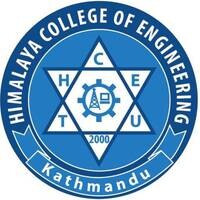


**TRIBHUVAN UNIVERSITY**

**INSTITUTE OF ENGINEERING**



**HIMALAYA COLLEGE OF ENGINEERING**

**CHYASAL, LALITPUR**

**Lab Report No: - 3**

**Title: - Class and objects in C++**

**Submitted by: - Submitted To: -**

**Name: - Atul Bhattarai Department Of Computer and Electronics**

**Roll NO: - HCOE 081 BEI 011 Checked by: -**

**Date of submission: -**

**Objectives:**

* Understand the concept of classes and objects in C++.
* Learn how to define member functions inside and outside a class.
* Demonstrate access specifiers (public, private, protected) for data encapsulation.

**Theory:**

**Class:**

A class in C++ is a user-defined data type that serves as a blueprint for creating objects. It encapsulates:

* Data members (attributes/properties): Variables that store the state of an object
* Member functions (methods): Functions that define the behavior of objects

Key Characteristics:

* Acts as a template for object creation
* Supports data abstraction and encapsulation
* Can contain different access specifiers (public, private, protected)
* May include special member functions (constructors, destructors)

Basic syntax:

class ClassName {

private:

// data members

public:

// member functions

};

**Object:**

An object is an instance of a class that occupies memory and contains actual data.

Key Properties:

* Represents a real-world entity
* Has a unique identity (memory address)
* Contains its own copy of data members (unless static)
* Can access all public members of its class

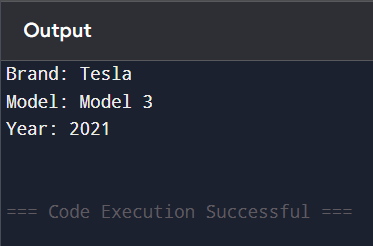
Basic syntax:

ClassName objectName; // object declaration

objectName.memberFunction(); // function call using object

**Lab Assignments:**

1. Define a class Car with private members brand, model, and year. Include public member functions to set and get these private members. Ensure that only member functions can access these private members.
2. #include <iostream>
3. using namespace std;
4. class Car {
5. private:
6. string brand;
7. string model;
8. int year;
9. public:
10. void setBrand(string b) {
11. brand = b;
12. }
13. void setModel(string m) {
14. model = m;
15. }
16. void setYear(int y) {
17. year = y;
18. }
19. string getBrand() {
20. return brand;
21. }
22. string getModel() {
23. return model;
24. }
25. int getYear() {
26. return year;
27. }
28. };
29. int main() {
30. Car myCar;
31. myCar.setBrand("Tesla");
32. myCar.setModel("Model 3");
33. myCar.setYear(2021);
34. cout << "Brand: " << myCar.getBrand() << endl;
35. cout << "Model: " << myCar.getModel() << endl;
36. cout << "Year: " << myCar.getYear() << endl;
37. return 0;
38. }



2. Define a class Book with private members title, author, and year. Implement both default and parameterized constructors. The default constructor should initialize the members with default values, and the parameterized constructor should set these values based on user input. Provide a method to display the details of a book.

#include <iostream>

using namespace std;

class Book {

private:

    string title;

    string author;

    int year;

public:

    Book() {

        title = "Unknown";

        author = "Unknown";

        year = 0;

    }

    Book(string t, string a, int y) {

        title = t;

        author = a;

        year = y;

    }

    void display() {

        cout << "Title: " << title << ", Author: " << author << ", Year: " << year << endl;

    }

};

int main() {

    Book b1;

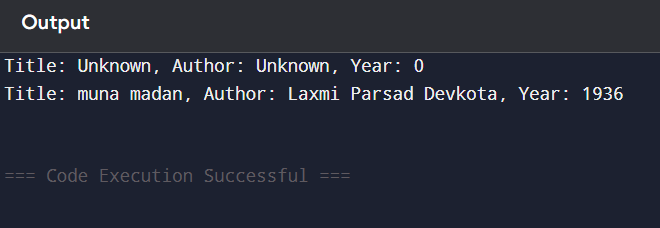
    Book b2("muna madan", "Laxmi Parsad Devkota", 1936);

    b1.display();

    b2.display();

    return 0;

}



3. Create a class Employee with private members name and age. Implement a copy constructor to create a deep copy of an existing Employee object. Provide a method to display the details of an employee.

#include <iostream>

using namespace std;

class Employee {

private:

    string name;

    int age;

public:

    Employee(string n, int a) {

        name = n;

        age = a;

    }

    Employee(const Employee &e) {

        name = e.name;

        age = e.age;

    }

    void display() {

        cout << "Name: " << name << ", Age: " << age << endl;

    }

};

int main() {

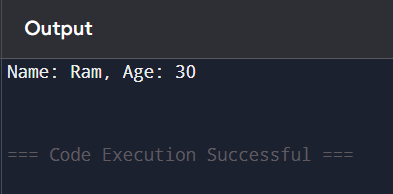
    Employee emp1("Ram", 30);

    Employee emp2 = emp1;

    emp2.display();

    return 0;

}



4. Define a class Book with a private member title. Implement a constructor that initializes title and a destructor that prints a message when the Book object is destroyed. Create instances of Book objects in main() to demonstrate the usage of the destructor.

