计 算 机 科 学 丛 书

C语言程序设计进阶教程

[美] 陆永祥（Yung-Hsiang Lu） 普渡大学 著
徐东 北京航空航天大学 译

Intermediate C Programming

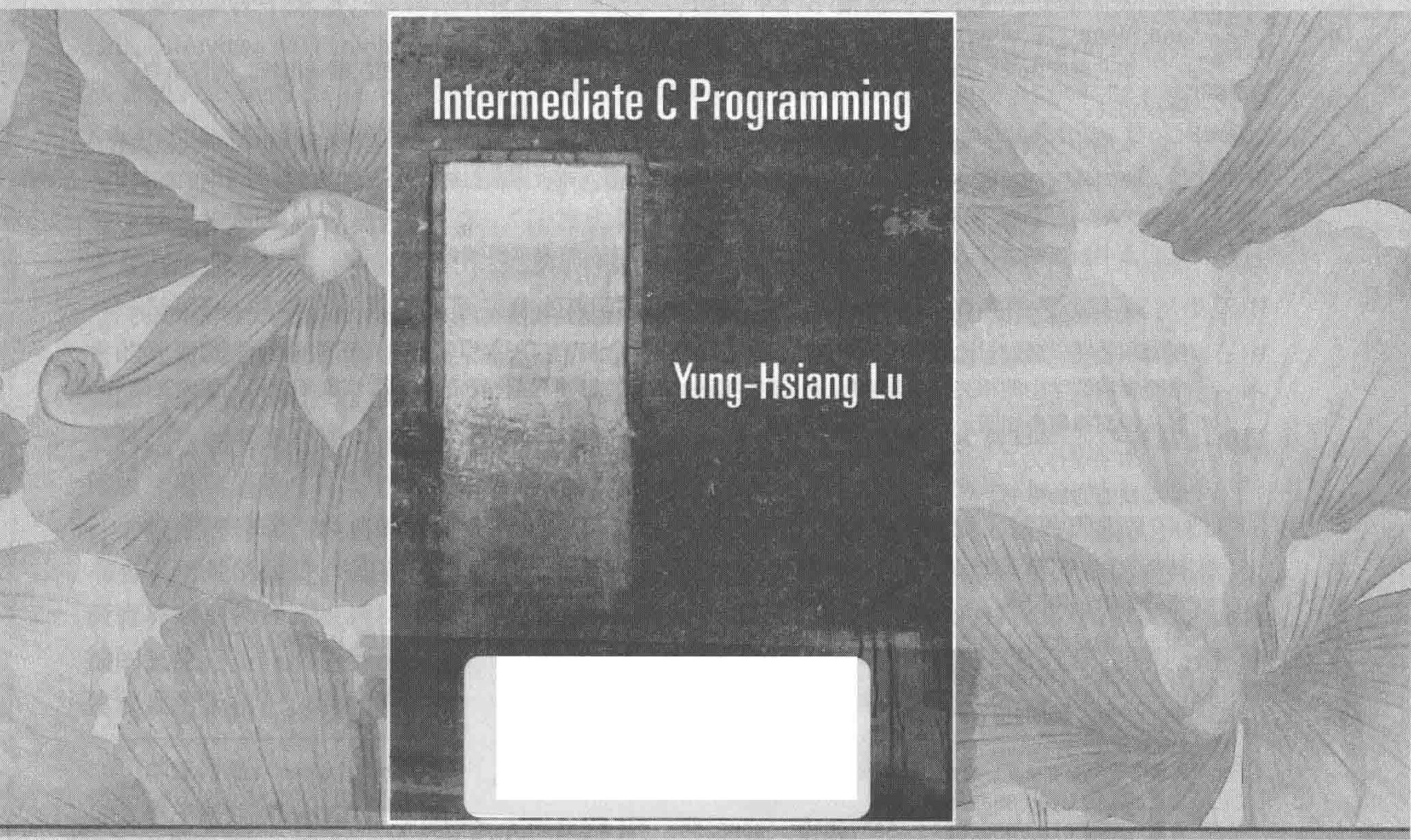
机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

