学　　号：	0121910870616

武漢理工大學

《面向对象程序设计 C》
课程大作业

学　　院：计算机科学与技术学院

专业班级：计算机类 y1906

姓　　名：邓晟淼

指导教师：耿枫

2020 年 6 月

选题说明	**题目： 设计一个人工设置的双计时器（高考倒计时器和备考时间累积器），支持起始时间的人工设定和人工加、减时间操作。**

作业内容

一、基本要求

1. 设计一个日期类 date。类体内需包含描述年、月、日等信息的数据成员，以及用于设置与读取这些数据成员的成员函数。

2. 在类体内定义用于初始化对象的构造函数，包含一个重载默认值方式。

3. 在 date 类中定义成员函数，用于处理月、日的进位的改变问题，特别注意不同月份天数的问题，判断闰年问题。

4. 在 date 类中重载 + 或 -，++ 或 -- 运算符重载，用于实现对日期对象进行加或减 n 天，加或减 1 天操作。

5. 设计一个双计时器界面，提供人工设定起始日期，人工加、减日期操作，测试 date 类中各成员函数是否能正确运行，并给出测试结果的运行截图。

6. 对本大作业进行总结（存在的不足、问题、经验等）。

二、提高要求

1. 通过继承方式设计出时间类（类名为 DateTime ）（包含年、月、日、时、分、秒），给出类 DateTime 的定义和实现，描述设计思路。

2. 给出设完整的日历表类（类名为 CDate）设计，描述设计思路。

三、其他要求

1. 运用课堂所学的知识，按题目的要求独立完成大作业及报告。

2. 被认定为抄袭或被抄袭者，本课程最终成绩作不及格处理。

3. 在网络教学平台上提交：**（最后截止日期为：2020 年 7 月 12 日）**

（1）大作业报告（word 版和 pdf 版各一份）

（2）源代码（仅提交 .h 和 .cpp 文件）

四、评分标准

基本 1	**基本 2**	**基本 3**	**基本 4**	**基本 5**	**基本 6**	**提高 1**	**提高 2**	**报告**	**延期提交**
10 分	5 分	15 分	20 分	15 分	5 分	5 分	5 分	20 分	扣 5 分

成绩评定

	基本 1	基本 2	基本 3	基本 4	基本 5	基本 6	提高 1	提高 2	报告	延期提交	合计
标准分数	10 分	5 分	15 分	20 分	15 分	5 分	5 分	5 分	20 分	扣 5 分	100 分
实际完成情况											

教师签名：

2020 年　　月　　日

报告正文

（一）要求的实现及演示：

①基本要求 1：设计一个日期类 date。类体内需包含描述年、月、日等信息的数据成员，以及用于设置与读取这些数据成员的成员函数。

在源码中，我定义了 date 类，其中包括了年，月，日的数据成员，并且定义了构造函数和复制构造函数，用以实现之后所需要的各种功能。除此之外，我定义了 set 函数用以设置数据成员，定义了 get 函数用以返回数据成员的值。在这些函数中，我检测了设置的参数的正确性，例如，年份必须大于 0，月份必须在 1 至 12 之间，日在调用了判断平闰年和月份的函数之后，必须不能小于 1，大于这个月日期的最大值。（当然，在之后使用派生类 DateTime 的时候，因为需要用一个日期加减另一个没有年月日的日期，因此当年月日输入均为-1 的时候，将会将 date 中的年月日置 0）。date 类的成员函数以及数据成员如下图（图 1-1）所示：（关于这些成员函数是否能够成功运行，我将在下方的倒计时和累计时的实现中进行演示。）

```
class date {
public:
    date(int y = 1, int m = 1, int d = 1);
    date(date& dat);
    void setYear(int y);
    void setMonth(int m);
    void setDay(int d);
    int getYear() { return year; }
    int getMonth() { return month; }
    int getDay() { return day; }
    void showDate();
    date normal(int oper,int days);
    date operator + (int days);
    date operator - (int days);
    date operator ++ ();
    date operator -- ();
    date operator ++ (int);
    date operator -- (int);
    int operator - (date& b);
protected:
    int  year, month, day;
};
```

（图 1-1）

②基本要求 2：在类体内定义用于初始化对象的构造函数，包含一个重载默认值方式。

如第一个基本要求中的方法。我设置了初始化对象的构造函数，并且设计了一个复制构造函数。其中初始化对象的构造函数中，我包含了默认参数，若构造时输入的数据非法，将会将时间设置为 1 年 1 月 1 日。具体非法条件的判断，我使用了一个全局变量数组来表示每个月的天数，并且设计了一个函数 bool isLeapYear(int y);函数来进行判断年份是闰年还是平年，倘若是闰年，则将全局变量数组 CommonYear[1]设置为 29，此变量代表 2 月天数，倘若平年，则设置为 28。这样在知道年和月之后，就能马上得到这个月的具体天数，用以判断日期输入是否非法。有一个特例，因为我在后面的 DateTime 类设计时，在进行时间加减的过程中，加减的那个时间年月日必须都默认设置为 0，因此，我在 date 的构造函数中，设置了如果接收到的年月日参数均为-1，则将年月日都设置为 0。这样不会在派生类的实现中出现构造错误。

③基本要求 3：在 date 类中定义成员函数，用于处理月、日的进位的改变问题，特别注意不同月份天数的问题，判断闰年问题。

为了实现处理年月日的进位问题，我在 date 类中定义了成员函数 date normal(int oper,int days);这个成员函数可以在重载加减操作的时候提供进位的方法，进行进位操作。此时 oper 参数代表进行的是加法还是减法，如果是 1，则进行加法，如果是 0，则进行减法。days 则是需要进行加减的天数。

我想了一下，为了保证加减运算方便，我采取了一天一天加，或者一天一天减，这样会很便于代码的书写。具体的算法则是首先计算当前的年份是平年还是闰年，如果是平年，则将数组中表示 2 月的数赋值为 28，如果是闰年，则将数组中表示 2 月的数赋值为 29，这样在这一年的计算中就不会出现问题计算出错的地方。之后倘若出现加减导致年份变化，则再进行一次判断。对于加减的运算则首先判断这个月的总天数和当前天数，倘若处于临界值，如减法时的第一天，或加法时的最后一天，这样则对月份进行加减，若不是临界值，则不用对月份进行改变只需要直接加减日期天数。之后如果月份处于 1 月或者是 12 月，则对年进行加或减 1 年，对年份进行操作之后，需要再次判断平年闰年。具体的演示在倒计时，累计时中一并展示。

