1. Create a base class called Person with a virtual function work(). Derive two classes Employee and Manager from the base class. Implement the work() function for each class.

Program :

#include<iostream>

using namespace std;

class Person

{

public:

virtual void work()

{

cout << "Person is working.";

}

};

class Employee: public Person

{

public:

void work()

{

cout << "\nI'm a Employee.";

}

};

class Manager: public Person

{

public:

void work()

{

cout << "\nI'm the Manager.";

}

};

int main()

{

Person person;

Employee employee;

Manager manager;

Person \*P1,\*P2,\*P3;

P1 = &person;

P2 = &employee;

P3 = &manager;

P1->work();

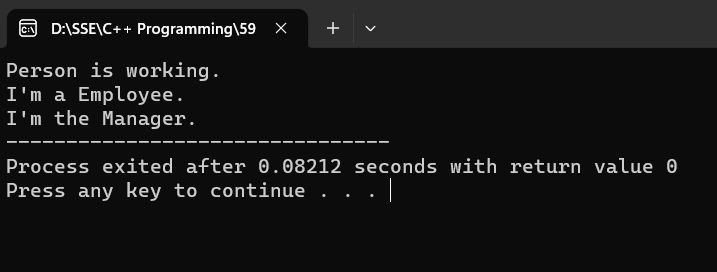
P2->work();

P3->work();

return 0;

}

Output :



1. Create a base class called Animal with a virtual function eat(). Derive two classes Herbivore and Carnivore from the base class. Implement the eat() function for each class.

Program :

#include<iostream>

using namespace std;

class Animal

{

public:

virtual void eat()

{

cout << "Animal is eating.";

}

};

class Herbivore: public Animal

{

public:

void eat()

{

cout << "\nI'm a Herbivore.";

}

};

class Carnivore: public Animal

{

public:

void eat()

{

cout << "\nI'm a Carnivore.";

}

};

int main()

{

Animal animal;

Herbivore herbivore;

Carnivore carnivore;

Animal \*A1,\*A2,\*A3;

A1 = &animal;

A2 = &herbivore;

A3 = &carnivore;

A1->eat();

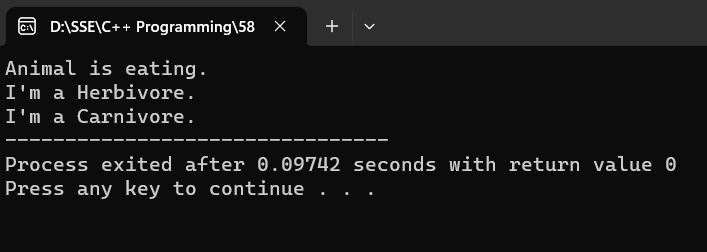
A2->eat();

A3->eat();

return 0;

}

Output :



1. Create a base class called Shape with virtual functions area() and volume(). Derive two classes Sphere and Cylinder from the base class. Implement the area() and volume() functions for each class.

Program :

#include <iostream>

#include <cmath>

using namespace std;

class Shape {

public:

virtual double area() {

return 0.0;

}

virtual double volume() {

return 0.0;

}

};

class Sphere : public Shape {

public:

double radius;

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

double area() {

return 4 \* M\_PI \* radius \* radius;

}

double volume() {

return (4.0 / 3.0) \* M\_PI \* radius \* radius \* radius;

}

};

class Cylinder : public Shape {

public:

double radius;

double height;

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

double area() {

return 2 \* M\_PI \* radius \* (radius + height);

}

double volume() {

return M\_PI \* radius \* radius \* height;

}

};

int main() {

double sphereRadius, cylinderRadius, cylinderHeight;

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

cin >> sphereRadius;

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

cin >> cylinderRadius;

cout << "Enter the height of the cylinder: ";

cin >> cylinderHeight;

Sphere sphere(sphereRadius);

Cylinder cylinder(cylinderRadius, cylinderHeight);

cout << "Sphere Area: " << sphere.area() << std::endl;

cout << "Sphere Volume: " << sphere.volume() << std::endl;

