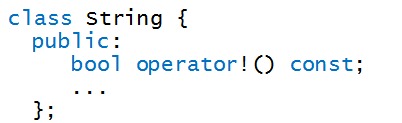
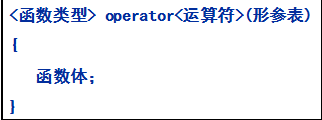
一元运算符重载

尽量避免用友元，除非必要

作为类成员函数：



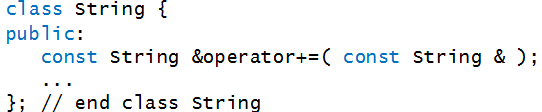
语法



赋值必须是成员函数：友元没有this

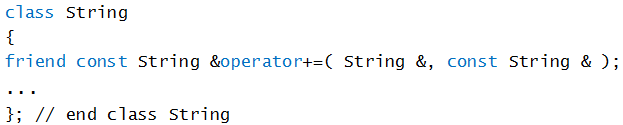
二元运算符重载

非静态成员函数：一个参数 【额，应该不会有重载是写静态成员函数的吧？】

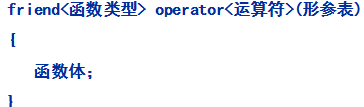


与相等

非成员函数(友元)：两个参数



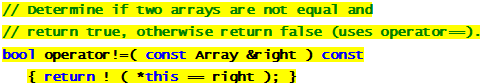
语法



与相同

an array class（数组类）

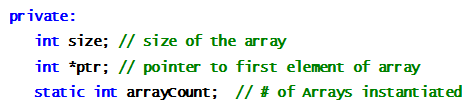
第25页： ！=



【前面还有一个==的bool，那个this指针是指向自身的对象，然后用==函数判断是否与传入的right对象相等，最后返回==的布尔值的非】

这里你要理解一点…它用的是\*this，而不是直接用this，this是一个指针，\*this是一个Array对象，对于这个对象使用==运算符，就要去寻找它的重载…所以，这里是利用了前面的==的重载，返回值是==的取反。

第26页



【静态数据成员arrayCount为计数器，用来计数类实例化的对象】

看构造函数的实现，

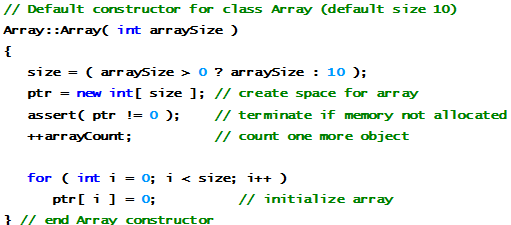
它这里已经说了…这个是计数器，存储当前有的对象个数。

再看析构函数，看它有没有减回去：



这样子就很明显看出来是计数器了…

第28页：关于assert



【我真的有问过度娘TAT，assert是“断言”。这里是如果ptr！=0返回条件为错误，就不执行下面的了？这里加这个的意义是？】

assert是一个函数，如果括号里面的值为0，则程序会停止，并输出错误信息。

这里括号内是一个判断语句，如果ptr = 0的话，那么括号内的ptr ！= 0的返回值是false，就是0，所以就要终止，反之则不会终止。

类型转换

char与int可以互相转换：占用空间一样

转换操作符必须是一个非静态成员函数。

不能是一个友元函数

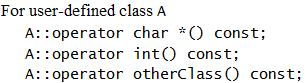
形参表必须为空

不能指定返回类型，但是必须显式返回一个指定类型的值【没懂……】

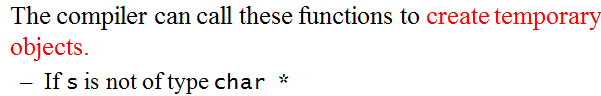
形式：

type 表示内置类型名、类类型名或由类型别名定义的名字，对任何可作为函数返回类型的类型（除了 void 之外）都可以定义转换函数。

在类里面定义：



使用的时候其实就是调用



这个或许这样说你会比较好理解吧…

Int a = 10;

double d = (double) a / 3;

