C++语言程序设计作业1
Code
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
以下部分可能超出本周所学
Part 8*
Part 9*

C++语言程序设计作业1

助教：叶开 ye_kai@pku.edu.cn

2022年9月22日

- 说明：补充已有代码，实现一个日期类
- 注意：
  - `main` 函数中包含9个部分，用于验证你的实现是否正确（为了通过编译，附件的 `main.cpp` 中，这些部分都被注释了）
  - 只能在 `TODO` 所示的空行附近补充代码，而且，请按照要求在指定位置补充
    - 有些部分其实既可以在 `TODO 4` 补充，也可以在 `TODO 5` 补充；但是为了第9部分的教学需要，请按照说明，补充在指定位置
  - 如果有问题随时联系助教，包括讲义作业错误和学习上的困难
- 评分规则：
  - 通过编译，输出无误 (70%)
  - 代码是按照说明正确实现的 (30%)
  - 每逾期一天，减少 10%，至多减少 50%
- 提交：
  - 截止：2022年10月4日23:59
  - 请在教学网提交 `main.cpp` ，重命名为学号，例如 `2100012345.cpp` ，无需其他任何文件

Code

```
#include <iostream>

using namespace std;

bool isLeap(int y) {

    // TODO 1
```

```
    return true;
}

int numDaysOfYear(int y) {

    // TODO 2

    return 0;
}

int numDaysOfMonth(int y, int m) {

    // TODO 3

    return 0;
}

class Date {
private:

    int daysFromEpoch() const {
        int cnt = 0;
        for (int y = 1970; y < this->year; ++y) {
            cnt += numDaysOfYear(y);
        }
        for (int m = 1; m < this->month; ++m) {
            cnt += numDaysOfMonth(this->year, m);
        }
        cnt += this->day - 1;
        return cnt;
    }


    void makeFromTimestamp(int timestamp) {
        this->year = 1970;
        this->month = 1;
        this->day = 1;
        while (timestamp >= numDaysOfYear(this->year)) {
            timestamp -= numDaysOfYear(this->year);
            ++this->year;
        }
        while (timestamp >= numDaysOfMonth(this->year, this->month)) {
            timestamp -= numDaysOfMonth(this->year, this->month);
            ++this->month;
        }
        this->day += timestamp;
    }


public:
    int year;
    int month;
    int day;
```

```
    int weekDay() const {
        return (4 + daysFromEpoch()) % 7;
    }

    // TODO 4

};

// TODO 5


ostream& operator <<(ostream& os, const Date& rhs) {
    return os << rhs.year << '/' << rhs.month << '/' << rhs.day;
}



int main() {


    cout << "----------------" << endl;

    // 1. constructor

    cout << "1. constructor" << endl;

    Date a(2022);
    Date b(2022, 2);
    Date c{ 2022, 9, 14 };

    cout << "----------------" << endl;

    // 2. Date - Date

    cout << "2. Date - Date" << endl;

    cout << b - a << endl;
    cout << c - Date{ 1970,1,1 } << endl;

    cout << "----------------" << endl;

    // 3. weekDay

    cout << "3. weekDay" << endl;

    cout << c.weekDay() << endl;

    cout << "----------------" << endl;

    // 4. Date + int

    cout << "4. Date + int" << endl;

    cout << Date{ 1970,1,1 } + 10000 << endl;
    cout << Date{ 1970,1,1 } + (c - Date{ 1970,1,1 }) << endl;
```

```
    cout << "----------------" << endl;

    // 5. int + Date

    cout << "5. int + Date" << endl;

    cout << 10000 + Date{ 1970,1,1 } << endl;

    cout << "----------------" << endl;

    // 6. Date - int

    cout << "6. Date - int" << endl;

    cout << c - (c - Date{ 1970,1,1 }) << endl;

    cout << "----------------" << endl;

    // 7. Date += int

    cout << "7. Date += int" << endl;

    a += 100;
    cout << a << endl;

    cout << "----------------" << endl;

    // 8*. (Date -= int) -> Date&

    cout << "8*. (Date -= int) -> Date&" << endl;

    cout << (a -= 50) << endl;
    cout << ((a -= 25) -= 25) << endl;
    cout << a << endl;

    cout << "----------------" << endl;

    // 9*. const this

    cout << "9*. const this" << endl;

    const Date d = a;
    cout << d + 100 << endl;
    cout << 100 + d << endl;
    cout << d - 100 << endl;
    cout << d - d << endl;

    cout << "----------------" << endl;

    return 0;
}
```