C语言程序设计进阶教程

[美] 陆永祥（Yung-Hsiang Lu） 著

Intermediate C Programming



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

C 语言程序设计进阶教程 /（美）陆永祥（Yung-Hsiang Lu）著；徐东译 . —北京：机械工业出版社，2017.6
（计算机科学丛书）
书名原文：Intermediate C Programming

ISBN 978-7-111-56840-7

I. C…　II. ① 陆…　② 徐…　III. C 语言 - 程序设计 - 高等学校 - 教材　IV. TP312.8

中国版本图书馆 CIP 数据核字（2017）第 109706 号

本书版权登记号：图字：01-2016-6030

本书讲解了栈、堆、指针、文件等各类编程概念和数据结构及其应用，介绍如何成为优秀程序员的经验和技巧，通过对比编程中的常见错误与正确的程序之间的区别来提高编程人员的技能。特别地，本书将离散数学中的相关概念与程序设计紧密相连，细致地阐述递归程序的思想、实现和应用，使读者能够从中习得更多知识，掌握高级编程技巧。

本书可作为高等院校 C 语言相关课程的本科生教材，也可作为中等编程水平的程序设计人员提升编程技能的参考书。

出版发行：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：唐晓琳　　责任校对：殷　虹
印　　刷：北京文昌阁彩色印刷有限责任公司　　版　　次：2017 年 7 月第 1 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：25.25
书　　号：ISBN 978-7-111-56840-7　　定　　价：79.00 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：hzjsj@hzbook.com

出版者的话

Intermediate C Programming

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brian W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com

电子邮件：hzjsj@hzbook.com

联系电话：（010）88379604

联系地址：北京市西城区百万庄南街 1 号

邮政编码：100037

华章教育

华章科技图书出版中心

译 者 序

Intermediate C Programming

2013 年我到普渡大学陆教授的实验室做访问学者时，读到了《 Intermediate C Programming 》这本书的初稿。作为一个多年使用 C 语言基于嵌入式系统编程的科研人员，读完之后我直接找到陆教授，问他能不能把这本书翻译成中文出版。当时本书英文版还没有正式出版，我们便商定说等书出版后，我再联系出版社商量翻译事项。直至今天，我终于结束了翻译工作，也算完成了一个心愿。

读完本书的初稿，我为什么就会产生把它译成中文出版的冲动呢？因为基于自己使用 C 语言的体会，我产生了强烈的共鸣。读完之后，我觉得十分痛快，用这个词来描述当时的感受挺恰当。就好像有一个问题，自己已经理解到了意会的程度；在和一个朋友讨论时，他条理清晰地把问题言传出来了。本书所编排的内容，就是自己在使用 C 语言进行开发的过程中逐渐深入理解的内容，我相信这也是 C 语言使用者在入门之后急需学习的内容。

本书是作者十多年教学经验积累的成果，内容的编排、例程的选择、论述的角度都渗透着美国大学课堂教学中的轻松却严谨的特色。程序是算法 + 数据结构。作者从 C 语言中数据的组织形式入手，用对“栈”这一重要概念的讲解开始了 C 语言的进阶，进而对堆、指针、文件进行了深入形象的阐述。此后，第二部分展开分析了递归程序的思想、实现和应用，用一个典型的问题具体分析了 C 语言函数的特点。以此为基础，第三部分讨论了几种数据结构的编程应用。最后一部分用应用实例加深读者对前面内容的理解。这种结构的编排体现了本书提高读者 C 语言编程水平的宗旨，使读者能够循序渐进地理解和掌握 C 语言的内涵。

我的研究生彭祥、黄倩、曾海宁、吕志宇承担了部分书稿的翻译和校对，在本书的翻译工作中付出了辛勤的劳动，在此表示感谢。还要感谢机械工业出版社华章公司对本书翻译出版的支持。由于译者的水平有限，尽管翻译过程中谨慎动笔，仔细校对，但难免还会存在疏漏，恳请广大读者批评指正。

