全球版 | GLOBAL EDITION

国外著名高等院校信息科学与技术优秀教材

Pearson

# 计算机科学概论

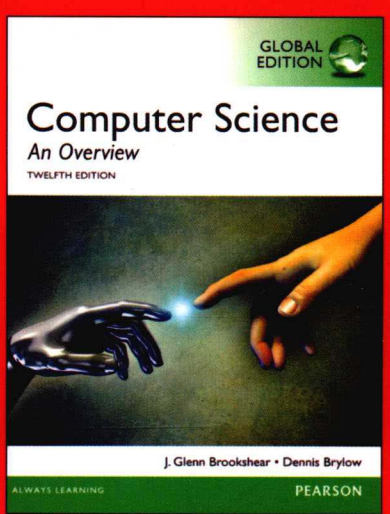
## （第12版）

[美] J. 格伦·布鲁克希尔（J. Glenn Brookshear）
[美] 丹尼斯·布里罗（Dennis Brylow）
著
吕云翔 张竹君 高峻逸 译

# COMPUTER SCIENCE

## AN OVERVIEW

Twelfth Edition




中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

GLOBAL EDITION | 全球版

国外著名高等院校信息科学与技术优秀教材

# 计算机科学概论（第12版）

[美] J. 格伦·布鲁克希尔（J. Glenn Brookshear）
[美] 丹尼斯·布里罗（Dennis Brylow）
著

吕云翔 张竹君 高峻逸 译

# COMPUTER SCIENCE

## AN OVERVIEW

Twelfth Edition

人民邮电出版社

北京

**图书在版编目（CIP）数据**

计算机科学概论 ：第12版 /（美）J.格伦·布鲁克希尔（J. Glenn Brookshear），（美）丹尼斯·布里罗（Dennis Brylow）著 ；吕云翔，张竹君，高峻逸译. -- 北京 ：人民邮电出版社，2019.1
国外著名高等院校信息科学与技术优秀教材
ISBN 978-7-115-47741-5

Ⅰ. ①计… Ⅱ. ①J… ②丹… ③吕… ④张… ⑤高… Ⅲ. ①计算机科学－高等学校－教材 Ⅳ. ①TP3

中国版本图书馆CIP数据核字(2018)第002020号

## 版权声明



◆ 著　[美] J. 格伦·布鲁克希尔（J. Glenn Brookshear ）
[美]丹尼斯·布里罗（Dennis Brylow）
译　吕云翔　张竹君　高峻逸
责任编辑　邹文波
责任印制　彭志环

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路 11 号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
固安县铭成印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：31.75　2019 年 1 月第 1 版
字数：802 千字　2019 年 1 月河北第 1 次印刷
著作权合同登记号　图字：01-2015-8763 号

定价：108.00 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

# 内容提要

本书是计算机科学概论课程的经典教材，充分展现了计算机科学的历史背景、发展历程和新的技术趋势。主要内容包括信息编码及计算机体系结构的基本原理，操作系统和组网及因特网，算法、程序设计语言及软件工程，数据抽象和数据库系统，图形学的主要应用以及人工智能，计算理论。本书在内容编排上由具体到抽象层层递进，适合教学安排。此外，书中还包含大量的图、表和示例，有助于增进读者对知识的理解与把握。

第 12 版主要是将 Python 程序设计语言方面的介绍纳入了重点章节，除了增加与 Python 相关的内容，几乎每一章都能看到对前一版对应章节的修订、更新以及修正。

本书适合作为高等院校计算机以及相关专业本科生的教材，也可作为有意在计算机方面发展的专业读者的入门参考书。

# 前言

本书是计算机科学的一本入门书籍。本书力求保持学科广度，同时还兼顾了深度，对涉及的主题进行了中肯的评价。

## 读者对象

这本书适合计算机科学专业的学生，也适合主修其他学科的学生。大多数计算机科学专业的学生最初都误以为计算机科学就是编程、网页浏览和因特网上的文件分享，因为这就是他们实际看到的一切。但是，计算机科学可远不止这些。刚开始学习计算机科学的学生需要了解他们将要主修的这门学科的广度。不妨认为呈现这种广度就是本书的宗旨。本书为学生提供了计算机科学的总体概述，基于此，他们将可以理解该领域未来要学的学科之间的关联和相互关系。实际上，本书采用的方法也是自然科学入门课程的常用模式。

