Mc Graw Hill Education

学习指导与习题解答

Schaum's Outline of Principles of Computer Science

计算机科学导论
学习指导与习题解答

Carl Reynolds
Paul Tymann 著

陈宗斌 等 译

清华大学出版社

Mc Graw Hill Education

学习指导与习题解答

Schaum's Outline of Principles of Computer Science

计算机科学导论学习指导与习题解答

Carl Reynolds
Paul Tymann 著

陈宗斌 等 译

清华大学出版社
北京

Carl Reynolds, Paul Tymann
Schaum's Outline of Principles of Computer Science
EISBN: 978-0-07-146051-4

图书在版编目(CIP)数据

计算机科学导论学习指导与习题解答 / (美)雷诺兹(Reynolds, C.),(美)泰曼(Tymann, P.)著;陈宗斌等译. --北京: 清华大学出版社, 2010.5
书名原文: Schaum's Outline of Principles of Computer Science
ISBN 978-7-302-22267-5

Ⅰ. ①计…　Ⅱ. ①雷… ②泰… ③陈…　Ⅲ. ①电子计算机—概论　Ⅳ. ①TP3

中国版本图书馆 CIP 数据核字(2010)第 047684 号

责任编辑: 龙啟铭
责任校对: 徐俊伟
责任印制: 李红英

出版发行: 清华大学出版社　　**地　　址**: 北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**: 100084
社　总　机: 010-62770175　　**邮　购**: 010-62786544
投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn
质　量　反　馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn
印 装 者: 北京市清华园胶印厂
经　　销: 全国新华书店
开　　本: 185×230　　**印　张**: 18　　**字　数**: 381 千字
版　　次: 2010 年 5 月第 1 版　　**印　次**: 2010 年 5 月第 1 次印刷
印　　数: 1～3000
定　　价: 39.50 元

产品编号: 034570-01

译者序

提到计算机科学，很多人可能会把它等同于编写程序，事实上计算机科学远不止这么简单。计算机科学包括许多分支学科，要想成为计算机科学方面的专家，就必须理解这个领域的所有分支学科。计算机科学的一些重要的学科包括算法、程序设计、程序设计语言、计算机硬件、网络、操作系统、数据库系统、分布式计算，以及关于使用计算机技术的道德问题。

在讲授计算机科学方面的知识时，主要采用两种方法：深度优先方法和广度优先方法，前者是指深入研究一个特定的主题，后者则涵盖计算机科学的许多分支学科，但是其深度只要求基本理解每个学科的原理，本书采用了后一种方法。在引导学生从总体上认识了这个领域之后，学生就可以深入学习某些分支学科。

本书首先讨论了计算机科学的概念以及计算的发展历史，然后逐步介绍了各个分支学科，为学生今后深入学习其他计算机专业课程打下坚实的基础。在每一章末尾还提供了相关的复习题，有利于学生巩固所学的知识；并且在本书附录中给出了所有复习题的答案，为学生学习提供了方便。

本书内容简明扼要、结构组织合理，非常适合用作高等院校计算机专业的基础课教材，也可供有兴趣了解计算机科学的相关读者参考。

参加本书翻译的人员有陈宗斌、王馨、陈红霞、张景友、易小丽、陈婷、管学岗、王新彦、金惠敏、张海峰、徐晔、戴锋、张德福、李振国、高德杰、宫飞、刘淑妮、赵中元、史宁、赵敏、俞群。

由于时间紧迫，加之译者水平有限，错误在所难免，恳请广大读者批评指正。

出版说明

计算机专业知识学习的最佳方式，就是阅读良好设计且具有完整解释的示例，并自己动手进行实践练习。本系列图书就是遵循这种写作风格，每本书首先提纲携领地说明其重要知识点，然后通过大量丰富的示例和习题，并给出详尽的答案，让读者全面透彻地理解概念，掌握每个知识点，使读者能举一反三，灵活运用理论知识解决实际问题，并检验知识的掌握程度，因此，本系列书不仅很适合自学，也非常适合用作高等院校计算机专业核心课程的参考教材或辅助教材。