徐　东

于北航　主北 106

序

Intermediate C Programming

想象你在管理一个研究小组或者开发小组，通过编写软件来检验新的物理现象或设计。你在寻找在特定物理学或者科学领域有技术背景的学生或者雇员，同时也希望他们有一些软件项目经验。通常你会发现学生上过编程课或者修补过一些小程序，但是大体上他们从未写过复杂的程序，也从未与软件团队一起工作过，并且他们会害怕钻研已经开发的科学软件代码。

这就是我现在的处境。我的研究小组在为你未来的电脑提供动力的晶体管中研究纳米级别电子流。作为教学人员，我发现如今大部分毕业于工程或者物理科学专业的本科学生习惯用脚本语言写小程序，但是对编译、实际调试程序或者良好的编程实践却还是很不熟悉。

我相信这种情况肯定不是特例，而是在学术界或者工业界中普遍存在。如何让这些新手快速进入状态？如何能快速地向他们传递我通过很多年的实验经验才得到的日常性的洞察力？

大多数高级编程书籍会介绍正确优美的复杂程序或者是架构很大的程序。可以用读一本写得很好的书和自己创作一篇小说做个类比。文学分析可以帮助读者鉴赏小说的内容或背景。很多人都能够正确地阅读用 C 语言写出的算法，但是即使在给定伪代码（故事线）的时候也很少有人能写出这段代码。本书为你写出自己的 C 语言代码提供了有效途径。

我相信这本新书为进入实际软件开发实践提供了一个绝佳的入口，它会帮助新生和高年级学生在他们的日常工作中通过避免一些典型的错误和写一些更清晰的代码让工作更富有成效，因为他们更好地理解了代码潜在的含义。本书同时通过典型的代码风格和实践使团队内的合作更便利。

本书解释了检测隐藏问题的重要性。大多数学生会犯的错误就是他们只关注表面，也就是他们编写的程序输出的结果。他们的程序内部含有很多很严重的问题。如果程序在少数情况下输出了正确的结果，学生就会认为程序是对的。这在这个彼此连通的世界中是很危险的。一个粗心的小错误也许会成为大型复杂系统的一个安全隐患。创建一个安全可靠的系统首先需要关注细节。本书论述了很多粗心的小错误导致严重问题的细节，这对加深对隐藏问题的理解很有用处。

希望在 20 年前，在读完 Kernighan 和 Richie 写的两本书之后，我就能够拥有这本书。在那个时候在我开始用 C 语言写大量的代码之前一直都是用 FORTRAN。传引用、传值这些简单的概念都在本书中以合理的形式列出了。我很喜欢书里可以实际动手操作的例子，这些例子非常有启发性。

我把这本书推荐给任何想要写软件并且高于修补代码水平的人。你会学习如何更好地编程，如何识别和减少错误。你还会学习如何写清晰的代码，因此这些代码可以被调用数以百万次而不会把你自己的或者别人的电脑弄崩溃。你会学习如何与别人分享代码。同时你将开始使用标准的基于 Linux 的工具，比如 ddd、valgrind 等。

Gerhard Klimeck
普渡大学电机与计算机工程学院教授
可预测材料与设备中心（c-primed）主任，计算网络和计算中心（NCN）主任
英国皇家物理学会（IOP）会员，美国物理学会（APS）会员，
电气与电子工程师协会（IEEE）会员

前　言

Intermediate C Programming

为什么要写这本书

市面上有成百上千种关于编程的书籍，其中有很多都是关于C语言编程的，那么为什么我还要写这本书呢？为什么建议你花时间读它呢？这本书跟其他书有什么不同呢？跟很多作者一样，我写这本书是因为我觉得有必要，觉得这本书中的方法比其他书中的更好。

我将现在已有的关于编程的书分为两类：入门和进阶。入门类书是给初学者写的，一般都假设读者没有编程基础，所以主要是介绍基本的概念。通常以“Hello World!”程序开始，也就是将“Hello World!”输出到电脑屏幕的程序。这种类型的书主要是一步步地介绍语言特点，包括关键词、数据类型、控制结构、字符串、文件操作等，而这些书一般都有一个特点：程序很短，一般是1～2页。这很奏效，因为短程序有助于解释编程语言的新概念。如果把学编程语言比作学自然语言，如英语、汉语、法语、韩语等，这些书就相当于教导如何造句和撰写短段落。

