**Polymorphism**

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>

using namespace std;

class person{

protected:

string name;

public:

person(string a){

name=a;

}

virtual void work(){

cout<<name<<"is working"<<endl;

}

virtual~person(){

}

};

class employee:public person{

public:

employee(string emp):person(emp){

}

void work()override{

cout<<name<<" is working"<<endl;

}

};

class manager:public person{

public:

manager(string man):person(man){

}

void work()override{

cout<<name<<" is managing team"<<endl;

}

};

int main(){

person\*p1=new employee("prudhvi");

person\*p2=new manager("raj");

p1->work();

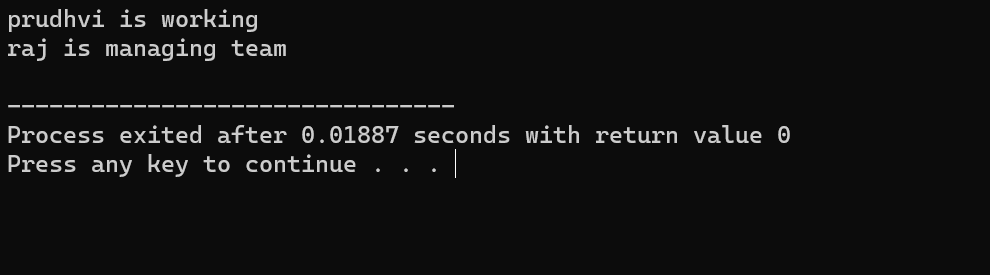
p2->work();

delete p1;

delete p2;

return 0;

}



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.

#include<iostream>

using namespace std;

class Animal{

protected:

string name;

public:

Animal(string an){

name=an;

}

virtual void eat(){

cout<<name<<" is a plant eating animal"<<endl;

}

virtual~Animal(){

}

};

class herbivore:public Animal{

public:

herbivore(string her):Animal(her){

}

void eat()override{

cout<<name<<" is plant eating animal"<<endl;

}

};

class carnivore:public Animal{

public:

carnivore(string car):Animal(car){

}

void eat()override{

cout<<name<<" is flesh eating animal"<<endl;

}

};

int main(){

Animal\*animal1=new herbivore("goat");

Animal\*animal2=new carnivore("lion");

animal1->eat();

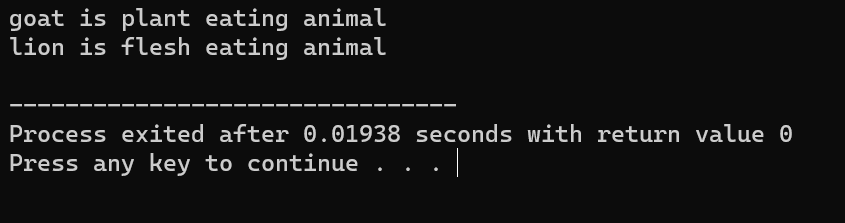
animal2->eat();

delete animal1;

delete animal2;

return 0;

}



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

#include<iostream>

#include<cmath>

using namespace std;

class shape{

public:

virtual double area()const=0;

virtual double volume()const=0;

virtual~shape(){

}

};

class sphere:public shape{

double radius;

public:

sphere(double r):radius(r){

}

double area()const override{

cout<<"area of circle="<<M\_PI\*pow(radius,2)<<endl;

}

double volume()const override{

cout<<"volume of circle="<<(4/3)\*M\_PI\*pow(radius,3)<<endl;

}

};

class cylinder:public shape{

double radius;

double height;

public:

cylinder(double r,double h):radius(r),height(h){

}

double area()const override{

cout<<"area of cylinder="<<2\*M\_PI\*radius\*(radius+height)<<endl;

}

double volume()const override{

cout<<"volume of cylinder="<<M\_PI\*pow(radius,2)\*height<<endl;

}

};

int main(){

shape\*shape1=new sphere(2);

shape\*shape2=new cylinder(3,4);

shape1->area();

shape1->volume();

shape2->area();

