学号 专业 信息安全 姓名

实验日期 **2024/04/24** 教师签字 成绩

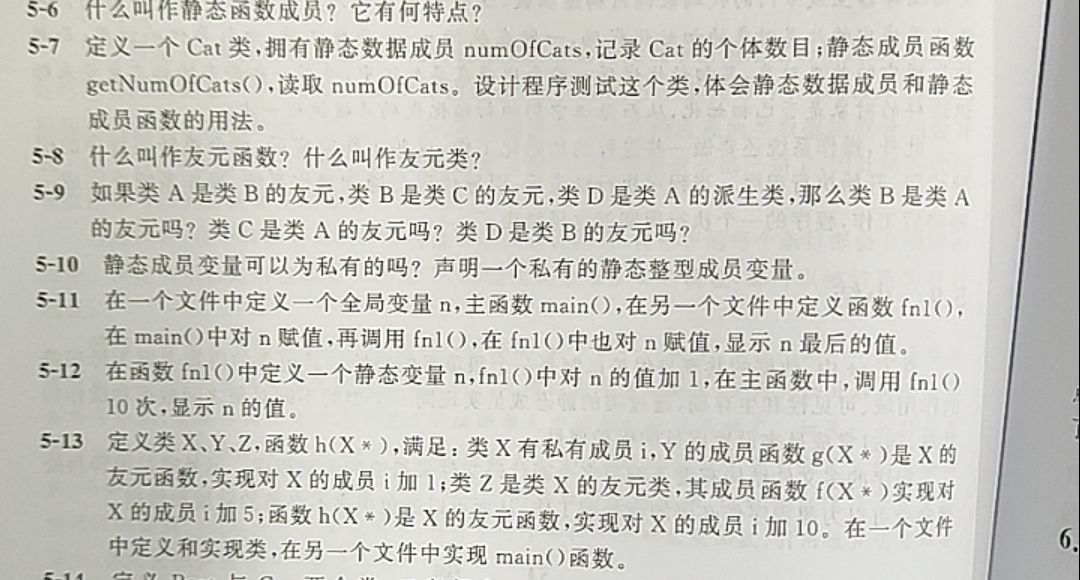
实验报告

【实验名称】 类，静态函数，友元函数

【实验目的】

本次实验的目的是深入理解C++中的类、静态函数和友元函数的概念及其应用。

【题目】



5-7和5-13

【实验原理】

1. **静态成员变量：**

　类的静态变量即是在类中，用关键字static声明成员变量。无论建立了多少个对象，都只有一个静态变量的拷贝。静态成员变量，不属于某一个类对象，而只是属于这个类，对于所有类对象时共享的。

**1. 类的静态变量的分配空间**

类的静态变量，在编译阶段就已经分配了空间，即还没有创建类对象时，就已经分配空间。同static变量一样，类的静态变量也是在编译阶段就分配好了空间。

**2. 在为类对象分配空间中不包括静态成员所占空间。**

静态变量并不属于某个类对象，它是共享的，因此独立的类对象分配空间是不包括静态成员的空间的。类的静态变量可以通过类对象来访问，也可以通过类名直接访问。

1. **静态成员函数：**

1.静态成员函数就是在声明时前面加了 static 关键字的成员函数。  
2.静态成员函数没有 this 指针，不知道指向哪个对象，无法访问对象的成员变量。  
3.静态成员函数可以通过类来直接调用(或者通过对象名.函数名)。  
4.静态成员变量只有一份，被所有同类对象共享。

**3. 友元函数**

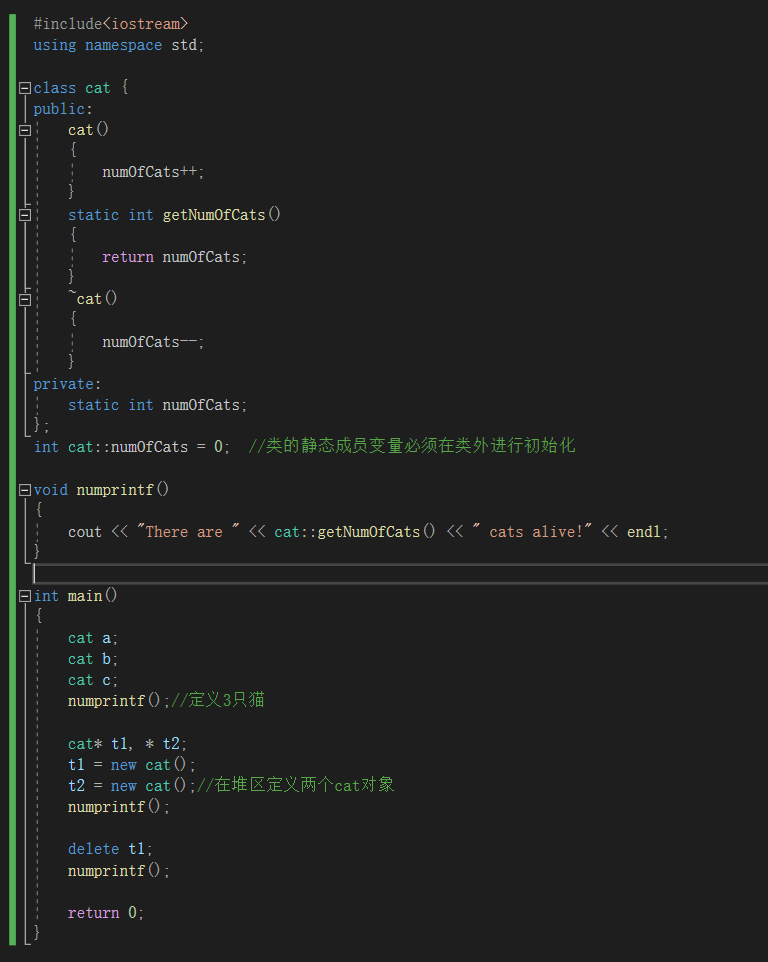
类的友元函数是定义在类外部，但有权访问类的所有私有（private）成员和保护（protected）成员。尽管友元函数的原型有在类的定义中出现过，但是友元函数并不是成员函数。

