《C 编程规范》笔记（设计风格） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/133/2021_2022__E3_80_8AC___E7_BC_96_E7_c97_133701.htm 第1条：一个实体应该只有一个紧凑的职责 单一职责原则。这个原则并不那么容易执行，即使是STL这样的程序库，也一样会犯违反该原则的错误。在这里，举了两个违反这一原则的著名实现：realloc和stl 中的basic_string。不过，对于basic_string，我想比起MFC中的CString还是好了不少。在《Exceptional C style》中，对basic_string作了剖析，并且得出一个普遍的原则：尽量将函数实现为独立的函数而不是成员函数。 尝试用一句话来说明一个模块的功能，既不多，也不少。如果无法用这样的一句话加以概括，那么重新考虑规划该模块的职责。 第2条：正确、简单和清晰第一 简单的说，坚持KISS原则：正确优于速度，简单优于复杂，清晰优于机巧，安全优于不安全。 程序必须为阅读它的人编写，只是顺便用于机器执行 * 编写程序应该以人为本，计算机第二 计算机系统中最便宜、最快速、最可靠的组件都还不存在 .....简单设计的重要性怎么强调也不过分 使一个正确的程序变快，比使一个快速的程序正确要容易的多 避免使用程序设计语言的冷僻特性，应该使用最简单的有效技术 不要毫无节制地重载运算符。 不要滥用匿名变量，合理使用命名变量。当然，这不是说连vector().swap(other)这样的惯用法也要排斥。 第3条：编程中应知道何时和如何考虑可伸缩性 从字面上来看，这差不多等于外交辞令。答案无非是“ 适当的 ”时候“ 适当地 ”考虑可伸缩性。这非常依赖于软件工程师的经验和知识。所以，本条目也“ 适当地 ”

回避了那种缺乏营养的教导，着重讨论算法复杂度的选择问题。 基本上，线性复杂度可以作为一个算法是否可选的分界点。值得花费精力避免选择差于线性复杂度的算法，而不差于线性复杂度的算法则可以接受。所以，把性能放在嘴边的兄弟们注意了，你的精力可别放错了地方，高德纳言犹在耳：不成熟的优化是程序设计中的万恶之源。必要时，先努力优化复杂度（选择好的算法----算法无用论者，去面壁！）。顺便提一句排序算法，通用排序算法的复杂度最好是O(NlgN)，但是特定领域完全可以有更好复杂度的算法。