SSN COLLEGE OF ENGINEERING, KALAVAKKAM

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UCS1602 - Compiler Design

Programming Assignment 2

Implementation of Lexical Analyser and Symbol Table

Name: Jayannthan P T Dept: CSE 'A' Roll No.: 205001049

Programming Assignment-2 - Implementation of Lexical Analyzer using LEX tool (Java Programming Language: Lexical Construct)

Source code:

```
%{
#include<stdio.h>
#include<string.h>
%}


digit [0-9]
letter [a-zA-Z]
id ({letter}|"_")({letter}|{digit})*
arithop [\+\-\*\/\%]
logicop (\&\&|\|\||\!)
relop [<>!]=?
bitwise "|"|"&"|"^"|"~"|"<<"|">>"
commentstart  "/*"
commentend  "*/"
assnop "="
datatype ("boolean"|"byte"|"short"|"int"|"long"|"float"|"double"|"char"|"String"|"void")
access ("private"|"public"|"proteted")
keyword
("abstract"|"System"|"assert"|"boolean"|"break"|"byte"|"case"|"catch"|"char"|"class"|"cons
t"|"continue"|"default"|"do"|"double"|"else"|"enum"|"extends"|"final"|"finally"|"float"|"f
or"|"if"|"implements"|"import"|"instanceof"|"int"|"interface"|"long"|"native"|"new"|"packa
ge"|"return"|"short"|"static"|"strictfp"|"super"|"switch"|"synchronized"|"this"|"throw"|"t
hrows"|"transient"|"try"|"void"|"volatile"|"while")
funcstart
("System.out.println"|"System.out.print"|"Math."|"String."|"Integer."|"Double."|"Float."|"
Boolean."|"Character.")
%x COMMENT
spl [;\(\)\{\},]
functiondefn {id}\(.*\).*"{"
%%
import.*$           printf("Importing Package \t\t-->\t\t %s\n", yytext);
```

```
{keyword}       printf("Keyword \t\t-->\t\t %s\n", yytext);
{datatype}       printf("datatype \t\t-->\t\t %s\n", yytext);
{funcstart}.* printf("function call \t\t-->\t\t %s\n", yytext);
{functiondefn}.* {
                    printf("function defn \t\t-->\t\t");
                    int i=0;
                    while(yytext[i]!='{')
                    {
                        printf("%c",yytext[i]);
                        i++;
                    }
                    printf("\n");
                    printf("Spl char \t\t-->\t\t {\n");
                }

"/*"            { BEGIN(COMMENT); printf("Multiline comment \t\t-->\t\t %s", yytext);}
<COMMENT>"*/"   { BEGIN(INITIAL); printf("%s\n", yytext); }
{access}    printf("Access Specifiers \t\t-->\t\t %s\n", yytext);
"//"(.*)$      printf("Singleline Comment \t\t-->\t\t %s\n", yytext);
{digit}+      printf("Number \t\t\t-->\t\t %s\n", yytext);
{id}          printf("Identifier \t\t-->\t\t %s\n", yytext);
({id}"[".*"]")          printf("Identifier \t\t-->\t\t %s\n", yytext);
\.{id}      printf("Attribute call \t\t-->\t\t %s\n", yytext);
\.{id}\(\)      printf("Attribute function call -->\t\t %s\n", yytext);
{bitwise}          printf("bitwise operator \t\t-->\t\t %s\n", yytext);
{arithop} printf("Arithmetic operator \t-->\t\t %s\n", yytext);
{arithop}{assnop} printf("Arithmetic assignment operator \t-->\t\t %s\n", yytext);
{relop}       printf("Relational operator \t-->\t\t %s\n", yytext);
{assnop} printf("assignment operator \t-->\t\t %s\n", yytext);
{relop}{assnop}       printf("Relational assignment operator \t\t-->\t\t %s\n", yytext);
{assnop}{assnop}       printf("Relational operator \t\t-->\t\t %s\n", yytext);
{logicop}       printf("Logical operator \t\t-->\t\t %s\n", yytext);
{spl}           printf("Spl char \t\t-->\t\t %s\n", yytext);
" " { } ;           /* skip whitespace */
"\n" { } ;             /* skip newlines */
"\t" { } ;             /* skip tabs */

%%

int yywrap(void){}

int main(int argc, char* argv[]) {
    FILE *file;

    if (argc < 2) {
        fprintf(stderr, "Usage --> %s file\n", argv[0]);
        return 1;
    }

    file = fopen(argv[1], "r");
    if (file == NULL) {
        fprintf(stderr, "Error --> Unable to open file %s\n", argv[1]);
        return 1;
    }
```

```
    yyin = file;
    yylex();

    fclose(file);
    return 0;
}
```

