IMPLEMENTATION OF LEXICAL ANALYZER

NAME: **RISHAL RAMESH** ***EXP NO:*** **1**

REG NO: **RA1911030010084**

DATE: **10/01/2022**

AIM:

To study and code a lexical analyser in any of the programming languages.

LANGUAGE USED:

C++

ALGORITHM:

- Start
- Get the input program from the file prog.txt.
- Read the program line by line and check if each word in a line is a keyword, identifier, constant or an operator.
- If the word read is an identifier, assign a number to the identifier and make an entry into the symbol table stored in sybol.txt.
- For each lexeme read, generate a token as follows:
    a) If the lexeme is an identifier, then the token generated is of the form <id, number>
    b) If the lexeme is an operator, then the token generated is <op, operator>.
    c) If the lexeme is a constant, then the token generated is <const, value>.
    d) If the lexeme is a keyword, then the token is the keyword itself.
- The stream of tokens generated are displayed in the console output.
- Stop.

CODE:

```
#include<bits/stdc++.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

using namespace std;

int isKeyword(char buffer[])

{

    char keywords[32][10] =

    {"auto","break","case","char","const","continue","default",

                            "do","double","else","enum","extern","float","for","goto",

                            "if","int","long","register","return","short","signed",

                            "sizeof","static","struct","switch","typedef","union",

                            "unsigned","void","volatile","while"};

    int i, flag = 0;

    for(i = 0; i < 32; ++i)

    {

        if(strcmp(keywords[i], buffer) == 0)

        {

            flag = 1;

            break;

        }

    }

    return flag;

}

int main()

{

    char ch, buffer[15],b[30], logical_op[] = "><",math_op[]="+-
*/=",numer[]=".0123456789",other[]=",;\(){}[]'':";

    ifstream fin("Program.txt");

    int mark[1000]={0};

    int i,j=0,kc=0,ic=0,lc=0,mc=0,nc=0,oc=0,aaa=0;
```

```
vector < string > k;
vector<char >id;
vector<char>lo;
vector<char>ma;
vector<string>nu;
 vector<char>ot;
if(!fin.is_open())
{
    cout<<"error while opening the file\n";
    exit(0);
}
while(!fin.eof())
{
    ch = fin.get();
    for(i = 0; i < 12; ++i)
    {
        if(ch == other[i])
        {
            int aa=ch;
            if(mark[aa]!=1)
            {
                ot.push_back(ch);
                mark[aa]=1;
                ++oc;
            }
        }
    }
    for(i = 0; i < 5; ++i)
    {
        if(ch == math_op[i])
        {
            int aa=ch;
```

```
                if(mark[aa]!=1)

                {

                    ma.push_back(ch);

                    mark[aa]=1;

                    ++mc;

                }

            }

        }

        for(i = 0; i < 2; ++i)

        {

            if(ch == logical_op[i])

            {

                int aa=ch;

                if(mark[aa]!=1)

                {

                    lo.push_back(ch);

                    mark[aa]=1;

                    ++lc;

                }

            }

        }

        if(ch=='0' || ch=='1' || ch=='2' || ch=='3' || ch=='4' || ch=='5' || ch=='6' || ch=='7' || ch=='8' ||
ch=='9' || ch=='.' ||ch == ' ' || ch == '\n' || ch == ';')

        {

            if(ch=='0' || ch=='1' || ch=='2' || ch=='3' || ch=='4' || ch=='5' || ch=='6' || ch=='7' ||
ch=='8' || ch=='9' || ch=='.')

              b[aaa++]=ch;

            if((ch == ' ' || ch == '\n' || ch == ';') && (aaa != 0))

            {

                b[aaa] = '\0';

                aaa = 0;

                char arr[30];

                strcpy(arr,b);
            nu.push_back(arr);
            ++nc;
        }
    }
    if(isalnum(ch))
        buffer[j++] = ch;
    else if((ch == ' ' || ch == '\n') && (j != 0))
    {
        buffer[j] = '\0';
        j = 0;
        if(isKeyword(buffer) == 1)
        {
            k.push_back(buffer);
            ++kc;
        }
        else
        {
            if(buffer[0]>=97 && buffer[0]<=122)
            {
                if(mark[buffer[0]-'a']!=1)
                {
                    id.push_back(buffer[0]);
                    ++ic;
                    mark[buffer[0]-'a']=1;
                }
            }
        }
    }
}
fin.close();
printf("Keywords: ");
for(int f=0;f<kc;++f)
```