如果想要很好地融入当今这个技术社会，其他学科的学生也需要了解这种宽泛的背景知识。适合这类学生的计算机科学课程应该提供对整个学科的真正的实际剖析，而不是仅仅介绍如何上网或者使用几种常用的软件。当然，有合适的地方如何对使用各种软件进行培训，但是本书是用作计算机科学教育的教科书。

在编写本书时，我们就力求让非技术类的学生可以看懂。因此，本书已成功成为不同学科学生的教科书，读者群体囊括了从高中到研究生不同教育程度的学生。这次的第12版也会继续保持这一传统。

## 第12版新增的内容

编写第12版时一个根本性的原则是将Python编程语言包含到重要的章节中。在前几章中，这些补充的内容是选学的。到第5章，我们将以前版本中Pascal风格的表示法换成了Python或Python风格的伪代码。

这对一本历史上一直努力避免突出任何一种特定语言的书而言是一个重大的改变。使我们做出这个改变有许多原因。首先，本书已经包含相当多的代码，而且是用多种编

程语言编写的，有几章里还有很详细的伪代码。从一定程度上，读者已经能理解相当数量的语法规则，此时将这些语法规则重新定向到他们后续课程中实际会见到的一种语言似乎是很合适的。其次，更重要的是，越来越多使用本书做教科书的导师发现，即使是一本广度优先的计算机入门书，没有编程工具来进行自行探索和实验，学生们也很难掌握其中的一些主题。

但是为什么选择 Python 呢？选择语言总是一件充满争议的事，选择任何一种语言，有多少人支持就有多少人反对。但是具有下列特点的 Python 是很好的选择。

- 语法规则简洁易学。
- 输入/输出原语简单。
- 数据类型和控制结构与之前版本中的伪代码表示密切符合。
- 支持多种程序设计范式。

Python 是一门成熟的语言，有充满活力的开发社区以及丰富的在线资源，能方便学生今后的学习。在某种标准下，Python 仍是业内最常用的 10 种语言之一，而且被越来越多地用于计算机科学入门课程。Python 在非计算机专业的入门课中尤其受欢迎，并且，作为为计算机科学应用课选择的语言，Python 在其他诸如物理、生物等的 STEM 领域也得到了广泛的使用。

## 章节组织

本书遵循从具体到抽象、自下而上的内容编排。这种编排结构对教学十分有利，因为每个主题都能引出下一个主题。从信息编码、数据存储和计算机体系结构（第 1 章和第 2 章）的基本原理开始学习；进而发展到学习操作系统（第 3 章）和计算机网络（第 4 章）；然后研究算法、程序设计语言和软件开发（第 5 章～第 7 章）；随后探索如何更好地访问信息（第 8 章和第 9 章）；紧接着学习计算机图形学技术（第 10 章）和人工智能（第 11 章）的主要应用；最后以介绍抽象的计算理论（第 12 章）来结束全书。

本书的组织自然连贯，各个章节又相互独立，通常可以作为单独的单元来阅读，也可以重新安排形成不同的学习顺序。事实上，本书常被用作相关课程的教科书，教学的顺序多种多样。其中一种顺序是从第 5 章和第 6 章（算法和程序设计语言）开始，然后再按需回到更早的章节。我还了解到一门课程是从第 12 章有关可计算性的内容开始的。本书还曾经被用于“高级研讨班”——作为深入其他不同领域项目的基础。对于那些不太以技术为主的课程，学习的重点可以放在第 4 章（组网及因特网）、第 9 章（数据库系统）、第 10 章（计算机图形学）和第 11 章（人工智能）。

在每章的开始部分，我们已经用星号标识出可以作为选学的部分。这些部分要么是探讨更专业的话题，要么是对传统话题做更深入的探索。我们的目的仅仅是为想采取其他学习顺序的人提供一些建议。当然，还有其他的方式。特别是，如果读者想快速阅读，我们建议采取下面的顺序。

| 章节 | 主题 |
| --- | --- |
| 1.1～1.4 | 数据编码和存储基础 |
| 2.1～2.3 | 计算机体系结构和机器语言 |

续表

| 章节 | 主题 |
| --- | --- |
| 3.1～3.3 | 操作系统 |
| 4.1～4.3 | 组网及因特网 |
| 5.1～5.4 | 算法和算法设计 |
| 6.1～6.4 | 程序设计语言 |
| 7.1～7.2 | 软件工程 |
| 8.1～8.3 | 数据抽象 |
| 9.1～9.2 | 数据库系统 |
| 10.1～10.2 | 计算机图形学 |
| 11.1～11.3 | 人工智能 |
| 12.1～12.2 | 计算理论 |

