Title : 1. Write a program to check whether a number is even or odd using if-else.

#include <iostream>

using namespace std;

int main() {

    int number;

    cout << "Enter an integer: ";

    cin >> number;

    if (number % 2 == 0) {

        cout << number << " is even." << endl;

    } else {

        cout << number << " is odd." << endl;

    }

    return 0;

}

OUTPUT :

Enter an integer: 7

7 is odd.

Title : 2 . Write a program to calculate the sum and average of three numbers using control structure

#include <iostream>

using namespace std;

int main() {

    // Declare variables

    float num1, num2, num3, sum, average;

    // Input three numbers

    cout << "Enter three numbers: ";

    cin >> num1 >> num2 >> num3;

    // Calculate sum

    sum = num1 + num2 + num3;

    // Calculate average

    average = sum / 3;

    // Output the results

    cout << "Sum: " << sum << endl;

    cout << "Average: " << average << endl;

    return 0;

}

OUTPUT :

Enter three numbers: 10 20 30

Sum: 60

Average: 20

Title : 3. Write a program to determine whether a number is prime or composite.

#include <iostream>

using namespace std;

bool isPrime(int n) {

    if (n <= 1) return false;

    for (int i = 2; i \* i <= n; i++) {

        if (n % i == 0) return false;

    }

    return true;

}

int main() {

    int num;

    cout << "Enter a number: ";

    cin >> num;

    if (isPrime(num)) {

        cout << num << " is a prime number." << endl;

    } else {

        cout << num << " is a composite number." << endl;

    }

    return 0;

}

OUTPUT :

Enter a number: 12

12 is a composite number.

Title : 4. Write a program to calculate the sum, difference, product, and quotient of two integers.

#include <iostream>

using namespace std;

int main() {

    int num1, num2;

    // Input two integers

    cout << "Enter the first integer: ";

    cin >> num1;

    cout << "Enter the second integer: ";

    cin >> num2;

    // Perform calculations

    int sum = num1 + num2;

    int difference = num1 - num2;

    int product = num1 \* num2;

    float quotient = 0;

    if (num2 != 0) {

        quotient = (float)num1 / num2;

    }

    // Display results

    cout << "Sum: " << sum << endl;

    cout << "Difference: " << difference << endl;

    cout << "Product: " << product << endl;

    if (num2 != 0) {

        cout << "Quotient: " << quotient << endl;

    } else {

        cout << "Quotient: Division by zero is not allowed." << endl;

    }

    return 0;

}

OUTPUT :

Enter the first integer: 12

Enter the second integer: 3

Sum: 15

Difference: 9

Product: 36

Quotient: 4

Title : 5. Write a program to demonstrate use of function overloading. (e.g., area of a circle, rectangle, and triangle).

#include <iostream>

#define \_USE\_MATH\_DEFINES

#include <cmath>

using namespace std;

// Function to calculate the area of a circle

double area(double radius) {

    return M\_PI \* radius \* radius;

}

// Function to calculate the area of a rectangle

double area(double length, double width) {

    return length \* width;

}

// Function to calculate the area of a triangle

double area(double base, double height, bool isTriangle) {

    if (isTriangle) {

        return 0.5 \* base \* height;

    }

    return 0.0; // Default return if not a triangle

}

int main() {

    // Area of a circle

    double circleRadius = 5.0;

    cout << "Area of Circle: " << area(circleRadius) << endl;

    // Area of a rectangle

    double rectangleLength = 4.0, rectangleWidth = 6.0;

    cout << "Area of Rectangle: " << area(rectangleLength, rectangleWidth) << endl;

    // Area of a triangle

    double triangleBase = 3.0, triangleHeight = 7.0;

    cout << "Area of Triangle: " << area(triangleBase, triangleHeight, true) << endl;

    return 0;

}

OUTPUT :

Area of Circle: 78.5398

Area of Rectangle: 24

Area of Triangle: 10.5

Title : Write a program to demonstrate encapsulation using of class

#include <iostream>

using namespace std;

class Student {

private:

    int age;  // Private variable (data hiding)

public:

    // Setter method to assign a value to age

    void setAge(int a) {

        if (a > 0) {

            age = a;

        } else {

            cout << "Invalid age!" << endl;

        }

    }

    // Getter method to access age

    int getAge() {

        return age;

    }

};

int main() {

    Student s1;

    s1.setAge(20);  // Setting age using setter method

    cout << "Student's age: " << s1.getAge() << endl;  // Getting age using getter method

    return 0;

}

OUTPUT :