第二类书是写给有程序开发经验的读者的。这些书主要介绍解决现实中的问题的程序，比如关于电脑游戏或者图像。而这类书的例子一般很长，有些甚至几千行代码，因此不会全部印在书本上。书中只会解释程序的其中一部分，而源程序一般保存在CD或者某个网址上。这类书一般不会再介绍如何编程，而是大多专注于解决特定问题的算法研究，有时包括算法性能的详细信息。读者不可能再找到类似于“Hello World!”这样的例子。再比作自然语言的例子，这类书就是在教导如何撰写可能超过20页的短篇小说。

问题是，从写一个段落到写一篇小说，这种跨越太难了。

一本针对中级编程能力的学生的书

市面上很少有针对中级编程能力学生的书籍。这些学生往往已经掌握了编程的基本知识，在看到`if`或者`while`时不会茫然，知道如何创建函数和调用函数，有能力编写几十上百行的短代码，却不知道如何处理上千行的程序。他们经常会犯错误，因为大多数入门级的书籍只教导如何编写正确的程序，却不会教导避免常见的错误。他们往往对大多数的概念和那些可以帮助提高编程能力的工具都不太熟悉，他们需要这样一个台阶：可以帮助他们从有能力编写短代码到有能力编写解决现实问题的程序。

现在入门和进阶的空档已经被数据结构和算法的书籍填充了一部分，这类图书一般提供实现数据结构或算法的完整例子。然而这并不是最合适的解决方法，这类图书致力于介绍数据结构和算法，却罕有提供帮助读者编写正确代码的信息。事实上，它们大多只提供程序，而很少解释。它们往往不解释编程概念，比如函数需要一个指针作为实参的原因或者深拷贝与浅拷贝之间的差异等。因此，读者只能自学这些编程技巧。

为了迎合这个需求，我写下这本针对中级编程能力的学生的书，本书适合作为学习编程的第二本教材。

避免出错和调试的重点

我们可以看到有很多关于如何编程的书籍，却很少关于开发软件的书籍。开发软件不是简单地输入代码，它需要更多的知识和技能。为了弥补这种不足，最好就是去研究什么是对的、什么是错的。只解释如何编写正确的程序是不够的，还需要解释常见的错误并将它们与正确的程序进行对比。

一次疏忽可能使程序运行出乎意料，甚至是某些情况下运行正确而另一些情况下出错。这种类型的错误往往很难发现，更别说更正了。本书将介绍一些常见的错误以教导读者如何避免这些错误。调试过程在大多数书中都不会涉及，罕有书籍会提到“调试器”这个词，以至于有些读者都不知道这类工具的存在。学会如何使用调试器一般不超过 30 分钟，这可以帮助程序员节省很多时间。关于如何使用调试器和调试策略的书籍则更少了。

程序设计和离散数学

程序设计和离散数学是计算机科学中的两个重要学科，然而，大多数书籍都将这两个主题分开，所以很少会在编程的书籍中看到数学公式，同样也很难在离散数学中看到代码。在本书中，这两个主题紧密结合，我相信读者可以从中学到更多的知识。

为什么本书使用 C 语言?

C 语言诞生于 20 世纪 60 年代后期和 20 世纪 70 年代早期。在 C 语言发明之后，很多语言也相继出现，这些语言也深受 C 语言的影响。除了它的历史影响之外，C 语言的简单易用也保证了它在几乎所有现代化平台中的重要地位。与许多操作系统一样，Linux 是就用 C 语言编写的，Android 基本都是用 Java 编写的但仍有叫作 JNI（Java Native Interface，Java 本地接口）的 C 语言接口。大多数计算机语言都可以与 C 语言进行通信或通过 C 语言进行通信，事实上这对一种编程语言而言是有用的，因为大多数操作系统接口都使用 C 语言。当一个全新的系统被设计出来，C 语言通常是第一种（很多情况下是唯一一种）被系统支持的编程语言。

