**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**

#include <iostream>

#include <string>

class Person {

public:

virtual void work() {

std::cout << "Person is working." << std::endl;

}

};

class Employee : public Person {

public:

void work() override {

std::cout << "Employee is working." << std::endl;

}

};

class Manager : public Person {

public:

void work() override {

std::cout << "Manager is working." << std::endl;

}

};

int main() {

Person\* person1 = new Employee();

Person\* person2 = new Manager();

person1->work();

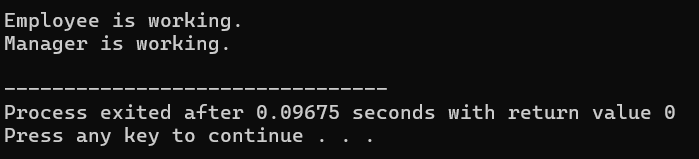
person2->work();

delete person1;

delete person2;

return 0;

}

****

**2. 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.**

#include <iostream>

#include <string>

class Animal {

public:

virtual void eat() {

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

}

};

class Herbivore : public Animal {

public:

void eat() override {

std::cout << "Herbivore is eating plants." << std::endl;

}

};

class Carnivore : public Animal {

public:

void eat() override {

std::cout << "Carnivore is eating meat." << std::endl;

}

};

int main() {

Animal\* animal1 = new Herbivore();

Animal\* animal2 = new Carnivore();

animal1->eat();

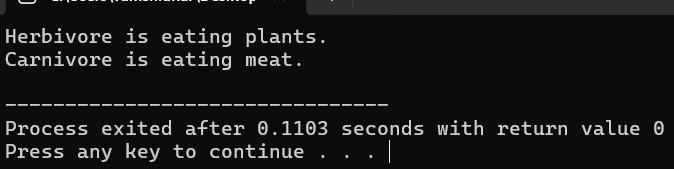
animal2->eat();

delete animal1;

delete animal2;

return 0;

}

****

**3.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**

#include <iostream>

#include <cmath>

class Shape {

public:

virtual double area() const = 0;

virtual double volume() const = 0;

};

class Sphere : public Shape {

private:

double radius;

public:

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

double area() const override {

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

}

double volume() const override {

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

}

};

class Cylinder : public Shape {

private:

double radius;

double height;

public:

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

double area() const override {

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

}

double volume() const override {

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

}

};

int main() {

Sphere sphere(3.0);

Cylinder cylinder(2.0, 5.0);

std::cout << "Sphere:" << std::endl;

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

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

// Calculating and printing area and volume for cylinder

std::cout << "\nCylinder:" << std::endl;

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

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

return 0;

}

****

**4. 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**

#include <iostream>

#include <string>

class Person {

public:

virtual void greet() const {

std::cout << "Greetings from a person!" << std::endl;

}

};

class Student : public Person {

public:

void greet() const override {

std::cout << "Hello, I am a student!" << std::endl;

}

};

class Teacher : public Person {

public:

void greet() const override {

std::cout << "Hello, I am a teacher!" << std::endl;

}

};

int main() {

Person\* person1 = new Student ();

Person\* person2 = new Teacher ();

person1->greet();

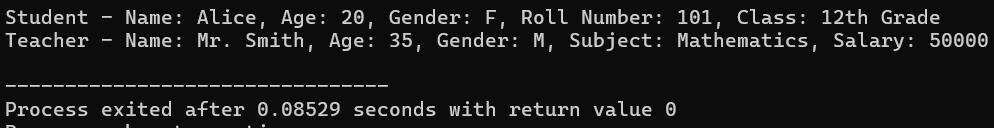
person2->greet();

delete person1;

delete person2;

return 0;

}

****

**5 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**

#include <iostream>

#include <string>

class Person {

public:

virtual void greet() const {

std::cout << "Greetings from a person!" << std::endl;

}

};

class Student : public Person {

public:

void greet() const override {

std::cout << "Hello, I am a student!" << std::endl;

}

};

class Teacher : public Person {

public:

void greet() const override {

std::cout << "Hello, I am a teacher!" << std::endl;

}

};

int main() {

Person\* person1 = new Student();

Person\* person2 = new Teacher();

person1->greet();

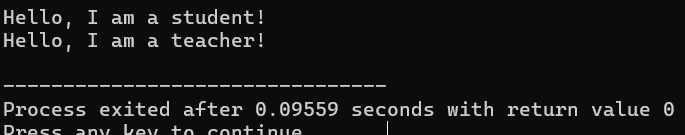
person2->greet();

delete person1;

delete person2;

return 0;

}

****

**6.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.**

#include <iostream>

#include<cmath>

class Shape {

public:

virtual double area() const = 0;

virtual double perimeter() const = 0;

};

class Rectangle : public Shape {

private:

double length;

double width;

public:

