面向对象方法与C++程序设计

第3章　　运算符重载

大连理工大学
主讲人-赵小薇

理解运算符重载



- 使用运算符对各种数据进行操作，高效快捷且清晰易懂。
- 如果运算符也能作用于各种类对象，会使代码简洁而优雅。

```
v1 = v1+ v2 ;    //向量加法
M1 = M1+ M2 ;  //矩阵相加
v1 = M1*v2 ;    // 向量与矩阵相乘
M1 = v1*v2 ;    // 向量相乘
int day = d1- d2 ;  // 计算日期差
double d = p1 – p2 ;// 计算两点距离
```

举例



- 完成复数类，该类能够完成复数的加法。

```
class Complex
{ public:
        Complex(double r, double i){ d                    =i;}
        Complex complexAdd(Comple
   private:
        double dReal, dImag;
};
Complex Complex :: complexAdd(Complex &c2)
 {      Complex c;
        c.dReal = dReal+c2.dReal;
        c.dImag = dImag+c2.dImag;
        return c;
 }
```

```
c = c1.add( c2 );
```



```
class Complex
{ public:
        Complex(double r, double i){ dReal=r;  dImag=i;}
         Complex operator + (Complex &c2);
   private:
        double dReal, dImag;
};
```

c = c1 + c2; // c1.operator+(c2)

```
Complex Complex::operator + (Complex &c2)
{       Complex c;
        c.dReal = dReal+c2.dReal;
        c.dImag = dImag+c2.dImag;
        return c;
// return   Complex ( dReal+c2.dReal, dImag+c2.dImag);
}
```

运算符重载规则



- 重载运算符函数可以对运算符作出新的解释，原有基本语义不变：
- 不能创建新的运算符
- 不能改变运算符的优先级和结合性
- 不能改变运算符所需要的操作数
- 不能改变该运算符用于基本类型变量时的含义

运算符重载规则



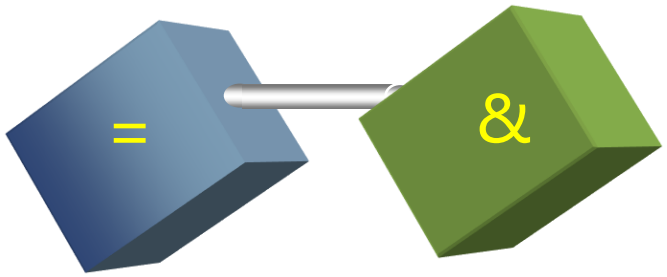
- 可以重载大部分已有的运算符

特殊的运算符不允许被重载

不能重载的算符

. :: .* (.->) ? : sizeof #

不必重载的算符



运算符重载方法



1 成员函数

2 友元函数

3 普通函数

重载方法

成员函数实现运算符重载



- 成员函数的语法形式

函数返回类型

函数名

```
类型  类名 :: operator op ( 参数表 )
{
    // 相对于该类定义的操作
}
```

重载该运算符的类

举例



```
class Complex
{
        double dReal, dImag;
 public:
        Complex(double r, double i);
        Complex operator + (Complex &c2);
        Complex operator + (double d);
} ;
```

```
Complex  z ( 2.0 , 3 ) ,   k ( 3 , 4.5 ) ;
z = z  + k;
k = z + 27 ;
k = 27 + z ;
```

z . operator + (k)

z . operator + (27)

?

普通函数实现运算符重载

```
Complex operator + ( int & a ,Complex &c2)
{  Complex c;
   c.dReal = a+c2.dReal;
   c.dImag = 0+c2.dImag;
   return c;
}
```

```
k = 27 + z ;
```

k=operator + (27, z)