which that token occurs, and the last one is the position of the token in that line. Your lexical analyzer should not return a quadraple for a comment, it should return the next token after skipping the comments. Then you should design a driver program which reads the name of a file (Ex. test1.toy) which holds a sample program in this toy programming language, and prints the tokens of this sample program into another file (test1.tok). Each line of the output file should contain a token (quadraple that representing that token.

When you design your lexical analyzer, make sure that you resolve the following issues.

- What are your token types? What are attribute values of tokens? What do they represent?
- What is the structure of the symbol table? What do you store and how?
- What kind of lexical errors can your lexical analyzer handle, and how it can recover from those errors?

The lexical structure of this toy programming language is defined as follows:

- **comments**: Comments starts with `//` characters and ends with the end of line character.
- **keywords**: The keywords given in the grammar:
  `int real if then else while`
- **identifiers**: An identifier starts with a letter and continues with a letter or digit. A keyword cannot be an identifier, and the maximum length of an identifier is 64 characters.
- **operators**: The followings are operators: **`+ - / * = == < <= > >= !=`**
- **delimiters**: The followings are delimeters: **`( ) { } ; ,`**
- **numbers**: The numbers are defined as follows:

```
digit      ← 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
intnumber  ← digit⁺                         (the maximum integer number is 2^31)
exponent   ← (E|e) ( + | - | ε ) digit⁺     (the maximum exponent value is 128)
fraction   ← . digit⁺
realnumber ← digit⁺ exponent | digit⁺ fraction ( exponent | ε)
```

Test your lexical analyzer with sample programs of this toy programming language. This test should read a sample program from the given input file (your program should read the input file name), and prints its tokens into an output file.

You should hand-in:

- One or two page document (type-written) explaining your design and answering the design questions above.
- Your source program
- Some test runs of your program (an input file and the corresponding output file).
- Above all email to xmju@sei.ecnu.edu.cn or upload 202.120.91.126

DO IT YOURSELF – CHEATING WILL BE PUNISHED