对于具有中级编程能力的学生来说，C 语言是一个很好的选择，因为学习 C 语言需要了解很多计算机概念。langpop.com 网站对比了编程语言的受欢迎程度，得出 C 语言是最受欢迎的语言，紧接着是 Java。IEEE Spectrum[⊖] 中的一个报告将编程语言进行排行，主要考虑四类软件：移动应用、企业软件、嵌入式系统和网页。其中嵌入式系统中最受欢迎的就是 C 语言。四种类型都考虑时，前五名编程语言如下所示：

1. Java（100%）
2. C（99.3%）
3. C++（95.5%）
4. Python（93.4%）
5. C#（92.4%）

可以发现，基于 C 的编程语言（C、C++、C#）占据了前五席的三席。而 Java 是受 C++ 影响的。

⊖ Stephen Cass, Nick Diakopoulos, Joshua J.Romero, “Interactive: The Top Programming Languages IEEE Spectrums 2014 Ranking”, July 1, 2014, http:// spectrum.ieee.org/static/ interactive-the-top-programming-languages。

为什么需要读这本书?

如果你是计算机科学、计算机工程或者电子工程专业的学生，那么就绝对应该读本书。本书包含了很多基本概念，这些概念对于理解计算机中程序的运行方式十分重要。如果你是工程、科学、数学或者技术专业的学生，在学习工作中就很有可能需要用到计算机，而阅读本书将会有很大帮助。即便你不是上述专业的学生，仍然可以在本书中学到很多有用的概念(比如递归)。

作者、审稿人及封面设计师

作者简介

Yung-Hsiang Lu 是美国印第安纳州西拉法叶普渡大学电子与计算机工程学院的副教授。他是美国计算机协会（ACM）的杰出科学家和演说家。2011 年 8 ～ 12 月，曾是新加坡国立大学计算机科学系的客座副教授。他在美国加利福尼亚州斯坦福大学电子工程系取得博士学位。

审稿人简介

Aaron Michaux 是美国印第安纳州西拉法叶普渡大学电子与计算机工程学院的一名研究生。他在澳大利亚昆士兰科技大学取得计算机科学学士学位，在加拿大新布伦瑞克圣托马斯大学获得心理学学士学位。Aaron 在重新回到学校攻读博士学位之前已经作为专业程序员工作了 10 年。他的研究方向主要围绕计算机视觉和人类视觉感知。

Pranav Marla 是美国印第安纳州西拉法叶普渡大学电子与计算机工程学院的一名本科生。他主修的专业是计算机科学。在计算机工程、心理学和哲学上也稍有涉猎。他希望可以专攻机器学习和人工智能。

封面设计师简介

本书封面是由 Kyong Jo Yoon 描绘的。他是一名韩国画家，经常在自然场景中加入英雄人物。他是韩国美术协会的一名顾问，他的作品在美国伊利诺伊州芝加哥市的安内森美术馆中展出。

软件开发中的规则

如果由于软件错误，银行每天将你的钱减少 0.1%，你会满意吗？你能接受每个月都少走 40 分钟的手表吗？这两种情况都是“成功了 99.9%”，但却都让人无法接受。计算机现在用在很多应用程序中，有些甚至会影响人类的安全。即使你编写的程序在 99.9% 的时间里都可以正确地运行，那也有可能会在剩余的 0.1% 的时间里危害人类的生命。这是绝对让人无法接受的，这是一个失败的程序。因此，99.9% 的成功就是失败。

如果你住在加州的帕萨迪纳，现在想去纽约，你会走哪条路呢？也许你会去洛杉矶机场，然后坐飞机去纽约。但是纽约在帕萨迪纳的东边，而机场在帕萨迪纳的西边。你为什么不直接开车去东边呢？你为什么要绕路去机场呢？如果你直接开车去东边，而不是去机场排队的话，你就离纽约越来越近了啊。答案很简单：相对于汽车来讲，飞机这一交通工具更适

合长途旅行。在程序开发中，有许多用以管理大型软件的开发工具，你需要学习这些工具。没错，学习使用工具会消耗一些时间，但是如果你使用不合适的工具或者不使用工具的话，将浪费更多的时间。花些时间学习使用编程工具将在软件开发和调试时节省大量时间。

