张涵之 191220154 第四次作业

概念题

1. C++中操作符的重载遵循哪些基本原则?

a) 只能重载 C++语言中已有的操作符，不可臆造新的操作符;
b) 不能重载下列操作符: “. ”， “.* ”， “?: ”， “:: ”， “sizeof ”;
c) 遵循原有操作符的语法和语义，不改变操作数个数，不改变优先级和结合性。

2. 简述单目运算符(++，--)前置重载和后置重载的差别。

前置为“先加后用”，对原来的对象进行自增或自减操作，返回操作后的对象;
后置为“先用后加”，先保存原来的对象，调用前置++/--重载函数，返回原来的对象。

编程题

1. 阅读以下程序，思考该函数能否正常运行。说明理由并给出修改后的代码。

不能，执行 a2 = a1 操作时，a2.p 与 a1.p 指针指向同一区域。
a1 与 a2 消亡时分别调用析构函数，对 p 指向的同一块内存回收两次，产生运行错误。

```
#include <iostream>
using namespace std;

class A {
public:
    int x;
    int* p;
    A() {
        p = new int(0);
        x = 0;
    }
    A(int m, int n) {
        p = new int(n);
        x = m;
    }
    ~A() {
        delete p;
        x = 0;
    }
    A& operator = (const A& a) {
        *p = *(a.p);
        x = a.x;
```

```
        return *this;
    }
};

int main()
{
    A a1(6, 8);
    A a2;
    a2 = a1;
    cout << "a1.x = " << a1.x << ", " << "*(a1.p) = " << *(a1.p) << endl;
    cout << "a2.x = " << a2.x << ", " << "*(a2.p) = " << *(a2.p) << endl;
    cout << "a1.p = " << a1.p << endl;
    cout << "a2.p = " << a2.p << endl;
    return 0;
}
```

2. 现需要设计一个日期类 Date，它包含年、月、日等数据成员。

```
int day_per_month[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
int day_per_month_leap[13] = { 0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

class Date {
    int year;
    int month;
    int day;

public:
    Date(int y, int m, int d) {
        year = y;
        month = m;
        day = d;
    }
    bool leapyear () const {
        return ((year % 100 != 0 && year % 4 == 0) || year % 400 == 0);
    }
    Date& operator ++() {
        if (month == 12 && day == 31) {
            year += 1;
            month = 1;
            day = 1;
        }
        else {
            int temp = day + 1;
            if (leapyear()) {
```

```
            day = temp % day_per_month_leap[month];
            month += temp / day_per_month_leap[month];
        }
        else {
            day = temp % day_per_month[month];
            month += temp / day_per_month[month];
        }
    }
    return *this;
}
const Date operator ++(int) {
    Date temp = *this;
    ++(*this);
    return temp;
}
Date& operator --() {
    if (month == 1 && day == 1) {
        year -= 1;
        month = 12;
        day = 31;
    }
    else if (day == 1) {
        month -= 1;
        if (leapyear())
            day = day_per_month_leap[month];
        else
            day = day_per_month[month];
    }
    else
        day -= 1;
    return *this;
}
const Date operator --(int) {
    Date temp = *this;
    --(*this);
    return temp;
}
Date operator +(int days) {
    Date d = *this;
    while (days >= 366) {
        if (leapyear() && (month == 1 || day < 29))
            days -= 366;
        else
            days -= 365;
```

```
            d.year += 1;
        }
        while (days >= 31) {
            if (leapyear())
                days -= day_per_month_leap[month];
            else
                days -= day_per_month[month];
            d.month += 1;
            if (d.month == 13) {
                d.year += 1;
                d.month = 1;
            }
        }
        while (days > 0) {
            days -= 1;
            ++(d);
        }
        return d;
    }
    Date operator -(int days) {
        Date d = *this;
        while (days >= 366) {
            d.year -= 1;
            if (leapyear() && (month == 1 || day < 29))
                days -= 366;
            else
                days -= 365;
        }
        while (days >= 31) {
            d.month -= 1;
            if (month == 0) {
                d.year -= 1;
                d.month = 12;
            }
            if (leapyear())
                days -= day_per_month_leap[month];
            else
                days -= day_per_month[month];

        }
        while (days > 0) {
            days -= 1;
            --(d);
        }
```

```
        return d;
    }
    void printDate() {
        cout << year << "-" << month << "-" << day << endl;
    }
    int countDays() const {
        int i;
        int leap = (year / 4 - year / 100 + year / 400);
        int count = leap * 366 + (year - leap) * 365;
        if (leapyear()) {
            for (i = 1; i < month; i++)
                count += day_per_month_leap[i];
        }
        else {
            for (i = 1; i < month; i++)
                count += day_per_month[i];
        }
        count += day;
        return count;
    }
    int operator -(const Date d) const {
        return (countDays() - d.countDays());
    }
};

int main() {
    Date d1(2020, 2, 29);
    d1.printDate();
    //d1++;
    //d1--;
    //d1.printDate();
    (d1 + 10).printDate();
    (d1 - 30).printDate();
    Date d2(2020, 3, 10);
    int d = d2 - d1;
    cout << d << endl;
    return 0;
}
```

3. String 类包含一个字符数组。对运算符[]进行重载，使得不会发生数组越界。

```
class String {
    int length;
    char* string;
```

```
public:
    String(int len) {
        length = len;
        string = new char[len + 2];
        memset(string, 0, len);
    }
    void inputChar() {
        for (int i = 0; i < length; i++)
            cin >> string[i];
        string[length + 1] = '\0';
        string[length + 2] = '\0';
    }
    char* getChar() {
        return string;
    }
    char& operator [](int i) {
        if (i >= length) {
            cout << "Index out of range!" << endl;
            return string[length + 2];
        }
        else
            return string[i];
    }
};

int main()
{
    String s(10);
    s.inputChar();
    cout << s.getChar() << endl;
    char x = s[20];
    s[20] = 'a';
    cout << s.getChar() << endl;
    return 0;
}
```