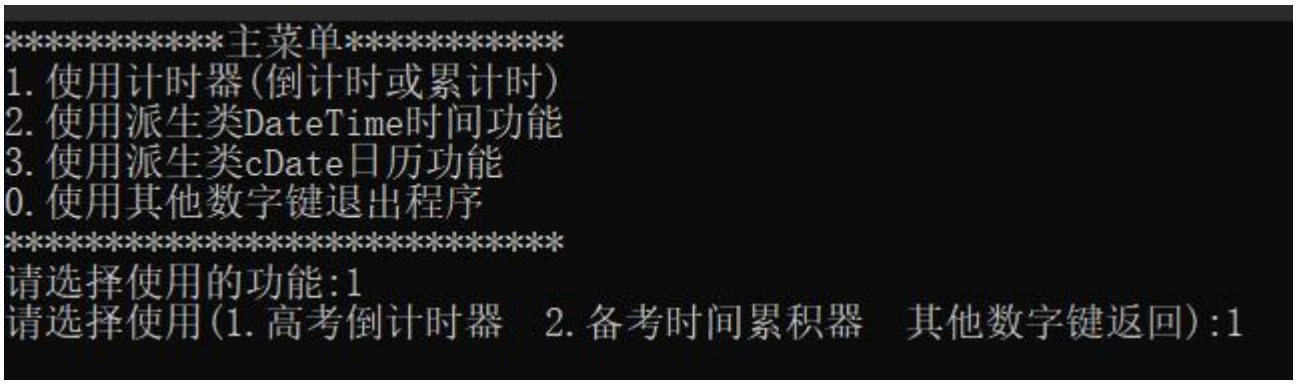
④基本要求 4：在 date 类中重载 + 或 -，++ 或 -- 运算符重载，用于实现对日期对象进行加或减 n 天，加或减 1 天操作。

为了完成加法与减法，自加自减的操作符重载，我在第三问之中写的 normal()函数则派上了用场，通过该函数，只需要提供加或减的符号，并且提供加或减的天数，就可以得到加减完成后的天数作为返回值，并且改变原来的数值。自加自减又分为前置与后置，我一一通过调用了+或者是-的重载函数进行了操作，能够准确完成操作。除此之外，我又重载了操作符-(减)，将其作为两个 date 日期的差值。这样在之后的函数操作中，能够更方便的判断差值。同样，具体的操作我在之后的倒计时，累计时中一一展示。

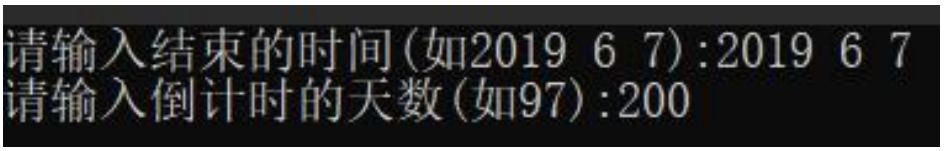
⑤基本要求 5：设计一个双计时器界面，提供人工设定起始日期，人工加、减日期操作，测试 date 类中各成员函数是否能正确运行，并给出测试结果的运行截图。

我在文件 calcugraph.cpp 和相应的.h 文件中写出了高考倒计时和备考时间累计器的相关操作，提供了各种选择功能。对之前的类 date 成员函数，功能进行了相应的测试，未出现 bug，并且我在所有输入的地方，基本上都设置了相应的判断语句，确保输入的数据正常，并且我在每次输入完成后都清除了缓冲区，使得程序不会出现输入非数字后卡死的情况。下面进行截图演示。

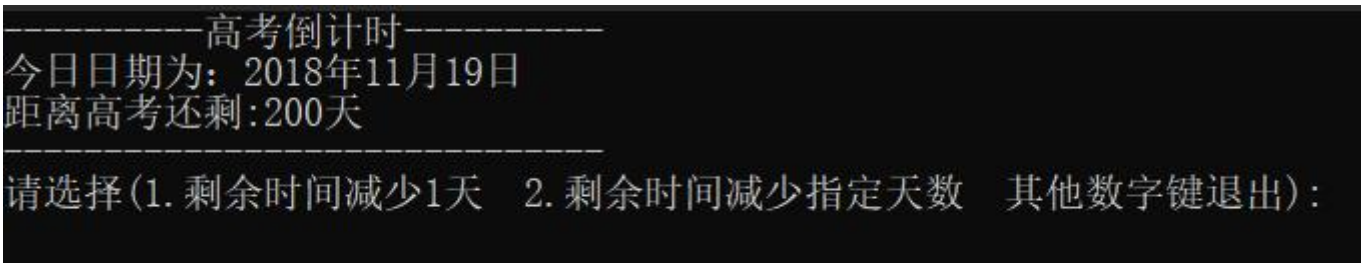
首先我们进行高考倒计时的演示，这个功能中，使用了构造函数、复制构造函数、输出 date 类的函数、以及+，-号重载的功能来进行实现。因为输入的是高考的时间，和倒计时的总天数，因此要确定天数从 xx 号开始，需要使用-的功能，以确定第一天的时间。之后的倒计时功能则是采用了+的重载，以实现天数的增加。如图，我设置时间 2019 年 6 月 7 日，倒计时时间 200 天。



