第一图书网, tushu007.com

<<C++编程规范>>

图书基本信息

书名：<<C++编程规范>>

13位ISBN编号：9787115239402

10位ISBN编号：7115239401

出版时间：20101112

出版时间：人民邮电出版社

作者：Herb Sutter,Andrei Alexandrescu

页数：216

译者：刘基诚

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

前言

尽早进入正轨：以同样的方式实施同样的过程。
不断积累惯用法。
将其标准化。
如此，你与莎士比亚之间的唯一区别将只是掌握 惯用法的多少，而非词汇的多少。
——Alan Perlis 标准最大的优点在于，它提供了如此多样的选择。
——出处尚无定论 我们之所以编写本书，作为各开发团队编程规范的基础，有下面两个主要原因。
编程规范应该反映业界最久经考验的经验。
它应该包含凝聚了经验和对语言的深刻理解的公认的惯用法。
具体而言，编程规范应该牢固地建立在大量丰富的软件开发文献的基础之上，把散布在各种来源的规则、准则和最佳实践汇集在一起。
不可能存在真空状态。
通常，如果你不能有意识地制定合理的规则，那么就会有其他人推行他们自己喜欢的规则集。
这样产生的编程规范往往具有各种最不应该出现的属性。
例如，许多这样的编程规范都试图强制尽量少地按C语言的方式使用C++。
许多糟糕的编程规范都是由一些没有很好地理解语言、没有很好地理解软件开发或者试图标准化过多东西的人制定的。
糟糕的编程规范会很快丧失可信度，如果程序员不喜欢或者不同意其中一些糟糕的准则，那么即使规范中有一些合理的准则，也可能被不抱幻想的程序员所忽略，这还是最好的情况，最坏的情况下，糟糕的标准可能真会被强制执行。
如何使用本书 三思而行。
应该遵循好的准则，但是不要盲从。
在本书的各准则中，请注意“例外情况”部分阐明了该准则可能不适用的不太常见的情况。
任何准则，无论如何正确（当然，我们自认为本书中的准则是正确的），都不能代替自己的思考。
每个开发团队都应该制定自己的标准，制定标准的时候都应该尽职尽责。
这项工作是整个团队的事情。
如果你是团队负责人，应该让团队成员都参与制定标准。
人们当然更愿意遵守“自己的”标准，而非别人强加的一堆规矩。
编写本书的目的是为各开发团队提供编程规范的基础和参考。
它并不是要成为终极编程规范，因为不同的团队会有适合特定群体或者特定任务的更多准则，应该大胆地将这些准则加入本书的条款中。
但是我们希望本书能够通过记载和引用广泛接受的、权威的、几乎可以通用的（“例外情况”指出的除外）实践经验，减少读者制定或重新制定自己的编程规范的工作量，从而帮助提高读者所用编程规范的质量和一致性。
让团队人员阅读这些准则及其原理阐释（也就是本书全文，根据需要还包括所选条款引用的其他书籍和论文），共同决定是否有团队根本无法接受的内容（比如，由于某些项目特殊的情况），然后实践其余规范。
一旦采纳，如果未与整个团队协商，任何人不得违反团队编程规范。
最后，团队还需定期复查这些准则，加入实际应用中得出的经验和反馈。

第一图书网, tushu007.com

内容概要

在本书中，两位知名的C++专家将全球C++界20年的集体智慧和经验凝结成一套编程规范。这些规范可以作为每一个开发团队制定实际开发规范的基础，更是每一位C++程序员应该遵循的行事准则。
书中对每一条规范都给出了精确的描述，并辅以实例说明；从类型定义到错误处理，都给出了最佳的C++实践，即使使用C++多年的程序员也会从本书中受益匪浅。
本书适合于各层次C++程序员使用，也可作为高等院校C++课程的教学参考书。

作者简介

作者：（加拿大）萨特（Herb Sutter） （罗马）亚历山德雷斯库（Andrei Alexandrescu） 译者：刘基诚
萨特，（Herb Sutter） 曾任ISO C++标准委员会主席，是C++ Users Journal杂志特邀编辑和专栏作家。他目前在微软公司领导.NET环境下C++语言扩展的设计工作。
除本书外，他还撰写了三本广受赞誉的图书：Exceptional C++ Style、Exceptional C++和More Exceptional C++。
前者中译本也由人民邮电出版社出版。
亚历山德雷斯库(Andrei Alexandrescu)世界顶尖的C++专家，C++ Users Journal杂志的专栏作家，他的Modem C++ Design一书曾荣获2001年最佳C++图书称号，所开发的Loki已经成为最负盛名的C++程序库之一。

<<C++编程规范>>

书籍目录

组织和策略问题　第0条　不要拘泥于小节(又名：了解哪些东西不应该标准化)　第1条　在高警告级别干净利落地进行编译　第2条　使用自动构建系统　第3条　使用版本控制系统　第4条　做代码审查　9设计风格　第5条　一个实体应该只有一个紧凑的职责　第6条　正确、简单和清晰 第一　第7条　编程中应知道何时和如何考虑可伸缩性 第8条　不要进行不成熟的优化　第9条　不要进行不成熟的劣化　第10条　尽量减少全局和共享数据　第11条　隐藏信息　第12条　懂得何时和如何进行并发性编程　第13条　确保资源为对象所拥有。
使用显式的RAII和智能指针　编程风格　第14条　宁要编译时和连接时错误，也不要运行时错误　第15条　积极使用const　第16条　避免使用宏　第17条　避免使用“魔数”　第18条　尽可能局部地声明变量　第19条　总是初始化变量　第20条　避免函数过长，避免嵌套过深　第21条　避免跨编译单元的初始化依赖　第22条　尽量减少定义性依赖。
避免循环依赖　第23条　头文件应该自给自足　第24条　总是编写内部#include保护符，决不要编写外部#include保护符　函数与操作符　第25条　正确地选择通过值、(智能)指针或者引用传递参数　第26条　保持重载操作符的自然语义　第27条　优先使用算术操作符和赋值操作符的标准形式　第28条　优先使用++和--的标准形式。
优先调用前缀形式　第29条　考虑重载以避免隐含类型转换　第30条　避免重载&&、||或　,(逗号)　第31条　不要编写依赖于函数参数求值顺序的代码　类的设计与继承　第32条　弄清所要编写的是哪种类　第33条　用小类代替巨类　第34条　用组合代替继承　第35条　避免从并非要设计成基类的类中继承　第36条　优先提供抽象接口　第37条　公用继承即可替换性。
继承，不是为了重用，而是为了被重用　第38条　实施安全的覆盖　第39条　考虑将虚拟函数声明为非公用的，将公用函数声明为非虚拟的　第40条　要避免提供隐式转换　第41条　将数据成员设为私有的，无行为的聚集(C语言形式的struct)除外　第42条　不要公开内部数据　第43条　明智地使用Pimpl　第44条　优先编写非成员非友元函数　第45条　总是一起提供new和delete　第46条　如果提供类专门的new，应该提供所有标准形式(普通、就地和不抛出)　构造、析构与复制　第47条　以同样的顺序定义和初始化成员变量　第48条　在构造函数中用初始化代替赋值　第49条　避免在构造函数和析构函数中调用虚拟函数　第50条　将基类析构函数设为公用且虚拟的，或者保护且非虚拟的　第51条　析构函数、释放和交换绝对不能失败　第52条　一致地进行复制和销毁　第53条　显式地启用或者禁止复制 第54条　避免切片。
在基类中考虑用克隆代替复制　第55条　使用赋值的标准形式　第56条　只要可行，就提供不会失败的swap(而且要正确地提供)　名字空间与模块　模板与泛型　错误处理与异常　STL：容器　STL：算法　类型安全　参考文献　摘要汇总　索引

