

深入理解

C++11

C++11新特性解析与应用

Understanding C++11
Analysis and Application of New Features

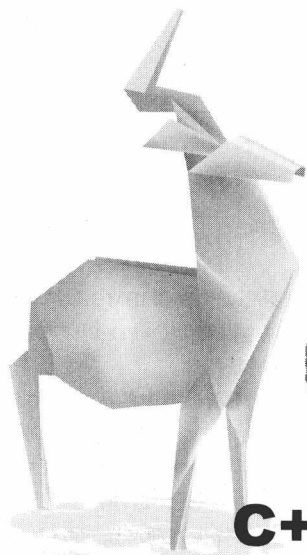
(加) Michael Wong 著
IBM XL编译器中国开发团队



机械工业出版社
China Machine Press

013043552

TP312C
2177



深入理解

C++11

C++11新特性解析与应用

Understanding C++11
Analysis and Application of New Features

(加) Michael Wong
IBM XL编译器中国开发团队 著



北航 C1651723



机械工业出版社
China Machine Press

TP312C

2177

013043222

图书在版编目 (CIP) 数据

深入理解 C++11 : C++11 新特性解析与应用 / Michael Wong, IBM XL 编译器中国开发团队著. —北京 : 机械工业出版社, 2013.6
(原创精品系列)

ISBN 978-7-111-42660-8

I. 深… II. ①M… ②I… III. C 语言 - 程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2013) 第 110482 号

本书法律顾问 北京市展达律师事务所

封底无防伪标均为盗版

版权所有·侵权必究

©Copyright IBM Corp.2013

国内首本全面深入解读 C++11 新标准的专著, 由 C++ 标准委员会代表和 IBM XL 编译器中国开发团队共同撰写。不仅详细阐述了 C++11 标准的设计原则, 而且系统地讲解了 C++11 新标准中的所有新语言特性、新标准库特性、对原有特性的改进, 以及如何应用所有这些新特性。

全书一共 8 章: 第 1 章从设计思维和应用范畴两个维度对 C++11 新标准中的所有特性进行了分类, 呈现了 C++11 新特性的原貌; 第 2 章讲解了在保证与 C 语言和旧版 C++ 标准充分兼容的原则下增加的一些新特性; 第 3 章讲解了具有广泛可用性、能与其他已有的或者新增的特性结合起来使用的、具有普适性的一些新特性; 第 4 章讲解了 C++11 新标准对原有一些语言特性的改进, 这些特性不仅能让 C++ 变得更强大, 还能提升程序员编写代码的效率; 第 5 章讲解了 C++11 在安全方面所做的改进, 主要涵盖枚举类型安全和指针安全两个方面的内容; 第 6 章讲解了为了进一步提升和挖掘 C++ 程序性能和让 C++ 能更好地适应各种新硬件的发展而设计的新特性, 如多核、多线程、并行编程方面的新特性; 第 7 章讲解了一些颠覆 C++ 一贯设计思想的新特性, 如 lambda 表达式等; 第 8 章讲解了 C++11 为了解决 C++ 编程中各种典型实际问题而做出的有效改进, 如对 Unicode 的深入支持等。附录中则介绍了 C++11 标准与其他相关标准的兼容性和区别、C++11 中弃用的特性、编译器对 C++11 的支持情况, 以及学习 C++11 的相关资源。

机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码 100037)

责任编辑: 孙海亮

三河市杨庄长鸣印刷装订厂印刷

2013 年 6 月第 1 版第 1 次印刷

186mm × 240 mm · 20.5 印张

标准书号: ISBN 978-7-111-42660-8

定 价: 69.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88378991 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzsj@hzbook.com

免责声明

本书言论仅为作者根据自身实践经验所得，仅代表个人观点，不代表 IBM 公司的官方立场和看法，特此声明。

IBM XL 编译器中国开发团队简介

IBM 拥有悠久的编译器开发历史（始于 20 世纪 80 年代），在全球拥有将近 400 名高素质工程师组成的研发团队，其中包括许多世界知名的研究学者和技术专家。IBM 一直以来都是编程语言的制定者和倡导者之一，并将长期在编译领域进行研发和投资。IBM XL 编译器中国开发团队于 2010 年在上海成立，现拥有编译器前端开发人员（C/C++）、后端开发人员，测试人员，以及性能分析人员等。团队与 IBM 北美编译器团队紧密合作，共同开发、测试和发布基于 POWER 系统的 AIX 及 Linux 平台下的 XL C/C++ 和 XL Fortran 系列产品，并对其提供技术支持。虽然团队成立时间不长，但已于 2012 年成功发布最新版本的 XL C/C++ for Linux V12.1 & XL Fortran for Linux V14.1，并获得 7 项发明专利。团队成员拥有丰富的编译器开发经验，对编译技术、编程语言、性能优化和并行计算等都有一定的研究，也对 C++11 标准的各种新特性有较早的研究和理解，并正在实际参与 C++11 新特性的开发工作。

欢迎广大读者关注 IBM XL 编译器中国开发团队博客：<http://ibm.co/HK0GCx>，及新浪微博：www.weibo.com/ibmcompiler，并与我们一起学习、探讨 C++11 和编译技术。

作者个人简介

官孝峰 毕业于上海交通大学软件学院。多年致力于底层软件开发，先后供职于 AMD 上海研发中心，IBM 上海研发中心。在嵌入式系统及应用、二进制翻译、图形驱动等领域有丰富的实践经验。2010 年加入 IBM XL 中国编译器中国开发团队，负责 XL C/C++/Fortran 编译器后端开发工作，专注于编译器后端优化、代码生成，以及语言标准等领域。在 C++11 标准制定后，率先对标准进行了全面深入的研究，并组织团队成员对新语言标准进行学习探讨。是数项国内外专利的发明人，并曾于 DeveloperWorks 社区发表英文论文一篇。

陈晶 毕业于西安交通大学通信与信息工程专业，有多年的编译器文档开发写作经验。2008 年起供职于 IBM 上海研发中心，一直致力于研究 C++11 标准的各项新特性，并负责该部分的技术文档撰写。精通 C/C++ 语言，对编译器领域有浓厚的兴趣。负责工作部门内部的编译器产品技术培训。在 DeveloperWorks 社区发表过多篇论文。拥有一项国内专利。

任剑钢 毕业于复旦大学计算机专业。2010 年加入 IBM XL 中国编译器中国开发团队，先后从事 XL Fortran 编译器前端、XL C/C++/Fortran 编译器后端等各种开发工作。对于技术敏感而热衷，擅长 C/C++/Java 等多种编程语言，现专注于编译器代码优化技术。拥有一项专利，并带领团队在 IBM Connections 平台的技术拓展大赛中赢得大奖。

朱鸿伟 毕业于浙江大学计算机科学与技术专业。资深软件开发工程师，有多年系统底层软件开发和 Linux 环境开发经验。一直致力于 C/C++ 语言、编译器、Linux 内核等相关技术的研究与实践，关注新技术和开源社区，对于 Linux 内核以及 Linux 平台软件的开发有着浓厚的兴趣，曾参与 Linux 开源软件的开发与设计。2010 年加入 IBM XL 中国编译器中国开发团队，现专注于编译器前端技术研究与开发工作。

张青山 毕业于福州大学计算机系。从事嵌入式开发多年，曾致力于 Linux 内核和芯片驱动程序的开发、及上层应用程序的编写。2010 年加入 IBM XL 编译器中国开发团队，负责 XL C++ 编译器前端的研发工作。对 C99、C++98、C++11 等语言标准及编译理论有深入理解，

并实际参与 C++11 前端各种特性的实现。此外还致力于编译器兼容性的研究和开发。

纪金松 中国科学技术大学计算机体系结构专业博士，有多年编译器开发经验。2008 年起先后就职于 Marvell（上海）、腾讯上海研究院、IBM 上海研发中心。一直致力于系统底层软件的研究与实践，在编译器后端优化、二进制工具链、指令集优化、可重构计算等相关领域有丰富的实战经验。在国内外杂志和会议中发表过 10 多篇论文，并拥有多项国内外专利。

郭久福 毕业于华东理工大学控制理论与控制工程专业。拥有近 10 年的系统软件开发经验，曾就职于柯达开发中心、HP 中国以及 IBM 中国软件实验室。对 C 语言标准以及 C++ 语言标准有深入研究，近年来致力于编译器前端的开发和研究。

林科文 毕业于复旦大学计算机软件与理论专业，有多年底层系统开发经验。2010 年起供职于 IBM 上海研发中心，现从事编译器后端开发工作。一直致力于系统软件的研究和实践，以及编译器代码优化等领域。活跃于 DeveloperWorks 社区，是三项国内专利的发明人。

班怀芸 毕业于南京理工大学计算机应用技术专业。资深测试工程师，在测试领域耕耘多年，曾任职于 Alcatel-Lucent 公司，有丰富的项目经验。2011 年加入 IBM XL 编译器中国开发团队，现从事和 C/C++ 语言相关的测试领域研究，负责用编译器实现相关需求分析、自动化测试方案制定及实现等工作。持续关注语言和测试技术的发展，对 C/C++ 新特性和语言标准有较深的理解。

蒋健 毕业于复旦大学计算机科学系，资深编译器技术专家。先后供职于 Intel、Marvell、Microsoft 及 IBM 等各公司编译器开发部门，参与并领导业界知名的编译器后端多种相关研发工作，并且拥有近十项编译器方面的专利。现负责 XL C/C++/Fortran 编译器后端开发，并领导 IBM XL 中国编译器中国开发团队各种技术工作。

宋瑞 毕业于北京大学微电子学院，在软件测试领域工作多年，对于测试架构以及软件的发布流程具有丰富的经验，曾就职于 Synopsys 和 Apache design solution。2011 年加入 IBM XL 编译器中国开发团队，从事和编译器、C/C++ 语言相关的研究与测试，对 C/C++ 新语言特性和标准有较深的理解和研究。

刘志鹏 毕业于南京理工大学计算机应用专业，2010 年加入 IBM XL 中国编译器中国开发团队。先后从事 Fortran、C++ 前端的开发工作。现致力于 C++ 语言新标准、语言兼容性等研究与开发。对编译器优化技术、Linux 内核开发有浓厚兴趣，擅长 C/C++/Java 等多种语言。

毛一贻 毕业于上海交通大学，现任职于 IBM 编译器中国开发团队，从事 XL C/C++ 编译器前端的开发工作，在此领域有多年研究经验。具有丰富的实际项目经验并持续关注语言发展。擅长 C++、C、Java 等语言，对 C++ 模板有深入研究。此外对 C#/VB/perl/Jif/Fabric 等语言也均有所涉猎。热爱编程与新技术，活跃于 DeveloperWorks 等社区，发表过技术博客数篇。

张寅林 毕业于上海交通大学信息安全工程专业。2010 年起加入 IBM XL 编译器中国开发团队，专注于编译器后端性能优化开发工作。对于编译器优化算法，Linux 操作系统体系结构与实现有浓厚的兴趣，擅长 C/C++/Python 编程语言。目前从事 IBM 企业级存储服务器的开发工作。

刘林 东南大学计算机科学与工程学院硕士，有多年底层系统开发经验。2010—2012 年间就职于 IBM XL 编译器中国开发团队，先后从事编译器测试及后端开发工作。对嵌入式、Linux 操作系统体系结构与实现有浓厚的兴趣。

Preface

If you are holding this book in the store, you might be wondering why you should read this book among many C++ books. First, you should know this book is about the latest new C++11 (codenamed C++0x) Standard ratified at the end of 2011. This new Standard is almost like a new language, with many new language and library features but there is a strong emphasis for compatibility with the last C++98/03 Standard during design. At the time of printing of this book in 2013, this is one of the first few C++11 books published. All books that do not mention C++11 will invariably be talking about C++98/03.

What makes this book different is that it is written by Chinese writers, in its original Chinese language. In fact, all of us are on the C++ compiler team for the IBM x1C++ compiler, which has been adding C++11 features since 2008.

For my part, I am the C++ Standard representative for IBM and Canada and have been working in compilers for 20 years, and is the author of several C++11 features while leading the IBM C++ Compiler team.

For C++ users who read Chinese, many prefer to read an original Chinese language book, rather than a translated book, even if they can read other languages. While very well written also by experts from the C++ Standard Committee, these non-Chinese books' translation can take time, or result in words or meanings that are loss in translation. The translator has a tough job as technical books contain many jargon and new words that may not have an exact meaning in Chinese. Different translators may not use the same word, even within the same book. These are reasons that lead to a slow dissemination of C++ knowledge and slows the adoption of C++11 in Chinese.

These are all reasons that lead to weak competitiveness. We aim to improve that competitiveness with a book written by Chinese-speaking writers, with a uniform language for jargons, who understand the technology gap as many of the writers work in the IBM Lab in Shanghai. We know there are many Chinese-speaking C++ enthusiasts who are eager to learn and use the updates to their favorite language. The newness of C++11 also demands a strong candidate in the beginner to intermediate level of C++11, which is the level of this book.

You should do well reading this book, if you are:

- an experienced C programmer who wants to see what C++11 can do for you.
- already a C++98/03 programmer who wants to learn the new C++11 language.
- anyone interested in learning the new C++11 language.

We structure this book using the design principles that Professor Bjarne Stroustrup, the father of C++ followed in designing C++11 through the Standard Committee. In fact, we separated this book into chapters based on those design principles. These design principles are outlined in the first chapter. The remaining chapters separate every C++11 language features under those design classifications. For each feature, it will explain the motivation for the feature, the rules, and how it is used, taking from the approved C++11 papers that proposed these features. A further set of appendices will outline the current state of the art of compiler support for C++11, incompatibilities, deprecated features, and links to the approved papers.

After reading this book, you should be able to answer questions such as:

- What is a lambda and the best way to use it?
- How is decltype and auto type inference related?
- What is move semantics and how does it solve the forwarding problem?
- I want to understand default and deleted as well as explicit overrides.
- What did they replace exception specifications with and how does noexcept work?
- What are atomics and the new memory model?
- How do you do parallel programming in C++11?

