# Lab 3: CLASS, OBJECTS, CONSTRUCTORS AND DESTRUCTORS

# Objectives

* To understand and implement the concept of **classes and objects** in C++.
* To learn the **role of constructors and destructors** in object-oriented programming.
* To gain experience in defining, initializing, and cleaning up objects using constructors and destructors.

# Tools and Libraries Used

• Programming Language: C++

• IDE: Code::Blocks

• Libraries: #include <iostream>, #include <cmath>, #include <string>

# Theory

Class and Object:  
A **class** in C++ is a user-defined data type that contains variables (called data members) and functions (called member functions or methods). An **object** is an instance of a class that holds actual values and can access class methods.

SYNTAX:

class ClassName {

// Data Members

// Member Functions

};

EXAMPLE:

#include<iostream>

Using namespace std;

class Student {

public:

string name;

void display() {

cout << "Name: " << name << endl;

}

};

int main() {

Student s1; // Object creation

s1.name = "Alice";

s1.display();

return 0;

}

#### Constructor

* A constructor is a special function that automatically runs when an object is created.
* Its name is the same as the class and it has no return type.

Types of Constructors:

* Default Constructor
* Parameterized Constructor
* Copy Constructor

SYNTAX:

class ClassName {

public:

ClassName(); // Default Constructor

ClassName(int x); // Parameterized Constructor

ClassName(const ClassName &obj); // Copy Constructor

};

EXAMPLE:

class Student {

public:

string name;

Student() { // Default Constructor

name = "Unknown";

}

Student(string n) { // Parameterized Constructor

name = n;

}

void display() {

cout << "Name: " << name <<endl;

}

};

int main() {

Student s1; // Default Constructor called

Student s2("Sayal"); // Parameterized Constructor called

s1.display();

s2.display();

return 0;

}

#### Destructor

* A destructor is used to release resources when an object is destroyed.
* It has the same name as the class but with a tilde (~) prefix.
* It has no parameters and no return type.

SYNTAX:

class ClassName {

public:

~ClassName(); // Destructor

};

EXAMPLE:

#include<iostream>

using namespace std;

class Student {

public:

Student() {

cout << "Constructor called."<<endl;

}

~Student() {

cout << "Destructor called."<<endl;

}

};

int main() {

**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.**

#include <iostream>

#include <string>

using namespace std;

class Employee {

int employeeID;

string name;

float salary;

public:

void readData() {

cout << "Enter Employee ID: ";

cin >> employeeID;

cin.ignore(); // to clear newline from input buffer

cout << "Enter Name: ";

getline(cin, name);

cout << "Enter Salary (Rs): ";

cin >> salary;

}

void displayData() {

cout << "Employee ID: " << employeeID << ", Name: " << name << ", Salary: Rs. " << salary << endl;

}

};

int main() {

int n;

cout << "How many employees do you want to enter? ";

cin >> n;

Employee\* employees = new Employee[n]; // dynamic array of Employee

for (int i = 0; i < n; i++) {

cout << "\nEnter details for employee " << (i + 1) << ":\n";

employees[i].readData();

}

cout << "\nEmployee Details:\n";

for (int i = 0; i < n; i++) {

employees[i].displayData();

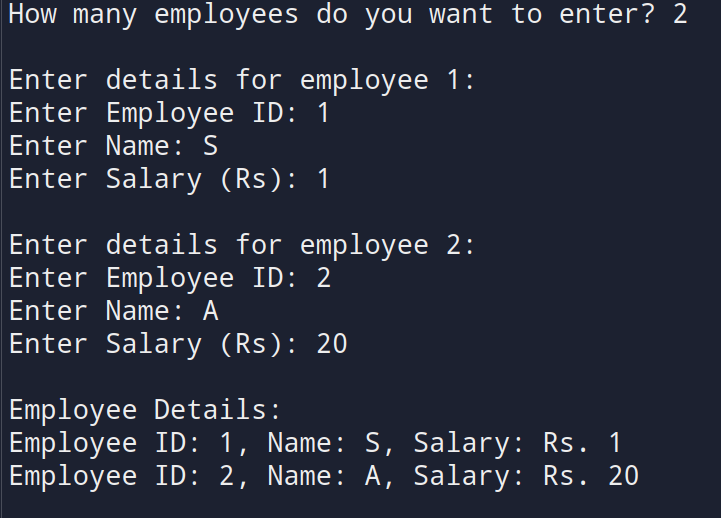
}

delete[] employees; // free allocated memory

return 0;

