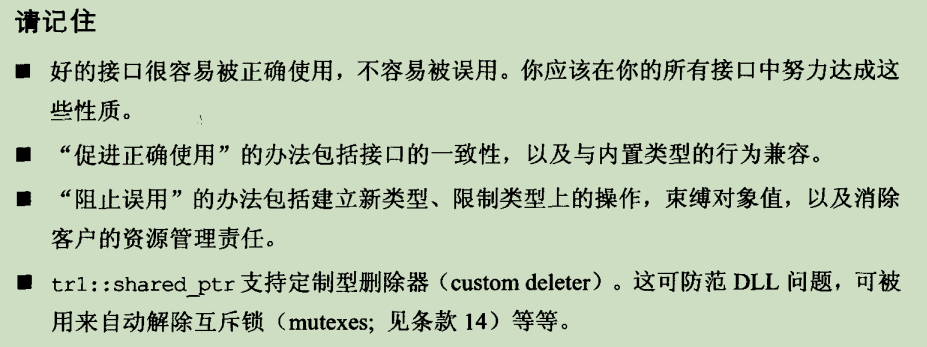
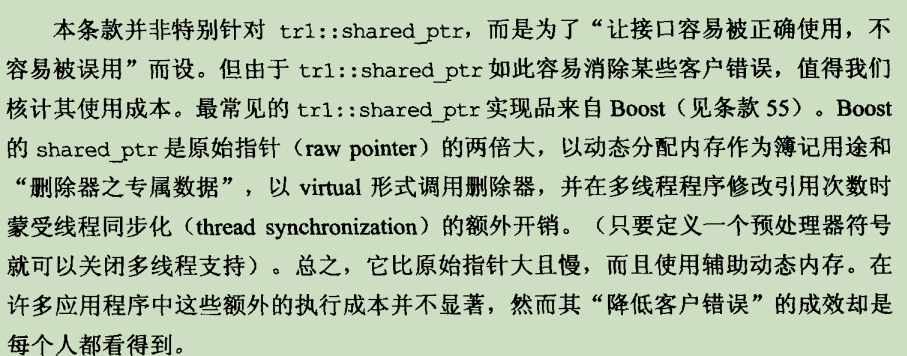
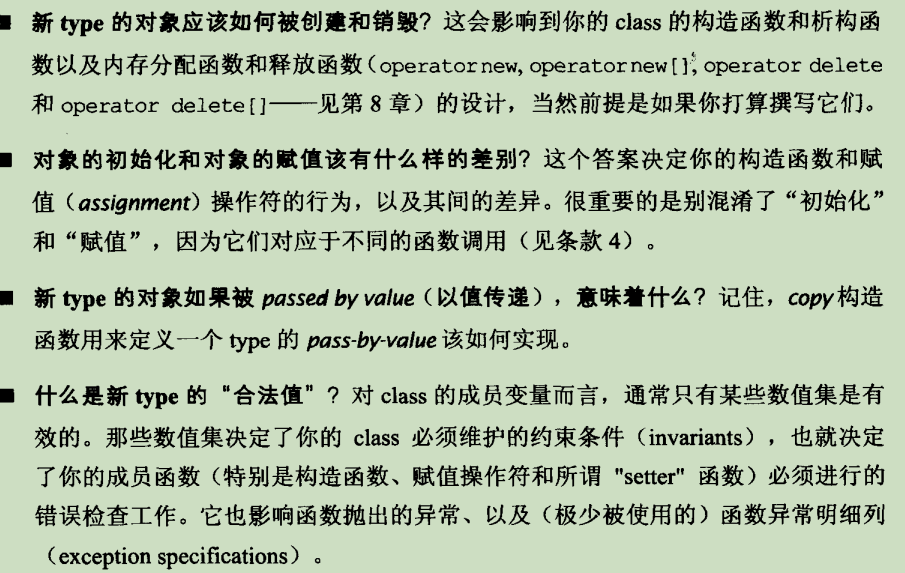


