MODULE: 4

OOPS Concept

**1. WAP to print “Hello World” using C++**

ans:

Code:#include <iostream>

using namespace std;

int main()

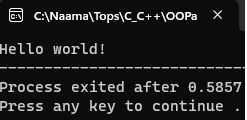
{

cout<<"Hello world!";

return 0;

}

Output:



**2. What is OOP? List OOP concepts**

ans:OOP is known as object oriented programming, this type of programming relies more on objects

and classes.

OOP Concepts:

Class:

⦁ A blueprint or template for creating objects.

Object:

⦁ Represents a real-world entity and encapsulates data and behavior.

Encapsulation:

⦁ Access to the internal details of the object is controlled by access

modifiers.

Inheritance:

⦁ Promotes code reuse and establishes a hierarchy of classes.

Polymorphism:

⦁ The ability of objects of different classes to be treated as objects of a

common base class.

**3. What is the difference between OOP and POP?**

ans:Object-Oriented Programming (OOP) organizes code around objects, emphasizing encapsulation, inheritance, and polymorphism for efficient and modular design.

Procedural Programming (POP) centers on procedures or functions manipulating data sequentially, with less emphasis on object-oriented concepts, making it suitable for simpler, linear tasks. OOP fosters code reuse and abstraction, while POP focuses on straightforward procedural execution.

**4.WAP to create simple calculator using class**

ans:

Code:

#include <iostream>

using namespace std;

class Calculator {

public:

double add(double num1, double num2) {

return num1 + num2;

}

double subtract(double num1, double num2) {

return num1 - num2;

}

double multiply(double num1, double num2) {

return num1 \* num2;

}

double divide(double num1, double num2) {

if (num2 == 0) {

cout << "Error! Cannot divide by zero." << endl;

return 0;

}

return num1 / num2;

}

};

int main() {

double num1, num2;

char op;

Calculator calc;

cout << "Enter two numbers: ";

cin >> num1 >> num2;

cout << "Enter operator (+, -, \*, /): ";

cin >> op;

switch (op) {

case '+':

cout << "Result: " << calc.add(num1, num2);

break;

case '-':

cout << "Result: " << calc.subtract(num1, num2);

break;

case '\*':

cout << "Result: " << calc.multiply(num1, num2);

break;

case '/':

cout << "Result: " << calc.divide(num1, num2);

break;

default:

cout << "Invalid operator!";

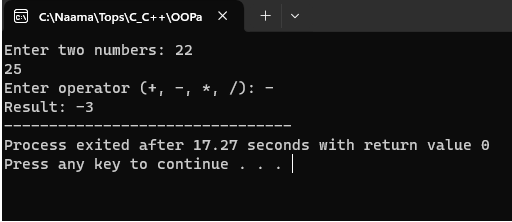
break;

}

return 0;

}

Output:



**6. Write a C++ program to implement a class called Circle that has private member variables for radius. Include member functions to calculate the circle's area and circumference**

ans:

Code:

#include <iostream>

using namespace std;

class Circle {

private:

double radius;

public:

// Constructor

Circle(double r) {

radius = r;

}

double calculateArea() {

return 3.14 \* radius \* radius;

}

// Function to calculate the circumference of the circle

double calculateCircumference() {

return 2 \* 3.14 \* radius;

}

};

int main() {

double r;

cout << "Enter the radius of the circle: ";

cin >> r;

Circle circle(r);

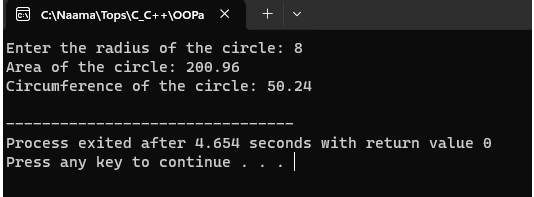
cout << "Area of the circle: " << circle.calculateArea() << endl;

cout << "Circumference of the circle: " << circle.calculateCircumference() << endl;

return 0;

}

Output:



**7. Write a C++ program to create a class called Rectangle that has private member variables for length and width. Implement member functions to calculate the rectangle's area and perimeter.**

Code:

#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;

}

double calculatePerimeter() {

return 2 \* (length + width);

}

};

int main() {

double l, w;

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

cin >> l;

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

cin >> w;

Rectangle rectangle(l, w);

cout << "Area of the rectangle: " << rectangle.calculateArea() << endl;

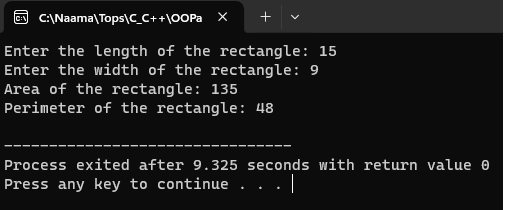
// Calculate and display the perimeter

cout << "Perimeter of the rectangle: " << rectangle.calculatePerimeter() << endl;

return 0;

}

Output:



**8. Write a C++ program to create a class called Person that has private member variables for name, age and country. Implement member functions to set and get the values of these variables**

Code:

