LEXICAL ANALYZER

NAME: ATHRESH KUMAR LABDE
(RA1911033010146)
M1 CSE (S.E)

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
    b. If the lexeme is an operator, then the token generated is .
    c. If the lexeme is a constant, then the token generated is .
    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")
```

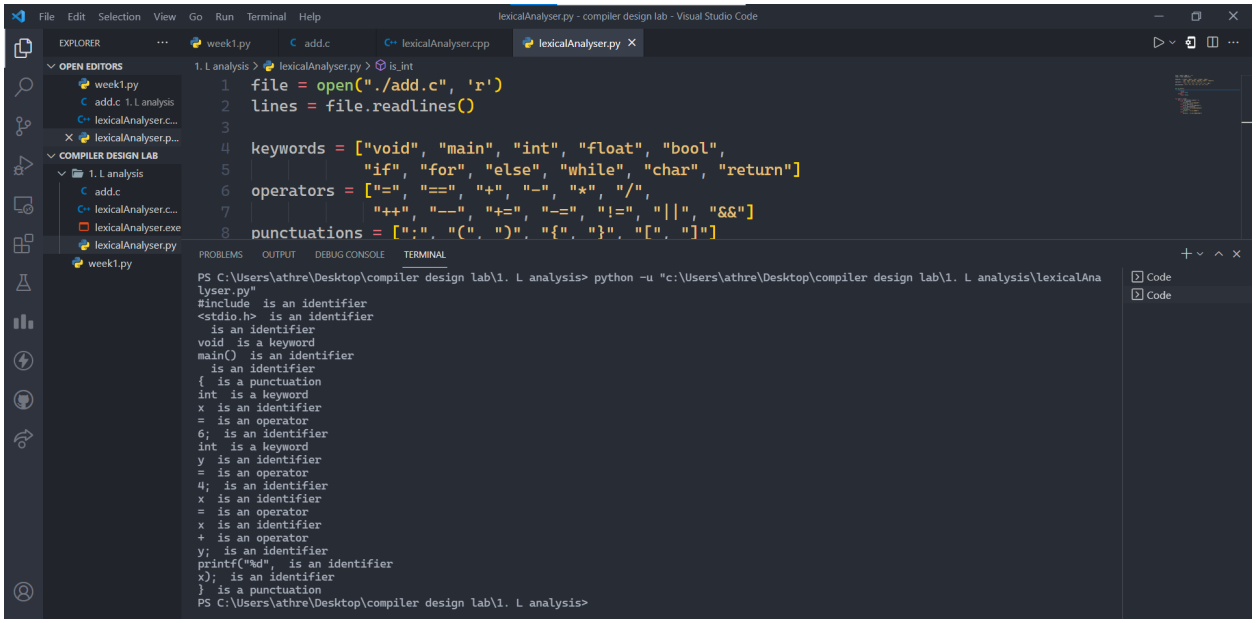
INPUT :

```
#include <stdio.h>

void main ( )


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

Output:



RESULT :

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