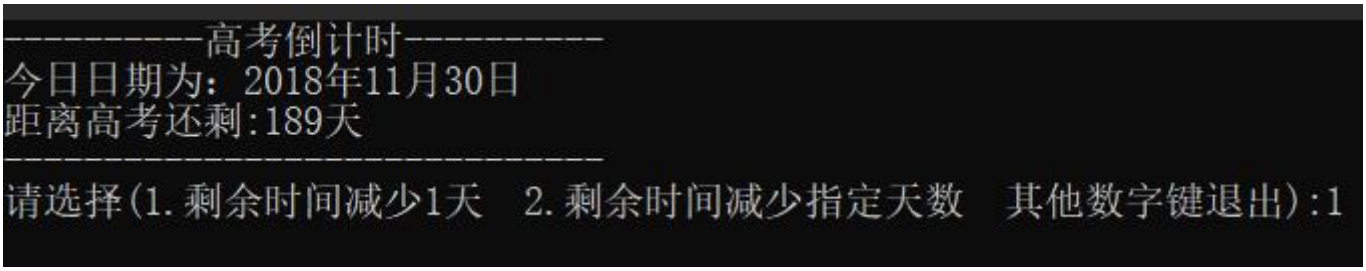
（图 5-1）选择功能



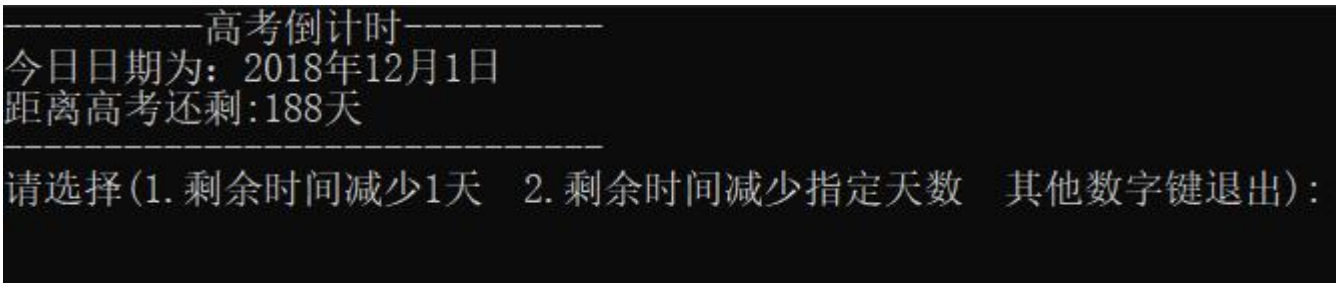
（图 5-2）输入数据



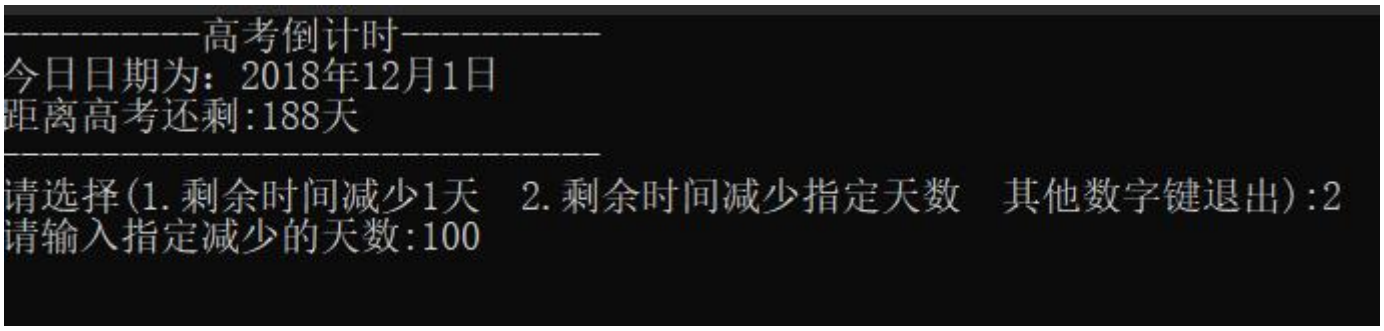
（图 5-3）确认后进入倒计时界面



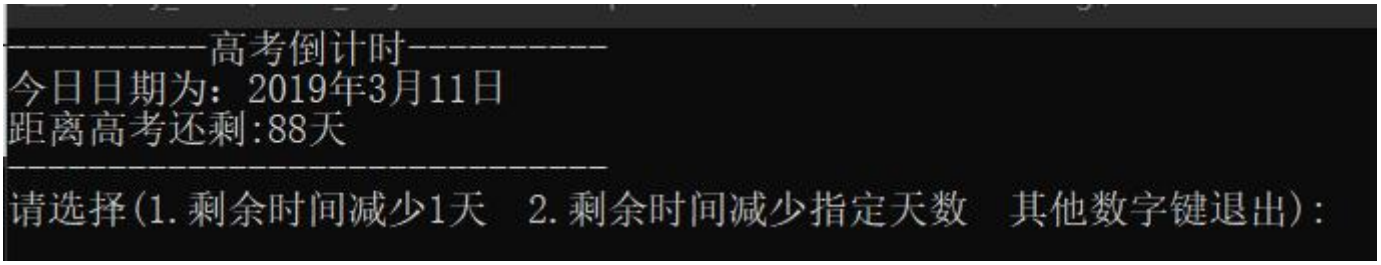
（图 5-4）为了展示进位，我手动使用减少一天功能调整至 11 月 30 日



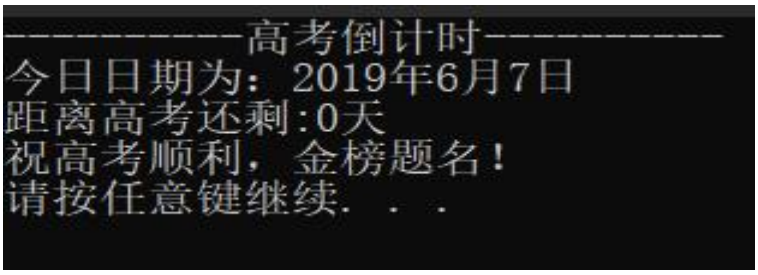
（图 5-5）可以看到确实减少了 1 天时间变成了 12 月 1 日



（图 5-6）演示直接减少 100 天

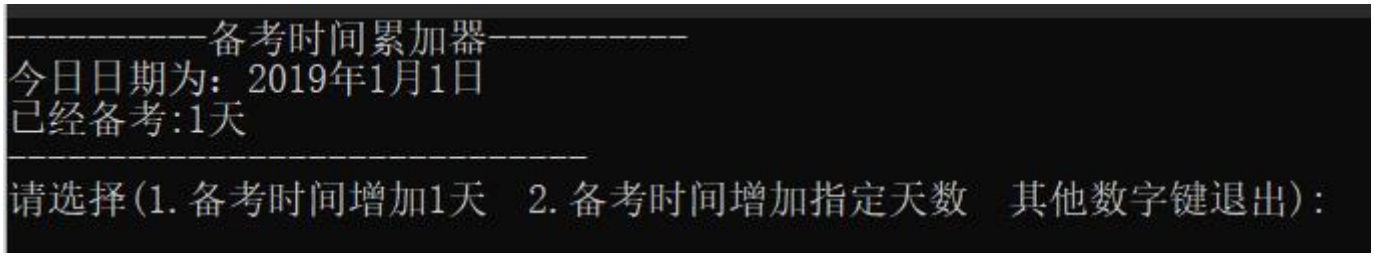


（图 5-7）确实减少了 100 天



（图 5-8）演示直接减少 88 天，无错误，直接到了设定日期

接下来继续演示累加器的功能，我的理解，这个功能是你输入开始的时间，帮你进行累加，看一共复习了多少天。我们这个功能就直接来作为测试看看加减准不准，我们为了测试平闰年等时间的变化，以 2019 年 1 月 1 日为起点，直接加上比较大的时间，与网页上的计算器进行比对。



（图 5-8）设定起点日期

（图 5-9）随意设置增加 1973 天，此数应该大于 4 年，有说服性

（图 5-10）此时日期为 2024.5.27

（图 5-11）网站上使用计算器，确实是 2024 年 5 月 27 日，应该不存在 bug 了

⑥基本要求 6：总结与自我反思

我在这个大作业中还是很费尽心思，希望能够将他做的尽善尽美，我先来说说我感觉存在的优点，再来说说我的反思与缺点。

首先我在设计这个程序的时候进行了不少构思，为了将这个程序做的比较美观，我在每次操作之后都会进行清屏，使得内容更加集中，有条理。此外也设计了一些分隔框之类的东西，将功能进行分割，这样会显得更加条理分明，更美观。

其次，因为之前有过程序输入的东西不正确因而进入死循环或者是代码运行错误的情况发生，这次我在所有的输入流之后都添加了代码清空缓冲区，所有的输入都会有判断，如果输入的数据不合规，那么程序要么会使用默认值，要么会提示你重新输入，比如说日期输入的是不是标准日期，比如出现2 月 30 日肯定就不符合规矩。比如倒计时的时候，减的时间比倒计时剩余的时间还要多，那么显然也是不正确的，或者是在这个时候的输入中使用了负数，程序也会进行报错，提示重新输入。我把可执行文件给了我的室友帮我当测试员，应该还没有发现在在输入上有 bug 的情况，当然，这也只是我们现在的水准，不一定真就能找不出 bug，这个肯定是有的，比如你输入-1 年-1 月-1 日就会设置成 0 年 0 月 0 日，这个是我为了后面的 datetime 类方便所写的，只能说我尽力把可能出现的 bug 修改了很多。

