

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 and Destructors in a program.

# BACKGROUND THEORY

C++ classes are user-defined data types that encapsulate data and functions, with objects being instances of these classes, representing real-world entities. Constructors, which can be default, parameterized, or copy constructors, are special functions invoked upon object creation to initialize data members. Conversely, destructors are automatically called when an object goes out of scope or is deleted, releasing resources and performing cleanup tasks. These features collectively form the basis of object-oriented programming, enhancing 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 a way to bind the data and its associated functions together. It allows the data (and functions) to be hidden, if necessary, from external use. When defining a class, we are creating a new abstract data type that can be treated like any other built-in data type.

* Once a class has been defined, we can create multiple 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 helps in data abstraction.
* For example, EmployeeID, Name , Salary are the members of the class Employee.

**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 Employee { private:

int EmployeeId; protected: float Salary; public: string Name; void setValues() {

EmployeeId=01 ;

Salary = 20000;

Name = “Ramesh”;

} void display() { cout << "Private ID= " << EmployeeId << endl; cout << "Protected Salary = " << Salary << endl; cout << "Public Name = " << Name << endl;

}}; int main() { Employee e;

e.setValues();

// e. EmployeeId = 01; //Error: private member // e.Salary = 20000; // Error: protected member

e.Name= Ramesh; // OK: public member

e.display(); return 0;

}

Explanation:

* EmployeeId is private → Only accessible inside class methods.
* Salary is protected → Also accessible inside the class and in derived classes.
* Name 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 and has same name as Class.
2. Automatically invoked when an object is created without arguments.
3. Provided by the compiler if no other constructor is defined.

**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 an 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

}

## Lab Tasks

1. Create a class Employee with data members employeeID, name, and salary. Add member functions to read and display these details. Write a program to input and display any number of employees as desired by the user.

**Source Code:**

#include<iostream> #include<string> using namespace std; class Employee{

int employeeid; string name; float salary; public:

void readData(){ cout<<"Enter the id of Employee: ";

cin>>employeeid;

cout<<"Enter name of Employee: ";

cin>>name;

cout<<"Enter Salary of Employee: ";

cin>>salary;

}

void display(){

cout<<"Employee ID: "<<employeeid<<" "<<"Employee Name:

"<<name<<" "<< " Salary: "<< salary;

}

}; int main()

{

int n;

cout<<"Enter the number of Employees:";

cin>>n;

Employee \*employees=new Employee[n]; for(int i=0;i<n;i++)

{

cout<<"Enter details for the Employee:"<<i+1<<endl; employees[i].readData();

}

cout<<"---Employee details---"<<endl; for(int i=0;i<n;i++)

{

employees[i].display();

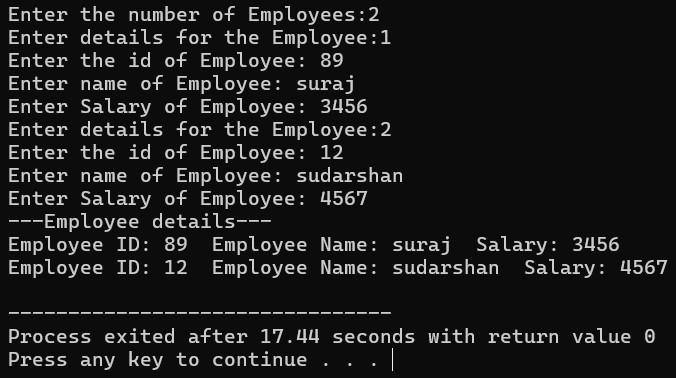
cout<<endl;

}

delete[] employees; return 0;

**}**

**Output:**



2. Design a class Student with members name, roll, and marks. Provide appropriate methods to read and display data.

**Source Code:**

#include<iostream> #include<string> using namespace std; class Student{

int roll; string name; float marks; public:

void readData(){

cout<<"Enter the roll number of the Student: ";

cin>>roll;

cout<<"Enter name of the Student: ";

cin>>name;

cout<<"Enter marks of the Student: ";

cin>>marks;