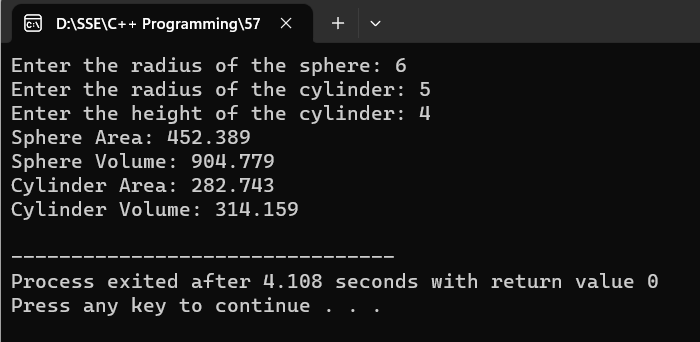
cout << "Cylinder Area: " << cylinder.area() << std::endl;

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

return 0;

}

Output :



1. Create a base class called Person with a virtual function greet(). Derive two classes Student and Teacher from the base class. Implement the greet() function for each class.

Program :

#include <iostream>

using namespace std;

class Person {

public:

virtual void greet() {

cout << "Hello, I am a person." << endl;

}

};

class Student : public Person {

public:

Student(string name) : name(name) {}

void greet() {

cout << "Hello, I am a student. My name is " << name << "." << endl;

}

string name;

};

class Teacher : public Person {

public:

Teacher(string subject) : subject(subject) {}

void greet() {

cout << "Hello, I am a teacher. I teach " << subject << "." << endl;

}

string subject;

};

int main() {

Person person;

Student student("John");

Teacher teacher("Math");

Person\* p1 = &person;

Person\* p2 = &student;

Person\* p3 = &teacher;

p1->greet();

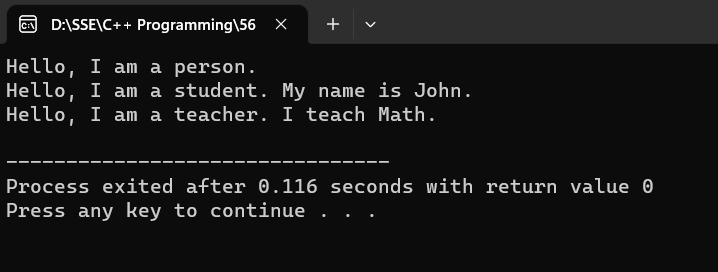
p2->greet();

p3->greet();

return 0;

}

Output :



1. Create a base class called Animal with a virtual function move(). Derive two classes Bird and Fish from the base class. Implement the move() function for each class.

Program :

#include <iostream>

#include <string>

class Animal {

public:

virtual void move() {

std::cout << "Animal is moving." << std::endl;

}

};

class Bird : public Animal {

public:

void move() {

std::cout << "Bird is flying." << std::endl;

}

};

class Fish : public Animal {

public:

void move() {

std::cout << "Fish is swimming." << std::endl;

}

};

int main() {

Animal animal;

Bird bird;

Fish fish;

Animal\* a1 = &animal;

Animal\* a2 = &bird;

Animal\* a3 = &fish;

a1->move();

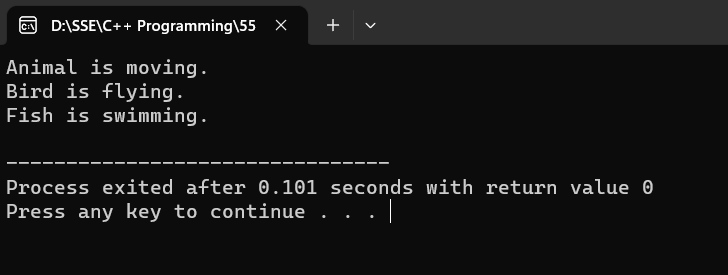
a2->move();

a3->move();

return 0;

}

Output :



1. Create a base class called Shape with virtual functions area() and perimeter(). Derive two classes Rectangle and Triangle from the base class. Implement the area() and perimeter() functions for each class.

