## 实验一 Classes：A Deeper Look，PartI

## 学号: 09017423 姓名：杨彬

### 实验目的

1．掌握类的定义和操作（包括常成员函数）

2．掌握对常量对象的访问

3．掌握使用友元访问对象的私有数据成员

4．掌握静态数据成员和成员函数

### 实验内容

**Exp1.**

Description of the Problem

(Complex Class) Create a class called Complex for performing arithmetic with complex numbers. Write a program to test your class.

Complex numbers have the form

realPart + imaginaryPart \* i

where i is 1−

Use double variables to represent the private data of the class. Provide a constructor that enables an object of this class to be initialized when it is declared. The constructor should contain default values in case no initializers are provided. Provide public member functions that perform the following tasks:

(a) Adding two Complex numbers: The real parts (实部) are added together and the imaginary parts (虚部) are added together.

(b) Subtracting two Complex numbers: The real part of the right operand is subtracted from the real part of the left operand, and the imaginary part of the right operand is subtracted from the imaginary part of the left operand.

Printing Complex numbers in the form (a, b), where a is the real part and b is the imaginary part.

**Exp2**

(Enhancing Class Time) Modify the Time class of Figs. 9.89.9 to include a tick member function that increments the time stored in a Time object by one second. The Time object should always remain in a consistent state. Write a program that tests the tick member function in a loop thatprints the time in standard format during each iteration of the loop to illustrate that the tick member function works correctly. Be sure to test the following cases:

􀁺 Incrementing into the next minute.

􀁺 Incrementing into the next hour.

􀁺 Incrementing into the next day (i.e., 11:59:59 PM to 12:00:00 AM).

注意以下额外要求：

Then change the tick member function to a friend function of class Time, which will access the private data member of Time directly. You should get the same output as above.

friend void tick(Time &t); // increment one second

**Exp3**

(HugeInteger Class) Create a class HugeInteger that uses a 40-element array of digits to store integers as large as 40 digits each. Provide member functions:

(a) Constructor, destructor

(b) input, output, add and substract

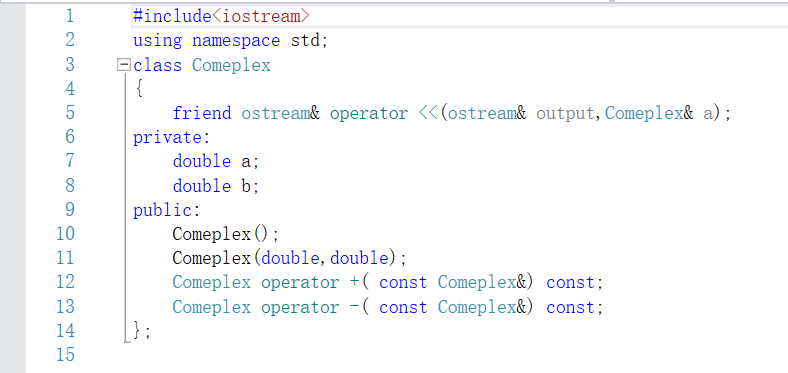
(c) For comparing HugeInteger objects, provide functions isEqualTo, isNotEqualTo, isGreaterThan, isLessThan, isGreaterThanOrEqualTo and isLessThanOrEqualToeach of these is a "predicate" function that simply returns TRue if the relationship holds between the two HugeIntegers and returns false if the relationship does not hold. Also, provide a predicate function isZero.

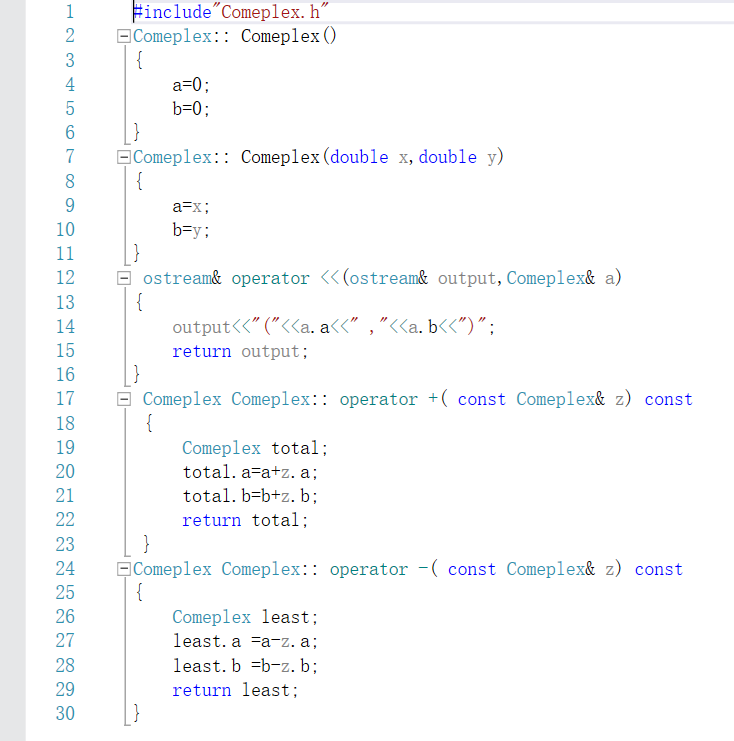
If you feel ambitious, provide member functions multiply, divide and modulus.