#include <iostream>

#include <string>

using namespace std;

class Person {

private:

string name;

int age;

string country;

public:

void setName(string n) {

name = n;

}

string getName() {

return name;

}

void setAge(int a) {

age = a;

}

int getAge() {

return age;

}

void setCountry(string c) {

country = c;

}

string getCountry() {

return country;

}

};

int main() {

Person person;

person.setName("John");

person.setAge(30);

person.setCountry("USA");

cout << "Name: " << person.getName() << endl;

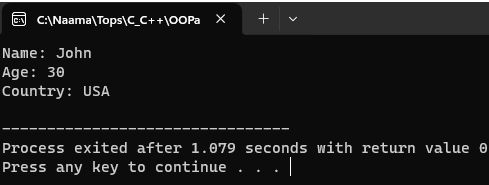
cout << "Age: " << person.getAge() << endl;

cout << "Country: " << person.getCountry() << endl;

return 0;

}

Output:



**9. Write a program to find the multiplication values and the cubic values using inline function**

Code:

#include <iostream>

using namespace std;

inline int multiply(int a, int b) {

return a \* b;

}

inline int cube(int num) {

return num \* num \* num;

}

int main() {

int num1, num2;

cout << "Enter two numbers: ";

cin >> num1 >> num2;

// Calculate and display multiplication value

cout << "Multiplication of " << num1 << " and " << num2 << " is: " << multiply(num1, num2) << endl;

int num3;

cout << "Enter a number to find its cubic value: ";

cin >> num3;

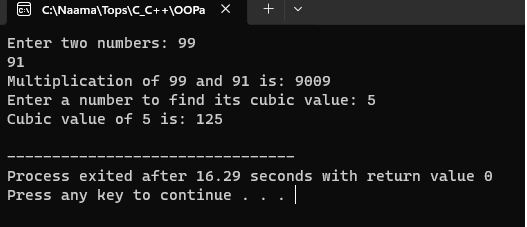
// Calculate and display cubic value

cout << "Cubic value of " << num3 << " is: " << cube(num3) << endl;

return 0;

}

Output:



**10.Write a program of Addition, Subtraction, Division, Multiplication using constructor**

Code:

#include <iostream>

using namespace std;

class Calculator {

private:

double num1, num2;

public:

Calculator(double n1, double n2) {

num1 = n1;

num2 = n2;

}

double addition() {

return num1 + num2;

}

double subtraction() {

return num1 - num2;

}

double division() {

if (num2 != 0)

return num1 / num2;

else {

cout << "Error! Division by zero.";

return -1;

}

}

double multiplication() {

return num1 \* num2;

}

};

int main() {

double num1, num2;

cout << "Enter two numbers: ";

cin >> num1 >> num2;

Calculator calc(num1, num2);

cout << "Addition: " << calc.addition() << endl;

cout << "Subtraction: " << calc.subtraction() << endl;

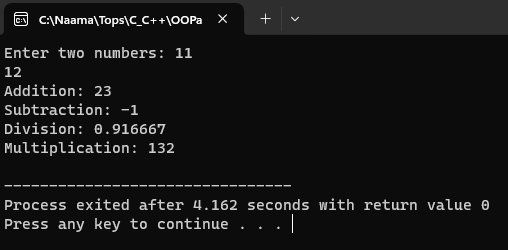
cout << "Division: " << calc.division() << endl;

cout << "Multiplication: " << calc.multiplication() << endl;

return 0;

}

Output:



**11.Write a C++ program to create a class called Car that has private member variables for company, model, and year. Implement member functions to get and set these variables.**

Code:

#include <iostream>

#include <string>

using namespace std;

class Car {

private:

string company;

string model;

int year;

public:

void setCompany(string c) {

company = c;

}

string getCompany() {

return company;

}

void setModel(string m) {

model = m;

}

string getModel() {

return model;

}

void setYear(int y) {

year = y;

}

int getYear() {

return year;

}

};

int main() {

Car car;

// Set values using setter functions

car.setCompany("Toyota");

car.setModel("Corolla");

car.setYear(2022);

// Get values using getter functions and display

cout << "Company: " << car.getCompany() << endl;

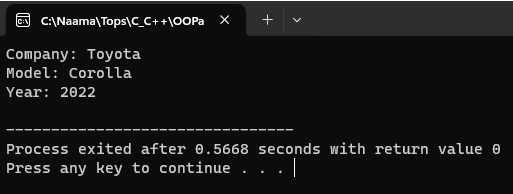
cout << "Model: " << car.getModel() << endl;

cout << "Year: " << car.getYear() << endl;

return 0;

}

Output:



**12.Write a C++ program to implement a class called Bank Account that has private member variables for account number and balance. Include member functions to deposit and withdraw money from the account**

Code:

#include <iostream>

using namespace std;

class BankAccount {

private:

int accountNumber;

double balance;

public:

// Constructor to initialize account number and balance

BankAccount(int accNum, double bal) {

accountNumber = accNum;

balance = bal;

}

// Function to deposit money

void deposit(double amount) {

balance += amount;

cout << "Deposited $" << amount << ". New balance: $" << balance << endl;