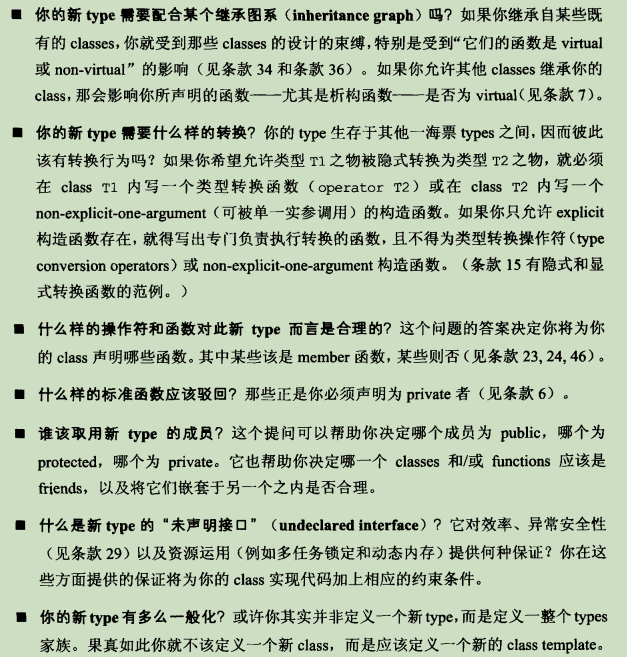
**条款18**

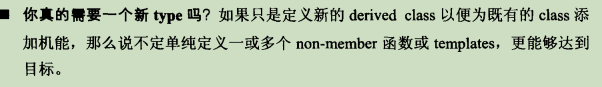




**条款19**

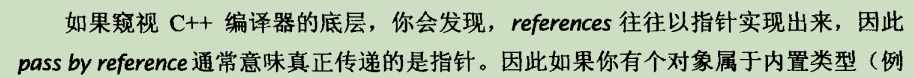


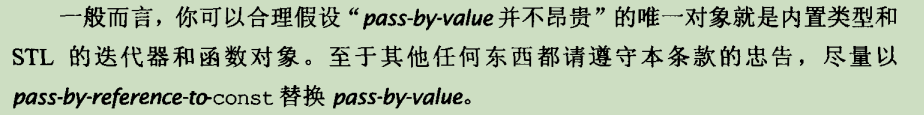




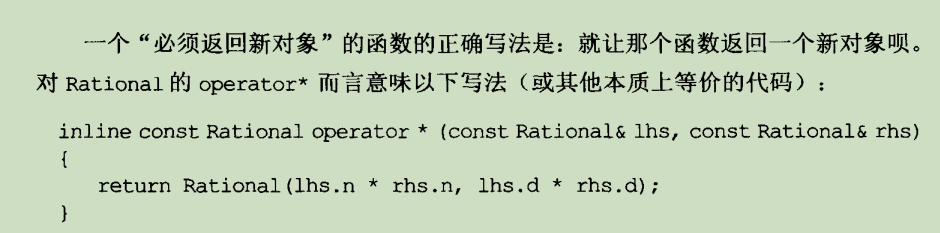
**条款20**

如果是值传递，对于我们自己编写的类，往往会对类中的成员变量，进行构造，析构，成员变量越多，构造和析构的次数越多，效率越低，所以推荐by reference的操作。





