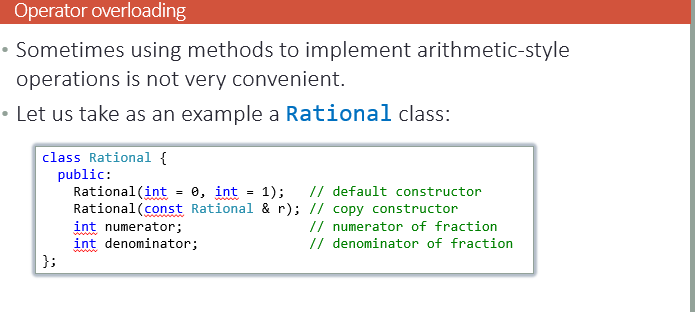
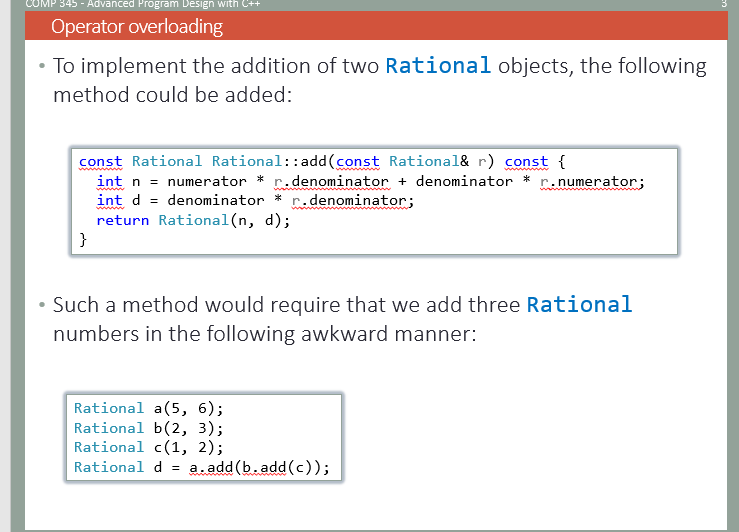
有的时候使用method来实装一个数学操作符不是特别方便

