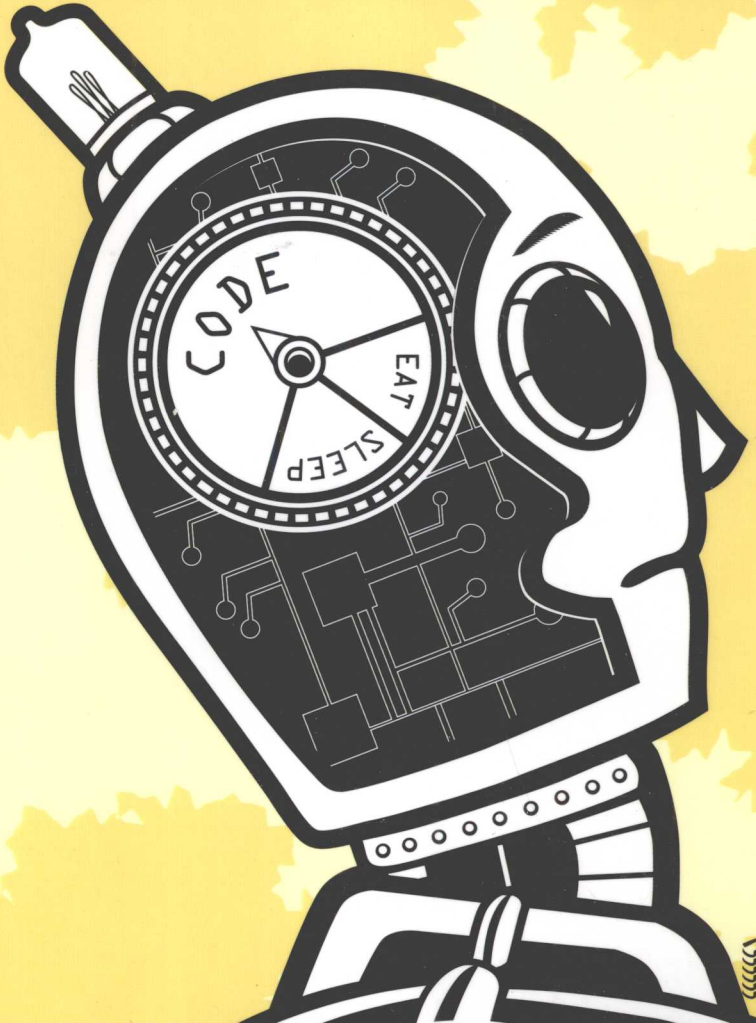
学会像程序员一样思考，构建创新性问题解决方案！

no starch press

像程序员一样思考

THINK LIKE A PROGRAMMER

[美] V. Anton Spraul 著　　徐波 译



人民邮电出版社
POSTS & TELECOM PRESS

像程序员一样思考

THINK LIKE A PROGRAMMER

[美] V. Anton Spraul 著　　徐波 译

人民邮电出版社
北京

图书在版编目（CIP）数据

像程序员一样思考 /（美）斯保尔（Spraul, V. A.）著；徐波译. -- 北京：人民邮电出版社，2013.6
ISBN 978-7-115-31658-5

Ⅰ. ①像… Ⅱ. ①斯… ②徐… Ⅲ. ①程序设计－基本知识 Ⅳ. ①TP311.1

中国版本图书馆CIP数据核字(2013)第074759号

版权声明

◆ 著 [美] V.Anton Spraul
译 徐 波
责任编辑 陈冀康
责任印制 程彦红 杨林杰

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：16
字数：322 千字 2013 年 6 月第 1 版
印数：1－3 500 册 2013 年 6 月北京第 1 次印刷
著作权合同登记号 图字：01-2012-7214 号

定价：49.00 元

读者服务热线：(010)67132692 印装质量热线：(010)67129223
反盗版热线：(010)67171154

内 容 提 要

编程的真正挑战不是学习一种语言的语法，而是学习创造性地解决问题，从而构建美妙的应用。本书分析了程序员解决问题的方法，并且教授你其他图书所忽略的一种能力，即如何像程序员一样思考。

全书分为 8 章。第 1 章通对几个经典的算法问题切入，概括了问题解决的基本技巧和步骤。第 2 章通过实际编写 C++代码来解决几个简单的问题，从而让读者进一步体会到问题解决的思路和应用。第 3 到 7 章是本书的主体部分，分别探讨了用数组、指针和动态内存、类、递归和代码复用来解决问题的途径和实际应用。最后，第 8 章从培养程序员思维的角度，进行了总结和概括，告诉读者如何才能像程序员一样思考。