本系列图书包括：

- 计算机科学导论学习指导与习题解答
- 离散数学学习指导与习题解答
- 数据结构学习指导与习题解答(C++ 语言版)
- 数据结构学习指导与习题解答(Java 语言版)
- C++ 程序设计学习指导与习题解答
- Java 程序设计学习指导与习题解答
- 操作系统学习指导与习题解答
- 计算机体系结构学习指导与习题解答
- 软件工程学习指导与习题解答
- 计算机网络学习指导与习题解答
- 关系数据库基础学习指导与习题解答
- 计算机图形学学习指导与习题解答

本系列图书由“计算机专业课程研究组”策划、翻译和审校，读者如对本书有任何建议和意见，请来信：tuppub. cs@gmail. com。

作者简介

Carl Reynolds 在罗彻斯特理工学院(RIT)的计算机科学系讲授数据库、操作系统、程序设计以及程序设计语言理论方面的课程。他在大学任教 10 年,其中有 4 年涉及计算机行业。在来到 RIT 之前,Reynolds 在计算机行业花了 19 年的时间致力于提高硬件和软件供应商的技术和培训能力,其中有 6 年的时间是与道琼斯工业制造商一起创建用于机器控制的专家系统。他感兴趣的主题包括遗传算法、专家系统和图像处理。

Paul Tymann 是罗彻斯特理工学院的计算机科学系的教授和系主任。他讲授基本和高级程序设计技术有超过 15 年的经验。最近,他参与了 RIT 的一种新型的生物信息程序的开发。在进入学术界之前,Tymann 教授在为销售点终端开发控制软件的行业中工作。在过去 5 年,他一直在生物信息领域中工作,并且完成了罗彻斯特与罗格斯大学的联合软件开发项目。

目录

第 1 章　计算机科学简介

1.1　什么是计算机科学

不同的作者以不同的方式定义计算机科学(Computer Science)。Wikipedia(http://en.wikipedia.org/wiki/Computer_science)把计算机科学定义为与计算相关的多种学科的集合，包括理论和实际两方面：涉及信息和计算的理论基础、语言理论、算法分析和开发、计算系统的实现、计算机图形学、数据库、数据通信等。

美国的网络和信息技术研究与发展(NITRD)的国家协调办公室以一种类似的广义方式定义计算机科学为：

计算系统和计算的语义研究。源于该学科的知识体系包含用于理解计算系统和方法的理论，设计方法学、算法和工具，用于概念测试的方法，分析和验证方法以及知识表示和实现(http://www.nitrd.gov/pubs/bluebooks/1995/section.5.html)。

另一种广义定义来自于美国计算机学会(ACM)的模型课程。该定义指出计算机科学是“计算机和算法过程的研究，包括它们的原理、它们的硬件和软件设计、它们的应用以及它们对社会的影响”。

Gibbs 和 Tucker 给出的计算机科学的著名定义(Gibbs 和 Tucker，“A Model Curriculum for a Liberal Arts Degree in Computer Science，”*Comm. of the ACM*，vol. 29，no. 3，March 1986)强调了算法开发和分析，并把它作为计算机科学的核心。

人们也经常问到另一个问题：“计算机科学怎样成为一种科学?”与物理学、生物学和化学相比，计算机科学并不是基于自然世界的研究。从这种意义上讲，计算机科学更像是数学，而不是科学。一些人坚持认为计算机科学实际上是计算机艺术(这里的“艺术”意指实践)。另一方面，计算机科学家确实使用科学的方法来提出和测试假设，并且计算机科学中的一些非常不明显的发现具有重要的现实意义。例如，人们发现一些重要的问题不能简单地通过计算加以解决，我们将在后面讨论它。

尽管关于计算机科学的定义有许多种，但它们实质上都强调了算法的研究。算法在某种意义上是计算机科学的中心。计算机科学把算法设计与分析的理论概念同如何在计算机上实现算法并解决实际问题的实际考虑有机地结合在一起。

算法定义了用于解决特定问题或者执行某项任务的详细、无歧义的动作序列。如果你曾经遵照菜谱烹饪、遵守一组驾驶指令，或者填写所得税表格，那就是在与算法打交道。

例如，你可能在某个时间学习如何确定两个数的**最大公约数**(greatest common divisor，GCD)。万一你忘记了，提醒一下两个正整数的 GCD 是两个数的每个公约数中的那个最大的整数。例如，42 和 30 的 GCD 是 6。下面给出的算法可用于计算两个正整数 a 和 b 的 GCD：