#include <iostream>

using namespace std;

class Book {

private:

    string title;

public:

    Book(string t) {

        title = t;

        cout << "Book \"" << title << "\" created." << endl;

    }

    ~Book() {

        cout << "Book \"" << title << "\" destroyed." << endl;

    }

};

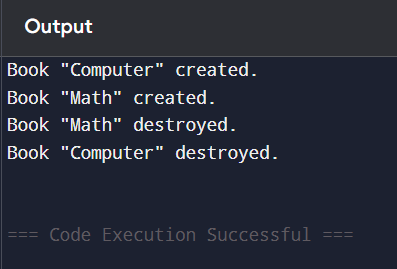
int main() {

    Book b1("Computer");

    Book b2("Math");

    return 0;

}



5. Define a class Rectangle with private members length and width. Implement a constructor to initialize these members. Write a function calculateArea() that calculates and returns the area of the rectangle. Define another function doubleDimensions(Rectangle rect) that takes a Rectangle object as an argument and doubles its length and width. Demonstrate the usage of both functions in main().

#include <iostream>

using namespace std;

class Rectangle {

private:

    double length;

    double width;

public:

    Rectangle(double l, double w) {

        length = l;

        width = w;

    }

    double calculateArea() {

        return length \* width;

    }

    void displayDimensions() {

        cout << "Length: " << length << ", Width: " << width << endl;

    }

    friend void doubleDimensions(Rectangle rect);

};

void doubleDimensions(Rectangle rect) {

    rect.length \*= 2;

    rect.width \*= 2;

    cout << "After Doubling:" << endl;

    rect.displayDimensions();

    cout << "Area after doubling: " << rect.calculateArea() << endl;

}

int main() {

    Rectangle rect(4.0, 5.0);

    cout << "Original:" << endl;

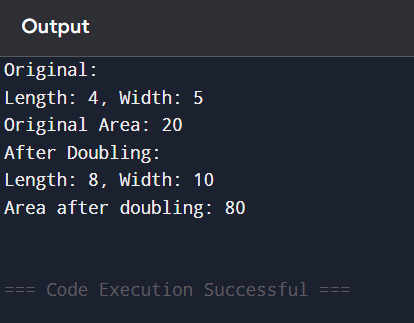
    rect.displayDimensions();

    cout << "Original Area: " << rect.calculateArea() << endl;

    doubleDimensions(rect);

    return 0;

}



6. Define a class Student with private members name and age. Implement a constructor to initialize these members. Create an array studentArray of size 3 to store objects of type Student. Populate the array with student details and display the details using a method displayStudentDetails().

#include <iostream>

using namespace std;

class Student {

private:

    string name;

    int age;

public:

    Student() {}

    Student(string n, int a) {

        name = n;

        age = a;

    }

    void displayStudentDetails() {

        cout << "Name: " << name << ", Age: " << age << endl;

    }

};

int main() {

    Student studentArray[3] = {

        Student("Ram ", 20),

        Student("Shyam", 19),

        Student("Hari", 21)

    };

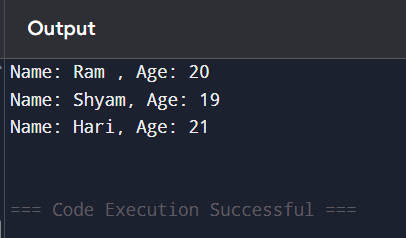
    for (int i = 0; i < 3; i++) {

        studentArray[i].displayStudentDetails();

    }

    return 0;

}



7. Define a class Math with two overloaded methods add(). The first method should take two integers as parameters and return their sum. The second method should take three integers as parameters and return their sum. Demonstrate the usage of both methods in main().

#include <iostream>

using namespace std;

class Math {

public:

    int add(int a, int b) {

        return a + b;

    }

    int add(int a, int b, int c) {

        return a + b + c;

    }

};

int main() {

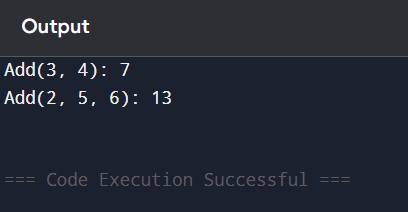
    Math m;

    cout << "Add(3, 4): " << m.add(3, 4) << endl;

    cout << "Add(2, 5, 6): " << m.add(2, 5, 6) << endl;

    return 0;

}



8. Implement a class Area with overloaded methods calculate(). The first method should take the radius of a circle as a parameter and return the area of the circle. The second method should take the length and width of a rectangle as parameters and return the area of the rectangle. Demonstrate the usage of both methods in main().

#include <iostream>

using namespace std;

class Area {

public:

    double calculate(double radius) {

        return 3.1416 \* radius \* radius;

    }

    double calculate(double length, double width) {

        return length \* width;

    }

};

int main() {

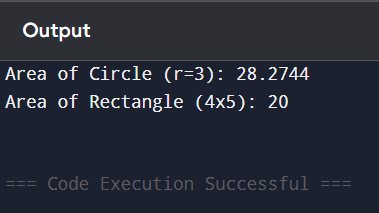
    Area a;

    cout << "Area of Circle (r=3): " << a.calculate(3.0) << endl;

    cout << "Area of Rectangle (4x5): " << a.calculate(4.0, 5.0) << endl;

    return 0;

}



**Discussion:**

In C++, a **class** is a blueprint for creating **objects,** which are instances that hold data and functions together. Classes help organize code and promote **encapsulation** by keeping data private and allowing access only through public methods. We also explored **constructors**, which are special functions used to initialize objects. **Default constructors** set initial values automatically, while **parameterized constructors** allow values to be passed during object creation. The **copy constructor** was used to create an exact copy of an object, ensuring safe data handling. These concepts are essential for writing clean, organized, and reusable C++ programs.

**Conclusion:**

Through these exercises, we successfully demonstrated the core features of C++ Object-Oriented Programming, including encapsulation, constructors (default, parameterized, copy), destructors, function and method overloading, friend functions, and arrays of objects. Each class was implemented with a clear structure and purpose, helping to strengthen the understanding of OOP principles and their practical application. These programs collectively serve as a foundational base for designing more complex systems in software development.