



Ibtisam Hassan

46150

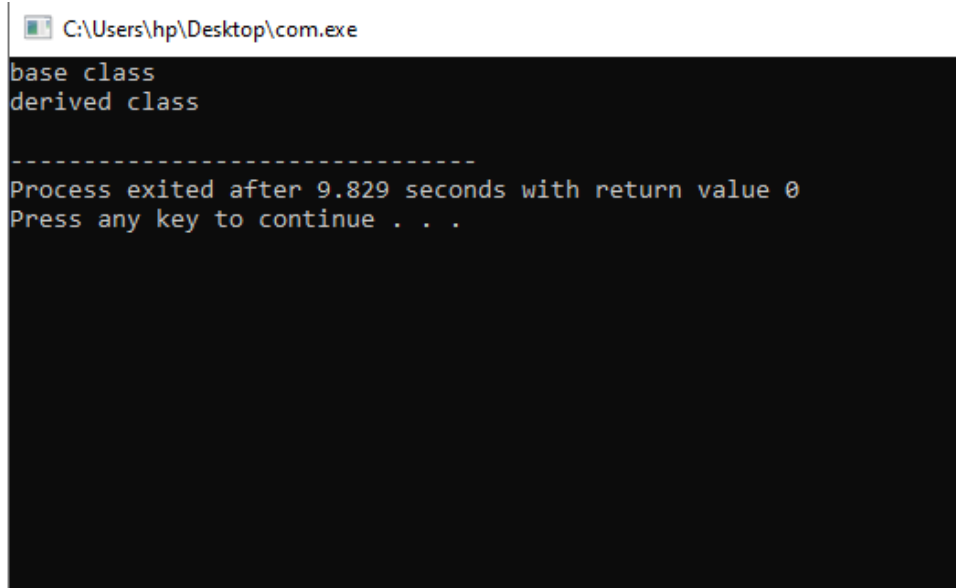
OOP_LABTASK-AFTER MIDS

Task 1.

```
task1.h  task1.cpp  main.cpp  com.cpp  mammal1,h.cpp  mammal1.cpp  comp2.cpp

1  #include<iostream>
2  using std::cout;
3  using std::endl;
4  class base{
5      public:
6          virtual void testfunction();
7  };
8  class derived : public base{
9      public:
10         void testfunction();
11 };
12 void base::testfunction(){
13     cout<<"base class"<<endl;
14 }
15 void derived::testfunction(){
16     cout<<"derived class"<<endl;
17 }
18 int main(void){
19     base*ptr = new base;
20     ptr->testfunction();
21     delete ptr;
22     ptr = new derived;
23     ptr -> testfunction();
24     delete ptr;
25     return 0;
26 }
```

OUTPUT

A screenshot of a Windows command prompt window. The title bar at the top reads "C:\Users\hp\Desktop\com.exe". The command prompt shows the following text: "base class", "derived class", a line of dashes "-----", "Process exited after 9.829 seconds with return value 0", and "Press any key to continue . . .".

```
C:\Users\hp\Desktop\com.exe
base class
derived class
-----
Process exited after 9.829 seconds with return value 0
Press any key to continue . . .
```

Changing:

We have used virtual function for polymorphism

TASK 2

```

1  #pragma once
2  #include<iostream>
3  using std::cout;
4  using std::endl;
5  class mammal{
6  public:
7      mammal(void);
8      ~mammal(void);
9      virtual void move() const;
10     virtual void speak()const;
11     protected:
12         int itsage;
13 };
14 class Dog:public mammal{
15 };
16 mammal::mammal(void):itsage(1){
17     cout<<"mammal constructor..."<<endl;
18 }
19 mammal::~~mammal(void){
20     cout<<"mammal destructor..."<<endl;
21 }
22 void mammal::move()const{
23     cout<<"mammal moves a step!"<<endl;
24 }
25 void mammal::speak()const{
26     cout<<"what does a mammal speak? mammilian!"<<endl;
27 }
28
29 int main()
30 {
31     mammal *pDog = new Dog;
32     pDog->move();
33     pDog->speak();
34     return 0;
35 }
36