}

void display(){

cout<<"Roll Number: "<<roll<<" "<<"Student Name: "<<name<<"

"<< " Marks Obtained: "<< marks;

}

};

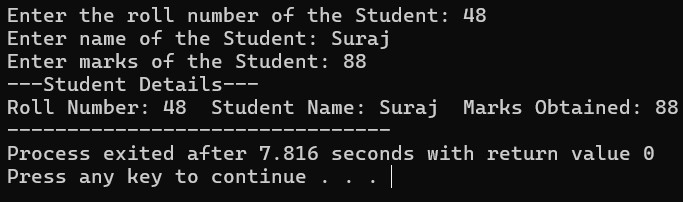
Student s;

s.readData(); cout<<"---Student Details---"<<endl;

s.display(); return 0;

}

**Output:**



3. Write a program designing a class Customer with member variables firstName, lastName, address, and phoneNo. Add member functions to accept input and display these variables.

**Source Code:**

#include<iostream> #include<string> using namespace std; class Customer{ string firstName; string lastName; string address; string phoneNo; public:

void readData(){ cout<<"Enter the Name of the Customer: "; cin>>firstName>>lastName;

cout<<"Enter the Address of the Customer: ";

cin>>address;

cout<<"Enter the Phone Number of the Customer: ";

cin>>phoneNo;

}

void display(){

cout<<"Name: "<<firstName<<lastName<<endl << "Address:

"<<address<<endl<<"Phone Number: "<<phoneNo;

}

};

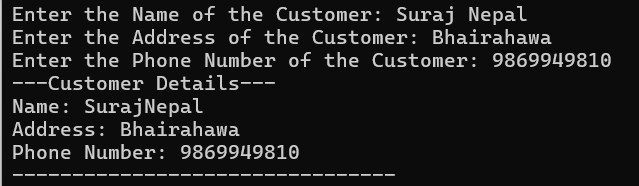
Customer c;

c.readData(); cout<<"---Customer Details---"<<endl;

c.display(); return 0;

}

**Output:**



4. Create a class Information to store name and address of students. Store information for two students and write a function to swap the contents of these two objects.

**Source Code:**

#include<iostream> #include<string> using namespace std; class Information{

string name; string address; public:

void readData(){ cout<<"Enter the Name of the Student: ";

cin>>name;

cout<<"Enter the Address of the Student: ";

cin>>address;

}

void display(){ cout<<"Name: "<<name<< endl << "Address: "<<address<<endl;

}

void swap(Information &obj)

{

string tempName=name; name=obj.name; obj.name=tempName;

string tempAddress=address; address=obj.address; obj.address=tempAddress;

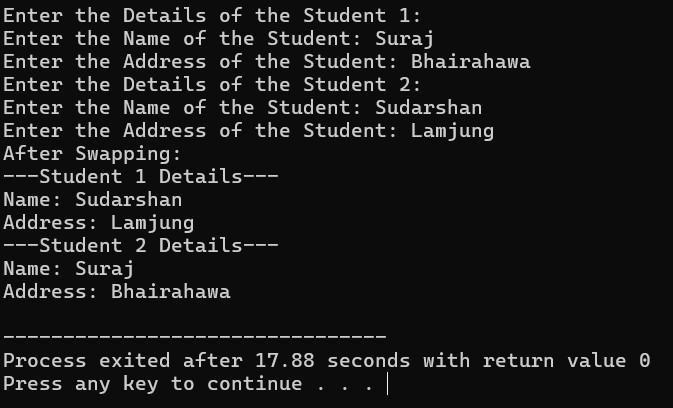
}

};

Information s1,s2; cout<<"Enter the Details of the Student 1: "<<endl; s1.readData(); cout<<"Enter the Details of the Student 2: "<<endl; s2.readData(); s1.swap(s2); cout<<"After Swapping:"<<endl; cout<<"---Student 1 Details---"<<endl; s1.display(); cout<<"---Student 2 Details---"<<endl; s2.display(); return 0;

}

**Output:**



5. Design a class to represent a bank account with members: depositor’s name, account number and account type. Include member functions to initialize and display the depositor’s name and account type.