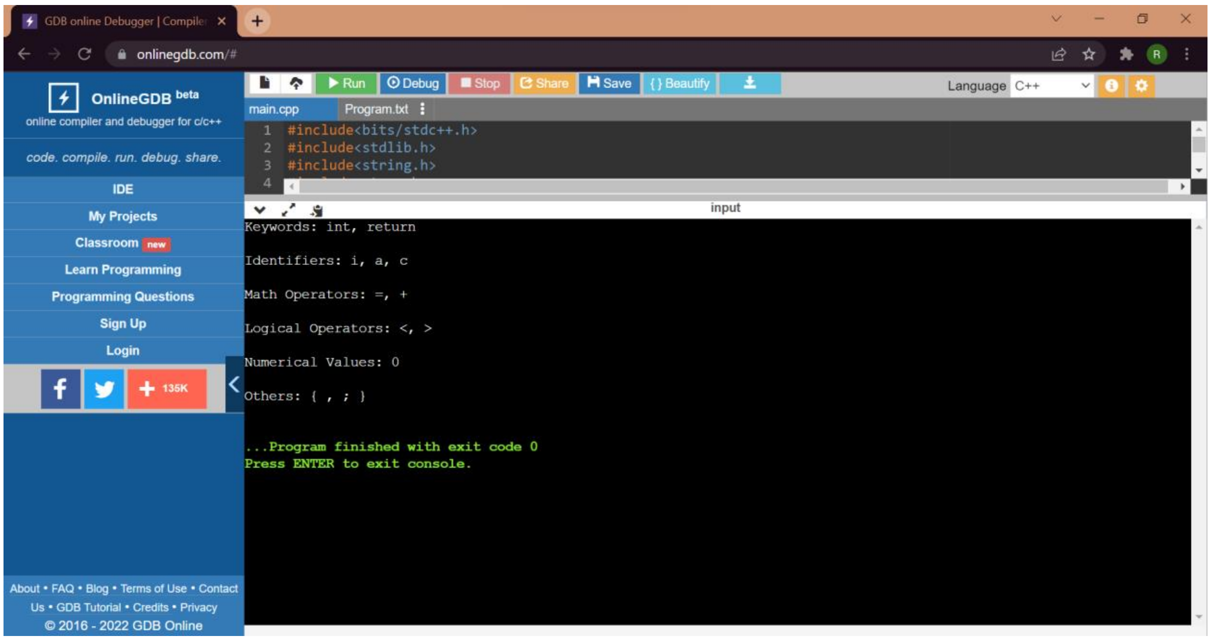
```
{
    if(f==kc-1)
      cout<<k[f]<<"\n";
    else
      cout<<k[f]<<", ";
}
printf("\nIdentifiers: ");
for(int f=0;f<ic;++f)
{
    if(f==ic-1)
        cout<<id[f]<<"\n";
    else
        cout<<id[f]<<", ";
}
printf("\nMath Operators: ");
for(int f=0;f<mc;++f)
{
    if(f==mc-1)
        cout<<ma[f]<<"\n";
    else
        cout<<ma[f]<<", ";
}
printf("\nLogical Operators: ");
for(int f=0;f<lc;++f)
{
    if(f==lc-1)
        cout<<lo[f]<<"\n";
    else
        cout<<lo[f]<<", ";
}
printf("\nNumerical Values: ");
for(int f=0;f<nc;++f)
```

```
    {
        if(f==nc-1)
            cout<<nu[f]<<"\n";
        else
            cout<<nu[f]<<", ";
    }
    printf("\nOthers: ");
    for(int f=0;f<oc;++f)
    {
        if(f==oc-1)
            cout<<ot[f]<<"\n";
        else
            cout<<ot[f]<<" ";
    }
    return 0;
}
```

PROGRAM.TXT:

```
#include<iostream>
{
  int a,b,c;
  cin>>b>>c;
  a=b+c;
  cout<<a;
  return 0;
}
```

OUTPUT:



```
#include<bits/stdc++.h>
#include<stdlib.h>
#include<string.h>
```

```
Keywords: int, return

Identifiers: i, a, c

Math Operators: =, +

Logical Operators: <, >

Numerical Values: 0

Others: ( , ; )


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

Lexical Analyser was studied and executed successfully in C++.