}

**OUTPUT**



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

#include <iostream>

#include <string>

using namespace std;

class Student {

string name;

int roll;

float marks;

public:

void readData() {

cout << "Enter student name: ";

getline(cin, name);

cout << "Enter roll number: ";

cin >> roll;

cout << "Enter marks: ";

cin >> marks;

cin.ignore(); // clear newline for next input

}

void displayData() {

cout << "Name: " << name << ", Roll No: " << roll << ", Marks: " << marks << endl;

}

};

int main() {

Student s;

s.readData();

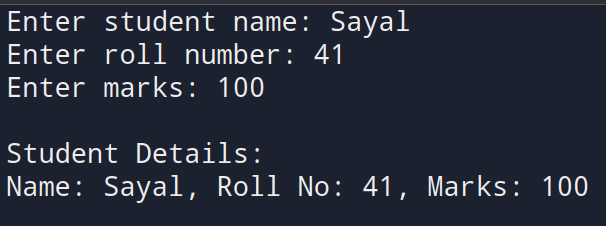
cout << "\nStudent Details:\n";

s.displayData();

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.**

#include <iostream>

#include <string>

using namespace std;

class Customer {

string firstName;

string lastName;

string address;

string phoneNo;

public:

void readData() {

cout << "Enter First Name: ";

getline(cin, firstName);

cout << "Enter Last Name: ";

getline(cin, lastName);

cout << "Enter Address: ";

getline(cin, address);

cout << "Enter Phone No: ";

getline(cin, phoneNo);

}

void displayData() {

cout << "\nCustomer Details:\n";

cout << "First Name: " << firstName << endl;

cout << "Last Name: " << lastName << endl;

cout << "Address: " << address << endl;

cout << "Phone No: " << phoneNo << endl;

}

};

int main() {

Customer customer;

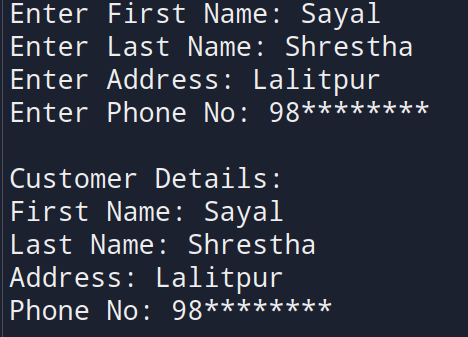
customer.readData();

customer.displayData();

return 0;

}

**OUTPUT**



**4. 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.**

#include <iostream>

#include <string>

using namespace std;

class BankAccount {

string depositorName;

long long accountNumber;

string accountType;

public:

void initialize() {

cout << "Enter depositor's name: ";

getline(cin, depositorName);

cout << "Enter account number: ";

cin >> accountNumber;

cin.ignore(); // clear newline

cout << "Enter account type: ";

getline(cin, accountType);

}

void display() {

cout << "\nDepositor's Name: " << depositorName << endl;

cout << "Account Type: " << accountType << endl;

}

};

int main() {

BankAccount account;

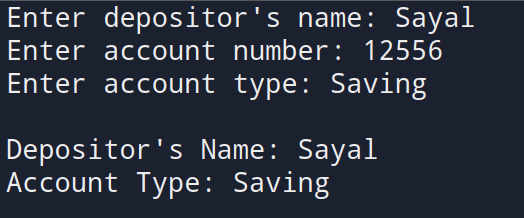
account.initialize();

account.display();

return 0;

}

**OUTPUT**



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

#include <iostream>

using namespace std;

class Rectangle {

double length;

double width;

public:

Rectangle() {// Default constructor

length = 0;

width = 0;

}

Rectangle(double l, double w) {// Parameterized constructor

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;

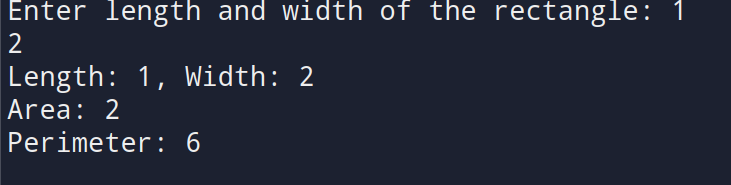
Rectangle rect(l, w);

rect.display();

return(0);

}

**OUTPUT**



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

#include <iostream>

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 objects...\n";

Demo d1(1);

Demo d2(2);

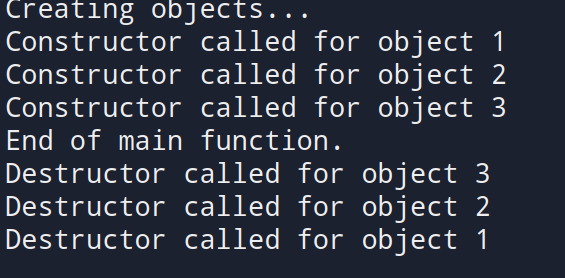
Demo d3(3);

cout << "End of main function.\n";

return 0;