Input Code:

```
import java.util.Scanner;
/*
Public
Class
Quicksort
*/
public class Quicksort {
    public static void sort(int[] arr, int low, int high) {
        if (low < high) {
            int pivot = partition(arr, low, high);
            sort(arr, low, pivot - 1);
            sort(arr, pivot + 1, high);
        }
    }

    private static int partition(int[] arr, int low, int high) {
        int pivot = arr[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (arr[j] < pivot) {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        int temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;
        return i + 1;
    }
    //main
    public static void main(String[] args) {

        int n;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements you want to store: ");

        n = sc.nextInt();

        int[] array = new int[n];
        System.out.println("Enter the elements of the array: ");
```

```
        for (int i = 0; i < n; i++) {

            array[i] = sc.nextInt();
        }

        sort(array, 0, array.length - 1);
        System.out.print("sorted: ");

        for (int i : array) {
            System.out.print(i + " ");
        }
    }
}
```

Output:

```
Importing Package       -->     import java.util.Scanner;
Multiline comment       -->     /*
Public
Class
Quicksort
*/

Access Specifiers       -->     public
Keyword         -->     class
Identifier      -->     Quicksort
Spl character       -->     {

Access Specifiers       -->     public
Keyword         -->     static
Keyword         -->     void
function defn       -->     sort(int[] arr, int low, int high) {
Keyword         -->     if
Spl char        -->     (
Identifier      -->     low
Relational operator     -->     <
Identifier      -->     high
Spl char        -->     )
Spl char        -->     {

Keyword         -->     int
Identifier      -->     pivot
assignment operator     -->     =
Identifier      -->     partition
Spl char        -->     (
Identifier      -->     arr
Spl char        -->     ,
Identifier      -->     low
Spl char        -->     ,
Identifier      -->     high
```

```
Spl char        -->       )
Spl char        -->       ;

Identifier      -->       sort
Spl char        -->       (
Identifier      -->       arr
Spl char        -->       ,
Identifier      -->       low
Spl char        -->       ,
Identifier      -->       pivot
Arithmetic operator     -->       -
Number          -->       1
Spl char        -->       )
Spl char        -->       ;

Identifier      -->       sort
Spl char        -->       (
Identifier      -->       arr
Spl char        -->       ,
Identifier      -->       pivot
Arithmetic operator     -->       +
Number          -->       1
Spl char        -->       ,
Identifier      -->       high
Spl char        -->       )
Spl char        -->       ;

Spl char        -->       }

Spl char        -->       }


Access Specifiers       -->       private
Keyword         -->       static
Keyword         -->       int
function defn       -->       partition(int[] arr, int low, int high) {
Keyword         -->       int
Identifier      -->       pivot
assignment operator     -->       =
Identifier      -->       arr[high]
Spl char        -->       ;

Keyword         -->       int
Identifier      -->       i
assignment operator     -->       =
Identifier      -->       low
Arithmetic operator     -->       -
Number          -->       1
Spl char        -->       ;

Keyword         -->       for
Spl char        -->       (
Keyword         -->       int
Identifier      -->       j
```

```
assignment operator     -->      =
Identifier      -->      low
Spl char        -->      ;
Identifier      -->      j
Relational operator     -->      <
Identifier      -->      high
Spl char        -->      ;
Identifier      -->      j
Arithmetic operator     -->      +
Arithmetic operator     -->      +
Spl char        -->      )
Spl char        -->      {

