1. **Program**

#include <iostream>

#include <string>

using namespace std;

class Student {

public:

string name;

int rollNo;

char grade;

void getData() {

cout << "Enter name: ";

cin >> name;

cout << "Enter roll number: ";

cin >> rollNo;

cout << "Enter grade: ";

cin >> grade;

}

void displayData() {

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

cout << "Roll No.: " << rollNo << endl;

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

}

};

int main() {

const int numStudents = 3;

Student students[numStudents];

cout << "Enter data for " << numStudents << " students:" << endl;

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

cout << "Student " << i + 1 << ":" << endl;

students[i].getData();

}

cout << "Displaying data of students:" << endl;

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

cout << "Student " << i + 1 << ":" << endl;

students[i].displayData();

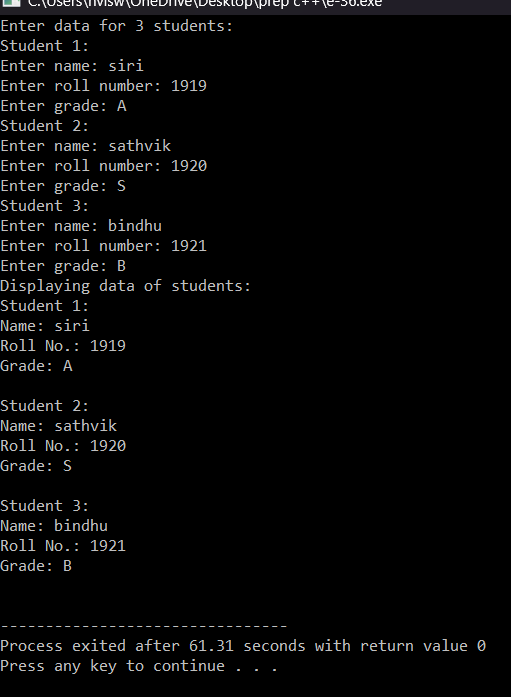
cout << endl;

}

return 0;

}

**Output:**

****

1. **Program**

#include <iostream>

#include <string>

using namespace std;

class MyClass {

public:

int myNumber;

string myString;

MyClass(int num, const string& str) {

myNumber = num;

myString = str;

}

void display() {

cout << "My Number: " << myNumber << endl;

cout << "My String: " << myString << endl;

}

};

int main() {

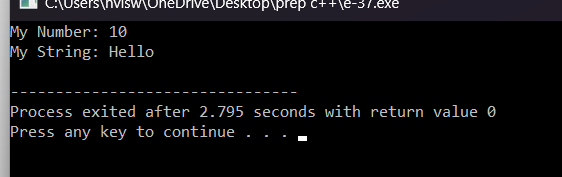
MyClass obj(10, "Hello");

obj.display();

return 0;

}

**Output:**

****

1. **Program**

#include <iostream>

using namespace std;

inline float areaCircle(float radius) {

return 3.14f \* radius \* radius;

}

inline float areaRectangle(float length, float width) {

return length \* width;

}

inline float areaSquare(float side) {

return side \* side;

}

inline float areaTriangle(float base, float height) {

return 0.5f \* base \* height;

}

int main() {

float radius, length, width, side, base, height;

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

cin >> radius;

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

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

cin >> length >> width;

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

cout << "Enter the side length of the square: ";

cin >> side;

cout << "Area of the square: " << areaSquare(side) << endl;

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

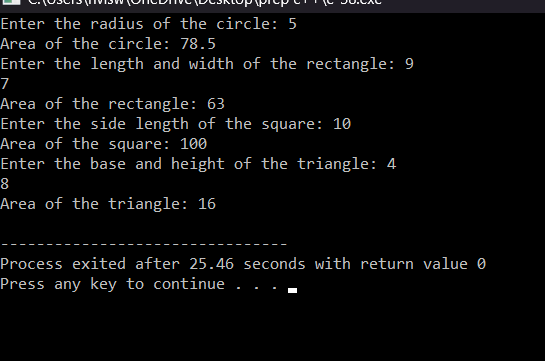
cin >> base >> height;

