实验七：类和对象的进一步讨论

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

实验作业

作业一（习题 9.5，复数类）

1. 问题描述

(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

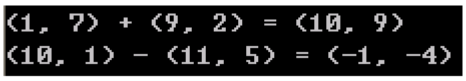
realPart + imaginaryPart * *i*

where *i* is $\sqrt{-1}$

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

2. 结果示例



```
(1, 7) + (9, 2) = (10, 9)
(10, 1) - (11, 5) = (-1, -4)
```

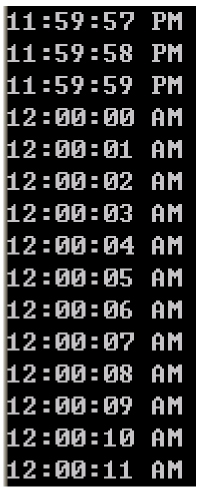
作业二（习题 9.7，增强的 Time 类）

1. 问题描述

(Enhancing Class Time) Modify the ***Time*** class of Figs. 9.4-9.5 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 12:00:00 AM).

2. 结果示例



```
11:59:57 PM
11:59:58 PM
11:59:59 PM
12:00:00 AM
12:00:01 AM
12:00:02 AM
12:00:03 AM
12:00:04 AM
12:00:05 AM
12:00:06 AM
12:00:07 AM
12:00:08 AM
12:00:09 AM
12:00:10 AM
12:00:11 AM
```

作业三（习题 9.14，大整数类）

1. 问题描述

(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***.

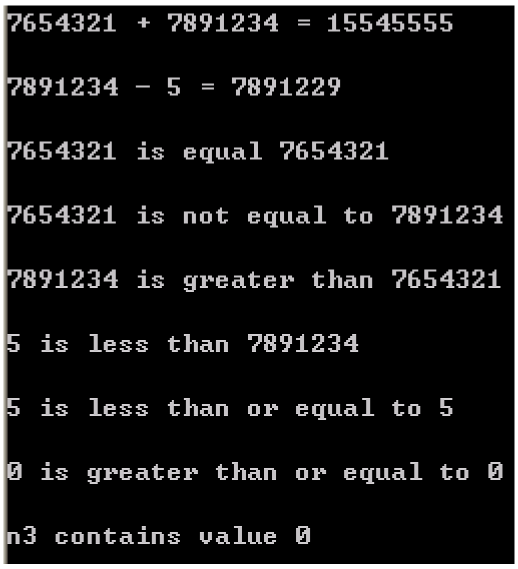
If you feel ambitious, provide member functions ***multiply***, ***divide*** and ***modulus***.

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

2. HugeInteger 类定义参考（仅供参考）

```
#ifndef HUGEINTEGER_H
#define HUGEINTEGER_H

class HugeInteger
{
public:

    HugeInteger( int = 0 ); // conversion/default constructor
    HugeInteger( const char * ); // conversion constructor

    // 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 * );

    // subtraction operator; HugeInteger - HugeInteger
    HugeInteger subtract( const HugeInteger & );

    // subtraction operator; HugeInteger - int
    HugeInteger subtract( int );

    // subtraction operator;
    // HugeInteger - string that represents large integer value
    HugeInteger subtract( const char * );

    bool isEqualTo( HugeInteger & ); // is equal to
    bool isNotEqualTo( HugeInteger & ); // not equal to
    bool isGreaterThan(HugeInteger & ); // greater than
    bool isLessThan( HugeInteger & ); // less than
    bool isGreaterThanOrEqualTo( HugeInteger & ); // greater than
                                                   // or equal to
    bool isLessThanOrEqualTo( HugeInteger & ); // less than or equal
    bool isZero(); // is zero
    void input( const char * ); // input
    void output(); // output

private:
    int integer[ 40 ]; // 40 element array
}; // end class HugeInteger

#endif
```

3. 结果示例



```
7654321 + 7891234 = 15545555

7891234 - 5 = 7891229

7654321 is equal 7654321

7654321 is not equal to 7891234

7891234 is greater than 7654321

5 is less than 7891234

5 is less than or equal to 5

0 is greater than or equal to 0

n3 contains value 0
```

作业四（习题 9.19，Date 类）

1. 问题描述

(英文) Modify class Date in Fig. 9.17 to have the following capabilities:

a. Output the date in multiple formats such as
   DDD YYYY
   MM/DD/YY
   June 14, 1992

b. Use overloaded constructors to create Date objects initialized with dates of the formats in part (a).

c. Create a Date constructor that reads the system date using the standard library functions of the <ctime> header and sets the Date members.

