实验十二 继承与派生

一、 问题描述

1. 实验目的：

掌握派生类的若干基本概念和特性，并能够应用于程序编写。

掌握验证性实验的基本方法和过程(认知、实验、总结）。

2. 实验内容：

分别编写一段测试代码来回答任务书中的相关问题（每一个问题，用一个 工程文件，同时需要记录相应的调试过程），具体问题请参考“实验任务 说明12.doc”；

调试的过程；（动态调试的相关截图，比如 设置断点、查看当前变量值等）；

编译出来的可执行程序单独放在一个目录下（bin/exe/debug目录下，同时附上输入数据说明和输出结果）

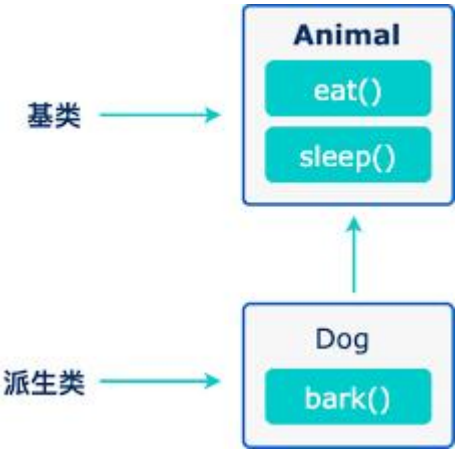
二、 实验过程

一、名词解释并画图

1、继承、派生

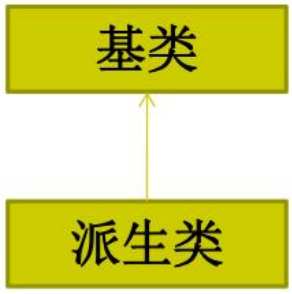
答：继承：新的类从已有类那里得到已有的特性；

派生：从已存在的类产生一个新的类；

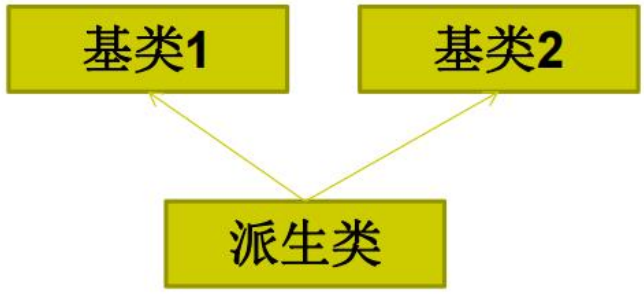


2、单继承、多继承、多层继承

答：单继承：派生类只从一个基类派生。



多继承：派生类从多个基类派生



多重派生：由一个基类派生出多个不同的派生类。

多层派生：派生类又作为基类，继续派生新的类

3、基类、派生类

答：基类和派生类是相对而言的，基类（父类）：在继承关系中,被继承的类（或已存在的类）；派生类（子类）：通过继承关系定义出来的新类（新建立的类）；派

生类通过增加信息将抽象的基类变为某种有用的类型。派生类是基类定义的延续。

二、简答题

1、如果派生类希望将从基类继承到的公有和保护成员作为自己的保护成员，那么在声明派生类时应该采用哪一个继承访问控制符？

答：继承方式：即派生类的访问控制方式，用于控制基类中声明的成员在多大的范围内能被派生类的用户访问。如果派生类希望将从基类继承到的公有和保护成员作为自己的保护成员，那么在声明派生类时应该采用protected。

