运算符重载

> 16 - 18 分

1. 复数相加
2. 点相加
3. 有理数相加

类内函数顺序无区别。

运算符重载：能使得自定义的数据类型实现和普通运算符一样的基本运算。

```
返回类型 operator运算符 (参数列表) {}
```

不能被重载的运算符

```
?:
.
.*
::
sizeof
```

MyPoint例子

```
#include<bits/stdc++.h>
using namespace std;

class MyPoint
{
        int x, y;

    public:
        MyPoint(int x = 0, int y = 0)
        {
            this->x = x;
            this->y = y;
        }

        MyPoint operator+ (MyPoint &b)
        {
            MyPoint tmp;
            tmp.x = x + b.x;
            tmp.y = y + b.y;
            return tmp;
        }
        // 全局函数，不属于类，可以访问私有属性
        friend MyPoint operator- (MyPoint &a, MyPoint &b);
        friend bool operator>= (MyPoint &a, MyPoint &b);
        friend void operator*= (MyPoint &a, MyPoint &b);

        int getX()
        {
```

```
            return x;
        }

        int getY()
        {
            return y;
        }
        // 参数列表带 int 默认后增，反之前增
        const MyPoint operator++(int)
        {
            cout << "后增 post - increment" << endl;
            MyPoint old(x, y);
            x++;
            y++;
            return old;
        }
        // 返回引用避免再次调用拷贝构造函数
        const MyPoint& operator++()
        {
            cout << "前增 pre - increment" << endl;
            ++x;
            ++y;
            return *this;
        }

        friend const MyPoint & operator--(MyPoint &a);
        friend const MyPoint operator--(MyPoint &a, int);

        // 输出符号必须作为友元函数
        friend ostream& operator<<(ostream &, MyPoint &);
        friend istream& operator>>(istream &, MyPoint &);

};
// 两点的分量直接相减
MyPoint operator- (MyPoint &a, MyPoint &b)
{
    MyPoint tmp;
    tmp.x = a.x - b.x;
    tmp.y = a.y - b.y;
    return tmp;
}
// 两点的 x 相比较
bool operator>= (MyPoint &a, MyPoint &b)
{
    return a.x >= b.x;
}
// 两点分量直接相乘
void operator*= (MyPoint &a, MyPoint &b)
{
    a.x *= b.x;
    b.y *= b.y;
//  return a;
}

const MyPoint& operator--(MyPoint &a)
{
```

```
    cout << "pre - decrement" << endl;
    --a.x;
    --a.y;
    return a;
}

const MyPoint operator--(MyPoint &a, int)
{
    cout << "post - decrement" << endl;
    MyPoint old(a.x, a.y);
    a.x--;
    a.y--;
    return old;
}

ostream& operator<<(ostream &out, MyPoint &obj)
{
    out << obj.x << ' ' << obj.y;
    return out;
}

istream& operator>>(istream &in, MyPoint &obj)
{
    in >> obj.x >> obj.y;
    return in;
}

int main()
{
    MyPoint a(1, 2);
    MyPoint b(3, 4);

    MyPoint c = a + b;
    cout << c.getX() << ' ' << c.getY() << endl;

    MyPoint d = a - b;
    cout << d.getX() << ' ' << d.getY() << endl;

    cout << (a >= b) << endl;

    a *= b;
    cout << a.getX() << endl;
    // 前增与后增
    cout << a.getX() << ' ' << a.getY() << endl;
    ++a;
    a++;
    cout << a.getX() << ' ' << a.getY() << endl;
    // 前减与后减
    cout << a.getX() << ' ' << a.getY() << endl;
    --a;
    a--;
    cout << a;

    // 验证输入输出
    MyPoint p4;
    cin >> p4;
```

```
141     cout << p4 << endl;
142 
143     return 0;
144 }
145 
```