本书有几条贯穿始终的主线。其中之一是计算机科学的动态变化，本书从历史的角度反复呈现各个主题，然后讨论其当下的状态，并言明研究方向；另一条主线是抽象的作用以及用抽象的工具控制复杂性的方式，这条主线从第 0 章开始引入，然后在操作系统体系结构、网络、算法开发、程序设计语言、软件工程、数据组织和计算机图形学等内容中反复体现。

## 致教师

本书所含内容很难在一个学期内全部学完，因此请果断跳过与自己教学目标无关的主题，或者重新安排合理的学习顺序。尽管本书是按顺序进行组织的，但是主题之间在很大程度上相互独立，因此可以按照需要选择。本书应当被用于课程参考书而不是圈定课程的内容。我们建议留下部分内容让学生自己课下阅读，如果认为所有的内容都需要在课堂上讲解的话，就太低估学生了！我们应当引导学生学会自学。

我们认为有必要说明一下本书自底而上、从具体到抽象的组织结构。作为学者，我们总认为学生会欣赏我们对一个学科的观点——这些观点常常是我们在这一领域工作多年形成的。但作为老师，我认为从学生的角度呈现教学内容效果更好。这就是为什么本书首先介绍数据的表示/存储、计算机体系结构、操作系统以及网络。这些话题都是容易引起学生共鸣的，他们极有可能已经听说过 JPEG、MP3 等术语；他们可能已经在 CD 或者 DVD 上刻录过信息；他们可能已经买过计算机的各个组件；他们可能已经与操作系统打过交道；他们可能已经上过网。从这些话题开始，学生们会发现许多困惑已久的“为什么”得到了解答，从而认为本书是实用的而非理论的。由此出发，就可以比较自然地过渡到更加抽象的问题上：算法、算法结构、程序设计语言、软件开发方法、可计算性以及复杂度等我们这一领域的人认为计算机学科的主要内容。正如前文所述，本书并不强制你使用自底而上的教学顺序，但是我们建议教师们可大胆尝试这种教学方法。

我们都知道，学生学到的东西远比我们直接教给他们的多，而且他们吸收潜移默化传授的东西往往更好。这在涉及“传授”如何解决问题时显得很重要。学生并不能通过学习解决问题的方法来成为问题解决者的，他们得真的解决问题才行，而且不能仅仅是解决那些精心设计过的“教科书式的问题”。所以本书包含大量的问题，而且有些问题故意设计得比较模棱两可，这意味着有些问题并不是必须只有一个正确答案或者只有一种正确的解决方法。我建议教师们使用并拓展这些问题。

其他适合“潜移默化学习”的主题还有职业精神、伦理和社会责任感。我们认为这些内容不应该仅仅是书上附加的独立的一章，而应该在所有相关的内容被提及时加以探讨。而这正是本书的编排方式。你会发现，3.5 节、4.5 节、7.9 节、9.7 节以及 11.7 节分别在操作系统、网络、软件工程、数据库系统和人工智能部分提到了安全、隐私、责任和社会意识的问题。你还会发现每一章都有“社会问题”小节，用以激发学生思考本章内容与现实社会之间的关系。

感谢你考虑将本书用于你的课程。不管您是否会选择本书作为教材，我都希望你认同本书为计算机科学教育文献做出了贡献。

## 教学特色

本书是作者从事教育事业多年的成果，因此非常有助于辅助教学。其中，最突出的一点是本书有丰富的问题来提高学生的参与度——在第 12 版里有超过 1 000 道题，包括“问题与练习”“复习题”与“社会问题”。

“问题与练习”部分出现在每节最后（除了第 0 章），用于回顾刚刚学习的内容，延伸之前的讨论，或者提示后续有关章节的内容。在附录 F 中有答案。

每一章的最后都有“复习题”（除了第 0 章），这些题目被设计成课后作业，内容涵盖整章的内容，但书中没有答案。

“社会问题”同样列在每章的最后，鼓励学生思考和讨论，其中许多问题可以用来作为调查研究的问题，最终以短篇报告或口头报告的形式提交。

每章最后还列出了“课外阅读”，内容主要是与本章主题相关的其他参考资料。前言、正文还有侧边栏中列出的网站的内容也有很好的参考作用。

## 补充材料

本书的配套网站 www.pearsonhighered.com/brookshear 提供了丰富的参考资料。下列内容面向所有读者。

- 每章的实践活动，可以拓展本书内容并关联相关主题。
- 每章的“自我测试”部分，帮助读者加深对书中内容的理解。
- 与本书内容配套的介绍 Java、C++基本原理的手册，它在教学顺序上与本书是兼容的。

此外，教师可以通过 Pearson Education 的教师资源中心获取下列教学资料（登录 www.pearsonhighered.com 网站或者咨询本书销售代表如何下载）。

- 包含本书“复习题”答案的教师指南。

- 讲课用的 PPT。
- 测试题库。

本书的勘误表（如果有的话）在 www.mscs.mu.edu/~brylow/errata/上。

## 致学生

格伦·布鲁克希尔是那种有点不愿墨守成规的人（他的朋友们可能会觉得远不是一点），所以他在写本书的时候常常不肯接受别人的建议。特别值得一提的是，很多人批评他书中有些内容对初学者而言过于超前了。但是我们认为如果一个话题是有意义的，那么即使学术界认为这个主题可能是个“超前的话题”，也依然是一个有意义的主题。你值得拥有一本能呈现计算机科学完整画卷的书，而不是一本“缩水”的版本——只包含人为简化了的被认为适合初学者的内容。因此，我们没有回避任何主题，而是探寻更好的解释。我们试图提供足够的深度来让读者领略计算机科学真实的一面，正如挑选菜谱里的调味品一样，我们列出了全部的话题，你可以选择跳过部分主题，但是如果你愿意的话，它们就在那供你品尝，而且我们鼓励你去尝试。

我们还要指出，在任何技术相关的课程上，你今日所学可能并不是你将来需要的知识。这个领域是动态变化的，而这也正是让人激动的部分。本书将呈现目前的学科内容，也会提供历史的视角。有了这些背景知识，你就可以与技术一同成长。我们鼓励你现在就开始行动，不局限于本书而是展开探索，学会学习。

感谢你的信任，选择阅读本书。作为作者，我们有责任写一本不浪费你时间的有价值的书。我们期待你认为我们尽到了这份责任。

## 致谢

首先，我要感谢称本书为“自己的孩子”的格伦·布鲁克希尔，他编写的过去 11 个版本横跨近 30 年间计算机科学领域的快速发展和纷繁变化。尽管本书他第一次允许合著者全程审查修订，但绝大多数内容收录的依然是格伦的“声音”，并由他指导。本书的任何瑕疵都是我造成的，而优雅的底层框架都是格伦先生搭建的。

我和格伦一同感谢支持本书（阅读和使用之前版本）的人，我们深感荣幸。

David T. Smith（宾夕法尼亚印第安纳大学）在与我合著第 11 版时贡献良多，许多内容仍保留在第 12 版里。David 的仔细阅读和对补充材料的仔细检查也是至关重要的工作。Andrew Kuemmel（Madison West）、George Corliss（Marquette）和 Chris Mayfield（James Madison）都对本版的草稿提出了有价值的回馈、洞察和鼓励；James E. Ames（Virginia Commonwealth）、Stephanie E. August（Loyola）、Yoonsuck Choe（Texas A&M）、Melanie Feinberg（UT-Austin）、Eric D. Hanley（Drake）、Sudharsan R. Iyengar（Winona State）、Ravi Mukkamala（Old Dominion）和 Edward Pryor（Wake Forest）对 Python 的修订提出了有建设性的意见。

其他对本版或之前版本有过贡献的人有 J. M. Adams、C. M. Allen、D. C. S. Allison、E. Angel、R. Ashmore、B. Auernheimer、P. Bankston、M. Barnard、P. Bender、K. Bowyer、P. W. Brashear、C. M. Brown、H. M Brown、B. Calloni、J. Carpinelli、M. Clancy、R. T. Close、

D. H. Cooley、L. D. Cornell、M. J. Crowley、F. Deek、M. Dickerson、M. J. Duncan、S. Ezekiel、C. Fox、S. Fox、N. E. Gibbs、J. D. Harris、D. Hascom、L. Heath、P. B. Henderson、L. Hunt、M. Hutchenreuther、L. A. Jehn、K. K. Kolberg、K. Korb、G. Krenz、J. Kurose、J. Liu、T. J. Long、C. May、J. J. McConnell、W. McCown、S. J. Merrill、K. Messersmith、J. C. Moyer、M. Murphy、J. P. Myers、Jr.、D. S. Noonan、G. Nutt、W. W. Oblitey、S. Olariu、G. Riccardi、G. Rice、N. Rickert、C. Riedesel、J. B. Rogers、G. Saito、W. Savitch、R. Schlafly、J. C. Schlimmer、S. Sells、Z. Shen、G. Sheppard、J. C. Simms、M. C. Slattery、J. Slimick、J. A. Slomka、J. Solderitsch、R. Steigerwald、L. Steinberg、C. A. Struble、C. L. Struble、W. J. Taffe、J. Talburt、P. Tonellato、P. Tromovitch、P. H. Winston、E. D. Winter、E. Wright、M. Ziegler，还有一位匿名者。我们对他们中的每一位致以诚挚的感谢。