以rational class为例子 //rational有理数



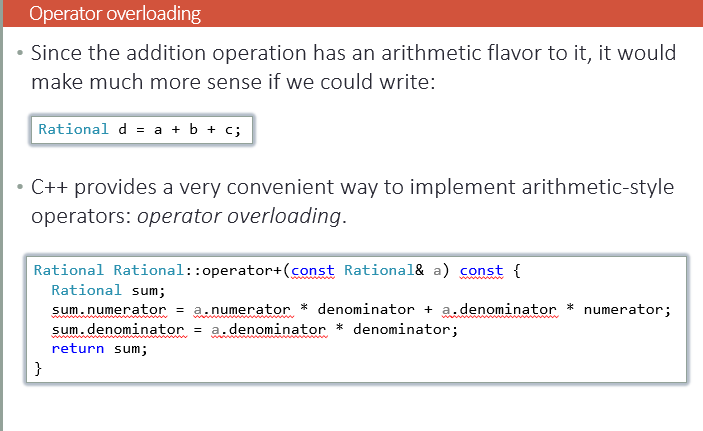
numerator分子

denominator分母



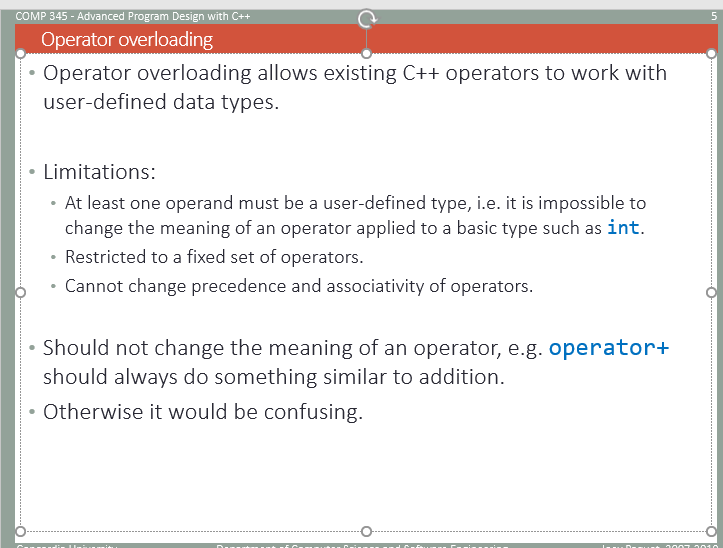
为了add两个rationalobject，可能需要addfunction

然而，当我们增加到三个以后，add会看起来很怪



用纯数学符号更好

C++提供了一种方便的方法直接overload 具体operator



operator overloading可以让 加减乘除这种 operator对用户类型野生小

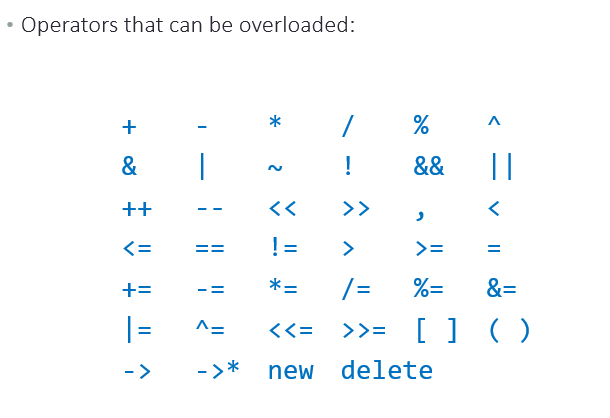
限制：

Operator左右两边至少有一个operand 是用户定义类型，你无法改变int之间的operator

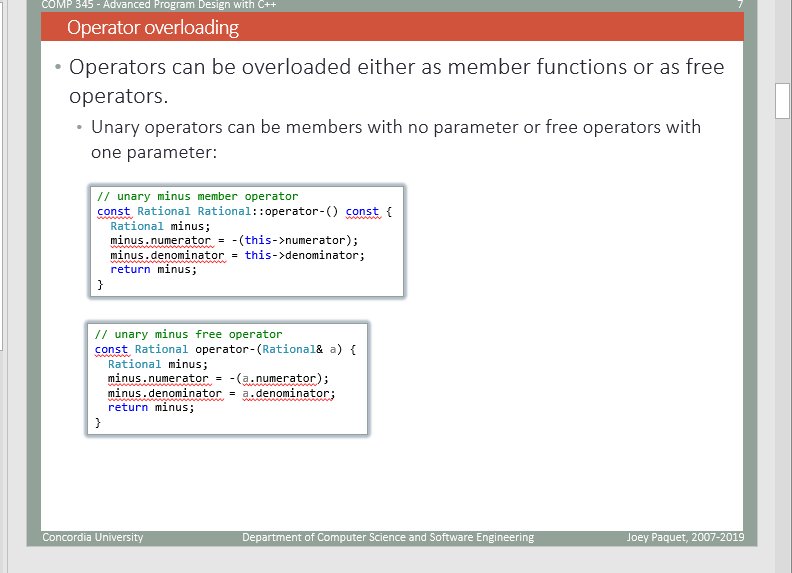
只有固定几个operator能改

不能改变operator的优先级

可以但不应该改变operator的意思，例如加号永远代表某一类加号



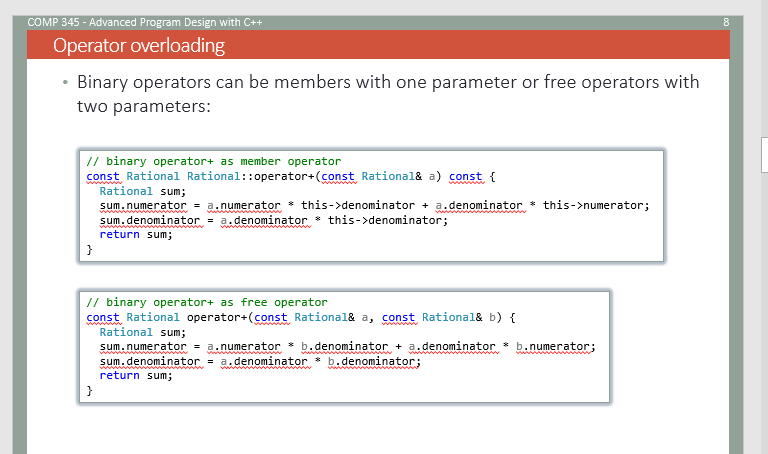
operator可以作为member function或 free operator被overload



