# C++程序设计：实验报告

学号：2014211218

班级：2014211304

姓名：史文翰

1. **分程序详述及感想**
2. **猜价格游戏：**

没什么好说的，在这个程序中学会了C++一些标准的开头语，如iostream和using操作符，不过值得注意的是，我对类似的命名空间进行了深入的考究，发现namespace的用法与.h类似，而且更加直观方便，如一些教材上提到的，using一个命名空间有时候是没有必要的，可以使用::操作符来因地制宜。如 std::cout<<...。

1. **点类和矩形类：**

这是第一次接触类，同时也入门了组合类的概念：有些时候一个类需要作为另一个类的成员而出现。这就不得不精心设计接口，包括构造函数和拷贝构造函数（程序里没体现出来），当然还有析构函数，如果使用了new操作符的话就需要格外小心。

1. **友元：**

友元函数使得外部函数可以直接访问类的私有成员，这看似破坏了OOP的数据封装准则，但正如老师所讲，这其实是为了给用户以更好的接口来使用类。我查阅了一些书籍，友元在一些操作符的重载上格外有用，一些直接对内部数据起作用的（例如重载向量类中的“+”）操作符如果使用友元，那么它的定义会更加清晰。但在绝大多数情况下，友元应谨慎使用。

**4.5.6.从数组实现到指针再到类的矩阵：**

这三组实验分别要求用不同的类型（结构）来实现矩阵。数组实现最为简单，同时也最不容易出错，如果是数值类二维数组，我们应该多使用数组来实现。而指针的实现涉及到了new和delete操作符的使用，重要的是，我们申请的动态内存，使用完毕后要及时归还。仍然有一些细节需要注意：比如说delete时，应该先对行数组delete，再对列向量整体delete，先后的逻辑顺序决定你的程序没有僵尸内存。至于类，在类中积极的使用了new方法，但现实中不值得提倡。

值得注意的是，涉及到new的类的一些接口的设置需要格外小心，比如拷贝构造函数（未实现）这需要所谓的“深度复制”，而不能仅仅的复制指针。

**7.类的继承：**

类的公有继承需要我们仔细设计构造函数的传参问题，本质上是先使得父类被合理地构造，其次才是子类。析构时道理相同，先析构子类，最后析构父类。在这样的层次下，拷贝、析构函数如果涉及new运算符时将更为复杂。

对于类指针的问题，我们放到虚函数那个程序上做详细讨论。

**8.虚函数：**

我们沿用7的程序，在最后，我们申请了父类的两个指针，分别指向一个父类和他的一个子类

Rectangle \*pRec,\*pSqur;

pRec=&R1;

pSqur=&S1;

在写下这段代码的时候，我心里就怀疑编译器是否能正确识别函数，也就是它是否知道它应该做什么，因为我们将要调用得Area函数是同名继承的。事实上，由于没有动态联编，编译器确实不知道该如何做，得到的结果是打印出来的数据均是父类的，也就是说，他们都调用的是父类的Area方法。

解决这个问题需要使用虚函数，我们让Shape类的Area方法变为纯虚函数，这样，Shape类也成为了抽象类，抽象类不应该存在对象实体，我们也没有这样做。

float Area(){};//pure virtual and shape is abstract

实际编译后，程序可以正确打印两个图形的面积。

更一般地，我们可以用一个小测试程序来解释它：

class A{

public:

void print(){cout<<"This is A"<<endl;}

};

class B: public A{

public:

void print(){cout<<"This is B"<<endl;}

};

int main(void)

{

A a;

B b;

A\* p1=&a;

A\* p2=&b;//the ptr of the base class

p1->print();

p2->print();

return 0;

}

运行效果是，程序均打印了This is A，与我们预想的完全相同。

**9.重载++和--：**

对于重载++和--这种一元操作符，如果它是前置的，很好说，我们像当初重载负号运算符那样，将其设置为类的成员函数，参数数量为0，返回一个类的引用即可。这符合我们一般的重载操作符时的要求和准则。

Point& operator ++(void);

但是如果它是后置的，我们不能这样做，正确的做法是传递一个虚拟的int参数来弥补空缺。

Point& operator ++(int);

对于操作符的重载，我也做了一些课外的功课：

一些操作符我们希望忽略它的顺序以突出本质，比如\*操作符，我们发现如果在类方法中定义重载，那么由于返回一个类引用，我们只能这样做：

Otherclass=someclass\*I;

而不能反过来，这是由于本质上是这样的：

Otherclass=someclass.operator\*(I);

明白这一点，我们得到了一个好消息和一个坏消息，好消息是这种函数调用可以使得我们连乘一些参数，比如这样的是允许的：

Otherclass=someclass\*I\*J;

坏消息是，我们想这样写就不对了：

Otherclass=I\*someclass;

而我们想让\*操作符更具一般性，这里我们不得不在类之外再次声明一个新的重载操作符函数，仍然是对\*而言，不过它return一个class\*int；这样我们的问题就解决了。

一些重载操作符用友元的话会更清晰，我们在上面也提到过了。

还有对于<<和[]的重载也格外有用，在此就不赘述了。

**10.测试fstream：**

Ifstream和ofstream为我们的文件操作开辟了新的思想：用流来操作文件。其实我们惯用的cout和cin，不过只是标准输入输出流的一个小小的部分。

当一个变量被声明为文件对象，那么很多文件控制方法可以直接调用成员函数，还有一些格式控制函数，都非常实用，程序以及注释中已经有说明，在此不赘述了。

**11.为文件添加行号：**

在本代码中最终要的成员函数是getline，用于读取文件中的一行到我设定的缓冲区数组中。本代码还有很多不足的地方，例如缓冲区的设置是个数组，有可能会发生溢出，没有溢出处理，并且行号的设置也不是很灵活，最多可以处理10\*n-1行，需要程序员根据情况设定。

**12.综合设计电话本：**

这个综合设计下了比较多的功夫，包括完备的构造函数、拷贝构造函数和析构函数。并且采用了组合类的方式实现，每一条记录模拟一个note类，而book类又是一个装载了note类数组的另一个类。Note类负责单记录的初始化，set和get操作，用于信息交互，而book类更像是一个抽象的电话本，包括了打印、查找、删除、追加、从文件读取和保存到文件六种操作，这六种操作也在main函数中设计的用户接口——菜单上有所体现。但是有些地方还深感不足，由于编程之时功力欠缺，没有去拓展一些更便捷的功能，例如排序、重载<<之类的。此外，每条记录用字符数组实现，并没有采用动态内存以较少错误的产生。总体来看，是一个低级且基础的设计，距离优秀精巧的设计还差得很远。

1. **总体收获与心得**

综上所述，通过这些实验，让我基本了解了C++的运作方式和代码模式，可以编写简单的程序。但其实我的C++之路才刚刚开始，在课余时间我正在自习更多关于C++的知识，比如近期接触到的重载类型操作符、explicit声明、关闭默认函数的方法、对[]和<<的重载、编写自己的命名空间等等，深知编程唯有不断实验才可融会贯通，达到真正的水平的提高，而这些程序就是实验的开始。

抛开这些可喜之处，在这些实验中仍有很多问题等待我去解决：是否考虑到了自赋值问题？类的接口是否合理？需要用new的是否偷懒而用了数组？与用户交互的接口是否还有待人性化？……不过也正是因为这些问题的发现，才使我在这条路上越走越远，收货也越来越多。