尽管经过了数十年的努力，现在的计算机依然没有达到智能化的水平。计算机无法猜测你的想法。如果你写的程序让计算机去做一件错事，那么计算机就会跟随指令去做。如果你的程序是错的，那就是你的责任。计算机无法猜测你的想法。在很多例子中，计算机程序中一个微小的错误就可以造成巨大的财产损失，甚至危及人类的生命。缺少一个分号“；”或者用“,”取代“.”，程序将无法执行。计算机程序不能容忍任何微小的错误。

通过测试方案并不能保证程序的正确性。测试只能得出一个程序有错误的结论但不能表明一个程序是正确的。为什么呢？测试方案能覆盖所有可能的情景吗？覆盖全部情景是非常困难的，而且在很多情况下是不可能的。因为测试方案很难检测特殊行为，所以一些问题可能会隐藏在你的程序中。

产生正确的输出并不意味着程序是正确的。你会认为一架已经安全起飞并着陆的飞机是安全的吗？如果飞机在漏油，你会在登机前要求航空公司维修飞机吗？如果航空公司回复：“之前没有人受伤，说明这架飞机是安全的。”你会接受航空公司的回应吗？如果司机闯红灯而没出事故的话，是否意味着闯红灯是安全的呢？一个可以产生正确输出的程序就像是一架安全着陆的飞机。可能有很多问题隐藏在安全的表面之下。很多工具可以检测人类隐藏的健康问题，比如 X- 射线，核磁共振以及超声波扫描。我们也需要工具来检测出隐藏在电脑程序中的问题。即使程序可以产生正确的输出，我们也要对它们进行修改。

你必须假设程序会出问题，并开发出检测和改正问题的策略。在写程序时，应该每次只专注于一小部分。在仔细检查并确保没有问题之后再进行下一部分。对于大多数程序，你需要为了测试这些小的模块而编写额外的程序。虽然这些测试代码并不包含在最终的程序中，但是编写测试程序可以节省大量时间。有时，测试代码可能会比程序本身还要多。我自己的经验建议 1:3 的比例：最终程序中的 1 行代码，需要 3 行测试代码。

没有什么工具可以取代一个清醒的大脑。工具可以提供一些帮助，但是对概念清晰深刻的理解才是最重要的。如果你想成为一名优秀的程序员，那么你需要完全理解每一个细节。不要指望工具可以替你思考：它们做不到。

源代码

本书中所有的程序都可以从 github.com 上获取。请使用下面的命令获取文件：

```
.....
$ git clone ‘ https://github.com/yunghsianglu/IntermediateCProgramming.git ’
$ 是 Linux 终端的 shell 提示符
.....
```

目　录

Intermediate C Programming

第一部分
Intermediate C Programming

计算机存储：内存和文件

第1章

Intermediate C Programming

程序的执行

1.1 编译

本章讲述怎样在 Linux 下编写、编译和执行程序。我们使用 Linux 终端并解释你需要键入的命令。为什么要学习怎样使用终端呢？首先，此终端对于使用计算机工作而言是一个灵活且方便的界面。意识到这一点可能需要一些经验的积累，但是学习怎样使用此终端将会提高你的生产力。其次，许多云计算或者 Web 服务提供终端访问。这是提供计算资源的一个很自然的方法，尤其是当涉及在多台计算机上工作的时候（像在数据中心一样）。当仅在一台计算机上工作的时候，图形用户界面（GUI）十分优秀。然而当面对多台计算机的时候，GUI 就可能造成困扰。同时，使用终端可以帮助你理解 UNIX 系统是如何工作的。在熟悉了终端指令之后，你将会理解集成开发环境（IDE）和它可以为你做些什么。Eclipse IDE 将会在本书后续的章节中介绍。

在 Linux 下启动终端并输入：

```
$ cd
$ pwd
$ mkdir cprogram
$ cd cprogram
```

本书中，`$` 被用作终端提示符。

第一条命令 `cd` 的意思是“ change directory”。如果像第一条命令中那样在 `cd` 后面没有添加参数，那么它将返回至你的主目录（也叫“文件夹”）。

