**Practical No 4.**

**Program :**

#include<stdio.h>

#include<conio.h>

#include<ctype.h>

void main()

{

char str[30];

int state=0,i=0;

printf("\n\t\*\*\*\*\*\* LEXICAL ANALYZER \*\*\*\*\*\*\n\t");

printf("\n\n\tPlease enter any string:\n\t");

scanf("%s", str);

while(str[i]!='\0')

{

switch(state)

{

case 0: if(isalpha(str[i]))

{

i++;

}

else if(isdigit(str[i]))

{

i++;

}

else if(str[i]=='+')

{

state=1;

}

else if(str[i]=='-')

{

state=3;

}

else if(str[i]=='=')

{

state=5;

}

else if(str[i]=='<')

{

state=7;

}

else if(str[i]=='>')

{

state=8;

}

else if(str[i]=='}')

{

state=9;

}

else if(str[i]=='(')

{

state=10;

}

else if(str[i]==')')

{

state=11;

}

else if(str[i]=='[')

{

state=12;

}

else if(str[i]==']')

{

state=13;

}

else if(str[i]==';')

{

state=14;

}

else if(str[i]=='/')

{

state=15;

}

else if(str[i]=='\*')

{

state=16;

}

else if(str[i]=='{')

{

state=17;

}

else if(str[i]==' ')

{

i++;

}

break;

case 1: i++;

if(str[i]=='+')

{

state=2;

}

else

{

i--;

printf("\n\t +: is an arithmatic operator\n");

state=0;

i=i+1;

}

break;

case 2: printf("\n\t++: is an increment operator\n");

state=0;

i++;

break;

case 3:i++;

if(str[i]=='-')

{

state=4;

}

else

{

i--;

printf("\n\t-: is an arithmatic operator\n");

state=0;

i=i+1;

}

break;

case 4:printf("\n\t--: is a decrement operator\n");

state=0;

i++;

break;

case 5:i++;

if(str[i]=='=')

{

state=6;

}

else if((str[i-2]=='<')||(str[i-2]=='>'))

{

state=6;

}

else

{

i--;

printf("\n\t=: is an assignment operator\n");

state=0;

i=i+1;

}

break;

case 6: printf("\n\t==: is an equality relation operator\n");

state=0;

i++;

break;

case 7: printf("\n\t<: is a 'less than' relation operator\n",str[i]);

state=0;

i++;

break;

case 8: printf("\n\t>: is a 'greater than' relation operator\n");

state=0;

i++;

break;

case 9: printf("\n\t}: is a closing curly bracket\n");

state=0;

i++;

break;

case 10:printf("\n\t(: is an opening round bracket\n");

state=0;

i++;

break;

case 11:printf("\n\t): is a closing round bracket\n");

state=0;

i++;

break;

case 12:printf("\n\t[: is an opening square bracket\n");

state=0;

i++;

break;

case 13:printf("\n\t]: is a closing square bracket\n");

state=0;

i++;

break;

case 14:printf("\n\t;: is a end-of-statement operator\n");

state=0;

i++;

break;

case 15:printf("\n\t/: is a forward slash\n");

state=0;

i++;

break;

case 16:printf("\n\t\*: is a multiplication operator\n");

state=0;

i++;

break;

case 17:printf("\n\t{: is an opening curly bracket\n");

state=0;

i++;

break;

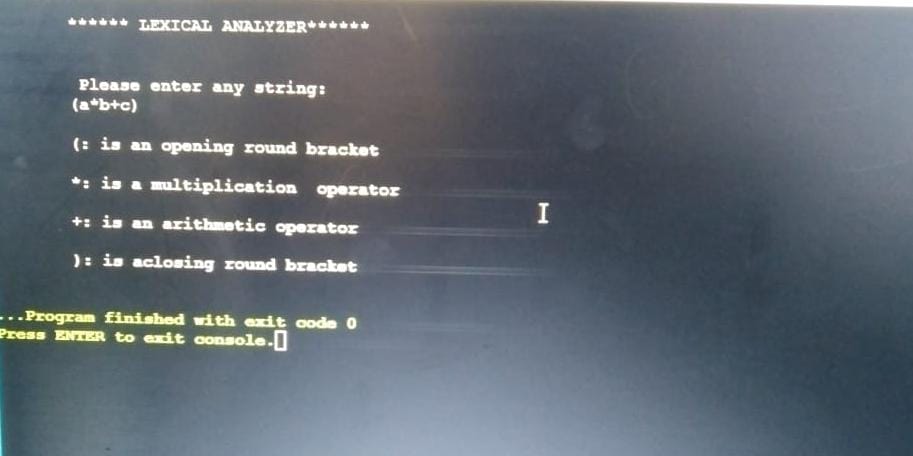
}

}

getch();

}

**Output :**

****