EXP NO: 1 **LEXICAL ANALYZER**

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:

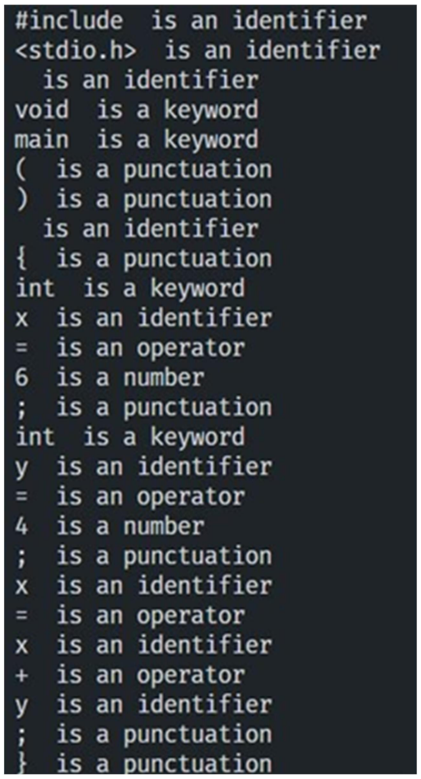
```
file = open("add.c", 'r')
lines = file.readlines()
keywords    = ["void", "main", "int", "float", "bool", "if", "for", "else", "while", "char", "return"]
operators = ["=", "==", "+", "-", "*", "/", "++", "--", "+=", "-=", "!=", "||", "&&"]
punctuations= [";", "(", ")", "{", "}", "[", "]"]
def is_int(x):
    try:
        int(x)
        return True
    except:
        return False
```

```
for line in lines:
    for i in line.strip().split(" "):
        if i in keywords:
            print (i, " is a keyword")
        elif i in operators:
            print (i, " is an operator")
        elif i in punctuations:
            print (i, " is a punctuation")
        elif is_int(i):
            print (i, " is a number")
        else:
            print (i, " is an identifier")
```

INPUT :

```
#include <stdio.h>
void main ( )
{
   int x = 6 ;
   int y = 4 ;
   x = x + y ;
   printf("%d", x);
}
```

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