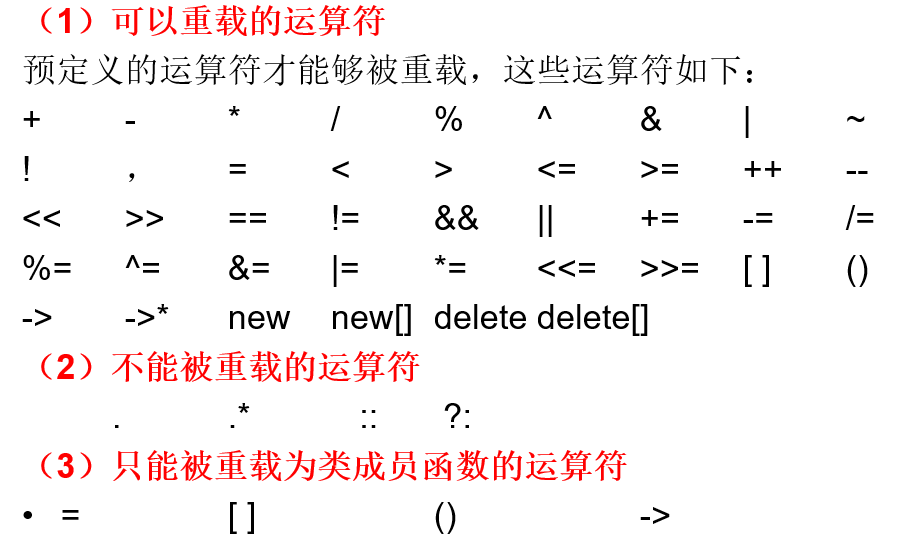
**运算符重载**

* 通过重载，可以**扩展C++运算符的功能**，使它们**能够操作用户自定义的数据类型**，增加程序代码的直观性和可读性。
  + - **int i=2+3; int operator+(int,int);**
    - **double j=2+4.8; double operator+(int,double);**
    - **float f=float(3.1)+float(2.0); float operator+(float,float);**



不能改变运算符的优先级。

② 不能改变运算符的结合顺序（如+、-、\*、/ 等运算符按照从左到右结合，这个顺序不能改变）。

③ 重载运算符不能使用默认参数。

④ 不能改变运算符所需要的参数个数。

⑤ 不能创造新运算符，只能重载系统已有的运算符。

⑥ 不能改变运算符的原有含义。

⑦ 若运算符被重载为类的成员函数，则只能是非静态的成员函数。

**返回类型 operator@(参数表)**

类运算符重载方式

* + - 重载为类的非静态成员函数
    - 重载为类的友元函数
    - 重载为普通函数

运算符重载为类成员函数时，其第1个参数是由编译器通过this指针隐式传递，因此其参数个数要比**该运算符实际的参数个数少一个**

**Complex operator+(Complex x){……}**

* 1. 算术运算（+、\*、/、-等）、相等与否的比较、关系运算、位运算等运算符具有对称性，通常允许运算符左、右两边的对象进行交换或类型转换，则适宜重载为非成员函数。
  2. ++（自增）、--（自减）、解引用运算符，也适宜重载为类成员函数

**调用类的重载运算符时，作为类成员函数运算符的左参数必须是一个类对象**

a @ b; //隐式调用二元运算符@

a.operator@(b) //显式调用二元运算符@

**Counter& operator++();**

**Counter& Counter::operator++() {**

**++n;**

**return \*this;**

**}**

**Counter Counter::operator++(int) {**

**Counter t(\*this);**

**n++;**

**return t;**

**}**

Complex c1,c2(1,2);

c1=c2+2; //L1

c1.display();

c1=2+c2; //L2

只能完成L1语句的运算，L2语句则不能实现。

* + **对于非类成员的重载运算符函数，C++编译器在参数不匹配的情况下将对第一个参数进行隐式类型转换；但不会对类成员运算符函数的第一个参数进行任何隐式类型转换。**

通过1个友元函数重载对两个Complex类型相加的加法运算符函数。

friend Complex **operator+**(**Complex** a, **Complex** b)

{ return Complex(a.r + b.r, a.i+b.i); }

这种方案要求：类应具有能够接受一个参数的构造函数，此构造函数具有将此参数转换为类类型的能力！

* **一元运算符作为类成员函数重载时不需要参数**
* **像++、--这样能够实现连续自增、自减的运算符，其重载函数应该返回对象的引用。否则，就不能实现对象的连续运算。**
* **用友元函数重载一元运算符时需要一个参数**

在用友元和普通函数重载++、--这类一元运算符函数时，**不能把运算结果返回给调用对象**

* **C++编译器通过在运算符函数参数表中是否插入关键字int 来区分这两种方式。**
* 自减前缀

**operator -- ();**

**operator -- (X & x);**

* 自减后缀：**加入一个无用的类型参数，表示后缀**

**operator -- (int);**

**operator -- (X & x, int);**

**重载下标运算符[]**

**通过下标运算符[]重载，可以在访问数组元素题进行下标值检测，禁止越界访问**

**class X{**

**……**

**X& operator[](int n);**

**}；**

**①** [ ]是一个二元运算符，其第1个参数是通过对象的this指针传递的，第2个参数代表数组的下标

② 由于[ ]既可以出现在赋值符“=”的左边，也可以出现在赋值符“=”的右边，所以重载运算符[ ]时常返回引用。

③ [ ]只能被重载为类的非静态成员函数，不能被重载为友元和普通函数。

如果**对象构造时分配了动态存储空间**，默认赋值运算符函数多数时候都不能正确地进行对象的赋值拷贝，有时需要通过赋值运算符**实现特殊的对象赋值拷贝操作**，需要为类重载赋值运算符函数。

**类型转换**

**用于将类X转换为类Y类型。语法**

**class X**

**{**

**public: operator Y ( )**

**{ ……**

**return Y类型的数据；};**

**};**

* + 类型转换函数**没有参数，没有返回类型。**
  + 类型转换函数**必须返回将要转换成的type类型数据。**
  + **一旦定义类型转换运算符，就可以显示或隐式地进行类型转换，如同系统预定义的类型转换一样。**

operator float(){return (float)3.14\*r\*r;}

float area=c; //调用operator float()，将Circle类型转换成float

**函数调用运算符重载**

**运算符( )是函数调用运算符，也能被重载**

**class X{**

**……**

**X& operator( )(参数表);**

**}；**

**其中的参数表可以包括任意多个参数。**

**Obj(参数表); //调用形式**

<<的重载语法

ostream &operator<<(ostream &os,classType object) {

……

os<< … //输出对象的实际成员数据

return os; //返回ostream对象

}

>>的重载语法

istream &operator>>(istream &is,class\_name &object)

{

……

is>> … //输入对象object的实际成员数据

return is; //返回istream对象

}