

# Working with Classes

---

【目标】：

## 运算符重载

---

数学符号一样的调用方式

std::cout, std::cin

## 友元函数

---

对某些特定的情况（类或者函数），放开对类成员的访问控制

```
1  class Vector2D
2  {
3  private:
4      double x;
5      double y;
6  };
7
8  void Test()
9  {
10     Vector2D v1(1.0, 1.0), v2(2.0, 2.0), v3;
11
12     std::cin >> v1;
13
14     v3 = v1.Sum(v2);
15     Vector2D v4 = v1 + v2;
16     Vector2D v5 = v1 * 4.0;
17     Vector2D v6 = 4.0 * v1;
18
19     (std::cout << v6) << v5 << std::endl;
20
21     double x, y, z;
22     z = (y = (x = 0));
23     v6 = v5 = v4;
24 }
```

解决方案

```
1  class Vector2D
2  {
3  private:
4      double x;
5      double y;
6  public:
7      Vector2D()
8      {
9          x = y = 0;
```

```

10     }
11
12     Vector2D(double x, double y)
13     {
14         this->x = x;
15         this->y = y;
16     }
17
18     void Show()const
19     {
20         std::cout << "<" << x << ", " << y << ">" << std::endl;
21     }
22
23     Vector2D Sum(const Vector2D& v)const
24     {
25         Vector2D res;
26         res.x = this->x + v.x;
27         res.y = this->y + v.y;
28         return res;
29         //return Vector2D(this->x + v.x, this->y + v.y);
30     }
31
32     Vector2D operator+(const Vector2D& v)const
33     {
34         return Vector2D(this->x + v.x, this->y + v.y);
35     }
36
37     Vector2D operator*(double m)const
38     {
39         return Vector2D(this->x * m, this->y * m);
40     }
41
42     friend Vector2D operator*(double m, const Vector2D& v);
43     friend std::ostream& operator<<(std::ostream& os, const Vector2D& t);
44     friend std::istream& operator>>(std::istream& is, Vector2D& t);
45 };
46
47 Vector2D operator*(double m, const Vector2D& v)
48 {
49     return Vector2D(v.x * m, v.y * m);
50 }
51
52 std::ostream & operator<<(std::ostream& os, const Vector2D& t)
53 {
54     os << "<" << t.x << ", " << t.y << ">" << std::endl;
55     return os;
56 }
57
58 std::istream& operator>>(std::istream& is, Vector2D& t)
59 {
60     is >> t.x >> t.y;
61     return is;

```

```

62 }
63
64 void Test()
65 {
66     Vector2D v1(1.0, 1.0), v2(2.0, 2.0), v3;
67     std::cin >> v1;
68     v3 = v1.Sum(v2);
69     Vector2D v4 = v1 + v2;
70     Vector2D v5 = v1 * 4.0;
71     Vector2D v6 = 4.0 * v1;
72     (std::cout << v6) << v5 << std::endl;
73
74     double x, y, z;
75     z = (y = (x = 0));
76 }

```

特别的

```

1  class UPInt { // "unlimited precision int"
2  public:
3      UPInt& operator++(); // ++ 前缀
4      const UPInt operator++(int); // ++ 后缀
5
6      UPInt& operator+=(int); // += 操作符, UPInts
7  private:
8
9  };
10
11 // 前缀形式: 增加然后取回值
12
13 UPInt& UPInt::operator++()
14 {
15     *this += 1; // 增加
16     return *this; // 取回值
17 }
18
19 // postfix form: fetch and increment
20
21 const UPInt UPInt::operator++(int)
22 {
23     UPInt oldValue = *this; // 取回值
24     ++(*this); // 增加
25     return oldValue; // 返回被取回的值
26 }
27
28 UPInt i;
29
30 ++i; // 调用 i.operator++();
31 i++; // 调用 i.operator++(0);

```