正如之前提到的，本书的配套网站上有 Java 和 C++的手册，讲解这两种语言的基础知识，与本书相得益彰。这些是由 Diane Christie 编写的，谢谢你，Diane。另外还得感谢 Roger Eastman，他是网站上每章实践活动题的出题者。

我们还要谢谢支持本书的 Pearson 出版集团的员工，尤其是 Tracy Johnson、Camille Trentacoste 和 Carole Snyder，与他们合作非常愉快，在合作过程中，他们用智慧给本书带来很大的提高。

最后我要谢谢我的太太 Petra，这一版是专门献给她的。她的耐心和毅力远远超过我，而本书在她持久的影响下也变得更好。

D.W.B.

# 目 录


第 0 章　绪论　1

0.1　算法的作用　2
0.2　计算机器的由来　3
0.3　学习大纲　7
0.4　计算机科学的首要主题　9

第 1 章　数据存储　16

1.1　位和位存储　17
1.2　主存储器　22
1.3　海量存储器　25
1.4　用位模式表示信息　29
1.5　二进制系统　34
1.6　整数存储　38
1.7　小数存储　43
1.8　数据与程序设计　47
1.9　数据压缩　52
1.10　通信差错　56

第 2 章　数据操控　65

2.1　计算机体系结构　66
2.2　机器语言　68
2.3　程序执行　73
2.4　算术/逻辑指令　79
2.5　与其他设备通信　82
2.6　数据操控编程　87
2.7　其他体系结构　93

第 3 章　操作系统　102

3.1　操作系统的历史　103
3.2　操作系统的结构　106
3.3　协调机器的活动　112
3.4　处理进程间的竞争　114
3.5　安全性　118

第 4 章　组网及因特网　125

4.1　网络基础　126
4.2　因特网　133
4.3　万维网　141
4.4　因特网协议　148
4.5　安全性　152

第 5 章　算法　163

5.1　算法的概念　164
5.2　算法的表示　166
5.3　算法的发现　173
5.4　迭代结构　177
5.5　递归结构　186
5.6　效率与正确性　192

第 6 章　程序设计语言　206

6.1 历史回顾 207
6.2 传统的程序设计概念 213
6.3 过程单元 223
6.4 语言实现 229
6.5 面向对象程序设计 236
6.6 程序设计并发活动 242
6.7 说明性程序设计 244

第 7 章 软件工程 253

7.1 软件工程学科 254
7.2 软件生命周期 256
7.3 软件工程方法学 259
7.4 模块化 261
7.5 行业工具 267
7.6 质量保证 273
7.7 文档编制 276
7.8 人机界面 277
7.9 软件所有权和责任 279

第 8 章 数据抽象 285

8.1 基本的数据结构 286
8.2 相关概念 289
8.3 数据结构的实现 290
8.4 一个简短的案例 302
8.5 定制的数据类型 307
8.6 类和对象 310
8.7 机器语言中的指针 311

第 9 章 数据库系统 319

9.1 数据库基础 320
9.2 关系模型 324
9.3 面向对象数据库 332
9.4 维护数据库的完整性 334
9.5 传统文件结构 337
9.6 数据挖掘 344
9.7 数据库技术的社会影响 345

第 10 章 计算机图形学 352

10.1 计算机图形学的范围 353
10.2 3D 图形学概述 354
10.3 建模 356
10.4 渲染 362
10.5 处理全局照明 370
10.6 动画 373

第 11 章 人工智能 379

11.1 智能与机器 380
11.2 感知 384
11.3 推理 389
11.4 其他研究领域 399
11.5 人工神经网络 402
11.6 机器人学 409
11.7 后果的思考 411

第 12 章 计算理论 419

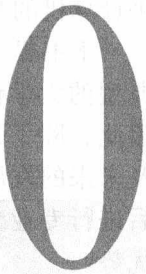
12.1 函数及其计算 420
12.2 图灵机 422
12.3 通用程序设计语言 425
12.4 一个不可计算的函数 430
12.5 问题的复杂性 434
12.6 公钥密码学 441