(中文) 用下列条件生成 Date 类：

a. 用多种格式输出日期，例如：
   DDD YYYY（YYYY 年的第 DDD 天）
   MM/DD/YY

June 14, 1992

b. 用重载的构造函数生成 Date 对象，用 a)中的日期格式初始化。
c. 生成一个 Date 构造函数，用<ctime>头文件的标准库函数读取系统日期和设置 Date 成员。

2. 实验提示

1）类定义示例：

```
#ifndef DATE_H
#define DATE_H

#include <string>
using std::string;

class Date
{
public:
    Date(); // default constructor uses <ctime> functions to set date
    Date( int, int ); // constructor using ddd yyyy format
    Date( int, int, int ); // constructor using dd/mm/yy format
    Date( string, int, int ); // constructor using Month dd, yyyy format
    void setDay( int ); // set the day
    void setMonth( int ); // set the month
    void print() const; // print date in month/day/year format
    void printDDDYYYY() const; // print date in ddd yyyy format
    void printMMDDYY() const; // print date in mm/dd/yy format
    void printMonthDDYYYY() const; // print date in Month dd, yyyy format
    ~Date(); // provided to confirm destruction order
private:
    int month; // 1-12 (January-December)
    int day; // 1-31 based on month
    int year; // any year

    // utility functions
    int checkDay( int ) const; // check if day is proper for month and year
    int daysInMonth( int ) const; // returns number of days in given month
    bool isLeapYear() const; // indicates whether date is in a leap year
    int convertDDToDDD() const; // get 3-digit day based on month and day
    void setMMDDFromDDD( int ); // set month and day based on 3-digit day
    string convertMMToMonth( int ) const; // convert mm to month name
    void setMMFromMonth( string ); // convert month name to mm
    int convertYYYYToYY() const; // get 2-digit year based on 4-digit year
    void setYYYYFromYY( int ); // set year based on 2-digit year
}; // end class Date
```

```
#endif
```

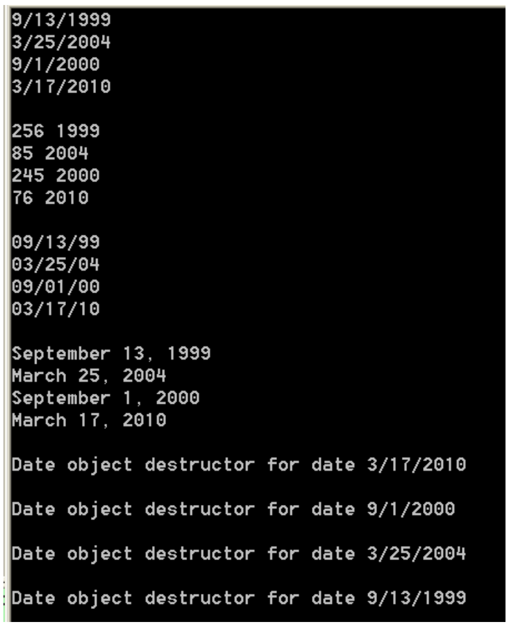
2） 使用<ctime>头文件读取系统日期示例（要求自学该部分内容）：

```
#include <ctime>

// default constructor that sets date using <ctime> functions
Date::Date()
{
    // pointer of type struct tm which holds calendar time components
    struct tm *ptr;

    time_t t = time( 0 ); // determine current calendar time

    // convert current calendar time pointed to by t into
    // broken down time and assign it to ptr
    ptr = localtime( &t );

    day = ptr->tm_mday; // broken down day of month
    month = 1 + ptr->tm_mon; // broken down month since January
    year = ptr->tm_year + 1900; // broken down year since 1900
} // end Date constructor
```