free operator要价格rational参数

unary operator且是member function，可以不要参数

const returnType(className) className::operator-() const

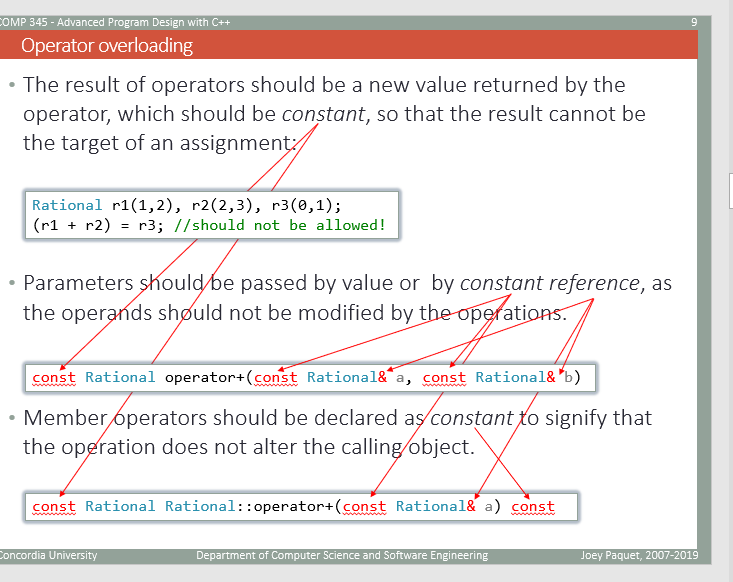


binary operator

member要一个参数

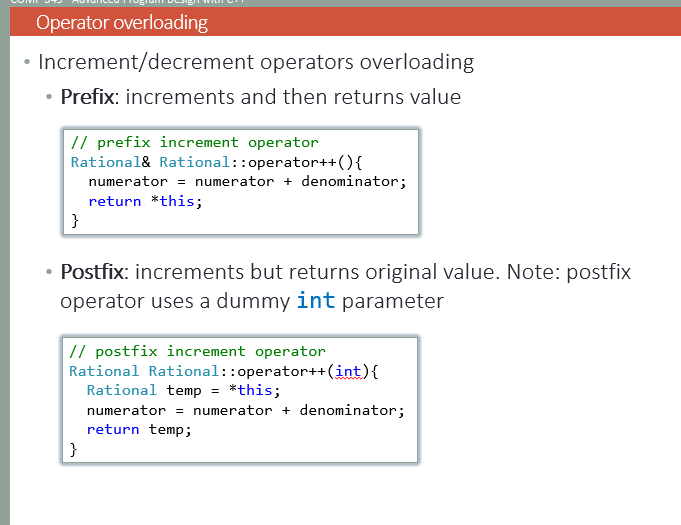
free要两个参数

const Rational Rational::operator +(const Rational& a)const



我们不能给operator的结果赋值

