实验报告

实验目的：

1、掌握运算符重载的含义；

2、理解使用成员函数重载与友元函数重载运算符的区别；

3、掌握单目运算符、双目运算符使用成员函数、友元函数的重载方法；

4、理解被用户重载的运算符，其优先级、运算顺序与原先系统中的一致性。

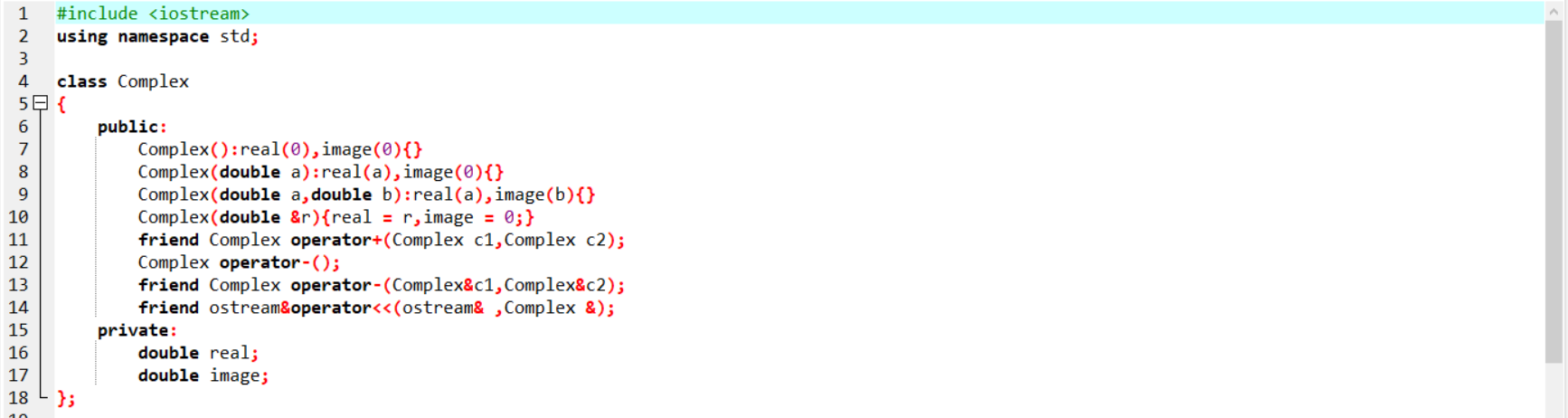
实验内容：

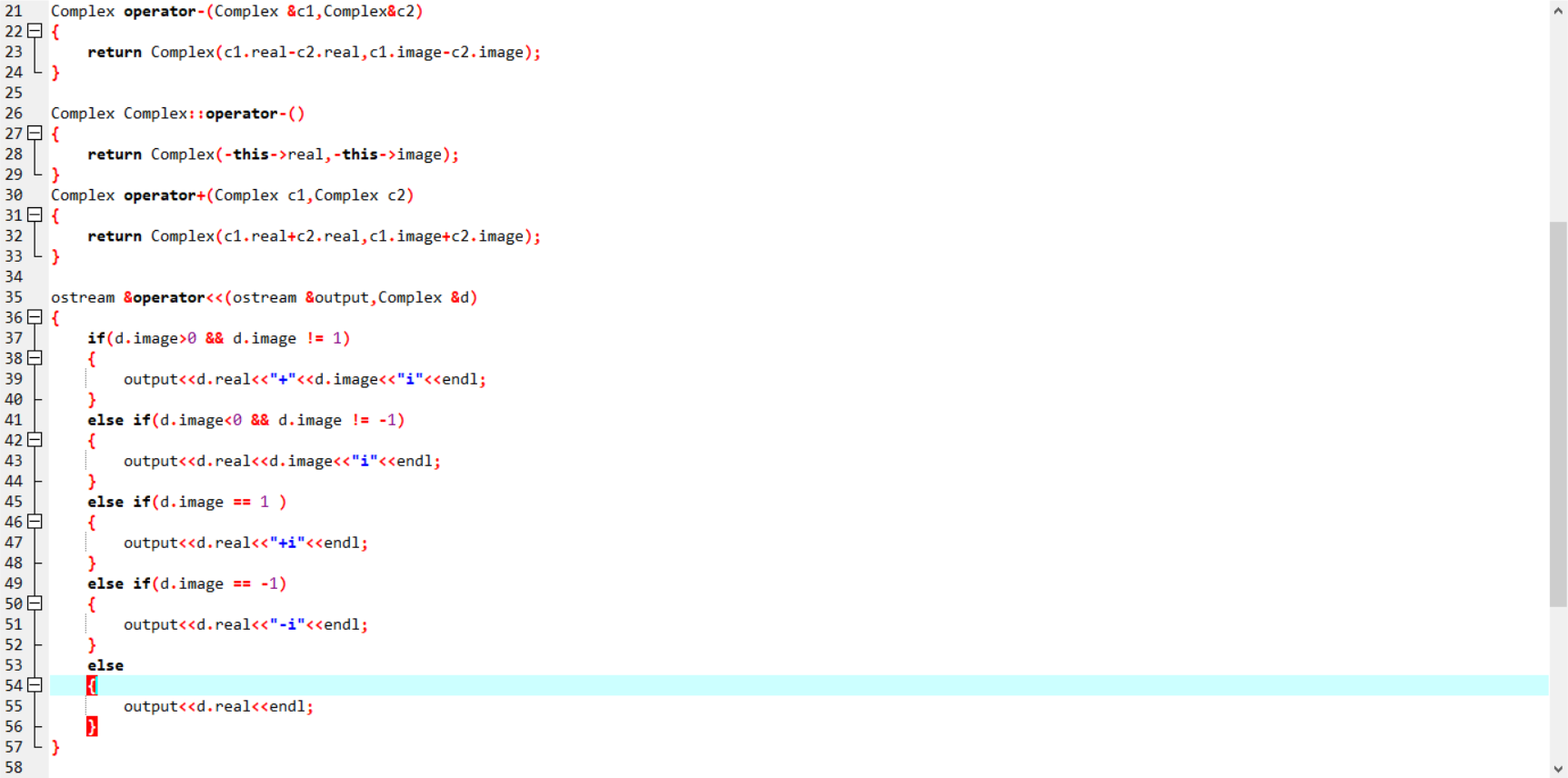
1. 下面是测试Complex类的程序和输出结果，请设计并实现符合本例要求的Complex类。

