不编鬼故事了.
COMMAND PATTERN: 代码变数据
OMG,LISP 的棺材压不住了!!!
这里没啥好说的, 要是我直觉上就会这么做.
倒是后面的" 更多用途" 说明了它" 代码变数据" 的本质

后面的 ADAPTER 和 FACADE 更有用些.
ADAPTER 实质上是某种情况下的必需品. 而 FACADE... 是一种... 神奇的... 简化方案.

偷偷告诉你,C&C++ 中小 FACADE 由 MACRO 实现.
这个 MACRO... 就是编译前来一次 SUBSITUATION.
FOR EXAMPLE:
#define MOVIE_START do{Popper.on();/*...*/}while(0)

提到类适配器,C++ 有一种 private 继承
private 继承让父类 public 和 protected 的 Method 和 Property 变为 private, 表达 HAS ONLY ONE (不是 IS A) 的关系
哦, 对了, 还有 protected 继承, 父类的东西于是就不是 private 而是 protected 的了.
这两种继承关系似乎正适用于类适配器???
现在没了鬼故事, 没了抄书, 只剩下补充的东东, 篇幅短了呢!
不过就这样吧,886(预计还有加餐)

加餐来也!!!
EMMM, 我还能说啥呢???
瞎说吧.
ADAPTER 你也看到了, 他适用于新旧连接处.
所以说, 在你自己的代码里, 它的出现表示你发现了一些类似的接口.
尽可能统一接口吧.
当然, 如果这两个接口无法统一 (使用同样的接口), 那么就要 ADAPTER 上了.
FACADE 你会连你用了都不知道.
例如–你写了一个类, 传入一条 SQL, 返回一个 SQL_Lines
这就是一个外观模式
但外观模式仅仅是一堆常用代码的汇总,Java 却要用一个” 函数调用” 来完成,C&C++ 能不能省掉来加速呢???
能! 你可以用宏或内联函数.
宏其实就是替换, 还是比较简单的
提到宏, 讲一下 C&C++ 的预处理.
C&C++ 程序编译前会被一个叫 $CP_{re}P_{roccesser}$ 的程序处理一遍.
这个玩意儿只处理”#” 开头的行, 其他的从不解析, 所以你也可以用他处理其他语言的代码.
但他会生成另一些”#” 开头的行, 所以你还得把这些”#” 去了.

你可以看一看 code/6A7/Before.cpp, 然后用 gcc -E Before.cpp 看一看预处理的结果
虽然宏解决了一个函数调用的浪费, 但它也有问题–
宏的另一种形式–宏调用允许把一些参数传递给宏, 但...
Bad.cpp 中, 虽然 () 保证了没有结合错误, 但 ++ 仍让它束手无策,.
SO!
C++ 提供了 inline 关键字, 尽可能 (不是一定!) 节省这类函数调用
但宏依旧存在. 以它能在任何地方 (包括函数定义在内) 改变的神奇能力出现.
当然, 还有向后兼容 233
嗯, 瞎扯扯够 200 字了 (话说如果大家认为我该抄书的话尽管开口)