章节摘录

插图：外部加锁：调用者负责加锁。
在这种选择下，由使用对象的代码负责了解是否跨线程共享了对象，如果是，还要负责串行化所有对该对象的使用。
例如，字符串类型通常使用外部加锁（或者不变性，见第三种选择）。
内部加锁：每个对象将所有对自己的访问串行化，通常采用为每个公用成员函数加锁的方法来实现，这样调用者就可以不用串行化对象的使用了。
例如，生产者／消费者队列通常使用内部加锁，因为它们存在的目的就是被跨线程共享，而且它们的接口就是为了在单独的成员函数调用（Push，Pop）期间能够进行适当的层次加锁而设计的。
更一般的情况下，需要注意，只有在知道了以下两件事情之后这个选项才适用。
第一，必须事先知道该类型的对象几乎总是要被跨线程共享的，否则到头来只不过进行了无效加锁。
请注意大多数类型都不会遇到这种情况，即使是在多线程处理分量很重的程序中，大多数对象也不会被跨线程共享（这是好现象，见第10条）。
第二，必须事先知道成员函数级加锁的粒度是合适的，而且能满足大多数调用者的需要。
具体而言，类型接口的设计应该有利于粗粒度的、自给自足的操作。
如果调用者总是需要对多个而不是一个操作加锁，那么就不能满足需要了，只能通过增加更多的（外部）锁，将单独加锁的函数组装成一个更大规模的已加锁工作单位。
例如一个容器类型，如果它返回一个迭代器，则迭代器可能在用到之前就失效了；如果它提供find之类的能返回正确答案的成员算法，那么答案可能在用到之前就出错了；如果它的用户想要编写这样的代码：if（c,empty（））c.push-back（x）；，同样会出现问题。
（更多的例子，参阅[Sutter02]。
）在这些情况下，调用者需要进行外部加锁，以获得生存期能够跨越多个单独成员函数调用的锁，这样一来每个成员函数的内部加锁就毫无用武之地了。

编辑推荐

《C++编程规范:101条规则、准则与最佳实践》：良好的编程规范可以改善软件质量，缩短上市时间，提升团队效率，简化维护工作。
在《C++编程规范:101条规则、准则与最佳实践》中，两位全世界最受尊敬的c++专家将全球c++社区的集体智慧和经验凝结成一整套编程规范。
这些规范可以作为每一个开发团队制定实际开发规范的基础，更是每一位c++程序员应该遵循的行事准则。
《C++编程规范:101条规则、准则与最佳实践》涵盖了c++程序设计的每一个方面，包括设计和编码风格、函数、操作符、类的设计、继承、构造与析构、赋值、名字空间、模块、模板、泛型、异常、stl容器和算法等。
书中对每一条规范都给出了言简意赅的叙述，并辅以实例说明；另外还给出了从类型定义到错误处理等方面的大量c++ 最佳实践，包括许多最新总结和标准化的技术，即使使用c++多年的程序员也会从中受益匪浅。
通过阅读《C++编程规范:101条规则、准则与最佳实践》，可以找到以下问题的答案哪些东西值得标?准化?哪些东西不值得标准化?使代码可扩展的最佳方法是什么?合理的错误处理策略有哪些要素?如何(和为什么要)避免不必要的初始化、循环依赖和定义依赖?何时应该(以及如何)同时使用静态和动态的多态性?如何实践“安全的”改写?何可时该提供不会失败的交换?为什么要阻止异常跨越模块边界传播7如何阻止?为什么不应该在头文件中写名字空间声明或指令?为什么应该使用STL vector和string代替数组?如何选择正确的STL搜索算法?为了保证代码的类型安全，应该遵从哪些规则?C++界20年集大成之作两位世界顶级专家联袂巨献适合所有层次C++程序员

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