国外著名高等院校
信息科学与技术优秀教材

PEARSON Addison Wesley

计算机科学概论(第8版)

Computer Science

AN OVERVIEW Eighth Edition

〔美〕J. Glenn Brookshear 著

(英文版)

人民邮电出版社
POSTS & TELECOM PRESS

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8版）

（英文版）

Computer Science: An Overview, Eighth Edition

[美] J. Glenn Brookshear 著

人民邮电出版社

图书在版编目（CIP）数据

计算机科学概论：第8版/（美）布鲁西尔（Brookshear, J. G.）著.
—北京：人民邮电出版社，2006.7
（国外著名高等院校信息科学与技术优秀教材）

ISBN 7-115-14918-6

Ⅰ. 计… Ⅱ. 布… Ⅲ. 计算机科学—教材—英文 Ⅳ. TP3

中国版本图书馆CIP数据核字（2006）第069499号

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8版）（英文版）

◆ 著 ［美］J. Glenn Brookshear

责任编辑 李 际

◆ 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 电子函件 315@ptpress.com.cn

网址 http://www.ptpress.com.cn

北京隆昌伟业印刷有限公司印刷

新华书店总店北京发行所经销

◆ 开本：800×1000 1/16

印张：36

字数：819千字 2006年7月第1版

印数：1－4 000册 2006年7月北京第1次印刷

著作权合同登记号 图字：01-2006-2741号

ISBN 7-115-14918-6/TP・5497

定价：49.00元

读者服务热线：(010)67132705 印装质量热线：(010)67129223

内容提要

本书内容覆盖了计算机科学各学科的主要领域，分别以历史的眼光、发展的角度、当前的水平以及现阶段研究的方向几个方面，对各领域的知识作了全景式的阐述。第8版中还增加和更新了有关内容，密切关注这些领域的最新进展。全书深入浅出、图文并茂，内容安排灵活，便于读者学习。通过本书，读者可以对计算机科学这一学科有一个全面的了解。书中每节都有问题与练习题，每章都有复习题，每章后都附有与本章内容相关的那些需要关心或可能引起争议的社会问题供读者思考、讨论。书后还提供了为深入学习有关专题内容应阅读的参考书目。这些都使本书内容和系统更加完整，更能激发学生的学习兴趣，也便于教师教学。

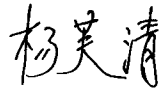
本书适合作为高等学校计算机概论或计算机科学基础课程的教材。对于计算机工作者及希望全面了解计算机科学的一般读者，本书也是一本优秀的基础读物。

序　言

在计算机学科的大学教育中，有一门重要的课程，即该学科各专业的学生都必须学习的一门专业基础课，在我国一般称作“计算机概论”。学生从中学进入大学，开始正规而系统地学习计算机专业课程，需要首先对计算机科学技术的基础知识有一个概括而准确的了解，否则其他任何一门专业课的教学都会遇到许多障碍。所以这门课程对于计算机软件与理论、计算机体系结构、计算机应用技术等专业的教学都是非常重要的。

J．Glenn Brookshear 著的《计算机科学概论》（Computer Science：an Overview）就是这样一本适合作为上述课程教材的好书。该书在美国哈佛大学、加州大学等各所大学被采用。自第一版之后，作者根据计算机科学技术的新发展不断地对该书进行更新和补充，目前已经是第 8 版。书中介绍了计算机硬件、软件、数据库和计算理论等方面的内容。对这些内容的论述深浅适当，文字通俗易懂而又保持简练和准确；每一节都带有精心挑选的习题；给出的插图也颇具匠心，能够很好地表现书中阐述的内容。总之，这是一本很值得引进和推广的好教材。