Program :

#include <iostream>

#include <cmath>

using namespace std;

class Shape {

public:

virtual double area() {

return 0.0;

}

virtual double perimeter() {

return 0.0;

}

};

class Rectangle : public Shape {

public:

double length;

double width;

double area() {

return length \* width;

}

double perimeter() {

return 2 \* (length + width);

}

};

class Triangle : public Shape {

public:

double side1;

double side2;

double side3;

double area() {

double s = (side1 + side2 + side3) / 2;

return sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

}

double perimeter() {

return side1 + side2 + side3;

}

};

int main() {

double length, width, side1, side2, side3;

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

cin >> length;

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

cin >> width;

cout << "Enter the sides of the triangle (side1, side2, side3): ";

cin >> side1 >> side2 >> side3;

Rectangle rectangle;

rectangle.length = length;

rectangle.width = width;

Triangle triangle;

triangle.side1 = side1;

triangle.side2 = side2;

triangle.side3 = side3;

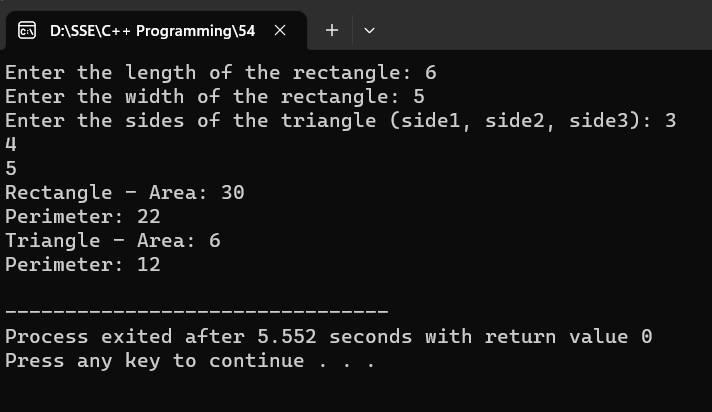
cout << "Rectangle - Area: " << rectangle.area() << "\nPerimeter: " << rectangle.perimeter() << endl;

cout << "Triangle - Area: " << triangle.area() << "\nPerimeter: " << triangle.perimeter() << endl;

return 0;

}

Output :



1. Create a base class called Vehicle with a virtual function drive(). Derive two classes Car and Truck from the base class. Implement the drive() function for each class.

Program :

#include <iostream>

using namespace std;

class Vehicle {

public:

virtual void drive() {

cout << "The vehicle is being driven." << endl;

}

};

class Car : public Vehicle {

public:

void drive() {

cout << "The car is on the road and moving." << endl;

}

};

class Truck : public Vehicle {

public:

void drive() {

cout << "The truck is hauling cargo on the highway." << endl;

}

};

int main() {

Vehicle genericVehicle;

Car myCar;

Truck myTruck;

Vehicle\* vehicle1 = &genericVehicle;

Vehicle\* vehicle2 = &myCar;

Vehicle\* vehicle3 = &myTruck;

vehicle1->drive();

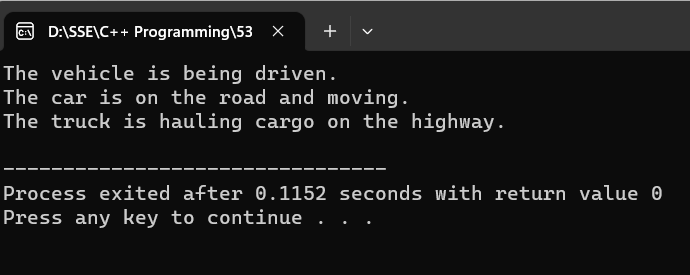
vehicle2->drive();

vehicle3->drive();

return 0;

}

Output :



1. Create a base class called Employee with a virtual function calculatePay(). Derive two classes Manager and Engineer from the base class. Implement the calculatePay() function for each class.

Program :

#include <iostream>

using namespace std;

class Employee {

public:

virtual double calculatePay() {

return 0.0;

}

};

class Manager : public Employee {

public:

double calculatePay() {

double salary,hours;

cout << "\nEnter manager's salary for hourly rate: ";

cin >> salary;

cout << "\nEnter hours worked: ";

cin >> hours;

return salary\*hours;

}

};

class Engineer : public Employee {

public:

double calculatePay() {

double hourlyRate;

int hoursWorked;

cout << "\nEnter engineer's hourly rate: ";

cin >> hourlyRate;

cout << "\nEnter engineer's hours worked: ";

cin >> hoursWorked;

return hourlyRate \* hoursWorked;

}

};

int main() {

Manager manager;