}

// Function to withdraw money

void withdraw(double amount) {

if (balance >= amount) {

balance -= amount;

cout << "Withdrawn $" << amount << ". New balance: $" << balance << endl;

} else {

cout << "Insufficient balance!" << endl;

}

}

// Function to get current balance

double getBalance() {

return balance;

}

};

int main() {

// Create a BankAccount object with initial balance

BankAccount account(123456, 1000);

// Deposit some money

account.deposit(500);

// Withdraw some money

account.withdraw(200);

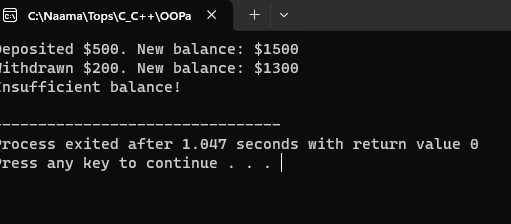
// Withdraw more money than available balance

account.withdraw(2000);

return 0;

}

Output:



**13.Write a C++ program to create a class called Triangle that has private member variables for the lengths of its three sides. Implement member functions to determine if the triangle is equilateral, isosceles, or scalene**.

Code:#include <iostream>

using namespace std;

class Triangle {

private:

double side1, side2, side3;

public:

// Constructor to initialize the lengths of the sides

Triangle(double s1, double s2, double s3) {

side1 = s1;

side2 = s2;

side3 = s3;

}

bool isEquilateral() {

return (side1 == side2 && side2 == side3);

}

bool isIsosceles() {

return (side1 == side2 || side1 == side3 || side2 == side3);

}

bool isScalene() {

return (!isEquilateral() && !isIsosceles());

}

};

int main() {

// Create a Triangle object with side lengths

Triangle triangle(5, 5, 5);

// Check the type of triangle and display the result

if (triangle.isEquilateral()) {

cout << "The triangle is equilateral." << endl;

} else if (triangle.isIsosceles()) {

cout << "The triangle is isosceles." << endl;

} else if (triangle.isScalene()) {

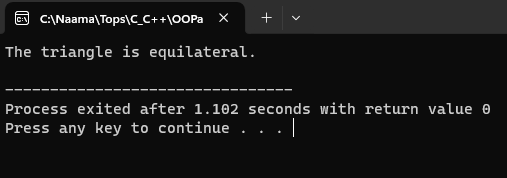
cout << "The triangle is scalene." << endl;

}

return 0;

}

Output:



**14.Write a C++ program to implement a class called Employee that has private member variables for name, employee ID, and salary. Include member functions to calculate and set salary based on employee performance. Using of constructor**

Code:

#include <iostream>

#include <string>

using namespace std;

class Employee {

private:

string name;

int employeeID;

double salary;

public:

Employee(string n, int id, double sal) {

name = n;

employeeID = id;

salary = sal;

}

void setSalary(double performance) {

if (performance >= 0 && performance <= 1) {

salary \*= performance;

} else {

cout << "Invalid performance value!" << endl;

}

}

void displayDetails() {

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

cout << "Employee ID: " << employeeID << endl;

cout << "Salary: $" << salary << endl;

}

};

int main() {

// Create an Employee object with initial details

Employee emp("John Doe", 123456, 50000);

// Display initial employee details

cout << "Initial Employee Details:" << endl;

emp.displayDetails();

// Set salary based on performance

emp.setSalary(0.9); // Assume 90% performance

// Display updated employee details

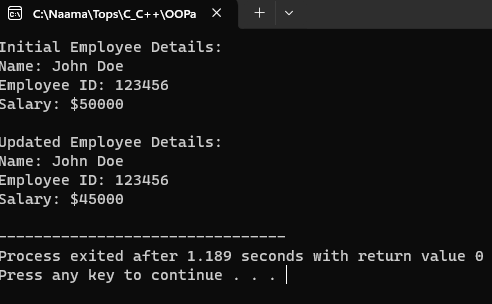
cout << "\nUpdated Employee Details:" << endl;

emp.displayDetails();

return 0;

}

Output:



**15.Write a C++ program to implement a class called Date that has private member variables for day, month, and year. Include member functions to set and get these variables, as well as to validate if the date is valid**

Code:#include <iostream>

using namespace std;

class Date {

private:

int day, month, year;

public:

// Constructor to initialize day, month, and year

Date(int d, int m, int y) {

day = d;

month = m;

year = y;

}

// Function to set the day

void setDay(int d) {

day = d;

}

// Function to get the day

int getDay() {

return day;

}

// Function to set the month

void setMonth(int m) {

month = m;

}

// Function to get the month

int getMonth() {

return month;

}

// Function to set the year

void setYear(int y) {

year = y;

}

// Function to get the year

int getYear() {

return year;

}

// Function to validate if the date is valid

bool isValidDate() {

if (year < 0) {

return false;

}

if (month < 1 || month > 12) {

return false;

}

if (day < 1 || day > 31) {

return false;

}

if ((month == 4 || month == 6 || month == 9 || month == 11) && day > 30) {

return false;