因为operator是const，所以return的值也是常量无法改变



自增：

前者是前缀自增

++A

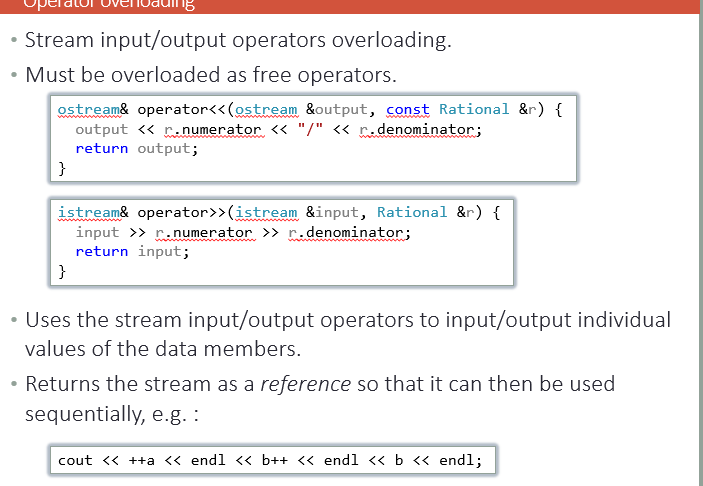
会让数据变化

后者是

A++

数据会加一，但是return的是之前的数据

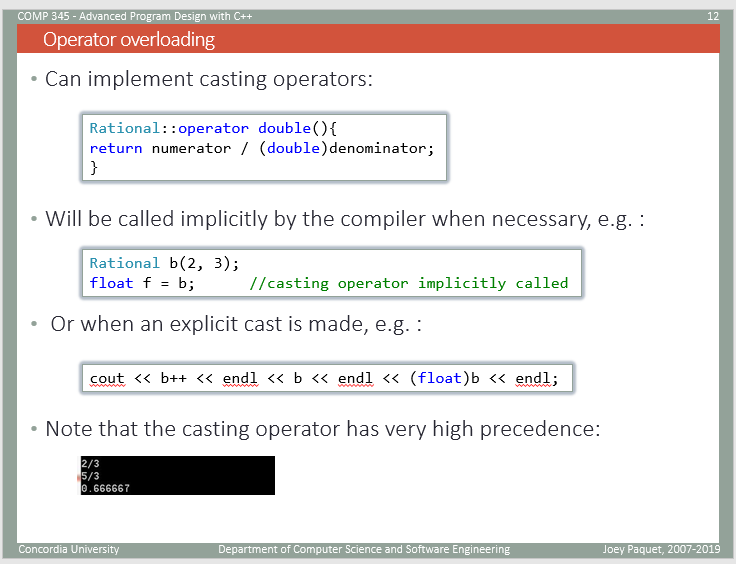
后者有一个虚假的Paramter



stream input output

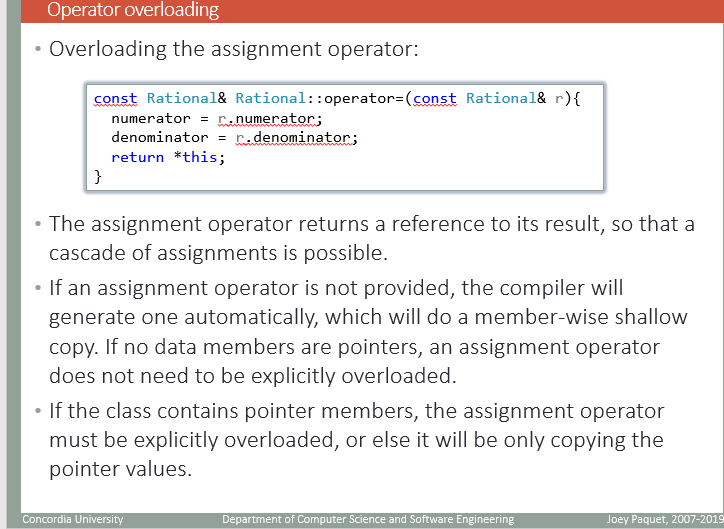
必须作为free oprator被overload

ostream& operator<<()