实验过程：

1. 对类初始化；
2. 按需重载+，- 以及<< 运算符；
3. 测试输出；

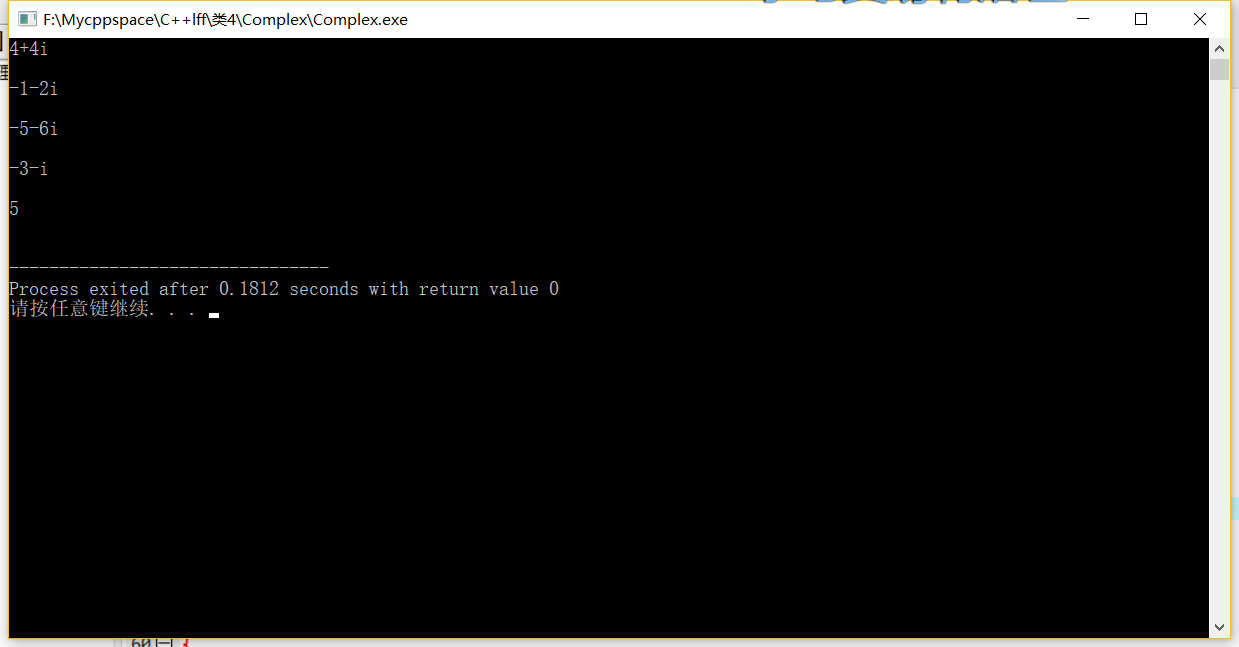
程序代码：







运行结果：



结果分析：

输出的第一行1+4i

是c1=c1+c2+c3;

cout<<c1<<endl;的结果，是对+运算符重载使其可以实现Complex类对象的加法

输出的第二行-1-2i

是c2=-c3;

cout<<c2<<endl;的结果，是对-运算符进行重载实现了对Complex类对象取负值

输出的第三行-5-6i

是c3=c2-c1;

cout<<c3<<endl;的结果，是对-运算符进行重载实现了Complex类对象相减的功能，同时拷贝

输出第四行-3-i

是c4=c4-c1;

cout<<c4<<endl;的结果，是对运算符-运算符进行重载实现了Complex类对象相减的功能，同时拷贝

第五行5

是c5=c5+(c4+4);

cout<<c5<<endl;的结果，首先是将（c4+4）中的Complex型+double型转变成Complex型+Complex型，即将4转变为Complex（4，0），再相加。

实验内容：

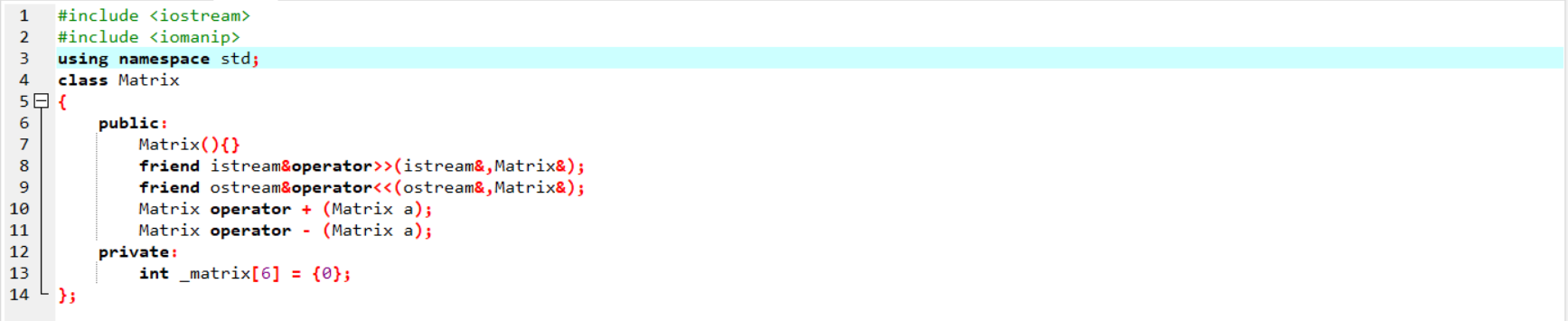
1. 下面是测试Matrix类的程序和输出结果，请设计并实现符合本例要求的Matrix类。

