Program1:

#include <iostream>

#include <cmath>

class Shape {

public:

// Default constructor

Shape() {}

// Destructor

~Shape() {}

// Pure virtual function to calculate area

virtual double calculateArea() = 0;

};

class Circle : public Shape {

private:

double radius;

public:

Circle(double r) : radius(r) {}

double calculateArea() override {

return M\_PI \* radius \* radius;

}

};

class Rectangle : public Shape {

private:

double length;

double width;

public:

Rectangle(double l, double w) : length(l), width(w) {}

double calculateArea() override {

return length \* width;

}

};

class Triangle : public Shape {

private:

double base;

double height;

public:

Triangle(double b, double h) : base(b), height(h) {}

double calculateArea() override {

return 0.5 \* base \* height;

}

};

int main() {

Circle circle(5);

Rectangle rectangle(4, 6);

Triangle triangle(3, 8);

// Calculating and displaying areas

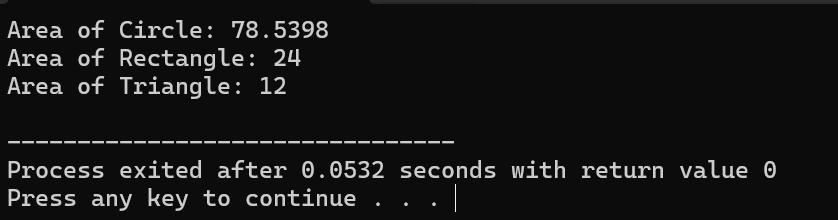
std::cout << "Area of Circle: " << circle.calculateArea() << std::endl;

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

std::cout << "Area of Triangle: " << triangle.calculateArea() << std::endl;

return 0;

}



Program 2:

#include <iostream>

#include <cmath>

class Shape {

public:

// Default constructor

Shape() {}

// Destructor

~Shape() {}

// Pure virtual function to calculate volume

virtual double calculateVolume() = 0;

};

class Cube : public Shape {

private:

double side;

public:

// Constructor with parameter

Cube(double s) : side(s) {}

// Method to calculate volume of cube

double calculateVolume() override {

return std::pow(side, 3);

}

};

class Cylinder : public Shape {

private:

double radius;

double height;

public:

Cylinder(double r, double h) : radius(r), height(h) {}

double calculateVolume() override {

return M\_PI \* std::pow(radius, 2) \* height;

}

};

int main() {

Cube cube(4);

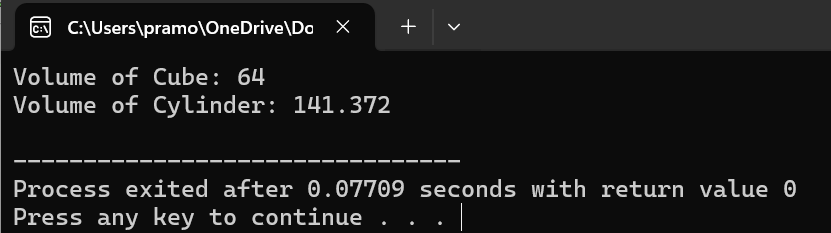
Cylinder cylinder(3, 5);

std::cout << "Volume of Cube: " << cube.calculateVolume() << std::endl;

std::cout << "Volume of Cylinder: " << cylinder.calculateVolume() << std::endl;

return 0;

}



Program 3:

#include <iostream>

class MyClass {

private:

int x;

int y;

public:

MyClass(int a, int b);

void displayValues();

};

MyClass::MyClass(int a, int b) : x(a), y(b) {}

void MyClass::displayValues() {

std::cout << "x = " << x << ", y = " << y << std::endl;

}

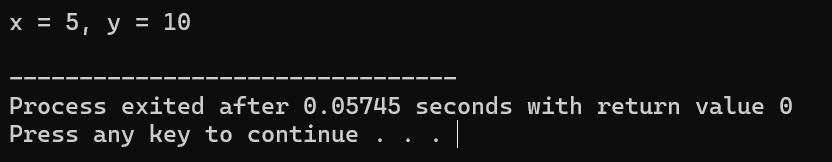
int main() {

MyClass obj(5, 10);

obj.displayValues();

return 0;

}



Program 4:

#include <iostream>

class FloydTriangle {

private:

int rows;

public:

FloydTriangle(int r) : rows(r) {}

~FloydTriangle() {

std::cout << "Destructor called. Memory released." << std::endl;

}

void printTriangle() {

int number = 1;

for (int i = 1; i <= rows; ++i) {

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

std::cout << number << " ";

++number;

}

std::cout << std::endl;

}

}

};

int main() {

int numRows;

std::cout << "Enter the number of rows for Floyd's Triangle: ";

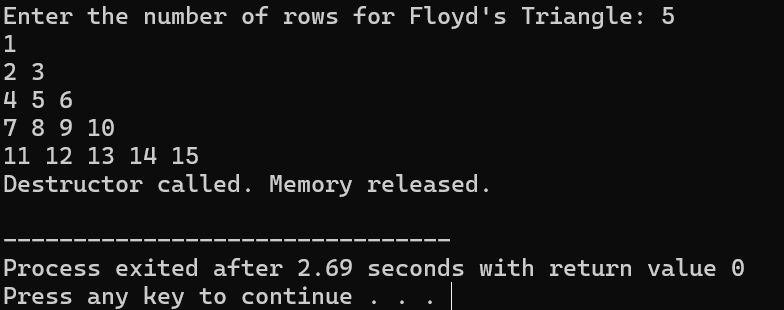
std::cin >> numRows;

FloydTriangle floydTriangle(numRows);

floydTriangle.printTriangle();

return 0;

}



Program 5:

#include <iostream>

class DecimalToBinary {

private:

long long int decimal;

long long int binary;

public:

DecimalToBinary() : decimal(0), binary(0) {}

DecimalToBinary(long long int dec) : decimal(dec) {

binary = convertToBinary();

}

long long int convertToBinary() {

long long int tempDecimal = decimal;

long long int binaryNumber = 0;

long long int base = 1;

while (tempDecimal > 0) {

int remainder = tempDecimal % 2;

binaryNumber += remainder \* base;

base \*= 10;

tempDecimal /= 2;

}

return binaryNumber;

}

void displayBinary() {

std::cout << "Binary representation of " << decimal << " is: " << binary << std::endl;

}

};

int main() {

long long int decimalNumber;

std::cout << "Enter a decimal number: ";

std::cin >> decimalNumber;

DecimalToBinary obj(decimalNumber);

obj.displayBinary();

return 0;

}

