

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。

```
1 | 返回类型 operator运算符 (参数列表) {}
```

不能被重载的运算符

```
1 | ?:  
2 | .  
3 | .*  
4 | ::  
5 | sizeof
```

MyPoint例子

```
1 | #include<bits/stdc++.h>  
2 | using namespace std;  
3 |  
4 | class MyPoint  
5 | {  
6 |     int x, y;  
7 |  
8 |     public:  
9 |         MyPoint(int x = 0, int y = 0)  
10 |         {  
11 |             this->x = x;  
12 |             this->y = y;  
13 |         }  
14 |  
15 |         MyPoint operator+ (MyPoint &b)  
16 |         {  
17 |             MyPoint tmp;  
18 |             tmp.x = x + b.x;  
19 |             tmp.y = y + b.y;  
20 |             return tmp;  
21 |         }  
22 |         // 全局函数，不属于类，可以访问私有属性  
23 |         friend MyPoint operator- (MyPoint &a, MyPoint &b);  
24 |         friend bool operator>= (MyPoint &a, MyPoint &b);  
25 |         friend void operator*= (MyPoint &a, MyPoint &b);  
26 |  
27 |         int getX()  
28 |         {
```

```

29         return x;
30     }
31
32     int getY()
33     {
34         return y;
35     }
36     // 参数列表带 int 默认后增，反之前增
37     const MyPoint operator++(int)
38     {
39         cout << "后增 post - increment" << endl;
40         MyPoint old(x, y);
41         x++;
42         y++;
43         return old;
44     }
45     // 返回引用避免再次调用拷贝构造函数
46     const MyPoint& operator++()
47     {
48         cout << "前增 pre - increment" << endl;
49         ++x;
50         ++y;
51         return *this;
52     }
53
54     friend const MyPoint & operator--(MyPoint &a);
55     friend const MyPoint operator--(MyPoint &a, int);
56
57     // 输出符号必须作为友元函数
58     friend ostream& operator<<(ostream &, MyPoint &);
59     friend istream& operator>>(istream &, MyPoint &);
60
61 };
62 // 两点的分量直接相减
63 MyPoint operator- (MyPoint &a, MyPoint &b)
64 {
65     MyPoint tmp;
66     tmp.x = a.x - b.x;
67     tmp.y = a.y - b.y;
68     return tmp;
69 }
70 // 两点的 x 相比较
71 bool operator>= (MyPoint &a, MyPoint &b)
72 {
73     return a.x >= b.x;
74 }
75 // 两点分量直接相乘
76 void operator*= (MyPoint &a, MyPoint &b)
77 {
78     a.x *= b.x;
79     b.y *= b.y;
80     // return a;
81 }
82
83 const MyPoint& operator--(MyPoint &a)
84 {

```

```

85     cout << "pre - decrement" << endl;
86     --a.x;
87     --a.y;
88     return a;
89 }
90
91 const MyPoint operator--(MyPoint &a, int)
92 {
93     cout << "post - decrement" << endl;
94     MyPoint old(a.x, a.y);
95     a.x--;
96     a.y--;
97     return old;
98 }
99
100 ostream& operator<<(ostream &out, MyPoint &obj)
101 {
102     out << obj.x << ' ' << obj.y;
103     return out;
104 }
105
106 istream& operator>>(istream &in, MyPoint &obj)
107 {
108     in >> obj.x >> obj.y;
109     return in;
110 }
111
112 int main()
113 {
114     MyPoint a(1, 2);
115     MyPoint b(3, 4);
116
117     MyPoint c = a + b;
118     cout << c.getX() << ' ' << c.getY() << endl;
119
120     MyPoint d = a - b;
121     cout << d.getX() << ' ' << d.getY() << endl;
122
123     cout << (a >= b) << endl;
124
125     a *= b;
126     cout << a.getX() << endl;
127     // 前增与后增
128     cout << a.getX() << ' ' << a.getY() << endl;
129     ++a;
130     a++;
131     cout << a.getX() << ' ' << a.getY() << endl;
132     // 前减与后减
133     cout << a.getX() << ' ' << a.getY() << endl;
134     --a;
135     a--;
136     cout << a;
137
138     // 验证输入输出
139     MyPoint p4;
140     cin >> p4;

```

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