第二条命令 `pwd` 的意思是“ print the current working directory”。它类似于 `/home/yourname/`。

第三条命令 `mkdir` 的意思是“make a directory”。命令 `mkdir cprogram` 的意思是“make a directory whose name is cprogram”。

不能创建名字里包含空格的目录或文件。原因很简单：关于目录和文件名的国际标准（称为国际标准化组织或 ISO 9660）不允许出现空格。如果一个目录或者文件名中含有空格，那么一些程序将无法运行。

最后一条命令 `cd cprogram` 的意思是“ change directory to（i.e.,enter）cprogram”。这是刚刚创建的目录。

1
~
3

在终端中输入：

```
$ which emacs
```

如果在终端上没有显示或者出现“Command not found”，那么请先安装 Emacs。如果你不知道怎样在 Linux 下安装软件，请阅读附录 A.5 部分。

在终端中输入：

```
$ emacs prog1.c &
```

这条命令会启动 Emacs 去编辑一个叫作 prog1.c 的文件。添加 & 允许你同时使用终端和 Emacs 编辑器。如果结尾处没有 &，终端将强制等待直到 Emacs 退出。在 Emacs 中输入如下代码：

```
1  // prog1.c
2  #include <stdio.h>
3  #include <stdlib.h>
4  int main(int argc, char * * argv)
5  {
6     int a = 5;
7     int b = 17;
8     printf("main:␣a␣=␣%d,␣b␣=␣%d,␣argc␣=␣%d\n", a, b, argc);
9     return EXIT_SUCCESS;
10 }
```

保存文件。你可能猜到了此程序将显示：

```
main: a = 5, b = 17, argc =
```

这是本书中出现的第一个完整的程序，需要做一些说明。此程序通过调用 printf 来打印信息。printf 是 C 语言提供的一个函数，但是你需要在使用此函数之前把 stdio.h 包含进来，stdio.h 是标准化输入输出函数的一个头文件。在一个 C 程序中，起始点是 main 函数。这个程序在它成功打印地址后将返回 EXIT_SUCCESS。你能够猜到，如果一个程序能够返回 EXIT_SUCCESS，那么另一个程序便可以返回 EXIT_FAILURE。为什么程序要返回 EXIT_SUCCESS 或是 EXIT_FAILURE 呢？在今天这样复杂的电脑系统中，许多程序是被其他电脑程序调用的。因此，让你的程序告诉调用程序其是否完成了设定去做的任务是十分重要的。此信息允许调用程序去决定下一步采取什么行动。EXIT_SUCCESS 和 EXIT_FAILURE 是在 stdlib.h 中定义的，所以在第 2 行中将它包含进来。

本书中，源代码用从 1 开始的行编号标示出来。有时，代码会涉及前面提到的例子，行编号和前例中的值一致。

main 函数是一个 C 程序的起始点，但是这并不总是适用于 C++ 程序。如果一个 C++ 程序有一个静态对象，那么此对象的构造函数将会在 main 函数之前被调用。因为本书是关于 C 编程的，那么假定 main 函数是所有程序的起始点是保险的。

argc 是什么？通过运行程序更容易来回答这个问题。首先，我们需要解释如何把这个程序从人类可读的格式转换为计算机可读的格式。

输入进 Emacs 的是一个“C 源文件”。它大致和英语类似，并由拉丁字母构成。然而，因为计算机不理解这种格式，“源文件”需要转换为一种叫作可执行文件的计算机可读格式。编译器是这种转换的必需工具，gcc 则是 Linux 中一种很受欢迎的编译器。在终端中，输入：

```
$ gcc prog1.c -o prog
```

4

这条指令有如下含义：

- 执行安装在 Linux 下的 gcc 指令。
- 使用 prog1.c 作为 gcc 指令的输入。
- 把输出文件命名为 prog（-o 指定输出文件的名字）。此输出文件是一个可执行文件，意思是计算机可以运行它。

别这样输入：

```
$ gcc prog1.c -o prog1.c
```