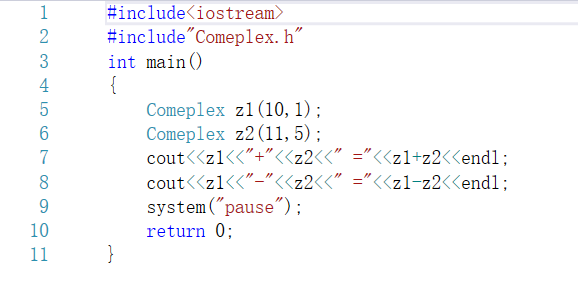
注：不考虑负数情况，即hugeintA-hugeintB确保hugeintA大于hugeintB；而hugeintA+hugeintB，确保不溢出

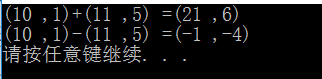
### 实验代码及结果：

**Exp1:**



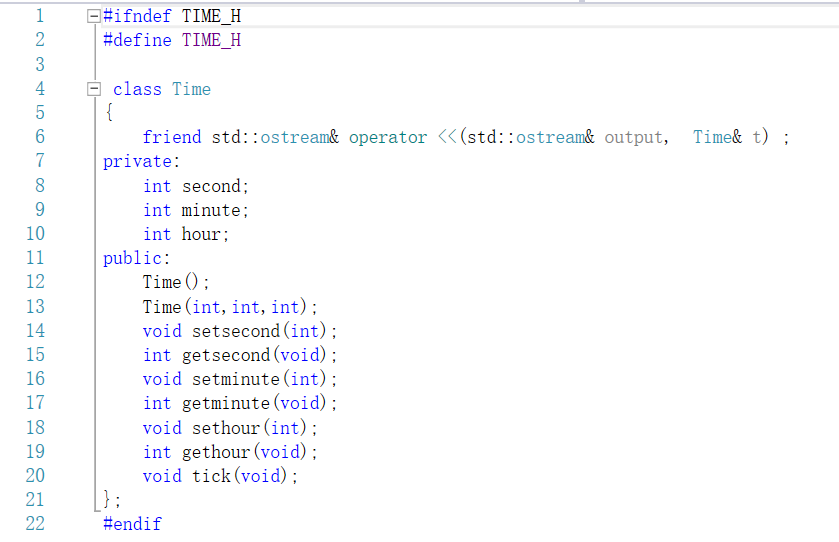


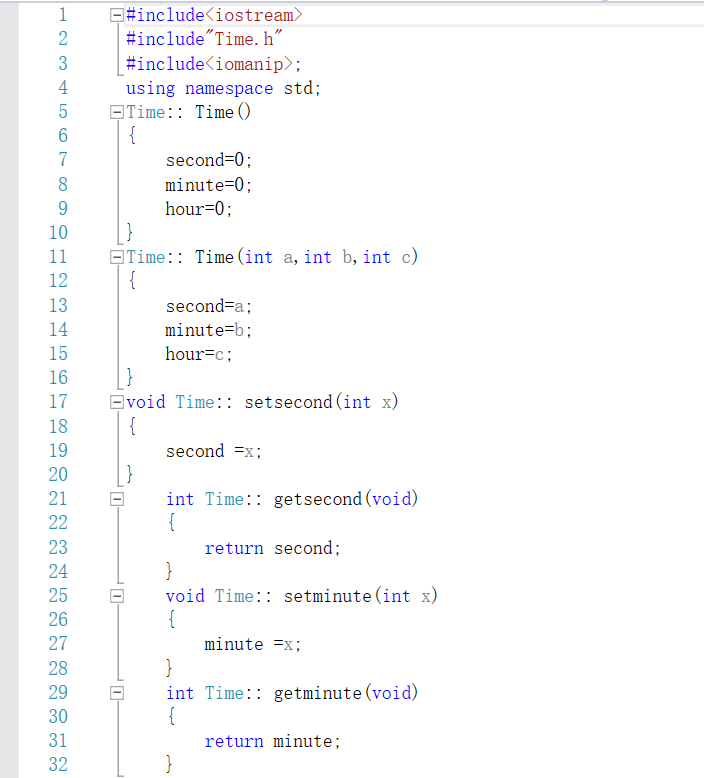


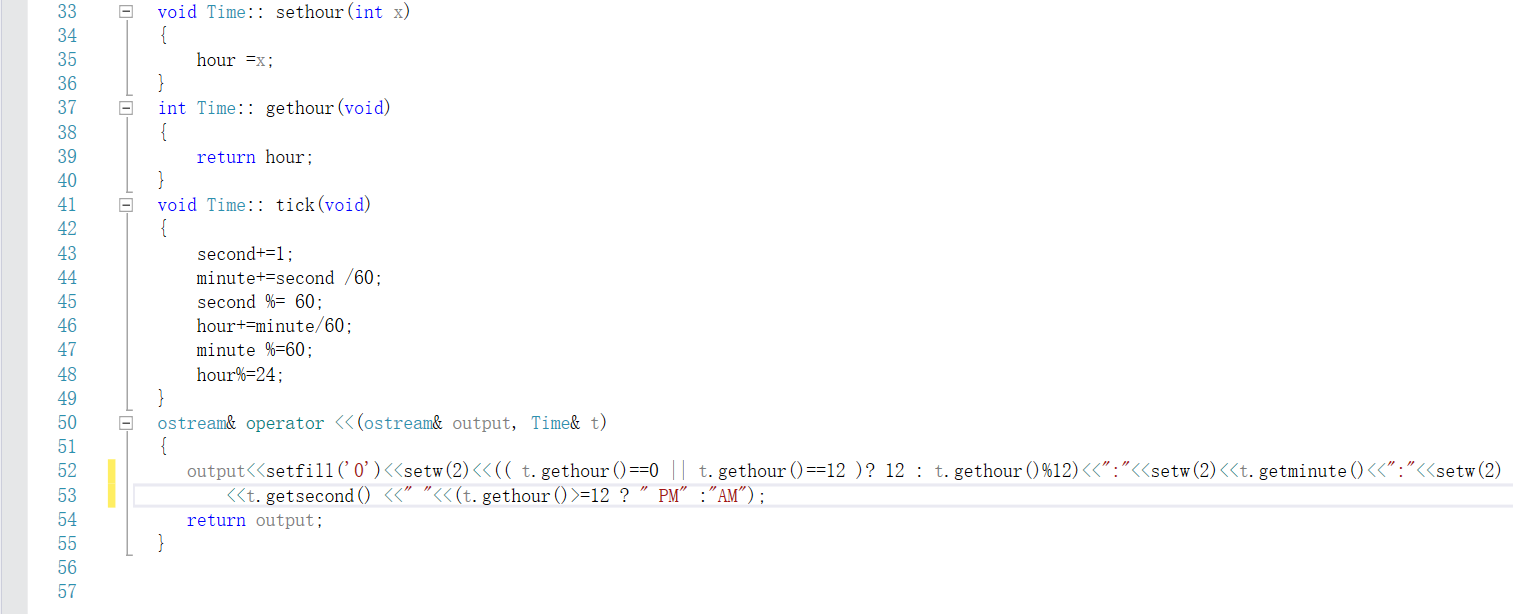


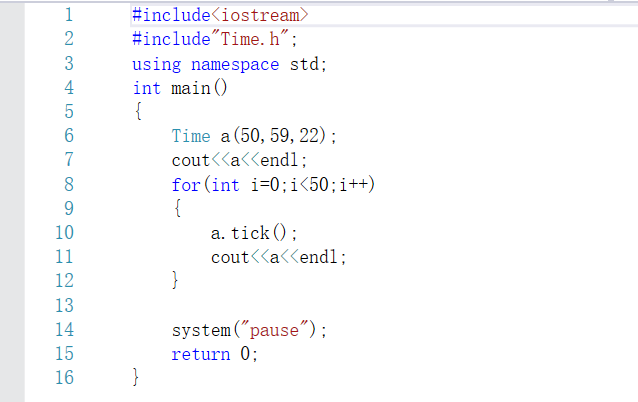
