HW4

181220076 周韧哲

一. 概念题

1. 基本思想是一个类的成员函数除了能访问自身类结构的直接子结构外，不能以任何方式依赖于任何其他类的结构，并且每个成员函数只应向某个有限集合中的对象发送消息。过度使用迪米特法则会使系统产生大量的中介类，从而增加系统的复杂性，使模块之间的通信效率降低。
2. 对于自定义的类，c++没有定义类操作符的含义，操作符重载可以对这些类定义类操作符的含义，实现多态。问题是混淆直觉，很多操作并不像内建操作那样轻巧，有的操作符可以对指针进行操作，容易导致 bug。
3. 一种是用成员函数重载运算符，运算符函数可以以类成员函数或友元函数的形式进行重载；另一种是用全局函数或友元函数重载运算符。区别是当重载为成员函数时，会有一个this指针，指向当前的类，而当重载为全局函数时，将没有隐含的参数this指针，这样将会多一个参数。

二. 编程题

```cpp
class Date{
private:
    int _year, _month, _day;
public:
    Date(){}
    Date(int year, int month, int day){
        _year = year;
        _month = month;
        _day = day;
    }
    friend class Datetime;
};

class Time{
private:
    int _hour, _minute, _second;
public:
    Time(){}
    Time(int hour, int minute, int second){
        _hour = hour;
        _minute = minute;
        _second = second;
    }
    friend class Datetime;
};

class Datetime{
```

```cpp
private:
    Date _date;
    Time _time;
    bool is_leap_year(int year) const{
        if(year%4 != 0)
            return false;
        if(year%100 != 0)
            return true;
        if(year%400 != 0)
            return false;
        return true;
    }
    bool is_31(int month) const{
        return month==1 || month==3 || month==7 || month==8 || month==10 || month==12;
    }
    bool is_30(int month) const{
        return month==4 || month==6 || month==9 || month==11;
    }
    bool is_28(int year, int month) const{
        return month==2 && !is_leap_year(year);
    }
    bool is_29(int year, int month) const{
        return month==2 && is_leap_year(year);
    }
    //increment 1
    void increment(int &year, int &month, int &day, int &hour, int &minute, int &second) const{
        if(second<59) second++;
        else{
            second = 0;
            if(minute<59) minute++;
            else{
                minute = 0;
                if(hour<23) hour++;
                else{
                    hour = 0;
                    if((is_31(month) && day<31) ||
                       (is_30(month) && day<30) ||
                       (is_29(year, month) && day<29) ||
                       (is_28(year, month) && day<28)){
                           day++;
                    }else{
                        day = 1;
                        if(month<12) month++;
                        else{
                            month = 1;
                            year++;
                        }
                    }
                }
            }
        }
    }
    //decrement 1
    void decrement(int &year, int &month, int &day, int &hour, int
&minute, int &second) const{
        if(second>0) second--;
        else{
            second = 59;
            if(minute>0) minute--;
            else{
                minute = 59;
                if(hour>0) hour--;
                else{
                    hour = 23;
                    if(day>1) day--;
                    else{
                        if(month==1) day = 31;
                        else if(is_31(month-1)) day = 31;
                        else if(is_30(month-1)) day = 30;
                        else if(is_29(year, month-1)) day = 29;
                        else if(is_28(year, month-1)) day = 28;
                        else assert(0);
                        if(month>1) month--;
                        else{
                            month = 12;
                            year--;
                        }
                    }
                }
            }
        }
    }
public:
    Datetime(const Date &date, const Time &time){
        _date = date;
        _time = time;
    }
    bool operator==(const Datetime &datetime) const{
        return this->_date._day == datetime._date._day &&
               this->_date._month == datetime._date._month &&
               this->_date._year == datetime._date._year &&
               this->_time._hour == datetime._time._hour &&
               this->_time._minute == datetime._time._minute &&
               this->_time._second == datetime._time._second;
    }
    bool operator<(const Datetime &datetime) const{
        return ! (*this==datetime) &&
               this->_date._day <= datetime._date._day &&
               this->_date._month <= datetime._date._month &&
               this->_date._year <= datetime._date._year &&
               this->_time._hour <= datetime._time._hour &&
               this->_time._minute <= datetime._time._minute &&
```

```cpp
               this->_time._second <= datetime._time._second;
    }
    Datetime operator+(long seconds) const{
        Date date = _date;
        Time time = _time;
        Datetime datetime(date, time);
        for(int i=0;i<seconds;i++)
            increment(datetime._date._year, datetime._date._month, datetime._date._day,
                      datetime._time._hour, datetime._time._minute, datetime._time._second);
        return datetime;
    }
    Datetime operator-(long seconds) const{
        Date date = _date;
        Time time = _time;
        Datetime datetime(date, time);
        for(int i=0;i<seconds;i++)
            decrement(datetime._date._year, datetime._date._month, datetime._date._day,
                      datetime._time._hour, datetime._time._minute, datetime._time._second);
        return datetime;
    }
    void operator++(int){
        increment(_date._year, _date._month, _date._day, _time._hour, _time._minute, _time._second);
    }
    void operator--(int){
        decrement(_date._year, _date._month, _date._day, _time._hour, _time._minute, _time._second);
    }
};
```