Programming Assignment #3: Lexical Analysis

CS 310 – Programming Languages – Spring 2023

Goals for this assignment

- Write our own Lexer in C++

Academic Honesty

Please read carefully the section titled “Academic Integrity” in the syllabus. If you have any questions, contact your instructor before you begin this assignment.

Overview

In this assignment, we’ll focus on lexical analysis. You’ll write your own lexer in C++ and test it on a sample file.

Build your own simple lexer

Start by downloading the Lexer class from the web site: **Lexer.cc** and **Lexer.h**. This is the same as the example code from class, and based on the example lexer in the textbook.

- Compile and test the lexer on an input file of your own creation. Verify that it recognizes the lexemes and produces the correct token classes. Read the code thoroughly and understand how it works. Please *do not proceed* until you fully understand how the code works.
- Note the difference between a character class and a token class. They are used for different purposes.
- Starting with the state diagram from lecture that recognizes int literals and identifiers, add states and transitions that would recognize each of the following tokens.
  - String literals (token code 40): a quote character, followed by zero or more characters (anything except another quote or a end of line), followed by another quote character.
  - Float literals (token code 41): zero or more digits, followed by a dot character (’.’), followed by one or more digits, followed by an optional **f** or F.
  - Turn in a copy of your state diagram.
- Add a token code in Lexer.h for each of the two above tokens, and a code for each of the following:
  – **Keywords (code 50)**: **if**, **else**, **for**, do, **while**, **switch**, case, **default**, **break**, **void** or **return**
  – **{** (code 27)
  – **}** (code 28)
  – **;** (code 100)
  – **,** (code 60)
  – = (code 20, already there)
- You may also find it useful to add an additional character class for the quotation mark (") and one for the dot (.).
- Update the Lexer class to recognize the above additional tokens. Using the appropriate token code for each. (Note: do not try to put all of your additional code into the switch statement. Instead, I recommend that you break each case out into a separate (private) method.) If your lexer encounters a token error (e.g. no digits following a dot), it should print an appropriate error message and exit.

- You can test your lexer on the input file provided on the web site: `test.txt`. Example output is shown in: `sampleout.txt`

Submission Instructions

- Add a comment header to the top of your main source code file including your name, semester, date, sources used, and any special instructions.
- Submit your revised `Lexer.cc`, and `Lexer.h` on the due date.