

Himalaya College of Engineering

**Advanced C++ Programming Lab Report**

Lab 3: Class, Object, Constructor and Destructor in C++

**Prepared By :** Monsoon Sapkota (HCE081BEI022)

**Subject :** Object-Oriented Programming (OOP)

**Program :** Bachelor of Electronics, Communication and Information Engineering

**Institution :** Himalaya College of Engineering

**Date :** May 30, 2025

# OBJECTIVE

* To understand the concept of classes and objects in C++.
* To implement constructors for automatic initialization of objects.
* To use destructors for releasing resources and observing object lifecycle.
* To develop modular and reusable code using object-oriented principles.

# BACKGROUND THEORY

In C++, classes are user-defined data types that encapsulate data and functions. An object is an instance of a class, allowing us to represent real-world entities in code. A constructor is a special function that is automatically invoked when an object is created, used to initialize data members. Constructors can be default, parameterized, or copy constructors. A destructor is called automatically when an object goes out of scope or is deleted; it is used to release resources or perform cleanup tasks. Together, these features form the foundation of objectoriented programming, promoting code reusability, abstraction, and maintainability.

**Object:**

An object is any entity, thing or organization that exists in real world that consists of two fundamentals characteristics: its attributes and behavior. In OOP, the problem is divided into a group of objects and each object consists of own properties (data) and behavior (functions).

* Objects are the basic runtime entities which may be created or destroyed at run time.
* Object can communicate with others by using message passing mechanism.
* The member function of an object can only access its data.
* For example; a dog having attributes such as color, weight, age etc. and behaviors such as barking, wagging tail etc.

**Class:**

A class is the collection of similar objects which is defined as the template or prototype to define the common attributes and behavior for all the objects of the class. In fact objects are variables of type class. The entire set of data and code of an object can be made a user-defined data type with the help of a class.

* Once a class has been defined, can create any number of Objects associated with that class.
* No memory is allocated when class is created.
* Class has three access specifiers: public, private and protected. So, class incorporates the concept of data hiding.
* For example, mango, apple and orange are members of class fruit.

**Syntax:**

class ClassName { private:

// private data members // private member functions protected:

// protected members (optional) public:

// public data members

// public member functions

};

**Example:**

#include <iostream> using namespace std; class Demo { private:

int a;

protected:

int b;

public:

int c;

void setValues() {

a = 10; b = 20; c = 30;

} void display() { cout << "Private a = " << a << endl; cout << "Protected b = " << b << endl; cout << "Public c = " << c << endl;

}}; int main() { Demo obj; obj.setValues(); obj.display();

// obj.a = 1; //Error: private member // obj.b = 2; // Error: protected member obj.c = 3; // OK: public member return 0;

}

Explanation:

* a is private → Only accessible inside class methods.
* b is protected → Also accessible inside the class and in derived classes.
* c is public → Can be accessed directly from outside the class (like in main())

**Constructors:**

* In C++, a constructor is a special member function of a class that is automatically called when an object of the class is created.
* It is used to initialize the data members of the class.

**Purpose:**

Automatically initializes objects when no values are passed.

Sets default or fixed values to class members.

Useful for creating multiple objects with the same initial state

**Characteristics:**

1. No parameters.
2. Automatically invoked when an object is created without arguments.
3. Provided by the compiler if no other constructor is defined.
4. Can be explicitly defined by the user.

**Syntax:**

class ClassName { public:

ClassName(); // Default constructor

};

**Types of constructors:**

* 1. Default constructor:
     + If you do not define any constructor, the compiler automatically provides a default constructor that does nothing but creates the object.

class Test {

// No constructor defined

}; int main() {

Test t1; // Compiler provides default constructor return 0;

}

* 1. Parameterized constructor:
     + A parameterized constructor is a constructor that takes arguments/parameters.
     + It is used to initialize objects with specific values at the time of creation.

Syntax:

class ClassName { public:

ClassName(data\_type parameter1, data\_type parameter2, ...);

};

* 1. Copy constructor:
  + A copy constructor is a special constructor in C++ used to create a new object as an exact copy of an existing object.
  + It copies the data members of one object to another.

Syntax:

class ClassName { public:

ClassName(const ClassName &obj); // Copy constructor declaration

};

**Destructor:**

* + A destructor is a special member function in C++ that is automatically called when an object goes out of scope or is deleted.
  + It is used to free resources allocated to the object.

**Purpose:**

* + To perform clean-up tasks (e.g., releasing memory, closing files).
  + To avoid memory leaks in programs using dynamic memory.

**Characteristics:**

* + Name is the same as the class, prefixed with a tilde ~.
  + Takes no arguments and returns nothing.
  + Only one destructor is allowed per class (no overloading).
  + Automatically invoked at the end of object’s lifetime.

**Syntax:**

~ClassName() {

// code to release resources

}

**Example of constructor and destructor:**

#include <iostream> using namespace std; class Demo { public: Demo() { cout << "Constructor called." << endl;

}

~Demo() { cout << "Destructor called." << endl;

} void show() { cout << "Inside show function." << endl;

} }; int main() {

Demo obj; // Constructor is called obj.show(); // Function call return 0; // Destructor is called automatically