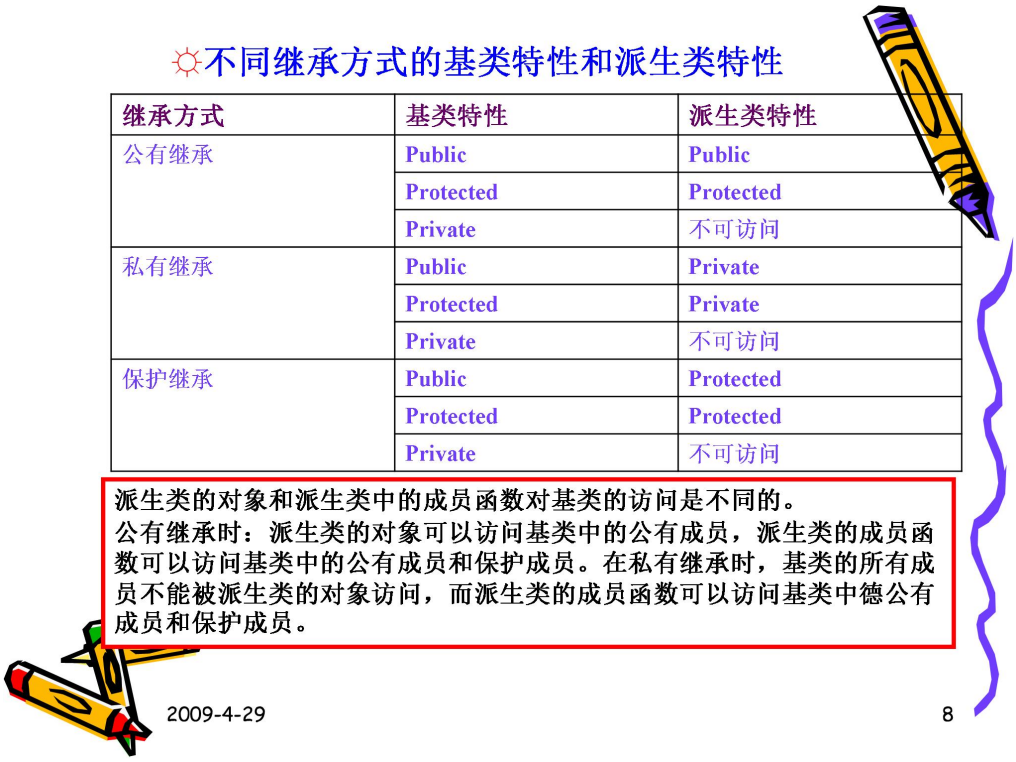
2、扼要说明基类、类成员和派生类的构造函数和析构函数的调用顺序。

答：派生类构造函数执行的一般顺序是：（1）基类构造函数（2）派生类对象成员类的构造函数（3）派生类构造函数体中的内容。而析构函数的调用则与之相反。

3、什么是公有继承下的赋值兼容原则？

答：C++中，以public方式继承的派生类可以看成基类的子类型；所谓赋值兼容规则指的是不同类型的对象间允许相互赋值的规定；面向对象程序设计语言中，在公有派生的情况下，**允许将派生类的对象赋值给基类的对象**，但反过来却不行，即不允许将基类的对象赋值给派生类的对象。这是因为一个派生类对象的存储空间总是大于它的基类对象的存储空间。若将基类对象赋值给派生类对象，**这个派生类对象中将会出现一些未赋值的不确定成员。**

4、列表描述，不同继承方式下，基类成员的不同访问控制属性在派生类中的变化。



三、程序设计题

1、请设计一个带继承的C++程序，来说明和验证protected成员的特点与作用。要求有文字的注释

备注：要体现**基类对象**对protected的访问；**派生类对象**对protected的访问；**派生类成员函数**对基类成员的访问。

答：与私有成员一样，在基类中，不能被使用类程序员进行公共访问，但可以被类内部的成员函数访问；如果使用的类是派生类成员，则可以被访问（即能够被派生类的成员函数访问）。

```cpp
#include<iostream>
using namespace std;

//A为基类
class A {
protected:
    int x,y;
public:
    void f(){
        cout << x << endl;//可以被类内部的成员函数访问
    };
};

class B :public A {
public:
    void h() {
        cout << x << y << endl; //能够被派生类的成员函数访问
        f();
    }
};

int main() {
    A a;
    cout << a.x << endl;//派生类对象（main函数）不能进行访问
    B b;
    cout << b.x << endl;
    a.f();
    b.h();
}
```

2、在第1题的基础上，将基类中protected成员函数在派生类中调整为public

答：此时在main函数中b.x可以成功访问。

```cpp
#include<iostream>
using namespace std;

//A为基类
class A {
protected:
    int x, y;
public:
    void f() {
        cout << x << endl;//可以被类内部的成员函数访问
    };
};

class B :public A {
public:
    void h() {
        cout << x << y << endl; //能够被派生类的成员函数访问
        f();
    }
    A::x;
    A::y;
};

int main() {
    A a;
    //cout << a.x << endl;//ERROR:派生类对象（main函数）不能进行访问
    B b;
    cout << b.x << endl;//可以访问
    a.f();
    b.h();
}
```

3、设计一个汽车类Vehicle，包含数据成员车轮数和车重，由它派生出类Car和类Truck，前者包含载客数，后者包含载重量。

要求：

（1）每个类都有构造函数，且有显示车属性信息、启动、停止操作。

（2）对涉及的相关运算符进行重载。

（3）给出测试数据，编写程序实现。

备注：这道题目，要求大家更要从“抽象到具体”的高度来思考问题，即：对于汽车类，这个抽象层面，我们究竟要提供哪些成员；为何要继承，我们要通过继承来达到什么目的；如何继承，那种继承方式是最合适的；在Car和Truck，如何更加“具体”。