}

if (month == 2) {

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {

if (day > 29) {

return false;

}

} else {

if (day > 28) {

return false;

}

}

}

return true;

}

};

int main() {

// Create a Date object with initial values

Date date(31, 12, 2023);

// Display the initial date

cout << "Initial Date:" << endl;

cout << "Day: " << date.getDay() << endl;

cout << "Month: " << date.getMonth() << endl;

cout << "Year: " << date.getYear() << endl;

// Validate the date

if (date.isValidDate()) {

cout << "The date is valid." << endl;

} else {

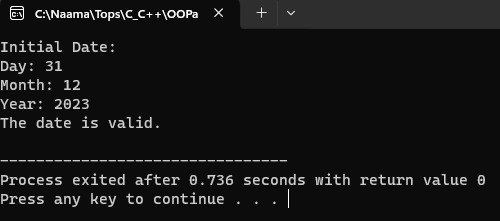
cout << "The date is not valid." << endl;

}

return 0;

}

Output:



**16. Write a C++ program to implement a class called Student that has private member variables for name, class, roll number, and marks. Include member functions to calculate the grade based on the marks and display the student's information. Accept address from each student implement using of aggregation**

Code:#include <iostream>

#include <string>

using namespace std;

class Address {

private:

string street;

string city;

string state;

string zipCode;

public:

Address(string s, string c, string st, string zip) {

street = s;

city = c;

state = st;

zipCode = zip;

}

void displayAddress() {

cout << "Address: " << street << ", " << city << ", " << state << " - " << zipCode << endl;

}

};

class Student {

private:

string name;

string className;

int rollNumber;

int marks;

char grade;

Address address;

public:

Student(string n, string c, int roll, int m, char g, Address addr) : address(addr) {

name = n;

className = c;

rollNumber = roll;

marks = m;

grade = g;

}

void calculateGrade() {

if (marks >= 90) {

grade = 'A';

} else if (marks >= 80) {

grade = 'B';

} else if (marks >= 70) {

grade = 'C';

} else if (marks >= 60) {

grade = 'D';

} else {

grade = 'F';

}

}

void displayStudentInfo() {

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

cout << "Class: " << className << endl;

cout << "Roll Number: " << rollNumber << endl;

cout << "Marks: " << marks << endl;

cout << "Grade: " << grade << endl;

address.displayAddress();

}

};

int main() {

// Create an Address object for each student

Address addr1("123 Main St", "Cityvila", "Giftcity", "12345");

Address addr2("456 ATV St", "Townvila", "Giftcity", "54321");

// Create Student objects with initial details and addresses

Student student1("Priya", "XII", 101, 85, 'B', addr1);

Student student2("raj", "X", 201, 75, 'C', addr2);

// Calculate grades for students

student1.calculateGrade();

student2.calculateGrade();

// Display student information

cout << "Student 1 Information:" << endl;

student1.displayStudentInfo();

cout << endl;

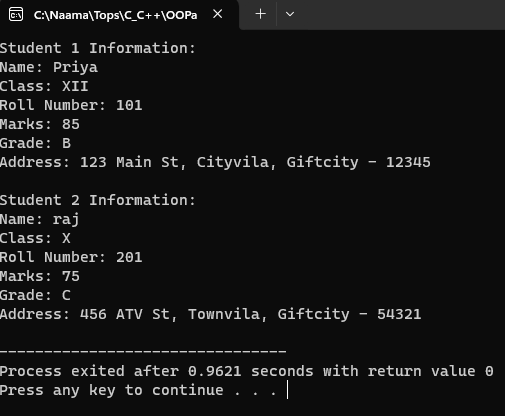
cout << "Student 2 Information:" << endl;

student2.displayStudentInfo();

return 0;

}

Output:



**17.Assume a class cricketer is declared. Declare a derived class batsman fromcricketer. Data member of batsman. Total runs, Average runs and best performance. Member functions input data, calculate average runs, Display data.(Single Inheritance)**

Code:#include <iostream>

#include <string>

using namespace std;

class Cricketer {

protected:

string name;

int age;

public:

void inputData() {

cout << "Enter name: ";

cin >> name;

cout << "Enter age: ";

cin >> age;

}

void displayData() {

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

cout << "Age: " << age << endl;

}

};

class Batsman : public Cricketer {

private:

int totalRuns;

float averageRuns;

int bestPerformance;

public:

void inputData() {

Cricketer::inputData(); // Input base class data

cout << "Enter total runs: ";

cin >> totalRuns;

cout << "Enter best performance: ";

cin >> bestPerformance;

}

void calculateAverageRuns() {

averageRuns = static\_cast<float>(totalRuns) / 5; // Assuming 5 matches

}

void displayData() {

Cricketer::displayData(); // Display base class data

cout << "Total Runs: " << totalRuns << endl;

cout << "Average Runs: " << averageRuns << endl;

cout << "Best Performance: " << bestPerformance << endl;

}

};

int main() {

Batsman batsman;

// Input batsman data

cout << "Enter Batsman's information:" << endl;

batsman.inputData();

// Calculate average runs

batsman.calculateAverageRuns();

