Computer Programming Paradigm Lab
Lab Experiment No 01

Name: Aamir Ansari **Batch** A **Roll no.** 01

Aim: Write a C++ program to implement Triangle class which has the following members - three sides, four constructors (one with no parameter, one with single parameter (equilateral triangle), one with two parameters (isosceles triangle), one with three parameters (scalene triangle), a destructor, methods to read data and display data along with area of respective triangles.'

Theory:

1. Advantages of encapsulation

i) Data Hiding: Encapsulation protects an object from unwanted access by clients. Encapsulation allows access to a level without revealing the complex details below that level.

ii) Flexibility: With this, we can make the data as read-only or write-only as we require it to be. It also improves the maintainability and flexibility of code

iii) Reusability: It allows the user to use the existing code again and again in an effective way

iv) Testing of the code: Ease of testing, so it is better for Unit testing

2. Differentiate between object and class.

Class	Object
A class is a template for creating objects in program.	The object is an instance of a class.
A class is a logical entity	Object is a physical entity
A class does not allocate memory space when it is created.	Object allocates memory space whenever they are created.
You can declare class only once.	You can create more than one object using a class.

3. **Constructors, and its types**

A constructor is a member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object(instance of class) create. It is special member function of the class.

Types of Constructor:

1. Default constructor: It is the constructor which doesn't take any argument. It has no parameters.
2. Parameterized Constructors: It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor's body, use the parameters to initialize the object.

4. **When are constructors invoked? How are they different from functions?**

Each time an instance of a class is created the constructor method is called. Constructors is a special member function of class and it is used to initialize the objects of its class. It is treated as a special member function because its name is the same as the class name. These constructors get invoked whenever an object of its associated class is created. It is named as "constructor" because it constructs the value of data member of a class. Initial values can be passed as arguments to the constructor function when the object is declared.

A constructor is different from normal functions in following ways:

1. Constructor has same name as the class itself
2. Constructors don't have return type
3. A constructor is automatically called when an object is created.
4. If we do not specify a constructor, C++ compiler generates a default constructor for us (expects no parameters and has an empty body).

5. **Explain the concept of constructor overloading.**

As we know function overloading is one of the core feature of the object oriented languages. We can use the same name of the functions; whose parameter sets are different. Here we will see how to overload the constructors of C++ classes. The constructor overloading has few important concepts.

Overloaded constructors must have the same name and different number of arguments
The constructor is called based on the number and types of the arguments are passed.
We have to pass the argument while creating objects, otherwise the constructor cannot understand which constructor will be called.

Program:

```
//code
#include <iostream>
#include <cmath>

using namespace std;

class Triangle {
   public:
      //Class members
      float s1, s2, s3;

      //Non-Parameterised
      Triangle() {
         float area = 0;
         s1 = 0;
         s2 = 0;
         s3 = 0;
      }

      //One parameter, EQUILATERAL
      Triangle(float side) {
         float area;
         s1 = side;
         s2 = side;
         s3 = side;
      }

      //Two parameters, ISOSCELES
      Triangle(float side1, float side2) {
         s1 = side1;
         s2 = side1;
         s3 = side2;
      }

      //Three parameters, SCALENE
      Triangle(float side1, float side2, float side3) {
         s1 = side1;
         s2 = side2;
         s3 = side3;
      }

      //Display side and area
      void area() {
         float a, s;
```

```
        s = 0.5*(s1+s2+s3);
        a = sqrt((s*(s-s1)*(s-s2)*(s-s3)));
        cout << "Area of triangle with sides  " << s1 << ", " << s2 << ", " << s3 << " is :   "
<< a << endl;
    }

    //destructor
    ~Triangle(){};

};

int main(){

   int n=0;
   float tempSides[3];
   cout << "Enter number of known sides :  " << endl;
   cout << "*0 Zero " << endl;
   cout << "*1 One Side (EQUILATERAL)" << endl;
   cout << "*2 Two Sides (ISOSCELES)" << endl;
   cout << "*3 Three Sides (SCALENE)" << endl;
   cin >> n;

   //input value of sides
   for (int i=0 ; i<n ; i++) {
      cout << "Enter side number " << i+1 << " :  ";
      cin >> tempSides[i];
   }

   //Send to different constructors
   switch(n) {
      case 0: {   //zero
         Triangle myObj;
         myObj.area();
         break;
      }
      case 1: {   //EQUILATERAL
         Triangle myObj(tempSides[0]);
         myObj.area();
         break;
      }
      case 2: {   //ISOSCELES
         Triangle myObj(tempSides[0], tempSides[1]);
         myObj.area();
         break;
      }
      case 3: {   //SCALENE
         Triangle myObj(tempSides[0], tempSides[1], tempSides[2]);
         myObj.area();
         break;
      }
      default:
```

```
            cout << "Invalid Input" << endl;
        }

        return 0;
    }
```

//output

```
Enter number of known sides :
*0 Zero
*1 One Side (EQUILATERAL)
*2 Two Sides (ISOSCELES)
*3 Three Sides (SCALENE)
0
Area of triangle with sides  0, 0, 0 is :   0



Enter number of known sides :
*0 Zero
*1 One Side (EQUILATERAL)
*2 Two Sides (ISOSCELES)
*3 Three Sides (SCALENE)
1
Enter side number 1 :  12
Area of triangle with sides  12, 12, 12 is :   62.3538


 Enter number of known sides :
 *0 Zero
 *1 One Side (EQUILATERAL)
 *2 Two Sides (ISOSCELES)
 *3 Three Sides (SCALENE)
 2
 Enter side number 1 :  10
 Enter side number 2 :  5
 Area of triangle with sides  10, 10, 5 is :   24.2061

Enter number of known sides :
*0 Zero
*1 One Side (EQUILATERAL)
*2 Two Sides (ISOSCELES)
*3 Three Sides (SCALENE)
3
Enter side number 1 :  4
Enter side number 2 :  7
Enter side number 3 :  6
Area of triangle with sides  4, 7, 6 is :   11.9765

Process returned 0 (0x0)   execution time : 15.118 s
Press any key to continue.
```