学生姓名：谢烔 学生学号：180011102261 学生班级：信息工程3班

实验日期：2020年11月22日 实验机房： ZA201 任课教师：李冰

实验（二）：Inheritance

# 一、实验目的

（1）理解inheritance概念

（2）编写简单程序掌握inheritance

# 二、实验内容

## 程序运行（1）

运行英文教材Listing 11.4，将程序和运行结果截屏粘贴并解释程序结果。

1.程序代码：

#include <iostream>

using namespace std;

enum BREED { GOLDEN, CAIRN, DANDIE, SHETLAND, DOBERMAN, LAB };//BREED

class Mammal

{

public:

// constructors

Mammal();

Mammal(int age);

~Mammal();

//accessors

int GetAge() const { return itsAge; }

void SetAge(int age) { itsAge = age; }

int GetWeight() const { return itsWeight; }

void SetWeight(int weight) { itsWeight = weight; }

//Other methods

void Speak() const { cout << "Mammal sound!\n"; }

void Sleep() const { cout << "shhh. I'm sleeping.\n"; }

protected:

int itsAge;

int itsWeight;

};

class Dog : public Mammal

{

public:

// Constructors

Dog();

Dog(int age);

Dog(int age, int weight);

Dog(int age, BREED breed);

Dog(int age, int weight, BREED breed);

~Dog();

// Accessors

BREED GetBreed() const { return itsBreed; }

void SetBreed(BREED breed) { itsBreed = breed; }

// Other methods

void WagTail() const { cout << "Tail wagging...\n"; }

void BegForFood() const { cout << "Begging for food...\n"; }

private:

BREED itsBreed;

};

Mammal::Mammal():

itsAge(1),

itsWeight(5)

{

cout << "Mammal constructor..." << endl;

}

Mammal::Mammal(int age):

itsAge(age),

itsWeight(5)

{

cout << "Mammal(int) constructor..." << endl;

}

Mammal::~Mammal()

{

cout << "Mammal destructor..." << endl;

}

Dog::Dog():

Mammal(),

itsBreed(GOLDEN)

{

cout << "Dog constructor..." << endl;

}

Dog::Dog(int age):

Mammal(age),

itsBreed(GOLDEN)

{

cout << "Dog(int) constructor..." << endl;

}

Dog::Dog(int age, int weight):

Mammal(age),

itsBreed(GOLDEN)

{

itsWeight = weight;

cout << "Dog(int, int) constructor..." << endl;

}

Dog::Dog(int age, int weight, BREED breed):

Mammal(age),

itsBreed(breed)

{

itsWeight = weight;

cout << "Dog(int, int, BREED) constructor..." << endl;

}

Dog::Dog(int age, BREED breed):

Mammal(age),

itsBreed(breed)

{

cout << "Dog(int, BREED) constructor..." << endl;

}

Dog::~Dog()

{

cout << "Dog destructor..." << endl;

}

int main()

{

Dog Fido;

Dog rover(5);

Dog buster(6,8);

Dog yorkie (3,GOLDEN);

Dog dobbie (4,20,DOBERMAN);

Fido.Speak();

rover.WagTail();

cout << "Yorkie is " << yorkie.GetAge()

<< " years old" << endl;

cout << "Dobbie weighs ";

cout << dobbie.GetWeight() << " pounds" << endl;

return 0;

}

2.程序运行结果截图：

1. 程序解释：

对Mammal函数进行重写，父类Mammal将数据成员声明为protected以便派生类Dog进行访问，父类Mammal定义了itsAge ；itsWeight两个数据成员，子类Dog继承Mammal后进行重写为Dog() ; Dog(int age) ; Dog(int age, int weight) ; Dog(int age, BREED breed) ; Dog(int age, int weight, BREED breed)，并向父类Mammal传参数，每次都输出"Mammal constructor..."以及子类对应的结果，这其中Dog类要初始化时他的父类Mammal也要初始化且Mammal只能调用一个参数在同时调用两个或三个参数时需要进行补充。Fido.Speak调用了父类Mammal的speak函数输出 "Mammal sound!"且通过Fido调用了Dog类扩展的方法wagTail，所以输出"Tail wagging...",在main函数中rover；buster；yorkie ；dobbie分别调用Dog类，yorkie以及dobbie还调用了BREED中的一个常量，其中由于BREED是一种枚举常量所以breed常量只能取BREED的枚举常量中的一个，于是分别输出Yorkie is 3 years old ; Dobbie weighs 20 pounds ，最后输出Dog和Mammal的destructor 。

