Broadview
www.broadview.com.cn
Pearson
· 原味精品书系 ·

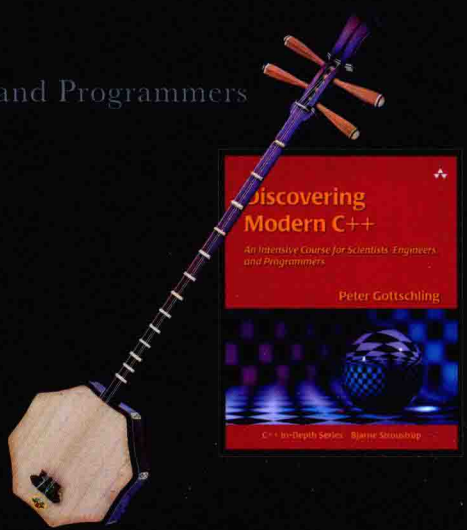
现代C++探秘（英文版）

编码、工程与科研必修（基于C++14）

Discovering Modern C++

An Intensive Course for Scientists, Engineers, and Programmers

[德] Peter Gottschling 著



中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

· 原味精品书系 ·

现代C++探秘（英文版）

编码、工程与科研必修（基于C++ 14）

Discovering Modern C++
An Intensive Course for Scientists, Engineers, and Programmers

[德] Peter Gottschling 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内容简介

如今科学工程项目越来越大、越来越复杂，许多项目都采用 C++ 编程语言来完成。本书深入介绍了基于 C++ 编程语言高级功能的复杂方法，旨在帮助您快速入门，实现如表达式模板之类的高级技术。您还将学习如何使用 C++ 编程语言的强大类库：标准模板库（STL）以及用于算法、线性代数、微分方程、图形的科学类库。书中演示了如何使用面向对象、泛型编程、元编程和过程技术来编写清晰明了、富有表达力的软件。当您学完本书，将掌握如何使用 C++ 编程语言来编写高质量、高性能的软件。

Original edition, entitled Discovering Modern C++: An Intensive Course for Scientists, Engineers, and Programmers 9780134383583 by Peter Gottschling, published by Pearson Education, Inc., publishing as Addison-Wesley Professional, Copyright © 2016 Pearson Education,Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education,Inc.

China edition published by Pearson Education Asia Ltd. and Publishing House of Electronics Industry Copyright © 2017.The edition is manufactured in the People's Republic of China, and is authorized for sale and distribution only in the mainland of China exclusively(except Hong Kong SAR, Macau SAR, and Taiwan).

本书英文影印版专有出版权由 Pearson Education 培生教育出版亚洲有限公司授予电子工业出版社。未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书仅限中国大陆境内（不包括中国香港、澳门特别行政区和中国台湾地区）销售发行。

本书英文影印版贴有 Pearson Education 培生教育出版集团激光防伪标签，无标签者不得销售。

版权贸易合同登记号　图字：01-2016-1168

图书在版编目（CIP）数据

现代 C++ 探秘：编码、工程与科研必修：基于 C++ 14：英文 /（德）彼得 · 哥特史林（Peter Gottschling）著. — 北京：电子工业出版社，2017.4
（原味精品书系）
书名原文：Discovering Modern C++: An Intensive Course for Scientists, Engineers, and Programmers
ISBN 978-7-121-30854-3
I. ① 现…II. ① 彼…III. ① C 语言－程序设计－英文 IV. ① TP312.8
中国版本图书馆 CIP 数据核字（2017）第 018608 号

策划编辑：刘佳禾
责任编辑：徐津平
印　　刷：三河市良远印务有限公司
装　　订：三河市良远印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980 1/16　　印张：29　　字数：576 千字
版　　次：2017 年 4 月第 1 版
印　　次：2017 年 4 月第 1 次印刷
定　　价：108.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
本书咨询联系方式：（010）51260888-819　faq@phei.com.cn。

献给我的父母：Helga 和 Hans-Werner。

用户注册

轻松注册成为博文视点社区用户（www.broadview.com.cn），您即可享受以下服务。

- **下载资源**：本书所提供的示例代码及资源文件均可在【下载资源】处下载。
- **提交勘误**：您对书中内容的修改意见可在【提交勘误】处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **与我们交流**：在页面下方【读者评论】处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：*http://www.broadview.com.cn/30854*