**Source Code:**

#include <iostream> #include <string> using namespace std; class BankAccount { string depositorName; long long accountNumber; string accountType; public:

void input() { cout << "Enter Depositor's name: "; getline(cin, depositorName); cout << "Enter Account number: "; cin >> accountNumber;

cin.ignore(); cout << "Enter Account type: "; getline(cin, accountType);

} void display() { cout << "\nDepositor's Name: " << depositorName << endl; cout << "Account Type: " << accountType << endl;

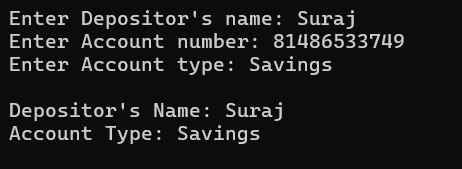
}

};

int main() {

BankAccount acc; acc.input(); acc.display(); return 0; }

**Output:**



6. Write a program to design a class Rectangle with constructors and member functions to calculate the area and perimeter.

**Source Code:**

#include<iostream> #include<string> using namespace std; class Rectangle{ float length,width;

public:

Rectangle()

{

length=0; width=0;

}

Rectangle(double l,double w)

{

length=l; width=w;

}

double area()

{

return length\*width;

}

double perimeter()

{

return 2\*(length+width);

}

void display()

{

cout<<"Length: "<<length<<" Width: "<<width<<endl; cout<<"Area: "<<area()<<endl; cout<<"Perimeter: "<<perimeter()<<endl;

}

}; int main()

{

double l,w;

cout<<"Enter length and width of the rectangle:";

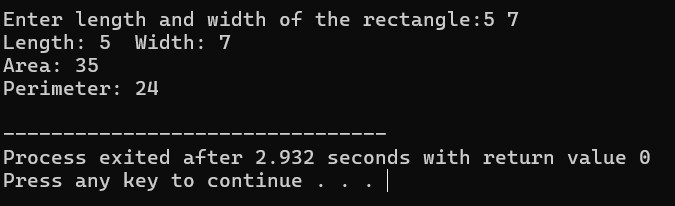
cin>>l>>w;

Rectangle r(l,w);

r.display(); return 0;

}

**Output:**



7. Write a program demonstrating that destructors execute in the reverse order of constructors. Display the corresponding object IDs during construction and destruction.

**Source Code:**

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

int id;

public:

Demo(int i)

{

id=i;

cout<<"Constructor called for Object: "<<id<<endl;

}

~Demo(){

cout<<"Destructor called for Object: "<<id<<endl;

}

}; int main(){

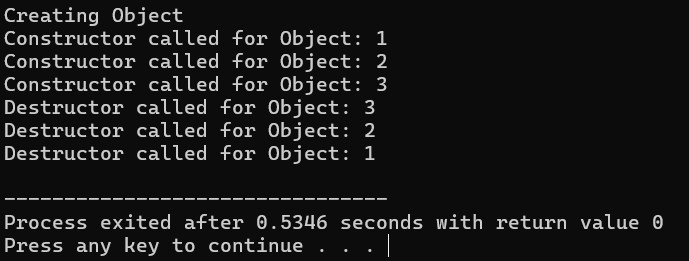
cout<<"Creating Object"<<endl;

Demo d1(1);

Demo d2(2); Demo d3(3); return 0;

}

**Output:**



8. Create a class Time with members for hours, minutes, and seconds. Write a member function AddTime() that adds two Time objects passed as arguments and returns the result.