在我国，改革开放以来计算机科学技术的学科建设和教材建设一直在稳步发展。各高校的教师为此付出了大量心血，写作出版了许多高质量的教材。其中有许多教材不但具有很好的学术水平，而且适合我国的国情与文化背景。同时，学习和借鉴国际上先进的科学技术和优秀文化，是培养人才的需要。有选择地引进国外的优秀教材，必将有效地促进我国教育事业的健康发展。这本书的影印出版，将对我国的计算机专业基础课的教学和教材建设起到良好的作用。它也可以用于非计算机专业的计算机教学和面向计算机产业界的技术培训。



中国科学院　院士
北京大学　信息与工程科学学部　主任

前　言

本书是对计算机科学的初步概览，在阐明每个主题的内容时，有足够的深度和广度。

本书面向的读者

这本教程既适合主修计算机科学专业的学生，也适合其他专业的学生。大多数计算机科学专业的学生在刚开始学习时都有这样的错觉，似乎计算机科学只是程序设计和网站浏览，因为他们平时所见到的几乎就是这些。然而，计算机科学的内容远不止这些。把计算机科学作为专业的学生从一开始就需要拓展对这门学科的认识，本书的目的就在于此。本教程概述了计算机科学，既有一定的深度，又有一定的广度。学生们从这个基础性的介绍中能够领悟到计算机科学的真谛以及它与该领域其他课程的相互关系。

对于其他专业的学生来说，为了理解他们生活在其中的技术社会，这些知识也是有用的。为非计算机专业提供的计算机课程应该讲述整个学科的基本知识，不应停留在只是介绍流行的软件包和因特网的应用上。这种鸟瞰式通论的方法是自然科学引导性课程所用的模式，也是我写本教程所遵循的模式。因而，本教程的前几个版本已成功地在范围广泛的多种专业的课程中使用。这一版将延续这个传统。

简言之，本教程的理念是要成为通常称之为 CS0 型引导性的计算机科学通论课程。而且，如由 IEEE/ACM 联合工作组编写的“计算课程 2001”和“计算课程 2003：计算机科学基础课程指导纲要”主张广度优先的课程，对于考虑这类课程的任何人来说，本教程也是有意义的。的确，自第 1 版以来，本教程已经成为广度优先课程变迁的非正式标准。

本书的组织结构

本教程是按照自底向上的方法，即从具体到抽象的方法组织的。这种安排使课程前后主题连贯，结构合理。本教程从计算机体系结构的基本知识开始（第1章、第2章），然后讲述软件及其开发过程（第3章～第7章），接着讲述数据组织和数据存储的问题（第8章、第9章），最后，在探究计算机技术当前的和未来的应用中（第10章、第11章）结束。

在编写本教程时，我考虑到要设置一些情节。因此，当许多学生说他们是像阅读小说那样阅读本教程时，我并不感到惊讶。另一方面，本教程划分成了许多独立的章节，它们既可以作为单独的单元来学习，也可以重新组织顺序以另外的方式进行学习。确实，本教程用作教材时经常按照各种次序进行重新组合。其中一种方法是，从第5章和第6章（算法和程序设计语言）开始，然后再按需要回到前面的章节。相反，我知道也有的是从第11章有关可计算性的知识讲起的。在另一些讲授方法中，本教程用作“高年级顶端班”的教材，作为学生转入不同领域前的一个主干课程。

对于那些需要缩略性版本的读者，我建议按照下列顺序阅读：

章节	标题
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
3.1~3.3	操作系统
4.1~4.3	组网和因特网
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
8.1~8.2	数据抽象
9.1~9.2	数据库系统
10.1~10.3	人工智能
11.1~11.2	计算理论

除了整体情节外，还有几个贯穿全书的论题。论题之一是计算机科学是不断发展变化的。本教程以历史的观点来陈述各个专题，讨论这些专题的技术状况，并指出当今研究的一些方向。另一个论题是关于抽象的作用以及抽象工具用于控制复杂性的方法。该论题在第10章中引入，然后在操作系统体系结构、算法开发、编程语言设计、软件工程、数据表示和数据库系统等部分反复进行讨论。