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

return 0;

}

**Output:**

****

1. **Program**

#include <iostream>

using namespace std;

inline float add(float a, float b) {

return a + b;

}

inline float subtract(float a, float b) {

return a - b;

}

inline float divide(float a, float b) {

if (b != 0)

return a / b;

else {

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

return 0;

}

}

inline int remainder(int a, int b) {

if (b != 0)

return a % b;

else {

cout << "Error: Modulus by zero!";

return 0;

}

}

inline float multiply(float a, float b) {

return a \* b;

}

int main() {

float num1, num2;

int num3, num4;

cout << "Enter two numbers for addition: ";

cin >> num1 >> num2;

cout << "Addition: " << add(num1, num2) << endl;

cout << "Enter two numbers for subtraction: ";

cin >> num1 >> num2;

cout << "Subtraction: " << subtract(num1, num2) << endl;

cout << "Enter two numbers for division: ";

cin >> num1 >> num2;

cout << "Division: " << divide(num1, num2) << endl;

cout << "Enter two numbers for modulus: ";

cin >> num3 >> num4;

cout << "Modulus: " << remainder(num3, num4) << endl;

cout << "Enter two numbers for multiplication: ";

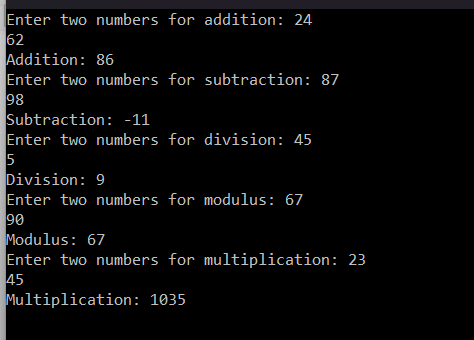
cin >> num1 >> num2;

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

return 0;

}

**Output:**

****

1. **Program**

#include <iostream>

using namespace std;

void swap(int num1, int num2) {

int temp = num1;

num1 = num2;

num2 = temp;

}

int main() {

int num1, num2;

cout << "Enter two numbers to swap: ";

cin >> num1 >> num2;

cout << "Before swapping: " << endl;

cout << "First number: " << num1 << endl;

cout << "Second number: " << num2 << endl;

swap(num1, num2);

cout << "After swapping: " << endl;

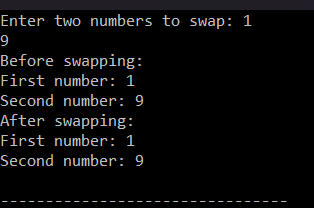
cout << "First number: " << num1 << endl;

cout << "Second number: " << num2 << endl;

return 0;

}

**Output:**

****

1. **Program**

#include <iostream>

using namespace std;

class Employee {

protected:

string name;

int emp\_id;

double basic\_salary;

public:

Employee(string name, int emp\_id, double basic\_salary) {

this->name = name;

this->emp\_id = emp\_id;

this->basic\_salary = basic\_salary;

}

};

class Salary : public Employee {

private:

double DA, HRA, gross\_pay;

public:

Salary(string name, int emp\_id, double basic\_salary)

: Employee(name, emp\_id, basic\_salary) {

DA = 0.1 \* basic\_salary;

HRA = 0.5 \* basic\_salary;

gross\_pay = basic\_salary + DA + HRA;

} double calculate\_net\_pay() {

double PTax = 0.2 \* gross\_pay;

return gross\_pay - PTax;

}

void display\_details() {

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

cout << "Employee ID: " << emp\_id << endl;

cout << "Basic Salary: " << basic\_salary << endl;

cout << "Dearness Allowance (DA): " << DA << endl;

cout << "House Rent Allowance (HRA): " << HRA << endl;

cout << "Gross Pay: " << gross\_pay << endl;

cout << "Net Pay: " << calculate\_net\_pay() << endl;