**Exp2:**

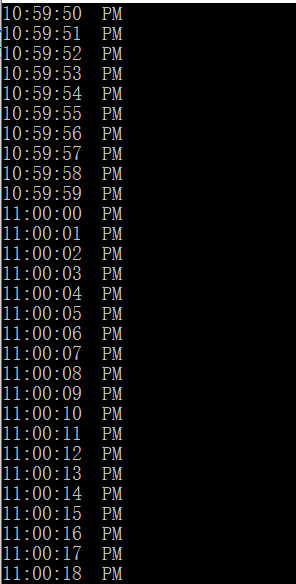
**2.1:**



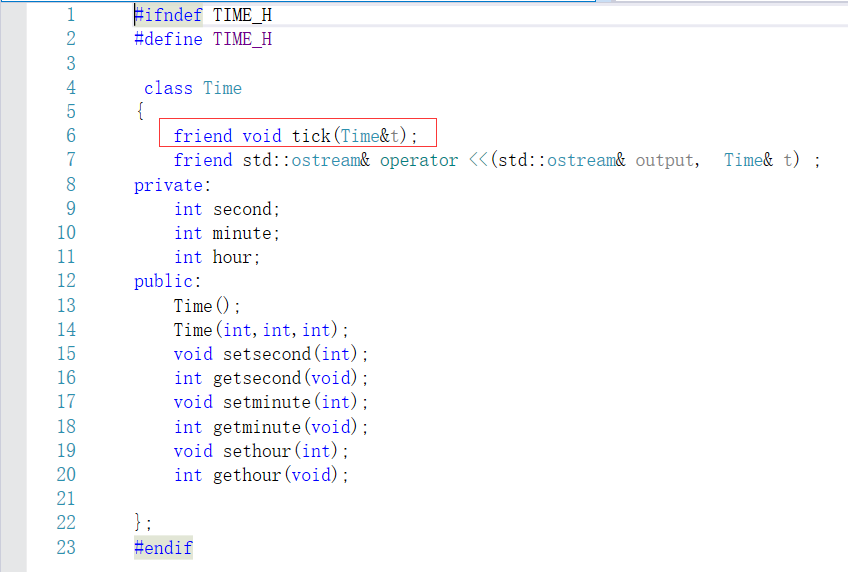


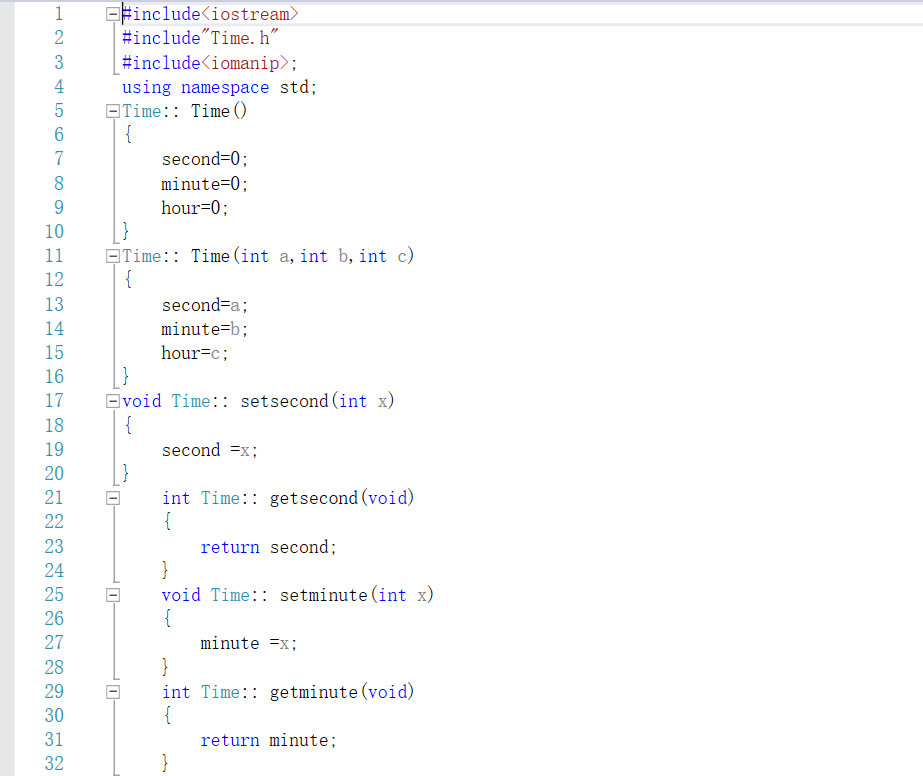


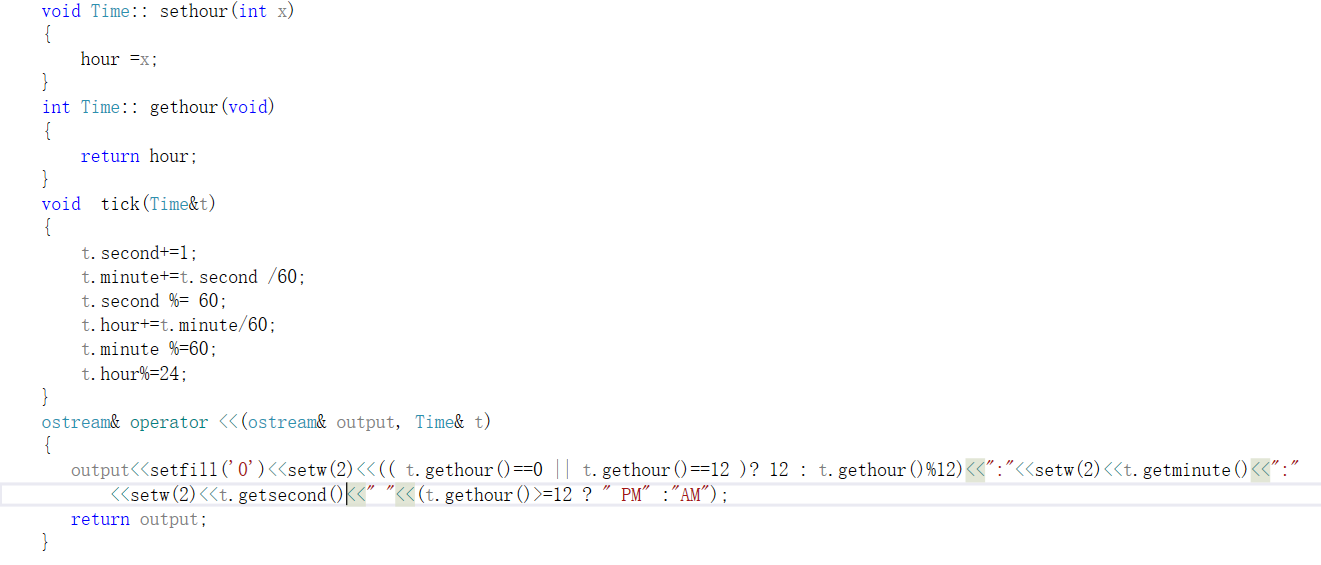


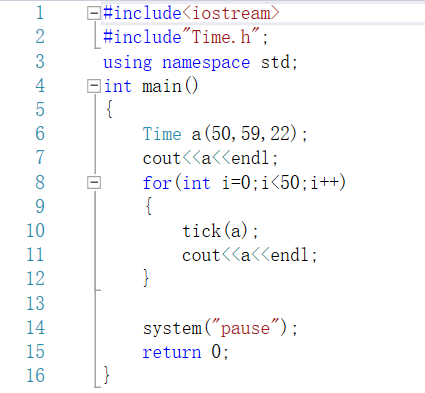


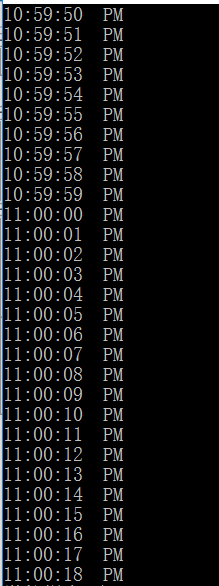
**2.2**



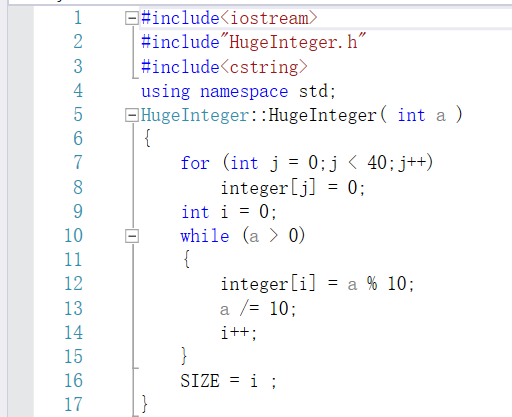


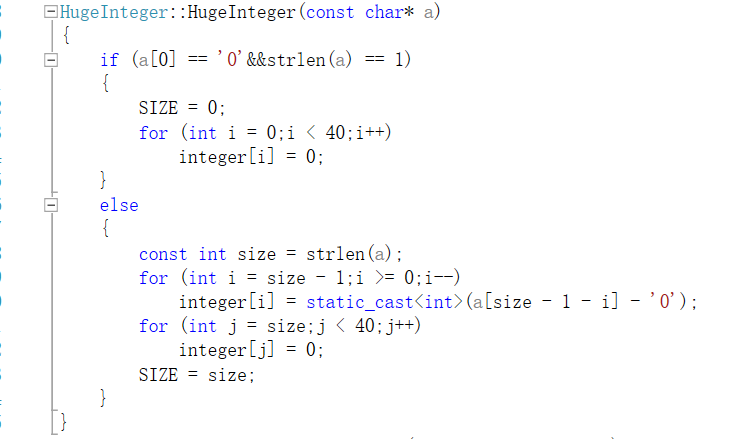