附录 450

附录 A ASCII 码 451
附录 B 处理二进制补码表示的电路 452
附录 C 一种简单的机器语言 454
附录 D 高级编程语言 456
附录 E 迭代结构与递归结构的等价性 458
附录 F 问题与练习答案 460

第 0 章

# 绪论



在开篇的这章，我们来了解计算机科学的范围，建立一种历史的观点，为今后的学习奠定基础。


0.1 算法的作用
0.2 计算机器的由来
0.3 学习大纲
0.4 计算机科学的首要主题

计算机科学这门学科，是为计算机设计、计算机编程、信息处理、解决问题的算法方案和算法过程本身等主题建立一个科学的基础。计算机科学为当今的计算机应用和未来计算机的基础构造提供了支持。

本书提供了对计算机科学的综合介绍。我们将会调查大范围的主题，包括组成一个典型的大学计算机课程的大多数主题。我们得感谢这个领域的博大精深和动态发展。因此，除了这些主题本身，我们还会涉及它们的历史发展、当前调查的发展状态，以及未来的发展前景。我们的目标是建立对计算机科学实用性的理解，可以帮助到希望日后进行专业化科学研究的人，也能帮助其他领域的人在这个越来越技术化的社会占据优势。

## 0.1 算法的作用

我们从算法这一计算机科学最基础的概念开始。通俗地讲，算法就是定义了任务如何一步步执行的一套步骤（第 5 章会更精确地表述）。举例来说，生活中有很多地方用到算法：烹饪时用到的食谱；寻路时依靠的指南；奏乐时需要的乐谱；操作洗衣机时要遵照的步骤以及表演魔术时完成的系列手法（见图 0.1），这些都是“算法”的体现。

**效果：** 表演者从一副普通的扑克牌中取出若干张牌，充分洗牌后正面朝下摆放在桌面上。然后，表演者会根据观众的要求相应地翻出红牌或者黑牌。

**秘诀：**

第1步：从一副普通的扑克牌中选出10张黑牌和10张红牌。根据颜色分成两组，正面朝上摆在桌子上。

第2步：告诉观众你已经选取了若干红牌和若干黑牌。

第3步：拿起红牌，装作整理成一摞的样子，用左手正面朝下拿好牌，同时用右手的大拇指和食指挤压这一摞牌的两端，把牌面向下推，使每张牌呈轻微向下的弧形。然后把这组牌正面朝下放在桌子上并宣布“这是其中的红牌”。

第4步：拿起黑牌，采用和第3步一样的方式，使每张牌呈轻微向上的弧形。然后把这些牌正面朝上放在桌子上并宣布“这是其中的黑牌”。

第5步：把黑牌放在桌子上后，立即用两只手混合两组牌（仍然正面朝下），平铺在桌面上。告诉观众你已经彻底洗好牌。

第6步：只要桌面上还有正面朝下的扑克牌，就重复执行以下步骤。

① 请观众要一张红牌或黑牌。

② 如果观众要红牌，并且桌面上尚有倒扣的呈凹型的牌，翻开其中一张并说“这是一张红牌”。

③ 如果观众要黑牌，并且桌面上尚有倒扣的呈凸面的牌，翻开一张并宣布“这是一张黑牌”。

④ 否则，就告诉大家，桌面上没有所要求颜色的牌了，然后翻开剩下的所有牌来证明你说的话。

图 0.1 魔术用到的算法

在一个机器（如计算机）执行任务之前，必须先找到与之兼容的执行该任务的算法。一个算法的表示称作一个**程序**（Program）。为了方便人类读写，计算机程序通常是打印在纸上或显示在计算机屏幕上；而为了便于机器执行，程序要以一种与机器技术兼容的形式编码。开发程序并将之编码成与机器兼容的形式然后输入机器中的过程就叫作**程序设计**（Programming）。程序及其体现的算法共同称为**软件**（Software），而机器本身叫作**硬件**（Hardware）。

对算法的研究始于数学。事实上，早在当今的计算机发展之前，研究算法已经是数学中非常重要的活动了；研究的目的是要找一组能解决某一特定类型所有问题的指令。这类早期研究中最著名的是求解两个多位数商的长除算法。另一个例子是