实验过程：

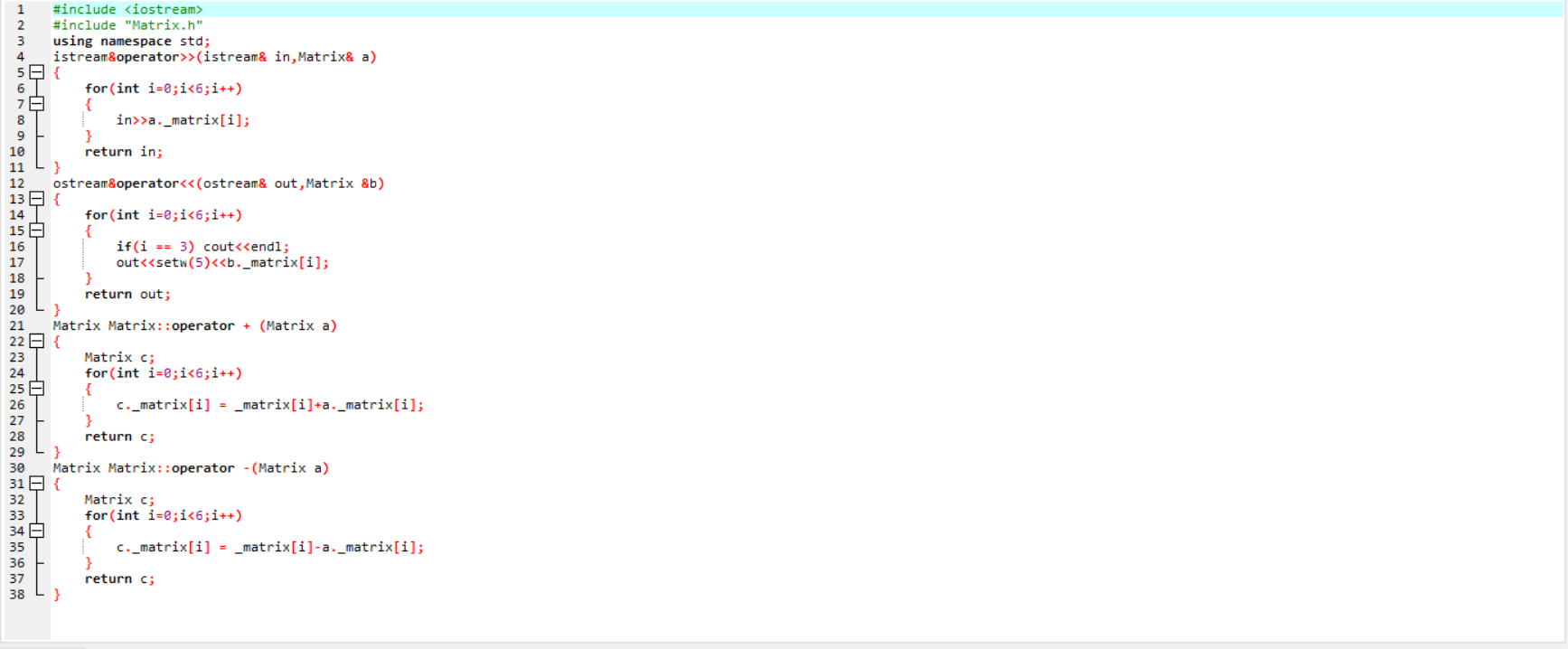
1. 对Matrix类进行定义
2. 按需重载<<,>>,+,-四个运算符,使其可以满足矩阵的运算
3. Main函数中测试输出

