**Medium Programs**

A.Ganesh reddy

192110023

11.

#include <iostream>

#include <cmath>

double calculateVolume(double side);

double calculateVolume(double length, double width, double height);

double calculateVolume(double radius, double height);

int main() {

int choice;

std::cout << "Choose a shape to calculate volume:\n";

std::cout << "1. Cube\n";

std::cout << "2. Rectangle\n";

std::cout << "3. Cylinder\n";

std::cin >> choice;

switch (choice) {

case 1: {

double side;

std::cout << "Enter the side length of the cube: ";

std::cin >> side;

std::cout << "Volume of the cube: " << calculateVolume(side) << std::endl;

break;

}

case 2: {

double length, width, height;

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

std::cin >> length >> width >> height;

std::cout << "Volume of the rectangle: " << calculateVolume(length, width, height) << std::endl;

break;

}

case 3: {

double radius, height;

std::cout << "Enter the radius and height of the cylinder: ";

std::cin >> radius >> height;

std::cout << "Volume of the cylinder: " << calculateVolume(radius, height) << std::endl;

break;

}

default:

std::cout << "Invalid choice!" << std::endl;

break;

}

return 0;

}

double calculateVolume(double side) {

return std::pow(side, 3);

}

double calculateVolume(double length, double width, double height) {

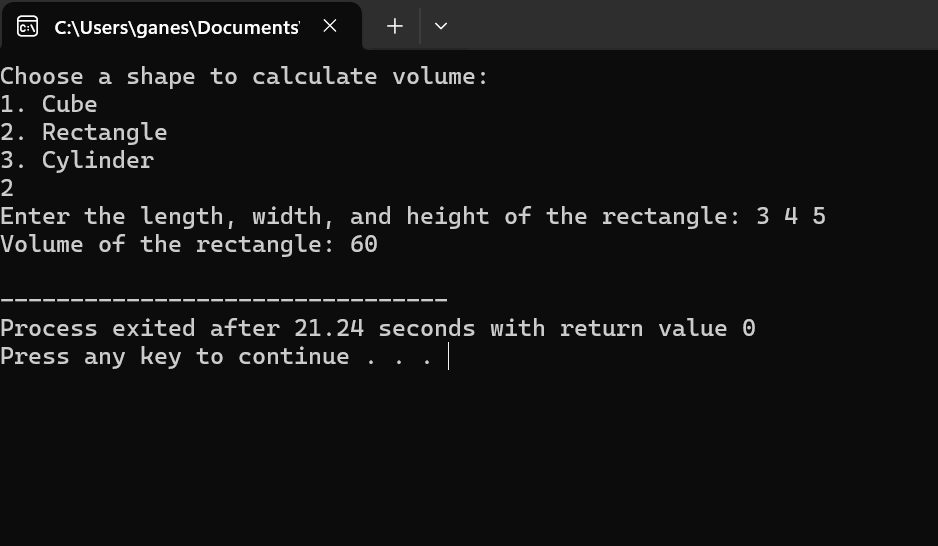
return length \* width \* height;

}

double calculateVolume(double radius, double height) {

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

}



12.

#include <iostream>

#include <string>

class Student {

private:

std::string name;

int id;

int grades[3];

public:

// Constructor

Student() : name(""), id(0) {

for (int i = 0; i < 3; ++i) {

grades[i] = 0;

}

}

void setName(const std::string& newName) {

name = newName;

}

void setId(int newId) {

id = newId;

}

void setGrades(int subject1, int subject2, int subject3) {

grades[0] = subject1;

grades[1] = subject2;

grades[2] = subject3;

}

std::string getName() const {

return name;

}

int getId() const {

return id;

}

int getGrade(int subjectIndex) const {

if (subjectIndex >= 0 && subjectIndex < 3)

return grades[subjectIndex];

else

return -1;

}

};

int main() {

Student student;

student.setName("John Doe");

student.setId(12345);

student.setGrades(85, 90, 88);

std::cout << "Student Name: " << student.getName() << std::endl;

std::cout << "Student ID: " << student.getId() << std::endl;

std::cout << "Grades:\n";