友元可以是一个函数，该函数被称为友元函数；友元也可以是一个类，该类被称为友元类，在这种情况下，整个类及其所有成员都是友元。

如果要声明函数为一个类的友元，需要在类定义中该函数原型前使用关键字**friend**

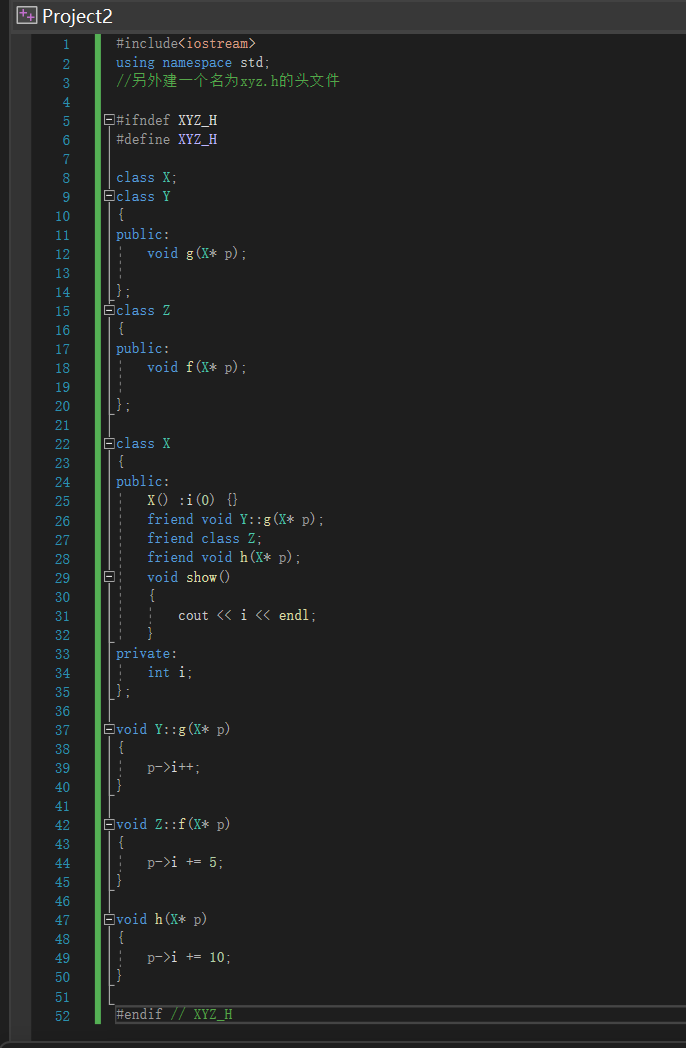
【源码】

5-7：

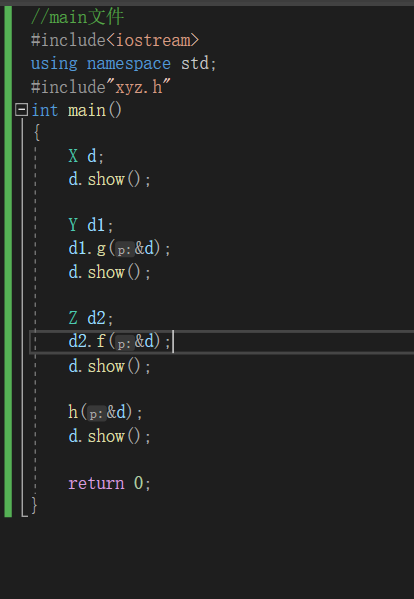


5-13：

xyz.h:

