Practical no: 5

Problem Statement: Write a program using Lex specifications to implement lexical analysis Phase of compiler to generate tokens of subset of Java program.

Name: Aditi Dinesh Mulay

Class: T.E. Computer

Subject: SPOS

Div: A

Roll no: 02

PRN No. 71918146B

SPOS

Assignment B-2

Aditi Dinesh Mulay
T.E. Comp Div: A
Roll no: 02

Aim: Design Lex program for to generate token of given input file.

Problem statement: Write a program using Lex specifications to implement lexical analysis phase of compiler to generate tokens of subset of Java Program.

Theory:

Lex stands for Lexical Analyzer. Lex is a tool for generating Scanners. Scanners are programs that recognize lexical patterns in text. These lexical patterns are defined in a particular syntax. A matched regular expression may have an associated action. This action may also include returning a token. When Lex receives input in the form of file or text, it takes i/p one character at a time & conditions until a pattern is matched, then lex performs the associated action. If on the other hand, no regular expression can be matched, further processing stops & Lex displays an error message. Lex and C are tightly coupled. A .lex file is passed through the lex utility, & produces o/p files in C. These files are coupled to produce an executable version of lexical analyzer.

Regular Expression in Lex:

A regular expression is a pattern discription using a Meta language. An expression is made up of symbols

Normal symbols are characters & numbers, but there are other symbols that have special meaning in Lex.

Programming in Lex:

It divided into 3 steps:

1) Specify the pattern-associated actions in a form that Lex can understand.
2) Run Lex over this file to generate C code for scanner.
3) Compile and link the C code to produce the executable scanner.

Regular expressions are used for pattern matching. A character class defines a single character & normal operators lose their meaning. Two operators supported in a character class are hyphen ("-") & circumflex ("^"). When used between two characters the hyphen represents a range of characters. The circumflex, when used as first character, negates the expression. If two patterns match the same string the longest match wins. In case both matches are the same length, then the first pattern listed is used

```
... definitions ...
%%
... rules ...
%%
... subrotines ...
```

Input to Lex is divided into three sections with %% dividing the sections. This is best illustrated by example. The first example is shortest possible lex: %% Input is copied to o/p one character at a time. The first %% is always required as there must always be a rule section. However if we don't specify any rules then the default action is to match everything & copy it to output. Defaults for i/p & o/p are stdin & stdout.

Conclusion:

Thus, we have studied lexical analyzer & implemented an application for lexical analyzer to perform scan the program & generates token of subset of java.