实验代码：

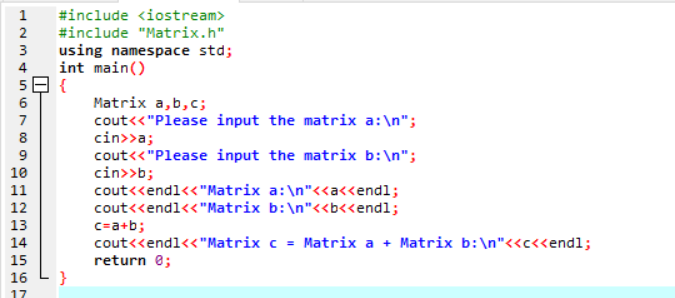
1. 头文件

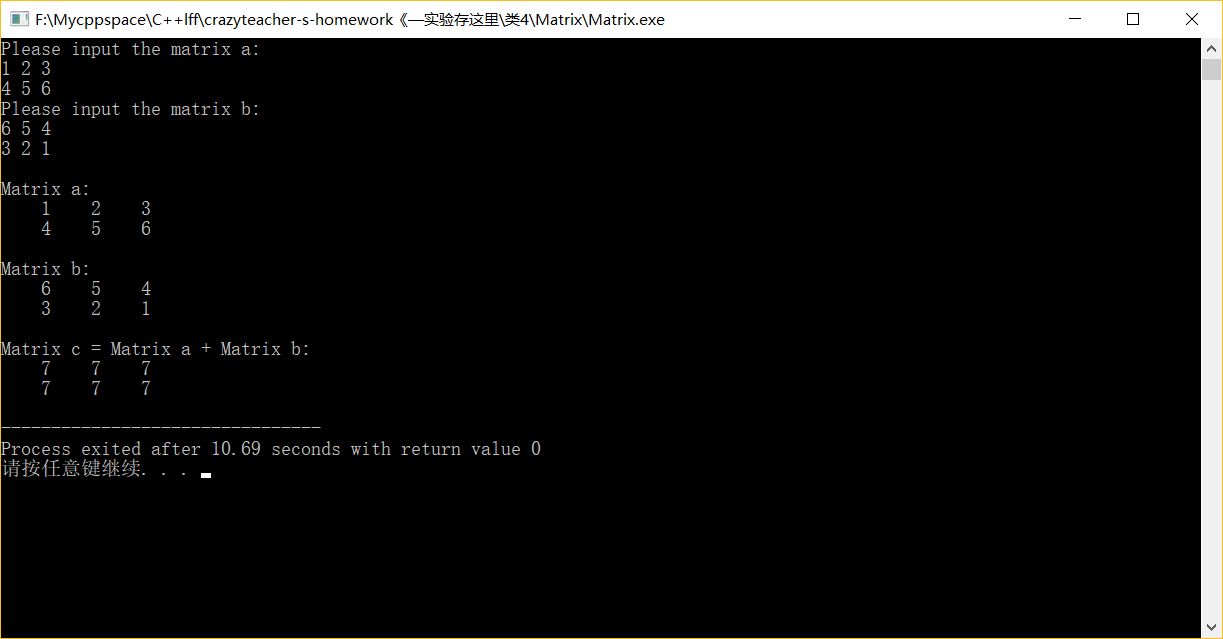


1. Matrix函数：



1. main函数部分



运行结果：  


由键盘输入两个矩阵的数值输出并输出他们本身和他们的和。

结果分析：

首先是一般的<<,>>不能对Matrix类进行，对其进行重载使其满足对Matri类的输入输出，因为常规的+，-不能对矩阵类进行运算，所以首先重载，使矩阵各个位置的数可以相加减。

Ps：写完之后才发现main函数里的测试没有相减这个要求。

实验内容:

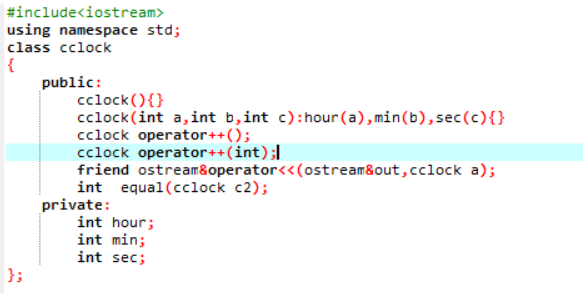
3、下面是测试CClock类的程序和输出结果，请设计并实现符合本例要求的CClock类。