给学生

在美国海军服役期间，我开始进入计算领域，那是在20世纪60年代末70年代初。(的确，服役使我的年龄变大了，但是年长使人变得明智，从而有可能写出一本好的教材)。我服役的大部分时间是维护位于英国伦敦的海军计算机设备的系统软件。服役期满后，我回到了学校，1975年获得了数学博士学位。从那时候开始，我一直从事计算机科学和数学的教学工作。

在这期间，计算机科学里许多东西已经改变了，但也有许多东西依然没有变。特别是，计算机依然是那么迷人，它衍生了许多令人敬畏的东西。因特网的发展、人工智能的进步以及信息获取和传播的能力以前所未有的速度提高，这些将影响人们生活的方式。生活在这令人振奋、富于变化的世界里，你们将大有作为，行动吧！

我有一点儿不听规劝（我的一些朋友会说远不只是“一点儿”）。在编写这本教程的过程中，我并不总是听从所收到的各种规劝。特别是，好多人认为，本教程的某些内容对于初学的大学生来说过于高深。但是，我相信，如果一个论题对于本教程是切题的，那么它就是合适的，即使这个论题被学院派认为是“高级论题”。人们应该得到一本全面而生动地介绍计算机科学的教科书，而不应该是个缩了水的版本——只包括那些被认为初学的大学生可以接受的主题，而且只做简化的介绍。因此，我不回避任何论题，而且尝试做出更好的讲解。我试图给出一幅具有足够深度的关于计算机科学的全面真实写照。如同给食谱加入调料，你们可以有选择地跳过某些专题，但是，将它们放在那儿是供你需要时品味的——我也鼓励你们这样做。

最后，我应当指出，在任何一门关于技术的课程中，你们今天学习的细节可能并不是你们明天想要知道的细节。这个领域是不断发展变化的，这也是它充满激情的地方。本教程将给你们一幅本学科当前情况的写照以及它的历史背景。具有这些基础性知识，你们就能够伴随技术的进步而成长。我鼓励大家通过探索超越本教程所谈到的知识，开始成长的过程。学习，再学习。

感谢你们对我的信任，选择了这本教材，作为一个作者，我有责任写出一本值得你们花费时间去阅读的作品。我希望你们会发现，我尽到了这个责任。

给教师

本教程所包含的内容比一个学期能够讲授的要多，所以当你们想略掉那些不符合你的课程的那些主题时，或者当你们想选择合适的主题重新编排顺序讲授时，请不要犹豫。我写这本书是把它作为一种课程资源，而不是作为课程的定义。你们会发现，尽管本教程是遵循一

条线索撰写的，但是有关的主题是以独立的方式讲述的，所以你们可以按照你们所希望的那样选择其中的内容。

在每章开始的目录中，我使用星号（*）来标识那些我建议的作为可选的章节。按我的看法，这些章节所探究的内容程度比较深，或者转到了你们不想涉及的方向。但这仅仅是建议，你们可以有许多选择。如果我把你们所喜欢的部分标为可选，那么你们也不必因此而烦恼。你们可以认为这是我的错误，你们尽可选择所需要的内容。

我还建议你们把某些题目的内容作为阅读材料，鼓励学生自己学习课堂上没有讲过的内容。我想，如果认为需要在课堂上讲清楚每一个问题，那么这就低估学生了。我们应该帮助他们学会如何自主地学习。

我前面已经说过，本教程遵循一种自底向上、从具体到抽象的结构，对此我还想做进一步的说明。作为学者，我们往往认为学生们会欣赏我们对某一学科的观点，这是我们多年来在特定领域里工作所积累的。作为老师，我认为最好从学生的观点来提供教材。这就是本教程为什么从数据的表示和存储、计算机体系结构和机器语言开始的道理，因为这些正是学生们极其关心的问题——他们能够看到计算机部件，并且能够拿一拿它们，而且大多数学生将会购买并使用计算机。从这些主题开始这门课程的讲授，我发现学生们自己找到了那些已经困惑他们多年的“为什么”问题的答案，并且学会将本课程看成是一门实践课程而不是理论课程。从这出发，就会很自然转到诸如算法的开发、设计、表示和复杂性等比较抽象的主题（那些被该领域的人认为是这一课程主要论题）上来。