Keyword         -->      if
Spl char        -->      (
Identifier      -->      arr[j]
Relational operator     -->      <
Identifier      -->      pivot
Spl char        -->      )
Spl char        -->      {

Identifier      -->      i
Arithmetic operator     -->      +
Arithmetic operator     -->      +
Spl char        -->      ;

Keyword         -->      int
Identifier      -->      temp
assignment operator     -->      =
Identifier      -->      arr[i]
Spl char        -->      ;

Identifier      -->      arr[i] = arr[j]
Spl char        -->      ;

Identifier      -->      arr[j]
assignment operator     -->      =
Identifier      -->      temp
Spl char        -->      ;

Spl char        -->      }

Spl char        -->      }

Keyword         -->      int
Identifier      -->      temp
assignment operator     -->      =
Identifier      -->      arr[i + 1]
Spl char        -->      ;

Identifier      -->      arr[i + 1] = arr[high]
Spl char        -->      ;

Identifier      -->      arr[high]
```

```
assignment operator     -->      =
Identifier      -->      temp
Spl char        -->      ;

Keyword         -->      return
Identifier      -->      i
Arithmetic operator     -->      +
Number          -->      1
Spl char        -->      ;

Spl char        -->      }

Singleline Comment      -->      //main
Access Specifiers       -->      public
Keyword         -->      static
Keyword         -->      void
function defn       -->      main(String[] args) {

Keyword         -->      int
Identifier      -->      n
Spl char        -->      ;

Identifier      -->      Scanner
Identifier      -->      sc
assignment operator     -->      =
Keyword         -->      new
Identifier      -->      Scanner
Spl char        -->      (
Keyword         -->      System
Attribute call      -->      .in
Spl char        -->      )
Spl char        -->      ;

function call       -->      System.out.print("Enter the number of elements you want to
store: ");

Identifier      -->      n
assignment operator     -->      =
Identifier      -->      sc
Attribute function call -->      .nextInt()
Spl char        -->      ;


Identifier      -->      int[] array = new int[n]
Spl char        -->      ;

function call       -->      System.out.println("Enter the elements of the array: ");
Keyword         -->      for
Spl char        -->      (
Keyword         -->      int
Identifier      -->      i
assignment operator     -->      =
Number          -->      0
Spl char        -->      ;
```

```
Identifier      -->      i
Relational operator     -->      <
Identifier      -->      n
Spl char        -->      ;
Identifier      -->      i
Arithmetic operator     -->      +
Arithmetic operator     -->      +
Spl char        -->      )
Spl char        -->      {


Identifier      -->      array[i]
assignment operator     -->      =
Identifier      -->      sc
Attribute function call -->      .nextInt()
Spl char        -->      ;

Spl char        -->      }


Identifier      -->      sort
Spl char        -->      (
Identifier      -->      array
Spl char        -->      ,
Number          -->      0
Spl char        -->      ,
Identifier      -->      array
Attribute call      -->      .length
Arithmetic operator     -->      -
Number          -->      1
Spl char        -->      )
Spl char        -->      ;

function call       -->      System.out.print("sorted: ");

Keyword         -->      for
Spl char        -->      (
Keyword         -->      int
Identifier      -->      i
:Identifier         -->      array
Spl char        -->      )
Spl char        -->      {

function call       -->      System.out.print(i + " ");
Spl char        -->      }

Spl char        -->      }

Spl char        -->      }
```

Learning Outcome:

- Understood the working of lexical analyser for debugging of programs.

- Understood the role of lexical analyser in running a program
- Understood how to write lex programs