**实验一：类和对象（A Deeper Look）**

**实验目的：1. 使用类的预处理包装命令**

**2. 定义类的构造函数和析构函数**

**3. 通过句柄访问类和对象**

**实验作业**

**作业一(习题9.5复数类)**

(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 

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:

1. Adding two Complex numbers: The real parts are added together and the imaginary parts are added together.
2. 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.
3. Printing Complex numbers in the form (a, b), where a is the real part and b is the imaginary part.

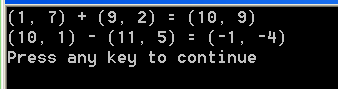
**要求：**

1. 在定义add和subtract函数时需要避免潜在的对被操作对象进行修改的可能
2. 要尽量体现软件工程学的思想

Complex add( const Complex & ); // function add

Complex subtract( const Complex & ); // function subtract

**结果输出示例：**



**作业二(习题9.6，分数类)**

(Rational Class) Create a class called Rational for performing arithmetic with fractions. Write a program to test your class.

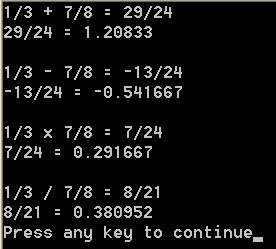
Use integer variables to represent the private data of the class the numerator and the denominator. Provide a constructor that enables an object of this class to be initialized when it is declared. The constructor should contain default values in case no initializers are provided and should store the fraction in reduced form. For example, the fraction



would be stored in the object as 1 in the numerator and 2 in the denominator. Provide public member functions that perform each of the following tasks:

1. Adding two Rational numbers. The result should be stored in reduced form.
2. Subtracting two Rational numbers. The result should be stored in reduced form.
3. Multiplying two Rational numbers. The result should be stored in reduced form.
4. Dividing two Rational numbers. The result should be stored in reduced form（化简后的分数格式）.——思考最优解法
5. Printing Rational numbers in the form a/b, where a is the numerator and b is the denominator.
6. Printing Rational numbers in floating-point format.

**结果输出示例：**



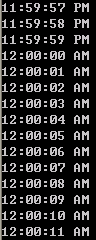
作业三（习题9.7，增强的Time类）

1. **Description of the Problem**

(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 that prints 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 00:00:00 AM).

**2. 结果示例**



**作业四（9.14，大数类）**

1. **问题描述**

Give a class *HugeInt* definition. This class uses a 10 element array of digits to store integers as large as 10 digits each. Provide member function:

(a) Constructor, destructor.

(b) *input, output* and *add*., using comparing this class’s objects.

(c) *EqualTo* and *isGreaterThan* that simply return true or false if the relationship holds between the two this classs objects.

Write a program to test all interface of this class.

1. **HugeInt类定义如下：**

#ifndef HUGEINTEGER\_H

#define HUGEINTEGER\_H

class HugeInteger

{

public:

HugeInteger( long = 0 ); // conversion/default constructor

HugeInteger( const char \* );

// addition operator; HugeInteger + HugeInteger

HugeInteger add( const HugeInteger & );

// addition operator; HugeInteger + int

HugeInteger add( int );

// addition operator;

// HugeInteger + string that represents large integer value

HugeInteger add( const char \* );

bool isEqualTo( HugeInteger & ); // is equal to

bool isGreaterThan(HugeInteger & ); // greater than

void input( const char \* ); // input

void output(); // output

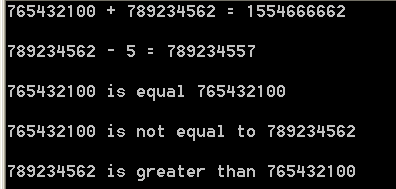
private:

short integer[ 40 ]; // 40 element array

}; // end class HugeInteger

**#endif**

1. **结果示例**

****

提示：如何处理进位与退位？