最后，就是我比较引以为傲的地方------DateTime 的实现，虽然他只有可怜的 5 分，并不会因为我多写了就拉开差距，但是我还是想讲，哈哈。我在写这个地方的时候，就想，这个应该只是需要检测我们写派生类，耿老师是想检测我们派生类的操作是否扎实。我问了好几个同学，他们都跟我说：我只写了日期的加减法，比如设置了一个 19 年 1 月 1 日 10 点 30 分 30 秒，可以调用之前重载的+-法操作符，进行加 10 天或者减 5 天的操作，这就足够了。但是我觉得，既然设计了这个类，肯定还是要发挥作用，应该要让人能够直接加减任何时间（以 23 时 47 分 50 秒这种时间为例，这样符合大多数人的习惯），我想除此之外，要是有人输的时间是溢出的，比如输入了个 100 小时，或者输入了 800 分钟，1000 秒，也应该能够进行计算，不然我认为就很反人类。所以我冥思苦想，最终想出了一个进位函数，这个函数通过一个控制符来进行控制，判断是否需要进位到天数上。也就是判断进位到小时，还是天，因为天的时间就不是单纯的加减了。通过这个标准化进位成员函数，直接把这个问题解决了，此时该类的加减重载不超过 10 行代码就完成了。具体实现，我在下面的 DateTime 类演示的时候来描述，哈哈。

下面说说这次大作业的缺点和反思吧，我在写这个的时候，一开始就没有把这个要求 3 看清楚，而是直接将进位方法写到了加减法的重载之中。后来写报告的时候才发现，赶忙进行重构。我想这要是在工作中，可能因为我的这个错误，就导致设计的程序没有达到测试或者要求方的满意，这是很要不得的，我盲目的书写也只是因为我的盲目自大，我以为把所有的实现方法全部都想清楚了，事实上，我根本没有，我只是自以为清楚了，但是写出来的东西，却和需要的东西不一样。这是我需要改掉的毛病，很自负，这是个不好的习惯。

我希望在之后学习计算机的道路上，我能保持现在这样的热情与兴趣，不断地学习，并且改掉程序中，包括自身的各种坏毛病，精益求精，写出更好的代码。

⑦提高要求 1：通过继承方式设计出时间类（类名为 DateTime ）（包含年、月、日、时、分、秒），给出类 DateTime 的定义和实现，描述设计思路。

首先，我定义了一个继承 date 的时间类 DateTime，这个类包含了以下的功能和参数，其作用则是进行计算包含了年月日时分秒之间的加减。类的具体成员函数及数据成员如下图所示（图 7-1）

```
#pragma once
#include"date.h"
class DateTime :public date {
public:
    DateTime(int y = 1, int m = 1, int d = 1, int h = 0, int min = 0, int sec = 0);
    DateTime(DateTime& dat);
    DateTime operator +(DateTime& dat);
    DateTime operator -(DateTime& dat);
    DateTime normalForm(int n);     //将this转换成标准形式
    void show();
private:
    int hour, minute, second;
};
```

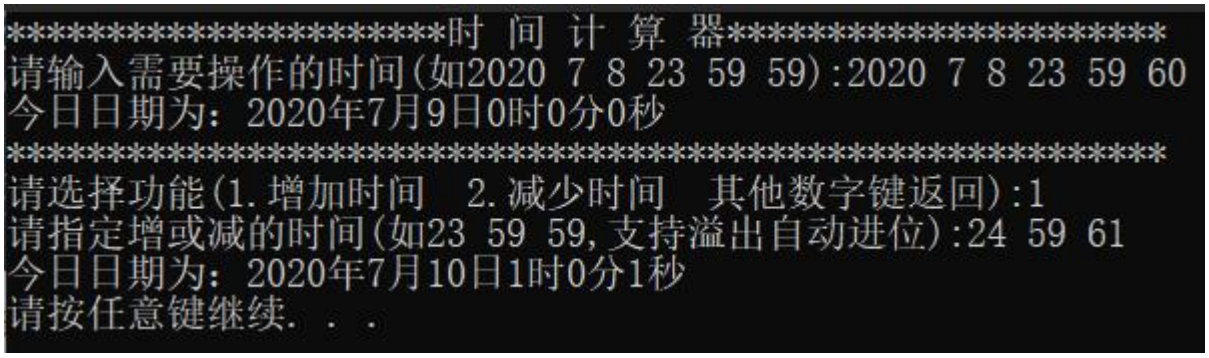
（图 7-1）类的具体成员函数及数据成员

成员函数包括了构造函数和复制构造函数，重载运算符+和-，以及转换标准函数 normalForm 通过这个函数，可以将时间进行转换，转换成标准的时间。我来详细说明一下这个运算的方法：首先这个标准转换函数有个参数，可以控制是否进位，如果进位，则是将后面大于 60s 的进位到分钟上，大于 60 分钟进位到小时上，大于 24 小时的调用 date 类中的成员函数+或者-对天数进行增加或减少。如果不进位，则是将秒和分钟的数字进位到小时上，允许 hour 大于 24 小时。这样做的原因则是因为时间和日期之间的操作不太相同，时间是单纯的相加减，而日期则必须要通过重载运算符来进行操作而不能简单的进行加减。这样的操作就很简单了，我主要讨论一下减法的操作步骤，因为减法相比于加法运算应该更为复杂，重载函数如下图（图 7-2）：

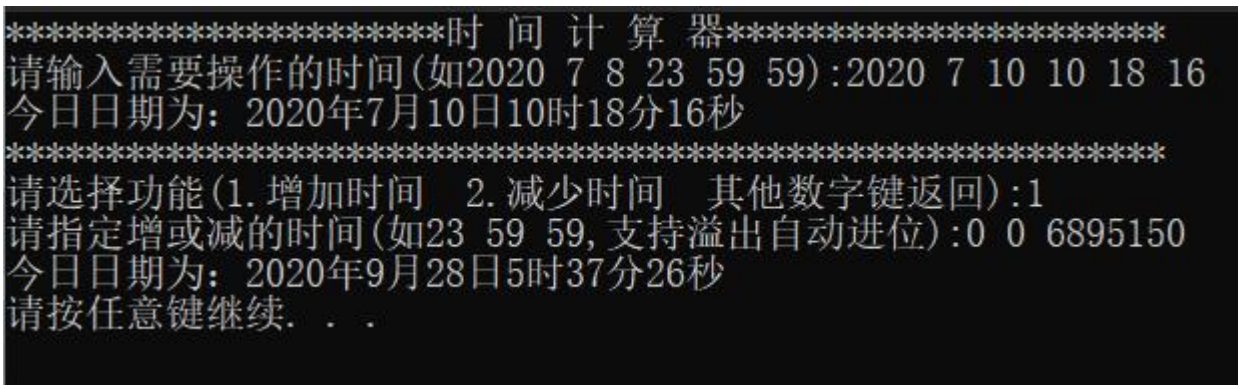
```
//[函数]    DateTime::重载运算符-
//[功能]    重载运算符-，用以实现计算两个DateTime对象数据之差
//[参数]    DateTime& dat:进行减法运算的对象
//[返回]    DateTime: 返回加法的结果
DateTime DateTime::operator -(DateTime& dat) {
    dat.normalForm(0);      //防止输入数据溢出，不进位
    this->hour -= dat.hour;
    this->minute -= dat.minute;
    this->second -= dat.second;
    this->normalForm(1);        //将this标准化，进位
    return *this;
}
```