实验过程：

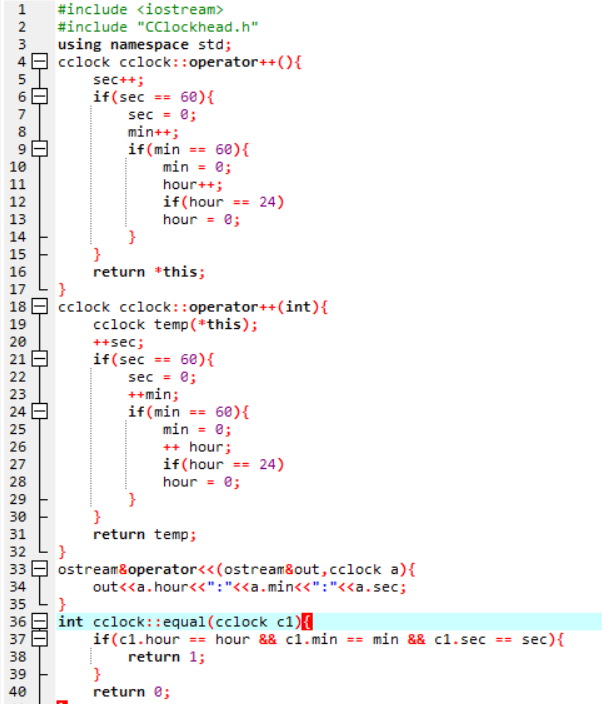
1. 对运算符++，<<,==进行了重载。
2. 完成测试。

实验代码

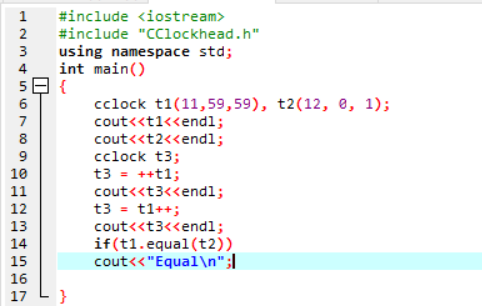
1. 头文件



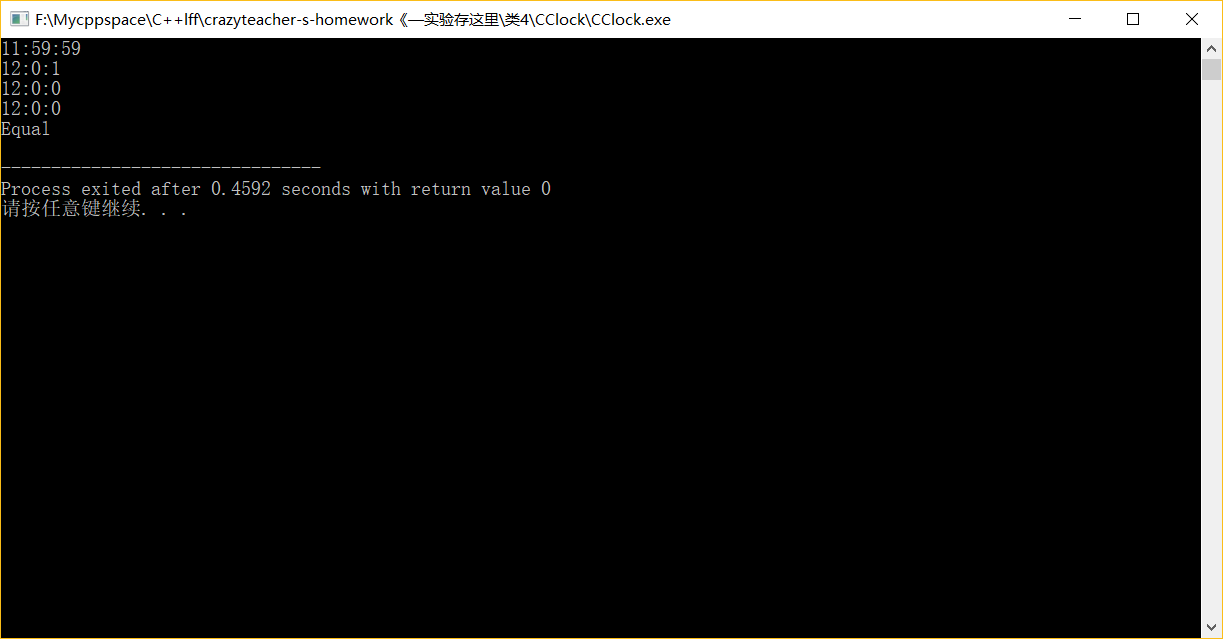
1. 函数部分



1. main函数



运行结果：



结果分析：

总体来说还是有一定的难度的，再一个就是要区分前++和后++的区别，operator++()和operator++(int),前者直接对\*this对象进行“+1”，返回的是引用，后者先创建一个对象的副本，再对对象进行“+1”，最后返回副本对象。

<<输出重载和前面道理一样，不表。

实验内容：

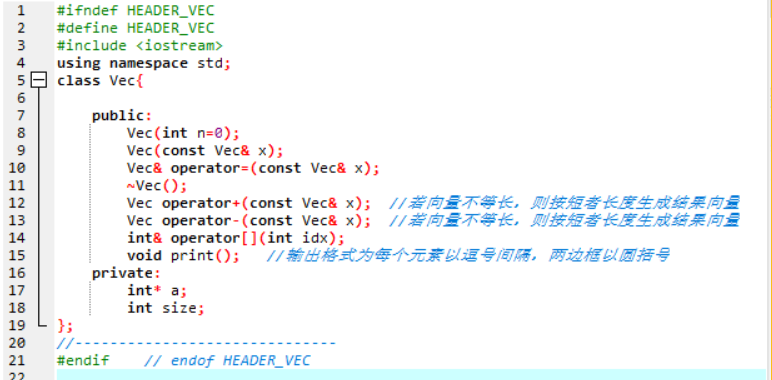
1. 下列程序工程由代码app.cpp，vec.h，vec.cpp三者构成。代码app.cpp中使用了自定义的向量类Vec，其在vec.h和vec.cpp中定义和实现。

