计算机学院　高级语言程序设计　课程实验报告

实验题目：继承与派生(一)		学号：202200400053
日期：2024-04-18	班级： 2202	姓名： 王宇涵
Email：1941497679@qq.com		

实验目的：

1. 会使用继承与派生不同的方式及访问控制
2. 认识多继承编程方式
3. 正确识别继承中的类型兼容
4. 能正确判断继承中的构造、析构函数及成员初始化顺序；

实验软件和硬件环境：

软件环境 ：VSCODE + DEV-C++

硬件环境 ：Legion Y7000P

实验步骤与内容：

1. 练习例 7-1 公有继承。如果直接将其中的 init 函数改为构造函数，会出现什么问题？怎样改才是正确的？

请在分析实验附件中的代码中的问题，解释运行结果。

(1)出现问题: 直接改为构造函数, 会导致 Rectangle 的构造函数中创建了一个 point 类的对象, 而没有给派生类的基类变量赋值, 因此最终基类变量 x,y 仍然为默认值 0.

(2)修改方式 ：

```
Rectangle(float x, float y, float w, float h) : Point(x, y), w(w), h(h) {
        // 在构造函数的初始化列表中调用基类的构造函数，然后初始化派生类的数据成员
    }
```

(3)解释附件代码运行结果 ：

```
In Point constructor:0,0,this:0x62fe00
In Point constructor:2,3,this:0x62fda8
0, 0
The data of rect(x,y,w,h):
0, 0, 20, 10
The data of rect(x,y,w,h):
3, 2, 20, 10
```

原因: 没有给派生类的基类变量赋值

解决方式

```
class Rectangle: public Point { //派生类定义部分，因为继承会自动继承一个Point对象，并调用默认的构造函数。
public: //新增公有函数成员
    Rectangle(float a, float b, float w, float h) :Point(a, b) {

        cout << getX() <<", "<< getY() <<endl;  //输出矩形的特征参数

        this->w = w;
        this->h = h;
    }
    float getH() const { return h; }
    float getW() const { return w; }
private:    //新增私有数据成员
    float w, h;
};
```

输出结果

```
In Point constructor:2,3,this:0x62fe00
2, 3
The data of rect(x,y,w,h):
2, 3, 20, 10
The data of rect(x,y,w,h):
5, 5, 20, 10
```

2. 练习例 7-2 私有继承，其中哪些函数是函数是重新定义的同名隐藏？与函数重载有什么区别？如果没有重新定义，程序会出错吗？为什么？

运行结果如下

```
The data of rect(x,y,w,h):
5, 5, 20, 10
```

(1)getX() getY() move()函数均为同名隐藏

(2)区别 ：函数重载是指在同一个作用域中定义了多个同名函数，它们的参数列表不同。函数重载的函数必须在参数数量或类型上有所不同。而同名隐藏使得在派生类中直接使用成员名就只能访问到派生类中声明的同名函数，对参数的数量和类型没有特殊的要求。

(3)如果没有重新定义 move()、getX()和 getY()函数，程序不会出错，因为这些函数在基类中已经被定义了。但如果是私有继承，则会报错，因为无法调用私有成员.

3. 练习例 7-3，继承中的类型兼容。请解释运行结果。

如果在 main()中加入以下语句，会显示什么？

base2.display();

derived.display();

```
Base1::display()
Base1::display()
Base1::display()
```

(1)解释运行结果 ：

派生类对象指针可以转化为指向基类对象的指针，因此调用的函数均为基类 base1

(2)加入语句后显示如图

```
Base2::display()
Derived::display()
```

4. 练习 P28 页例，继承中的构造、析构函数执行顺序。请解释运行结果。

```
B's constructor called.
C's constructor called.
5
6
C's destructor called.
B's destructor called.
```

解释:
先运行 B 的构造函数, 再运行 C 的构造函数, 析构函数的运行顺序相反.
用 5, 6 初始化列表了 B 的值和 C 的值
之后用 B:: 这种父类名加作用域分辨符的形式，调用到了父类的 print 函数，因为直接调用 print 函数是错误的，原因是 C 类中的 print 函数同名隐藏了 B 类中的 print 函数。

5. 练习例 7-4，继承中的初始化执行顺序。请解释运行结果。

运行结果

```
Constructing Base2 2
Constructing Base1 1
Constructing Base3 *
Constructing Base1 3
Constructing Base2 4
Constructing Base3 *
```

对于构造函数而言: 按从左往右的顺序调用基类的构造函数, 因此顺序为 2, 1, 3
对于成员变量而言, 按定义的顺序调用成员变量的构造函数, 因此顺序为 1, 2, 3

6. 练习例 7-5 派生类析构函数，解释运行结果。

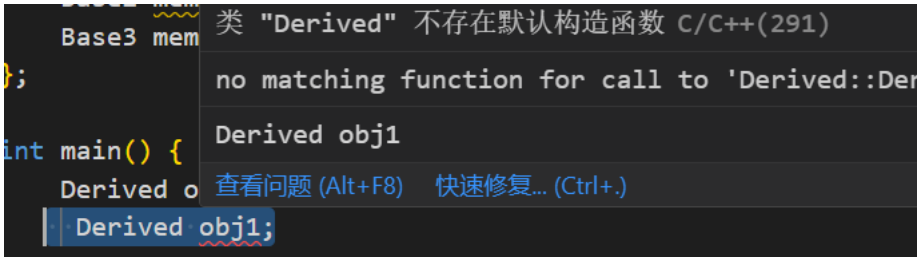
此题 main()函数中增加如下一句，会出现什么错误，请解释。

Derived obj1;