（图 7-2）减法操作的运算符重载

减法运算一开始得到一个需要进行减法操作的日期及时间，作为被减数。此时调用一次进位的标准转换函数，将后面可能存在的溢出全部修正，满足分和秒小于 60，时小于 24 这样在之后的减法中，秒和分最多只需要前面的分和时给他补一位。之后再输入一个时间，仅包括时分秒，当然这个时间也允许溢出，比如输入 0 0 80000（即 80000 秒）或者输入 27 60 100（即 27 时 60 分 100 秒）之类的数字，然后调用不进位的标准转换函数，将其修改为正常的时间（时允许溢出）。之后直接将开始的时间减去要减去的时间，这样时分秒上可能会出现负数。但是由于之前都将分和秒的数据标准化，因此最多只会出现-59 秒，-59 分这样的数，只需要前一位借 1 位给他，因此只要是符号为负，则将前一位再减 1。之后的时，则是将计算前面的天需要借多少天给他，这样就能再调用 date 中的-，将日期减去相应天数，最后返回的结果，则是准确的时间。下面进行操作演示：

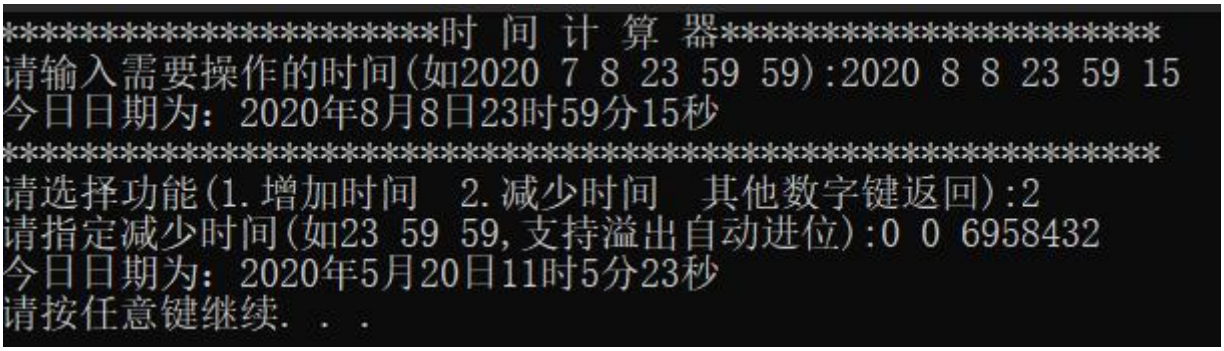


（图 7-3）输入时间溢出，会自动转换。增加 24 时 59 分 61 秒，即 25 时 0 分 1 秒，可以看到增加后，时间确实为 7 月 10 日 1 时 0 分 1 秒。



（图 7-4）随意测试溢出数据，因为在线网站不如我的代码，只能进行秒数溢出相加，所以进行如图演示。和网站时间一致

（图 7-5）和测试结果完全一致



（图 7-6）测试减法，也随便输入个秒数与网站对比吧

（图 7-7）完全一致，(●' ◡ '●)

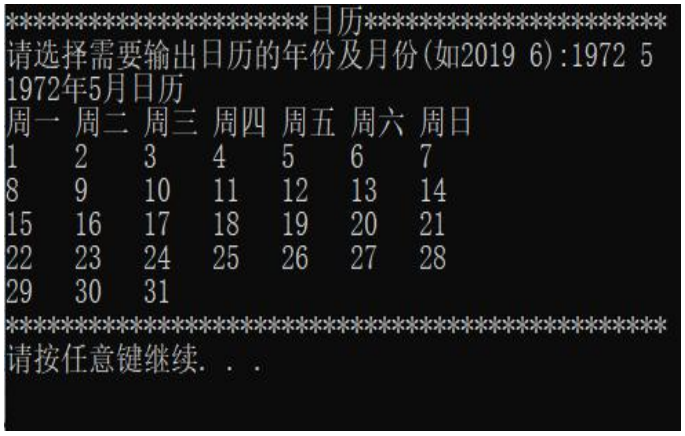
⑧提高要求 2：给出设完整的日历表类（类名为 CDate）设计，描述设计思路。

日历表类相对于之前的两个类的设计，就要简单多了。直接判断需要的年份时平年还是闰年。之后再利用之前的 date 类中的两个 date 对象相减得到天数的重载，确定和 1 年 1 月 1 日（星期一）的

时间差，判断这个月的第一天是星期几，在之后就能打印当前月的日历表了。（有一点很困惑，我之前运算的时候去查过 1 年 1 月 1 日是星期几，有人说星期一有人说星期六，但是我根据平闰年的计算得出这一天应该是星期一。我后来查了一下华为手机上的日历，他说是星期六……就很迷，据说是因为耶稣诞生星期六？？？我是共产主义接班人，相信科学，是不相信耶稣的，因此我还是当星期一来计算，反正日历表对的就完了……）下面是测试：