二维码：

此为试读，需要完整PDF请访问：www.ertongbook.com

前言

世界由 C++（以及它的 C 子集）构建。

——Herb Sutter

Google、Amazon 和 Facebook 的基础架构很多都由 C++ 构建。此外，相当一部分底层技术也是由 C++ 实现的。在电信领域，几乎所有固定电话和手机的连接都由 C++ 软件驱动。最重要的是，德国所有主要传输节点都是用 C++ 处理的，这意味着我的家庭也依赖于 C++ 软件。

即便是由其他语言撰写的软件也会依赖于 C++，因为最流行的编译器，例如 Visual Studio、Clang、GNU 编译器包和 Intel 编译器，都是用 C++ 实现的。Windows 平台上的程序也多由 C++ 实现，如 Microsoft Office 套件。可以说 C++ 是无所不在的。甚至您的手机和汽车也会包含由 C++ 开发的组件。C++ 的发明者 Bjarne Stroustrup 制作了一个网页，列出了由 C++ 开发的应用清单，上文的例子也多源于这个网页。

在科学和工程领域，许多高质量的软件包都是用 C++ 实现的。在项目超过一定大小且数据结构非常复杂的情况下更能凸显 C++ 的强大能力。这也是大量科学和工程模拟软件都使用 C++ 的原因。随便举几个例子，比如该领域的领头羊 Abaqus、deal.II、FEniCS 以及 OpenFOAM，知名 CAD 软件 CATIA 也由 C++ 开发。得益于更强大的处理器和改进的编译器，C++ 在嵌入式系统上的应用也越来越多。当然，并不是所有的现代语言特性和库都可以用在这些平台上。

最后，如果一个项目可以重新开发，则不知道有多少项目会使用 C++ 而不是 C 实现。例如，笔者的好友 Matt Knepley 是非常成功的科学计算程序库 PETSc 的作者之一，如果可以重写这个库，他会选择 C++ 来实现它。

学习 C++ 的理由

和其他语言不同，从充分贴近硬件的开发到高级抽象的开发，C++ 几乎都能胜任。底层开发——比如用户定制的内存管理（user defined memory management）——可以让程序员了解程序执行过程中的细节，以及由其他语言开发的程序的行为。利用 C++ 可以编写执行效率非常高的程序，仅仅可能比使用机器语言编写的程序慢一点点，但是后者要付出更多的努力。然而，在死磕性能调优之前（hardcore performance tuning），您应该首先关注如何开发逻辑清晰、表达准确的程序。

而这正是C++高层特性的用武之地。C++直接支持多种编程范式（programming paradigm）：面向对象编程（object oriented programming）（见第 6 章）、泛型编程（generic programming）（见第 3 章）、元编程（metaprogramming）（见第 5 章）、并发编程（concurrent programming）（见 4.6 节）和过程化编程（procedural programming）（见 1.5 节），等等。C++还发明了好几种编程技术，例如raii（见2.4.2.1节）和表达式模板（expression template）（见 5.3 节）。C++ 的表达能力如此之强，故而经常可以在不改变语言的情况下发明新的技术。也许有一天您也会发明一种新的技术。

阅读本书的理由

本书素材已经经过了多人的检验。作者执教课程 C++ for Scientists 已有三年，每年两个学期。这门课的学生多来自数学系，也有一些来自物理和工程专业。在课程学习之前，他们通常没有 C++ 基础，但在学完这门课程后就可以实现如表达式模板（见 5.3 节）之类的高级技术。您可以按照自己的节奏阅读本书：直接跟着本书路线完成学习，或者通过阅读附录 A 了解更多示例和背景知识。

美女与野兽

编写 C++ 程序有多种方式，本书将循序渐进地指导读者使用更加高级的风格。这需要使用 C++ 的高级特性。这些特性看起来挺吓人，但是习惯之后，它们不仅应用更加广泛，也更为高效且易读。

我们用下面这个例子作为您对 C++ 的第一印象：恒定步长的梯度下降法。原理非常简单，用其梯度函数（例如 $g(x)$）来计算 $f(x)$ 的最陡下降，并且用固定大小的步长追随这个梯度方向到相邻的局部最小值。算法的伪代码也非常简单：

算法 1：梯度下降算法

输入：初始值 x，步长 s，终止条件 ε，函数 f，梯度函数 g
输出：局部最小值 x

do
  $x = x - s \cdot g(x)$
while $|\Delta f(x)| \geqslant \varepsilon$;