## 程序运行（2）

运行英文教材Listing 11.9，将程序和运行结果截屏粘贴并解释程序结果。

1.程序代码

#include <iostream>

using namespace std;

class Mammal

{

public:

Mammal():itsAge(1) { }

virtual ~Mammal() { }

virtual void Speak() const { cout << "Mammal speak!\n"; }

protected:

int itsAge;

};

class Dog : public Mammal

{

public:

void Speak()const { cout << "Woof!\n"; }

};

class Cat : public Mammal

{

public:

void Speak()const { cout << "Meow!\n"; }

};

class Horse : public Mammal

{

public:

void Speak()const { cout << "Winnie!\n"; }

};

//LESSON 11: Implementing Inheritance

//2329417\_ch11.qxd 5/29/08 12:28 PM Page 24

class Pig : public Mammal

{

public:

void Speak()const { cout << "Oink!\n"; }

};

int main()

{

Mammal\* theArray[5];

Mammal\* ptr;

int choice, i;

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

{

cout << "(1)dog (2)cat (3)horse (4)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 Pig;

break;

default: ptr = new Mammal;

break;

}

theArray[i] = ptr;

}

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

theArray[i]->Speak();

return 0;

}

2.程序运行结果截图：

2

1. 程序解释：

Dog、Cat、Horse、Pig使用public的方式继承了Mammal的类，继承了Mammal的类虚函数Speck并重写成Woof、Meow、Winnie以及Oink，而Main函数中访问各类，使用control input运用键盘提供输出，使用switch函数和指针选择对应的子类调用父类的Speck，并通过for函数实现循环。

## 问答题：

（1）P.340, Exercise 2 （2）P.341, Exercise 3（3）P.341, Exercise 4

程序代码：

#include <iostream>

using namespace std;

class Shape {

public:

// constructor

Shape() {

cout << "This is Shape"<< endl;

}

virtual ~Shape() {

cout <<"Shape destructor" << endl ;

}

};

class Rectangle : public Shape {

public:

// constructor

Rectangle() {

cout << "This is Rectangle"<< endl;

}

virtual ~Rectangle() {

cout <<"Rectangle destructor" << endl ;

}

virtual int GetLength() const {

return Rec\_itsLength;

}

virtual void SetLength(int Rec\_Length) {

Rec\_itsLength = Rec\_Length;

}

int GetWidth() const {

return Rec\_itsWidth;

}

void SetWidth(int Rec\_Width) {

Rec\_itsWidth = Rec\_Width;

}

private:

int Rec\_itsLength = 0;

int Rec\_itsWidth = 0;

};

class Square : public Rectangle {

public:

// constructor

Square() {

cout << "This is Square"<< endl;

}

virtual ~Square() {

cout <<"Square destructor" << endl ;

}

int GetLength() const {

return itsLength;

}

void SetLength(int Length) {

itsLength = Length;

}

private:

int itsLength = 0;

};

int main() {

Rectangle ab;

Square bc;

ab.SetLength(5);

ab.SetWidth(6);

bc.SetLength(4);

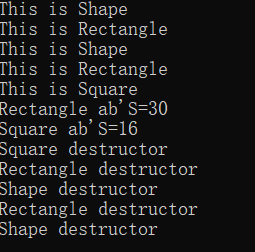
cout<<"Rectangle ab'S="<<ab.GetLength() \* ab.GetWidth()<<endl;

cout<<"Square ab'S="<<bc.GetLength() \* bc.GetLength()<<endl;

return 0;

}

运行结果：



# 三、实验总结

这次实验加深了派生类与父类的继承关系的认识，了解了子类对于父类数据成员的调用以及子类的初始化，通过观看课程回放以及查阅相关资料了解例如Fido 以及枚举常量的相关知识，在简答题中利用前面所学知识实现程序，起初理解不了面积计算的程序公式，后查阅相关资料后实现程序，也通过这道题更理解子类调用的工作原理。