# Department of Computing

**CS 354: Compiler Construction**

**Class:** BSCS-8AB

# Lab [05]: Lexical Analysis with flex

**Date:** 29th Oct, 2021

# Time: Friday (10:00 – 1:00 & 02:00 - 5:00)

# Instructor: Dr Syed Atif Ali Shah

# Lab Instructor: M . Danyal Sadiq

# Lab [05]: Lexical Analysis with flex

**Introduction**

The lexical analyzer is the part of the compiler that reads the source text, it may also perform certain secondary tasks at the user interface. One such task is stripping out comments and white space in the form of blanks, tabs and new line characters, from the source program. Another is correlating error messages from the compiler with the source program i.e. keeping a correspondence between errors and source line numbers.

**Objectives**

1. Successful understanding/implementation of basic Lexical Analysis using flex

**Tools/Software Requirement**

1. flex on Linux or Windows platform

**Description**

Lexical analysis is the process of converting a sequence of characters into a sequence of [tokens](http://en.wikipedia.org/wiki/Lexical_analysis#Token). A program or function which performs lexical analysis is called a lexical analyzer, lexer or scanner. A lexer often exists as a single function which is called by a [parser](http://en.wikipedia.org/wiki/Parser) or another function.

**Lab 05 Task 1**

Write a flex program to process a pascal-like toy language with the following specifications:

* + Match integers and floating point constants
  + Match Identifiers, starting with lower-case alphabets and allowing for integers in non-starting locations.
  + Keywords: if, then, begin, end, procedure, function
  + Operators: +, -, \*, /
  + Skipping of white-space characters i.e. new-line, tabs and spaces
  + Printing of un-recognized characters

Use the following example code to test your lexical analyzer.

|  |
| --- |
| procedure compute  begin  area = 3.141 \* radius \* radius  end  function main  begin  compute  end |

Your output should resemble:

|  |
| --- |
| A keyword: procedure  An identifier: compute  A keyword: begin  An identifier: area  Unrecognized character: =  A float: 3.141 (3.141)  An operator: \*  An identifier: radius  An operator: \*  An identifier: radius  A keyword: end  A keyword: function  An identifier: main  A keyword: begin  An identifier: compute  A keyword: end |

**Lab 05 Task 2**

* **Postfix formula evaluation:** Given an input text containing non-negative integers and three operator i.e. +, - and \*, evaluate the given postfix formula using flex based lexical analyzer. For example given the following input:

**44 33 22 \* + 1 –**

Helpful link: <http://scriptasylum.com/tutorials/infix_postfix/algorithms/postfix-evaluation/>

Your output should resemble:

|  |
| --- |
| 44 0 0 0  33 44 0 0  22 33 44 0  726 44 0 0  770 0 0 0  1 770 0 0  769 0 0 0  result = 769 |

**Deliverables**

You are required to upload your task (Sources & PDF document) using the link created on LMS followed by a viva.

**Note:** Java user can perform the above tasks using jflex

<http://jflex.de/>

Use of jflex is optional and NOT compulsory.