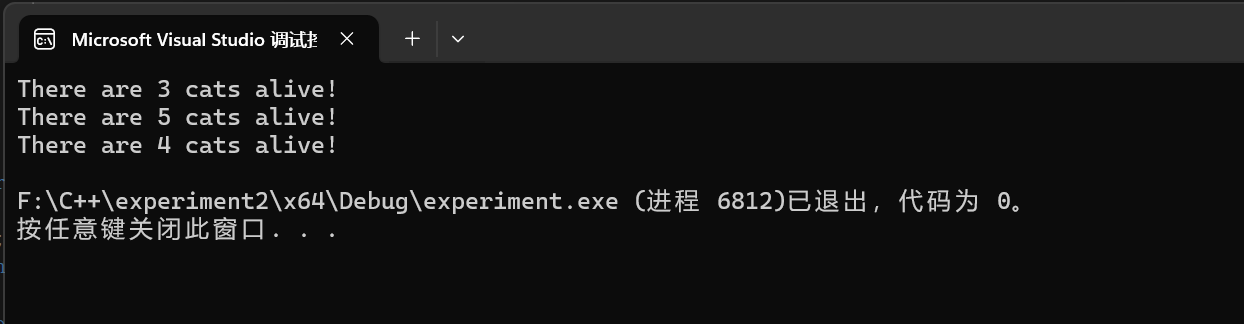


main.cpp:



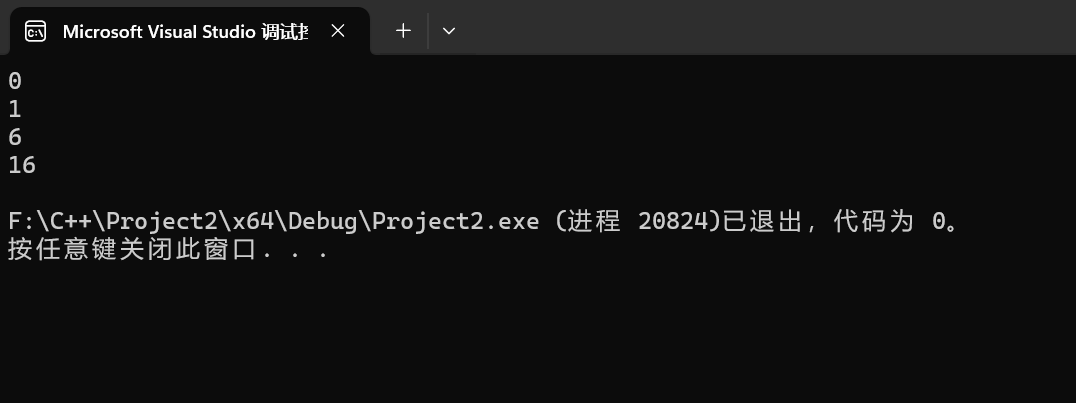
【运行结果截图】

5-7：



5-13：

确保三个文件在同一个目录下



【实验总结】

在实验中，我首先定义了几个类，每个类都有其独特的功能和数据成员。我为这些类添加了静态成员变量和静态成员函数，用于实现类级别的数据共享和操作。此外，我还实现了一些非成员函数作为友元函数，它们可以访问类的私有成员，但不属于类的一部分。

在实现过程中，我遇到了关于静态成员初始化和友元函数声明的问题。对于静态成员的初始化，我将其放在了类的外部定义中，确保了其在程序开始时就被正确初始化。对于友元函数，我确保了在类定义中正确地声明了友元关系，这样非成员函数才能访问类的私有成员。

本次实验加深了我对C++中类的设计、静态成员的使用以及友元函数的实现的理解。我认识到静态成员在实现类级别的数据共享和操作中的重要性，同时也意识到过度使用友元函数可能会破坏类的封装性。此外，实验还让我体会到了在面向对象设计中权衡封装性和功能需求的重要性。具体案例总结如下：

1. 静态数据成员numOfCats用来给cat类的对象个数进行计数，其被cat类的所有对象共同维护和使用，数据共享。

在类外可通过“类名::静态数据成员标识符”调用静态数据成员，而且类的静态成员变量必须在类外进行初始化，不能在类内进行初始化。

1. 头文件中 #ifndef #define #endif条件编译指令用途：防止头文件被多次定义，需用唯一的标识符来标记某文件是否已参加过编译，保证头文件中定义的变量不存在重复声明或定义。标识符一般大写来防止与其他变量冲突，便于区分。
2. 友元关系提供了不同类或对象之间的数据共享机制，从字面意思也可以看出来，像“朋友”一样，开了一个绿色通道，让函数g(X\*)、f（X\*）、h(X\*)能访问到类X的私有成员i。还需要注意的，友元函数并不属于这个类本身，故这个类本身的对象是无法访问到这些友元函数的。无论是友元函数还是友元类。都不能使用类内的this指针，同时也不可以被继承，如同父亲的朋友不一定是儿子的朋友这个道理。