What we do not cover are the C++11 changes to the Standard library. This part could be a book itself and we may continue with this as Book II. This means we will not talk about the new algorithms, or new class libraries, but we will talk about atomics since most compilers implement atomics as a language feature rather than a library feature for efficiency reason. However, in the C++11 Standard, atomics is listed as a library feature simply because it could be implemented at worst as a library, but very few compilers would do that. This book is also not trying to teach you C++. For that, we particularly recommend Professor Stroustrup's book *Programming principles & Practice Using C++* which is based on an excellent course he taught at Texas A&M University on programming.

This book could be read chapter by chapter if you are interested in every feature of C++11. More likely, you would want to learn about certain C++11 feature and want to target that feature. But while reading about that feature, you might explore other features that fall under the same design guideline.

We hope you find this book useful in your professional or personal learning. We learnt too during our journey of collaborating in writing this book, as we wrote while building the IBM C++ compiler and making it C++11 compliant.

The work of writing a book is long but it is well worth it. While I have been thinking about writing this book while working on the C++ Standard, it was really Xiao Feng Guan who motivated me to start really stop thinking and start doing it for real. He continued to motivate and lead others through their writing assignment process and completed the majority of the work of organizing

this book. I also wish to thank many who have been my mentors officially and unofficially. There are too many to mention but people such as Bjarne Stroustrup, Herb Sutter, Hans Boehm, Anthony Williams, Scott Meyers and many others have been my teachers and great examples of leaders since I started reading their books and watching how they work within large groups. IBM has generously provided the platform, the time, and the facility to allow all of us to exceed ourselves, if only just a little to help the next generation of programmers. Thank you above all to my family Sophie, Cameron, Spot the Cat, and Susan for lending my off-time to work on this book.

Michael

序

当你在书店里拿起这本书的时候，可能最想问的就是：这么多 C++ 的书籍，为什么需要选择这一本？回答这个问题首先需要知道的是，这是一本关于在 2011 年年底才制定通过的 C++11（代码 C++0x）的新标准的书籍。这个新标准看起来就像是一门新的语言，不仅有很多的新语言特性、标准库特性，而且在设计时就考虑了高度兼容于旧有的 C++98/03 标准。在 2013 年出版的 C++ 的书籍中，本书是少数几部关于 C++11 的书籍之一，而其他的，则会是仅讲解 C++98/03 而未提及 C++11 的书籍。

相比于其他书籍，本书还有个显著特点——绝大多数章节都是由中国作者编写。事实上，本书所有作者均来自 IBM XL C++ 编译器开发团队。而团队对于 C++11 新特性的开发，早在 2008 年就开始了。

而我则是一位 IBM 和加拿大的 C++ 标准委员会的代表。我在编译器领域已工作了 20 多年。除了是 IBM C++ 编译器开发团队的领导者之外，还是一些 C++11 特性的作者。

对于使用中文的 C++ 用户而言，很多人还是喜欢阅读原生的中文图书，而非翻译版本，即使是在他们具备阅读其他语言能力的时候。虽然 C++ 标准委员会的专家也在编写一些高质量的书籍，但是书籍从翻译到出版通常需要较长时间，而且一些词语或者意义都可能在翻译中丢失。而翻译者通常也会觉得技术书籍的翻译是门苦差，很多行话、术语通常难以找到准确的中文表达方式。这么一来不同的翻译者会使用不同的术语，即使是在同一本书中，有时同一术语也会翻译成不同的中文。这些状况都是 C++ 知识传播的阻碍，会拖慢 C++11 语言被中国程序员接受的进程。

基于以上种种原因，我们决定本书让母语是中文，并且了解国内外技术差距的 IBM 上海实验室的同事编写。我们知道，在中国有非常多的 C++ 狂热爱好者正等着学习关于自己最爱的编程语言的新知识。而新的 C++11 也会招来大量的初级、中级用户，而本书也正好能满足这些人的需求。

所以，如果你属于以下几种状况之一，将会非常适合阅读本书：

- C 语言经验非常丰富且正在期待着看看 C++11 新功能的读者。
- 使用 C++98/03 并期待使用新的 C++11 的程序员。
- 任何对新的 C++11 语言感兴趣的人。

在本书中，我们引述了 C++ 之父 Bjarne Stroustrup 教授关于 C++11 的设计原则。而事实上，本书的章节划分也是基于这些设计原则的，读者在第 1 章可以找到相关信息，而剩余章节则是基于该原则对每个 C++11 语言进行的划分。对于每个特性，本书将根据其相关的论文展开描述，讲解如设计的缘由、语法规则、如何使用等内容。而书后的附录，则包括当前的