实现过程：

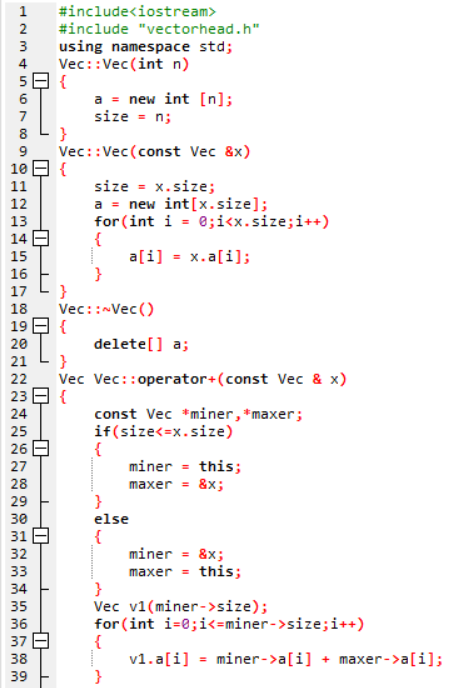
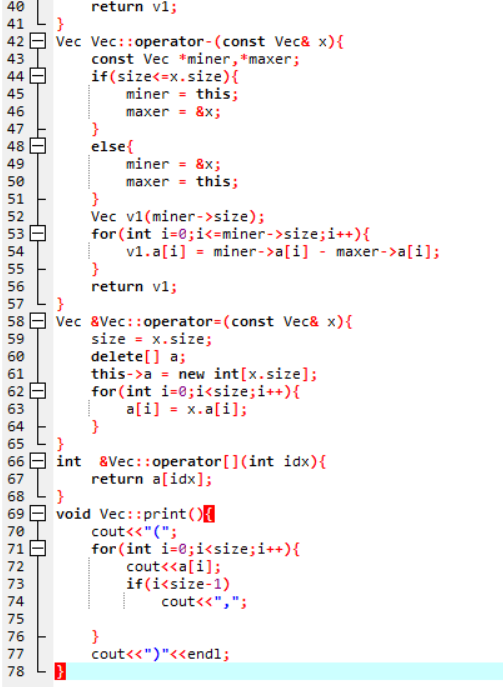
1. 向量的+，- ，还有对[]的重载。
2. Print函数

实验代码：

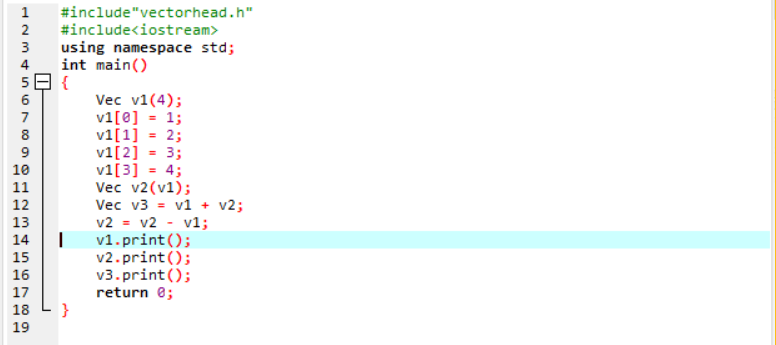
头文件部分



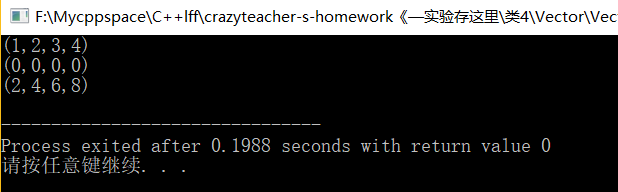
函数部分

Main函数：



运行结果：



结果分析：

对加和减的处理稍微点懒，只考虑短的部分，后面面看实验要求应该也是符合的，以前没有用过stl库里的vector，所以不是很了解这个东西，写起来有点陌生。最后看起来也就是像向量的操作。

实验感受：

感觉对函数重载这块掌握的还不是很好，尤其是二义性这个东西，一开始卡住我第一个小实验一天，最后上了课才知道是因为出现歧义了才有这样的报错。还有就是感觉有时候写成友元函数还是很方便的，避免了很多奇怪的错误。菜还是自己菜，平时得多练。。。