如果 b 是 0，那么 a 和 b 的 GCD 就是 a。算法结束。

设 r 是 a 整除 b 所得到的余数。

使用 b 和 r 重复这个过程。

考虑使用 42 和 30 的 GCD。设 a=42，b=30。从算法的第 1 步开始这个过程。由于 b 不为 0，就继续执行第 2 步。在第 2 步中，计算 42 除以 30 时得到的余数，它是 12。第 3 步指示我们重复这个过程，这一次使用 30 和 12。因此在第二次通过这个过程时，a 现在是 30，b 现在是 12。由于 b 不为 0，计算 30 和 12 的余数，它是 6，然后使用 12 和 6 重复这个过程。与以前一样，由于 b 不为 0，计算 12 和 6 的余数，并得到 0。现在将使用 6 和 0 重复这个过程。不过，这一次，由于 b 现在为 0，就可以断定 42 和 30 的 GCD 是 6。

算法是计算机处理信息所必需的，因为计算机程序实质上是算法的电子形式，它告诉计算机要执行哪些特定的步骤来完成指定的任务。为了研究算法的电子形式，计算机科学家还必须理解将用于执行这些算法步骤的计算机。术语“硬件”用于描述计算机的物理、实际的部件。键盘、鼠标、主板、图形卡和处理器全都是计算机硬件的例子。

就像赛车司机需要理解他们驾驶的车辆的能力和限制一样，计算机科学家也必须理解将要实现计算算法的硬件平台。对赛车司机来说，仅仅知道“如何驾驶”是不够的；而对计算机科学家来说，仅仅“知道算法”也是不够的。在特定硬件平台上表现最佳的算法在另一种硬件平台上可能并不是最佳的。

通常以人类容易理解的形式表达算法。例如，前面给出的用于计算两个数的 GCD 的算法是用英语编写的，使得你很容易理解它。

即使你可能理解多种语言，计算机则只能理解**机器语言**(machine language)。机器语言是计算机旨在解释的代码系统。机器语言中的每个单词代表计算机可以执行的一个简单的动作。例如，机器语言指令“add”指示计算机把两个数相加起来(在第 3 章“计算机组织结构”中，我们将更详细地解释机器语言)。指令集在由计算机执行时，将会执行称为程序的算法步骤。

人类很难直接与机器语言打交道。机器指令字由几行 1 和 0 组成，长度通常是 8、16、32 或 64 位，有时长度有会变化。由于人们很难直接操纵这些奇怪的代码，就开发了计算机语言，使得很容易把算法转变成计算机可以理解的形式。我们把这些语言称为**高级语言**(higher-level language)，因为与计算机的 1 和 0 的级别相比，这些语言旨在允许人类在“更高级别”上工作。另一方面，机器语言通常称为**低级语言**(low-level language)。Java、FORTRAN、Basic 和 ADA 只是高级语言的少数几个例子，它们被计算机科学家用来表达他们开发的算法。使用低级语言或高级语言表达算法的行动称为程序设计。

从 20 世纪 50 年代开始，多年来，计算机科学家已经创建了许多种高级语言。在早期，一些专家认为应该有可能发现一种将最适合于所有用户的语言。不过，从那时起，计算机科学家就发现语言设计总是要在一些特性和能力与另外一些特性和能力之间进行折中。因此，今天我们就具有许多种优秀的高级语言，其中一些特别适合于符号操纵，一些特别适合于教学编程，一些适合于矩阵代数应用，一些适合于一次性的程序，一些适合于关键任务的与生命相关的应用，一些适合于实时自动控制中的应用，另外还有许多高级语言是通用的。计算机科学家研究了计算机语言和形式语法的一般特征，并且通常精通几种或许多种不同的语言。