答：设计一个汽车类Vehicle，数据成员包括：车轮个数、车重。小车类Car是它的公有派生类，新增数据成员：载人数passenger；卡车truck是汽车类的公有派生类，新增数据成员：载重量payload，另外每个类都有相关数据输出方法。

```cpp
#include <iostream>
#include <stdlib.h>
#include <string>
using namespace std;

class vehicle //汽车类
{
protected:
    int wheels; // 车轮数
    double weight; // 重量
    bool isrun;//是否启动
public:
    vehicle(int a, double b); //构造函数
    int GetWheels() { return wheels; }
    double GetWeight() { return weight; }
    void GetStart() { isrun = 1; cout << "车辆启动" << endl; }//启动
    void GetStop() { isrun = 0; cout << "车辆停止" << endl; }//停止
    void show();
};
vehicle::vehicle(int a, double b) //构造函数
{
    isrun = 0;//初始未启动
    wheels = a;
    weight = b;
}
void vehicle::show()
{
    cout << "车轮数：" << wheels << endl;
    cout << "重量：" << weight << endl;
}

class car :public vehicle //小汽车类
{
    int passenger;//载人数
public:
    car(int wheels1, double weight1, int passenger1);//构造函数
    void show();//显示车属性信息
};
//构造函数
car::car(int wheels1, double weight1, int passenger1) :vehicle(wheels1, weight1)
{
    passenger = passenger1;
}
//显示车属性信息
void car::show()
{
    cout << "小车类：" << endl;
    vehicle::show();
    cout << "载人数：" << passenger << endl;
}
//显示车属性信息
class truck :public vehicle //卡车类
{
    double payload;//载重量
public:
    truck(int wheels1, double weight1,double payload1);//构造函数
    void show();
};
```

```
//构造函数
truck::truck(int wheels1, double weight1, double payload1) :vehicle(wheels1, weight1)
{
    payload = payload1;
}
//显示车属性信息
void truck::show()
{
    cout << "卡车类：" << endl;
    vehicle::show();
    cout << "载重量：" << payload << endl;
}

int main()
{
    car a(4, 100, 5);
    truck b(6, 500, 300);
    a.GetStart();
    b.GetStart();
    a.show();
    cout << endl;
    b.show();
    a.GetStop();
    b.GetStop();
    system("PAUSE");
    return 0;
}
```

```
D:\XPfile\学习资料\年级分类\大二下\C++\myexp\十一周\Homwork\x64\Debug\Homework.exe
车辆启动
车辆启动
小车类：
车轮数：4
重量：100
载人数：5

卡车类：
车轮数：6
重量：500
载重量：300
车辆停止
车辆停止
请按任意键继续. . .
```

四、代码分析题

1、请用代码验证以下描述（要求：代码中有注释）

当基类中声明有默认形式的构造函数或未声明构造函数时，派生类构造函数可以不向基类构造函数传递参数。

若基类中未声明构造函数，派生类中也可以不声明，全采用默认形式构造函数。

```
//定义另一个类，将作为类成员
class B{
private:
    int b ;
public:
    int get( ) { return b;}
};

class D :public B {
private:
    int d;
public:
    D(int d1) {
        d = d1;//当基类中声明有默认形式的构造函数或未声明构造函数时，派生类构造函数可以不向基类构造函数传递参数。
    }
};

class E :public B {
private:
    int e;
    //若基类中未声明构造函数，派生类中也可以不声明，全采用默认形式构造函数。
};
```

当基类声明有带形参的构造函数时，派生类也应声明带形参的构造函数，并将参数传递给基类构造函数。

```
#include<iostream>
using namespace std;

class A{ //定义基类
private:
    int a ;
public:
    A(int x) {
        a = x ;
        cout<<"A's constructor called."<<endl ;
    }
    void show( ) {
        cout<<a<<endl;
    }
};

class C : public A {//定义派生类
private:
    int c;
public:
    // 派生类构造函数
    //当基类声明有带形参的构造函数时，派生类也应声明带形参的构造函数，并将参数传递给基类构造函数。
    C(int x, int y, int z) :A(x){
        c = z;
        cout << "C's constructor called." << endl;
    }
    void show() {
        A::show();
        cout << c << endl;
    }
};
```

2、从C++继承关系角度，分析代码执行可能及结果

答：由于private成员派生类的类成员函数不能访问，所以getPrt函数会出现报错。而m_nPub在父类是public成员，public继承，可以在main函数通过派生类对象访问。而父类的protected成员，派生类对象无法访问，但是派生类成员函数可以访问，可以通

