学号 专业 信息安全 姓名

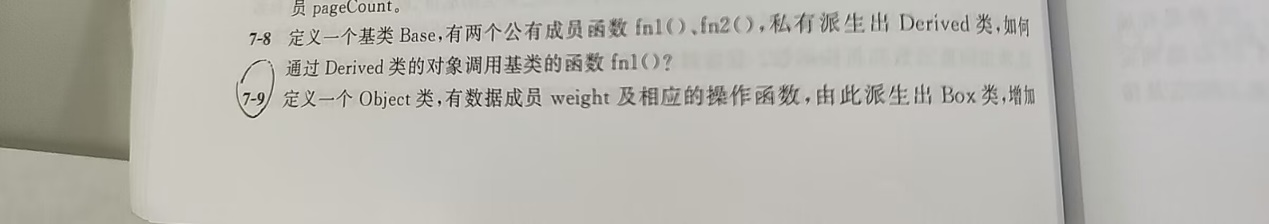
实验日期 **2024/05/27** 教师签字 成绩

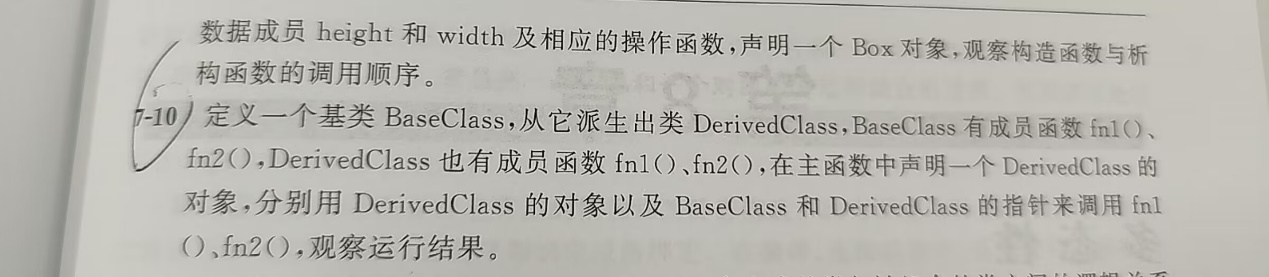
实验报告

【实验名称】 构造函数、析构函数与成员函数

【实验目的】

本次实验旨在掌握C++中构造函数与析构函数调用顺序，学生需理解基类与派生类的构造函数和析构函数的调用顺序。学会如何通过代码实现和观察对象的创建和销毁过程。同时学生需掌握成员函数的调用，掌握在继承关系中如何调用基类和派生类的成员函数，了解通过对象和指针调用成员函数的不同表现。【题目】





7-9和7-10

【实验原理】

**派生类构造函数和析构函数的调用顺序：**

在类中构造函数的主要作用就是对数据进行初始化，但是如果在派生类之中，如果对派生类中的新增加的成员如何进行初始化，就需要加入派生类中的构造函数，但是重要的一点是，基类的构造函数和析构函数是不能被它的派生类所继承的，所以说必须在派生类的构造函数中，也要对激烈的构造函数所需要的参数也同样进行设置，同时函数结束之后，照用的析构函数，对象进行收尾工作时，也需要加入新的析构函数来完成。当构造函数被调用时，会严格的按照先调用该基类中的构造函数，然后会调用派生类中的构造函数这个顺序来进行执行，其次析构函数的调用顺序，也就是对象生命周期结束时所调用的函数，而它的调用顺序与构造函数的调用顺序是完全相反的，它会先调用派生类中的析构函数，后来才会调用基类中的析构函数。