这里(double) a就是调用了类型转换操作符…然后生成了一个临时的对象来存储这个double类型的变量，它的值正好等于a，要注意a还是整型的，a是不变的，所以才需要一个临时空间来存放那个double类型的对象。

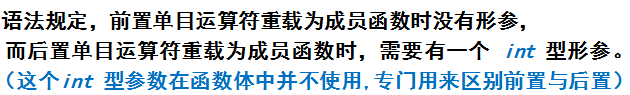
重载++和--

前置：（和其他一元运算符重载一样）

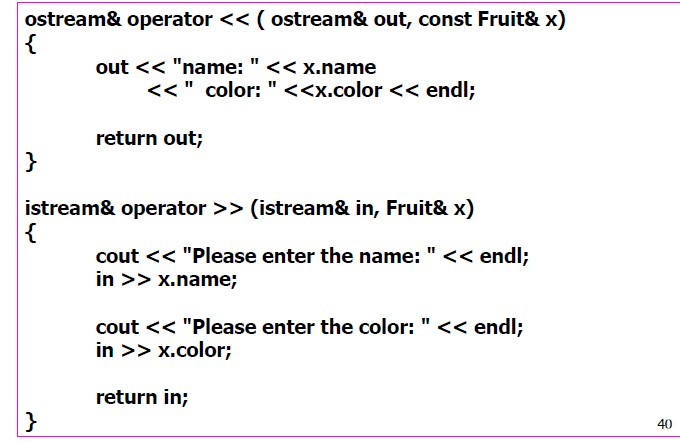
后置： for

里面可以不是0，而是





输入输出的重载



只分析输出的重载：

先把函数头抄下来：

ostream& operator <<（ostream &out，const Fruit& x）

再写一个来对比的函数头：

int& sum(int &a, int &b)

对照一下，下面的int&代表返回值类型是一个int类型的引用，那么上面的就是说返回值是ostream类型的引用，这个ostream类型为输出流类型。

接下来看operator <<，这个对比前面的+号的重载，应该能明白这个是对<<的重载吧…

接下来看后面括号里面的内容，括号里面的东西是什么…当然是形参列表啦…

下面的int& a是指第一个参数是一个int类型的引用参数，在函数里面这个变量名叫a，那么看上面的重载函数，ostream &out是指第一个参数是一个ostream类型的引用参数，在函数里面这个变量名叫out。

后面那个const Fruit& x就不用我再说了吧….

所以，这里不是你说的重载in和out…明白了么…in跟out只是一个形参…

然后，重载这两个东西有什么用…

还是会回到原来的问题，你为什么要重载一个add函数…因为它不支持那个类型，所以要重载让它能匹配那个类型。

重载完怎么用？

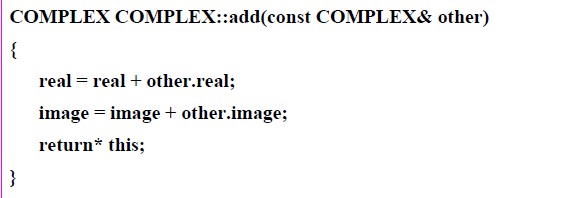
想想这两个符号在哪里出现过…

对了，就是cin>>…跟cout<<…啦…

所以重载完之后就能直接用这两个来进行输入输出咯…

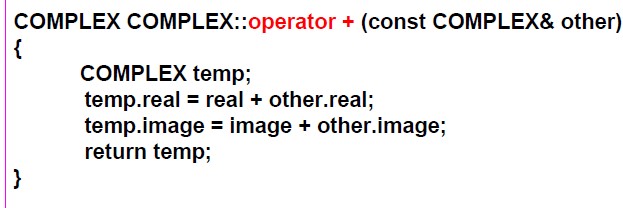
两种重载方式：

通过this指针隐式传递。



含指针成员的类应重载赋值运算符，实现深拷贝。

先释放指针原先指向的内存空间，再重新分配内存空间，再把需要拷贝的内容置入此内存空间，避免内存垃圾和指针重名。



第一种，只能c2.add(c1)

第二种，c2=c2+c1

第一种会改变real，第二种设了个temp临时变量就不会改变real

没设临时变量就return\* this