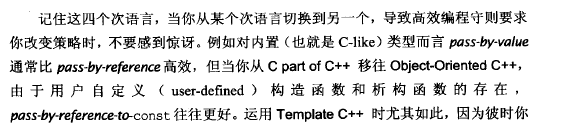
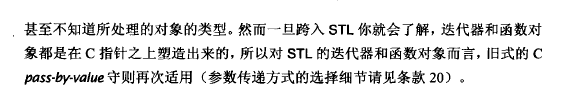


所以如果是引用的话~ Widget& w,则是调用Copy Assignement？就会相当于引用了。

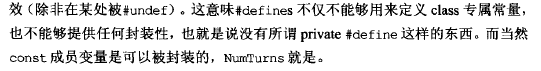


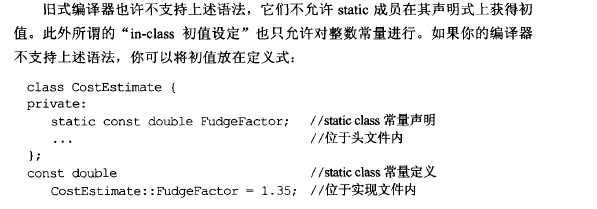


Const Widget& widget

## 用const等替换#define



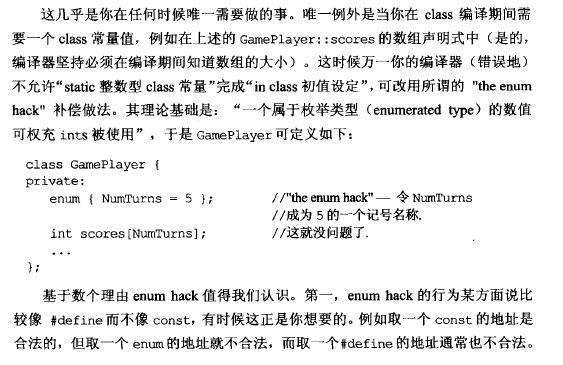




上述语法说的是，直接在类中声明的时候，就把值定了。

Static const double FudgetFactor = 1.35; 在CPP中，声明一下，不设初始值。

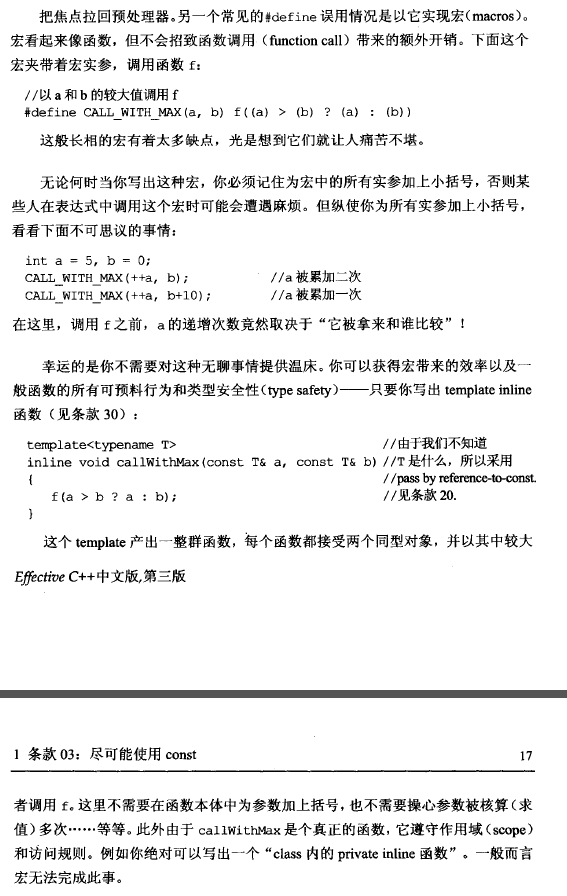
Const double CostEstimate::Fu…..

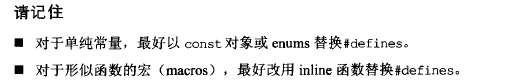


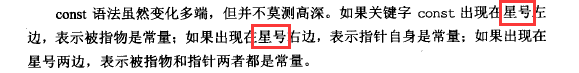
需要常量值，但是老编译器又不能通过static const的直接赋值时，

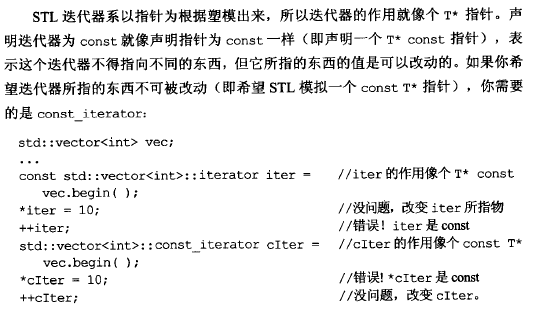


这个很有意思，用宏的话，可能会造成重复调用累加多次，当然有些函数也有可能，



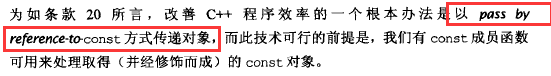




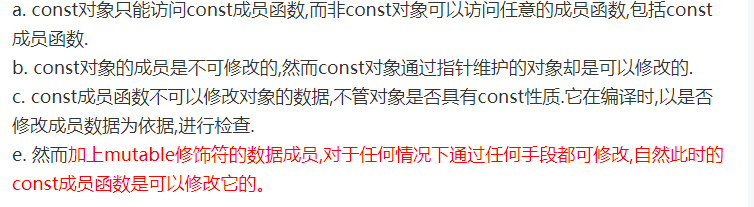


所以const 的迭代器，不能++,const\_iterator的迭代器，不能修改指向的值.

