**Lexical Analyser**

**CODE**

import java.util.\*;

class Lexical{

public static void main(String args[]){

Scanner s=new Scanner(System.in);

String keywords[]={"break","case","char","default","float","for","goto","if","int"};

String operator[] = {"+","-","\*","/","%","="};

String Sp[]={",",";","[","]",":","#","\"","}","{","(",")"};

String func[]={"main","printf","scanf"};

//Vector St=new Vector(10,3);

//Vector Lit=new Vector(10,3);

String program=new String();

System.out.println("enter the C program");

program=s.nextLine();

StringTokenizer a=new StringTokenizer(program," \",+=-\*%/#><;(){}",true);

int n=a.countTokens();

String tk[]=new String[n];

int i=0,k;

while(a.hasMoreTokens())

{tk[i]=a.nextToken();

i++;

}

for(int j=0;j<n;j++)

{System.out.println(tk[j]);

}

int varlen=1,conslen=1;

Map<String,Integer> kw=new HashMap<String,Integer>();

Map<String,Integer> op=new HashMap<String,Integer>();

Map<String,Integer> ssc=new HashMap<String,Integer>();

Map<String,Integer> fun=new HashMap<String,Integer>();

Map<String,Integer> var=new HashMap<String,Integer>();

Map<String,Integer> cons=new HashMap<String,Integer>();

for(k=0;k<keywords.length;k++)

{

kw.put(keywords[k],k+1);

}

for(k=0;k<operator.length;k++)

{

op.put(operator[k],k+1);

}

for(k=0;k<Sp.length;k++)

{

ssc.put(Sp[k],k+1);

}

for(k=0;k<func.length;k++)

{

fun.put(func[k],k+1);

}

for(int j=0;j<n;j++)

{

if(tk[j]==null||tk[j].isEmpty()||tk[j].trim().isEmpty())

continue;

else if(tk[j].equals("\"")){

int x;

System.out.println(tk[j]+" :Special Symbol "+ssc.get(tk[j]));

String lp=new String();

for(x=j+1;x<tk.length;x++)

{if(tk[x].equals("\""))

break;

else{

lp=lp.concat(tk[x]);

}

}

cons.put(lp,conslen);

conslen++;

System.out.println(lp+" :Literal "+cons.get(lp));

System.out.println(tk[x]+" :Special Symbol "+ssc.get(tk[x]));

j=x;

}

else

{

if(kw.containsKey(tk[j]))

{

System.out.println(tk[j]+" :Keyword "+kw.get(tk[j]));

}

else if(op.containsKey(tk[j]))

{

System.out.println(tk[j]+" :Operator "+op.get(tk[j]));

}

else if(ssc.containsKey(tk[j]))

{

System.out.println(tk[j]+" :Special Symbol "+ssc.get(tk[j]));

}

else if(fun.containsKey(tk[j]))

{

System.out.println(tk[j]+" :Function "+fun.get(tk[j]));

}

else if(tk[j].matches("[a-zA-Z\_$][a-zA-Z\_$0-9]\*$"))

{if(var.containsKey(tk[j]))

System.out.println(tk[j]+" :Variable "+var.get(tk[j]));

else{

var.put(tk[j],varlen);

varlen++;

System.out.println(tk[j]+" :Variable "+var.get(tk[j]));

}

}

else if(tk[j].matches("[0-9]\*")){

if(cons.containsKey(tk[j]))

System.out.println(tk[j]+" :Literal "+cons.get(tk[j]));

else

{

cons.put(tk[j],conslen);

conslen++;

System.out.println(tk[j]+" :Literal "+cons.get(tk[j]));

}

}

}

}

}

}

**OUTPUT:**