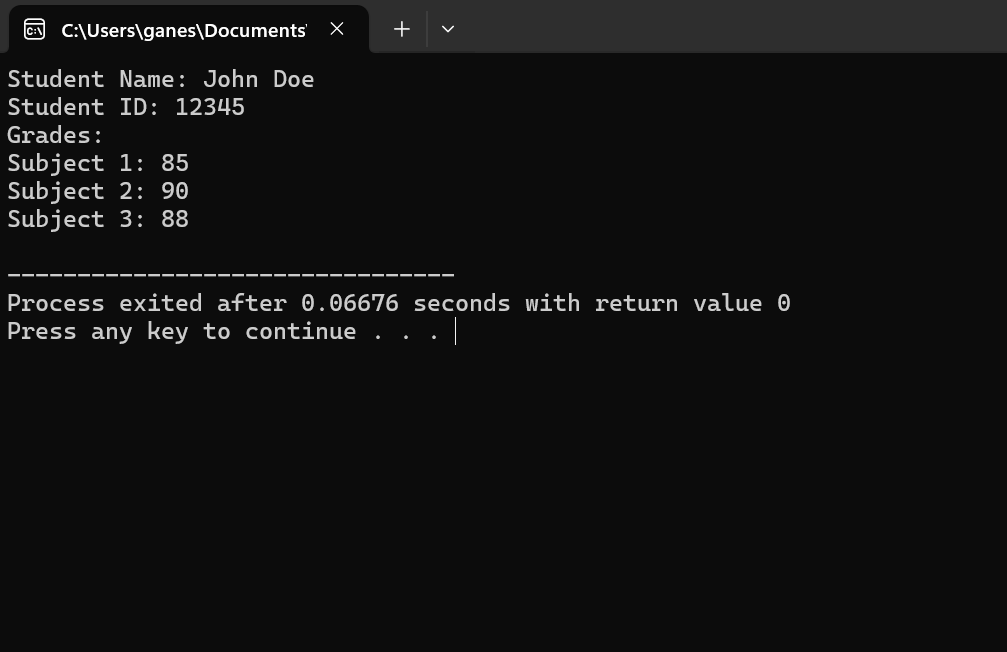
for (int i = 0; i < 3; ++i) {

std::cout << "Subject " << i+1 << ": " << student.getGrade(i) << std::endl;

}

return 0;

}



13.

#include <iostream>

inline double calculateArea(double length = 1.0, double width = 1.0) {

return length \* width;

}

int main() {

double length, width;

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

std::cin >> length;

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

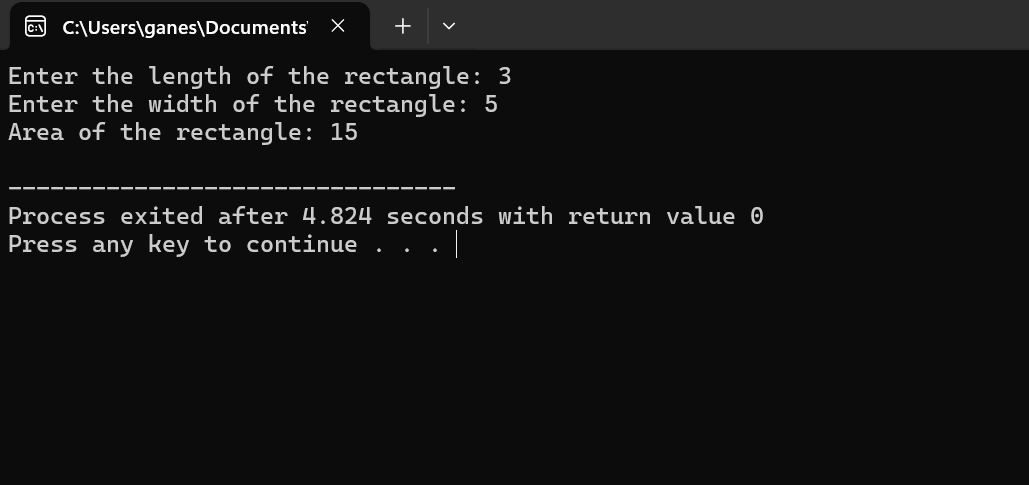
std::cin >> width;

double area = calculateArea(length, width);

std::cout << "Area of the rectangle: " << area << std::endl;

return 0;

}



14.

