**SHEIKH AHMAD IJAZ**

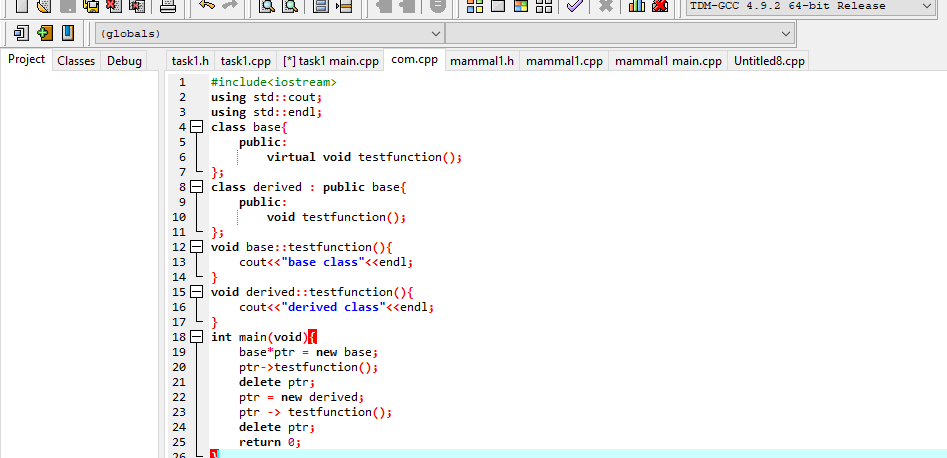
**SAP ID:47535, BSCS 2A**

**LAB TASK (OOP)**

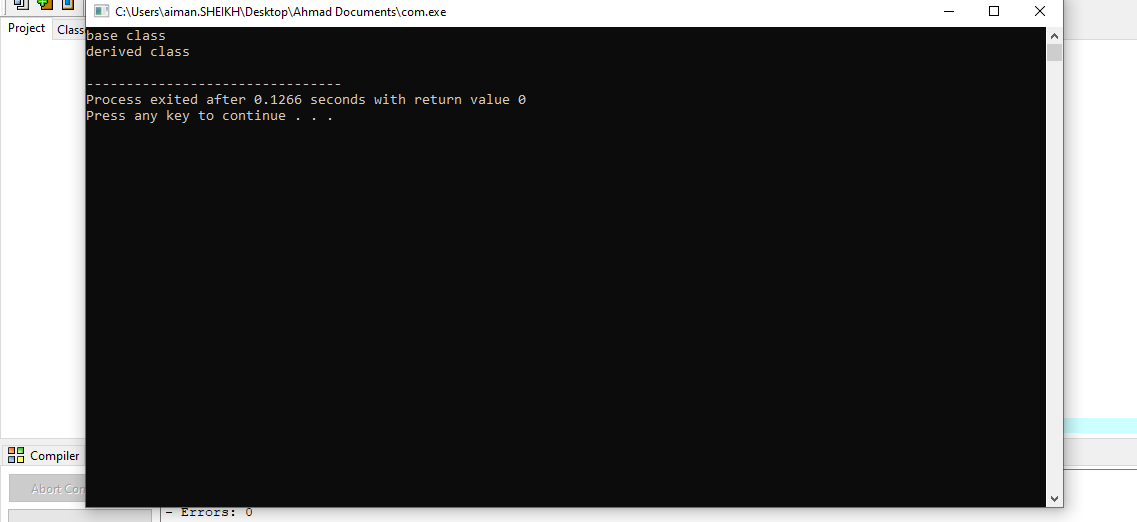
**SIR SHAHZAD**

TASK 1:

To gain a better understanding of polymorphic and virtual functions start with the following simple example. Notice we have not defined a virtual function yet.



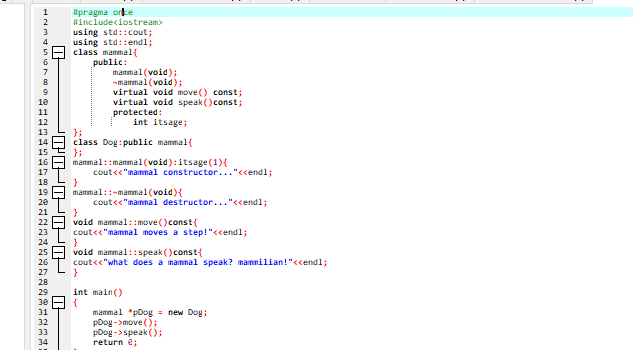
OUTPUT:



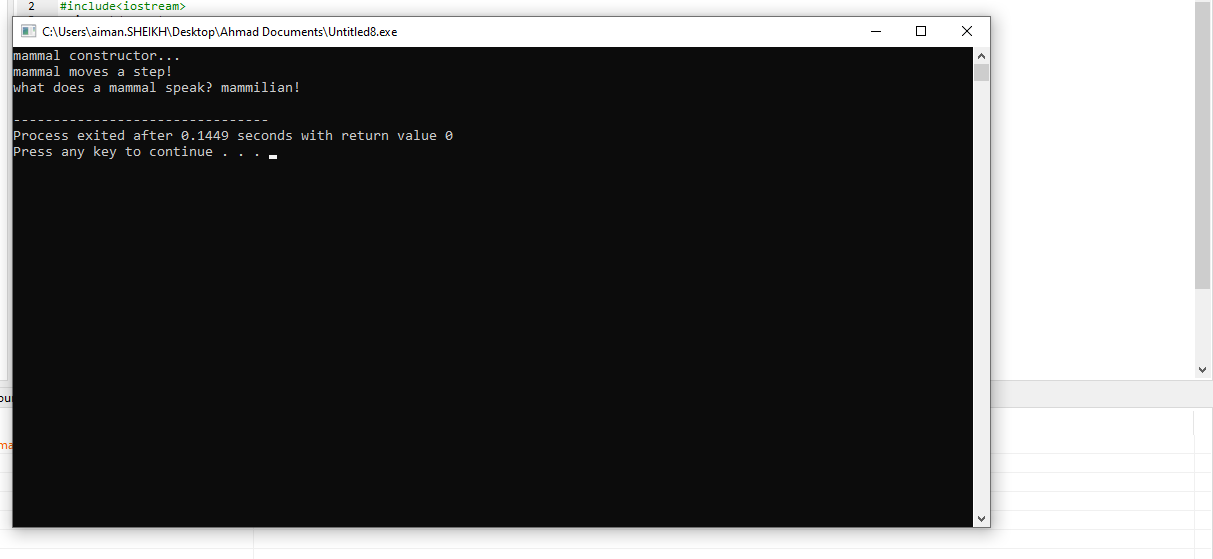
When we add the modification to this code, the second testFunction() call generates the message “Derived class”.

TASK 2:

You will first build two classes, Mammal and Dog. Dog will inherit from Mammal. Below is the Mammal class code. Once you have the Mammal class built, build a second class Dog that will inherit publicly from Mammal.



OUTPUT:



Above is the output of the program. First of all we had faced an error because the classDod was not inherited from classes mammal. Then we solve this error .It is the output what I expected. When we remove the keyword virtual from the class mammal,the code again executes the same output.

TASK 3: Develop additional classes for Cat, Horse, and GuineaPig overriding the move and speak methods. (If you do not know guinea pigs go “weep weep”)

#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 Neighing "<<endl;

cout<<"horse is walking"<<endl;

}

};

class guineapig:public mammal{

public:

guineapig(){

cout<<"guineapig is walking"<<endl;

}

};

class cat:public mammal{

public:

cat(){

cout<<"cat is walking"<<endl;

cout<<"cat is meowing"<<endl;

}

};

mammal::mammal(void):itsage(1)

{

cout<<"mammal constructor..."<<endl;

}

mammal::~mammal(void)

{

cout<<"mammal destructor..."<<endl;

}

void mammal::move() const

{

cout<<"mammal moves a step!"<<endl;

}

void mammal::speak() const

{

cout<<"what does a mammal speak?mammalian!"<<endl;

}

int main()

{

mammal\* theArray[5];

mammal\* ptr;

int choice,i;

for (i=0;i<5;i++)

{

cout<<"(1)dog (2)cat (3)horse (4)gruinea 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++)

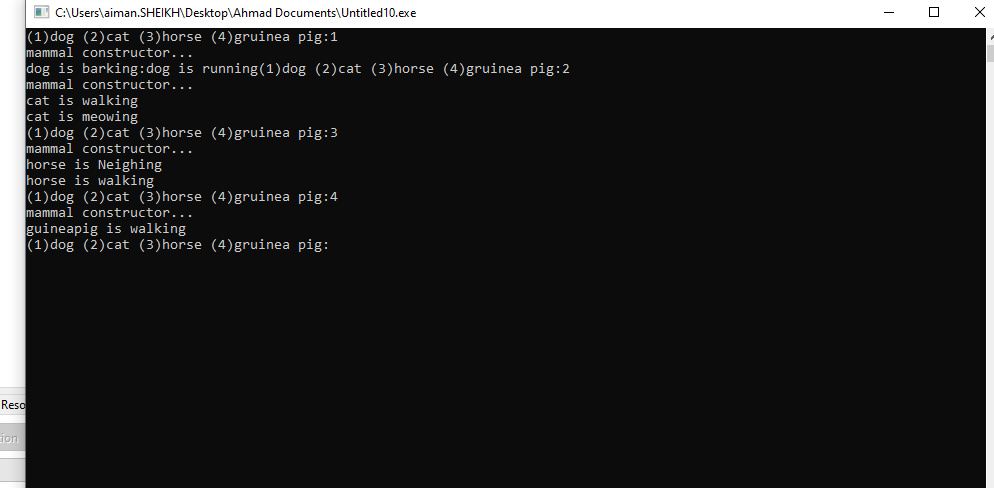
theArray[i]->speak();

for (i=0;i<5;i++)

delete theArray[i];

return 0;

}

OUTPUT: 

WE HAVE ADDED THE SOUNDS AND ACTIONS OF THE ANIMALS MENTIONED ABOVE IN THIS CODE AND WE HAVE COMPLETED THE CODE BEFORE THE MAIN FUNCTION.

Q1. What is a v-table?

Ans: To implement virtual functions, C++ uses a special 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.

Q2. What is a virtual destructor?

Ans: Virtual Destructor in C++ is a member function that is used to free up the memory allocated used by the object of a child class or derived class when it is removed from the memory using the parent class pointer object.

Q3. How do you show the declaration of a virtual constructor

Ans: 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.

Q4. How can you create a virtual copy constructor?

Ans: