

<<Exceptional C++ (中文) >>

图书基本信息

书名：<<Exceptional C++ (中文版) >>

13位ISBN编号：9787121170850

10位ISBN编号：712117085X

出版时间：2012-6

出版时间：电子工业出版社

作者：萨特

页数：255

字数：289000

译者：聂雪军

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

《exceptional c++ : 47个c++工程难题、编程问题和解决方案(中文版)》讲述如何用标准c++进行企业级的软件开发，通过“问题/解答”的方式，启发读者思考，帮助了解隐藏在问题背后的设计思想，以及各种编程指导原则适用的场合。本书列出的条款涵盖了许多方面的主题，尤其对异常安全性、类和模块的合理设计，正确的代码优化，以及编写符合c++标准的可移植代码进行了深入的讨论。

《exceptional c++ : 47个c++工程难题、编程问题和解决方案(中文版)》适于有一定c++编程基础的读者阅读。

作者简介

作者：(美国) 萨特 (Sutter H.) 译者：聂雪军

书籍目录

1 泛型程序设计与c++标准库

- 条款1：迭代器难度系数
- 条款2：大小写不敏感的字符串——之一
- 条款3：大小写不敏感的字符串——之二
- 条款4：可重用性最高的泛型容器——之一
- 条款5：可重用性最高的泛型容器——之二
- 条款6：临时对象
- 条款7：标准库的使用（或者，再论临时对象）

2 异常安全性相关的问题与技术

- 条款8：编写异常安全的代码——之一
- 条款9：编写异常安全的代码——之二
- 条款10：编写异常安全的代码——之三
- 条款11：编写异常安全的代码——之四
- 条款12：编写异常安全的代码——之五
- 条款13：编写异常安全的代码——之六
- 条款14：编写异常安全的代码——之七
- 条款15：编写异常安全的代码——之八
- 条款16：编写异常安全的代码——之九
- 条款17：编写异常安全的代码——之十
- 条款18：代码的复杂性——之一
- 条款19：代码的复杂性——之二

3 类的设计与继承

- 条款20：类的编写技巧
- 条款21：虚函数的重载
- 条款22：类之间的关系——之一
- 条款23：类之间的关系——之二
- 条款24：继承的使用和滥用
- 条款25：面向对象程序设计

4 编译器防火墙和pimpl惯用法

- 条款26：将编译期依赖性降到最低——之一
- 条款27：将编译期依赖性降到最低——之二
- 条款28：将编译期依赖性降到最低——之三
- 条款29：编译防火墙
- 条款30：fast pimpl惯用法

5 名字查找、名字空间和接口规则

- 条款31：名字查找与接口规则——之一
- 条款32：名字查找与接口规则——之二
- 条款33：名字查找和接口规则——之三
- 条款34：名字查找与接口规则——之四

6 内存管理

- 条款35：内存管理——之一
- 条款36：内存管理——之二
- 条款37：auto_ptr

7 误区、陷阱以及错误的惯用法

- 条款38：对象标识

条款39：自动转换

条款40：对象的生存期——之一

条款41：对象的生存期——之二

8 其他主题

条款42：变量的初始化

条款43：正确使用const

条款44：类型转换

条款45：bool

条款46：转调函数

条款47：控制流程

后记

参考书目

章节摘录

版权页：从Gury of the Week条款8的最初发布到现在，这一系列的条款已经走过了很长的一段路程。我希望能喜欢它们并且发现它们确实很有用。

我要特别感谢委员会成员Dave Abrahams和Greg Colvin，感谢他们在阐述异常安全性上的深刻洞察力，以及对这部分内容的草稿所提出的中肯批评。

Dave和Grep，以及Matt Austern，他们一起编写了两个完整的会议提案，这些提案的议题是将当前的异常安全性保证添加到标准库中。

在这个系列的条款中，我们将讨论两个主题：首先是C++的两个主要特性，异常处理和模板，我们将分析如何编写异常安全的代码（即在出现异常的情况下代码仍能正确运行）；其次是异常中立（即将所有的异常都转发给调用者）的泛型容器。

这些东西说起来很容易，但做起来绝不轻松。

现在，我们一起来实现一个简单的容器（一个能执行压入操作和弹出操作的栈），并分析如果要使这个容器成为异常安全的和异常中立的，我们需要解决哪些问题。

我们将从Cargill停下来的地方开始——也就是，逐步地创建一个异常安全的模板stack，而cargill当初也正是以Stack为例来提出他的问题的。

稍后，我们将降低对模板参数类型T的需求，显著改善Stack容器，然后还将给出一些高级技巧，用于在管理资源时实现异常安全性。

按照这种方法，我们可以找出下列问题的答案：异常安全性的不同“级别”指的是什么？

泛型容器可以是或者说应该是完全异常中立的吗？

标准库中的容器类是异常安全的还是异常中立的？

异常安全性会不会影响对容器公共接口的设计？

在泛型容器中应该使用异常规范吗？

这个构造函数是异常安全的和异常中立的吗？

要搞清楚这个问题，我们先考虑在哪些地方将可能抛出异常。

简单来说，答案就是：在每个函数中。

因此，第一步就是对上述代码进行分析并确定实际调用了哪些函数，包括自由函数、构造函数、析构函数、运算符重载函数，以及其他的成员函数。

在Stack的构造函数中，首先将vsize_设为10，然后通过new T (vsize_)来分配初始内存。

后者将首先调用operator new () () (或者是默认的operator new ()，或者是由T提供的operator new ())来分配内存，然后再调用vsize_次T::T函数。

在这个过程中有两个操作可能会失败。

第一个操作是内存分配操作本身，在失败的情况下。

operator new () ()将抛出一个bad alloc异常。

第二个操作是T的默认构造函数，在这个函数中可能抛出任意的异常，在这种情况下，所有已经构造的T对象都会被销毁，并且通过调用。

operator delete () ()来确保已分配的内存被自动回收。

因此，上面的函数是完全异常安全的和异常中立的，我们继续来看下一个问题.....什么？

你问为什么这个函数是健壮的？

好吧，我们对这个函数进行更为详细的讨论。

1.这个函数是异常中立的。

在函数中不会捕获任何异常，因此，如果new抛出了异常，那么这个异常就会像我们所要求的那样被正确地转发给调用者。

编辑推荐

《Exceptional C++:47个C++工程难题、编程问题和解决方案(中文版)》中的每个问题都给出了难度系数，在这些问题中阐释一些微妙的编程错误以及程序设计上的考虑。

在阅读书中给出的解答之前，你可以先尝试自己进行解答。

《Exceptional C++:47个C++工程难题、编程问题和解决方案(中文版)》将对这些问题进行详细的分析，并指出哪些地方是错误的，以及如何改正这个问题。

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>