}

};

int main() {

double basic\_salaries[] = {400700, 2789239, 150000, 0, -125486};

for (int i = 0; i < sizeof(basic\_salaries) / sizeof(basic\_salaries[0]); i++) {

Salary employee("", 0, basic\_salaries[i]);

cout << "\n\*\*Employee Details\*\*\n" << endl;

employee.display\_details();

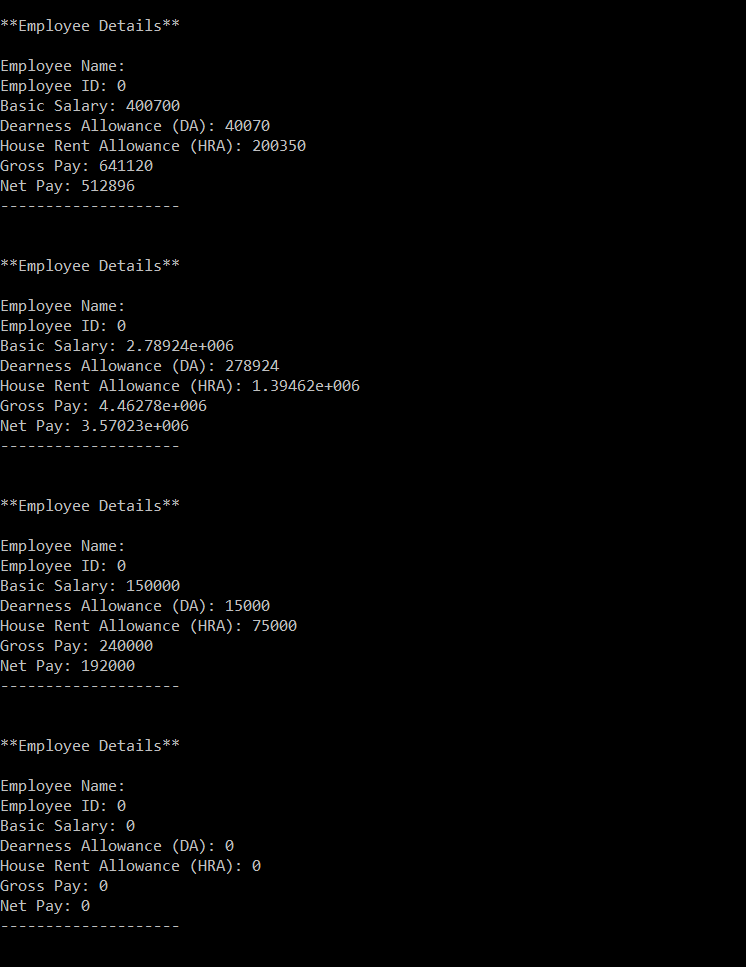
cout << "--------------------\n" << endl;

}

return 0;

}

**Output:**

****

1. **Program**

#include <iostream>

#include <cmath>

using namespace std;

class InterestCalculator {

public:

float calculateSimpleInterest(float principal, float rate, float time = 1) {

return (principal \* rate \* time) / 100;

}

float calculateCompoundInterest(float principal, float rate, float time = 1) {

float amount = principal \* pow((1 + rate / 100), time);

return amount - principal;

}

};

int main() {

float principal, rate, time;

InterestCalculator calculator;

cout << "Enter principal amount: ";

cin >> principal;

cout << "Enter rate of interest (in percentage): ";

cin >> rate;

cout << "Enter time period (in years): ";

cin >> time;

float simpleInterest = calculator.calculateSimpleInterest(principal, rate, time);

cout << "Simple Interest: " << simpleInterest << endl;

float compoundInterest = calculator.calculateCompoundInterest(principal, rate, time);

cout << "Compound Interest: " << compoundInterest << endl;

float defaultSimpleInterest = calculator.calculateSimpleInterest(principal, rate);

cout << "Simple Interest (Default time): " << defaultSimpleInterest << endl;

