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

```
class Vector2D
{
private:
    double x;
    double y;
};

void Test()
{
    Vector2D v1(1.0, 1.0), v2(2.0, 2.0), v3;

    std::cin >> v1;

    v3 = v1.Sum(v2);
    Vector2D v4 = v1 + v2;
    Vector2D v5 = v1 * 4.0;
    Vector2D v6 = 4.0 * v1;

    (std::cout << v6) << v5 << std::endl;

    double x, y, z;
    z = (y = (x = 0));
    v6 = v5 = v4;
}
```

解决方案

```
class Vector2D
{
private:
    double x;
    double y;
public:
    Vector2D()
    {
        x = y = 0;
```

```
    }

    Vector2D(double x, double y)
    {
        this->x = x;
        this->y = y;
    }

    void Show()const
    {
        std::cout << "<" << x << ", " << y << ">" << std::endl;
    }

    Vector2D Sum(const Vector2D& v)const
    {
        Vector2D res;
        res.x = this->x + v.x;
        res.y = this->y + v.y;
        return res;
        //return Vector2D(this->x + v.x, this->y + v.y);
    }

    Vector2D operator+(const Vector2D& v)const
    {
        return Vector2D(this->x + v.x, this->y + v.y);
    }

    Vector2D operator*(double m)const
    {
        return Vector2D(this->x * m, this->y * m);
    }

    friend Vector2D operator*(double m, const Vector2D& v);
    friend std::ostream& operator<<(std::ostream& os, const Vector2D& t);
    friend std::istream& operator>>(std::istream& is, Vector2D& t);
};

Vector2D operator*(double m, const Vector2D& v)
{
    return Vector2D(v.x * m, v.y * m);
}

std::ostream & operator<<(std::ostream& os, const Vector2D& t)
{
    os << "<" << t.x << ", " << t.y << ">" << std::endl;
    return os;
}

std::istream& operator>>(std::istream& is, Vector2D& t)
{
    is >> t.x >> t.y;
    return is;
```

```
}

void Test()
{
    Vector2D v1(1.0, 1.0), v2(2.0, 2.0), v3;
    std::cin >> v1;
    v3 = v1.Sum(v2);
    Vector2D v4 = v1 + v2;
    Vector2D v5 = v1 * 4.0;
    Vector2D v6 = 4.0 * v1;
    (std::cout << v6) << v5 << std::endl;

    double x, y, z;
    z = (y = (x = 0));
}
```

特别的

```
class UPInt { // "unlimited precision int"
public:
  UPInt& operator++(); // ++ 前缀
  const UPInt operator++(int); // ++ 后缀

  UPInt& operator+=(int); // += 操作符，UPInts
private:

};

// 前缀形式：增加然后取回值

UPInt& UPInt::operator++()
{
  *this += 1; // 增加
  return *this; // 取回值
}

// postfix form: fetch and increment

const UPInt UPInt::operator++(int)
{
  UPInt oldValue = *this; // 取回值
  ++(*this); // 增加
  return oldValue; // 返回被取回的值
}

UPInt i;

++i; // 调用 i.operator++();
i++; // 调用 i.operator++(0);
```