Compiler Design Lab

NAME: Priyanka Srinivas
REGISTER NUMBER: RA1911026010014
SECTION: K1

Exercise 1B: Implementation of Lexical Analyzer

AIM: To write a program to implement a lexical analyzer in C++

ALGORITHM:

1. Start the program.
2. Take the input sentence from the textfile prog.txt.
3. Read the program line-by-line to check if each word in the input sentence is a keyword, constant, operator, valid and invalid identifier.
4. For each lexeme read, generate tokens to store them into the symbol table in the text file symbol.txt
   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.
5. Display the stream of tokens generated in the console as output so that the different lexemes are identified properly.
6. Stop the program.

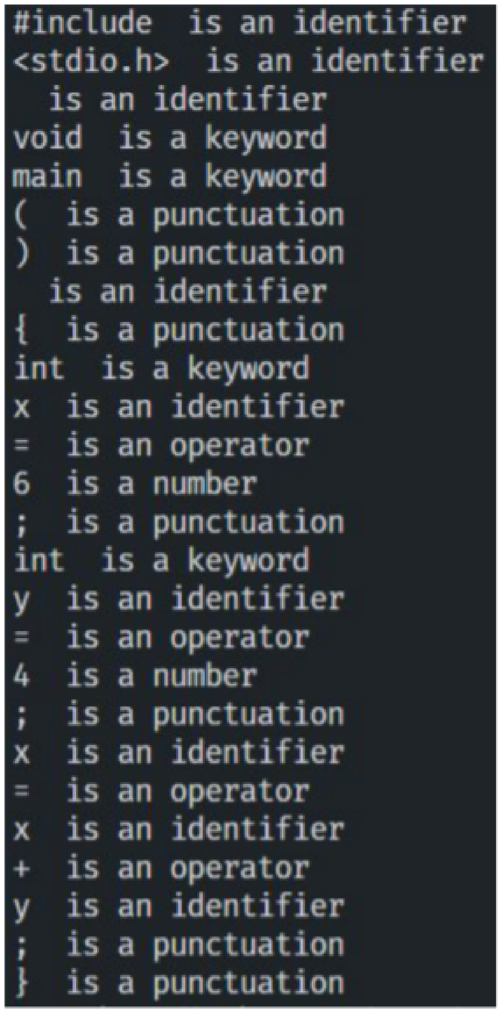
PROGRAM:

```
#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;
    }
```

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:

A lexical analyzer in C++ was compiled, executed and verified successfully.