对于这个简单算法我们提供了两种实现。看以下代码并进行思考，不考虑它们的技术细节。

```
void gradient_descent(double* x,
    double* y, double s, double eps,
    double(*f)(double, double),
    double(*gx)(double, double),
    double(*gy)(double, double))
{
    double val= f(*x, *y), delta;
    do {
        *x-= s * gx(*x, *y);
        *y-= s * gy(*x, *y);
        double new_val= f(*x, *y);
        delta= abs(new_val - val);
        val= new_val;
    } while (delta > eps);
}
```

```
template <typename Value, typename P1,
          typename P2, typename F,
          typename G>
Value gradient_descent(Value x, P1 s,
    P2 eps, F f, G g)
{
    auto val= f(x), delta= val;
    do {
        x-= s * g(x);
        auto new_val= f(x);
        delta= abs(new_val - val);
        val= new_val;
    } while (delta > eps);
    return x;
}
```

它们看起来非常相似，但我们会告诉您哪一个是我们喜欢的。左边版本差不多是纯 C 语言编写的，您可以用 C 编译器编译。这个版本的好处就是看起来很直观，使用 double 类型的 2D 函数。但是我们更喜欢右边版本，因为它用途更广，是具有任意值类型的任意维度函数。令人惊奇的是，这个多功能的版本并没有降低效率。相反，函数 F 和 G 可以内联（见 1.5.3 节）以节省函数调用开销，而在左边版本中显式的函数指针使得这种优化变得更困难。

如果您是一位有耐心的读者，那么可以在附录 A 中（A.1 节）找到一个比较新旧样式的更详细的例子。附录中的案例更能体现使用现代风格编程的好处。当然，我们不希望您为这些新旧样式的小冲突折腾太久。

科学和工程领域的计算机语言

“如果所有的数值类软件都可以用 C++ 编写而不损失效率，那自然很好。但在我们找到一种既能高效计算，又不违反 C++ 类型系统的方法之前，还是用 Fortran、汇编，或者针对体系结构优化过的方法更好。”

——Bjarne Stroustrup

科学计算和工程软件可以使用不同的编程语言编写，而哪种语言最合适则主要取决于我们的目标和可用的资源。

- 当我们仅使用现有算法时，最好选择数学工具软件，如 MATLAB、Mathematica、R 等。但如果希望在它们的基础上通过一些细粒度的运算（如标量计算）实现自己的算法，则会发现其性能将显著下降。当然如果问题本身很小，或者用户非常有“耐心”，这自然不是问题。否则，就需要寻找其他语言来解决这个问题。
- Python 非常适合快速开发。它也包含了大量科学计算的库，比如 scipy 和 numpy。使用这些库（这些库本身通常由 C 或者 C++ 实现）也能开发出很高效的应用程序。和数学工具软件一样，细粒度实现的用户定义算法也会牺牲性能。如果要快速实现中小型任务，Python 非常适合。但如果项目足够大，编译器的重要性就会逐渐体现出来（例如当参数不匹配时拒绝赋值）。
- Fortran 也会适用于工程和科学计算，因为 Fortran 上拥有大量经过充分优化的操作，如密集矩阵运算。它非常适合完成那些老派教授的作业（这些作业本身很适合 Fortran）。而依据作者的经验，在 Fortran 中引入新的数据结构非常麻烦，因此在 Fortran 中编写大规模的模拟程序是一个相当大的挑战 — 在今天，这样的选题只有少数人能勉为其难地接受。
- C 语言有着很好的性能，用 C 编写的软件数量非常多。C 的语言核心小，易于学习。使用 C 语言开发的挑战在于，需要使用简陋而危险的语言特性，例如指针（见 1.8.2 节）和宏（见 1.9.2.1 节），去实现大型的、无 Bug 的软件。
- 当应用程序的主要组件是 Web 或图形界面并且没有特别密集的运算时，Java、C#、PHP 这一类的语言是非常好的选择。
- 开发大型、高质量、高性能的应用程序是 C++ 尤为擅长的。并且开发过程也不会十分漫长而痛苦。通过正确的抽象，可以飞快地编写 C++ 程序。在未来会有更多的科学计算库出现在 C++ 标准中。

