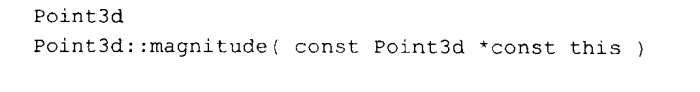
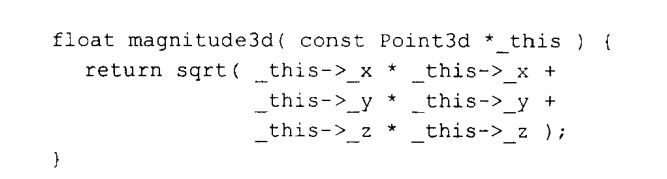


**1 非静态成员函数**

nostatic member functions编译器在保证他的效率很非成员函数效率一样，通过在nostatic member functions添加一个形参，this来进行调用。



非成员函数调用代码



所以可以得出两者调用时间（这里是因为代码一样）一样

**名称特殊改变**

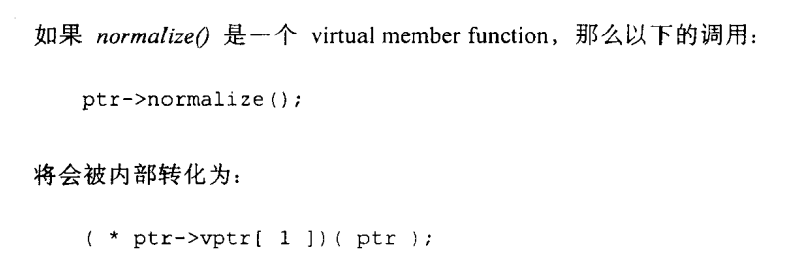
clipboard.png

clipboard.png

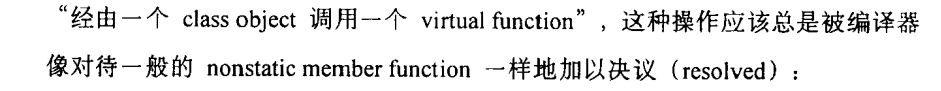
我们会注意到编译器在编译过程中实际上吧我们书写的函数和函数参数，变量名称都会在后面添加他的class名称，做到独一无二的存在。