3）测试函数

```
int main()
{
    Date date1( 256, 1999 ); // initialize using ddd yyyy format
    Date date2( 3, 25, 04 ); // initialize using mm/dd/yy format
    Date date3( "September", 1, 2000 ); // "month" dd, yyyy format
    Date date4; // initialize to current date with default constructor

    // print Date objects in default format
    date1.print();
    date2.print();
    date3.print();
    date4.print();
    cout << '\n';

    // print Date objects in 'ddd yyyy' format
    date1.printDDDYYYY();
    date2.printDDDYYYY();
    date3.printDDDYYYY();
    date4.printDDDYYYY();
    cout << '\n';
    // print Date objects in 'mm/dd/yy' format
    date1.printMMDDYY();
    date2.printMMDDYY();
    date3.printMMDDYY();
    date4.printMMDDYY();
    cout << '\n';

    // print Date objects in '"month" d, yyyy' format
    date1.printMonthDDYYYY();
    date2.printMonthDDYYYY();
    date3.printMonthDDYYYY();
    date4.printMonthDDYYYY();
    cout << endl;

    return 0;
} // end main
```

3. 结果示例



```
9/13/1999
3/25/2004
9/1/2000
3/17/2010

256 1999
85 2004
245 2000
76 2010

09/13/99
03/25/04
09/01/00
03/17/10

September 13, 1999
March 25, 2004
September 1, 2000
March 17, 2010

Date object destructor for date 3/17/2010

Date object destructor for date 9/1/2000

Date object destructor for date 3/25/2004

Date object destructor for date 9/13/1999
```

作业五（习题 9.20，SavingAccount 类）

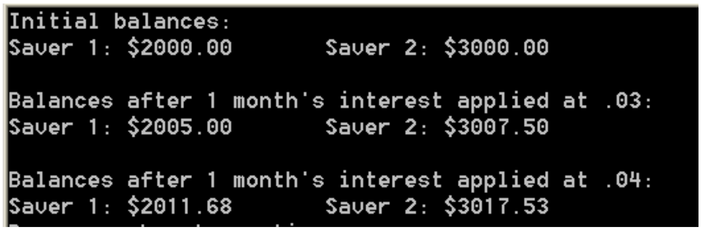
1. 问题描述

(英文) Create a *SavingsAccount* class. Use a static data member *annualInterestRate* to store the annual interest rate for each of the savers. Each member of the class contains a private data member *savingsBalance* indicating the amount the saver currently has on deposit. Provide member function *calculateMonthlyInterest* that calculates the monthly interest by multiplying the balance by *annualInterestRate* divided by 12; this interest should be added to *savingsBalance*. Provide a static member function *modifyInterestRate* that sets the static *annualInterestRate* to a new value. Write a driver program to test class *SavingsAccount*. Instantiate two different objects of class SavingsAccount, *saver1* and *saver2*, with balances of $2000.00 and $3000.00, respectively. Set the *annualInterestRate* to 3 percent. Then calculate the monthly interest and print the new balances for each of the savers. Then set the *annualInterestRate* to 4 percent, calculate the next month's interest and print the new balances for each of the savers.

(中文)

生成一个 SavingsAccount 类。用 static 数据成员包含每个存款人的 annualInterestRate（年利率）。类的每个成员包含一个 private 数据成员 savingsBalance，表示当前存款额。提供一个 calculateMonthlyInterest 成员函数，计算月利息，用 balance 乘以 annualInterestRate 除以 12 取得，并将这个月息加进 savingsBalance 中。提供一个 static 成员函数 modifyInterestRate，将 static annualInterestRate 设置为新值。实例化两个不同的 SavingsAccount 对象 saver1 和 saver2，结余分别为 2000.00 和 3000.00。将 annualInterestRate 设置为 3%，计算每个存款人的月息并打印新的结果。然后将 annualInterestRate 设置为 4%，再次计算每个存款人的月息并打印新的结果。