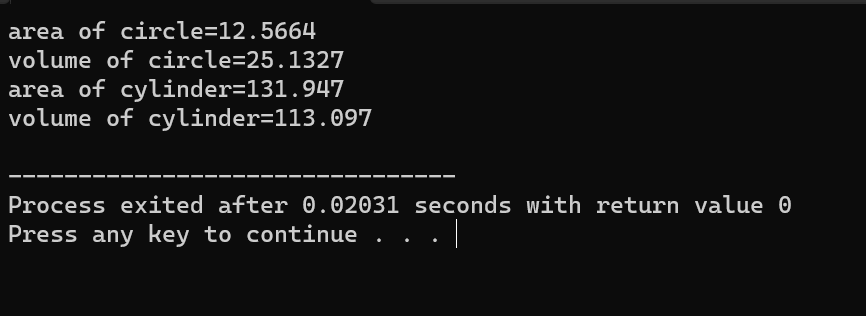
shape2->volume();

delete shape1;

delete shape2;

return 0;

}



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

#include<iostream>

using namespace std;

class person{

protected:

string name;

public:

person(string emp){

name=emp;

}

virtual void greet(){

cout<<name<<"good morning"<<endl;

}

virtual ~person(){}

};

class student:public person{

public:

student(string empl):person(empl){

}

void greet()override{

cout<<name<<"=good morning madam"<<endl;

}

};

class teacher:public person{

public:

teacher(string man):person(man){

}

void greet()override{

cout<<name<<"=good morning"<<endl;

}

};

int main(){

person\*person1=new student("prudhvi");

person\*person2=new teacher("madam");

person1->greet();

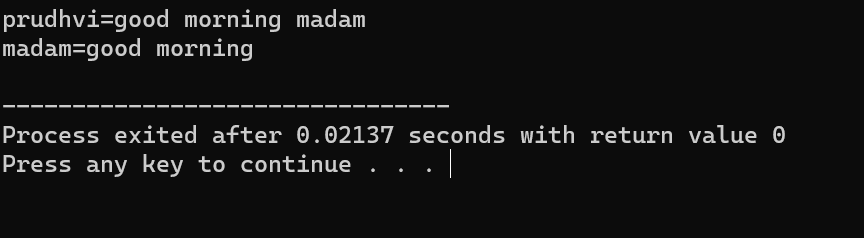
person2->greet();

delete person1;

delete person2;

return 0;

}



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

#include<iostream>

using namespace std;

class person{

protected:

string name;

public:

person(string emp){

name=emp;

}

virtual void greet(){

cout<<name<<"good morning"<<endl;

}

virtual ~person(){}

};

class student:public person{

public:

student(string empl):person(empl){

}

void greet()override{

cout<<name<<"=good morning madam"<<endl;

}

};

class teacher:public person{

public:

teacher(string man):person(man){

}

void greet()override{

cout<<name<<"=good morning"<<endl;

}

};

int main(){

person\*person1=new student("prudhvi");

person\*person2=new teacher("madam");

person1->greet();

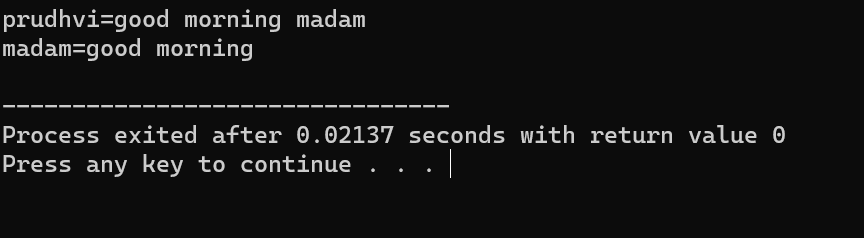
person2->greet();

delete person1;

delete person2;

return 0;

}



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.