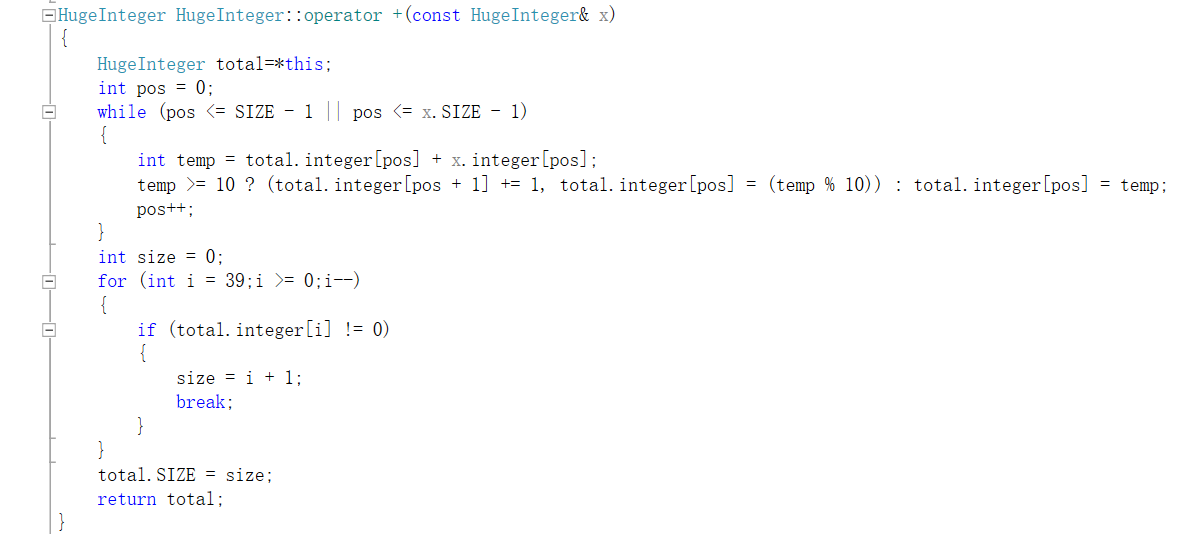


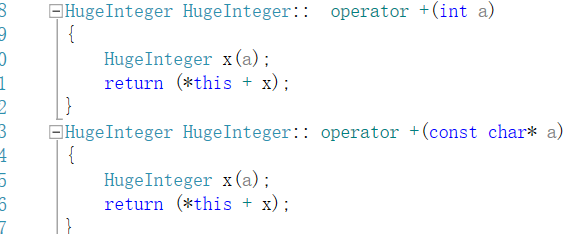


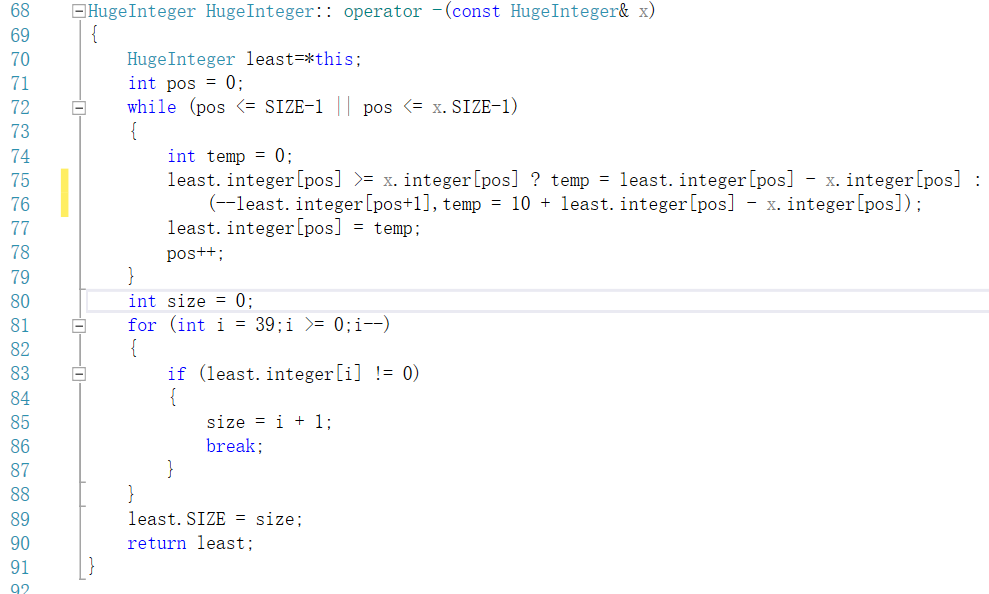
**Exp3:**

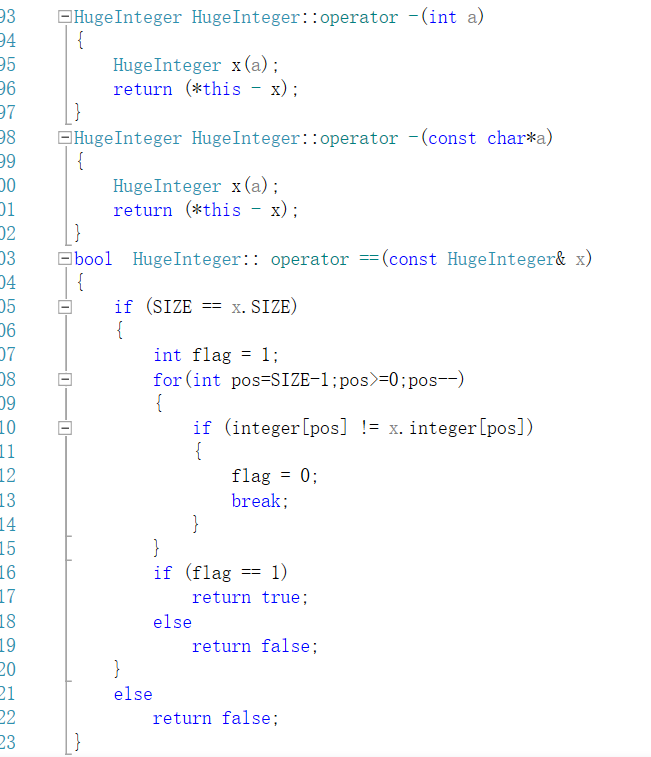
 

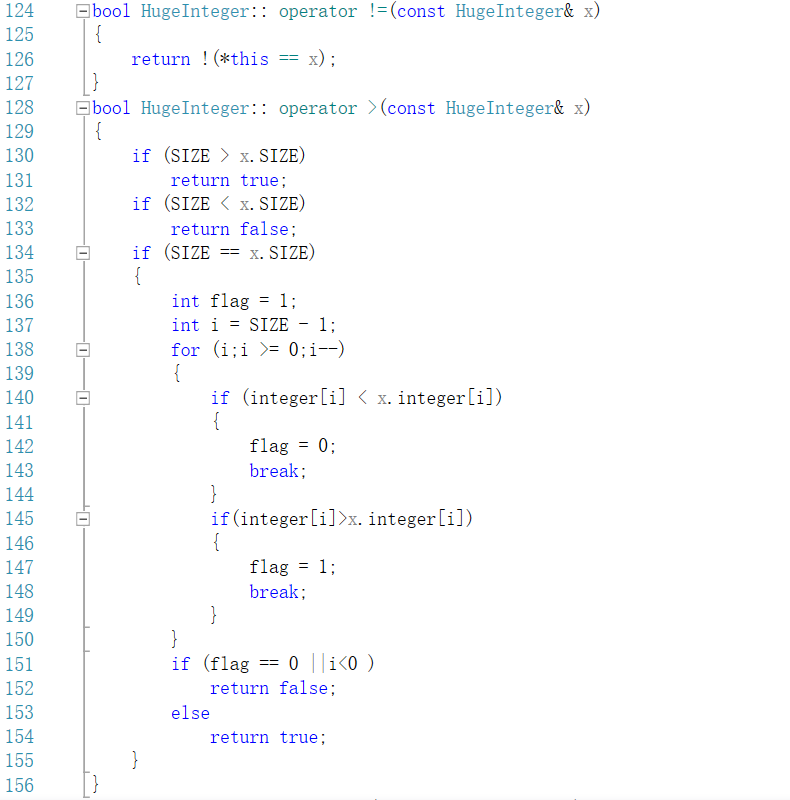


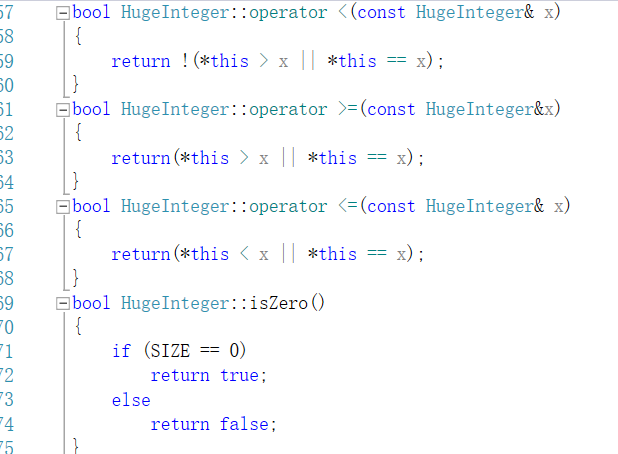


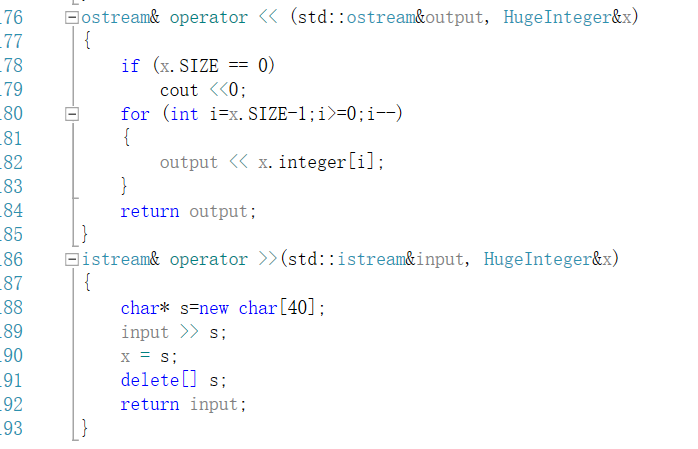


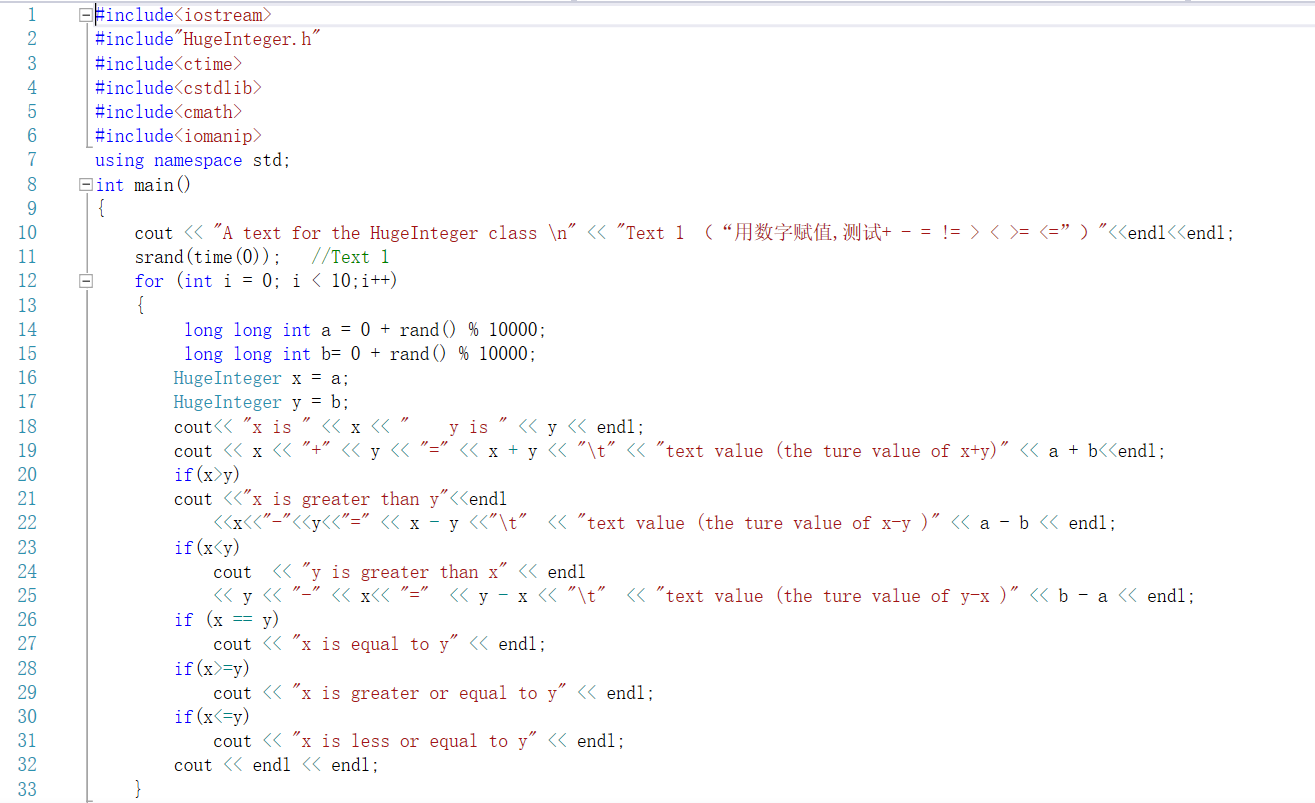


