第一图书网, tushu007.com

<<Effective STL中文版>>

图书基本信息

书名：<<Effective STL中文版>>

13位ISBN编号：9787121201257

10位ISBN编号：7121201259

出版时间：2013-5

出版时间：电子工业出版社

作者：梅耶

译者：潘爱民,陈铭,邹开红

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：http://www.tushu007.com

内容概要

《Effective STL中文版:50条有效使用STL的经验》是EffectiveC++的第3卷，被评为“值得所有C++程序员阅读的C++书籍之一”。
《Effective STL中文版:50条有效使用STL的经验》详细讲述了使用STL的50条指导原则，并提供了透彻的分析和深刻的实例，实用性极强，是C++程序员必备的基础书籍。
C++的标准模板库（STL）是革命性的，要用好STL并不容易。
《Effective STL中文版:50条有效使用STL的经验》作者ScottMeyers揭示了专家总结的一些关键规则，包括专家们总是采用的做法，以及专家们总是避免的做法。
通过这些规则，STL程序员可以最大限度地使用STL。

作者简介

作者:(美)梅耶 译者:潘爱民、陈铭、邹开红Scott Meyers是全世界最知名的c++软件开发专家之一。
他是畅销书《Effective C++》系列(Effective C++，More Fifective C++，Effective STL)的作者，义足创新产品《Effective c++CD》的设计者和作者，也是Addison-Wesley的“Effective Software Development Series”顾问编辑，以及《Software Development》杂志咨询板成员。
他也为若干新公司的技术咨询板提供服务。
Meyers于1993年白Brown大学获得计算机博士学位。
他的网址是WWW．aristeia．com。

书籍目录

引言1 1 容器9 第1条：慎重选择容器类型。
9 第2条：不要试图编写独立于容器类型的代码。
12 第3条：确保容器中的对象拷贝正确而高效。
16 第4条：调用empty而不是检查size（）是否为0。
18 第5条：区间成员函数优先于与之对应的单元素成员函数。
20 第6条：当心C++编译器最烦人的分析机制。
26 第7条：如果容器中包含了通过new操作创建的指针，切记在容器对象析构前将 指针delete掉。
28 第8条：切勿创建包含auto_ptr的容器对象。
32 第9条：慎重选择删除元素的方法。
34 第10条：了解分配子（allocator）的约定和限制。
38 第11条：理解自定义分配子的合理用法。
44 第12条：切勿对STL容器的线程安全性有不切实际的依赖。
47 2 vector和string51 第13条：vector和string优先于动态分配的数组。
51 第14条：使用reserve来避免不必要的重新分配。
53 第15条：注意string实现的多样性。
55 第16条：了解如何把vector和string数据传给旧的API。
60 第17条：使用“ swap技巧 ”除去多余的容量。
63 第18条：避免使用vector＜bool＞。
64 3 关联容器67 第19条：理解相等（equality）和等价（equivalence）的区别。
67 第20条：为包含指针的关联容器指定比较类型。
71 第21条：总是让比较函数在等值情况下返回false。
74 第22条：切勿直接修改set或multiset中的键。
77 第23条：考虑用排序的vector替代关联容器。
82 第24条：当效率至关重要时，请在map：：operator[]与map：：insert之间谨慎做出选择。
87 第25条：熟悉非标准的散列容器。
91 4 迭代器95 第26条：iterator优先于const_iterator、reverse_iterator及const_reverse_iterator。
95 第27条：使用distance和advance将容器的const_iterator转换成iterator。
98 第28条：正确理解由reverse_iterator的base（）成员函数所产生的iterator的用法。
101 第29条：对于逐个字符的输入请考虑使用istreambuf_iterator。
103 5 算法106 第30条：确保目标区间足够大。
107 第31条：了解各种与排序有关的选择。
110 第32条：如果确实需要删除元素，则需要在remove这一类算法之后调用erase。
115 第33条：对包含指针的容器使用remove这一类算法时要特别小心。
118 第34条：了解哪些算法要求使用排序的区间作为参数。
121 第35条：通过mismatch或lexicographical_compare实现简单的忽略大小写的字符 串比较。
124 第36条：理解copy_if算法的正确实现。
128 第37条：使用accumulate或者for_each进行区间统计。
130 6 函数子、函数子类、函数及其他135 第38条：遵循按值传递的原则来设计函数子类。
135 第39条：确保判别式是“ 纯函数 ”。
138 第40条：若一个类是函数子，则应使它可配接。
141 第41条：理解ptr_fun、mem_fun和mem_fun_ref的来由。
145 第42条：确保less＜T＞与operator＜具有相同的语义。
148 7 在程序中使用STL152 第43条：算法调用优先于手写的循环。
152 第44条：容器的成员函数优先于同名的算法。
159 第45条：正确区分count、find、binary_search、lower_bound、upper_bound和 equal_range。