我认为，学生们潜移默化学到的东西要比教师课堂上教给他们的多得多，而且他们对于潜移默化学到的知识更容易吸收。在“讲授”问题求解时，这一点尤为重要。我认为，学生们要在通过解决问题的活动中学会解决问题，而不是依靠学习问题求解来获得这种本领。所以，我在本教程中提供了大量的问题。我鼓励你们使用这些问题，并拓展它们。

我在这里要指出的另一个问题是职业道德、伦理和社会责任感。我不认为这些问题可以作为孤立的主题来讲解。相反，当涉及到这些问题时就应该面对，这是我在本教程中的处理方法。你们会发现，0.6、3.5、4.5、7.1、7.8、9.7 和 10.7 节提到了在操作系统、组网、数据库系统、软件工程和人工智能等领域里的安全、隐私、责任和社会意识问题。你们还将发现每一章都包含了一个称为“社会议题”的问题集合，这些问题向学生提出了一种挑战，让他们去思考本教程中的教材与他们生活的社会之间的关系。

感谢你们把本教程作为教科书。不管你们怎样判断本教程是否适合你们的情况，我还是希望你们会发现，本书是对计算机科学教学文献的一个贡献。

教学特点

本教程是多年教学的结晶。因此，它提供了丰富的教学辅助手段。最主要的是提供了许许多多的问题以加强学生们的参与——在第 8 版里包括了 1000 多个问题（确切地说是 1007

个)。它们分为“问题与练习(Questions/Exercises)”、“本章复习题(Chapter Review Problems)”和“社会议题(Social Issues)”三类。“问题与练习”在每节末尾，用于回顾刚刚讨论过的内容，扩展前面的讨论，或者暗示以后将会涉及的有关主题。这些问题的答案在附录 F 中。

“本章复习题”在每章的末尾(除第 0 章外)，它们被设计成“课外作业”形式，因为它们涉及整章的内容。这些问题本教程没有给出答案。

在每章的末尾还设有“社会议题”类问题，它们是供思考和讨论用的。许多问题可以用来开展课外研究，可采用简短的书面报告或口头报告的形式来描述研究的结果。

在每章的末尾还附有一个“课外阅读(Additional Reading)”，它列出了与本章内容有关的其他参考资料。

关于第 8 版

第 8 版相对以前各版本而言有几个重大的修改。以前版本有关操作系统和组网的那一章现在分为两章——第 3 章与第 4 章。在第 4 章中，扩充了组网的内容以正确反映网络发展的状况。“文件结构”那一章被取消了，有关的内容被分散到其他有关的各章中。(如果寻找有关顺序文件、索引文件和散列文件基本知识的资料，那么它们已经精简了，作为第 9 章中的可选节。)

第 8 版中其他的改变有：第 0 章的许多地方重写了。第 1 章有些内容作了重新安排，以加强二进制系统的分量。“程序设计语言”那一章(现在第 6 章)已经精简了。第 7 章关于设计模式的小节重写了，以包括组件系结构。关于“数据结构”一章的重点(和标题)已变成“数据抽象”(第 8 章)。第 9 章“数据库系统”中加了“数据挖掘”一节。第 10 章开始部分按照代理的观点重新写了，而“机器人”小节更新了。第 11 章讨论公开密钥加密方法时，现在是用 RSA 算法，而不再是背包问题。当然，整本教程处处都有许多细小的修改，以提供清晰的、更新的主题及其内容。

我很高兴编写第 8 版，希望大家也能够喜欢它。

Acknowledgments

First, I thank those of you who have supported this book by reading it and using it in previous editions. I am honored.

