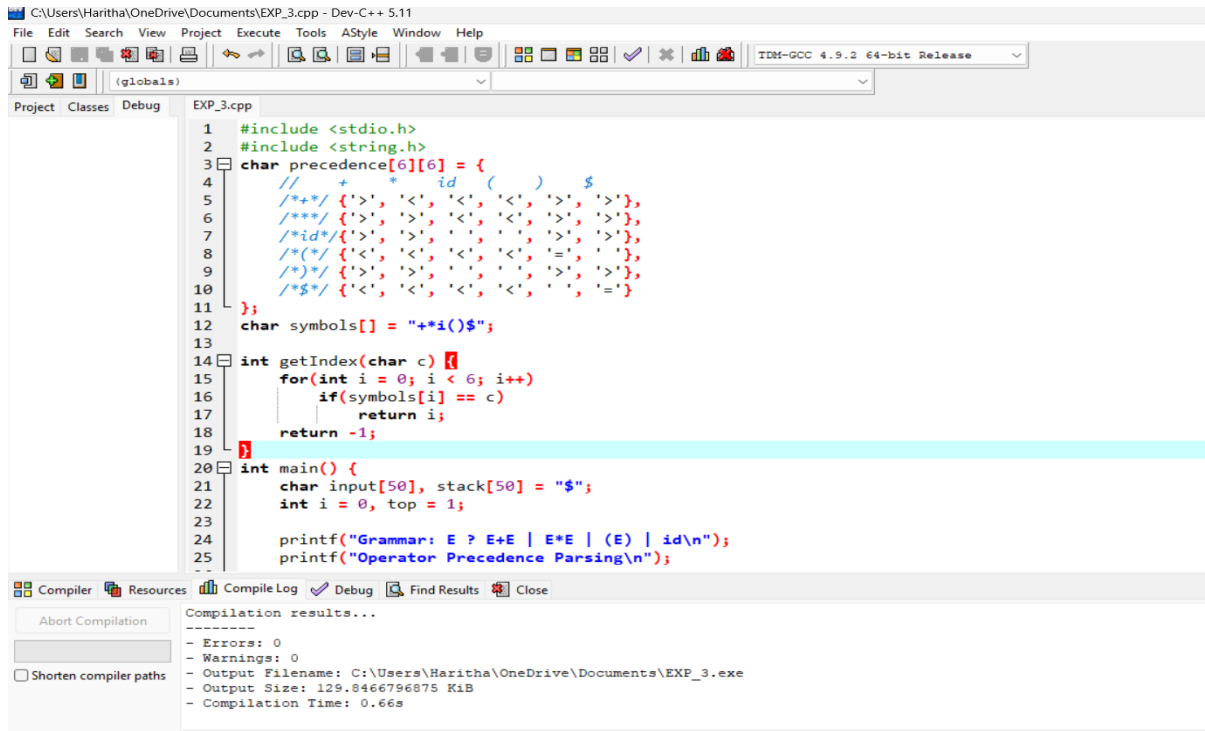


# EXPERIMENT-15

## AIM:

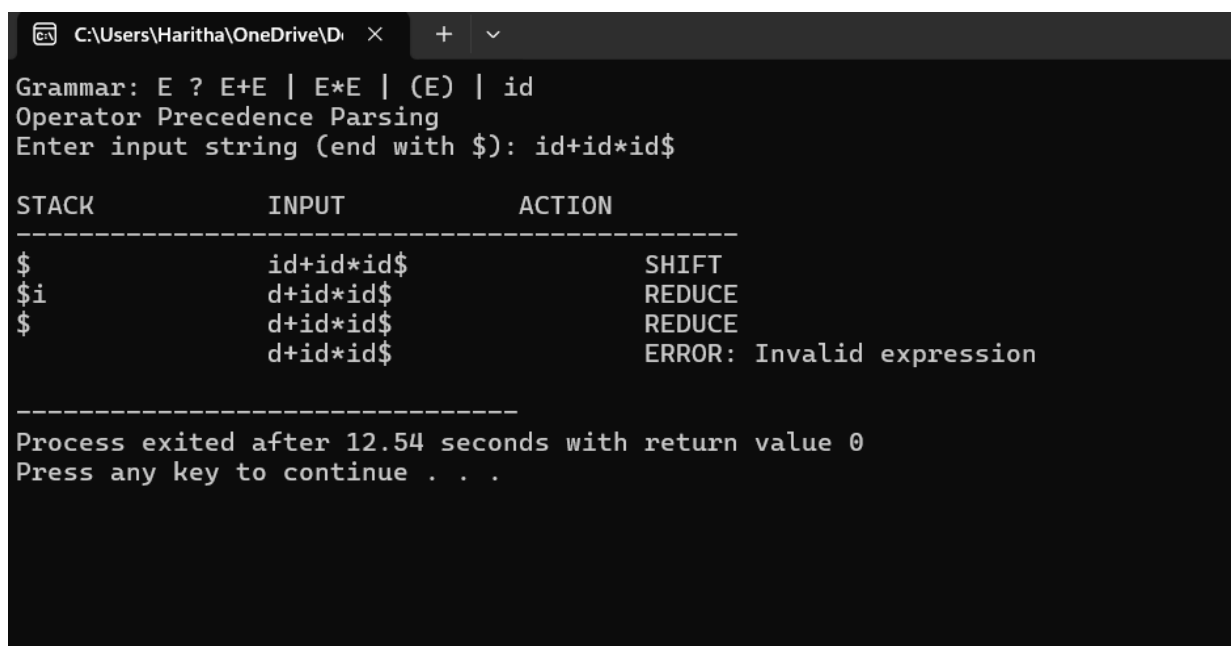
Write a C Program to implement the operator precedence parsing.

## PROGRAM:



```
1 #include <stdio.h>
2 #include <string.h>
3 char precedence[6][6] = {
4     // + * id ( ) $
5     /*+*/ {'>', '<', '<', '<', '<', '>'},
6     /**/ {'>', '>', '<', '<', '>', '>'},
7     /*id*/ {'>', '>', '>', '<', '>', '>'},
8     /*(*/ {'<', '<', '<', '<', '=', '>'},
9     /*)*/ {'>', '>', '>', '>', '>', '>'},
10    /*$*/ {'<', '<', '<', '<', '<', '='},
11 };
12 char symbols[] = "+*i()$";
13
14 int getIndex(char c) {
15     for(int i = 0; i < 6; i++)
16         if(symbols[i] == c)
17             return i;
18     return -1;
19 }
20
21 int main() {
22     char input[50], stack[50] = "$";
23     int i = 0, top = 1;
24
25     printf("Grammar: E ? E+E | E*E | (E) | id\n");
26     printf("Operator Precedence Parsing\n");
27
28     // Compiler output:
29     // Compilation results...
30     // - Errors: 0
31     // - Warnings: 0
32     // - Output Filename: C:\Users\Haritha\OneDrive\Documents\EXP_3.exe
33     // - Output Size: 129.8466796875 KiB
34     // - Compilation Time: 0.66s
```

## OUTPUT:



```
C:\Users\Haritha\OneDrive\Documents\EXP_3.exe
Grammar: E ? E+E | E*E | (E) | id
Operator Precedence Parsing
Enter input string (end with $): id+id*id$

STACK      INPUT      ACTION
-----
$          id+id*id$  SHIFT
$i         d+id*id$  REDUCE
$          d+id*id$  REDUCE
$          d+id*id$  ERROR: Invalid expression

-----
Process exited after 12.54 seconds with return value 0
Press any key to continue . . .
```