本书选取的话题切中程序员的痛点，针对他们最容易陷入挣扎的领域展开讨论，引发思考。每章后面都给出一些编程习题，使得读者能够应用该章所讨论的概念，训练和提升问题解决的能力。

本书适合初级到中级的程序员用来提升自己的问题解决能力和应用编程技能的能力，也适合计算机相关专业的学生作为参考书阅读。

前 言

你是否在编写程序时感受到挣扎，即使觉得已经理解了自己所使用的编程语言？你是否阅读了一本编程书籍的某一章并能够顺利理解，却无法把自己所读到的东西应用于程序中？你是否能够理解自己在线所阅读的一个程序，甚至能够把每行代码所完成的任务告诉其他人，但是自己在接到一个编程任务后，却面对文本编辑器的空白屏幕大脑一片空白？

并不是只有你才这样。我从事编程教学已经超过 15 年，几乎所有的学生都在某种程度上符合上面的描述。我们称之为缺乏问题解决能力。所谓问题解决能力，就是根据问题描述编写一个原创程序来解决这个问题。并不是所有的编程任务都需要深入的问题解决能力。如果只是对一个现有的程序进行微小的修改、调试或添加测试代码，这类编程在本质上是机械的，不会对我们的创造力提出多大的挑战。但是，所有的程序总会在某个时刻需要解决问题，所有优秀的程序员都需要具备解决问题的能力。

解决问题是非常困难的。但有些人确实使它变得非常轻松，因为他们天赋异禀，就像迈克尔·乔丹这样的体育天才一样。对于这帮天才，高级的思路几乎可以毫不费力地转换为源代码。用 Java 的一个比喻来说，他们的脑子天生就像能执行 Java 代码一样，而绝大多数人只能通过虚拟机来解释代码。

没有超常天赋对于成为程序员而言并不是致命的。如果真是如此，世界上也就没有多少程序员了。但是，我看到过许多值得尊重的学习者却在挫折面前挣扎太久。最坏的情况

是，他们完全放弃了编程，认为自己永远成不了程序员，觉得只有天赋异禀的人才能成为优秀的程序员。

学习解决编程问题为什么这么困难呢？

部分原因是解决问题是一种与学习编程语言不同的活动，它使用了一组不同能力的“肌肉”。学习编程的语法、阅读程序、记忆应用程序编程接口的元素，这些都是分析性的“左脑”活动。使用以前所学会的工具和技巧编写一个原创程序却是创造性的“右脑”活动。

假设我们想拿走掉落在自家房顶上的其中一条雨槽内的一根树枝，但是自家的梯子却不够长，够不着那根树枝。我们会跑到自己家的车库，寻找某样东西来帮助我们拿走雨槽上的树枝。有没有办法使梯子加长呢？我们站在梯子上时能不能手持某样东西来抓住或拔掉树枝？或许我们可以从某个位置爬上屋顶拿走树枝。解决问题是一种创造性的活动。不管你是否相信，在设计自己的原创程序时，我们的智力活动过程与想方设法从雨槽中拿走树枝的过程非常相似，但与调试一个现有的 for 循环的过程却截然不同。

但是，绝大多数编程书籍却把重点放在语法和语义上。学习编程语言的语法和语义是极为重要的，但这只是学会用这种语言进行编程的第一个步骤。在本质上，大多数面向新手的编程书籍所讲述的是怎样阅读程序而不是怎样编写程序。把重点放在怎样编写代码上的书籍常常是有效的“烹调书”，因为它们讲述在特定情况下所使用的特定“菜谱”。这种书籍对于节省时间而言是极有价值的，但它们并不是通往学会编写原创代码的正确道路。考虑原始意义上的烹调书。尽管优秀的厨师拥有自己的烹调书，但没人依靠烹调书就能成为优秀的厨师。优秀的厨师理解配料、备菜方法和烹饪方法，并知道怎样把它们组合在一起烹制美味佳肴。优秀的厨师在烹制美味时所需要的只是工具完备的厨房。同理，优秀的程序员理解语言的语法、应用程序框架、算法和软件工程原则，并知道怎样把它们结合起来创建优秀的程序。为优秀的程序员准备一个需求规范列表，给他一个功能齐全的开发环境，他就能创造优秀的程序。

一般而言，当前的编程教育并没有在解决问题这个领域提供太多的指导。相反，如果程序员可以使用所有的编程工具并按要求编写足够的程序，最终他们将学会编写这些程序并且完成得很好。这种说法并没有错，但是“最终”可能意味着相当长的时间。从蹒跚学步到初窥门径的道路上可能充满挫折，有太多的人信心满满地开始了旅程，却倒在了半途中。