// Display batsman data

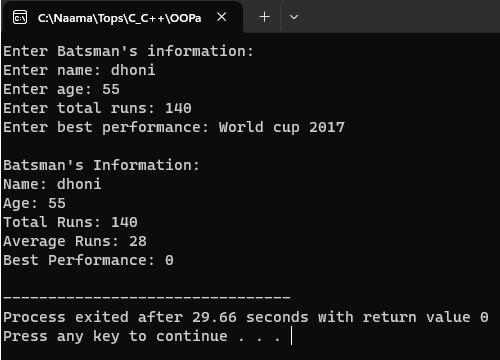
cout << "\nBatsman's Information:" << endl;

batsman.displayData();

return 0;

}

Output:



**18.Write a C++ Program to find Area of Rectangle using inheritance**

Code:#include <iostream>

using namespace std;

class Rectangle {

protected:

float length;

float width;

public:

void inputDimensions() {

cout << "Enter length: ";

cin >> length;

cout << "Enter width: ";

cin >> width;

}

float calculateArea() {

return length \* width;

}

};

class RectangleArea : public Rectangle {

public:

// display area of rectangle

void displayArea() {

cout << "Area of Rectangle: " << calculateArea() << endl;

}

};

int main() {

RectangleArea rectangle;

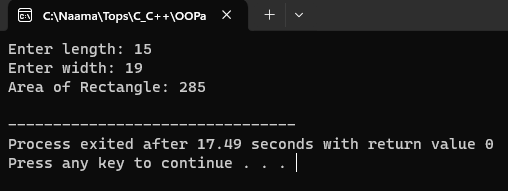
rectangle.inputDimensions();

rectangle.displayArea();

return 0;

}

Output:



**19.Create a class person having members name and age. Derive a class studenthaving member percentage. Derive another class teacher having member salary.Write necessary member function to initialize, read and write data. Write also Main function (Multiple Inheritance)**

Code:#include <iostream>

#include <string>

using namespace std;

class Person {

protected:

string name;

int age;

public:

void initializeData() {

cout << "Enter name: ";

cin >> name;

cout << "Enter age: ";

cin >> age;

}

void displayData() {

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

cout << "Age: " << age << endl;

}

};

class Student : public Person {

protected:

float percentage;

public:

void readStudentData() {

initializeData();

cout << "Enter percentage: ";

cin >> percentage;

}

void displayStudentData() {

displayData();

cout << "Percentage: " << percentage << "%" << endl;

}

};

class Teacher : public Person {

protected:

float salary;

public:

// Function to read teacher data

void readTeacherData() {

initializeData();

cout << "Enter salary: ";

cin >> salary;

}

// Function to display teacher data

void displayTeacherData() {

displayData();

cout << "Salary: $" << salary << endl;

}

};

int main() {

Student student;

Teacher teacher;

cout << "Enter student's information:" << endl;

student.readStudentData();

cout << "\nEnter teacher's information:" << endl;

teacher.readTeacherData();

cout << "\nStudent's Information:" << endl;

student.displayStudentData();

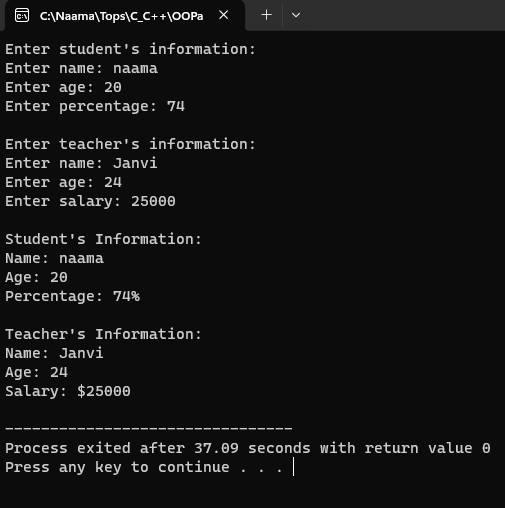
cout << "\nTeacher's Information:" << endl;

teacher.displayTeacherData();

return 0;

}

Output:



**20.Write a C++ Program display Student Mark sheet using Multiple inheritance**

Code:#include <iostream>

#include <string>

using namespace std;

class PersonalDetails {

protected:

string name;

int rollNumber;

public:

void inputPersonalDetails() {

cout << "Enter name: ";

cin >> name;

cout << "Enter roll number: ";

cin >> rollNumber;

}

// Function to display personal details

void displayPersonalDetails() {

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

cout << "Roll Number: " << rollNumber << endl;

}

};

class AcademicDetails {

protected:

float marks[5];

public:

void inputMarks() {

cout << "Enter marks for 5 subjects: ";

for (int i = 0; i < 5; ++i) {

cin >> marks[i];

}

}

// Function to display marks

void displayMarks() {

cout << "Marks:" << endl;

for (int i = 0; i < 5; ++i) {

cout << "Subject " << i + 1 << ": " << marks[i] << endl;

}

}

};

class MarkSheet : public PersonalDetails, public AcademicDetails {

public:

void displayMarkSheet() {

cout << "\nStudent Mark Sheet:" << endl;

displayPersonalDetails();

displayMarks();