基于该原理，完成7-9与7-10两个实验。

7-9设计思想：

 **继承**：展示了派生类继承基类的成员函数，并可以选择重写这些函数。

 **覆盖**：派生类重写了基类的 fn1 和 fn2 函数，这个过程称为方法覆盖。

 **静态绑定**：由于没有使用虚函数，通过基类指针调用成员函数时，调用的是基类版本的函数。这展示了静态绑定（编译时绑定）的概念。

7-10设计思想：

**继承**：box 类继承 object 类，复用了重量属性和相关函数。

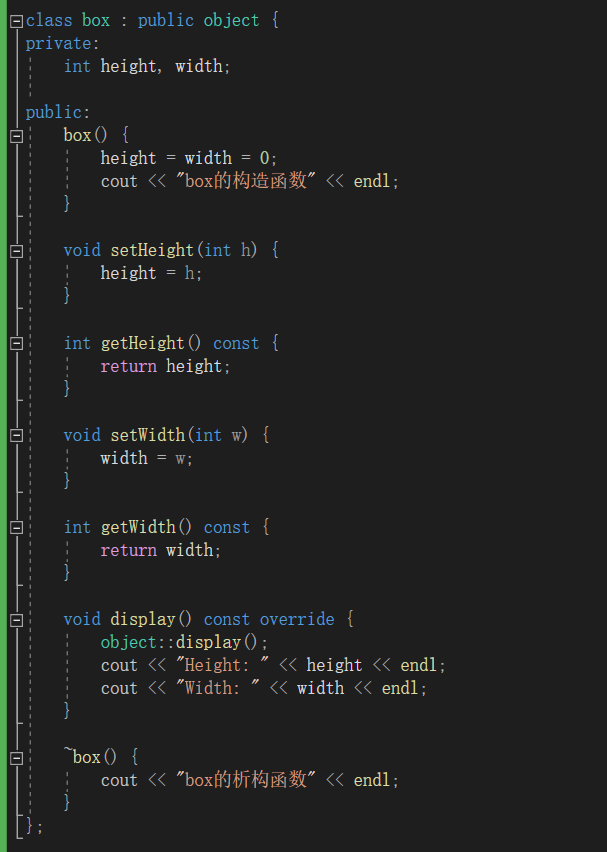
 **封装**：使用私有成员变量和公有成员函数，实现数据的封装和访问控制。

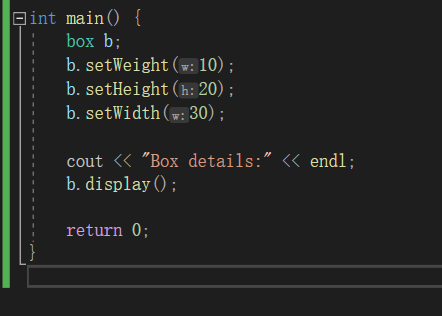
 **多态**：通过虚函数实现多态性，使基类指针可以调用派生类的函数。

 **构造与析构**：构造函数和析构函数用于对象的初始化和清理，确保资源的正确管理。

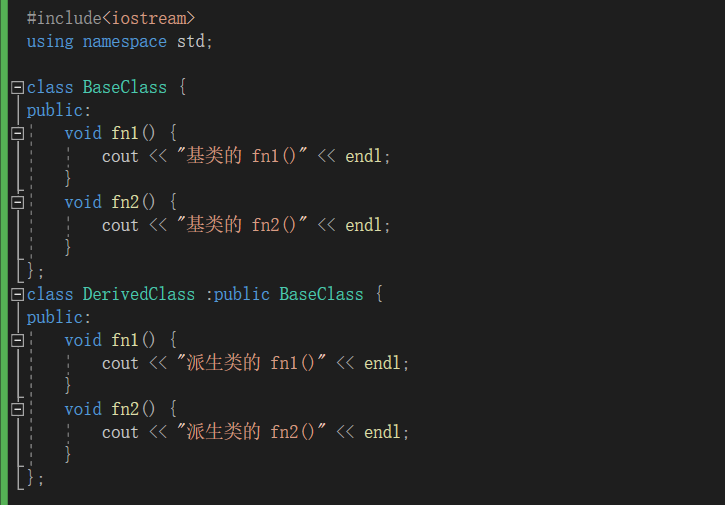
【源码】

7-9：





7-10：

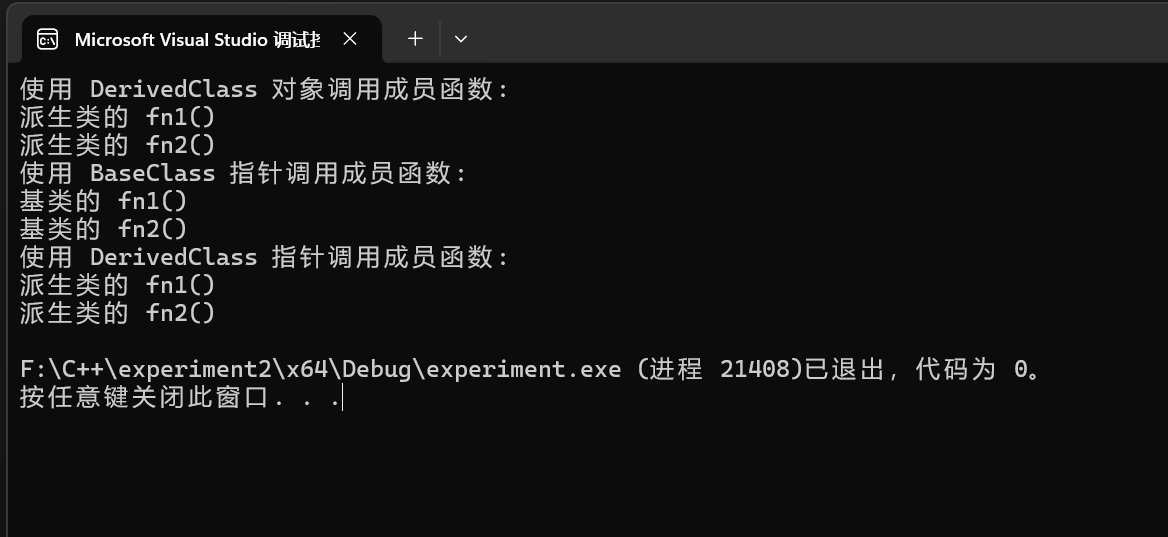




【运行结果截图】

7-9：

7-10：



【实验总结】

1. **构造函数与析构函数的调用顺序**

实验一展示了构造函数和析构函数的调用顺序：在对象创建时，先调用基类的构造函数，再调用派生类的构造函数；在对象销毁时，先调用派生类的析构函数，再调用基类的析构函数。这一过程符合C++语言的继承和对象生命周期管理规则，确保基类部分在派生类部分之前构造，并在之后销毁。

1. **成员函数的调用**

实验二展示了在继承关系中成员函数的调用情况：当通过对象直接调用函数时，调用的是派生类的函数；当通过基类指针调用函数时，调用的是基类的函数。这是因为在没有虚函数的情况下，指针调用函数是根据指针的类型来决定的，而不是指向对象的实际类型。这也说明了多态性的实现需要使用虚函数。

1. **对象生命周期管理**：

理解了构造函数和析构函数的调用顺序及其重要性。

学会了如何通过构造函数进行对象的初始化，以及通过析构函数进行资源的清理。

1. **继承和成员函数调用机制**：

了解了继承关系中成员函数的覆盖和调用规则。

通过对象直接调用函数时，调用派生类的函数；通过基类指针调用函数时，如果没有虚函数，则调用基类的函数。

1. **多态性**：

学会了通过虚函数实现多态性，使得基类指针可以调用派生类的重写函数。

理解了多态性的重要性及其在设计可扩展和灵活的代码中的应用。