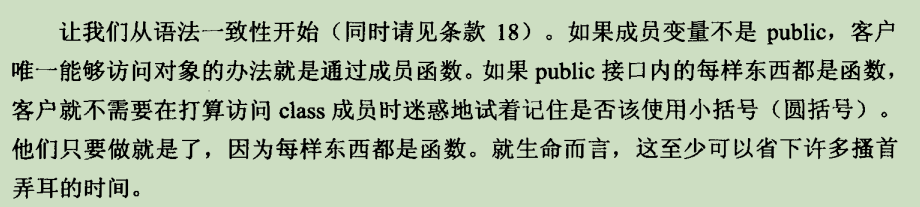
**条款21**



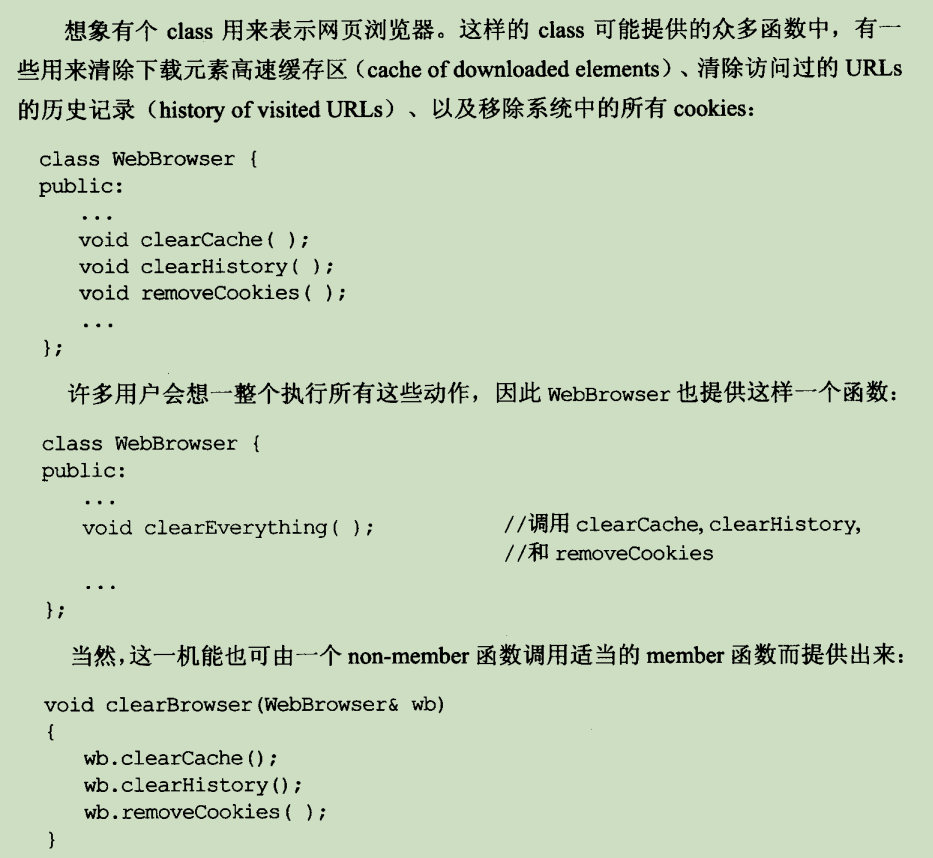
这个函数让两个参数乘积，然后返回新的rational 变量，这种时候就可以返回一个新的对象。

**条款22 为什么成员变量要声明成private**

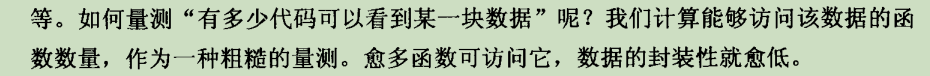
1.



**条款23**



**我们这里提出一个概论**

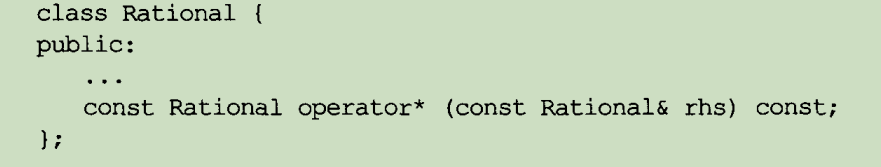


**这里的数据 代表的是private的成员**

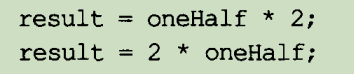
如果是non-member函数，不会增加private函数访问数量，如果是成员函数就会增加。所以对于上面的这种可以写成non-member函数，优先考虑。