}

};

int main() {

MarkSheet student;

student.inputPersonalDetails();

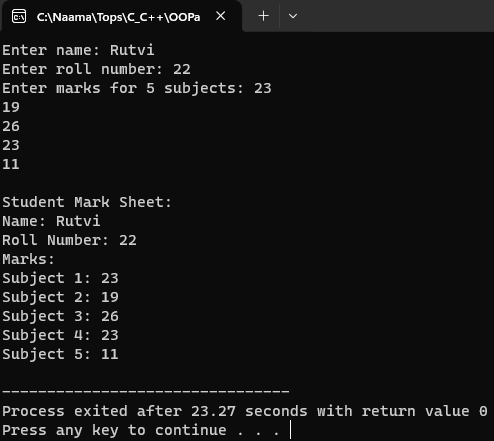
student.inputMarks();

student.displayMarkSheet();

return 0;

}

Output:



**21.Assume that the test results of a batch of students are stored in three different classes. Class Students are storing the roll number. Class Test stores the marks obtained in two subjects and class result contains the total marks obtained in the test. The class result can inherit the details of the marks obtained in the test and roll number of students. (Multilevel Inheritance)**

Code:#include <iostream>

#include <string>

using namespace std;

// Base class for student details

class Students {

protected:

int rollNumber;

public:

// Function to input student details

void inputStudentDetails() {

cout << "Enter roll number: ";

cin >> rollNumber;

}

// Function to display student details

void displayStudentDetails() {

cout << "Roll Number: " << rollNumber << endl;

}

};

// Intermediate class for test details

class Test : public Students {

protected:

float marksSubject1, marksSubject2;

public:

// Function to input test marks

void inputTestMarks() {

cout << "Enter marks for Subject 1: ";

cin >> marksSubject1;

cout << "Enter marks for Subject 2: ";

cin >> marksSubject2;

}

// Function to display test marks

void displayTestMarks() {

cout << "Marks for Subject 1: " << marksSubject1 << endl;

cout << "Marks for Subject 2: " << marksSubject2 << endl;

}

};

// Derived class for result

class Result : public Test {

protected:

float totalMarks;

public:

// Function to calculate total marks

void calculateTotalMarks() {

totalMarks = marksSubject1 + marksSubject2;

}

// Function to display result

void displayResult() {

cout << "Total Marks: " << totalMarks << endl;

}

};

int main() {

Result studentResult;

// Input student details and test marks

studentResult.inputStudentDetails();

studentResult.inputTestMarks();

// Calculate total marks

studentResult.calculateTotalMarks();

// Display student details, test marks, and result

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

studentResult.displayStudentDetails();

cout << "\nTest Marks:" << endl;

studentResult.displayTestMarks();

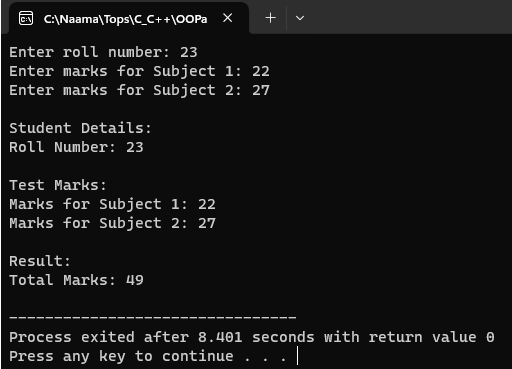
cout << "\nResult:" << endl;

studentResult.displayResult();

return 0;

}

Output:



**22.Write a C++ Program to show access to Private Public and Protected using Inheritance**

Code:#include <iostream>

using namespace std;

// Base class

class Base {

private:

int privateData;

protected:

int protectedData;

public:

int publicData;

Base() {

privateData = 10;

protectedData = 20;

publicData = 30;

}

void display() {

cout << "Private Data (Base): " << privateData << endl;

cout << "Protected Data (Base): " << protectedData << endl;

cout << "Public Data (Base): " << publicData << endl;

}

};

// Derived class

class Derived : public Base {

public:

void accessBaseMembers() {

// Accessing public data from base class

cout << "Public Data (Derived): " << publicData << endl;

// Accessing protected data from base class

cout << "Protected Data (Derived): " << protectedData << endl;

// Private members of base class cannot be accessed directly in derived class

// cout << "Private Data (Derived): " << privateData << endl; // Error: 'int Base::privateData' is private within this context

}

};

int main() {

Derived obj;

// Accessing public data from derived class object

cout << "Public Data (Main): " << obj.publicData << endl;

// Protected and Private members are not accessible outside the class scope

// cout << "Protected Data (Main): " << obj.protectedData << endl; // Error: 'int Base::protectedData' is protected within this context

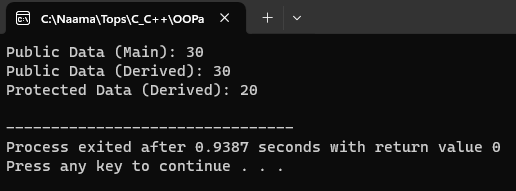
// cout << "Private Data (Main): " << obj.privateData << endl; // Error: 'int Base::privateData' is private within this context

obj.accessBaseMembers();

return 0;