C++11 编译器支持状况、不兼容性、废弃的特性，以及论文的连接等内容。

在读完本书后，读者应该能够回答以下问题：

- 什么是 lambda，及怎么样使用它是最好的？
- decltype 和 auto 类型推导有什么关系？
- 什么是移动语义，以及（右值引用）是如何解决转发问题的？
- default/deleted 函数以及 override 是怎么回事？
- 异常描述符被什么替代了？ noexcept 是如何工作的？
- 什么是原子类型以及新的内存模型？
- 如何在 C++11 中做并行编程？

对于标准程序库，我们在本书中并没有介绍。这部分内容可能会成为我们下一本书的内容。这意味着我们将在下一本书中不仅会描述新的算法、新的类库，还会更多地描述原子类型。虽然出于性能考虑，大多数的编译器都是通过语言特性的方式来实现原子类型的，但在 C++11 标准中，原子类型却被视为一种库特性，因其可以通过库的方式来实现。同样的，这样一本书也不会教读者基础的 C++ 知识，如果读者想了解这方面的内容，我们推荐 Stroustrup 教授的《Programming principles & Practice Using C++》（中文译为：《C++ 程序设计原理与实践》，华章公司已出版）。该书是 Stroustrup 教授以其在德克萨斯 A&M 大学教授的课程为基础编写的。

对 C++11 特性能感兴趣的读者可以顺序阅读本书。当然，读者也可以直接阅读自己感兴趣的章节，但是读者阅读时肯定会发现，这些特性基本和其他的特性一样，遵从了相同的设计准则。

我们也希望本书对你的职业或者个人学习起到积极的作用。当然，我们在合作写作本书，以及在为 IBM C++ 编译器开发 C++11 特性时，也颇有收获。

本书的编写经历了较长的时间，但这是值得的。我在 C++ 标准委员会工作的时候，只是在考虑写这样一本书，而官孝峰则让我从这样的考虑转到了动手行动。继而他还激励并领导其他成员共同参与，最终完成了本书。

此外，我要感谢我的一些正式的以及非正式的导师，比如 Bjarne Stroustrup、Herb Sutter、Hans Boehm、Anthony Williams、Scott Meyers，以及许多其他人，通过阅读他们的著作，或观察他们在委员会中的工作，我学会了很多。当然，更要感谢 IBM 为我们提供的平台、时间，以及各种便利，因为有了这些最终我们能够超越自我，为新一代的程序员做一些事情，即使这样的事情可能微不足道。还要感谢的是我的家人，Sophie、Cameron、Spot（猫）和 Susan，让我能够在空闲时间完成书籍编写。

Michael

前 言

为什么要写这本书

相比其他语言的频繁更新，C++ 语言标准已经有十多年没有真正更新过了。而上一次标准制定，正是面向对象概念开始盛行的时候。较之基于过程的编程语言，基于面向对象、泛型编程等概念的 C++ 无疑是非常先进的，而 C++98 标准的制定以及各种符合标准的编译器的出现，又在客观上推动了编程方法的革命。因此在接下来的很多年中，似乎人人都在学习并使用 C++。商业公司在邀请 C++ 专家为程序员讲课，学校里老师在为学生绘声绘色地讲解面向对象编程，C++ 的书籍市场也是百花齐放，论坛、BBS 的 C++ 板块则充斥了大量各种关于 C++ 的讨论。随之而来的，招聘启事写着“要求熟悉 C++ 编程”，派生与继承成为了面试官审视毕业生基础知识的重点。凡此种种，不一而足。于是 C++ 语言“病毒性”地蔓延到各种编程环境，成为了使用最为广泛的编程语言之一。

十来年的时光转瞬飞逝，各种编程语言也在快马加鞭地向前发展。如今流行的编程语言几乎无一不支持面向对象的概念。即使是古老的语言，也通过了制定新标准，开始支持面向对象编程。随着 Web 开发、移动开发逐渐盛行，一些新流行起来的编程语言，由于在应用的快速开发、调试、部署上有着独特的优势，逐渐成为了这些新领域中的主流。不过这并不意味着 C++ 正在失去其阵地。身为 C 的“后裔”，C++ 继承了 C 能够进行底层操作的特性，因此，使用 C/C++ 编写的程序往往具有更佳运行时性能。在构建包括操作系统的各种软件层，以及构建一些对性能要求较高的应用程序时，C/C++ 往往是最佳选择。更一般地讲，即使是由其他语言编写的程序，往往也离不开由 C/C++ 编写的编译器、运行库、操作系统，或者虚拟机等提供支持。因此，C++ 已然成为了编程技术中的中流砥柱。如果用个比喻来形容 C++，那么可以说这十年来 C++ 正是由“锋芒毕露”的青年时期走向“成熟稳重”的中年时期。

