 程序库的设计就是一个折衷的过程。理想的程序库应该是短小的、快速的、强大的、灵活的、可扩展的、直观的、普遍适用的、具有良好的支持、没有使用约束、没有错误的。这也是不存在的。为尺寸和速度而进行优化的程序库一般不能被移植。具有大量功能的的程序库不会具有直观性。没有错误的程序库在使用范围上会有限制。真实的世界里，你不能拥有每一件东西，总得有付出。 不同的设计者给这些条件赋予了不同的优先级。他们从而在设计中牺牲了不同的东西。因此一般两个提供相同功能的程序库却有着完全不同的性能特征。 例如，考虑iostream和stdio程序库，对于C 程序员来说两者都是可以使用的。iostream程序库与C中的stdio相比有几个优点（参见Effective C ）。例如它是类型安全的（type-safe），它是可扩展的。然而在效率方面，iostream程序库总是不如stdio，因为stdio产生的执行文件与iostream产生的执行文件相比尺寸小而且执行速度快。 首先考虑执行速度的问题。要想掌握iostream和stdio之间的性能差别，一种方法就是用这两个程序库来运行benchmark程序。不过你必须记住benchmark也会撒谎。不仅很难拿出一组能够代表程序或程序库典型用法的数据，而且就算拿出来也是没用，除非有可靠的方法判断出你或你的客户的具有什么样的特征。不过在解决一个问题的不用方法的比较上，benchmark还是能够提供一些信息，所以尽管完全依靠benchmark是愚蠢的，但是忽略它们也是愚蠢的。 让我们测试一个简单

的benchmark程序，只测试最基本的I/O功能。这个程序从标准输入读取30000个浮点数，然后把它们以固定的格式写到标准输出里。编译时预处理符号STDIO决定是使用stdio还是iostream。如果定义了这个符号，就是用stdio，否则就使用iostream程序库。#ifdef STDIO#include #else#include #include using namespace std.#endifconst int VALUES = 30000. // # of values to read/writeint main(){ double d. for (int n = 1. n #ifdef STDIO scanf("%lf", &amp.d). printf(".5f", d). #else cin >> d. cout #endif if (n % 5 == 0) { #ifdef STDIO printf("\n"). #else cout #endif } } return 0.} 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com