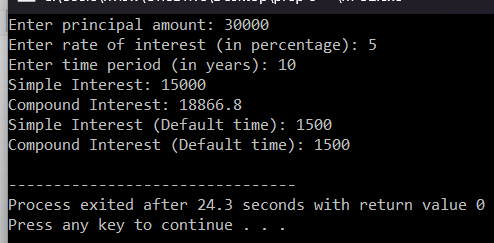
float defaultCompoundInterest = calculator.calculateCompoundInterest(principal, rate);

cout << "Compound Interest (Default time): " << defaultCompoundInterest << endl;

return 0;

}

**Output:**

****

1. **Program**

#include <iostream>

using namespace std;

class ObjectCounter {

private:

static int objectCount;

public:

ObjectCounter() {

objectCount++;

cout << "Object created. Total objects: " << objectCount << endl;

}

~ObjectCounter() {

objectCount--;

cout << "Object destroyed. Total objects: " << objectCount << endl;

}

};

int ObjectCounter::objectCount = 0;

int main() {

{

ObjectCounter obj1;

{

ObjectCounter obj2;

}

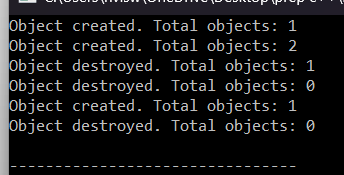
}

ObjectCounter obj3;

return 0;

}

**Output:**

****

1. **Program**

#include <iostream>

using namespace std;

class Shape {

public:

virtual float area() const = 0;

virtual float volume() const = 0;

};

class Rectangle : public Shape {

protected:

float length;

float width;

public:

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

float area() const override {

return length \* width;

}

float volume() const override {

return 0;

}

};

class Cuboid : public Rectangle {

protected:

float height;

public:

Cuboid(float l, float w, float h) : Rectangle(l, w), height(h) {}

float volume() const override {

return length \* width \* height;

}

};

int main() {

Cuboid cuboid(5, 3, 2);

float cuboidArea = cuboid.area();

float cuboidVolume = cuboid.volume();

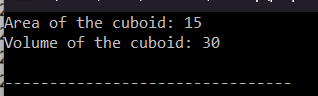
cout << "Area of the cuboid: " << cuboidArea << endl;

cout << "Volume of the cuboid: " << cuboidVolume << endl;

return 0;

}

**Output:**

****

**10.Program**

#include <iostream>

using namespace std;

class Circle {

protected:

float radius;

public:

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

float calculateArea() const {

return 3.14159 \* radius \* radius;

}

};

class Sphere : public Circle {

public:

Sphere(float r) : Circle(r) {}

float calculateVolume() const {

return (4.0 / 3.0) \* 3.14159 \* radius \* radius \* radius;

}

};

class Cylinder : public Sphere {

protected:

float height;

public:

Cylinder(float r, float h) : Sphere(r), height(h) {}

float calculateVolume() const {

return 3.14159 \* radius \* radius \* height;

}

};

int main() {

Sphere sphere(5);

Cylinder cylinder(3, 7);

cout << "Area of the circle (sphere's base): " << sphere.calculateArea() << endl;

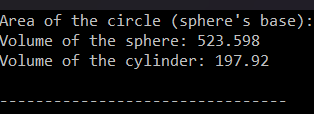
cout << "Volume of the sphere: " << sphere.calculateVolume() << endl;

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

return 0;

}

**Output:**

****

**11.Program**

#include <iostream>

#include <vector>

#include <iomanip>

using namespace std;

class Item {

private:

int codeNo;

double price;

public:

Item(int code, double p) : codeNo(code), price(p) {}

int getCodeNo() const { return codeNo; }

double getPrice() const { return price; }

};

class ShoppingList {

private:

vector<Item> items;

public:

void addItem(const Item& item) {

items.push\_back(item);

cout << "Item with code " << item.getCodeNo() << " added to the list." << endl;

}

void deleteItem(int code) {

for (vector<Item>::iterator it = items.begin(); it != items.end(); ++it) {

if (it->getCodeNo() == code) {

items.erase(it);

cout << "Item with code " << code << " deleted from the list." << endl;

return;