我们并不一定要饱经磨难才能获得成功，而是可以用一种系统的方式来学习怎样解决

问题，这正是本书的核心所在。我们可以学习对自己的想法进行组织的技巧、解决方案的发现过程以及对某些类型的问题所实行的策略。通过研究这些方法，我们可以释放自己的创造力。不要弄错了：编程，尤其是解决问题，是一种创造性的活动。创造性是神秘的，没人能够准确地说出创造性是怎样发挥作用的。但是，如果我们能够学会怎样作曲、能够领会别人所提出的有关写作创造能力的建议或者能够学会绘画，我们也可以学会怎样有创造力地解决编程问题。本书并不是讲述具体的做法，而是帮助读者开发属于自己的问题解决天赋，明白自己应该怎么做。本书的目的是帮助读者成为自己心目中的程序员。

我的目标是让本书的读者学会怎样用系统性的方法完成每个编程任务，并坚信自己最终能够解决一个特定的问题。当读者完成本书的学习时，我希望你能够像程序员一样思考，并坚信自己已经是个程序员。

本书内容简介

在解释了本书的必要性之后，我还需要对本书将讨论的内容和不会讨论的内容进行一些说明。

预备条件

本书假设读者已经熟悉了 C++的基本语法和语义，并且已经编写过一些程序。本书的大部分章节将预计读者已经了解了特定的 C++基础知识。这些章节将从对这些基础知识的回顾开始讨论。如果读者还在学习语言的基础知识阶段，也不必担心。关于 C++的语法有大量优秀的书籍可供参考，在学习解决问题的同时学习语法也是没有问题的。在尝试解决每章后面的习题之前，要保证自己已经学会了相关的语法。

所选择的话题

本书所覆盖的话题代表了我所看到的程序员新手最容易陷入挣扎的领域。它们还代表了初级和中级编程中许多跨领域的话题。

但是，我应该强调，这并不是一本用于解决特定问题的算法或模式的“烹调书”。尽管后面的章节讨论了怎样使用广为人知的算法或模式，但这本书并不适合作为解决特定问题的参考书，所以读者不应该只把注意力集中在直接与自己当前所面临的问题相关的章节中。

反之，读者应该从头研读全书，暂时只跳过那些由于缺乏预备知识而无法直接学习的内容。

编程风格

这里简单概述一下本书所使用的编程风格：本书并不追求高性能的编程，并不苛求产生最紧凑、最高效的代码。我为源代码例子所选择的风格是把容易理解作为首要的考虑因素。在有些情况下，我会采用多个步骤完成那些实际上只用一个步骤就可以完成的任务，其原则就是为了进行清晰的说明。

本书也会讨论编程风格的相关内容，但只是比较宽泛的问题，例如类里面应该包含什么或者不包含什么，而不会讨论细节的问题，例如代码应该怎样缩进。作为学习中的程序员，读者当然想要在自己的所有工作中采用一致的、容易阅读的风格。

习题

本书包含了一些编程习题，但本书并不是教科书，因此读者不会在书末找到这些习题的答案。这些习题为读者提供机会应用当前章节所讨论的概念。尝试完成哪些习题当然是由读者自行决定的，但是把书中所讨论的概念应用于实践是至关重要的。简单地完成全书的阅读并不能让自己的水平得到长进。记住，本书并不是告诉读者在每个解决方案中应该怎样做。在应用本书所展示的技巧时，读者能够通过拓展自己的能力来发现需要完成什么任务。而且，成功地完成这些习题也能促进本书的另一个基本目标，就是增加读者的自信心。事实上，有一种很好的方法能够知道读者是否充分地完成了一个特定问题领域的习题，那就是当读者对解决这个领域的其他问题充满信心之时。最后，编程习题应该是充满乐趣的。虽然有时候读者可能觉得辛苦而想做些其他事情，但是完成编程习题应该是一项充满回报的挑战。

读者应该把本书看成是大脑的一项障碍训练。障碍训练可以增强力量、毅力和敏捷性，为训练者培养信心。通过阅读全书并把书中所讨论的概念应用于尽可能多的习题中，可以增强信心并培养能够用于任何编程情况的问题解决技巧。在未来，当读者面临一个难题时，就会知道应该怎样克服它。

为什么用 C++

本书的编程例子是用 C++编写的。由于本书的主题是怎样解决编程问题而不是专门讨论 C++的，因此书中不会出现关于 C++特定的提示或技巧，并且本书所讨论的基本概念也适用于任何编程语言。然而，我们无法在不讨论具体程序的情况下讨论编程，因此必须选择一种特定的语言。

