

Grius

《Effective C++》常用代码风格总结

本文档总结了《Effective C++》一书中一些常用的，且与目前开发小组人员水平相符的有关改善程序设计与编码风格的知识点，《Effective C++》一书中还有很多很值得推崇的原则，有心之人可自己查看。

作者：张洁

最后编辑：2013.1.1

# 知识点总结

* **尽量以const,enum,inline替换#defines**

对于单纯常量，最好以const对象或enums替换#defines。这样可以证

编译器一定能看到你所定义的那个常量，避免报错。

对于形似函数的宏，最好改用inline函数替换#defines。这样，我们对预处理器的需求就降低了，并且由于#defines在定义形似函数的宏时，经常因为一些括号还有参数问题而导致了很诡异的计算结果。

* **尽可能的使用const**

使用到的场合可能有：函数参数，修饰成员函数（包括修饰返回值和修

饰函数体），修饰成员变量。特别强调下在修饰成员函数体的时候，不能在这个函数中修改类的成员变量。尽可能的使用const可以帮助编译器侦测出错误用法，对程序差错帮助很大。

* **确定对象被使用前已经先被初始化**

这样可以避免未初始化时那些莫名其妙的随机值带来的错误。

* **构造函数最好使用成员初值列初始化**

而不是在构造函数本体内使用赋值操作。初值列列出的成员变量，其排列次序应该和它们在类中声明次序相同。

* **以local static对象替换non-static对象，可以避免跨编译单元的初始化次序**
* **对于编译器自动为声明的类生产的那些函数(构造，析构，=运算符，拷贝构造)，不要用甚至不能用的请明确拒绝**

实现方法是：将其声明为私有。

* **为多态基类声明virtual析构函数**

避免由于基类的指针指向派生类的对象，删除该指针的时候出现的局

部删除（只删除了父类的数据）。

* **成对使用new和delete时要采取相同形式**

这个意思是：怎么new出来的对象，就要怎么delete（当然是在系统没有自动delete的情况下）。举个例子：string\* array = new string[100]; delete array;这样是不对的，还有至少99个对象没有被删除。

* **传递参数的时候，尽量用传引用的方式代替传值的方式**

因为传值的方式，特别是对对象来说，需要调用拷贝函数，如果该对象的属性中有另一个类与其是组合的关系，还得调用其的构造函数和析构函数，除此之外，如果该类有父类，还得调用父类的构造函数和析构函数。而传引用的方式是不需要的。且最好是这种方式：const A& a(A是类名。)

* **成员变量声明为private，保证封装性。**
* **尽量少做转型动作，转型破坏了类型系统，导致的错误可能会非常隐晦。**
* **将大多数inline函数限制在小型、被频繁调用的函数身上**

这也就意味着在声明类的成员函数的时候，没有必要的话，不要在类中写函数的定义，因为类中的成员函数都是默认的inline类型函数。

* **避免基类和派生类出现相同名字的成员变量、成员函数**（除非为了实现重写）， 因为这样会造成遮掩名称的问题。
* **绝对不要重新定义继承而来的非虚函数**

因为对于子类和基类而言，非虚函数对于两个类来说都是静态绑定的，也就是说通过基类对象的指针调用的这个函数，永远是基类所定义的版本，即使基类指针指向的是派生类的对象。多态应该是由virtual的动态联编实现的。

* **绝对不重新定义继承来的缺省参数值**

继承来的参数只会出现在非虚函数和虚函数中。对于前者，上条已经解释了。而后者呢，虚函数是动态绑定，但是缺省参数是静态绑定。

以上都是一些经常会用到的而且非常实用的原则点，《E》这本书中还有很多关于设计的改善风格的原则，本文档没有全部列举。