With each new edition, the list of those who have contributed to the book as reviewers and consultants grows. Today this list includes J. M. Adams, C. M. Allen, D. C. S. Allison, B. Auernheimer, P. Bankston, M. Barnard, P. Bender, K. Bowyer, P. W. Brashear, C. M. Brown, B. Calloni, M. Clancy, R. T. Close, C. L. Cocking, D. H. Cooley, L. D. Cornell, M. J. Crowley, F. Deek, M. Dickerson, M. J. Duncan, R. Eastman, S. Fox, N. E. Gibbs, D. E. Hamilton, J. D. Harris, D. Hascom, L. Heath, P. B. Henderson, L. Hunt, M. Hutchenreuther, L. A. Jehn, E. L. Kisling, K. Korb, G. Krenz, J. Liu, T. J. Long, C. May, W. McCown, S. J. Merrill, K. Messersmith, J. C. Moyer, M. Murphy, J. P. Myers, Jr., D. S. Noonan, S. Olariu, G. Rice, N. Rickert, C. Riedesel, J. B. Rogers, G. Saito, W. Savitch, R. Schlafly, J. C. Schlimmer, D. R. Scott, S. Sells, G. Sheppard, Z. Shen, J. C. Simms, M. C. Slattery, J. Slimick, J. A. Slomka, D. Smith, J. Solderitsch, R. Steigerwald, L. Steinberg, C. A. Struble, W. J. Taffe, J. Talburt, P. Tonellato, P. Tromovitch, E. D. Winter, E. Wright, M. Ziegler, and one anonymous contributor. To these individuals I give my sincere thanks.

A special thank you goes to Roger Eastman, who is responsible for the student activities/projects you will find at the book's companion web site. Roger has demonstrated a lot of creativity in developing these materials. I think you will like the results.

I also thank the people at Addison-Wesley who have contributed to this project. They have done a great job. They have pushed me when I needed pushing, criticized me when I needed criticizing, and praised me on those rare occasions when I was right, and they felt it was safe to tell me. They are a great bunch to work with—and good friends as well.

I am also grateful to my wife Earlene and daughter Cheryl who have been tremendous sources of encouragement over the years. Cheryl, of course, grew up and left home several years ago. But Earlene is still stuck with me and says she plans to stick it out—even if I continue to go through these crazy periods when I write books. I'm thankful that she looks out for me. On the morning of December 11, 1998, I survived a heart attack because she got me to the hospital in time. (For those of you in the younger generation I should explain that surviving a heart attack is sort of like getting an extension on a homework assignment.)

Finally, I thank my parents, to whom this book is dedicated. I close with the following endorsement whose source shall remain anonymous: "Our son's book is really good. Everyone should read it."

J. G. B.

CONTENTS

Chapter 0 Introduction 1

0.1 The Role of Algorithms 2
0.2 The Origins of Computing Machines 4
0.3 The Science of Algorithms 9
0.4 Abstraction 9
0.5 An Outline of Our Study 11
0.6 Social Repercussions 12
Social Issues 13
Additional Reading 15

Chapter 1 Data Storage 17

1.1 Bits and Their Storage 18
1.2 Main Memory 26
1.3 Mass Storage 28
1.4 Representing Information as Bit Patterns 34
1.5 The Binary System 41
1.6 Storing Integers 46
1.7 Storing Fractions 53
1.8 Data Compression 57
1.9 Communication Errors 61
Chapter Review Problems 65
Social Issues 70
Additional Reading 71

Chapter 2 Data Manipulation 73

2.1 Computer Architecture 74
2.2 Machine Language 77
2.3 Program Execution 83
2.4 Arithmetic/Logic Instructions 90
2.5 Communicating with Other Devices 94
2.6 Other Architectures 99
Chapter Review Problems 102
Social Issues 107
Additional Reading 108

