Anas Ahmed Ather

RA2011031010006

LEXICAL ANALYZER

EX. NO. 1

Date- 24/01/2023

AIM: To write a program to implement a lexical analyzer.

ALGORITHM:

1. Start.
2. Get the input program from the file prog.txt.
3. Read the program line by line and check if each word in a line is a keyword, identifier, constant or an operator.
4. 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.
5. 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.

6. The stream of tokens generated are displayed in the console output.
7. Stop.

PROGRAM:

```
#include<iostream>
#include<cstring>
#include<stdlib.h>
#include<ctype.h>
#include<fstream>
using namespace std;

string arr[] = { "void", "using", "namespace", "int", "include", "iostream", "std", "main",
"cin", "cout", "return", "float", "double", "string" };

bool
```

```
isKeyword (string a)
{
  for (int i = 0; i < 14; i++)
    {
      if (arr[i] == a)
            {
              return true;
            }
    }
  return false;
}

int main()
{

            fstream file;
            string s, filename;

            filename = "./add.c";

            file.open(filename.c_str());

            while (file >> s)
            {
        if (s == "+" || s == "-" || s == "" || s == "/" || s == "^" || s == "&&" || s == "||" || s == "=" || s ==
"==" || s == "&" || s == "|" || s == "%" || s == "++" || s == "--" || s == "+=" || s == "-=" || s == "/=" ||
s == "=" || s == "%=")
                {
                  cout << s << " is an operator\n";
                  s = "";
                }
              else if (isKeyword (s))
                {
                  cout << s << " is a keyword\n";
                  s = "";
                }
              else if (s == "(" || s == "{" || s == "[" || s == ")" || s == "}" || s == "]" || s == "<" || s == ">"
|| s == "()" || s == ";" || s == "<<" || s == ">>" || s == "," || s == "#")
                {
                  cout << s << " is a symbol\n";
                  s = "";

                }
              else if (s == "\n" || s == " " || s == "")
                {
                  s = "";
                }
              else if (isdigit (s[0]))
                {
                  int x = 0;
                  if (!isdigit (s[x++]))
                        {
                          continue;
                        }
                  else
                        {
                          cout << s << " is a constant\n";
                          s = "";
                        }
                }
              else
                {
                  cout << s << " is an identifier\n";
                  s = "";
                }
            }

            return 0;
}
```

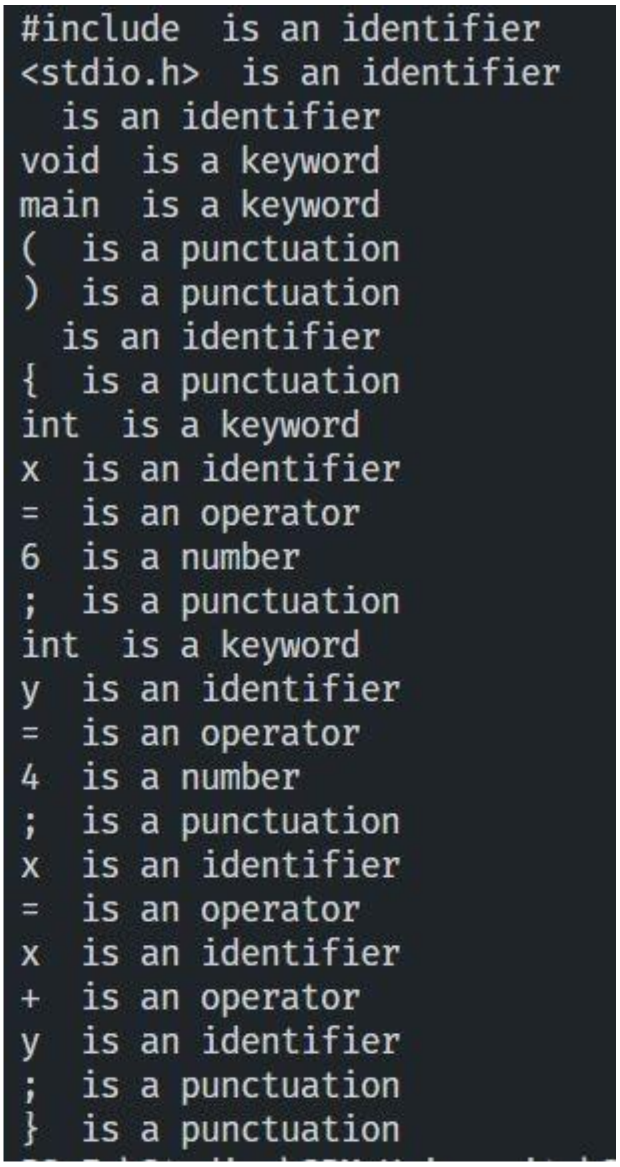
INPUT :

```
#include <stdio.h>

void main ( )

{
   int x = 6 ;
   int y = 4 ;
   x = x + y ;
}
```

OUTPUT :



```
#include  is an identifier
<stdio.h>  is an identifier
  is an identifier
void  is a keyword
main  is a keyword
(  is a punctuation
)  is a punctuation
  is an identifier
{  is a punctuation
int  is a keyword
x  is an identifier
=  is an operator
6  is a number
;  is a punctuation
int  is a keyword
y  is an identifier
=  is an operator
4  is a number
;  is a punctuation
x  is an identifier
=  is an operator
x  is an identifier
+  is an operator
y  is an identifier
;  is a punctuation
}  is a punctuation
```

RESULT :

The implementation of lexical analyser in C++ was compiled, executed andverified successfully.