

<<STL高效编程>>

图书基本信息

书名：<<STL高效编程>>

13位ISBN编号：9787111196242

10位ISBN编号：7111196244

出版时间：2006-8

出版时间：机械工业出版社

作者：迈耶斯

页数：260

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>

<<STL高效编程>>

内容概要

C++的标准模板库（STL）是革命性的，但是要想学会用好STL却充满了挑战性。

中国台湾技术作家侯捷先生曾经把STL的学习过程比喻为三个境界： 第一境界：熟用STL。

第二境界：了解泛型技术的内涵与STL的学理乃至实作。

第三境界：扩充STL。

本书无疑是你达到第二境界的最佳读本。

在本书中，C++技术权威Scott Meyers揭示了专家总结的一些关键规则，包括他们总是采用的做法以及总是避免的做法。

这些规则可以发挥STL的最大效用。

有些书只是描述STL中有些什么内容，而本书则讲述如何使用STL。

本书共有50条指导原则，在讲述每一条指导原则时，Scott Meyers都提供了透彻的分析和深刻的实例，所以读者不仅可以学到要做什么，而且还能够知道什么时候该这样做，以及为什么要这样做。

如同Meyers的其他著作一样，本书充满了从实践中总结出来的智慧。

清晰、简明、透彻的风格使本书成为每一位STL程序员的案头必备。

本书特色 关于选择容器的建议，涉及的容器有：标准STL容器（例如vector和list）、非标准的STL容器（例如hash_set和hash_map），以及非STL容器（例如bitset）。

一些提高效率的技术，通过它们可以最大程度地提高STL（以及使用STL的程序）的效率。

深入到迭代器、函数对象和分配子（allocator）的行为，也包括程序员总是应该避免的做法。

对于那些同名的算法和成员函数，如find，根据它们行为方式上的微妙差异，本书给出了一些指导原则，以保证它们能被正确地使用。

讨论了潜在的移植性问题，包括避免这些移植性问题的各种简单途径。

<<STL高效编程>>

作者简介

Scott Meyers，拥有布朗大学计算机科学博士学位，是世界顶级的C++软件开发技术权威之一。他的两本畅销书《Effective C++》和《More Effective C++》开创了技术图书“Effective”式的写作风格。他曾担任《C++ Report》杂志的专栏作家，还经常为《C/C++ Users Journal》和《Dr.

书籍目录

Preface Acknowledgments Introduction Chapter 1: Containers Item 1: Choose your containers with care. Item 2: Beware the illusion of container-independent code. Item 3: Make copying cheap and correct for objects in containers. Item 4: Call empty instead of checking size() against zero. Item 5: Prefer range member functions to their single-element counterparts. Item 6: Be alert for C++ 's most vexing parse. Item 7: When using containers of newed pointers, remember to delete the pointers before the container is destroyed. Item 8: Never create containers of auto_ptrs. Item 9: Choose carefully among erasing options. Item 10: Be aware of allocator conventions and restrictions. Item 11: Understand the legitimate uses of custom allocators. Item 12: Have realistic expectations about the thread safety of STL containers. Chapter 2: vector and string Item 13: Prefer vector and string to dynamically allocated arrays. Item 14: Use reserve to avoid unnecessary reallocations. Item 15: Be aware of variations in string implementations. Item 16: Know how to pass vector and string data to legacy APIs. Item 17: Use "the swap trick" to trim excess capacity. Item 18: Avoid using vector. Chapter 3: Associative Containers Item 19: Understand the difference between equality and equivalence. Item 20: Specify comparison types for associative containers of pointers. Item 21: Always have comparison functions return false for equal values. Item 22: Avoid in-place key modification in set and multiset. Item 23: Consider replacing associative containers with sorted vectors. Item 24: Choose carefully between map::operator[] and map::insert when efficiency is important. Item 25: Familiarize yourself with the nonstandard hashed containers. Chapter 4: Iterators Item 26: Prefer iterator to const_iterator, reverse_iterator, and const_reverse_iterator. Item 27: Use distance and advance to convert const_iterators to iterators. Item 28: Understand how to use a reverse_iterator 's base iterator. Item 29: Consider istreambuf_iterators for character by character input. Chapter 5: Algorithms Item 30: Make sure destination ranges are big enough. Item 31: Know your sorting options. Item 32: Follow remove-like algorithms by erase if you really want to remove something. Item 33: Be wary of remove-like algorithms on containers of pointers. Item 34: Note which algorithms expect sorted ranges. Item 35: Implement simple case-insensitive string comparisons via mismatch or lexicographical_compare. Item 36: Understand the proper implementation of copy_if. Item 37: Use accumulate or for_each to summarize ranges. Chapter 6: Functors, Functor Classes, Functions, etc. Item 38: Design functor classes for pass-by-value. Item 39: Make predicates pure functions. Item 40: Make functor classes adaptable. Item 41: Understand the reasons for ptr_fun, mem_fun, and mem_fun_ref. Item 42: Make sure less means operator

<<STL高效编程>>

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>