（图 8-1）

（图 8-2）



（图 8-1）

（图 8-2）

附录：项目源码：

```
//demo.cpp
#include"calculagraph.h"
#include"calTime.h"
#include"calendar.h"
int main() {
    int nSelection = 0;
    while (1) {
        cout << "***********主菜单***********" << endl;
        cout << "1.使用计时器(倒计时或累计时)" << endl;
        cout << "2.使用派生类 DateTime 时间功能" << endl;
        cout << "3.使用派生类 cDate 日历功能" << endl;
        cout << "0.使用其他数字键退出程序" << endl;
        cout << "*****************************" << endl;
        cout << "请选择使用的功能:";
        cin >> nSelection;
        cin.clear();
        cin.sync();
        switch (nSelection) {
            case 1:calculagraph(); break;
            case 2:calTime(); break;
            case 3:calendar(); break;
            default: return 0;
        }
    }
}

//date.h
#pragma once
#include<iostream>
#include<iomanip>
```

```
#include<windows.h>
using namespace std;
bool isLeapYear(int year);
class date {
public:
	date(int y = 1, int m = 1, int d = 1);
	date(date& dat);
	void setYear(int y);
	void setMonth(int m);
	void setDay(int d);
	int getYear() { return year; }
	int getMonth() { return month; }
	int getDay() { return day; }
	void showDate();
	date normal(int oper,int days);
	date operator + (int days);
	date operator - (int days);
	date operator ++ ();
	date operator -- ();
	date operator ++ (int);
	date operator -- (int);
	int operator - (date& b);
protected:
	int    year, month, day;
};
//date.cpp
#include"date.h"

int CommonYear[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };
//平年每月天数,如果是闰年则：CommonYear[1] = 29
```

```
//[函数]      isLeapYear
//[功能]      判断是否是闰年
//[参数]      int year：年份
//[返回]      true:是闰年 false:否
bool isLeapYear(int year) {
    bool OK = 1;
    if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) OK = 1;
    else OK = 0;
    return OK;
}

//[函数]      date::date（构造函数）
//[功能]      构造函数，能判断输入的信息是否非法。
//[参数]      y：年份          m：月份          d：日期
//[返回]      void
date::date(int y, int m, int d) :year(y), month(m), day(d) {
    if (isLeapYear(y)) CommonYear[1] = 29;          //判断当前年是否是闰年
    else CommonYear[1] = 28;
    if (y == -1 && m == -1 && d == -1) {
        year = 0; month = 0; day = 0;       //便于 calTime 中的构造函数使用
    }
    else if (m < 1 || m > 12 || d < 1 || d > CommonYear[m - 1]) {
        cout << "日期非法！已设置为 0001 年 1 月 1 日！" << endl;
        year = 1; month = 1; day = 1;       //设置为指定日期
    }
}

//[函数]      date::date（复制构造函数）
//[功能]      复制构造函数
//[参数]      date& dat
//[返回]      void
date::date(date& dat) {
```

```
    this->year = dat.year;
    this->month = dat.month;
    this->day = dat.day;
}

//[函数]      date::setYear
//[功能]      设置年份
//[参数]      int y ： 年份；
//[返回]      void
void date::setYear(int y) {
    if (y < 1) {
        cout << "年份设置非法！请重新设置！" << endl;
        return;
    }
    else {
        year = y;
    }
}

//[函数]      date::setMonth
//[功能]      设置月份
//[参数]      int m ： 月份；
//[返回]      void
void date::setMonth(int m) {
    if (m < 1 || m > 12) {
        cout << "月份设置非法！请重新设置！" << endl;
        return;
    }
    else {
        month = m;
    }
}
```

```
//[函数]      date::setDay
//[功能]      设置日期
//[参数]      int d ： 日期;
//[返回]      void
void date::setDay(int d) {
    if (d < 1 || d > CommonYear[month - 1]) {
        cout << "日期设置非法！请重新设置！" << endl;
        return;
    }
    else {
        day = d;
    }
}

//[函数]      date::showDate（复制构造函数）
//[功能]      打印输出年月日
//[参数]      void
//[返回]      void
void date::showDate() {
    cout << "今日日期为：" << year << "年" << month << "月" << day << "日" << endl;
}

//[函数]      date::重载运算符-
//[功能]      重载运算符-，用以实现计算两个 date 对象之间所差天数
//[参数]      date& b：进行减法运算的对象
//[返回]      int：返回所减天数
int date::operator - (date& b) {          //支持自动比较两个日期大小返回差值
    int count = 0;
    date high;          //high 为两日期中较大的
    date temp;          //temp 为两日期中较小的
    bool OK = 1;          //1 为 b 在*this 的日期之前。如 this（2001，8，30），b（2000，1，1）。0 相
```