术语**软件**(software)用于描述计算机用于执行算法的指令集或程序。软件包含指导硬件操作的指令。使人们可以访问计算机的基本功能的软件称为**系统软件**(system software)。系统软件负责控制和管理计算机系统的硬件，以及用于使程序开发人员和普通用户可以轻松地使用计算机。系统软件的例子包括操作系统、显示管理器、病毒扫描程序、语言处理器(称为编译器或解释器——将在关于软件的章节中讨论)和设备驱动程序。

像字处理器或电子数据表这样的程序称为**应用软件**(application software)。应用软件用于完成特定的任务。应用软件可以是一个程序，或者是一些程序的小集合，它们一起工作，为计算机的用户完成某项任务。

操作系统是特别重要和复杂的系统软件。它们之所以很重要，是因为操作系统的性能对计算机用户的体验和计算机系统的效率在整体上具有显著的影响。20 世纪 60 年代和 70 年代，在更简单的计算系统的时代，公司可能购买没有操作系统的计算机，其本意是编写或使用它自己的操作系统，但是，今天人们在购买计算机时总会购买一种操作系统。

操作系统允许轻松地访问：外围设备(如打印机和显示器)、用于存储信息(如数据、文档和程序)的文件系统、使得很容易启动应用程序的用户界面、时钟、使用标准网络协议的通往 Internet 的连接、应用程序可用于请求操作系统的服务的一组“调用”或“方法”、用于给多个同时在运行的程序分配内存的高效算法，以及用于在多个人和/或程序之间同时共享计算机访问的高效算法。

今天，流行的操作系统包括 Microsoft Windows、Mac OS、Unix、Linux(Unix 的一个变体)和 IBM 的 MVS，等等。事实上，操作系统开发的领域在计算机科学中仍然是一个非常活跃的领域。操作系统不仅变得更复杂(例如，添加防火墙及其他保护)，而且变得更多样化。当开始用计算机控制像恒温器和洗碗机这样更简单的设备时，计算机科学家为这些需求创建了专用的“嵌入式系统”的操作系统。

甚至在进入 20 世纪 80 年代时，许多(如果不是大多数)计算机都是独立的——没有相互连接。在 20 世纪 70 年代和 80 年代期间，计算机科学家探索了计算网络的优点，并提议了计算机之间的许多种不同的物理连接，以及不同的网络协议。当时，不同的计算机供应商之间都利用不同的标准进行着如火如荼的竞争，并且每个供应商都希望通过销售其特定的

网络产品来“锁住”客户。IBM 提供了 SNA(System Networking Architecture),Digital Equipment 推销的是 DECnet,Hewlett Packard 提供了 DS(Distributed Systems),而 Xerox 则提供了 XNS(Xerox Networking Systems)。甚至通用汽车公司也利用其 MAP (Manufacturing Automation Protocol)参与到这场竞争中来。其中所有的产品都不与其他任何产品直接兼容,但是都提供了通往其他系统的“桥梁”。

今天,计算机科学家在网络方面面临不同的问题。全世界的人基本上都认同将 IEEE 801标准和 TCP/IP 协议用于 Internet。现在的问题是:必须在不干扰旧有的“安装基础”运营的情况下扩展 Internet 地址的数量,适应新的、更快的物理连接(如光纤),提高无线连接的速度(它们本质上更慢并且更容易受到干扰),管理更大的数据传输(如电影,它也需要严格的实时性能,使得电影不会在中间停止播放),以及为成百上千的数字传感器的即兴连接提供低功耗、低成本的协议。

今天,支持几乎所有应用的是数据库技术。最具优势的数据库模型是关系数据库,它最初是在 20 世纪 80 年代投入商业应用的。计算机科学家开发了一些算法,用于从数据量非常之多的存储库中快速地存储和检索信息。例如,Google 几乎可以即时地从其数据库中的超过 15 亿幅图片中搜集出超过 40 万幅“红谷仓”的图片。

关于创建良好的数据库、从程序中访问数据库、扩大数据库以及管理数据库要知道的事情非常多。应用程序员和数据库管理员需要在这种级别上理解数据库以便高效地使用它们。今天,甚至重点关注其他专业的计算机科学家也需要知道关于数据库的知识。例如,一些较新的操作系统在它们的文件系统中使用数据库技术来存储所有的信息,而不仅仅是形式上专属于特定数据库的信息。这给操作系统带来的好处体现在速度、空间节省和数据安全等方面。

在更深的层次上,计算机科学家开发了用于同时在许多用户当中共享数据库访问的算法。例如,像 Amazon.com 这样的站点可能同时为超过 10 万个用户提供服务,将每个用户的选择和购物彼此区分开很重要。同样,在线预订航班座位时,不会给同时在线的两个人允诺相同的座位号也很重要!