161 第46条：考虑使用函数对象而不是函数作为STL算法的参数。
168 第47条：避免产生“ 直写型 ”（write—only）的代码。
172 第48条：总是包含（#include）正确的头文件。
175 第49条：学会分析与STL相关的编译器诊断信息。
176 第50条：熟悉与STL相关的Web站点。
183 参考书目189 A 地域性与忽略大小写的字符串比较193 B 对Microsoft的STL平台的说明202

章节摘录

版权页： 公平地讲，STL的实现者很清楚，begin、end（以及类似的函数，如size）都是被频繁使用的函数，所以他们尽最大可能提高其效率。
他们几乎肯定会使用inline来编译这些函数，并且努力改善这些函数的代码，尽可能让大多数编译器都能够将循环中的计算提到外面来，以避免重复计算。
然而，实践表明，实现者并不是每次都能成功，当他们不能成功的时候，因使用算法而避免重复计算所得到的性能优势，比起手写循环来就非常值得了。
但这只是很次要的性能增益。
最主要的是，类库实现者可以根据他们对于容器实现的了解程度对遍历过程进行优化，这是库的使用者所难以做到的。
例如，deque中的对象（在内部）通常被存放在一个或多个固定大小的数组中。
对于这些数组，基于指针的遍历比基于迭代器的遍历要快得多。
但是只有库的实现者才可以使用基于指针的遍历，因为只有他们才知道内部数组的大小，才知道如何从一个数组转移到另一个数组。
有些STL包含的算法实现考虑到了deque的内部数据结构，这些算法实现比“普通”的算法实现快了20%多。
这里的要点是，STL算法的实现未必一定都针对deque（或其他特定的容器类型）进行了优化，但无论如何实现者肯定比你更了解内部的实现细节，他们可以在算法实现中充分利用这些知识。
如果你避开算法调用而使用自己的手写循环，你也就放弃了这些算法实现可能提供的优化手段。
其次的性能增益在于，除了一些不太重要的算法以外，其他几乎所有的STL算法都使用了复杂的计算机科学算法，有些科学算法非常复杂，并非一般的C++程序员所能够达到的。
例如，你很难在算法级别上写出比STL的sort或类似的算法更有效的代码（见第31条）；STL中针对排序区间的查找算法也非常优秀（见第34条和第45条）；即使是那些普通的任务，比如从连续内存的容器中删除一些对象，使用erase.remove习惯用法所获得的性能也比一般程序员编写的循环要高效得多（见第9条）。
如果效率方面的原因还不能说服你，那么也许正确性对你更有吸引力。
当你编写循环代码的时候，最要紧的莫过于要保证你所使用的迭代器（a）都是有效的：（b）并且指向你所希望的地方。
例如，假设有一个数组（可能是由于某个遗留下来的C APl的原因，见第l6条），而你要让每个数组元素加上41，然后把它插入到一个deque的前部。

编辑推荐

《Effective STL中文版:50条有效使用STL的经验》适合广大编程爱好者及程序员阅读。
也适合作为高校相关专业教材，供学生学习和参考。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:http://www.tushu007.com