古希腊数学家欧几里得发现的欧几里得算法，用于求解两个正整数的最大公约数，如图 0.2 所示。

**描述**：算法假定输入是两个正整数，并计算两个数的最大公约数。
**步骤**：
第一步：将输入的两个数中较大的赋给$M$，较小的赋给$N$。
第二步：计算$M$除以$N$，余数称作$R$。
第三步：如果$R$不为0，将$N$的值赋给$M$，将$R$的值赋给$N$，然后回到第二步；否则，最大公约数是当前$N$的值。

图 0.2 求两个正整数最大公约数的欧几里得算法

一旦我们找到了执行一个任务的算法，执行任务时就不再需要理解算法的原理。相反，任务的执行缩减为只是遵循系列指令的过程（我们可以根据长除算法得到商或者根据欧几里得算法得到最大公约数，而不需要明白算法的工作原理）。从某种意义上，解决手头问题所需的智慧被编码到了算法里。

通过算法的方式捕获并传达智能（或者至少是智能行为）使我们能够让机器执行有意义的任务。因此，机器表现出来的智能受限于算法本身可以传达的智能。只有执行某任务的算法存在时，我们才可以制造出执行该任务的机器。换言之，如果执行某任务的算法还不存在，那么这个任务就已经超出该机器的能力范围了。

在 20 世纪 30 年代，伴随 Kurt Gödel 发表了不完备性理论的论文，确定算法能力局限性成为了数学学科中的研究命题。这一理论从本质上阐述了在任何包含传统算术系统的数学理论中，总有通过算法方式不能确定其真伪的命题。简单来说，对于算术系统的任何完整研究都超出了算法活动的能力范围。这一发现动摇了数学领域的基础，而对算法能力的研究相继而来，后者就是当今计算机领域的开端。的确，正是对算法的研究组成了计算机科学的核心。

## 0.2 计算机器的由来

今天的计算机有着广泛的族谱。早期的计算设备之一是算盘。历史表明，算盘很可能起源于古代中国，并得到早期希腊人和罗马人的使用。算盘本身的结构非常简单，一个矩形框上固定着一组小棍，小棍上串着许多珠子，如图 0.3 所示。在珠子被上下拨动时，它们的位置就代表着存储的值。正是这些珠子的位置代表了这个“计算机”表示和存储的值。算盘需要人的操作来控制一个算法的执行。因此，算盘本身仅仅是一个数据存储系统，必须和人一起才能组合成一个完整的计算设备。

从中世纪之后到近代，人们开始追寻更复杂精确的计算设备。借助齿轮技术，几个发明相继问世。采用这种技术的发明家有法国的 Blaise Pascal（1623—1662 年）、德国的 Gottfried Wilhelm Leibniz（1646—1716 年）和英国的 Charles Babbage（1792—1871 年）。这些机器通过齿轮的位置来表示数据，在规定了初始齿轮的位置后，机械地输入数据。观察最终齿轮的位置可以得到 Pascal 和 Leibniz 设计的计算机器的计算结果。而 Babbage 构想的机器可以将计算结果打印到纸上，这样就可以排除抄写错误的可能性。

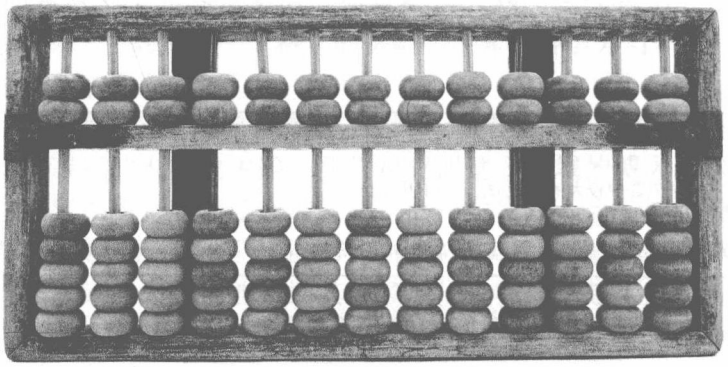


图 0.3 中国木质算盘（Pink Badger/Fotolia）

至于执行算法步骤的能力，我们可以看到这些机器的灵活性大大增强。Pascal 设计的机器只能进行加法运算，因此，机器本身已经嵌入需要执行的系列合适的步骤。相似地，Leibniz 的机器也将算法牢牢嵌入自身结构中，但操作员是可以在系列算术运算中选择的。Babbage 的差分机（只构建了一个展示模型）可以被修改来执行多种运算，但他的分析机（因缺乏资金而从未制造）可以读取纸卡片上以小孔形式表达的指令。因此，Babbage 的分析机是可编程的。事实上，Augusta Ada Byron（Ada Lovelace）发表了一篇论文演示了如何通过 Babbage 的分析机进行编程执行不同的计算，因此她通常被称为世界上第一个程序员。