```

OUTPUT

```
C:\Users\hp\Desktop\comp2.exe
mammal constructor...
mammal moves a step!
what does a mammal speak? mammilian!

-----
Process exited after 10.36 seconds with return value 0
Press any key to continue . . .
```

Task-3

```
int main()
{
    mammal* theArray[5];
    mammal* ptr;
    int choice,i;
    for (i=0;i<5;i++)
    {
        cout<<"(1)dog (2)cat (3)horse (4)guinea pig:";
        cin>>choice;
        switch(choice)
        {
            case 1: ptr=new dog;
                    break;
            case 2: ptr=new cat;
                    break;
            case 3: ptr=new horse;
                    break;
            case 4: ptr=new guineapig;
                    break;
            default : ptr=new mammal;
                    break;
        }
        theArray[i]=ptr;
    }
    for (i=0;i<5;i++)
        theArray[i]->speak();
    for (i=0;i<5;i++)
        delete theArray[i];
    return 0;
}
```

```

#include<iostream>
using std::cout;
using std::endl;
using std::cin;
class mammal{
public:
    mammal(void);
    ~mammal(void);
    virtual void move()const;
    virtual void speak()const;
protected:
    int itsage;
};
class dog:public mammal{
public:
    dog(){
        cout<<"dog is barking:";
        cout<<"dog is running";
    }
};
class horse:public mammal{
public:
    horse(){
        cout<<"horse is walking"<<endl;
        cout<<"horse is eating"<<endl;
    }
};
class guineapig:public mammal{
public:
    guineapig(){

```

```

31         cout<<"guineapig is walking"<<endl;
32         cout<<"guineapig is eating"<<endl;
33     }
34 };
35 class cat:public mammal{
36     public:
37     cat(){
38         cout<<"cat is running"<<endl;
39         cout<<"cat is eating"<<endl;
40     }
41 };
42 mammal::mammal(void):itsage(1)
43 {
44     cout<<"mammal constructor..."<<endl;
45 }
46 mammal::~~mammal(void)
47 {
48     cout<<"mammal destructor..."<<endl;
49 }
50 void mammal::move() const
51 {
52     cout<<"mammal moves a step!"<<endl;
53 }
54 void mammal::speak() const
55 {
56     cout<<"what does a mammal speak?mammalian!"<<endl;
57 }
58
59 int main()
60 {

```

Output

```

D:\oop\midtask3.exe
(1)dog (2)cat (3)horse (4)guinea pig:1
mammal constructor...
dog is barking:dog is running(1)dog (2)cat (3)horse (4)guinea pig:2
mammal constructor...
cat is running
cat is eating
(1)dog (2)cat (3)horse (4)guinea pig:3
mammal constructor...
horse is walking
horse is eating
(1)dog (2)cat (3)horse (4)guinea pig:4
mammal constructor...
guineapig is walking
guineapig is eating
(1)dog (2)cat (3)horse (4)guinea pig:

```

Questions/Answers

1. What is a v-table?

To implement virtual functions, C++ implementations typically use a form of late binding known as the virtual table. The **virtual table** is a lookup table of functions used to resolve function calls in a dynamic/late binding manner. The virtual table sometimes goes by other names, such as "vtable", "virtual function table", "virtual method table", or "dispatch table".

2. What is a virtual destructor?

Virtual Destructor in C++ is used to release or free the memory used by the child class (derived class) object when the child class object is being removed from the memory using the parent class's pointer object.

3. How do you show the declaration of a virtual constructor?

In C++, the constructor cannot be virtual, because when a constructor of a class is executed there is no virtual table in the memory, means no virtual pointer defined yet. So, the constructor should always be non-virtual.

4.s

5. How do you invoke a base member function from a derived class in which you've overridden that function?

6. How do you invoke a base member function from a derived class in which you have not overridden that function?

7. If a base class declares a function to be virtual, and a derived class does not use the term virtual when overriding that class, is it still virtual when inherited by a third-generation class?

8. What is the protected keyword used for?

It is used to make class attributes only accessible by derived class