Chapter 3 Operating Systems 109

3.1 The Evolution of Operating Systems 110
3.2 Operating System Architecture 113
3.3 Coordinating the Machine's Activities 120
3.4 Handling Competition Among Processes 123
3.5 Security 127
Chapter Review Problems 129
Social Issues 132
Additional Reading 133

Chapter 4 Networking and the Internet 135
4.1 Network Fundamentals 136
4.2 The Internet 141
4.3 The World Wide Web 147
4.4 Network Protocols 156
4.5 Security 164
Chapter Review Problems 167
Social Issues 169
Additional Reading 170

Chapter 5 Algorithms 171
5.1 The Concept of an Algorithm 172
5.2 Algorithm Representation 175
5.3 Algorithm Discovery 182
5.4 Iterative Structures 188
5.5 Recursive Structures 199
5.6 Efficiency and Correctness 207
Chapter Review Problems 216
Social Issues 222
Additional Reading 223

Chapter 6 Programming Languages 225
6.1 Historical Perspective 226
6.2 Traditional Programming Concepts 235
6.3 Procedural Units 246
6.4 Language Implementation 254
6.5 Object-Oriented Programming 263
6.6 Programming Concurrent Activities 269
6.7 Declarative Programming 272
Chapter Review Problems 278
Social Issues 282
Additional Reading 283

Chapter 7 Software Engineering 285
7.1 The Software Engineering Discipline 286
7.2 The Software Life Cycle 288
7.3 Modularity 293
7.4 Design Methodologies 299
7.5 Tools of the Trade 303
7.6 Testing 308
7.7 Documentation 309
7.8 Software Ownership and Liability 311
Chapter Review Problems 313
Social Issues 316
Additional Reading 317

Chapter 8 Data Abstractions 319

8.1 Data Structure Fundamentals 320
8.2 Implementing Data Structures 324
8.3 A Short Case Study 337
8.4 Customized Data Types 342
8.5 Classes and Objects 346
8.6 Pointers in Machine Language 348
Chapter Review Problems 350
Social Issues 355
Additional Reading 357

Chapter 9 Database Systems 359

9.1 Database Fundamentals 360
9.2 The Relational Model 364
9.3 Object-Oriented Databases 376
9.4 Maintaining Database Integrity 379
9.5 Traditional File Structures 382
9.6 Data Mining 391
9.7 Social Impact of Database Technology 393
Chapter Review Problems 395
Social Issues 400
Additional Reading 401

Chapter 10 Artificial Intelligence 403

10.1 Intelligence and Machines 404
10.2 Understanding Images 408
10.3 Reasoning 411
10.4 Artificial Neural Networks 423
10.5 Genetic Algorithms 434
10.6 Other Areas of Research 438
10.7 Considering the Consequences 446
Chapter Review Problems 448
Social Issues 453
Additional Reading 455

Chapter 11 Theory of Computation 457

11.1 Functions and Their Computation 458
11.2 Turing Machines 460
11.3 Universal Programming Languages 464
11.4 A Noncomputable Function 470
11.5 Complexity of Problems 476
11.6 Public Key Cryptography 485
Chapter Review Problems 489
Social Issues 493
Additional Reading 494

Appendixes 495

A ASCII 495
B Circuits to Manipulate Two's Complement Representations 497
C A Simple Machine Language 501
D High-Level Language Program Examples 503
E The Equivalence of Iterative and Recursive Structures 511
F Answers to Questions/Exercises 513