Part 0

- 阅读已有代码，理解 `Date` 类的结构，它包含一些已经实现的部分：
  - `daysFromEpoch` 成员函数：返回当前对象所示的日期，自1970年1月1日起经过的天数
  - `makeFromTimestamp` 成员函数：根据指定的参数（它是自1970年1月1日起经过的天数），设置当前对象所示的日期
  - `year, month, day` 成员变量：分别代表当前对象所示的年、月、日
  - `weekDay` 成员函数：返回当前对象所示的日期是星期几
    - 返回0、1、2、3、4、5、6，分别代表周日、周一、周二、周三、周四、周五、周六
    - 注1：1970年1月1日是周四
    - 注2：仅为完成作业而言，你可以不必关心这一函数，因为已经被实现好了
- 阅读已有代码，理解三个空函数是在何处被用到、需要实现什么功能：
  - `isLeap` 函数：接受一个参数为年份，返回这一年是否是闰年
  - `numDaysOfYear` 函数：接受一个参数为年份，返回这一年的天数
  - `numDaysOfMonth` 函数：接受两个参数年份和月份，返回这个月的天数（需要知道年份是因为2月的天数可变）

Part 1

- 补充 `TODO 4`，填写构造函数，使得 `1. constructor` 部分可以通过编译并正常运行
- 注意，`Date` 类的构造函数可以接受三种：
  - 仅提供年份，此时是1月1日
  - 仅提供年份和月份，此时是1日
  - 提供年、月、日
- 输出应为：

```
----------------
1. constructor
----------------
```

Part 2

- 补充 `TODO 1-3`，完成 `isLeap` 、 `numDaysOfYear` 、 `numDaysOfMonth` 三个函数的功能
- 补充 `TODO 4`，完成 `Date` 类之间的减法运算，减法的结果是返回第一个日期比第二个日期大的天数（你可以使用 `daysFromEpoch` 帮忙）
- 输出应为：

```
----------------
2. Date - Date
31
19249
----------------
```

Part 3

- 无需补充代码
- 输出应为：

```
----------------
3. weekDay
3
----------------
```

Part 4

- 补充 `TODO 4`，完成 `Date` 类加上一个整数的运算，它会得到一个新的日期对象，表示原来的日期经过指定的天数之后的日期
- 提示：你应该创建一个新的 `Date` 对象，修改后返回，而不能修改原有的 `Date` 对象
- 输出应为：

```
----------------
4. Date + int
1997/5/19
2022/9/14
----------------
```

Part 5

- 补充 `TODO 5`，完成一个整数加上 `Date` 类的运算，它会得到一个新的日期对象，表示原来的日期经过指定的天数之后的日期
- 提示：你应该创建一个新的 `Date` 对象，修改后返回，而不能修改原有的 `Date` 对象
- 输出应为：

```
----------------
5. int + Date
1997/5/19
----------------
```

Part 6

- 补充 `TODO 4`，完成 `Date` 类减去一个整数的运算，它会得到一个新的日期对象，表示原来的日期倒推指定的天数之后的日期
- 提示：你应该创建一个新的 `Date` 对象，修改后返回，而不能修改原有的 `Date` 对象
- 输出应为：

```
----------------
6. Date - int
1970/1/1
----------------
```

Part 7

- 补充 `TODO 4`，完成 `Date` 类加上一个整数的运算，它**不会**得到一个新的日期对象，而是修改原来的对象，使得它表示原来的日期加上指定的天数之后的日期
- 输出应为：

```
----------------
7. Date += int
2022/4/11
----------------
```

以下部分可能超出本周所学

- 如果感到困难可以暂且搁置，下周完成；或者自学引用和 `const` 成员函数的部分，然后完成
- 这两个部分可能不计分，根据下周学习情况而定

Part 8*

- 补充 `TODO 4`，完成 `Date` 类减去一个整数的运算，它**不会**得到一个新的日期对象，而是修改原来的对象，使得它表示原来的日期倒推指定的天数之后的日期
- **而且**，减法必须返回原对象自身的引用，你可以观察示例输出，理解 `((a -= 25) -= 25)` 的作用
- 输出应为：

```
----------------
8*. (Date -= int) -> Date&
2022/2/20
2022/1/1
2022/1/1
----------------
```

Part 9*

- 这一部分除了声明变量 `d`，还包含了四条语句，你可以逐条取消注释，尝试编译，看看其中是否有无法通过编译的部分
- 如果有，请修改相应的非 `const` 成员函数，使得它成为 `const` 修饰的成员函数，然后编译
- 思考（如果需要，查找资料）
  - 思考为什么一开始有的语句不能通过编译，而另一些却可以？
  - 思考 `const` 修饰后，为什么就可以通过编译？
- 输出应为：

```
----------------
9*. const this
2022/4/11
2022/4/11
2021/9/23
0
----------------
```