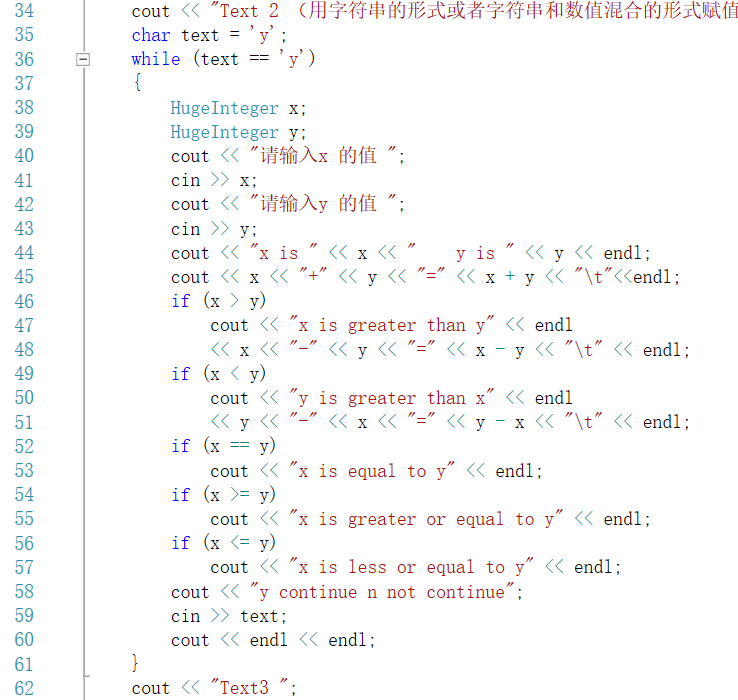


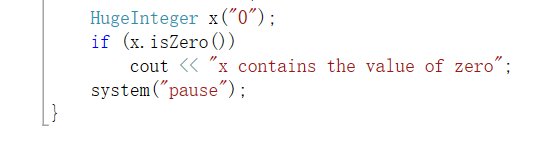


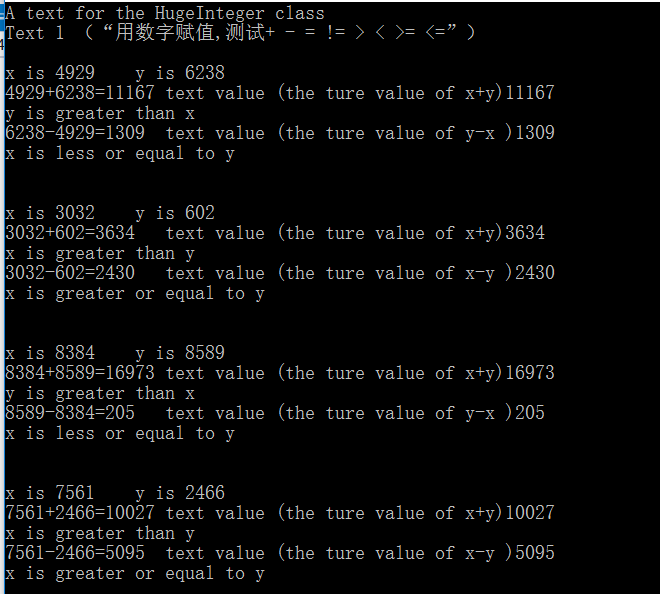


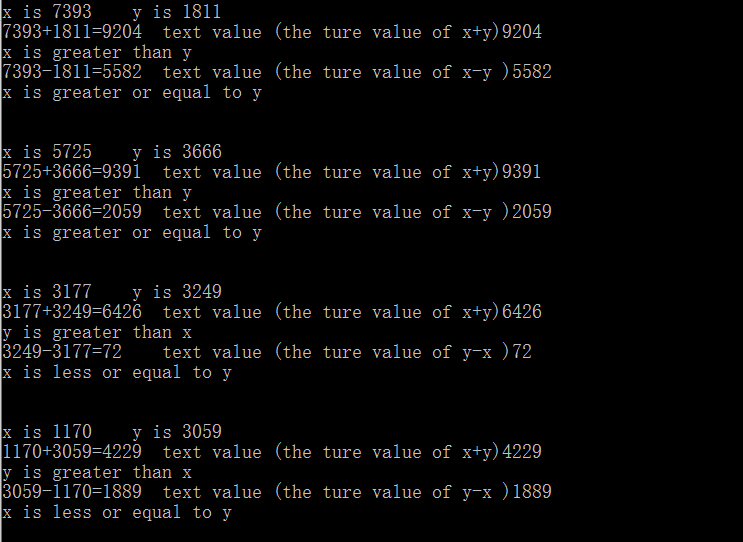


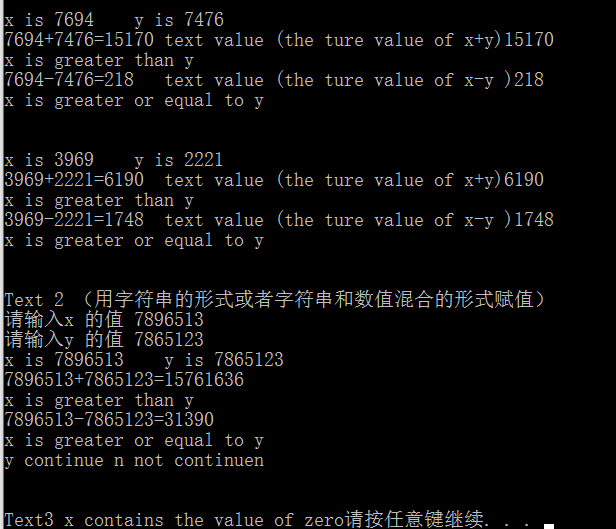












### 遇到的问题及解决过程

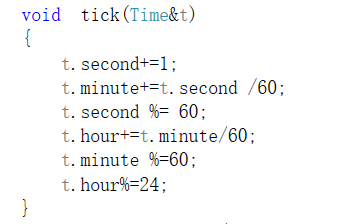
**Exp1:**nothing

**Exp2:**这里的问题是当我把tick函数从成员函数变成友元函数，在函数定义的时候使用一下语句导致编译器报错。