总结

如果在类的层次结构中没有“菱形”的继承结构出现，MI将是相当简单的（虽然基类中完全相同的那些函数识别标志仍然必须解析）。如果有菱形继承结构出现，就需要通过引入虚基类来消除重复子对象。这不仅增加了混乱，而且使接下来的表达方式变得更加复杂和低效。

多重继承已经被称做“百分之90的goto语句”。[1]这种形容似乎是适当的，像避免使用goto语句那样在平常的编程中最好避免使用MI，但有时候它却很有用。它在C++中的地位是“次要的”，但却是C++的更高级特征，这一特征设计是用来解决特殊情况下出现的问题。如果读者发现自己经常使用了它，那么就需要检查一下使用它的原因。问一下自己，“是必需要向上类型转换成为所有的基类类型吗？”如果答案是否定的，假如嵌入的所有类的实例都不需要进行向上类型转换，那么编程工作将会变得更加简单。

正如任何其他抽象的特点，设计模式的特点就是为了使工作更加容易。系统中总是有一些东西在变化—这可能是在软件项目生命周期中代码的变化，或许是在某个程序执行的生命周期期间某些对象的变化。找出变化的东西，利用设计模式封装这些变化，并使这些变化能够得到控制。

人们在进行程序设计时很容易迷恋于使用某个特定的设计模式，并且如果因为刚刚知道如何做就贸然去做也将给自己带来烦恼。最难做到的是什么？有点讽刺意味，是遵循《极限编程》（《Extreme Programming》）中的那句格言：“只要能用，就做最简单的。”仅仅做最简单的东西，不仅能够最快速的实现设计，而且其设计也很容易维护。如果这种最简单的东西不能完成工作，读者很快就会发现，除了花费时间编写复杂的实现方法之外，它们还是不起作用的。