Index 550

CHAPTER 0

INTRODUCTION

0.1 The Role of Algorithms

0.2 The Origins of Computing Machines

0.3 The Science of Algorithms

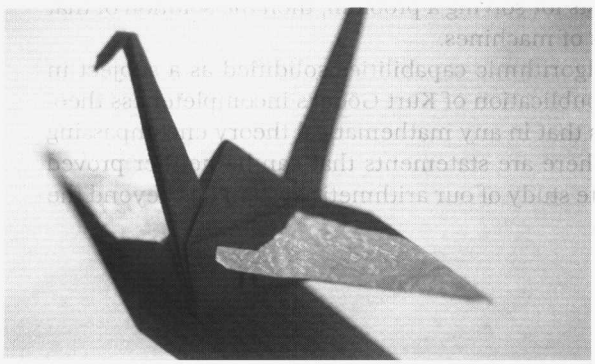
0.4 Abstraction

0.5 An Outline of Our Study

0.6 Social Repercussions

Computer science is the discipline that seeks to build a scientific foundation for such topics as computer design, computer programming, information processing, algorithmic solutions of problems, and the algorithmic process itself. It provides the underpinnings for today's computer applications as well as the foundations for tomorrow's applications. This breadth means that we cannot become knowledgeable in computer science by studying only a few topics as isolated subjects or by merely learning how to use the computing tools of today. Rather, to understand the science of computing, we must grasp the scope and dynamics of a wide range of topics.

This book is designed to provide such a background. It presents a comprehensive introduction to the subjects that constitute a typical university computer science curriculum. The book can therefore serve as a foundation for beginning computer science students or as a source for other students seeking an introduction to the science behind today's computer-oriented society.



0.1 The Role of Algorithms

We begin with the most fundamental concept of computer science—that of an algorithm. Informally, an **algorithm** is a set of steps that defines how a task is performed. (We will be more precise later in Chapter 5.) For example, there are algorithms for cooking (called recipes), for finding your way through a strange city (more commonly called directions), for operating washing machines (usually displayed on the inside of the washer's lid or perhaps on the wall of a laundromat), for playing music (expressed in the form of sheet music), and for performing magic tricks (Figure 0.1).

Before a machine such as a computer can perform a task, an algorithm for performing that task must be discovered and represented in a form that is compatible with the machine. A representation of an algorithm is called a **program.** For the convenience of humans, computer programs are usually printed on paper or displayed on computer screens. For the convenience of machines, programs are encoded in a manner compatible with the technology of the machine. The process of developing a program, encoding it in machine compatible form, and inserting it into a machine is called **programming.** Programs, and the algorithms they represent, are collectively referred to as **software,** in contrast to the machinery itself, which is known as **hardware.**

The study of algorithms began as a subject in mathematics. Indeed, the search for algorithms was a significant activity of mathematicians long before the development of today's computers. The goal was to find a single set of directions that described how all problems of a particular type could be solved. One of the best known examples of this early research is the long division algorithm for finding the quotient of two multiple-digit numbers. Another example is the Euclidean algorithm, discovered by the ancient Greek mathematician Euclid, for finding the greatest common divisor of two positive integers (Figure 0.2).

Once an algorithm for performing a task has been found, the performance of that task no longer requires an understanding of the principles on which the algorithm is based. Instead, the performance of the task is reduced to the process of merely following directions. (We can follow the long division algorithm to find a quotient or the Euclidean algorithm to find a greatest common divisor without understanding why the algorithm works.) In a sense, the intelligence required to solve the problem at hand is encoded in the algorithm.

It is through this ability to capture and convey intelligence (or at least intelligent behavior) by means of algorithms that we are able to build machines that perform useful tasks. Consequently, the level of intelligence displayed by machines is limited by the intelligence that can be conveyed through algorithms. Only if we find an algorithm that directs the performance of a task can we construct a machine to perform that task. In turn, if no algorithm exists for solving a problem, then the solution of that problem lies beyond the capabilities of machines.

Identifying the limitations of algorithmic capabilities solidified as a subject in mathematics in the 1930s with the publication of Kurt Gödel's incompleteness theorem. This theorem essentially states that in any mathematical theory encompassing our traditional arithmetic system, there are statements that can be neither proved nor disproved. In short, any complete study of our arithmetic system lies beyond the capabilities of algorithmic activities.