后来经过排查翻看笔记，知道了友元函数并不像成员函数那样属于类，所以在定义的时候不需要加上 Time::

**Exp3:**这个实验遇到的问题主要是在算法的设计上，我才用的是用数位进行运算，但是由于忽略了很多特殊的情况，导致运算的结果出现错误。这些错误都来源与没有考虑周密，或者自己的想法是有问题的。没有出现语法上的错误。

### 实验体会

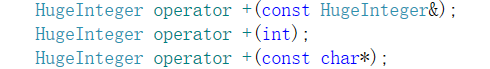
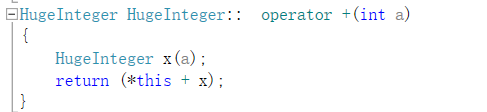
**Exp1:** 第一个实验，我在上学期的基础上，增加了运算符重载，使得在主程序中，类实例之间的计算变得十分简便。虽然在定义类，以及定义类函数的过程比较繁杂，但是如果能够构造一个成熟的类，那么使用它就会带来极大的方便。但同时我也发现:要定义成熟的类需要考虑到许多特殊情况，换句话说可能一个运算符需要重载很多次，还要考虑范围的问题，等等…

**Exp2：** 在这个实验中我第一次使用了 ？ ： 运算符，发现这个运算符可以代替if 语句，在大部分的情况下可以让程序更加简洁（当然如果再加上嵌套的话可能会影响可读性）。同时我认为这个实验的核心在于我按照我们人算时间进位的方法来设计这个函数，这也启发我们很多时候可以用我们人的思维来设计算法。

**Exp3:** 1和实验2一样算法的原理来自于我们进行运算时候的进位，与我们人在对数进行笔算一致，比较大小的方法逻辑也与我们人相同（比如从最高位开始比较）我认为这说明了算法可以来自于人解决问题的方法。

2调试的问题，一旦我们要写一个规模相对比较大的类的时候，如果我们全部写完再对类进行测试，那么出现错误就比较难找到。因为可能同时涉及到多个函数，这就给调试带来了不便降低了调试的效率。因此我认为最好每一个函数写完都对这个函数的功能进行测试，可以提高调试的总效率。此外我们再设计算法的时候可以无法一次性想得十分全面，通过一个一个测试，我认为有可能完善算法。

3随机数在测试中的应用：我写完之后采用随机数进行测试，发现了很多问题。随机数对于测试函数确实能起到很大的帮助。

4我们在进行运算符重载的时候，可能要涉及不同类型之间的运算，比如这组函数对加法运算符的重载。我才用的是将后面两种运算转化成第一种运算，这样能提高变成效率，但是弊端就是多调用了一次函数这样可能会对程序的效率产生影响。需要我们权衡与选择。

5这个实验并没有考虑到数的范围的问题，如果要构造完善的类，还需要考虑更多的特殊的情况。