#include<iostream>

#include<cmath>

using namespace std;

class shape{

public:

virtual double area()const=0;

virtual double perimeter()const=0;

virtual~shape(){

}

};

class rectangle:public shape{

double length,width;

public:

rectangle(double l,double w):length(l),width(w){

}

double area()const override{

cout<<"area of rectangle="<<length\*width<<endl;

}

double perimeter()const override{

cout<<"perimeter of rectangle="<<2\*(length+width)<<endl;

}

};

class triangle:public shape{

double side1,side2,side3;

public:

triangle(double a,double b,double c):side1(a),side2(b),side3(c){

}

double area()const override{

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

cout<<"area of triangle="<<sqrt(s\*(s-side1)\*(s-side2)\*(s-side3))<<endl;

}

double perimeter()const override{

cout<<"perimeter of triangle="<<side1+side2+side3<<endl;

}

};

int main(){

shape\*shape1=new rectangle(2,3);

shape\*shape2=new triangle(3,4,5);

shape1->area();

shape1->perimeter();

shape2->area();

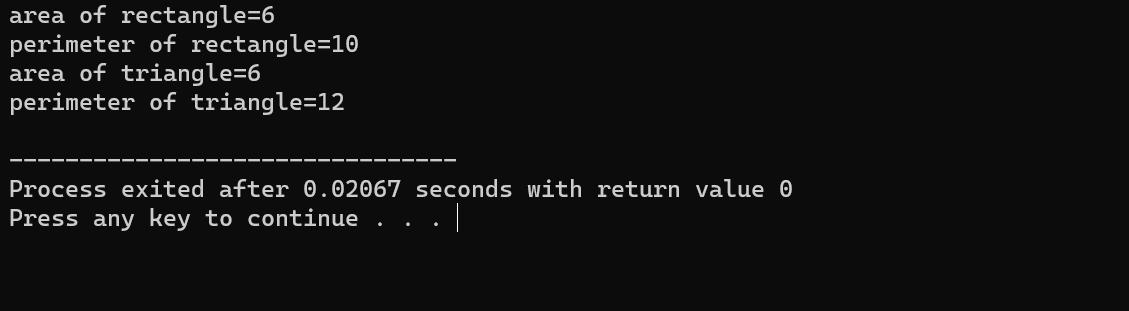
shape2->perimeter();

delete shape1;

delete shape2;

return 0;

}



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.

#include<iostream>

using namespace std;

class vehicle{

protected:

string name;

public:

vehicle(string emp){

name=emp;

}

virtual void drive(){

cout<<name<<"good morning"<<endl;

}

virtual ~vehicle(){}

};

class car:public vehicle{

public:

car(string empl):vehicle(empl){

}

void drive()override{

cout<<name<<"is a car"<<endl;

}

};

class truck:public vehicle{

public:

truck(string man):vehicle(man){

}

void drive()override{

cout<<name<<"is a lorry"<<endl;

}

};

int main(){

vehicle\*person1=new car("swift desire");

vehicle\*person2=new truck("ashok leyland");

person1->drive();

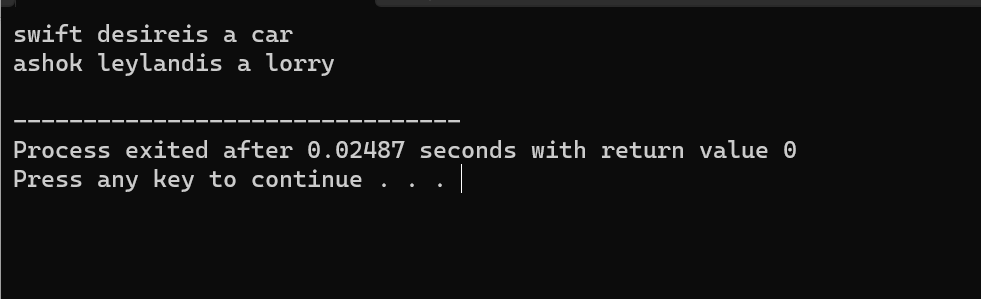
person2->drive();

delete person1;

delete person2;

return 0;