}

Output:



**23.Write a C++ Program to illustrates the use of Constructors in multilevel inheritanceCode:**

Code:#include <iostream>

using namespace std;

// Base class

class Base {

protected:

int baseValue;

public:

// Constructor with parameter

Base(int value) {

baseValue = value;

cout << "Base constructor called with value: " << baseValue << endl;

}

};

// Derived class

class Derived : public Base {

protected:

int derivedValue;

public:

// Constructor with parameters

Derived(int base, int derived) : Base(base) {

derivedValue = derived;

cout << "Derived constructor called with value: " << derivedValue << endl;

}

};

// Further derived class

class FurtherDerived : public Derived {

public:

// Constructor with parameters

FurtherDerived(int base, int derived, int further) : Derived(base, derived) {

cout << "FurtherDerived constructor called with value: " << further << endl;

}

};

int main() {

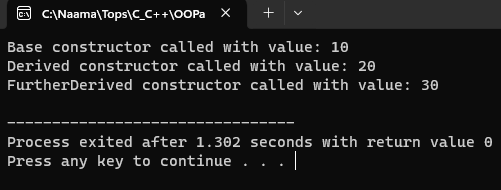
// Creating object of FurtherDerived class

FurtherDerived obj(10, 20, 30);

return 0;

}

Output:



**24.Write a program to Mathematic operation like Addition, Subtraction, Multiplication, Division Of two number using different parameters and Function Overloading**

Code:#include <iostream>

using namespace std;

// addition

int mathOperation(int a, int b) {

return a + b;

}

// overloading addition

double mathOperation(double a, double b) {

return a + b;

}

// subtraction

int mathOperation(int a, int b, int c) {

return a - b - c;

}

// multiplication

int mathOperation(int a, int b, int c, int d) {

return a \* b \* c \* d;

}

// division

double mathOperation(double a, double b, double c) {

return (a / b) / c;

}

int main() {

int intResult1 = mathOperation(10, 5); // Calling addition function with integer parameters

double doubleResult1 = mathOperation(10.5, 5.3); // Calling addition function with double parameters

int intResult2 = mathOperation(20, 5, 3); // Calling subtraction function with integer parameters

int intResult3 = mathOperation(2, 3, 4, 5); // Calling multiplication function with integer parameters

double doubleResult2 = mathOperation(100.0, 5.0, 2.0); // Calling division function with double parameters

cout << "Addition (int): " << intResult1 << endl;

cout << "Addition (double): " << doubleResult1 << endl;

cout << "Subtraction: " << intResult2 << endl;

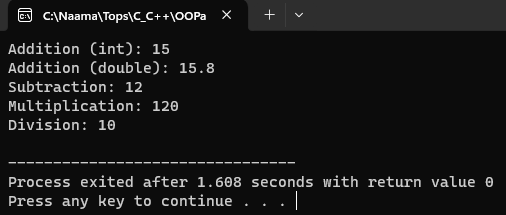
cout << "Multiplication: " << intResult3 << endl;

cout << "Division: " << doubleResult2 << endl;

return 0;

}

Output:



**26.Write a program to calculate the area of circle, rectangle and triangle using Function Overloading**

Code:#include <iostream>

#include <cmath>

using namespace std;

// Function to calculate the area of a circle

float area(float radius) {

return M\_PI \* radius \* radius; // M\_PI is a constant defined in the cmath library for pi

}

// Function to calculate the area of a rectangle

float area(float length, float width) {

return length \* width;

}

// Function to calculate the area of a triangle

float area(float base, float height, char) { // Using a dummy parameter to distinguish from rectangle

return 0.5 \* base \* height;

}

int main() {

float radius, length, width, base, height;

// Calculate area of a circle

cout << "Enter the radius of the circle: ";

cin >> radius;

cout << "Area of the circle: " << area(radius) << endl;

// Calculate area of a rectangle

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

cin >> length >> width;

cout << "Area of the rectangle: " << area(length, width) << endl;

// Calculate area of a triangle

cout << "Enter the base and height of the triangle: ";

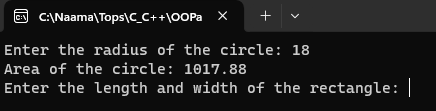
cin >> base >> height;

cout << "Area of the triangle: " << area(base, height, ' ') << endl;

return 0;

}

Output:



**27.Write a programto swap the two numbers using friend function without using third variable**

Code:#include <iostream>

using namespace std;

class SwapNumbers {

private:

int num1, num2;

public:

// Constructor to initialize num1 and num2

SwapNumbers(int n1, int n2) {

num1 = n1;

num2 = n2;

}

// Declaration of friend function

friend void swap(SwapNumbers&);

// Function to display numbers

void display() {

cout << "After swapping: num1 = " << num1 << ", num2 = " << num2 << endl;

}

};