不过十年来对于编程语言来说也是个很长的时间，长时间的沉寂甚至会让有的人认为，C++ 就是这样一种语言：特性稳定，性能出色，易于学习而难于精通。长时间使用 C++ 的程序员也都熟悉了 C++ 毛孔里每一个特性，甚至是现实上的一些细微的区别，比如各种编译器对 C++ 扩展的区别，也都熟稔于心。于是这个时候，C++11 标准的横空出世，以及 C++ 之父 Bjarne Stroustrup 的一句“看起来像一门新语言”的说法，无疑让很多 C++ 程序员有些诚惶诚恐：C++11 是否又带来了编程思维的革命？C++11 是否保持了对 C++98 及 C 的兼容？旧有的 C++ 程序到了 C++11 是否需要被推倒重来？

事实上这些担心都是多余的。相比于 C++98 带来的面向对象的革命性，C++11 带来的

却并非“翻天覆地”式的改变。很多时候，程序员保持着“C++98式”的观点来看待 C++11 代码也同样是合理的。因为在编程思想上，C++11 依然遵从了一贯的面向对象的思想，并深入加强了泛型编程的支持。从我们的观察来看，C++11 更多的是对步入“成熟稳重”的中年时期的 C++ 的一种改造。比如，像 auto 类型推导这样的新特性，展现出的是语言的亲和力；而右值引用、移动语义的特性，则着重于改变一些使用 C++ 程序库时容易发生的性能不佳的状况。当然，C++11 中也有局部的创新，比如 lambda 函数的引入，以及原子类型的设计等，都体现了语言与时俱进的活力。语言的诸多方面都在 C++11 中再次被锤炼，从而变得更加合理、更加条理清晰、更加易用。C++11 对 C++ 语言改进的每一点，都呈现出了经过长时间技术沉淀的编程语言的特色与风采。所以从这个角度上看，学习 C++11 与 C++98 在思想上是一脉相承的，程序员可以用较小的代价对 C++ 的知识进行更新换代。而在现实中，只要修改少量已有代码（甚至不修改），就可以使用 C++11 编译器对旧有代码进行升级编译而获得新标准带来的好处，这也非常具有实用性。因此，从很多方面来看，C++ 程序员都应该乐于升级换代已有的知识，而学习及使用 C++11 也正是大势所趋。

在本书开始编写的时候，C++11 标准刚刚发布一年，而本书出版的时候，C++11 也只不过才诞生了两年。这一两年，各个编译器厂商或者组织都将支持 C++11 新特性作为了一项重要工作。不过由于 C++11 的语言特性非常的多，因此本书在接近完成时，依然没有一款编译器支持 C++11 所有的特性。但从从业者的角度看，C++11 迟早会普及，也迟早会成为 C++ 程序员的首选，因此即使现阶段编译器对 C++ 新特性的支持还不充分，但还是有必要在这个时机推出一本全面介绍 C++11 新特性的中文图书。希望通过这样的图书，使得更多的中国程序员能够最快地了解 C++11 新语言标准的方方面面，并且使用最新的 C++11 编译器来从各方面提升自己编写的 C++ 程序。

读者对象

本书针对的对象是已经学习过 C++，并想进一步学习、了解 C++11 的程序员。这里我们假定读者已经具备了基本的 C++ 编程知识，并掌握了一定的 C++ 编程技巧（对于 C++ 的初学者来说，本书阅读起来会有一定的难度）。通过本书，读者可以全面而详细地了解 C++11 对 C++ 进行的改造。无论是试图进行更加精细的面向对象程序编写，或是更加容易地进行泛型编程，或是更加轻松地改造使用程序库等，读者都会发现 C++11 提供了更好的支持。

本书作者和书籍支持

本书的作者都是编译器行业的从业者，主要来自于 IBM XL 编译器中国开发团队。IBM XL 编译器中国开发团队创立于 2010 年，拥有编译器前端、后端、性能分析、测试等各方面的人员，工作职责涵盖了 IBM XL C/C++ 及 IBM XL Fortran 编译器的开发、测试、发布等与编译器产品相关的方方面面。虽然团队成立时间不长，成员却都拥有比较丰富的编译器开发经验，对 C++11 的新特性也有较好的理解。此外，IBM 北美编译器团队成员 Michael（他是 C++ 标准委员会的成员）也参加了本书的编写工作。在书籍的编写上，Michael 为本书

拟定了提纲、确定了章节主题，并直接编写了本书的首章。其余作者则分别对 C++11 各种新特性进行了详细研究讨论，并完成了书稿其余各章的撰写工作。在书稿完成后，除了请 Michael 为本书的部分章节进行了审阅并提出修改意见外，我们又邀请了 IBM 中国信息开发部及 IBM 北京编译器团队的一些成员对本书进行了详细的审阅。虽然在书籍的策划、编写、审阅上我们群策群力，尽了最大的努力，以保证书稿质量，不过由于 C++11 标准发布时间不长，理解上的偏差在所难免，因此本书也可能在特性描述中存在一些不尽如人意或者错误的地方，希望读者、同行等一一为我们指出纠正。我们也会通过博客（<http://ibm.co/HK0GCx>）、微博（www.weibo.com/ibmcompiler）发布与本书相关的所有信息，并与本书读者共同讨论、进步。