}



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

#include <iostream>

using namespace std;

class Employee {

public:

virtual double calculatePay() const = 0;

virtual ~Employee() {}

};

class Manager : public Employee {

double baseSalary;

double bonus;

public:

Manager(double salary, double bonusAmount) : baseSalary(salary), bonus(bonusAmount) {}

double calculatePay() const override {

return baseSalary + bonus;

}

};

class Engineer : public Employee {

double hourlyRate;

int hoursWorked;

public:

Engineer(double rate, int hours) : hourlyRate(rate), hoursWorked(hours) {}

double calculatePay() const override {

return hourlyRate \* hoursWorked;

}

};

int main() {

Employee\* manager = new Manager(5000.0, 1000.0);

Employee\* engineer = new Engineer(50.0, 160);

cout << "Manager's pay: $" << manager->calculatePay() << endl;

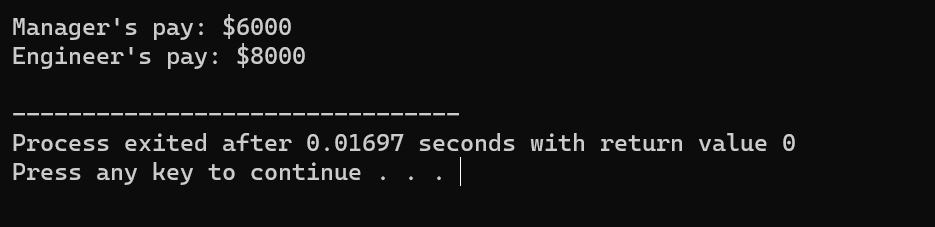
cout << "Engineer's pay: $" << engineer->calculatePay() << endl;

delete manager;

delete engineer;

return 0;

}



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.

#include<iostream>

using namespace std;

class Animal{

protected:

string name;

public:

Animal(string cname){

name=cname;

}

virtual void speak(){

cout<<name<<" speak meow"<<endl;

}

virtual~Animal(){

}

};

class cat:public Animal{

public:

cat(string cname):Animal(cname){

}

void speak()override{

cout<<name<<" speaks meow"<<endl;

}

};

class dog:public Animal{

public:

dog(string dogname):Animal(dogname){

}

void speak()override{

cout<<name<<" barks"<<endl;

}

};

int main(){

Animal\*animal1=new cat("bob");

Animal\*animal2=new dog("rex");

animal1->speak();

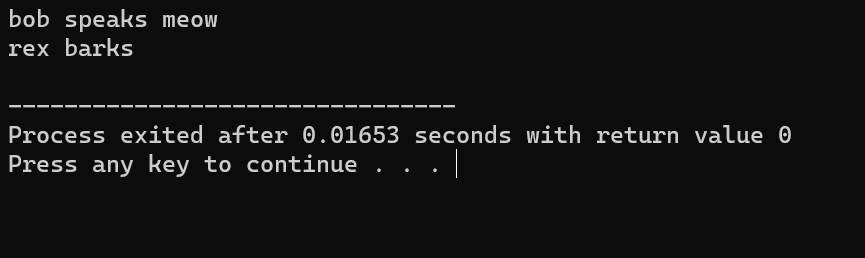
animal2->speak();

delete animal1;

delete animal2;

return 0;

}



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.

#include<iostream>

#include<cmath>

using namespace std;

class shape{

public:

virtual double area()const=0;

virtual~shape(){

}

};

class rectangle:public shape{

double length,width;

public:

rectangle(double l,double w):length(l),width(w){

}

double area()const override{

cout<<"area of rectangle="<<length\*width<<endl;

}

};

class circle:public shape{

double radius;

public:

circle(double r):radius(r){

}

double area()const override{

cout<<"area of circle="<<M\_PI\*pow(radius,2)<<endl;

}

};

int main(){

shape\*shape1=new rectangle(2,3);

shape\*shape2=new circle(3);

shape1->area();

shape2->area();

delete shape1;

delete shape2;

return 0;

}