}

}

cout << "Item with code " << code << " not found in the list." << endl;

}

void printTotalValue() const {

double totalValue = 0;

cout << "Items in the list:" << endl;

cout << setw(10) << "Code" << setw(10) << "Price" << endl;

cout << "---------------------------" << endl;

for (vector<Item>::const\_iterator it = items.begin(); it != items.end(); ++it) {

cout << setw(10) << it->getCodeNo() << setw(10) << it->getPrice() << endl;

totalValue += it->getPrice();

}

cout << "---------------------------" << endl;

cout << "Total Value of Order: " << totalValue << endl;

}

};

int main() {

ShoppingList shoppingList;

shoppingList.addItem(Item(101, 10.50));

shoppingList.addItem(Item(102, 20.75));

shoppingList.addItem(Item(103, 15.25));

shoppingList.printTotalValue();

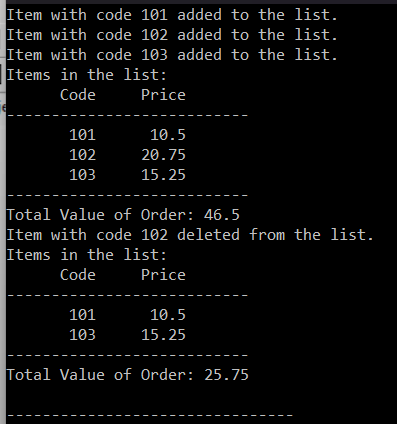
shoppingList.deleteItem(102);

shoppingList.printTotalValue();

return 0;

}

**Output:**

****

**12.Program**

#include <iostream>

#include <cmath>

using namespace std;

class Shape {

protected:

double height;

double width;

public:

Shape(double h, double w) : height(h), width(w) {}

virtual double calculateArea() const = 0;

virtual double calculatePerimeter() const = 0;

};

class Rectangle : public Shape {

public:

Rectangle(double h, double w) : Shape(h, w) {}

double calculateArea() const override {

return height \* width;

}

double calculatePerimeter() const override {

return 2 \* (height + width);

}

};

class Triangle : public Shape {

public:

Triangle(double h, double w) : Shape(h, w) {}

double calculateArea() const override {

return 0.5 \* height \* width;

}

double calculatePerimeter() const override {

double hypotenuse = sqrt(height \* height + width \* width);

return height + width + hypotenuse;

}

};

int main() {

Rectangle rect(5, 3);

Triangle tri(4, 6);

cout << "Rectangle:" << endl;

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

cout << "Perimeter: " << rect.calculatePerimeter() << endl;

cout << "\nTriangle:" << endl;

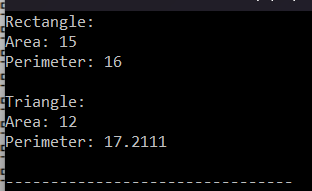
cout << "Area: " << tri.calculateArea() << endl;

cout << "Perimeter: " << tri.calculatePerimeter() << endl;

return 0;

}

**Output:**

****

**13.Program**

#include <iostream>

#include <string>

using namespace std;

class Employee {

protected:

string emp\_name;

int emp\_id;

string address;

string mail\_id;

string mobile\_no;

public:

Employee(string name, int id, string addr, string mail, string mobile)

: emp\_name(name), emp\_id(id), address(addr), mail\_id(mail), mobile\_no(mobile) {}

virtual void generatePaySlip() const = 0;

};

class Programmer : public Employee {

private:

double basic\_pay;

public:

Programmer(string name, int id, string addr, string mail, string mobile, double bp)

: Employee(name, id, addr, mail, mobile), basic\_pay(bp) {}

double calculateDA() const {

return 0.97 \* basic\_pay;

}

double calculateHRA() const {

return 0.10 \* basic\_pay;

}

double calculatePF() const {

return 0.12 \* basic\_pay;

}

double calculateStaffClubFund() const {

return 0.001 \* basic\_pay;