**Source Code:**

#include<iostream> #include<string> using namespace std; class Time{ int hour,minute,second;

public:

void read()

{

cout<<"Enter Hours: ";

cin>>hour;

cout<<"Enter Minutes: ";

cin>>minute;

cout<<"Enter Seconds: ";

cin>>second;

}

void display()

{

cout<<hour<<"h"<<minute<<"m"<<second<<"s"<<endl;

}

Time add(Time t2)

{

Time temp;

temp.second=second+t2.second; temp.minute=minute+t2.minute+(temp.second/60);

temp.second%=60;

temp.hour=hour+t2.hour+(temp.minute/60);

temp.minute%=60; return temp;

}

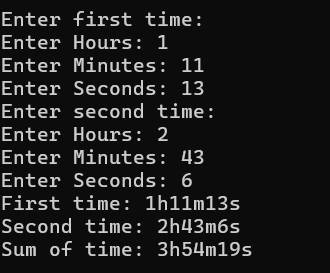
}; int main()

{

Time t1,t2,sum; cout<<"Enter first time:"<<endl; t1.read(); cout<<"Enter second time:"<<endl; t2.read(); sum=t1.add(t2); cout<<"First time: "; t1.display(); cout<<"Second time: "; t2.display(); cout<<"Sum of time: "; sum.display(); return 0;

}

**Output:**



9. Design a class Land that stores Ropani, Ana, Paisa, and Dam. Write a member function to add two Land objects and return their sum as a new object.

Nepali Land Measurement Reference:

1 Ropani = 16 Ana

1 Ana = 4 Paisa

1 Paisa = 4 Dam

**Source Code:** #include <iostream> using namespace std; class Land { int ropani; int ana; int paisa; int dam; public:

void read() { cout << "Enter Ropani: "; cin >> ropani; cout << "Enter Ana: "; cin >> ana; cout << "Enter Paisa: "; cin >> paisa; cout << "Enter Dam: "; cin >> dam;

} void display() {

cout << ropani << " Ropani " << ana << " Ana " << paisa << " Paisa " << dam << "Dam" << endl; }

Land add(Land l2) { Land result; result.dam = dam + l2.dam; result.paisa = paisa + l2.paisa + result.dam / 4; result.dam %= 4; result.ana = ana + l2.ana + result.paisa / 4; result.paisa %= 4; result.ropani = ropani + l2.ropani + result.ana / 16; result.ana %= 16; return result;

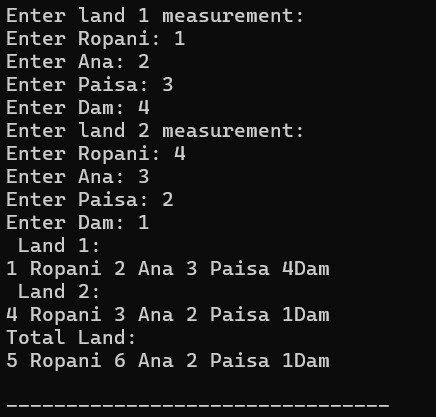
} }; int main() { Land l1, l2, sum; cout << "Enter land 1 measurement:"<<endl; l1.read(); cout << "Enter land 2 measurement:"<<endl; l2.read(); sum = l1.add(l2); cout << " Land 1: "<<endl; l1.display(); cout << " Land 2: "<<endl;

l2.display();

cout << "Total Land: "<<endl; sum.display(); return 0;

}

**Output:**



10. Create a class Employee and add a member function to increase the salary of each employee by 10%. Read and display the details of all employees before and after the salary increases.

