## 尽量用const和inline而不用#define

#define的问题

#define max(a,b) ((a) > (b) ? (a) : (b))

int a = 5, b = 0;

max(++a, b);// a 的值增加了2次

## 避免public接口出现数据成员

* 在public接口里放上数据成员无异于自找麻烦，所以要把数据成员安全地隐藏在与功能分离的高墙后。如果现在就开始这么做，那我们就可以无需任何代价地换来一致性和精确的访问控制。

## 尽可能使用const

* 以下两函数可以重载，const函数内部不能显示的修改成员变量，申明为mutable 的成员变量除外

int &f()

int &f() const

* const\_cast<char\*>const\_buf同等于c方式的(char \*)const\_buf

## 避免这样的成员函数：其返回值是指向成员的非const指针或引用，但成员的访问级比这个函数要低

* 劳累的编译器要费九牛二虎之力来确保你设置的访问限制不被破坏，你也不要糟蹋编译器的努力结果，返回const的引用是例外

## 将文件间的编译依赖性降至最低

* 如果可以使用对象的引用和指针，就要避免使用对象本身。定义某个类型的引用和指针只会涉及到这个类型的声明。定义此类型的对象则需要类型定义的参与。
* 尽可能使用类的声明，而不使用类的定义。因为在声明一个函数时，如果用到某个类，是绝对不需要这个类的定义的，即使函数是通过传值来传递和返回这个类。
* 不要在头文件中再#include其它头文件，除非缺少了它们就不能编译。相反，要一个一个地声明所需要的类，让使用这个头文件的用户自己（通过#include指令）去包含其它的头文件，以使用户代码最终得以通过编译。一些用户会抱怨这样做对他们来说很不方便，但实际上你为他们避免了许多你曾饱受的痛苦。事实上，这种技术很受推崇，并被运用到C++标准库。