}

**OUTPUT**



**7. 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.**

#include <iostream>

using namespace std;

class Time {

int hours;

int minutes;

int seconds;

public:

// Member function to read time

void readTime() {

cout << "Enter hours: ";

cin >> hours;

cout << "Enter minutes: ";

cin >> minutes;

cout << "Enter seconds: ";

cin >> seconds;

}

// Member function to display time

void displayTime() {

cout << hours << "h " << minutes << "m " << seconds << "s" << endl;

}

// Member function to add two Time objects and return the result

Time AddTime(Time t2) {

Time temp;

temp.seconds = seconds + t2.seconds;

temp.minutes = minutes + t2.minutes + (temp.seconds / 60);

temp.seconds %= 60;

temp.hours = hours + t2.hours + (temp.minutes / 60);

temp.minutes %= 60;

return temp;

}

};

int main() {

Time t1, t2, sum;

cout << "Enter first time:\n";

t1.readTime();

cout << "\nEnter second time:\n";

t2.readTime();

sum = t1.AddTime(t2);

cout << "\nFirst Time: ";

t1.displayTime();

cout << "Second Time: ";

t2.displayTime();

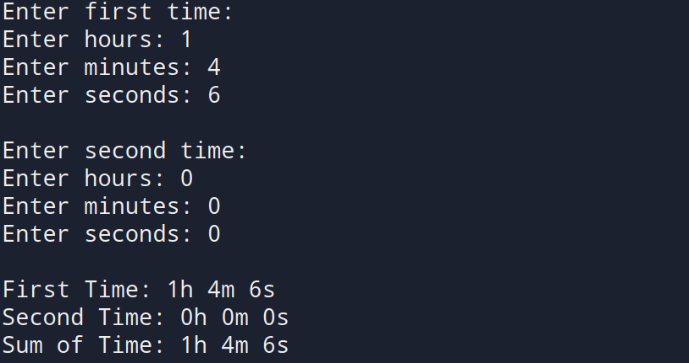
cout << "Sum of Time: ";

sum.displayTime();

return 0;

}

**OUTPUT**



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

#include <iostream>

using namespace std;

class LandMeasure {

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() const {

cout << ropani << " Ropani, "

<< ana << " Ana, "

<< paisa << " Paisa, "

<< dam << " Dam" << endl;

}

LandMeasure add(const LandMeasure& l2) const {

LandMeasure 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() {

LandMeasure l1, l2, sum;

cout << "Enter first land measurement:\n";

l1.read();

cout << "\nEnter second land measurement:\n";

l2.read();

sum = l1.add(l2);

cout << "\nFirst Land : "; l1.display();

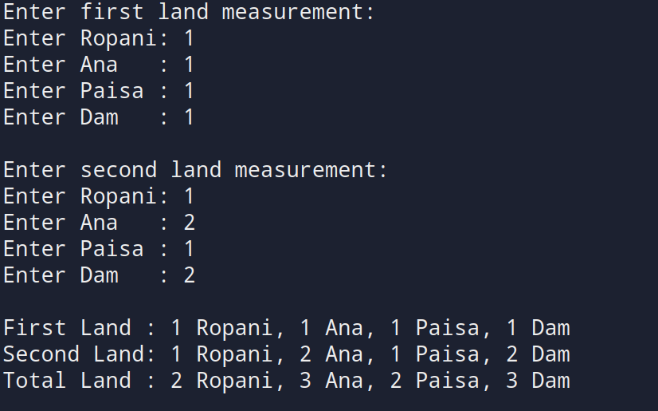
cout << "Second Land: "; l2.display();

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

return 0;

}

**OUTPUT**



**9. 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 increase.**

#include <iostream>

using namespace std;

class Employee {

int employeeID;

string name;

float salary;

public:

// Function to read employee data

void read() {

cout << "Enter Employee ID: ";

cin >> employeeID;

cout << "Enter Name: ";

cin.ignore(); // clear newline from input buffer

getline(cin, name);

cout << "Enter Salary: ";

cin >> salary;

}

// Function to display employee data

void display() {

cout << "ID: " << employeeID << ", Name: " << name << ", Salary: Rs. " << salary << endl;

}

// Function to increase salary by 10%

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 salary increase:\n";

for(int i = 0; i < n; i++) {

emp[i].display();