2. 结果输出示例



```
Initial balances:
Saver 1: $2000.00        Saver 2: $3000.00

Balances after 1 month's interest applied at .03:
Saver 1: $2005.00        Saver 2: $3007.50

Balances after 1 month's interest applied at .04:
Saver 1: $2011.68        Saver 2: $3017.53
```

作业六（习题 9.21，IntegerSet 类）

1. 问题描述

(英文) Create class *IntegerSet* for which each object can hold integers in the range 0 through 100. A set is represented internally as an array of ones and zeros. Array element a[i] is 1 if integer i is in the set. Array element a[j] is 0 if integer j is not in the set. The default constructor initializes a set to the so-called "empty set," i.e., a set whose array representation contains all zeros.

- Provide member functions for the common set operations. For example, provide a *unionOfSets* member function that creates a third set that is the set-theoretic union of two existing sets (i.e., an element of the third set’s array is set to 1 if that element is 1 in either or both of the existing sets, and an element of the third set’s array is set to 0 if that element is 0 in each of the existing sets).
- Provide an *intersectionOfSets* member function which creates a third set which is the set-theoretic intersection of two existing sets (i.e., an element of the third set’s array is set to 0 if that element is 0 in either or both of the existing sets, and an element of the third set’s array is set to 1 if that element is 1 in each of the existing sets).
- Provide an *insertElement* member function that inserts a new integer k into a set (by setting a[k] to 1). Provide a deleteElement member function that deletes integer m (by setting a[m] to 0).
- Provide a *printSet* member function that prints a set as a list of numbers separated by spaces. Print only those elements that are present in the set (i.e., their position in the array has a value of 1). Print --- for an empty set.
- Provide an *isEqualTo* member function that determines whether two sets are equal.
- Provide an additional *constructor* that receives an array of integers and the size of that array and uses the array to initialize a set object.

Now write a driver program to test your IntegerSet class. Instantiate several IntegerSet objects. Test that all your member functions work properly.

(中文)

生成一个 IntegerSet 类。IntegerSet 类的每个对象可以保存 0 到 100 之间的整数值。一个集合内部表示为 0 和 1 的数组。数组元素 a[i]为 1 表示整数 i 在集合中，数组元素 a[j]为 0 表示整数 j 不在集合中。默认构造函数将集合初始化为“空集”，即所有元素都是 0。

提供常用集合操作的成员函数。例如，提供一个 unionOfIntegerSets 成员函数，生成两个现有集合的并集（即只要其中一个集合的元素为 1，则并集的元素就是 1，如果两个集合的元素均为 0，则并集的元素就是 0）。

提供intersectionOfIntegerSets成员函数生成两个现有集合的交集（即只要其中一个集合的元素为0，交集的元素就是0，如果两个集合的元素均为1，则交集的元素就是1）。