反

```
    int y, m, d;
    //判断二者先后
    y = this->year - b.year;            //若 b 在 this 之前，此值为正
    m = this->month - b.month;
    d = this->day - b.day;
    if (y < 0) OK = 0;
    if (y == 0 && m < 0) OK = 0;
    if (y == 0 && m == 0 && d < 0)OK = 0;
    if (y == 0 && m == 0 && d == 0) {
        return 0;
    }
    if (OK == 1) { temp = b; high = *this; }
    else { temp = *this; high = b; }
    //进行计数；
    if (isLeapYear(temp.year)) CommonYear[1] = 29;          //判断当前年是否是闰年
    else CommonYear[1] = 28;
    while (temp.year != high.year || temp.month != high.month || temp.day != high.day) {
        if (temp.day < CommonYear[temp.month - 1]) {            //天数小于当前月份天数，可以自加
            temp.day++;       //日期自加 1
            count++;          //相差天数自加 1
        }
        else if (temp.day == CommonYear[temp.month - 1] && temp.month != 12) {       //非 12 月的月底
            temp.month++;
            temp.day = 1;
            count++;
        }
        else if (temp.day == CommonYear[temp.month - 1] && temp.month == 12) {       //12 月月底
            temp.year++;
            temp.month = 1;
            temp.day = 1;
            count++;
```

```
            if (isLeapYear(temp.year)) CommonYear[1] = 29;          //判断今年（下一年）是否是闰年
            else CommonYear[1] = 28;
        }
    }
    return count;
}

//[函数]      date::normal
//[功能]      进位函数用以对进位进行改变
//[参数]      int oper ： oper 为 1 则是加法，oper 为 0 则是减法
//[返回]      void
date date::normal(int oper, int days) {
    date temp(*this);
    if (oper == 1) {
        if (isLeapYear(temp.year)) CommonYear[1] = 29;          //判断当前年是否是闰年
        else CommonYear[1] = 28;
        while (days > 0) {
            if (temp.day < CommonYear[temp.month - 1]) {          //天数小于当前月份天数，可以自加
                temp.day++;      //日期自加 1
                days--;          //需要天数自减 1
            }
            else if (temp.day == CommonYear[temp.month - 1] && temp.month != 12) {       //非 12 月的
月底
                temp.month++;
                temp.day = 1;
                days--;
            }
            else if (temp.day == CommonYear[temp.month - 1] && temp.month == 12) {       //12 月月底
                temp.year++;
                temp.month = 1;
                temp.day = 1;
                days--;
```

```
                if (isLeapYear(temp.year)) CommonYear[1] = 29;          //判断今年（下一年）是否是闰
年
                else CommonYear[1] = 28;
            }
        }
    }
     if (oper == 0) {
        if (isLeapYear(temp.year)) CommonYear[1] = 29;          //判断当前年是否是闰年
        else CommonYear[1] = 28;
        while (days > 0) {
            if (temp.day > 1) {          //天数大于 1 天，可以自减
                temp.day--;         //日期自减 1
                days--;         //需要天数自减 1
            }
            else if (temp.day == 1 && temp.month != 1) {         //非一月初
                temp.month--;          //月份自减 1
                temp.day = CommonYear[temp.month - 1];      //日期变为当前（前一个月）月的总天数
                days--;
            }
            else if (temp.day == 1 && temp.month == 1) {         //一月初
                temp.year--;
                temp.month = 12;
                temp.day = 31;
                days--;
                if (isLeapYear(temp.year)) CommonYear[1] = 29;          //判断今年（下一年）是否是闰
年
                else CommonYear[1] = 28;
            }
        }
    }
    *this = temp;
    return temp;
```

```
}

//[函数]      date::重载运算符+
//[功能]      重载运算符+，用以实现对对象 date 加上任意天数。
//[参数]      int days：总共所加天数
//[返回]      date 型的对象，用以返回数据
date date::operator + (int days) {
    this->normal(1, days);
    return *this;
}

//[函数]      date::重载运算符-
//[功能]      重载运算符-，用以实现对对象 date 减去任意天数。
//[参数]      int days：总共所减天数
//[返回]      date 型的对象，用以返回数据
date date::operator - (int days) {
    this->normal(0, days);
    return *this;
}

//[函数]      date::重载运算符  前置自减--
//[功能]      重载运算符--，用以实现对对象 date 自减 1 天
//[参数]      调用重载运算符-进行减 1 天。
//[返回]      date 型的对象，用以返回数据
date date::operator -- (){
    *this = *this - 1;
    return *this;
}

//[函数]      date::重载运算符  后置自减--
//[功能]      重载运算符--，用以实现对对象 date 自减 1 天，但返回未自减的值
//[参数]      调用重载运算符-进行减 1 天。
```

```
//[返回]       date 型的对象 temp，用以返回未自减时的数据
date date::operator -- (int) {
    date temp(*this);
    *this = *this - 1;
    return temp;
}

//[函数]       date::重载运算符  前置自加++
//[功能]       重载运算符++，用以实现对对象 date 自加 1 天
//[参数]       调用重载运算符+进行加 1 天。
//[返回]       date 型的对象，用以返回数据
date date::operator ++ () {
    *this = *this + 1;
    return *this;
}

//[函数]       date::重载运算符  后置自加++
//[功能]       重载运算符++，用以实现对对象 date 自加 1 天，但返回未自加的值
//[参数]       调用重载运算符+进行加 1 天。
//[返回]       date 型的对象 temp，用以返回未自加时的数据
date date::operator ++ (int) {
    date temp(*this);
    *this = *this + 1;
    return temp;
}

//calculagraph.h
#pragma once
#include"date.h"
void calculagraph();
void countDownTimer();
void cumulativeTimer();
```

```cpp
//calculagraph.cpp
//计时器，用以实现高考倒计时器和备考时间累积器
#include"calculagraph.h"

//[函数]	calculagraph
//[功能]	提供选择使用倒计时器 countDownTimer，或累加器 cumulativeTimer
//[参数]	void
//[返回]	void
void calculagraph(){
	int nSelection;
	cout << "请选择使用(1.高考倒计时器　2.备考时间累积器　其他数字键返回):";
	cin >> nSelection;
	cin.clear();		//清空缓冲区
	cin.sync();
	system("cls");

	if (nSelection == 1) {		//倒计时器
		countDownTimer();
	}
	else if (nSelection == 2) {		//备考时间累计器
		cumulativeTimer();
	}
	else return;
	return;
}

//[函数]	countDownTimer
//[功能]	使用高考倒计时器,对指定时间进行累减操作
//[参数]	void
//[返回]	void
void countDownTimer() {
```

```
int y, m, d, days, nSelection = 0, nDay = 0;
cout << "请输入结束的时间(如 2019 6 7):";
cin >> y >> m >> d;
cin.clear();
cin.sync();
date end(y, m, d);
cout << "请输入倒计时的天数(如 97):";
cin >> days;
if (days < 0) {
    cout << "倒计时天数非法！" << endl;
    system("pause");
    system("cls");
    return;
}
cin.clear();
cin.sync();
system("cls");
date temp;
temp = end - days;          //置于初始天数
while (days >= 0) {
    cout << "----------高考倒计时----------" << endl;
    temp.showDate();          //输出今日日期
    cout << "距离高考还剩:" << days << "天" << endl;
    if (days != 0) {
        cout << "------------------------------" << endl;
        cout << "请选择(1.剩余时间减少 1 天   2.剩余时间减少指定天数   其他数字键退出):";
        cin >> nSelection;
        cin.clear();
        cin.sync();
        if (nSelection == 1) {
            temp++;
            days--;
```

```
        }
        else if (nSelection == 2) {
            cout << "请输入指定减少的天数:";
            cin >> nDay;
            if (nDay > days) {          //判断指定减少天数是否大于剩余天数
                cout << "指定减少的天数大于剩余天数!错误！"<< endl;
                system("pause");
                system("cls");
                continue;
            }
            if (nDay < 0) {         ////判断指定减少天数是否为负数
                cout << "指定减少的天数为负数!错误！" << endl;
                system("pause");
                system("cls");
                continue;
            }
            cin.clear();
            cin.sync();
            temp = temp + nDay;
            days -= nDay;
        }
        else {
            system("cls");
            return;
        }
        system("cls");
    }
    else {
        cout << "祝高考顺利，金榜题名！" << endl;
        system("pause");
        system("cls");
        return;
```

```
        }
    }
}

//[函数]      cumulativeTimer
//[功能]      使用累加器,对指定初始时间进行累加，计算总备考时间;
//[参数]      void
//[返回]      void
void cumulativeTimer() {
    int y, m, d, count = 1, nSelection = 0, nDay = 0;
    cout << "请输入开始累计时间的日期(如 2019 1 1):";
    cin >> y >> m >> d;
    cin.clear();
    cin.sync();
    date start(y, m, d);
    system("cls");
    date temp(start);
    while (1) {
        cout << "----------备考时间累加器----------" << endl;
        temp.showDate();          //输出今日日期
        cout << "已经备考:" << count << "天" << endl;
        cout << "----------------------------" << endl;
        cout << "请选择(1.备考时间增加 1 天   2.备考时间增加指定天数   其他数字键退出):";
        cin >> nSelection;
        cin.clear();
        cin.sync();
        if (nSelection == 1) {
            temp++;
            count++;
        }