还可以添加

cast 这种operation



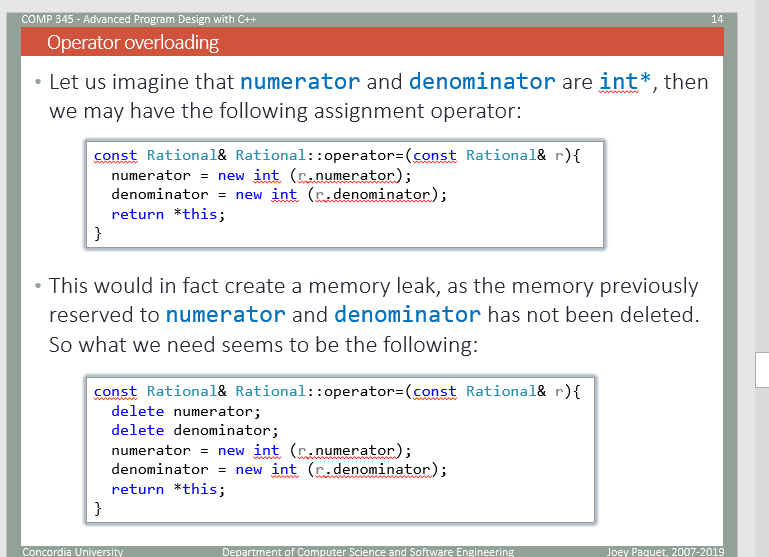
也可以overload

等于符号

如果你没写等于符号

会自动生成一个shallow copy  
如果没有pointer型数据，没必要overload

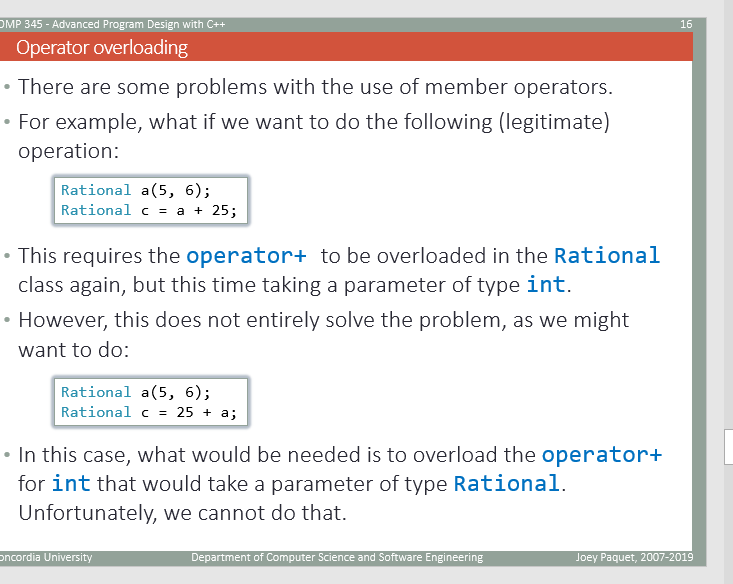
如果有pointer数据，必须overload



让它们变成pointer

那么等于符号就要先delete两个pointer （被指向的数据）

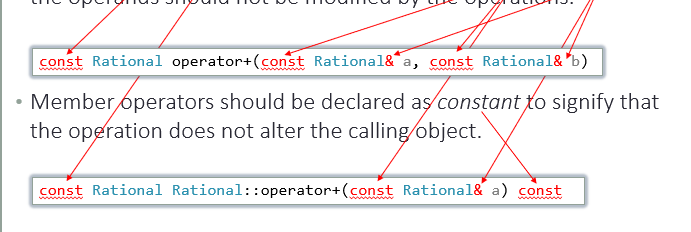
然后赋值



但是member operator是有一些问题的

例如我们想干这个

我们需要overload Rational class again，只是这次参数应该是Int



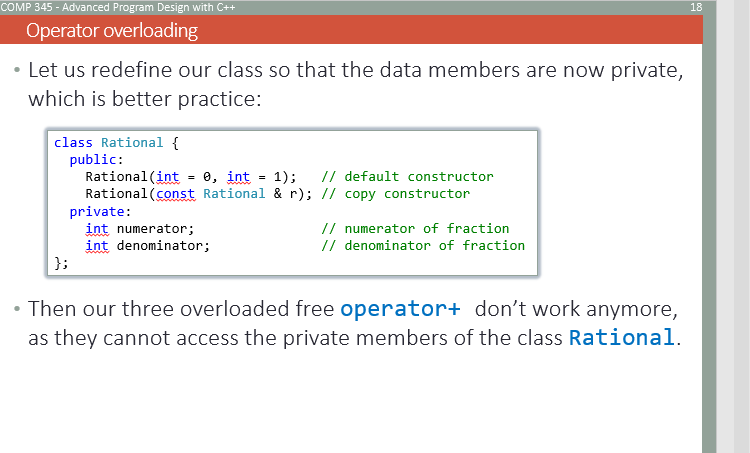
然而这并不一劳永逸

我们也无法调换过来因为我们无法定义Int等基本型

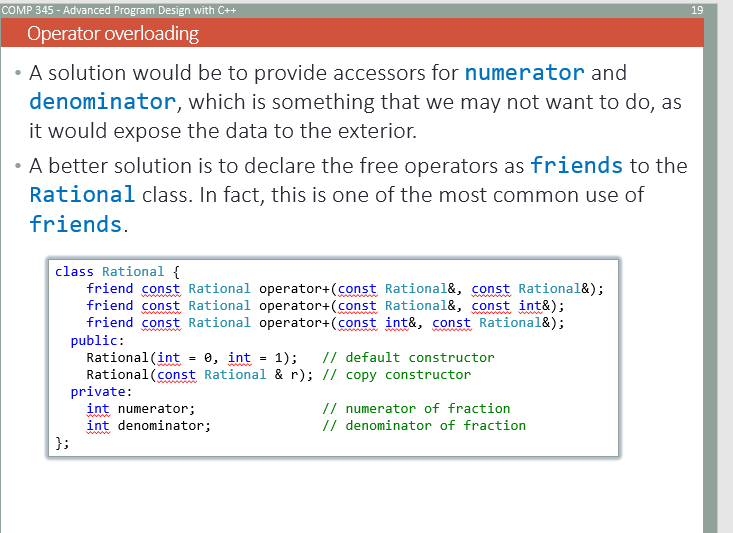


所以最好的就是free operator

根据需求无限定义



这时在外面overload就不管用了，因为Numerator与denominator都是private



应该在public之前给他们friend权限