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

double area() const override {

return length \* width;

}

double perimeter() const override {

return 2 \* (length + width);

}

};

class Triangle : public Shape {

private:

double side1;

double side2;

double side3;

public:

Triangle(double s1, double s2, double s3) : side1(s1), side2(s2), side3(s3) {}

double area() const override {

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

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

}

double perimeter() const override {

return side1 + side2 + side3;

}

};

int main() {

Rectangle rectangle(5.0, 3.0);

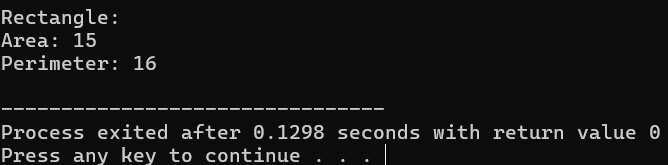
Triangle triangle(3.0, 4.0, 5.0);

std::cout << "Rectangle:" << std::endl;

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

std::cout << "Perimeter: " << rectangle.perimeter() << std::endl;

}

****

**7.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.**

#include <iostream>

class Vehicle {

public:

virtual void drive() const = 0;

};

class Car : public Vehicle {

public:

void drive() const override {

std::cout << "Car is being driven." << std::endl;

}

};

class Truck : public Vehicle {

public:

void drive() const override {

std::cout << "Truck is being driven." << std::endl;

}

};

int main() {

Vehicle\* vehicle1 = new Car();

Vehicle\* vehicle2 = new Truck();

vehicle1->drive();

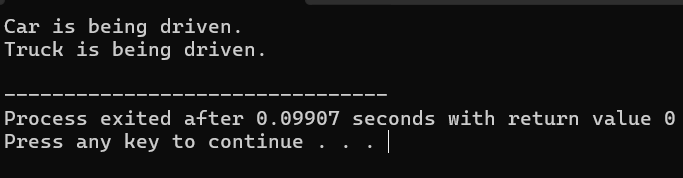
vehicle2->drive();

delete vehicle1;

delete vehicle2;

return 0;

}

****

**8.Create a base class called Employee with a virtual function calculate Pay(). Derive two classes Manager and Engineer from the base class. Implement the calculate Pay () function for each class.**

#include <iostream>

class Employee {

public:

virtual double calculatePay() const = 0;

};

class Manager : public Employee {

public:

double calculatePay() const override {

return 5000.0;

}

};

class Engineer : public Employee {

public:

double calculatePay() const override {

return 4000.0;

}

};

int main() {

Manager manager;

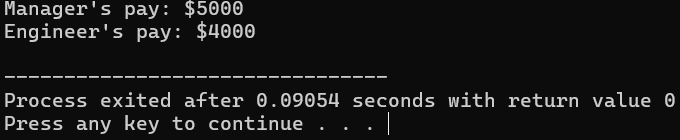
Engineer engineer;

std::cout << "Manager's pay: $" << manager.calculatePay() << std::endl;

std::cout << "Engineer's pay: $" << engineer.calculatePay() << std::endl;

return 0;

}

****

**9.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.**

#include <iostream>

// Base class

class Animal {

public:

virtual void speak() const = 0;

};

class Cat : public Animal {

public:

void speak() const override {

std::cout << "Meow!" << std::endl;

}

};

class Dog : public Animal {

public:

void speak() const override {

std::cout << "Woof!" << std::endl;

}

};

int main() {

Cat cat;

Dog dog;

std::cout << "Cat says: ";

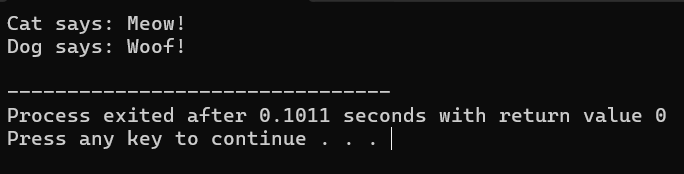
cat.speak();

std::cout << "Dog says: ";

dog.speak();

return 0;

}

****

**10.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.**

#include <iostream>

class Shape {

public:

virtual double area() const = 0;

};

class Rectangle : public Shape {

private:

double length;

double width;

public:

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

double area() const override {

return length \* width;

}

};

class Circle : public Shape {

private:

double radius;

public:

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

double area() const override {

return 3.14159 \* radius \* radius;

}

};

int main() {

Rectangle rectangle(5.0, 3.0);

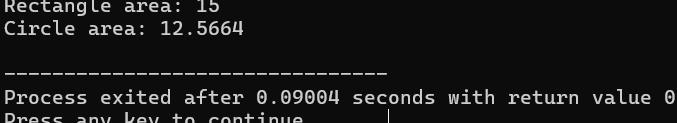
Circle circle(2.0);

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

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

return 0;

}

****