提供一个insertElement成员函数，在集合中插入新整数k（将a[k]设置为1）。提供deleteElement删除整数m（将a[m]设置为0）。

提供一个setPrint成员函数，将集合表示的值打印为以空格分隔的列表。只打印集合中存在的元素（即对应值为1的位置），空集打印---。

提供一个isEqualTo成员函数，确定两个集合是否相等。

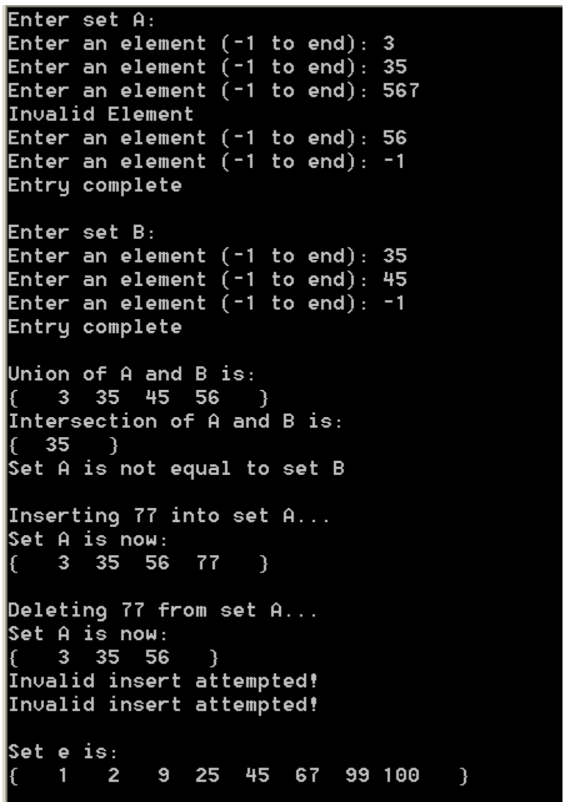
提供其他构造函数，取五个整数参数，可以初始化一组对象。如果提供的元素不到五个，其他元素用默认参数-1。

编写一个驱动程序，测试IntegerSet类。实例化几个IntegerSet对象。测试所有成员函数能否正确工作。

2．类定义参考

```
#ifndef INTEGER_SET_H
#define INTEGER_SET_H

class IntegerSet
{
public:
    // default constructor
    IntegerSet()
    {
        emptySet(); // set all elements of set to 0
    } // end IntegerSet constructor

    IntegerSet( int [], int ); // constructor that takes an initial set
    IntegerSet unionOfSets( const IntegerSet& );
    IntegerSet intersectionOfSets( const IntegerSet& );
    void emptySet(); // set all elements of set to 0
    void inputSet(); // read values from user
    void insertElement( int );
    void deleteElement( int );
    void printSet() const;
    bool isEqualTo( const IntegerSet& ) const;
private:
    int set[ 101 ]; // range of 0 - 100

    // determines a valid entry to the set
    int validEntry( int x ) const
    {
        return ( x >= 0 && x <= 100 );
    } // end function validEntry
}; // end class IntegerSet

#endif
```

3．结果输出示例



```
Enter set A:
Enter an element (-1 to end): 3
Enter an element (-1 to end): 35
Enter an element (-1 to end): 567
Invalid Element
Enter an element (-1 to end): 56
Enter an element (-1 to end): -1
Entry complete

Enter set B:
Enter an element (-1 to end): 35
Enter an element (-1 to end): 45
Enter an element (-1 to end): -1
Entry complete

Union of A and B is:
{   3  35  45  56   }
Intersection of A and B is:
{  35   }
Set A is not equal to set B

Inserting 77 into set A...
Set A is now:
{   3  35  56  77   }

Deleting 77 from set A...
Set A is now:
{   3  35  56   }
Invalid insert attempted!
Invalid insert attempted!

Set e is:
{   1   2   9  25  45  67  99 100   }
```

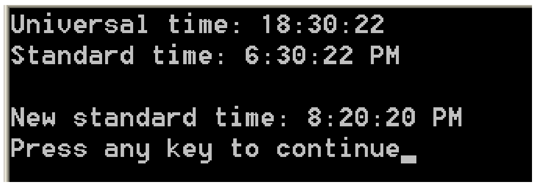
作业七（习题 9.22，Time 类）

1．问题描述

It would be perfectly reasonable for the Time class of Figs.9.4-9.5 to represent the time internally as the number of seconds since midnight rather than the three integer values hour, minute and second. Clients could use the same public methods and get the same results. Modify the Time class of Fig.9.4 to implement the time as the number of seconds since midnight and show that there is no visible change in functionality to the clients of the class. [Note: This exercise nicely demonstrates the virtues of implementation hiding.]

注：将原来的 hour，minute 和 second 三个数据成员用 totalSeconds 一个替换。类用户接口完全不变。

2. 结果输出示例



```
Universal time: 18:30:22
Standard time: 6:30:22 PM

New standard time: 8:20:20 PM
Press any key to continue
```