**Source Code:**

#include <iostream> #include<string> using namespace std; class Employee { int employeeID; string name; float salary; public:

void read() { cout << "Enter Employee ID: "; cin >> employeeID; cout << "Enter Name of Employee: "; cin.ignore(); getline(cin, name); cout << "Enter Salary: "; cin >> salary;

} void display() { cout << "ID: " << employeeID << ", Name: " << name << ", Salary: Rs. " << salary << endl;

} void increaseSalary() { salary += salary \* 0.10;

}

};

int main() { int n;

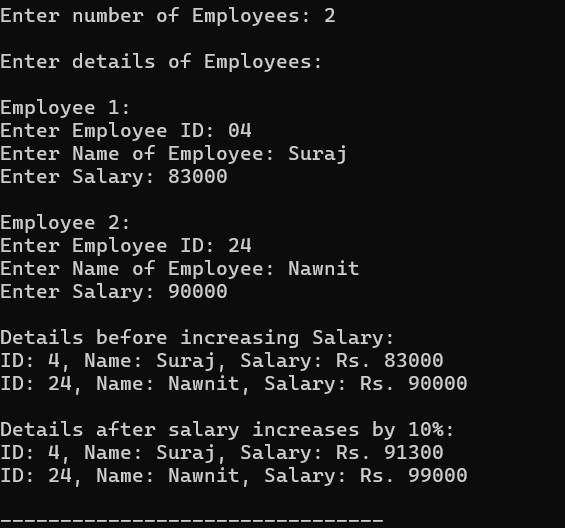
cout << "Enter number of Employees: "; cin >> n;

Employee emp[n]; cout << "\nEnter details of Employees:\n"; for(int i = 0; i < n; i++) { cout << "\nEmployee " << i + 1 << ":\n"; emp[i].read(); } cout << "\nDetails before increasing Salary:\n"; for(int i = 0; i < n; i++) { emp[i].display(); } for(int i = 0; i < n; i++) { emp[i].increaseSalary();

} cout << "\nDetails after salary increases by 10%:\n"; for(int i = 0; i < n; i++) { emp[i].display(); } return 0;

}

**Output:**



11. Write a program to find the area of a square and a rectangle using classes.

Initialize the objects dynamically using pointers.

**Source Code:**

#include<iostream> #include<string> using namespace std; class Square{

float side;

public:

void read()

{

cout<<"Enter the Side of the Square: ";

cin>>side;

}

float area()

{

return side\*side;

}

}; class Rectangle{ float length,breadth;

public:

void read()

{

cout<<"Enter the Length and Breadth of the rectangle: "; cin>>length>>breadth;

}

float area()

{

return length\*breadth;

}

}; int main()

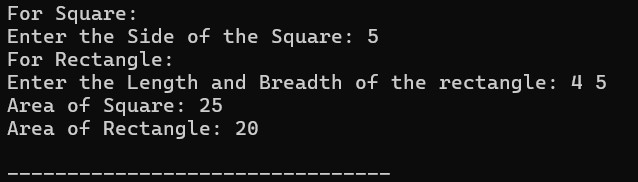
{

Square \*s=new Square; Rectangle \*r=new Rectangle; cout<<"For Square:"<<endl; s->read(); cout<<"For Rectangle:"<<endl; r->read();

cout<<"Area of Square: "<<s->area()<<endl; cout<<"Area of Rectangle: "<<r->area()<<endl; delete s; delete r; return 0;

}

**Output:**



12. Write a program defining an inline member function calculateVolume() outside the class using the inline keyword for a class Box.

**Source Code:** #include<iostream> using namespace std; class Box{ float length,breadth,height;

public:

void read()

{

cout<<"Enter the Length,Breadth and Height of the box: "; cin>>length>>breadth>>height;

}

inline float Volume();

}; inline float Box::Volume(){ return length\*breadth\*height;

} int main()

{

Box b;

b.read();

cout<<"Volume of the box: "<<b.Volume()<<endl; return 0;

}

**Output:**



**DISCUSSION**

In this lab, we practiced creating classes and objects in C++ to understand the basics of objectoriented programming. We implemented constructors to initialize objects and destructors to free up memory when objects are closed. By using default and parameterized constructors, we observed how object initialization can vary. The destructor helped demonstrate how C++ manages resources automatically.

**CONCLUSION**

This lab helped us obtain the core concepts of classes, objects, constructors, and destructors in C++. We learned how constructors initialize objects and how destructors free resources, while using C++ to prepare a program.