# 第四章 运算符重载

**written by SCNU第一帅----->孔文康**



## 对运算符重载的方法

运算符重载的方法是定义一个重载运算符的函数，使指定的运算符不仅能实现原有的功能，而且能实现在函数中指定的新的功能。

运算符的重载实际上就是函数的重载。
重载运算符的一般格式如下

函数类型 operator 运算符名称(形参表) { 对运算符的重载 }

例如，如果想将+用于Complex类的加法运算

```
Complex operator +(Complex & c1, Complex & c2);
```

**重载运算符的规则**

1. C++不允许用户自定义新的运算符，只能对已有的运算符进行重载。

2. C++允许重载的运算符如下表

| | |
|---|---|
| 双目算数运算符 | +,-,*,/,% |
| 关系运算符 | ==,!=,<,>,<=,>= |
| 逻辑运算符 | \|\|,&&,! |
| 单目运算符 | +,-,*,& |
| 位运算符 | \|,&,~,^,<<,>> |
| 赋值运算符 | =,+=,-=,*=,/=,%=,&=,\|=,^=,<<=,>>= |
| 空间申请与释放 | new,delete,new[],delete[] |
| 其他运算符 | (),->,->*,[], |

不能重载的运算符只有5个：

1. 成员访问运算符 (.)
2. 成员指针访问运算符(*)
3. 域运算符(::)
4. 长度运算符(sizeof)
5. 条件运算符(?:)

重载不能改变运算符的运算对象（操作数）的个数

重载不能改变运算符的优先级别

重载不能改变运算符的结合性

重载运算符的函数不能有默认的参数

重载的运算符必须和用户定义的自定义类型的对象一起使用，其参数应该至少有一个是类对象。

**运算符重载函数作为类成员函数和友元函数**

有的读者可能对程序中运算符重载的函数提出问题：

+是双目运算符，为什么重载函数中只有一个参数呢?

实际上，运算符重载函数应当有两个参数，但是，由于重载函数是Complex类中的成员函数，因此有一个参数是隐含的，运算符是用this指针隐式地访问类对象的数据成员。

在运算符函数重载为成员函数后，如果出现该运算符的表达式，如C1+C2，编译系统会把他解释为

```
c1.operator+(c2);
```

即通过对象c1调用运算符重载函数operator+。

但是如果放在类外，以友元函数形式定义，即需要显示两个参数，如。

```
class Complex {
 public:
 friend Complex operator+(Complex &c1, Complex &c2);
}
Complex operator+(Complex &c1, Complex& c2) {
 return Complex(.....)
```

```
}
```

**重载流插入运算符和流提取运算符**

C++的流插入运算符<<和流提取运算符>>是C++编译系统在类库中提供的。

所有C++编译系统都在其类库中提供输入流类istream和输入流类ostream。

cin和cout分别是istream类和ostream类的对象。

C++系统已经对<<和>>进行重载，使之能作为流插入运算符和流提取运算符，用来输出C++标准型数据。

用户自己定义的数据类型（如类对象）是不能直接<<和>>输入和输出的。

对<<和>>重载的函数形式如下

istream & operator >>(istream &, 自定义类 &);
ostream & operator <<(ostream &, 自定义类 &);

重载运算符>>第一个参数和函数类型都必须是istream &类型。
重载运算符<<第一个参数和函数类型都必须是ostream &类型。

```
class Complex {
 public:
 friend ostream& operator <<(ostream &, Complex &);
 friend istream& operator >>(istream &, Complex &);
 private:
 double real;
 double imag;
```

```
}
ostream & operator <<(ostream & output, Complex & c) {
 output<<"("<<c.real<<"+"<<c.imag<<"i"<<endl;
 return output;
}
istream & operator >>(istream & input, Complex & c) {
 cout<<"input the part and imaginary part of complex number:"<<endl;
 input>>c.real>>c.imag;
 return input;
}
```

## 不同类型数据间的转换

### 标准类型数据间的转换

C++中，不同类型数据之间可以自动转换，例如

```
int i = 6; //i为整型
i = 7.5 + i;
```

系统对7.5是作为double型数据处理的，在求解表达式时，先将i的值6转换为double型，然后与7.5相加，得到和为13.5，最后向整型i赋值，将13.5转为13。

这种转换称为隐式类型转换。

C++还提供显示类型转换。

类型名 (数据)

```
int(89.5);
```

### 用转换构造函数进行不同类型数据的转换

转换构造函数的作用是将一个其他类型的数据转换成一个类型的对象。

转换构造函数只有一个形参。

```
Complex (double r) {
 real = r;
 imag = 0;
}
```

**类型转换函数**

类型转换函数是将一个类的对象转换成另一类型的数据。

如果已经声明了一个Complex类，可以在Complex类中这样定义类型转换函数。

```
operator double() {
 return real;
}
```