# EXPERIMENT 7

- ASHISH KUMAR
- 2K18/SE/041

**AIM:-** Write a program to find Cyclomatic complexity of a program(triangle classification).

**THEORY:-** There are many paths in any program. If there are loops in a program, the number of paths increases drastically. In such situations, we may be traversing the same nodes and edges again and again. However, as defined earlier, an independent path should have at least one new node or edge which is to be traversed. We should identify every independent path of a program and pay special attention to these paths during testing. A few concepts of graph theory are used in testing techniques which may help us to identify independent paths. This concept involves using cyclomatic number of graph theory which has been redefined as cyclomatic complexity. This is nothing but the number of independent paths through a program.

We have three methods to calculate cyclomatic complexity:

i) $V(G) = e - n + 2P$
where V(G) = Cyclomatic complexity
G : program graph
n : number of nodes
e : number of edges
P : number of connected components

ii) Cyclomatic complexity is equal to the number of regions of the program graph.

iii) Cyclomatic complexity
V(G) =pi+ 1 (where pi is no. Of predicate nodes contained in the program graph)

**CODE:-**

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

int CyclomaticComlexity(int e, int n, int P)
{
        int V = e - n + 2 * P;
        return V;
}

int main()
{
        int cnt = 0;
        ifstream myfile;
        string line;
        myfile.open("triangle.cpp");
        if (myfile.is_open()) {
          int e = 0, n = 0, p = 1;
          bool isStartProg;
          size_t found;
```

```
while (getline(myfile, line)) {


        if (line.empty())
                continue;
        if (!isStartProg) {
                std::cout<<line<<endl;
                found = line.find("void main()");
                if (found != string::npos) {
                        isStartProg = true;
                        e++;
                        n++;
                }
        }
        else {
                if(line.find_first_of("//") != string::npos){
                        cout << line << endl;
                        continue;
        }
                if(line.find("if") != string::npos)
                {
                cnt++;
                }
                e++;
```

```
                n++;
            cout << line << endl;
            }
        }
        myfile.close();
        int no_edges = e+cnt-1;
    cout << " \nCyclomatic Complexity: V(G) = E - N + 2*P = " << no_edges << " - " << n << " + 2 * " << p << " = " << CyclomaticComlexity(no_edges, n, p) << endl;
    }
    else {
      cout << "Unable to open file";
    }
    return 0;
}
```

## OUTPUT:-

C:\Users\Ashish\Desktop\CC.exe

```
/*Program to classify whether a triangle is acute, obtuse or right angled given the sides of
the triangle*/
//Header Files
#include<stdio.h>
#include<conio.h>
#include<math.h>
1. void main()
2. {
3. double a,b,c;
4. double a1,a2,a3;
5. int valid=0;
6. clrscr();
7. printf("Enter first side of the triangle:"); /*Enter the sides of Triangle*/
8. scanf("%lf",&a);
9. printf("Enter second side of the triangle:");
10. scanf("%lf",&b);
11. printf("Enter third side of the triangle:");
12. scanf("%lf",&c);
/*Checks whether a triangle is valid or not*/
13. if(a>0&&a<=100&&b>0&&b<=100&&c>0&&c<=100) {
14. if((a+b)>c&&(b+c)>a&&(c+a)>b) {
15. valid=1;
16. }
17. else {
18. valid=-1;
19. }
20. }
21. if(valid==1) {
22. a1=(a*a+b*b)/(c*c);
23. a2=(b*b+c*c)/(a*a);
24. a3=(c*c+a*a)/(b*b);
25. if(a1<1||a2<1||a3<1) {
26. printf("Obtuse angled triangle");
27. }
28. else if(a1==1||a2==1||a3==1) {
29. printf("Right angled triangle");
30. }
31. else {
32. printf("Acute angled triangle");
33. }
34. }
35. else if(valid==-1) {
36. printf("\nInvalid Triangle");
37. }
38. else {
39. printf("\nInput Values are Out of Range");
40. }
41. getch();
42. } //Main Ends

Cyclomatic Complexity: V(G) = E - N + 2*P = 42 - 37 + 2 * 1 = 7
```