Engineer engineer;

Employee\* employee1 = &manager;

Employee\* employee2 = &engineer;

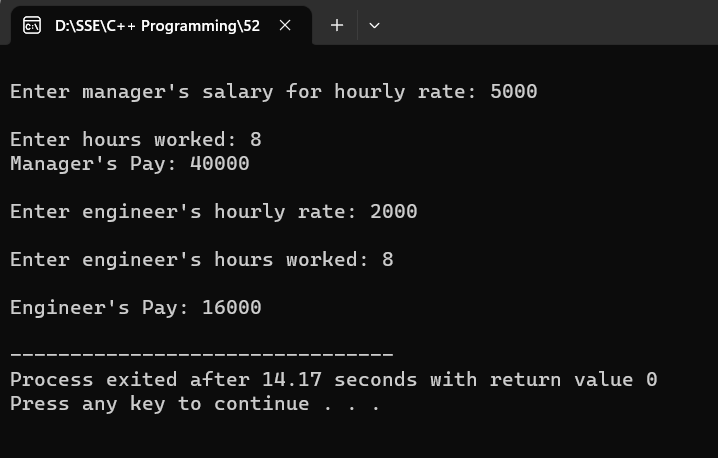
cout << "Manager's Pay: " << employee1->calculatePay() << endl;

cout << "\nEngineer's Pay: " << employee2->calculatePay() << endl;

return 0;

}

Output :



1. Create a base class called Animal with a virtual function speak(). Derive two classes Cat and Dog from the base class. Implement the speak() function for each class.

Program :

#include<iostream>

#include<string>

using namespace std;

class Animal

{

public:

virtual void speak()

{

cout << "Every animal makes a sound.";

}

};

class Cat: public Animal

{

public:

void speak()

{

cout << "\nMeow is my sound.";

}

};

class Dog: public Animal

{

public:

void speak()

{

cout << "\nI will bark.";

}

};

int main()

{

Animal animal;

Cat cat;

Dog dog;

Animal \*A1,\*A2,\*A3;

A1 = &animal;

A2 = &cat;

A3 = &dog;

A1->speak();

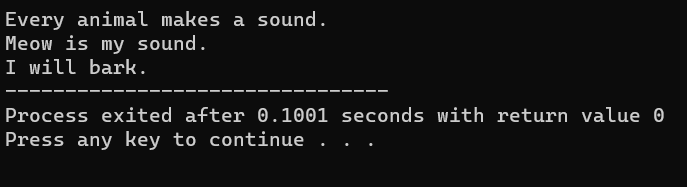
A2->speak();

A3->speak();

return 0;

}

Output :



1. Create a base class called Shape with a virtual function area(). Derive two classes Rectangle and Circle from the base class. Implement the area() function for each class.

Program :

#include <iostream>

#include <cmath>

using namespace std;

class Shape {

public:

virtual double area() {

return 0.0;

}

};

class Rectangle : public Shape {

public:

double length;

double width;

double area() {

return length \* width;

}

};

class Circle : public Shape {

public:

double radius;

double area() {

return M\_PI \* radius \* radius;

}

};

int main() {

int choice;

Rectangle rectangle;

Circle circle;

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

cin >> rectangle.length;

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

cin >> rectangle.width;

cout << "Rectangle Area: " << rectangle.area() << endl;

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

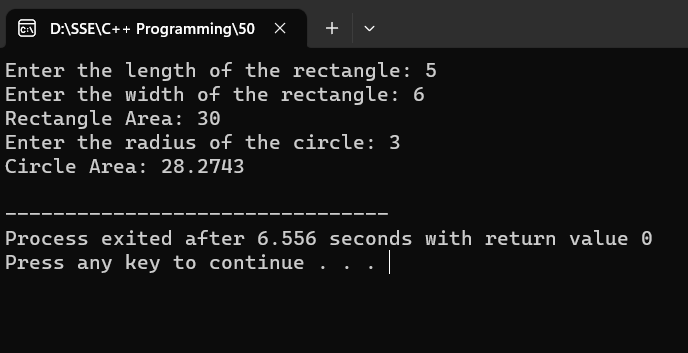
cin >> circle.radius;

cout << "Circle Area: " << circle.area() << endl;

return 0;

}

Output :



192211142