计算机科学家还开发了一些算法用于制作数据库的备份副本,以防止由于设备故障而丢失数据的可能性。对于像 Amazon 这样的站点,要求这类算法无需首先停止主数据库的运行即可允许备份,因为站点必须一直正常运行!开发用于可靠、高效地提供这种服务的算法非常具有挑战性。

应该可以轻松地使你确信计算机显著改变了人类的生活方式。像 Internet 和 World Wide Web(万维网)这样的技术给我们提供了巨量的信息。即时消息系统、电子邮件和手机彻底变革了人类的通信方式。警察机关正使用计算机监督系统使世界变得更安全。

虽然所有这些技术主要用于促进人类的善举,但也有可能使用这些技术造成伤害、获得对信息的未经授权的访问或者秘密监视他人。除了需要具有开发这些技术的能力之外,还需要合乎社会和道德标准地使用它们。了解技术对社会的潜在影响与构建技术同样重要,

有时也许甚至更重要。随着越来越多的人开始在他们的日常生活中依赖于计算技术，计算机科学家还必须考虑研究技术所产生的社会问题。

一个常见的误解是：计算机科学只研究计算机硬件和编程。现在应该很清楚，计算机科学涵盖的范围远远不止编写程序这么简单。它包括研究计算机硬件、计算机语言、操作系统、网络、数据库以及计算的社会后果。为了取得良好的效果，计算机科学家必须理解和掌握所有这些领域的科学知识。此外，计算机科学自从 20 世纪 40 年代诞生以来，仍然是一门在快速演进的年轻学科。在下一节中，将从硬件和软件角度简要探讨计算的发展历史。

1.2　计算的发展历史

尽管计算机科学是一个相对年轻的领域，只是在 20 世纪 40 年代人们才开始对它倾注热情，但是人们对计算和计算设备的兴趣则在早得多的时间就开始产生了。算盘是一种古老的计数设备，它被许多人认为是第一种计算设备。

1614 年，苏格兰男爵 John Napier（对数的发明者）发明了一种计算用的设备，它由一系列测杆（通常称为“骨骼”）组成，用于把乘法和除法的复杂过程简化为相对简单的加法和减法任务。一些人把他的发明视作是机械计算的首次尝试。

人们通常把 1642 年第一款机械式计算器 Pascaline（参见图 1-1）的发明归功于 Blaise Pascal（有证据表明莱昂纳多·达芬奇（Leonardo DaVinci）的发明可能比 Pascal 早 150 年）。依据 Pascal 的自传，他开发这种机器用于帮助他作为收税员的父亲完成工作任务。Pascal 的设备只能执行加法和减法运算，但它也可能使用一系列加法和减法运算来执行乘法和除法运算。关于 Pascal 的机器值得注意的是：它能够利用 8 位数字执行计算，并且使用补数（complement）技术执行减法运算。在大多数现代计算机中，都使用了补足加法的减法运算这种相同的技术来实现减法。

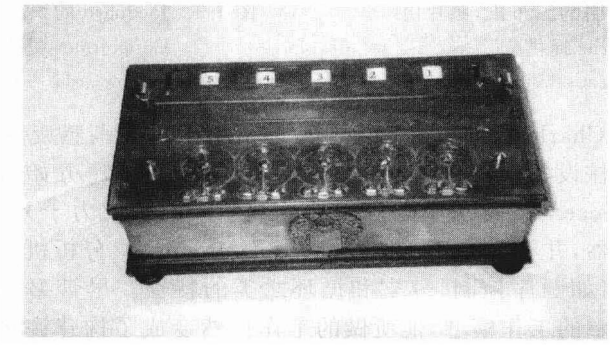

图 1-1　Pascaline，照片由 Yves Serra 提供
（http://pagesperso-orange.fr/yves.serra/）