**虚拟继承成员函数**

**一般的调用方式**

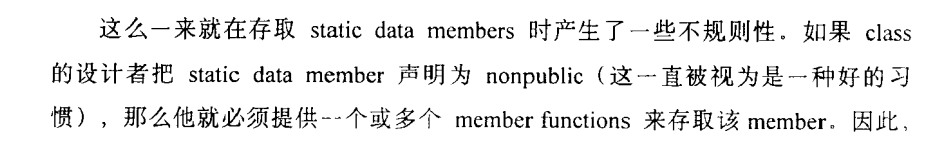


**经过优化和上面的调用一样**

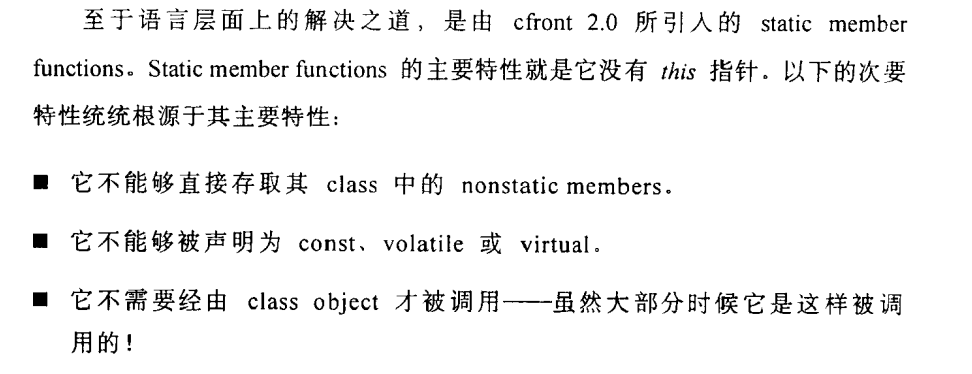


**静态成员函数**

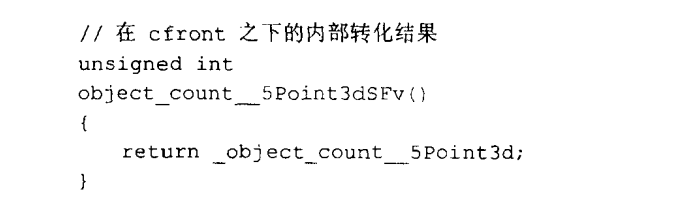
**建议**



**特性**



**内部名称转换**



扩展名多了一个SFV（）

**静态成员函数的地址是**

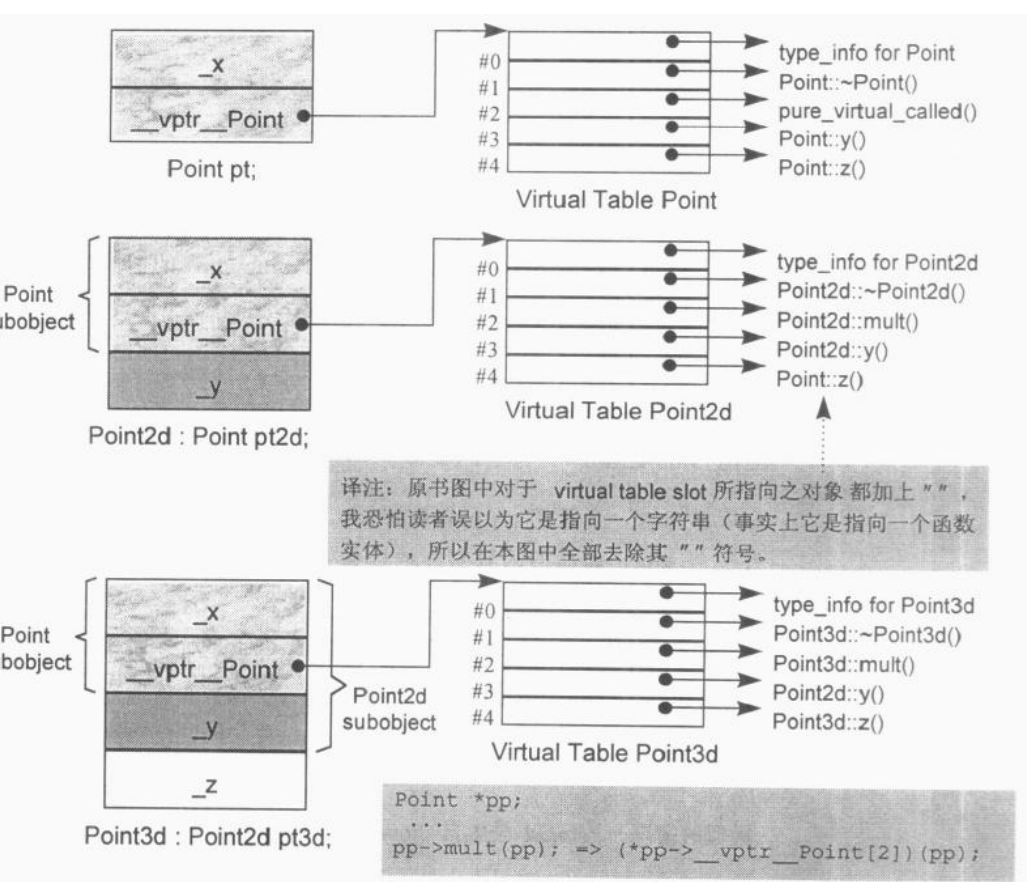
clipboard.png

而不是

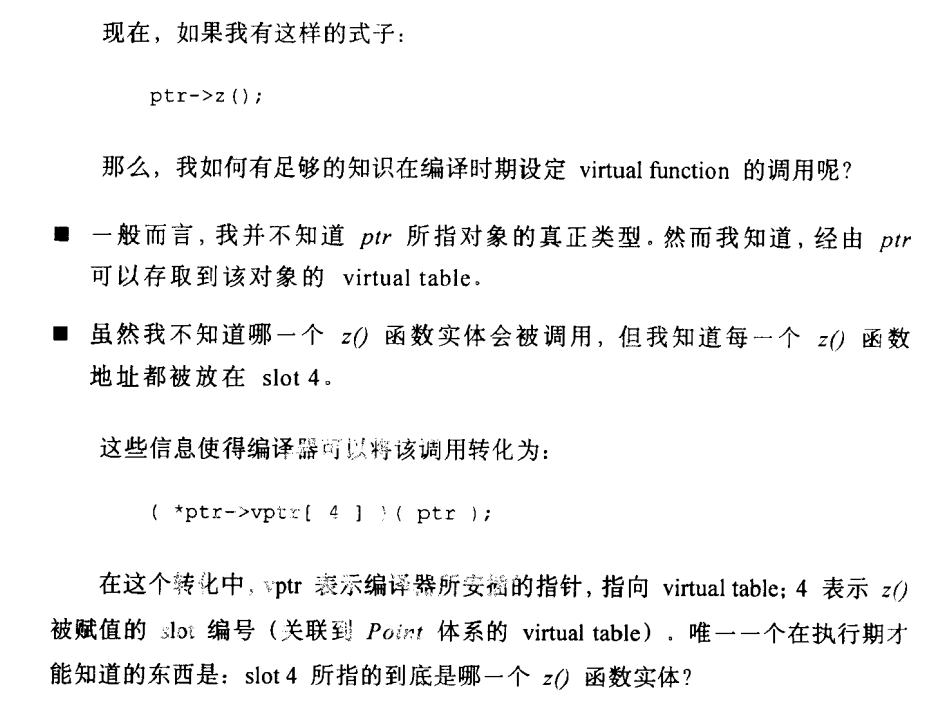
clipboard.png