        else if (nSelection == 2) {
            cout << "请输入指定增加的天数:";
```

```
            cin >> nDay;
            if (nDay < 0) {          //判断指定减少天数是否为负数
                cout << "指定减少的天数为负数!错误！" << endl;
                system("pause");
                system("cls");
                continue;
            }
            cin.clear();
            cin.sync();
            temp = temp + nDay;
            count += nDay;
        }
        else {
            system("cls");
            return;
        }
        system("cls");
    }
}

//DateTime.h
#pragma once
#include"date.h"
class DateTime :public date {
public:
    DateTime(int y = 1, int m = 1, int d = 1, int h = 0, int min = 0, int sec = 0);
    DateTime(DateTime& dat);
    DateTime operator +(DateTime& dat);
    DateTime operator -(DateTime& dat);
    DateTime normalForm(int n);       //将 this 转换成标准形式
    void show();
private:
```

```
    int hour, minute, second;
};

//DateTime.cpp
#include"DateTime.h"

//[函数]    DateTime::normalForm
//[功能]    将 this 转换为 24 小时制下的标准的时间
//[参数]    int n:取决于是否需要进位到月，进位为 1.不进位为 0；
//[返回]    DateTime
DateTime DateTime::normalForm(int n) {
    int temp_day = 0 ,temp_hour = 0, temp_min = 0;
    if (this->second >= 0) {
        temp_min = this->second / 60;          //保存需要进位到 minute 上的秒数
        this->second %= 60;
        this->minute += temp_min;
    }
    else {
        this->second += 60;
        this->minute -= 1;
    }
    if (this->minute >= 0) {
        temp_hour = this->minute / 60;
        this->minute %= 60;
        this->hour += temp_hour;
    }
    else {
        this->minute += 60;
        this->hour -= 1;
    }
    if (this->hour >= 0) {
        if (n == 0) { //若不需要进位，则直接加
```

```
            return *this;
        }
        else if (n == 1) {
            temp_day = this->hour / 24;
            this->hour %= 24;
            this->date::operator+(temp_day);        ////注意!!!!!!!!!!!!!! 记得测试这里能否成功运行
////结果：成功，嘻嘻！
        }
    }
    else {
        int count = 1;          //count 用来统计需要减几天补位到 hour 上
        while ((count * 24 + this->hour) < 0) {
            count++;
        }
        this->hour += (count * 24);
        this->date::operator-(count);        //减去 count 天
    }
    return *this;
}

//[函数]      DateTime::DateTime（构造函数）
//[功能]      构造包含年月日时分秒的标准时间派生类对象
//[参数]      int y = 1, int m = 1, int d = 1, int h = 0, int min = 0, int sec = 0
//[返回]      void
DateTime::DateTime(int y, int m, int d, int h, int min, int sec):date(y,m,d),hour(h),minute(min),second(sec) {
    if (y != -1 && m != -1 && d != -1) {
        this->normalForm(1);          //需要进位
    }
}

//[函数]      DateTime::DateTime（复制构造函数）
//[功能]      构造包含年月日时分秒的标准时间派生类对象
```

```
//[参数]       DateTime& dat
//[返回]       void
DateTime::DateTime(DateTime& dat){
    this->year = dat.year;
    this->month = dat.month;
    this->day = dat.day;
    this->hour = dat.hour;
    this->minute = dat.minute;
    this->second = dat.second;
}

//[函数]       DateTime::show
//[功能]       输出日期时间信息
//[参数]       void
//[返回]       void
void DateTime::show() {
    cout << "今日日期为：" << year << "年" << month << "月" << day <<
        "日" << hour << "时" << minute << "分" << second << "秒" << endl;
}

//[函数]       DateTime::重载运算符+
//[功能]       重载运算符+，用以实现计算两个 DateTime 对象数据之和
//[参数]       DateTime& dat:进行加法运算的对象
//[返回]       DateTime:  返回加法的结果
DateTime DateTime::operator +(DateTime& dat) {
    dat.normalForm(0);        //防止输入数据溢出，不进位
    this->hour += dat.hour;
    this->minute += dat.minute;
    this->second += dat.second;
    this->normalForm(1);          //将 this 标准化，进位
    return *this;
}
```

```
//[函数]       DateTime::重载运算符-
//[功能]       重载运算符-，用以实现计算两个 DateTime 对象数据之差
//[参数]       DateTime& dat:进行减法运算的对象
//[返回]       DateTime:  返回加法的结果
DateTime DateTime::operator -(DateTime& dat) {
    dat.normalForm(0);          //防止输入数据溢出，不进位
    this->hour -= dat.hour;
    this->minute -= dat.minute;
    this->second -= dat.second;
    this->normalForm(1);            //将 this 标准化，进位
    return *this;
}

//calTime.h
#pragma once
#include"DateTime.h"
void calTime();

//calTime.cpp
#include"calTime.h"

//[函数]       calTime
//[功能]       提供使用时间计算器
//[参数]       void
//[返回]       void
void calTime() {
    int nSelection;
    int y, m, d, h, min, sec;
    system("cls");
    cout << "**********************时 间 计 算 器**********************" << endl;
    cout << "请输入需要操作的时间(如 2020 7 8 23 59 59):";
```

```
    cin >> y >> m >> d >> h >> min >> sec;
    cin.clear();        //清空缓冲区
    cin.sync();
    DateTime start(y, m, d, h, min, sec);
    start.show();
    cout << "************************************************************" << endl;
    cout << "请选择功能(1.增加时间   2.减少时间   其他数字键返回):";
    cin >> nSelection;
    cin.clear();        //清空缓冲区
    cin.sync();
    if (nSelection == 1) {      //增加时间
        cout << "请指定增或减的时间(如 23 59 59,支持溢出自动进位):";
        cin >> h >> min >> sec;
        if (h < 0 || min < 0 || sec < 0) {
            cout << "时间指定时出现负数！错误！" << endl;
            system("pause");
            system("cls");
            return;
        }
        else {
            DateTime temp(-1, -1, -1, h, min, sec);      //便于构造 date 时将参数设置为 0；
            start = start + temp;
            start.show();
            system("pause");
            system("cls");
            return;
        }
    }
    else if (nSelection == 2) {      //减少时间
        cout << "请指定减少时间(如 23 59 59,支持溢出自动进位):";
        cin >> h >> min >> sec;
        if (h < 0 || min < 0 || sec < 0) {
```

```
            cout << "时间指定时出现负数！错误！" << endl;
            system("pause");
            system("cls");
            return;
        }
        else {
            DateTime temp(-1, -1, -1, h, min, sec);       //便于构造 date 时将参数设置为 0；
            start = start - temp;
            start.show();
            system("pause");
            system("cls");
            return;
        }
    }
    else return;
    return;
}

//CDate.h
#pragma once
#include"date.h"
class CDate :public date {
public:
    CDate(int y = 1, int m = 1);
    void display();
private:
    int firstDay;       //这个月的第一天
};
//CDate.cpp
#include"CDate.h"
#include"date.h"
```

```
int commonYear[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };
//平年每月天数,如果是闰年则：CommonYear[1] = 29


//[函数]       CDate:CDate(构造函数)
//[功能]       构造函数，并判断第一天是星期几
//[参数]       int y,int m          倘若 m 为 0 代表 firstDay 为 1 年的开始
//[返回]       void
CDate::CDate(int y, int m):date(y,m,1){
    date temp(1, 1, 1);          //这天为星期一，具体可推算，我们要相信科学!!!
    this->firstDay = (*this - temp) % 7 + 1;
}
//[函数]       CDate:display
//[功能]       打印该月的日历
//[参数]       void
//[返回]       void
void CDate::display() {
    int count = 0;
    if (isLeapYear(this->year)) commonYear[1] = 29;          //判断当前年是否是闰年
    else commonYear[1] = 28;
    count = commonYear[this->month - 1];
    cout << this->year << "年" << this->month << "月日历" << endl;
    cout << std::left << setw(5) << "周一" << setw(5) << "周二" << setw(5) << "周三" << setw(5) << "周
四"
        << setw(5) << "周五" << setw(5) << "周六" << setw(5) << "周日" << endl;
    int i, j;
    for (i = 1; i < this->firstDay; i++) {
        cout << setw(5) <<" ";
    }
    for (j = 1; j <= count; j++) {
        cout << std::left <<    setw(5) << j;
        if ((this->firstDay + j - 1) % 7 == 0) {
            cout << endl;
```

```
            }
        }
        cout << endl;
}
//calendar.h
#pragma once
#include"CDate.h"
void calendar();
//calendar.cpp
#include"calendar.h"
//[函数]        calendar
//[功能]        提供日历相关操作
//[参数]        void
//[返回]        void
void calendar() {
        int y, m;
        system("cls");
        cout << "**********************日历**********************" << endl;
        cout << "请选择需要输出日历的年份及月份(如 2019 6):";
        cin >> y >> m;
        cin.clear();
        cin.sync();
        CDate temp(y, m);
        temp.display();
        cout << "************************************************" << endl;
        system("pause");
        system("cls");
}
```