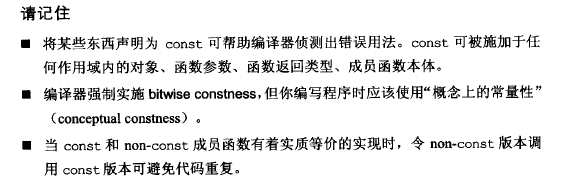
const 放在函数后面，成员函数，



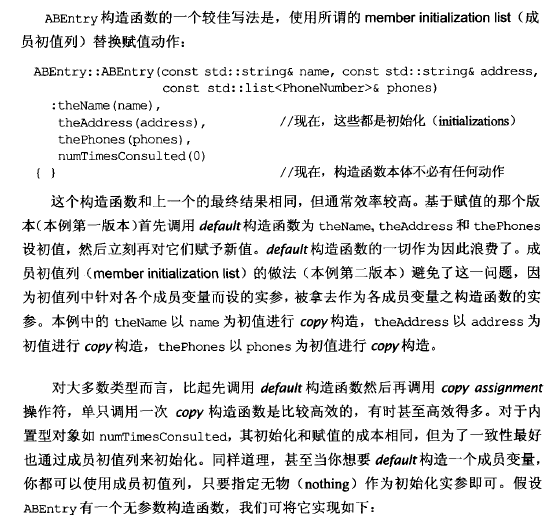


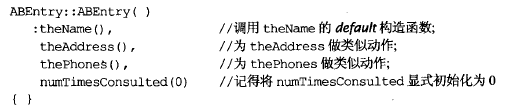


对const成员函数，可以通过指针去修改！！！(有空的话，去测试一下EventSystem里面的GetDesignSize还是啥的函数需要Size，size可以从Draw还是Arrange，是一个const的函数中获得，我想从中输出值却一直没办法，现在看来，可以用指针或是mutable！)



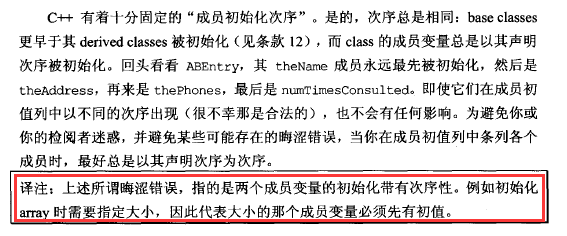
## 确定对象使用前已初始化





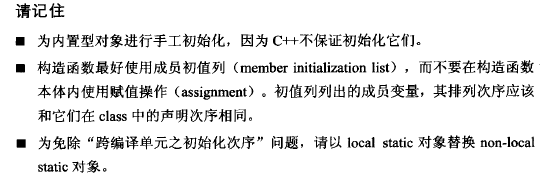
避免在构造函数中，使用theName = name这种，这样的是赋值，而不是初始化。

const 或是 references ，一定要初值，不能被赋值，所以最好总是使用成员初值列。



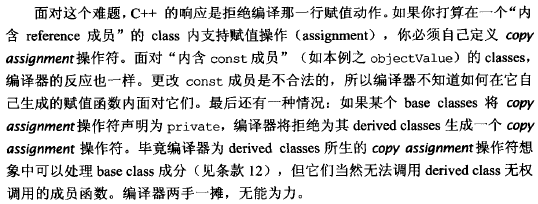
!!!这个很重要呀~用上一个声明来初始化下一个的数组这样的。





最后一条的意思，就很像是单例了~

# 构造，析构和赋值运算（64页）



总结一下其实是这个情况，就是如果你没有手动写，而是调用类中默认的copy assignment操作符的话，他会拒绝为string& nameValue这样的reference变量以及const T objectValue

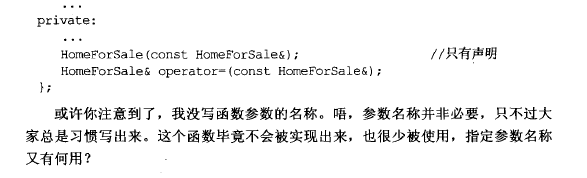
这样的const对象赋值。



## 若不想使用编译器自动生成的函数



你不声明的话，如果调用到，编译器也会帮你加上,方案1，自己进行显示声明为private，不够安全。



参数名称不是必要的！

方案2：用一个类，然后再去继承他,可能会多重继承就是了

为多态基类声明virtual析构函数70页