#include <iostream>

class Employee {

private:

std::string name;

double salary;

public:

Employee(const std::string& newName, double newSalary) : name(newName), salary(newSalary) {}

static double calculateAverageSalary(Employee\* employees, int numEmployees) {

double totalSalary = 0.0;

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

totalSalary += employees[i].salary;

}

return numEmployees > 0 ? totalSalary / numEmployees : 0.0;

}

};

int main() {

const int numEmployees = 3;

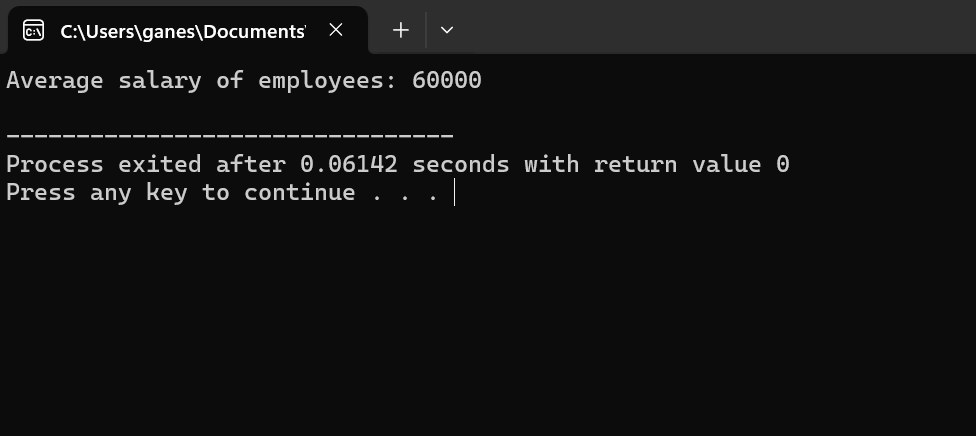
Employee employees[numEmployees] = {{"John", 50000.0}, {"Alice", 60000.0}, {"Bob", 70000.0}};

double averageSalary = Employee::calculateAverageSalary(employees, numEmployees);

std::cout << "Average salary of employees: " << averageSalary << std::endl;

return 0;

}



15.

#include <iostream>

class Complex {

private:

double real;

double imag;

public:

Complex(double r = 0.0, double i = 0.0) : real(r), imag(i) {}

friend class Matrix;

};

class Matrix {

public:

static Complex& multiplyWithConstant(Complex& complexNum, double constant) {

complexNum.real \*= constant;

complexNum.imag \*= constant;

return complexNum;

}

};

int main() {

Complex complexNum(3.0, 4.0);

std::cout << "Original Complex Number: " << complexNum.real << " + " << complexNum.imag << "i" << std::endl;

Matrix::multiplyWithConstant(complexNum, 2.0);

std::cout << "Modified Complex Number: " << complexNum.real << " + " << complexNum.imag << "i" << std::endl;

return 0;

}

16.

#include <iostream>

void swap(int& a, int& b);

void swap(int arr1[], int arr2[], int size);

void swap(int& a, int& b) {

int temp = a;

a = b;

b = temp;

}

void swap(int arr1[], int arr2[], int size) {

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

int temp = arr1[i];

arr1[i] = arr2[i];

arr2[i] = temp;

}

}

int main() {

int x = 5, y = 10;

std::cout << "Before swapping integers: x = " << x << ", y = " << y << std::endl;

swap(x, y);

std::cout << "After swapping integers: x = " << x << ", y = " << y << std::endl;

const int size = 3;

int arr1[size] = {1, 2, 3};

int arr2[size] = {4, 5, 6};

std::cout << "Before swapping arrays:\n";

std::cout << "Array 1: ";

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

std::cout << arr1[i] << " ";

}

std::cout << std::endl;

std::cout << "Array 2: ";

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

std::cout << arr2[i] << " ";

}

std::cout << std::endl;

swap(arr1, arr2, size);

std::cout << "After swapping arrays:\n";

std::cout << "Array 1: ";

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

std::cout << arr1[i] << " ";

}

std::cout << std::endl;

std::cout << "Array 2: ";

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

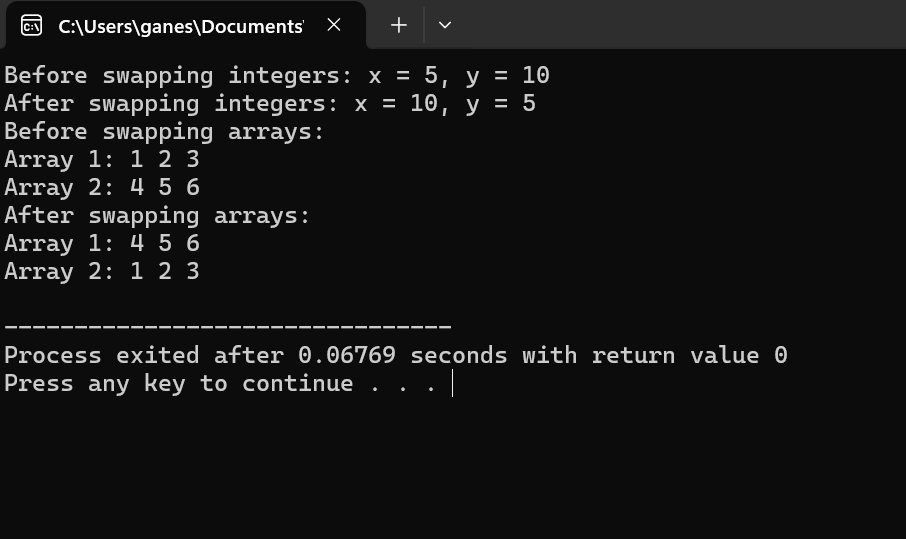
std::cout << arr2[i] << " ";

