

Assignment 4

一、概念题

1.1、简述Demeter法则的基本思想；过度使用Demeter法则会带来什么问题。

一个类的成员函数除了能访问自身类结构的类直接子结构（本类的数据成员）外，不能以任何方式依赖任何其他类的结构；只应向某个有限集合的中的类对象发送消息。

它会导致需要写很多包装（wrapper）函数来将函数调用传播到组件，这会带来不少的时间和空间开销（函数内几乎没有内容，会频繁的创建和销毁栈帧）。在函数层面，会导致接口的窄化。在类层面，一个比较宽的接口可能需要生成很多辅助函数。

https://en.wikipedia.org/wiki/Law_of_Demeter

1.2、简述为什么要对操作符重载进行重载；操作符重载会带来什么问题。

简单方便，提高代码的可读性，比起函数调用，使用运算符更加直观，给开发者带来方便。

很多工作都让编译器做了，复杂的重载有时很难理解代码内部逻辑；而且很容易发生内存泄露；有时会产生歧义。

1.3、简述操作符重载的两种形式；这两种形式有什么区别。

作为一个类的非静态成员函数重载或者作为一个全局(友元)函数来重载。

作为全局函数来实现操作符重载时，其参数至少要有一个具有类、结构、枚举以及它们的引用类型的参数，全部为内置类型不需要重载。

二、编程题

2.1、定义一个日期类Date，其实例为 `year` 年 `month` 月 `day` 日；定义一个时间类Time，其实例为 `hour` 时 `minute` 分 `second` 秒；定义一个日期时间类Datetime，其实例为某日/某时。

```
1  #include <sstream>
2  #include <string>
3  class Date {
4      friend class Datetime;
5  private:
6      int year, month, day;
7      static const int dayPerMonth[2][13];
8  public:
9      Date(int _year, int _month, int _day) {
10         year = _year;
11         month = _month;
```

```

12     day = _day;
13 }
14 static bool isLeapYear(int year) {
15     return year % 400 == 0 || (year % 4 == 0 && year % 100 != 0);
16 }
17 bool isLeapYear() const {
18     return isLeapYear(year);
19 }
20 bool operator<(const Date &d) const {
21     return year < d.year || (year == d.year && (month < d.month ||
22         (month == d.month && day < d.day)));
23 }
24 bool operator==(const Date &d) const {
25     return year == d.year && month == d.month && day == d.day;
26 }
27 Date &operator++() {
28     ++day;
29     if (day > dayPerMonth[isLeapYear()][month]) {
30         day = 1;
31         ++month;
32         if (month > 12) {
33             month = 1;
34             ++year;
35         }
36     }
37     return *this;
38 }
39 Date &operator--() {
40     --day;
41     if (day <= 0) {
42         --month;
43         if (month <= 0) {
44             --year;
45             month = 12;
46         }
47         day = dayPerMonth[isLeapYear()][month];
48     }
49     return *this;
50 }
51 Date operator+(long dayNum) const {
52     Date result(*this);
53     if (dayNum > 0) {
54         for (long i = 0; i < dayNum; ++i) {
55             ++result;
56         }
57     } else {
58         for (long i = 0; i > dayNum; --i) {
59             --result;
60         }

```

```

61     }
62     return result;
63 }
64 long operator-(const Date &d) const {
65     Date temp(d);
66     long result = 0;
67     while (temp < *this) {
68         ++temp;
69         ++result;
70     }
71     while (*this < temp) {
72         --temp;
73         --result;
74     }
75     return result;
76 }
77 };
78 const int Date::dayPerMonth[2][13] =
79     {{0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
80      {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}};
81
82 class Time {
83     friend class Datetime;
84 private:
85     int hour, minute, second;
86 public:
87     Time(int _hour, int _minute, int _second) {
88         hour = _hour;
89         minute = _minute;
90         second = _second;
91     }
92     Time add(long addSecond, long &overflowDay) const {
93         int newSecond = second + addSecond % 60;
94         addSecond /= 60;
95         int newMinute = minute + addSecond % 60 + (newSecond >= 60) -
96             (newSecond < 0);
97         newSecond = (newSecond + 60) % 60;
98         addSecond /= 60;
99         int newHour = hour + addSecond % 24 + (newMinute >= 60) -
100             (newMinute < 0);
101         newMinute = (newMinute + 60) % 60;
102         addSecond /= 24;
103         overflowDay += addSecond + (newHour >= 24) - (newHour < 0);
104         newHour = (newHour + 24) % 24;
105         return {newHour, newMinute, newSecond};
106     }
107     long operator-(const Time &t) const {
108         return (hour - t.hour) * 3600 + (minute - t.minute) * 60 + (second
109             - t.second);

```

```

110     }
111     bool operator<(const Time &t) const {
112         return hour * 3600 + minute * 60 + second < t.hour * 3600 +
113             t.minute * 60 + t.second;
114     }
115     bool operator==(const Time &t) const {
116         return hour * 3600 + minute * 60 + second == t.hour * 3600 +
117             t.minute * 60 + t.second;
118     }
119 };
120
121 class Datetime {
122     private:
123         Date date;
124         Time time;
125     public:
126         Datetime(const Date &_date, const Time &_time) : date(_date),
127                                                         time(_time) {}
128         Datetime operator+(long second) const {
129             long addDay = 0;
130             Time newTime = time.add(second, addDay);
131             Date newDate = date + addDay;
132             return {newDate, newTime};
133         }
134         Datetime operator-(long second) const {
135             return *this + (-second);
136         }
137         long operator-(const Datetime &d) const {
138             return (date - d.date) * 86400 + (time - d.time);
139         }
140         Datetime &operator++() {
141             long addDay = 0;
142             time = time.add(1, addDay);
143             date = date + addDay;
144             return *this;
145         }
146         Datetime &operator--() {
147             long addDay = 0;
148             time = time.add(-1, addDay);
149             date = date + addDay;
150             return *this;
151         }
152         bool operator<(const Datetime &d) const {
153             return date < d.date || (date == d.date && time < d.time);
154         }
155         bool operator==(const Datetime &d) const {
156             return date == d.date && time == d.time;
157         }
158     }

```

```
159     std::string to_string() const {
160         std::stringstream ss;
161         ss << date.year << '.' << date.month << '.' << date.day << '/' <<
162             time.hour << '.' << time.minute << '.' << time.second;
163         return ss.str();
164     }
165 };
```

咳咳：参考鲁大师的实现