(1)解释运行结果 ：

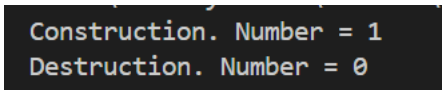
```
Constructing Base2 2
Constructing Base1 1
Constructing Base3 *
Constructing Base1 3
Constructing Base2 4
Constructing Base3 *
Destructing Base3
Destructing Base2
Destructing Base1
Destructing Base3
Destructing Base1
Destructing Base2
```

对于构造函数而言: 按从左往右的顺序调用基类的构造函数, 因此顺序为 2, 1, 3
对于成员变量而言, 按定义的顺序调用成员变量的构造函数, 因此顺序为 1, 2, 3
对于析构函数而言, 调用顺序与构造函数调用顺序相反, 因此为 3, 2, 1, 3, 1, 2
(2)增加语句报错



因为定义了有参构造函数, 则编译器不会自动生成默认构造函数

7. 练习继承派生中构造与析构函数的执行顺序，运行第 7 章实验 7，P98 实验任务（2）lab7_2.cpp。Lab7_2.cpp



按照对应顺序执行代码，首先是构造函数中 Number 的值由 0++成为 1，在析构函数中由 1--成为 0

8. 三种继承方式中，哪种子类的对象不用重写父类中的方法就可以直接替换父类的对象使用？（可替换原则）

答：公有继承(public)：在公有继承中，派生类继承了基类的接口和实现，并且可以被视为基类的一种类型。因此，如果使用公有继承，派生类的对象可以直接替换基类的对象使用，而不需要重写基类中的方法。这是因为派生类继承了基类的方法，并且可以直接调用这些方法

9. 练习例 7_1，不用继承，用组合 Point 的方式编写 Rectangle 类，总结与继承方式有什么不同？

```
class Rectangle {
public:
    Rectangle(float x, float y, float w, float h) : topLeft(x, y), width(w), height(h) {}

    float getWidth() const { return width; }
    float getHeight() const { return height; }

    // 其他与矩形相关的方法...

private:
    Point topLeft; // 使用组合的方式包含一个 Point 对象
    float width, height;
};
```

代码复用性：

继承：通过继承，派生类可以直接继承基类的接口和实现，从而实现代码的复用。

组合：使用组合时，Rectangle 类内部包含一个 Point 对象，但并不继承其接口和实现。因此，组合方式并没有直接提供代码复用性。

关系类型：

继承：使用继承时，子类与父类之间形成了 "is-a" 关系，即子类是父类的一种类型。

组合：使用组合时，Rectangle 类与 Point 类之间形成了 "has-a" 关系，即 Rectangle 类 "has-a" Point。

耦合性：

继承：当使用继承时，子类与父类之间存在较强的耦合性，子类的实现依赖于父类的具体实现细节。

组合：使用组合时，Rectangle 类与 Point 类之间的耦合性较低，因为 Rectangle 类只是使用 Point 类的功能，而不依赖于其具体实现。

结论分析与体会：

总体来说，通过实验和探索继承与派生的相关概念，我获得了以下收获：

1. **深入理解继承与派生：** 通过实际操作和分析代码，我加深了对继承与派生概念的理解。继承是面向对象编程中重要的特性之一，它可以实现代码的重用和扩展。

2. **掌握继承方式的特点：**了解了不同的继承方式（公有继承、私有继承、保护继承）的特点及适用场景，能够根据需求选择合适的继承方式。

3. **理解继承中的重要概念：**对继承中的重要概念，如构造函数与析构函数执行顺序、函数重载与函数隐藏、初始化执行顺序等有了更深入的了解。

4. **分析程序运行结果的能力：** 通过分析实验中的代码运行结果，我提高了分析问题和定位错误的能力，能够更好地理解程序的行为和调试代码。

5. **扩展探索能力：** 在解决问题和探索答案的过程中，我培养了扩展探索的能力，能够从多个角度思考问题，寻找最优解决方案。

总的来说，通过这次实验，我不仅加深了对继承与派生的理解，还提高了分析和解决问题的能力，为进一步学习和应用面向对象编程打下了良好的基础。

就实验过程中遇到的问题及解决处理方法，自拟 1－3 道问答题：

问题 ：

1. 如果派生类的初始化列表中含有基类的构造函数, 则基类构造函数会调用几次呢?

答 : 1 次, 显式调用基类构造函数只会调用一次

2. 组合关系和继承关系应该如何选择呢 ?

答 ：组合关系指的是一个类包含另一个类的对象作为其成员变量。这种关系表明一个类包含另一个类的对象; 继承关系指的是一个类从另一个类派生而来，并且继承了其所有的成员变量和方法。这种关系用于描述“是一个”的关系; 具体使用需要看具体的场景.