}

double calculateGrossSalary() const {

return basic\_pay + calculateDA() + calculateHRA();

}

double calculateNetSalary() const {

return calculateGrossSalary() - calculatePF() - calculateStaffClubFund();

}

void generatePaySlip() const override {

cout << "Pay Slip for Programmer" << endl;

cout << "-------------------------" << endl;

cout << "Employee Name: " << emp\_name << endl;

cout << "Employee ID: " << emp\_id << endl;

cout << "Address: " << address << endl;

cout << "Mail ID: " << mail\_id << endl;

cout << "Mobile No: " << mobile\_no << endl;

cout << "Basic Pay: " << basic\_pay << endl;

cout << "DA: " << calculateDA() << endl;

cout << "HRA: " << calculateHRA() << endl;

cout << "PF: " << calculatePF() << endl;

cout << "Staff Club Fund: " << calculateStaffClubFund() << endl;

cout << "-------------------------" << endl;

cout << "Gross Salary: " << calculateGrossSalary() << endl;

cout << "Net Salary: " << calculateNetSalary() << endl;

cout << "-------------------------" << endl;

}

};

int main() {

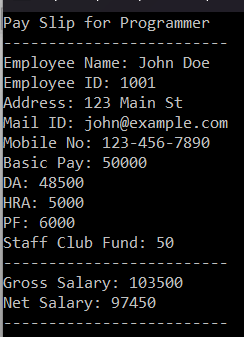
Programmer programmer("John Doe", 1001, "123 Main St", "john@example.com", "123-456-7890", 50000);

programmer.generatePaySlip();

return 0;

}

**Output:**

****

**14.Program**

#include <iostream>

using namespace std;

int main() {

int marks[4];

double total = 0, aggregate;

cout << "Enter marks for four subjects:\n";

for (int i = 0; i < 4; ++i) {

cout << "Subject " << i + 1 << ": ";

cin >> marks[i];

total += marks[i];

}

aggregate = total / 4;

cout << "\nTotal marks: " << total << endl;

cout << "Aggregate: " << aggregate << endl;

cout << "Grade: ";

if (aggregate > 75)

cout << "Distinction";

else if (aggregate >= 60 && aggregate < 75)

cout << "First Division";

else if (aggregate >= 50 && aggregate < 60)

cout << "Second Division";

else if (aggregate >= 40 && aggregate < 50)

cout << "Third Division";

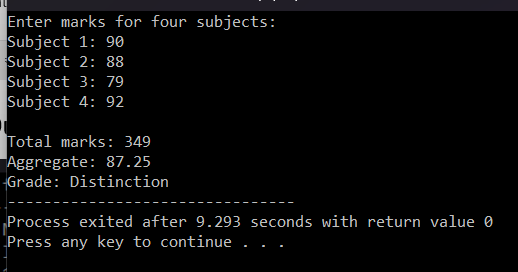
else

cout << "Fail";

return 0;

}

**Output:**

****

**15.Program**

#include <iostream>

using namespace std;

class Person {

protected:

double salary;

public:

virtual void getData() {

cout << "Enter salary: ";

cin >> salary;

}

virtual void displayData() const {

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

}

virtual double calculateBonus() const = 0;

};

class Admin : virtual public Person {

public:

double calculateBonus() const override {

return salary \* 1.1;

}

};

class Account : virtual public Person {

public:

double calculateBonus() const override {

return salary \* 1.1;

}

};

class Master : public Admin, public Account {

public:

double calculateBonus() const override {

return Person::salary \* 1.1;

}

};

int main() {

Master employee;

employee.Person::getData();

cout << "Employee details:" << endl;

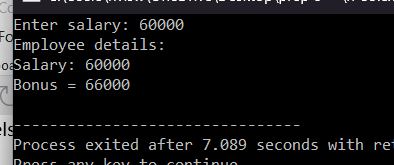
employee.Person::displayData();

cout << "Bonus = " << employee.calculateBonus() << endl;

return 0;

}

**Output:**

****