}

std::cout << std::endl;

return 0;

}



17.

#include <iostream>

class Circle {

private:

double radius;

double area;

public:

Circle() : radius(0.0), area(0.0) {}

void setRadius(double r) {

radius = r;

}

void calculateArea() {

area = 3.14159 \* radius \* radius;

}

void displayDetails() {

std::cout << "Circle Details:" << std::endl;

std::cout << "Radius: " << radius << std::endl;

std::cout << "Area: " << area << std::endl;

}

};

int main() {

Circle circle;

double r;

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

std::cin >> r;

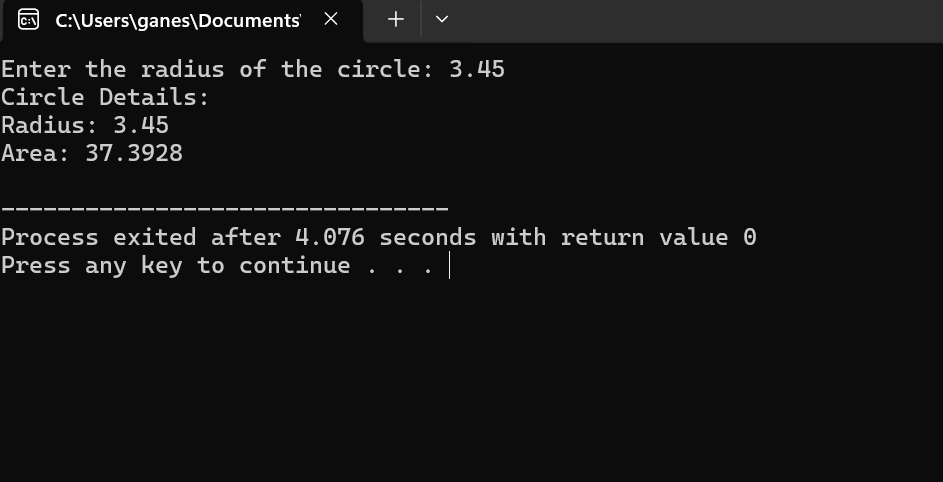
circle.setRadius(r);

circle.calculateArea();

circle.displayDetails();

return 0;

}



18.

#include <iostream>

#include <string>

class Car {

private:

std::string model;

int year;

double price;

public:

Car(const std::string& modelName, int carYear = 0, double carPrice = 0.0)

: model(modelName), year(carYear), price(carPrice) {}

void displayDetails() const {

std::cout << "Model: " << model << ", Year: " << year << ", Price: $" << price << std::endl;

}

};

int main() {

const int numCars = 3;

Car cars[numCars] = {

{"Toyota Camry", 2018, 25000.0},

{"Honda Accord"},

{"Ford Mustang", 2015}

};

std::cout << "Car Details:" << std::endl;