如何阅读本书

读者在书籍阅读中可能会发现，本书的一些章节对 C++ 基础知识要求较高，而某些特性很可能很难应用于自己的编程实践。这样的情况应该并不少见，但这并不是这门语言缺乏亲和力，或是读者缺失了背景知识，这诚然是由于 C++ 的高成熟度导致的。在 C++11 中，不少新特性都会局限于一些应用场景，比如说库的编写，而编写库却通常不是每个程序员必须的任务。为了避免这样的状况，本书第 1 章对 C++11 的语言新特性进行了分类，因此读者可以选择按需阅读，对不想了解的部分予以略过。一些本书的使用约定，读者也可以在第 1 章中找到。

致谢

在这里我们要对 IBM 中国信息开发部的陈晶（作者之一）、卢昉、付琳，以及 IBM 北京编译器团队的冯威、许小羽、王颖对本书书稿详尽细致的审阅表示感谢，同时也对他们专业的工作素养表示由衷的钦佩。此外，我们也要感谢 IBM XL 编译器中国开发团队的舒蓓、张嗣元两位经理在本书编写过程中给予的大力支持。而 IBM 图书社区的刘慎峰及华章图书的杨福川编辑的辛勤工作则保证了本书的顺利出版，在这里我们也要对他们以及负责初审工作的孙海亮编辑说声谢谢。此外，我们还要感谢各位作者的家人在书籍编写过程中给予作者的体谅与支持。最后要感谢的是本书的读者，感谢你们对本书的支持，希望通过这本书，我们能够一起进入 C++ 编程的新时代。

