面向对象程序设计基础作业四 设计文档

1. 模型部分

   a. 功能简述

   本程序实现了如下功能：在程序启动时，自动从系统中调用系统日期并存储于日历类内。之后进入程序循环，用户可以通过输入指令完成如下操作（详见程序内的 help 界面）：

diff	输出当前日期至给定日期之间的日期差距。
exit	退出程序。
future	输出当前日期起给定数量天数后的日期。
reset	重置当前日期到系统日期。
set	设置当前日期为给定日期。
show	显示当前日期

   b. 算法

      i. 获取系统日期

      采用 ctime 库内的 localtime 函数，并用 tm 结构储存。本程序只关心年月日信息，故直接舍弃其余信息，而把年月日信息存储于类内。

      ii. “下一日”的算法

      类内设计一私有函数 calendar::advance()，其效果为使当前天数前进一天。若日期大于当前月最大日期，则月份+1，日期重置为 1；若当前月份大于 12，则年份+1，月份重置为 1。特例为，公元前 1 年 12 月 31 日的下一天为公元 1 年 1 月 1 日。

      iii. 日期差距计算

      我们知道每 400 年为时间上的一个周期，且每周期的天数为：$365 \times 400 + 100 - 4 + 1 = 146097$天。因此，在计算日期差距

时，我们先计算两个日期之间相差多少个完整的“四百年”，之后在四百年内循环调用 advance()函数，即可算出最终的日期差距。

相较于直接循环调用 advance()函数，这样做的时间复杂度为 O(1)（尽管它的常数比较大）；也就是说，在代码长度没有限制的情况下，此算法在时间上和一次查表是等价的。

iv. 给定数量天数后的日期

与 ii 中一样，首先我们从日期数量中剥离 146097 的整数倍，并在当前日期的基础上增加或减少 400 年的对应倍数。之后逐年调整年份，并在期间判断是否经过了一个闰日，最后计算出开始、结束日期在当年的日期序号，并据此调整天数。这样做的时间复杂度同样是 O(1)。

c. 数据结构

本程序包括主函数源文件（main.cpp）和 1 个额外源文件及其头文件（calendar.cpp/h）。

在 calendar.cpp 内定义了 calendar 类，包含一系列公用接口和内部计算使用的私有函数，以及 3 个私有变量：year, month, day，存储年月日。具体接口、函数声明如下，注释为其用途解释：（date 是一个包含三个整型变量的结构体）

```
class calendar
{
public:
    calendar();
    calendar(int yy, int mm, int dd);
    // starts with the current date read from system.
    ~calendar();
    // does the cleanup. So far just a trivial destructor.
    bool setdate(int yy, int mm, int dd);           // sets the date to yy/mm/dd, returns whether this is valid
    void showdate() const;                          // shows the recorded date, a convenient function
    static void showdate(int yy, int mm, int dd);   // shows any date
    date readdate();                                // returns the stored date in terms of a date struct
    void show_date_after(long long time_interval);  // shows time_interval days after the recorded date
    void show_date_diff(int yy, int mm, int dd);    // inverse function of the last one
    bool is_valid(int yy, int mm, int dd);          // judges whether a certain date is valid
    void reset();                                   // resets the date to system date UTC+8


private:
    bool isleap(long long);                         // judges whether a year is a leap one
    long long date_diff(int yy, int mm, int dd);    // private function for date difference
    int yearorder(int yy, int mm, int dd);          // a convenient function: determines which day it is in a year
    void advance();
    int year, month, day;                           // these record as they seem
};
```

2. 验证部分

a. 单元测试

我们将逐一测试每个较为复杂的公有接口。

i. set_date()

```
Input command (help for manual): set
Input a date in YYYY MM DD format: 2022 13 34
INVALID DATE!
Input a date in YYYY MM DD format: 1900 2 29
INVALID DATE!
Input a date in YYYY MM DD format: 0 3 9
INVALID DATE!
Input a date in YYYY MM DD format: 2000 2 29
Date stored: A.D. 2000/02/29
Press ENTER to continue
```

我们希望能测试 set_date 是否拒绝所有不合法的日期。

如左图所示，此函数可以拒绝如下错误输入：月份、日期大于合法数值；在能被 100 整除但不能被 400 整除的年份的 2 月 29 日；并不存在的“公元 0 年”。

ii. show_date_after()

利用网络上的日期计算器[1]，我们可以对我们的程序进行交叉验证。下表的“预期日期”即为此网站的计算结果。

需要注意，此网站未能考虑并不存在的“公元 0 年”，因此跨越公元前后的计算结果需要相应地增加或减少一年，且由于此网站判断公元 0 年是闰年，故在公元交界附近，该网站的输出结果可能会与真实值相差一天。在考虑这一点之后，本程序的输出结果全部正确**。

内部存储的日期	天数	预期日期	程序输出结果
2022/3/20	2000	2027 年 9 月 10 日	A.D. 2027/09/10
	19023532	54106 年 11 月 17 日	A.D. 54106/11/17
	-2000	2016 年 9 月 27 日	A.D. 2016/09/27
	-12045189	-30957 年 8 月 11 日**	B.C. 30958/08/11
公元前 1/3/20	365	0 年 3 月 19 日**	A.D. 1/03/20

iii. show_date_diff()

利用 i 中的结果反向验证：

内部存储的日期	目标日期	预期间隔	程序输出结果
2022/03/20	2027/09/10	2000 天	2000
	54106/11/17	19023532 天	19023532
	2016/09/27	-2000 天	-2000
	公元前 30958/08/11	-12045189 天	-12045189
公元前 1/3/20	1/3/20	365 天	365

[1] http://bjtime.cn/riqi/