Babbage 不是第一个提出以纸片上小孔的形式来与机器交流算法的人，他的灵感来自于 Joseph Jacquard（1752—1834 年），后者在 1801 年发明了一种织布机，织布的步骤由木制或纸质的大卡片上孔洞的样式决定。用这种方式，织布的算法可以很容易改变，以此来生成不同的编制样式。Herman Hollerith（1860—1929 年）也同样从 Jacquard 的想法中获益，他利用这种以纸片上孔洞样式改变信息的理念，加速了美国 1890 年人口普查的制表过程。（正是因为有了 Hollerith 的这种做法，才有了后来的 IBM。）这种卡片最终被称为穿孔卡片而广为人知，直到 20 世纪 70 年代仍然是人与计算机交互的流行方式。

在 19 世纪，即使有资金支持，也不具备制造 Pascal、Leibniz 和 Babbage 设想的复杂齿轮驱动机器的技术。但随着 20 世纪初期电子技术的进步，人们克服了这一困难。在此期间，George Stibitz 1940 年在贝尔实验室建造了电子机械设备；Howard Aiken 和一组 IBM 的工程师于 1944 年，在哈佛大学中完成了马克一号（Mark I）。这些机器大量应用了电子控制的机械式继电器。在某种程度上，它们几乎一经建造出来就是过时的了，因为其他研究者正在应用电子管技术建造完全电子化的计算机。第一个这样的电子管机器显然是 Atanasoff-Berry 机器，由 John Atanasoff 及其助手 Clifford Berry 在 1937—1941 年间建于爱荷华州立学院（现在是爱荷华州立大学）。另一个这样的机器叫作巨人（Colossus），是在英国的 Tommy Flowers 的指导下建造的，用于在二战后期破译德军的情报。（事实上，这样的机器有十多台，但是由于军方的机密和国家安全问题，这些机器不在“计算机家庭树”上。）不久，出现了一些更灵活的机器，比如宾夕法尼亚大学莫尔电子工程学院的

John Mauchly 和 J. Presper Eckert 研制的 ENIAC（Electronic Numerical Integrator And Calculator，电子数字积分器和计算器，见图 0.4）。

图 0.4 三位女士在操作 ENIAC 的主控制板，此时 ENIAC 尚在莫尔学院，后来被移至美国的陆军弹道学研究实验室（图片由美国军队提供）

在此之后，计算机器的历史一直与先进科技紧密联系，包括晶体管的发明（物理学家 William Shockley、John Bardeen 和 Walter Brattain 因此获得了诺贝尔奖）和后来集成电路的发展（Jack Kilby 因此得到了诺贝尔物理奖）。随着技术几十年间的发展，20 世纪 40 年代曾经占据整间屋子的大机器缩小到了单柜大小。与此同时，计算机器的处理能力开始每两年就翻一翻（时至今日仍保持着这一趋势）。随着集成电路的发展，计算机内很多组件作为集成电路被封装在玩具大小的塑料上的芯片中，可以在开放市场上方便地买到。

台式机的发展是计算机普及化的重要一步，其起源为一些计算机爱好者利用芯片组合成家用计算机。正是在这些“地下的”计算机爱好活动中，Steve Jobs 和 Stephen Wozniak 制造了一台有商业价值的家用计算机，并于 1976 年成立了苹果公司来生产和销售他们的产品。当时市场上销售类似商品的公司还有 Commodore、Heathkit 和 RadioShack。虽然这些产品在计算机爱好者中广受欢迎，但是商业界并没有广泛接纳它们，而是继续青睐完备的 IBM 公司生产的大型机来满足他们大部分的计算需求。

1981 年，IBM 公司公布了他们的第一台台式机，称作“个人电脑”（PC）。PC 的基础软件是一个新成立的名叫“微软”的公司研发的。PC 一经推出就迅速获得成功，并使得台式机作为一种完备的商品而令商业界刮目相看。时至今日，PC 这个术语广泛指代所有从 IBM 最初那台台式机演化而来的机器（来自各个厂商），其中大多数仍与微软公司的软件一同销售。然而，有时候，PC 这个词也与通用的术语台式机或笔记本交换使用。