过调用派生类成员函数来进行访问。而对public对象进行protected继承，private继承，派生类特性变为protected，无法通过派生类对象访问。

protected对建立其所在类对象的模块来说，它与 private 成员的性质相同。对于其派生类来说，它与 public 成员的性质相同。既实现了数据隐藏，又方便继承，实现代码重用。protected派生类中的成员函数：可以直接访问基类中的public和protected成员，但不能直接访问基类的private成员；

```
#include <iostream>
using namespace std;

class Parent {
private:
    int m_nPrt;
protected:
    int m_nPtd;
public:
    int m_nPub;
public:
    Parent(int var = -1) {
        m_nPub = var;
        m_nPtd = var;
        m_nPrt = var;
    }
};

class Child1 : public Parent {
public:
    int getPub() { return m_nPub; }
    int getPtd() { return m_nPtd; }
    //int getPrt() { return m_nPrt; }   //private成员派生类不可访问
};

class Child2 : protected Parent {
public:
    int getPub() { return m_nPub; }
    int getPtd() { return m_nPtd; }
    //int getPrt() { return m_nPrt; }   //private成员派生类不可访问
};
```

```
class Child3 : private Parent {
public:
    int getPub() { return m_nPub; }
    int getPtd() { return m_nPtd; }
    //int getPrt() { return m_nPrt; }   //private成员派生类不可访问
};

int main() {
    Child1 cd1;
    Child2 cd2;
    Child3 cd3;
    int nVar = 0;

    Parent p;
    p.m_nPub;
    cd1.m_nPub = nVar;  // public 继承，可以访问
    //cd1.m_nPtd = nVar;   //父类的protected成员，派生类对象不能访问
    nVar = cd1.getPtd();   //可以通过调用Child1的成员函数访问

    //cd2.m_nPub = nVar;  // protected 继承，派生类对象不能访问
    nVar = cd2.getPtd();   //可以通过调用Child2的成员函数访问

    //cd3.m_nPub = nVar;  // private 继承，派生类对象不能访问
    nVar = cd3.getPtd();   //可以通过调用Child3的成员函数访问

    return 0;
}
```

3、派生类构造函数的书写及执行顺序结果分析

答：在main中，创建了C类对象，先调用C类的父类A的构造函数，再调用C的类成员B类的构造函数，最后调用自身的构造函数。因为创建了两个C类对象，所以调用了两遍。打印c1，c2，先调用父类A的show函数，打印a即1，再调用b的get方法，打印b即2，最后打印c即5，c2也同样，打印结果是3，4，7。

```
#include<iostream>
using namespace std;

class A {   //定义基类
private:
    int a;
public:
    A(int x) { a = x; cout << "A's constructor called." << endl; }
    void show() { cout << a << endl; }
};
class B {   //定义另一个类，将作为类成员
private:
    int b;
public:
    B(int x) { b = x; cout << "B's constructor called." << endl; }
    int get() { return b; }
};
class C : public A {        //定义派生类
private:
    int c;
    B obj_b;
public:
    C(int x, int y, int z) :A(x), obj_b(y)// 派生类构造函数
    {
        c = z;
        cout << "C's constructor called." << endl;
    }
    void show() {
        A::show();
        cout << obj_b.get() << "," << c << endl;
    }
};

void main() {
    C  c1(1, 2, 5), c2(3, 4, 7);
    c1.show();
    c2.show();
}
```

Microsoft Visual Studio 调试控制台

```
A's constructor called.
B's constructor called.
C's constructor called.
A's constructor called.
B's constructor called.
C's constructor called.
1
2,5
3
4,7
```

4、派生类析构函数的书写及执行顺序结果分析

答：创建了两个Y类型的对象y1，y2。调用Y类的print方法进行打印，Y类的print方

法又调用了X的print方法，先打印x1,x2即5，6再打印y即7。Y2同理。当main函数结束时，调用析构函数，析构函数的调用顺序与构造函数相反。先调用子类Y的析构函数，再调用X。由于有两个对象，所以调用了两遍。

```
#include<iostream>
using namespace std;

class X {
    int x1, x2;
public:
    X(int i, int j) { x1 = i; x2 = j; }
    void print() { cout << x1 << ", " << x2 << endl; }
    ~X() { cout << " X's destructor called." << endl; }
};
class Y :public X {
    int y;      // 派生类Y新增数据成员
public:
    Y(int i, int j, int k) : X(i, j) { y = k; }     //派生类构造函数
    void print() { X::print(); cout << y << endl; }
    ~Y() { cout << " Y's destructor called." << endl; }
};
void main()
{
    Y y1(5, 6, 7),
      y2(2, 3, 4);
    y1.print();
    y2.print();
}
```

三、 附录

源程序文件项目清单：src/homework

3.1.cpp 3.2.cpp 3.3.cpp 4.1.cpp 4.2.cpp 4.3.cpp 4.4.cpp