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

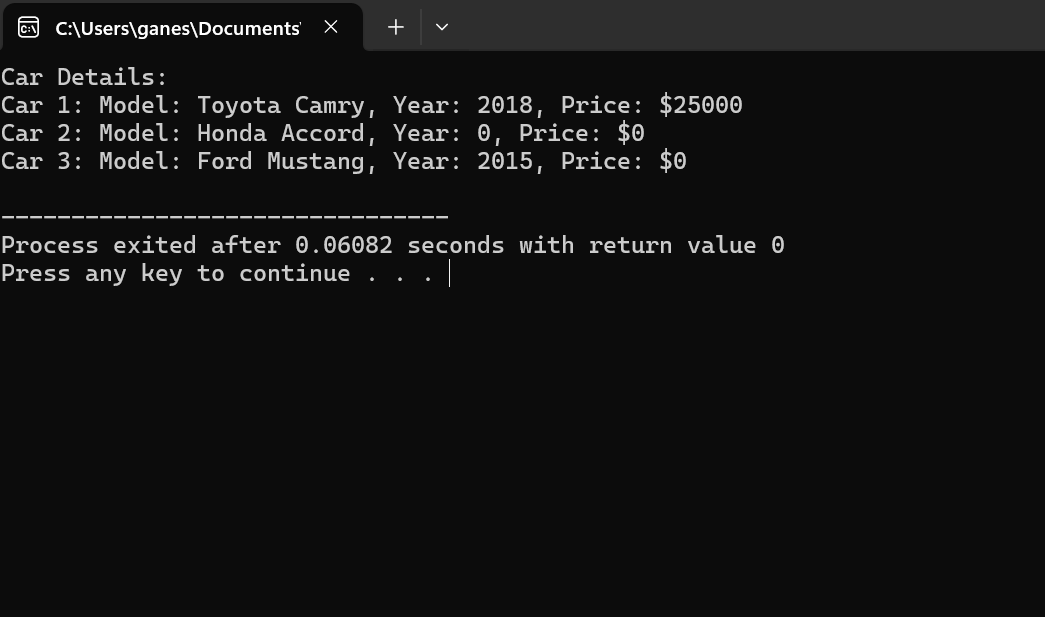
std::cout << "Car " << i + 1 << ": ";

cars[i].displayDetails();

}

return 0;

}



19.

#include <iostream>

#include <string>

class Car {

private:

std::string model;

int year;

double price;

public:

Car(const std::string& modelName, int carYear = 0, double carPrice = 0.0)

: model(modelName), year(carYear), price(carPrice) {}

void displayDetails() const {

std::cout << "Model: " << model << ", Year: " << year << ", Price: $" << price << std::endl;

}

};

int main() {

const int numCars = 3;

Car cars[numCars] = {

{"Toyota Camry", 2018, 25000.0},

{"Honda Accord"},

{"Ford Mustang", 2015}

};

std::cout << "Car Details:" << std::endl;

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

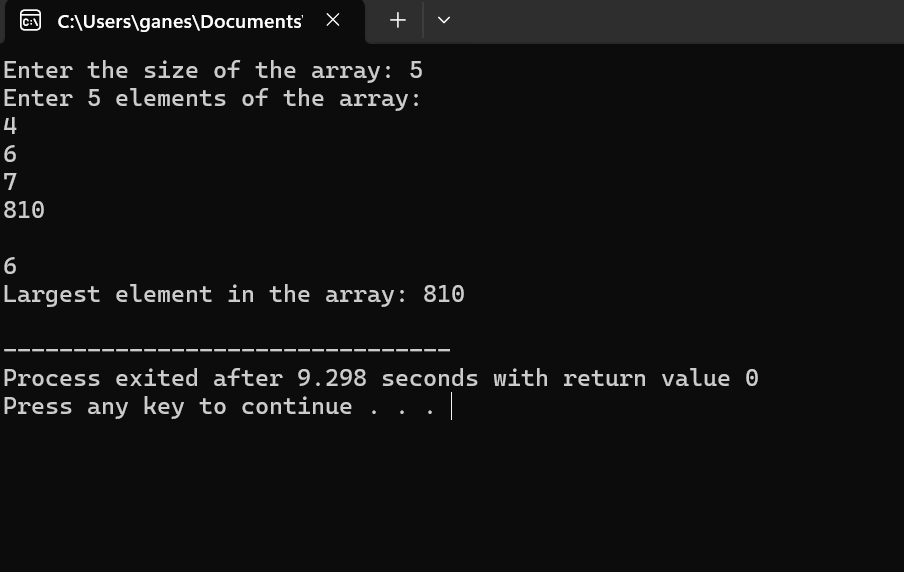
std::cout << "Car " << i + 1 << ": ";

cars[i].displayDetails();

}

return 0;

}



20. #include <iostream>

double calculateArea(double length = 1.0, double width = 1.0);

double calculatePerimeter(double length = 1.0, double width = 1.0);

int main() {

double length, width;

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

std::cin >> length;

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

std::cin >> width;

double area = calculateArea(length, width);

double perimeter = calculatePerimeter(length, width);

std::cout << "Area of the rectangle: " << area << std::endl;

std::cout << "Perimeter of the rectangle: " << perimeter << std::endl;

return 0;

}

double calculateArea(double length, double width) {

return length \* width;

}

double calculatePerimeter(double length, double width) {

return 2 \* (length + width);

}