}

for(int i = 0; i < n; i++) {

emp[i].increaseSalary();

}

cout << "\nDetails after 10% salary increase:\n";

for(int i = 0; i < n; i++) {

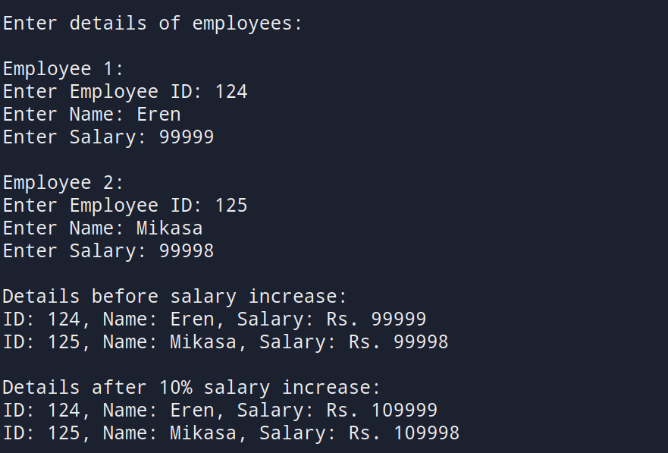
emp[i].display();

}

return 0;

}

**OUTPUT**



**10. Write a program to find the area of a square and a rectangle using classes. Initialize the objects dynamically using pointers.**

#include <iostream>

using namespace std;

// Class for Square

class Square {

float side;

public:

void read() {

cout << "Enter the side of the square: ";

cin >> side;

}

float area() {

return side \* side;

}

};

// Class for Rectangle

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() {

// Dynamically create square and rectangle objects

Square\* sq = new Square;

Rectangle\* rect = new Rectangle;

// Read values

cout << "Square:\n";

sq->read();

cout << "\nRectangle:\n";

rect->read();

// Display areas

cout << "\nArea of Square = " << sq->area() << endl;

cout << "Area of Rectangle = " << rect->area() << endl;

// Free memory

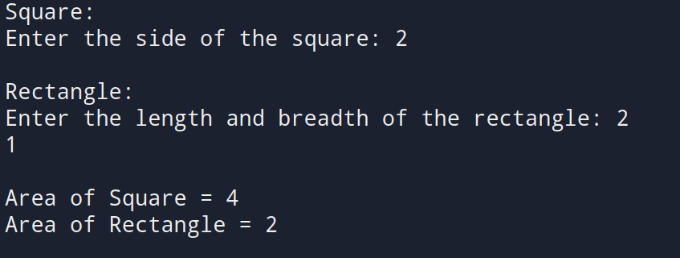
delete sq;

delete rect;

return 0;

}

**OUTPUT**



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

#include <iostream>

using namespace std;

class Box {

float length, breadth, height;

public:

void read() {

cout << "Enter length, breadth, and height of the box: ";

cin >> length >> breadth >> height;

}

inline float calculateVolume(); // Declaration of inline function

};

inline float Box::calculateVolume() {

return length \* breadth \* height;

}

int main() {

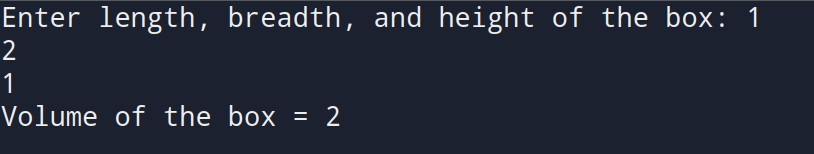
Box b;

b.read();

cout << "Volume of the box = " << b.calculateVolume() << endl;

return (0); }

**OUTPUT**



# DISCUSSION

This lab focused on the practical implementation of object-oriented programming (OOP) concepts in C++. We began by designing basic classes like Employee, Student, and Customer to demonstrate how data members and member functions interact. Input/output operations were implemented using cin, getline, and cout, reinforcing our understanding of data encapsulation and user interaction.Advanced features like constructor overloading and destructor behavior were introduced with the Rectangle and Demo classes. The destructor example effectively illustrated the reverse order of object destruction, a key concept in understanding memory management.

Finally, dynamic memory allocation and pointer-based object management were practiced, as well as demonstrating the utility of inline functions for performance optimization in simple computations.

# CONCLUSION

The lab successfully demonstrated the core principles of object-oriented programming in C++. By developing a variety of programs, we gained hands-on experience in creating and manipulating classes, encapsulating data, using constructors/destructors, and applying function overloading and pointer-based memory handling.

These exercises solidified our theoretical understanding. This foundation will be essential as we progress toward more complex software development tasks in C++.