Student's age: 20

Title : 7. Write a program to demonstrate the use of different types of constructors and a destructor in a class

#include <iostream>

using namespace std;

class Demo {

    private:

    int value;

    public:

    // Default constructor

    Demo() {

        value = 0;

        cout << "Default constructor called. Value initialized to 0." << endl;

    }

    // Parameterized constructor

    Demo(int val) {

        value = val;

        cout << "Parameterized constructor called. Value initialized to " << value << "." << endl;

    }

    // Copy constructor

    Demo(const Demo &obj) {

        value = obj.value;

        cout << "Copy constructor called. Value copied: " << value << "." << endl;

    }

    // Destructor

    ~Demo() {

        cout << "Destructor called. Value was: " << value << "." << endl;

    }

    // Function to display the value

    void display() {

        cout << "Value: " << value << endl;

    }

};

int main() {

    cout << "Creating object obj1 using default constructor." << endl;

    Demo obj1;

    obj1.display();

    cout << "\nCreating object obj2 using parameterized constructor." << endl;

    Demo obj2(42);

    obj2.display();

    cout << "\nCreating object obj3 using copy constructor." << endl;

    Demo obj3 = obj2;

    obj3.display();

    cout << "\nEnd of main function. Objects will be destroyed." << endl;

    return 0;

}

OUTPUT *:*

Creating object obj1 using default constructor.

Default constructor called. Value initialized to 0.

Value: 0

Creating object obj2 using parameterized constructor.

Parameterized constructor called. Value initialized to 42.

Value: 42

Creating object obj3 using copy constructor.

Copy constructor called. Value copied: 42.

Value: 42

End of main function. Objects will be destroyed.

Destructor called. Value was: 42.

Destructor called. Value was: 42.

Destructor called. Value was: 0.

Title : Write a program to demonstrate single inheritance.

#include <iostream>

using namespace std;

// Base class (Parent)

class Animal {

public:

    void speak() {

        cout << "Animal speaks" << endl;

    }

};

// Derived class (Child)

class Dog : public Animal {

public:

    void bark() {

        cout << "Dog barks" << endl;

    }

};

int main() {

    // Create object of derived class

    Dog myDog;

    // Call functions from both base and derived class

    myDog.speak();  // Inherited from Animal

    myDog.bark();   // Defined in Dog

    return 0;

}

Output :

Animal speaks

Dog barks

Title : Write a program to demonstrate multiple inheritance.

#include <iostream>

using namespace std;

// Base class 1

class Person {

public:

    void showPerson() {

        cout << "This is a Person." << endl;

    }

};

// Base class 2

class Worker {

public:

    void showWorker() {

        cout << "This is a Worker." << endl;

    }

};

// Derived class inheriting from both Person and Worker

class Engineer : public Person, public Worker {

public:

    void showEngineer() {

        cout << "This is an Engineer." << endl;

    }

};

int main() {

    Engineer e;

    // Accessing methods from both base classes and derived class

    e.showPerson();   // From Person

    e.showWorker();   // From Worker

    e.showEngineer(); // From Engineer

    return 0;

}

Output :

This is a Person.

This is a Worker.

This is an Engineer.

Title : Write a program to demonstrate use of unary operator overloading.

#include <iostream>

using namespace std;

class Number {

private:

    int value;

public:

    // Constructor

    Number(int v) {

        value = v;

    }

    // Overload unary minus (-) operator

    void operator- () {

        value = -value;

    }

    // Display function

    void display() {

        cout << "Value: " << value << endl;

    }

};

int main() {

    Number num(10);

    cout << "Before overloading:" << endl;

    num.display();

    -num; // Overloading - operator

    cout << "After overloading:" << endl;

    num.display();

    return 0;

}

Output :

Before overloading:

Value: 10

After overloading:

Value: -10

Title : Write a program to demonstrate use of binary operator overloading.

#include <iostream>

using namespace std;

class Complex {

private:

    float real, imag;

public:

    // Constructor

    Complex(float r = 0, float i = 0) {

        real = r;

        imag = i;

    }

    // Overload + operator

    Complex operator + (Complex obj) {

        Complex temp;

        temp.real = real + obj.real;

        temp.imag = imag + obj.imag;

        return temp;

    }

    // Display function

    void display() {

        cout << real << " + " << imag << "i" << endl;

    }

};

int main() {

    Complex c1(3.5, 2.5);

    Complex c2(1.5, 4.5);

    Complex result = c1 + c2;  // Calls overloaded + operator

    cout << "Result of addition: ";

    result.display();

    return 0;

}

Output :

Result of addition: 5 + 7i