IBM XL 编译器中国开发团队

目 录

免责声明

序

前言

第 1 章 新标准的诞生..... 1

1.1 曙光：C++11 标准的诞生..... 1

1.1.1 C++11/C++0x（以及 C11/C1x）
——新标准诞生..... 1

1.1.2 什么是 C++11/C++0x..... 2

1.1.3 新 C++ 语言的设计目标..... 3

1.2 今时今日的 C++..... 5

1.2.1 C++ 的江湖地位..... 5

1.2.2 C++11 语言变化的领域..... 5

1.3 C++11 特性的分类..... 7

1.4 C++ 特性一览..... 11

1.4.1 稳定性与兼容性之间的抉择..... 11

1.4.2 更倾向于使用库而不是扩展
语言来实现特性..... 12

1.4.3 更倾向于通用的而不是特殊
的手段来实现特性..... 13

1.4.4 专家新手一概支持..... 13

1.4.5 增强类型的安全性..... 14

1.4.6 与硬件紧密合作..... 14

1.4.7 开发能够改变人们思维方式
的特性..... 15

1.4.8 融入编程现实..... 16

1.5 本书的约定..... 17

1.5.1 关于一些术语的翻译..... 17

1.5.2 关于代码中的注释..... 17

1.5.3 关于本书中的代码示例与实验

平台..... 18

第 2 章 保证稳定性和兼容性..... 19

2.1 保持与 C99 兼容..... 19

2.1.1 预定义宏..... 19

2.1.2 `__func__` 预定义标识符..... 20

2.1.3 `_Pragma` 操作符..... 22

2.1.4 变长参数的宏定义以及
`__VA_ARGS__`..... 22

2.1.5 宽窄字符串的连接..... 23

2.2 `long long` 整型..... 23

2.3 扩展的整型..... 25

2.4 宏 `__cplusplus`..... 26

2.5 静态断言..... 27

2.5.1 断言：运行时与预处理时..... 27

2.5.2 静态断言与 `static_assert`..... 28

2.6 `noexcept` 修饰符与 `noexcept`
操作符..... 32

2.7 快速初始化成员变量..... 36

2.8 非静态成员的 `sizeof`..... 39

2.9 扩展的 `friend` 语法..... 40

2.10 `final/override` 控制..... 44

2.11 模板函数的默认模板参数..... 48

2.12 外部模板..... 50

2.12.1 为什么需要外部模板..... 50

2.12.2 显式的实例化与外部模板
的声明..... 52

2.13 局部和匿名类型作模板实参..... 54

2.14 本章小结..... 55

第 3 章 通用为本，专用为末..... 57

3.1 继承构造函数..... 57

3.2 委派构造函数	62	第 5 章 提高类型安全	155
3.3 右值引用：移动语义和完美转发	68	5.1 强类型枚举	155
3.3.1 指针成员与拷贝构造	68	5.1.1 枚举：分门别类与数值的名字	155
3.3.2 移动语义	69	5.1.2 有缺陷的枚举类型	156
3.3.3 左值、右值与右值引用	75	5.1.3 强类型枚举以及 C++11 对原有 枚举类型的扩展	160
3.3.4 std::move：强制转化为右值	80	5.2 堆内存管理：智能指针与垃圾 回收	163
3.3.5 移动语义的一些其他问题	82	5.2.1 显式内存管理	163
3.3.6 完美转发	85	5.2.2 C++11 的智能指针	164
3.4 显式转换操作符	89	5.2.3 垃圾回收的分类	167
3.5 列表初始化	92	5.2.4 C++ 与垃圾回收	169
3.5.1 初始化列表	92	5.2.5 C++11 与最小垃圾回收支持	170
3.5.2 防止类型收窄	96	5.2.6 垃圾回收的兼容性	172
3.6 POD 类型	98	5.3 本章小结	173
3.7 非受限联合体	106	第 6 章 提高性能及操作硬件的 能力	174
3.8 用户自定义字面量	110	6.1 常量表达式	174
3.9 内联名字空间	113	6.1.1 运行时常量性与编译时常量性	174
3.10 模板的别名	118	6.1.2 常量表达式函数	176
3.11 一般化的 SFINEA 规则	119	6.1.3 常量表达式值	178
3.12 本章小结	121	6.1.4 常量表达式的其他应用	180
第 4 章 新手易学，老兵易用	123	6.2 变长模板	183
4.1 右尖括号 > 的改进	123	6.2.1 变长函数和变长的模板参数	183
4.2 auto 类型推导	124	6.2.2 变长模板：模板参数包和函 数参数包	185
4.2.1 静态类型、动态类型与类型 推导	124	6.2.3 变长模板：进阶	189
4.2.2 auto 的优势	126	6.3 原子类型与原子操作	196
4.2.3 auto 的使用细则	130	6.3.1 并行编程、多线程与 C++11	196
4.3 decltype	134	6.3.2 原子操作与 C++11 原子类型	197
4.3.1 typeid 与 decltype	134	6.3.3 内存模型，顺序一致性与 memory_order	203
4.3.2 decltype 的应用	136	6.4 线程局部存储	214
4.3.3 decltype 推导四规则	140	6.5 快速退出：quick_exit 与 at_quick_exit	216
4.3.4 cv 限制符的继承与冗余的符号	143	6.6 本章小结	219
4.4 追踪返回类型	145		
4.4.1 追踪返回类型的引入	145		
4.4.2 使用追踪返回类型的函数	146		
4.5 基于范围的 for 循环	150		
4.6 本章小结	153		

第 7 章 为改变思考方式而改变.....	220	第 8 章 融入实际应用.....	258
7.1 指针空值—— <code>nullptr</code>	220	8.1 对齐支持.....	258
7.1.1 指针空值：从 0 到 <code>NULL</code> ， 再到 <code>nullptr</code>	220	8.1.1 数据对齐.....	258
7.1.2 <code>nullptr</code> 和 <code>nullptr_t</code>	223	8.1.2 C++11 的 <code>alignof</code> 和 <code>alignas</code>	261
7.1.3 一些关于 <code>nullptr</code> 规则的讨论.....	225	8.2 通用属性.....	267
7.2 默认函数的控制.....	227	8.2.1 语言扩展到通用属性.....	267
7.2.1 类与默认函数.....	227	8.2.2 C++11 的通用属性.....	268
7.2.2 “= default” 与 “= deleted”.....	230	8.2.3 预定义的通用属性.....	270
7.3 <code>lambda</code> 函数.....	234	8.3 Unicode 支持.....	274
7.3.1 <code>lambda</code> 的一些历史.....	234	8.3.1 字符集、编码和 Unicode.....	274
7.3.2 C++11 中的 <code>lambda</code> 函数.....	235	8.3.2 C++11 中的 Unicode 支持.....	276
7.3.3 <code>lambda</code> 与仿函数.....	238	8.3.3 关于 Unicode 的库支持.....	280
7.3.4 <code>lambda</code> 的基础使用.....	240	8.4 原生字符串字面量.....	284
7.3.5 关于 <code>lambda</code> 的一些问题及 有趣的实验.....	243	8.5 本章小结.....	286
7.3.6 <code>lambda</code> 与 STL.....	247	附录 A C++11 对其他标准的 不兼容项目.....	287
7.3.7 更多的一些关于 <code>lambda</code> 的讨论.....	254	附录 B 弃用的特性.....	294
7.4 本章小结.....	256	附录 C 编译器支持.....	301
		附录 D 相关资源.....	304