**虚拟成员函数**

**单一继承成员函数（讲解单一继承多态确定最终的函数问题）**

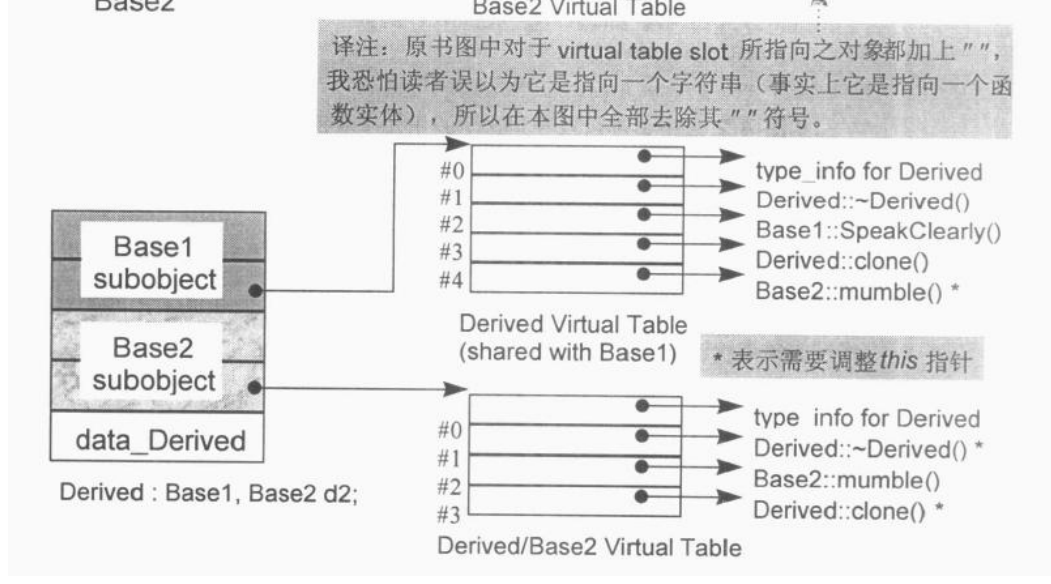


我们注意在point中 y 和z virtual函数，及时是在point2d中被重新 y 函数他的在virtual table中的位置没有发生变化，point2d中新写了mult函数，他被合理安排，拒绝打乱继承的virtual函数位置，

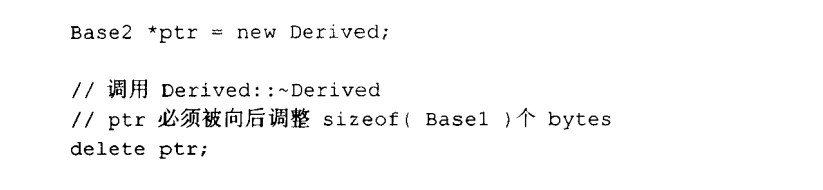


我们注意到前面图片，每个类都有一个自己的virtual table表，所以这要确定z函数他在表的solt的顺序,这样就可以利用多态，指针指向具体的类来定位具体的virtual table 表，来确定具体函数。