**条款24**

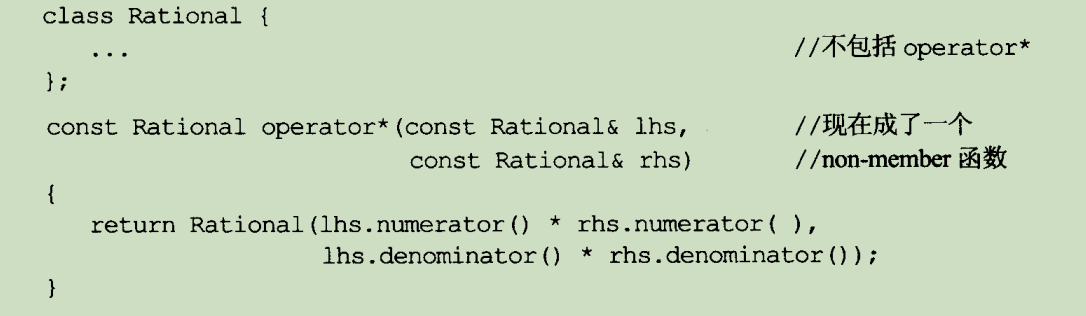
explicit 明确的，如果类中没有写explicit，也就是no-explict，就代表他支持隐士转换，



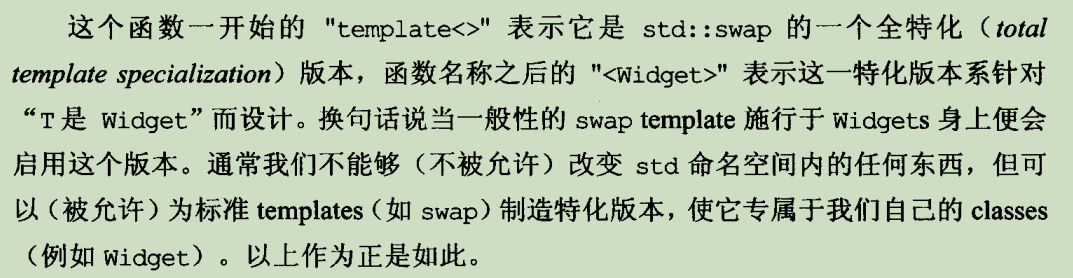
第一项没有问题，第二行有问题在于int没有实现乘法，所以



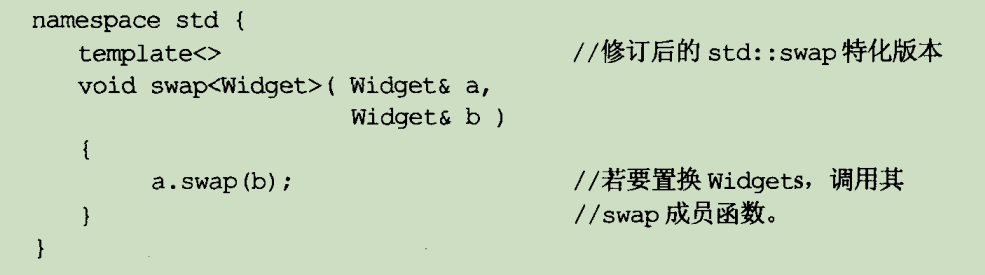
对于可以进行交换的乘法，需要隐士转换，就采用no-member函数的显示转换。



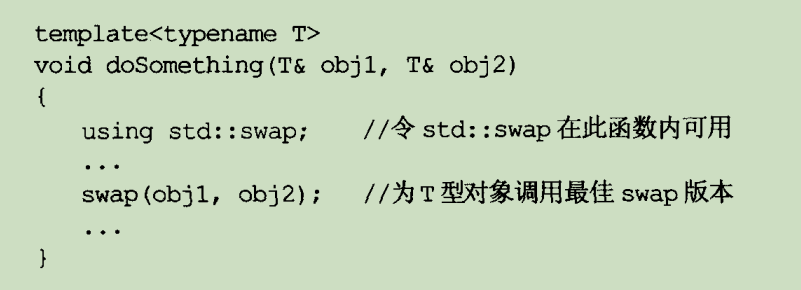
**条款25**



**这种特例化手法值得学习**



**看清楚下面的写法，有可能这个swap函数是非成员函数，这个函数效率跟高，所以不直接写std：：swap这种写法就是不固定死swap的调用。**



**总结**