选择C++的原因有几方面。首先，它在许多问题领域都是非常流行的一种语言。其次，它的来源是严格的过程性语言 C，因此 C++的代码既可以采用过程性的惯用法，也可以采用面向对象的惯用法。面向对象编程现在已经极为常见，在讨论解决问题时不可能忽略它，但是许多基本的问题解决方案可以用严格的过程性编程术语来讨论，这样不仅能简化代码，而且能简化具体的讨论。第三，作为一种具有高级程序库的低层语言，C++允许我们讨论不同层次的编程。最优秀的程序员可以在需要的时候“手工编写”解决方案，并利用高级程序库和应用程序编程接口来减少开发时间。选择C++的最后一个原因是一旦学会了怎样用C++解决问题，就很容易学会用其他任何语言来解决问题。许多程序员发现在一种语言中所学会的技巧可以很轻松地应用于另一种语言，尤其是当前者是 C++语言的时候，这是由于它的惯用法跨度广泛，或者坦率地说，是由于它的难度所决定的。C++是真正的实用型语言，它并不是面向教学而设计的。虽然初看上去有些吓人，但一旦取得进展之后，就会觉得自己不再只是会一点点编程的人了，而是将成长为一名真正的程序员。

作 者 简 介

V. Anton Spraul 讲授入门级编程和计算机科学已经超过 15 年。本书凝聚了他在多年的开发经历中所提炼的经验和技巧，并在面向许多遭遇瓶颈的程序员的一对一指导中收到了良好的效果。他还是《Computer Science Made Simple》（Broadway）的作者。

译 者 简 介

徐波，浙江宁波人，熟悉 C 和 C++、Java 等语言。2002 年开始从事计算机技术图书翻译。徐波技术视野广阔，翻译文笔优美。翻译有《C 专家编程》、《C 和指针》等。

致　　谢

没有一本书只通过一位作者就可以完成，在本书的写作中，我得到了很多人的帮助。

我非常感谢 No Starch 出版社的每位工作人员，尤其是 Keith Fancher 和 Alison Law，他们在本书的出版过程中担任了编辑、策划和指导工作。我还必须感谢 Bill Pollock，正是他与我签订了本书的合同，我希望他能够像我一样对本书的质量感到满意。No Starch 出版社的伙计们与我交流时表现出了从不消减的热心和诚恳。我希望有朝一日能够在私下里与他们会面，看看他们把公司网站上的卡通阿凡达装配成什么样了。

Dan Randall 作为技术编辑，他的工作非常出色。他还提出了大量技术之外的意见，帮助我在许多地方完善了手稿的质量。

在我的家庭里，生命中最重要的人 Mary Beth 和 Madeline 让我感受到爱、支持和热情，特别是为我提供了充足的写作时间。

最后，我还要感谢多年以来曾经听我讲授编程的学生们：感谢让我成为你们的老师。本书所描述的技巧和策略是在我们共同努力的过程中发展成型的。我希望我们能够让下一代程序员的学习旅程变得轻松一些。

对本书的赞誉

“作者在向初学者阐述困难概念方面显然具有广博的知识和丰富的经验。本书显示了他脚踏实地、一丝不苟却又令人愉悦的写作风格。”

—Adrian Woodhead, Slashdot

“本书所提供的习题类似于我在接受 Google 和 Facebook 的软件工程面试时所遇到的问题，因此对于打算通过面试寻找新工作的专业程序员，本书是极好的复习材料。”

—Ariane Coffin, Wired.com's GeekMom

“这是我所阅读过的收获最大的书籍之一，因为它指导我们设计一个属于自己的系统，而不是把思维固化为只能采取一种正确的方法才能达到目的。”

—Lucas Westermann, Full Circle Magazine

“如果读者能够认真研读本书，我保证它可以极大地拓展您的思维。”

—David Bolton, About.com C/C++/C#

“不管使用什么教材向新学生讲授编程和程序逻辑，我建议一定要把本书作为重要的参考书。”

—Joe Saur, The ACM's Software Engineering Notes Magazine

“V. Anton Spraul 所提供的建议简单、直观并且实用。本书的阅读是一个既轻松又极有价值的过程。”

—James Powell, Enterprise Systems

“对于所有想要培养创造性的解决问题能力的人以及已经学习了编程但觉得没有完全理解概念的人，我向他们强烈推荐本书。”

—Robert Perkins, Game Vortex

“如果我教其他人学习编程，这肯定是我将要选择的教材。”

—Stephen Chapman, Ask Felgall

目　录