**多重继承下的virtual functions**

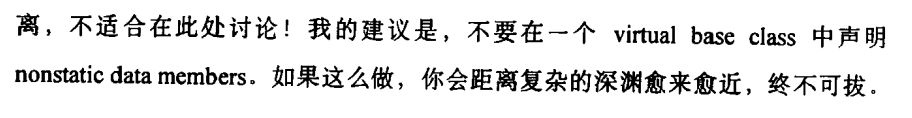


多重继承下，从图片中我们依据可以看出，继承了多个base的virtual table 的表格，如何定位第二个virtual table 是一个很重要的问题，因为第一个base virtual可以看做上面的单一继承



这里有一个技术通过一个this指针，this指向该derived对象的起始点，上面代码中 初始化ptr过程中，this指针会自然的往后移动指向第二个virtual table 以此来做到，通过this指针来调用 base2继承的函数（依照单一继承原理），避开了base1的virtual table。

**虚拟继承下的virtual function**

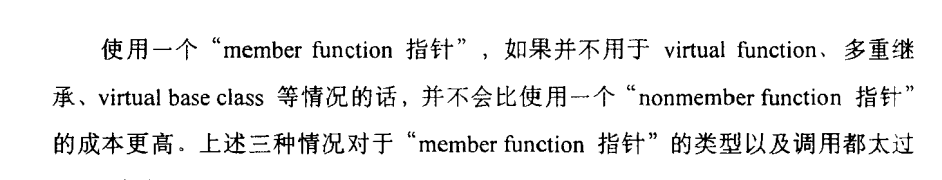


函数的效能

在函数实现过程中，有对该函数进行多次循环过程中，减少constructer函数对于变量的赋值操作，用参数调用来解决

**指向member function的指针**

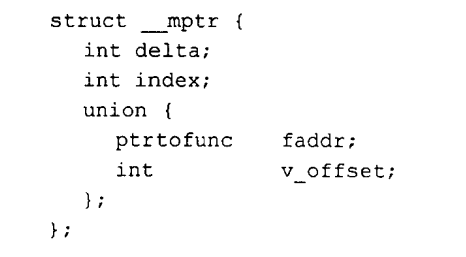
对于一个nostatic member function我们取他地址可以获得他的真实地址，所以就是一个正常函数指针使用，前面所说的，member function和no member function 效率一样，所以下面的可以得出



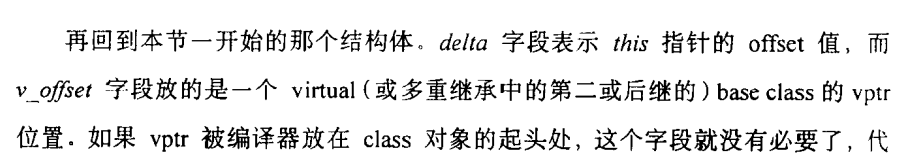
**指向virtual function下的指针**

virtual function 得出的地址是在virtual table 中存在的地址（序号 1），所以需要到实例化时才能得出是在那个class object的virtual function，这就是 virtual function指针的内部支持手段。

**如何支持多重继承的virtual function的指针**

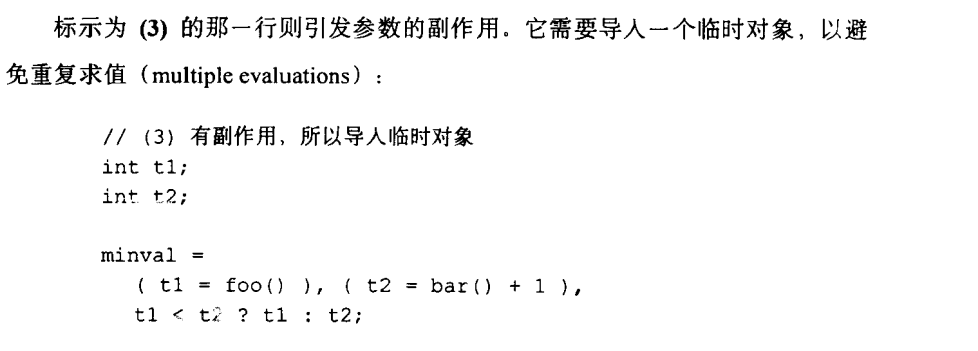


这个结构体回答前面我们提到过的多重继承应该内部流程。



**inline函数参数**

如果给inline函数传入常量表达式，形参会在inline函数扩展阶段被实参取代，如果传入参数是一个函数等需要求值的式子，inline扩展会创建两个·临时对象



给inline传入两个临时对象那么在inline扩展期间也会创建两个临时对象，像上面代码一样