显然，我们所了解的语言越多，选择的余地也就越大；对语言了解得越深入，我们做出的选择也越明智。此外，大型项目也常包含多个用不同语言开发的组件，多数情况下起码性能关键的内核部分都是由 C 或者 C++ 实现的。总而言之，学习 C++ 一定是一段有趣的过程，深入理解它，一定能帮助您成为一名出色的程序员。

体例

新术语使用斜体（*italic*）。C++ 源代码使用等宽字符（`monospace`）。一些重要的细节被标记为粗体（**boldface**）。类、函数、变量与常量一律使用小写字母，也可能会用到下画线。矩阵使用特殊的标记，我们一般使用单个大写字母来代表矩阵。模板参数和 Concept 以大写开头，大写分隔（CamelCase）。程序输出和命令行使用打字机字体（`typewriter font`）。

如果程序需要 C++3、C++11 或者 C++14 的特性，会用相应的边框来指出。一部分程序仅使用了少量 C++11 的特性，且很容易替换成 C++3 的实现，对于它们我们不再另行标注。

⇒ `directory/source_code.cpp`

除了一些特别短小的演示代码，所有编程示例都至少在一个编译器上测试过。在段落或小节的开头，我们会用箭头指示出所讨论的代码的路径。

所有程序都开放在 GitHub 上：`https://github.com/petergottschling/discovering_modern_cpp`。您可以使用以下命令来复制代码仓库：`git clone https://github.com/petergottschling/discovering_modern_cpp.git`。

在 Windows 上还可以使用更方便的工具 TortoiseGit（`tortoisegit.org`）。

致谢

按照时间顺序，我首先想感谢 Karl Meerbergen 和他的同事们。本书起始于我和 Karl 在 KU Leuven 执教时所使用的 80 页讲义。随着时间的推移，其中多数段落已被重写，但是初始版本仍然是本书写作过程中的原动力。在 7.1 节 ODE 解算器（Solver）实现的部分，我欠了 Mulansky 一个大人情。

非常感谢 Jan Christiaan van Winkel 和 Fabio Fracassi，他们对手稿的每一个细节都进行了仔细检查，并在 C++ 标准的合规性和行文的可理解性上提出了许多建议。

特别感谢 Bjarne Stroustrup 在成书上所给予的策略上的提示，助我与 Addison-Wesley 建立联系，并且慷慨地允许我使用他已有的材料，以及 — 创造了 C++。所有这些审校者都敦促我尽可能地使用 C++11 和 C++14 的特性来取代以前的讲稿内容。

此外，我还要感谢 Karsten Ahnert 的建议和 Markus Abel 帮助我精炼原本冗长的前言。

当我为 4.2.2.6 节寻找一个有趣的随机数应用时，Jan Rudl 建议我可以使用他在课程教学中所使用的股价演变的案例。

感谢德累斯顿工业大学，让我得以在数学部门教授了三年多的 C++，并且感谢我的学生们在课程中给予了我建设性的反馈。同样，非常感谢参与我的 C++ 培训的学员们。

十分感谢编辑 Greg Doench 在本书中接受我既严肃又轻松的风格，也感谢他长期与我讨论写作方案直到我们都满意为止。同时也感谢他提供的专业支持，没有他这本书永远不可能出版。

Elizabeth Ryan 负责管理全书的制作过程，同时耐心地满足了我所有的特殊要求。

最后，我衷心感谢我的家人 — 我的妻子 Yasmine，以及我的孩子 Yanis、Anissa、Vincent 和 Daniel — 你们为我牺牲了原本全家人共度的时光，令我可以将时间投入到本书的写作上。

此为试读，需要完整PDF请访问：www.ertongbook.com

关于作者

Peter Gottschling 热衷于编写前沿的科学计算软件，他希望他的热情也能感染读者。因为职业的缘故他编写了 MTL4（矩阵模板库 4），同时也是 Boost Graph Library 的作者之一。他曾在多个 C++ 课程和专业培训中分享过开发经验，并撰写了本书。

他是 C++ 标准委员会成员，德国程序语言标准委员会副主席，也是德累斯顿 C++ 用户组的创始人。他年轻时在德累斯顿工业大学就读，同时在数学和计算机科学专业上达到了学士水平，并最终获得了计算机科学的博士学位。莱比锡建城一千年时，他离开了学术机构，回到了他最爱的故乡莱比锡，创建了自己的公司 SimuNova。

他已婚并育有四名子女。

Contents

此为试读,需要完整PDF请访问: www.ertongbook.com