// Definition of friend function to swap numbers

void swap(SwapNumbers& obj) {

obj.num1 = obj.num1 + obj.num2;

obj.num2 = obj.num1 - obj.num2;

obj.num1 = obj.num1 - obj.num2;

}

int main() {

int n1, n2;

// Input numbers from user

cout << "Enter first number: ";

cin >> n1;

cout << "Enter second number: ";

cin >> n2;

// Create object of SwapNumbers class

SwapNumbers obj(n1, n2);

// Display original numbers

cout << "Before swapping: num1 = " << n1 << ", num2 = " << n2 << endl;

// Call friend function to swap numbers

swap(obj);

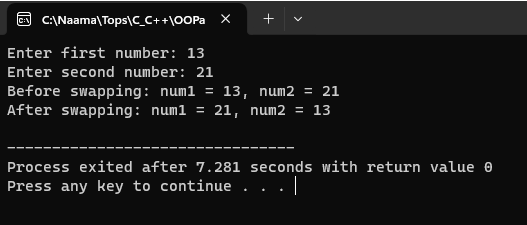
// Display swapped numbers

obj.display();

return 0;

}

Output:



**28.Write a program to find the max number from given two numbers using friend function**

Code:#include <iostream>

using namespace std;

// Forward declaration of class

class MaxNumber;

// Declaration of friend function to find maximum number

int findMax(const MaxNumber&);

// Class definition

class MaxNumber {

private:

int num1, num2;

public:

// Constructor to initialize num1 and num2

MaxNumber(int n1, int n2) {

num1 = n1;

num2 = n2;

}

// Declaration of friend function

friend int findMax(const MaxNumber&);

};

// Definition of friend function to find maximum number

int findMax(const MaxNumber& obj) {

return (obj.num1 > obj.num2) ? obj.num1 : obj.num2;

}

int main() {

int n1, n2;

// Input two numbers from user

cout << "Enter first number: ";

cin >> n1;

cout << "Enter second number: ";

cin >> n2;

// Create object of MaxNumber class

MaxNumber obj(n1, n2);

// Find maximum number using friend function

int maxNum = findMax(obj);

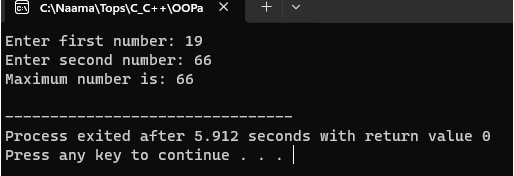
// Display maximum number

cout << "Maximum number is: " << maxNum << endl;

return 0;

}

Output:



**29.Write a program of to swap the two values using template**

Code:#include <iostream>

using namespace std;

template <typename T>

void swapValues(T& a, T& b) {

T temp = a;

a = b;

b = temp;

}

int main() {

int num1 = 5, num2 = 10;

cout << "Before swapping: num1 = " << num1 << ", num2 = " << num2 << endl;

swapValues(num1, num2);

cout << "After swapping: num1 = " << num1 << ", num2 = " << num2 << endl;

char char1 = 'A', char2 = 'B';

cout << "Before swapping: char1 = " << char1 << ", char2 = " << char2 << endl;

swapValues(char1, char2);

cout << "After swapping: char1 = " << char1 << ", char2 = " << char2 << endl;

// Swap doubles

double dbl1 = 3.14, dbl2 = 6.28;

cout << "Before swapping: dbl1 = " << dbl1 << ", dbl2 = " << dbl2 << endl;

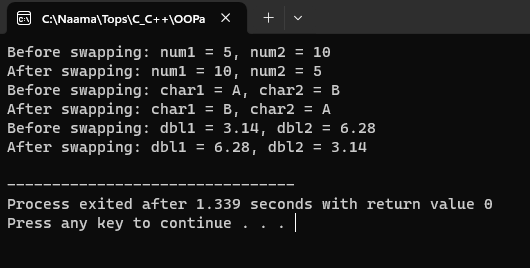
swapValues(dbl1, dbl2);

cout << "After swapping: dbl1 = " << dbl1 << ", dbl2 = " << dbl2 << endl;

return 0;

}

Output:



**30.Write a program of to sort the array using templates**

Code:#include <iostream>

using namespace std;

// Function template to sort an array

template <typename T, int size>

void sortArray(T (&arr)[size]) {

for (int i = 0; i < size - 1; ++i) {

for (int j = i + 1; j < size; ++j) {

if (arr[i] > arr[j]) {

// Swap elements if they are out of order

T temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

}

// Function to print an array

template <typename T, int size>

void printArray(T (&arr)[size]) {

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

cout << arr[i] << " ";

}

cout << endl;

}

int main() {

int intArray[] = {5, 2, 9, 1, 7};

cout << "Original array (integers): ";

printArray(intArray);

sortArray(intArray);

cout << "Sorted array (integers): ";

printArray(intArray);

double doubleArray[] = {3.14, 2.71, 1.618, 2.718, 3.142};

cout << "Original array (doubles): ";

printArray(doubleArray);

sortArray(doubleArray);

cout << "Sorted array (doubles): ";

printArray(doubleArray);

return 0;

}

Output:

