COMPUTER SCIENCE
ILLUMINATED

SIXTH EDITION

NELL DALE
The University of Texas at Austin

JOHN LEWIS
Virginia Tech

JONES & BARTLETT
LEARNING

World Headquarters
Jones & Bartlett Learning
5 Wall Street
Burlington, MA 01803
978-443-5000
info@jblearning.com
www.jblearning.com

Jones & Bartlett Learning books and products are available through most bookstores and online booksellers. To contact Jones & Bartlett Learning directly, call 800-832-0034, fax 978-443-8000, or visit our website, www.jblearning.com.

Substantial discounts on bulk quantities of Jones & Bartlett Learning publications are available to corporations, professional associations, and other qualified organizations. For details and specific discount information, contact the special sales department at Jones & Bartlett Learning via the above contact information or send an email to specialsales@jblearning.com.

06951-8

Production Credits
Publisher: Cathy L. Esperti
Acquisitions Editor: Laura Pagluica
Editorial Assistant: Taylor Ferracane
Director of Production: Amy Rose
Associate Production Editor: Sara Kelly
Associate Marketing Manager: Cassandra Peterson
Art Development Editor: Joanna Lundeen
Art Development Assistant: Shannon Sheehan
VP, Manufacturing and Inventory Control: Therese Connell
Composition: Cenveo Publisher Services
Cover Design: Kristin E. Parker
Rights and Photo Research Coordinator: Amy Rathburn
Cover Image: © Sergey Nivens/Shutterstock, Inc.
Printing and Binding: Courier Companies
Cover Printing: Courier Companies

Library of Congress Cataloging-in-Publication Data
Dale, Nell.
Computer science illuminated / Nell Dale, PhD, University of Texas-Austin, Department of Computer Science, John A. Lewis, Virginia Tech. — Sixth edition.
pages cm
Includes bibliographical references and index.
ISBN 978-1-284-05591-7 (pbk.) 1. Computer science. I. Lewis, John, 1963- II. Title.
QA76.D285 2015
004—dc23

2014032093

6048

Printed in the United States of America
19 18 17 16 15 10 9 8 7 6 5 4 3 2 1

Onion: © matka_Wariatka/ShutterStock, Inc.; Nebula: © Sergey Nivens/Shutterstock, Inc.

To my wife, Sharon, and our children, Justin, Kayla, Nathan, and Samantha.

—John Lewis

To all the students who will use this book: It is written for you.

—Nell Dale

John Lewis, Virginia Tech

John Lewis is a leading educator and author in the field of computer science. He has written a market-leading textbook on Java software and program design. After earning his PhD in Computer Science, John spent 14 years at Villanova University in Pennsylvania. He now teaches computing at Virginia Tech, his alma mater, and works on textbook projects out of his home. He has received numerous teaching awards, including the University Award for Teaching Excellence and the Goff Award for Outstanding Teaching. His professional interests include object-oriented technologies, multimedia, and software engineering. In addition to teaching and writing, John actively participates in the ACM Special Interest Group on Computer Science Education (SIGCSE) and finds time to spend with his family and in his workshop.

Nell Dale, The University of Texas at Austin

Well-respected in the field of computer science education, Nell Dale has served on the faculty of The University of Texas at Austin, for more than 25 years and has authored over 40 undergraduate Computer Science textbooks. After receiving her BS in Mathematics and Psychology from the University of Houston, Nell entered The University of Texas at Austin, where she earned her MA in Mathematics and her PhD in Computer Science. Nell has made significant contributions to her discipline through her writing, research, and service. Nell's contributions were recognized in 1996 with the ACM SIGCSE Award for Outstanding Contributions in Computer Science Education and in 2001 with the ACM Karl V. Karlstrom Outstanding Educator Award. She was elected an ACM Fellow in 2010. In 2013, she received the IEEE Taylor L. Booth Education Award. Nell has retired from full-time teaching, giving her more time to write, travel, and play tennis and bridge. She currently resides in Austin, Texas.

BRIEF CONTENTS

1 **Laying the Groundwork 2**

Chapter 1 The Big Picture 3

2 **The Information Layer 34**

Chapter 2 Binary Values and Number Systems 35
Chapter 3 Data Representation 55

3 **The Hardware Layer 92**

Chapter 4 Gates and Circuits 93
Chapter 5 Computing Components 121

4 **The Programming Layer 152**

Chapter 6 Low-Level Programming Languages and Pseudocode 153
Chapter 7 Problem Solving and Algorithms 197
Chapter 8 Abstract Data Types and Subprograms 247
Chapter 9 Object-Oriented Design and High-Level Programming Languages 287

5 **The Operating Systems Layer 336**

Chapter 10 Operating Systems 337
Chapter 11 File Systems and Directories 369

6 **The Applications Layer 394**

Chapter 12 Information Systems 395
Chapter 13 Artificial Intelligence 425
Chapter 14 Simulation, Graphics, Gaming, and Other Applications 459

7 **The Communications Layer. 500**

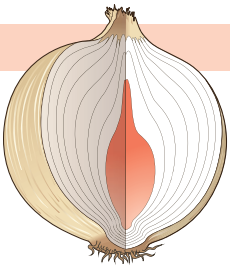
Chapter 15 Networks 501
Chapter 16 The World Wide Web 531
Chapter 17 Computer Security 563

8 **In Conclusion 592**

Chapter 18 Limitations of Computing 593

CONTENTS

Onion: © matka_Wariatka/ShutterStock, Inc.



1 Laying the Groundwork 2

Chapter 1 The Big Picture 3

1.1 Computing Systems 4

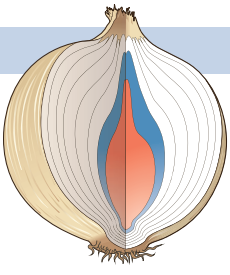
Layers of a Computing System 4
Abstraction 6

1.2 The History of Computing 9

A Brief History of Computing Hardware 9
A Brief History of Computing Software 19
Predictions 25

1.3 Computing as a Tool and a Discipline 26

Summary 29
Ethical Issues: Digital Divide 30
Key Terms 31
Exercises 31
Thought Questions 33



2 The Information Layer 34

Chapter 2 Binary Values and Number Systems 35

2.1 Numbers and Computing 36

2.2 Positional Notation 36

Binary, Octal, and Hexadecimal 38
Arithmetic in Other Bases 41
Power-of-2 Number Systems 42
Converting from Base 10 to Other Bases 44
Binary Values and Computers 45

Summary 48
Ethical Issues: The FISA Court 49
Key Terms 49
Exercises 50
Thought Questions 53

Chapter 3 Data Representation 55

3.1 Data and Computers 56

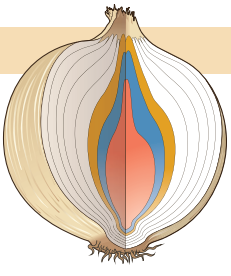
Analog and Digital Data 57
Binary Representations 59

Onion: © matka_Wariatka/ShutterStock, Inc.

3.2 Representing Numeric Data 61
Representing Negative Values 61
Representing Real Numbers 65
3.3 Representing Text 68
The ASCII Character Set 69
The Unicode Character Set 70
Text Compression 71
3.4 Representing Audio Data 76
Audio Formats 78
The MP3 Audio Format 78
3.5 Representing Images and Graphics 79
Representing Color 79
Digitized Images and Graphics 81
Vector Representation of Graphics 82
3.6 Representing Video 83
Video Codecs 83
Summary 85
Ethical Issues: The Fallout from Snowden's Revelations 86
Key Terms 86
Exercises 87
Thought Questions 91

3 The Hardware Layer 92

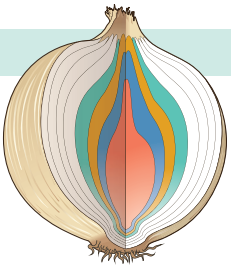
Chapter 4 Gates and Circuits 93
4.1 Computers and Electricity 94
4.2 Gates 96
NOT Gate 96
AND Gate 97
OR Gate 98
XOR Gate 98
NAND and NOR Gates 99
Review of Gate Processing 100
Gates with More Inputs 101



4.3 Constructing Gates 101
Transistors 102
4.4 Circuits 104
Combinational Circuits 104
Adders 108
Multiplexers 110
4.5 Circuits as Memory 111
4.6 Integrated Circuits 112
4.7 CPU Chips 113
Summary 113
Ethical Issues: Codes of Ethics 114
Key Terms 116
Exercises 116
Thought Questions 119

Chapter 5 Computing Components 121
5.1 Individual Computer Components 122
5.2 The Stored-Program Concept 127
von Neumann Architecture 129
The Fetch–Execute Cycle 133
RAM and ROM 135
Secondary Storage Devices 136
Touch Screens 141
5.3 Embedded Systems 143
5.4 Parallel Architectures 144
Parallel Computing 144
Classes of Parallel Hardware 146
Summary 147
Ethical Issues: Is Privacy a Thing of the Past? 148
Key Terms 148
Exercises 149
Thought Questions 151

Onion: © matka_Wariatka/ShutterStock, Inc.



4 The Programming Layer 152

Chapter 6 Low-Level Programming Languages and Pseudocode 153

6.1 Computer Operations 154

6.2 Machine Language 154

Pep/8: A Virtual Computer 155

6.3 A Program Example 162

Hand Simulation 163
Pep/8 Simulator 164

6.4 Assembly Language 166

Pep/8 Assembly Language 167
Assembler Directives 168
Assembly-Language Version of Program Hello 168
A New Program 169
A Program with Branching 171
A Program with a Loop 174

6.5 Expressing Algorithms 176

Pseudocode Functionality 176
Following a Pseudocode Algorithm 180
Writing a Pseudocode Algorithm 182
Translating a Pseudocode Algorithm 185

6.6 Testing 188

Summary 189
Ethical Issues: Software Piracy 191
Key Terms 192
Exercises 192
Thought Questions 195

Chapter 7 Problem Solving and Algorithms 197

7.1 How to Solve Problems 198

Ask Questions 199
Look for Familiar Things 199
Divide and Conquer 200

Algorithms 200
Computer Problem-Solving Process 202
Summary of Methodology 203
Testing the Algorithm 204

7.2 Algorithms with Simple Variables 205

An Algorithm with Selection 205
Algorithms with Repetition 206

7.3 Composite Variables 212

Arrays 212
Records 213

7.4 Searching Algorithms 214

Sequential Search 214
Sequential Search in a Sorted Array 215
Binary Search 218

7.5 Sorting 220

Selection Sort 221
Bubble Sort 224
Insertion Sort 226

7.6 Recursive Algorithms 227

Subprogram Statements 227
Recursive Factorial 229
Recursive Binary Search 230
Quicksort 230

7.7 Important Threads 234

Information Hiding 234
Abstraction 235
Naming Things 236
Testing 237

Summary 237
Ethical Issues: Open-Source Software 238
Key Terms 240
Exercises 240
Thought Questions 245

Chapter 8 Abstract Data Types and Subprograms 247

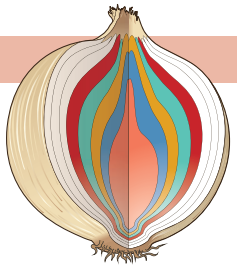
8.1 What Is an Abstract Data Type? 248

8.2 Stacks 248

8.3 Queues 249
8.4 Lists 250
8.5 Trees 253
Binary Trees 253
Binary Search Trees 256
Other Operations 261
8.6 Graphs 262
Creating a Graph 264
Graph Algorithms 265
8.7 Subprograms 271
Parameter Passing 272
Value and Reference Parameters 274
Summary 278
Ethical Issues: Workplace Monitoring 279
Key Terms 280
Exercises 280
Thought Questions 285

Chapter 9 Object-Oriented Design and High-Level Programming Languages 287
9.1 Object-Oriented Methodology 288
Object Orientation 288
Design Methodology 289
Example 292
9.2 Translation Process 297
Compilers 298
Interpreters 298
9.3 Programming Language Paradigms 301
Imperative Paradigm 301
Declarative Paradigm 302
9.4 Functionality in High-Level Languages 304
Boolean Expressions 305
Data Typing 307
Input/Output Structures 311
Control Structures 313

9.5 Functionality of Object-Oriented Languages 320
Encapsulation 320
Classes 321
Inheritance 323
Polymorphism 324
9.6 Comparison of Procedural and Object-Oriented Designs 325
Summary 326
Ethical Issues: Hoaxes and Scams 328
Key Terms 329
Exercises 330
Thought Questions 335



5 The Operating Systems Layer 336
Chapter 10 Operating Systems 337
10.1 Roles of an Operating System 338
Memory, Process, and CPU Management 340
Batch Processing 341
Timesharing 342
Other OS Factors 343
10.2 Memory Management 344
Single Contiguous Memory Management 346
Partition Memory Management 347
Paged Memory Management 349
10.3 Process Management 352
The Process States 352
The Process Control Block 353
10.4 CPU Scheduling 354
First Come, First Served 355
Shortest Job Next 356
Round Robin 356
Summary 358
Ethical Issues: Medical Privacy: HIPAA 360
Key Terms 361
Exercises 362
Thought Questions 367

Onion: © matka_Wariatka/ShutterStock, Inc.

Chapter 11 File Systems and Directories 369

11.1 File Systems 370

Text and Binary Files 370
File Types 371
File Operations 373
File Access 374
File Protection 375

11.2 Directories 376

Directory Trees 377
Path Names 379

11.3 Disk Scheduling 381

First-Come, First-Served Disk Scheduling 383
Shortest-Seek-Time-First Disk Scheduling 383
SCAN Disk Scheduling 384

Summary 386
Ethical Issues: Privacy: Opt-In or Opt-Out? 388
Key Terms 389
Exercises 389
Thought Questions 393

6 The Applications Layer 394

Chapter 12 Information Systems 395

12.1 Managing Information 396

12.2 Spreadsheets 396

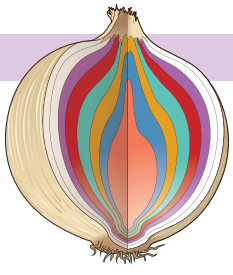
Spreadsheet Formulas 399
Circular References 402
Spreadsheet Analysis 405

12.3 Database Management Systems 406

The Relational Model 407
Relationships 409
Structured Query Language 411
Database Design 413

12.4 E-Commerce 414

Summary 415
Ethical Issues: Politics and the Internet: The Candidate's View 417



Key Terms 418
Exercises 419
Thought Questions 423

Chapter 13 Artificial Intelligence 425

13.1 Thinking Machines 426

The Turing Test 427
Aspects of AI 429

13.2 Knowledge Representation 429

Semantic Networks 431
Search Trees 433

13.3 Expert Systems 436

13.4 Neural Networks 438

Biological Neural Networks 438
Artificial Neural Networks 439

13.5 Natural Language Processing 441

Voice Synthesis 442
Voice Recognition 443
Natural Language Comprehension 444

13.6 Robotics 446

The Sense–Plan–Act Paradigm 446
Subsumption Architecture 448
Physical Components 450

Summary 451
Ethical Issues: Initial Public Offerings 452
Key Terms 453
Exercises 453
Thought Questions 457

Chapter 14 Simulation, Graphics, Gaming, and Other Applications 459

14.1 What Is Simulation? 460

Complex Systems 460
Models 461
Constructing Models 461

14.2 Specific Models 463

Queuing Systems 463
Meteorological Models 466
Computational Biology 472
Other Models 473
Computing Power Necessary 473

14.3 Computer Graphics 474

How Light Works 476
Object Shape Matters 478
Simulating Light 478
Modeling Complex Objects 480
Getting Things to Move 486

14.4 Gaming 488

History of Gaming 489
Creating the Virtual World 490
Game Design and Development 491
Game Programming 492

Summary 493
Ethical Issues: Gaming as an Addiction 495
Key Terms 496
Exercises 496
Thought Questions 499

7 The Communications Layer 500

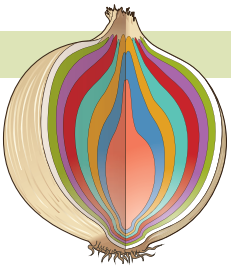
Chapter 15 Networks 501

15.1 Networking 502

Types of Networks 503
Internet Connections 505
Packet Switching 508

15.2 Open Systems and Protocols 510

Open Systems 511
Network Protocols 512
TCP/IP 512
High-Level Protocols 514
MIME Types 515
Firewalls 515



15.3 Network Addresses 516

Domain Name System 518
Who Controls the Internet? 521

15.4 Cloud Computing 521

Summary 523
Ethical Issues: The Effects of Social Networking 525
Key Terms 526
Exercises 527
Thought Questions 529

Chapter 16 The World Wide Web 531

16.1 Spinning the Web 532

Search Engines 533
Instant Messaging 534
Weblogs 535
Cookies 536
Web Analytics 536

16.2 HTML and CSS 537

Basic HTML Elements 541
Tag Attributes 542
More About CSS 543
More HTML5 Elements 546

16.3 Interactive Web Pages 547

Java Applets 547
Java Server Pages 548

16.4 XML 549

16.5 Social Networks 553

Summary 554
Ethical Issues: Gambling and the Internet 557
Key Terms 558
Exercises 558
Thought Questions 561

Chapter 17 Computer Security 563

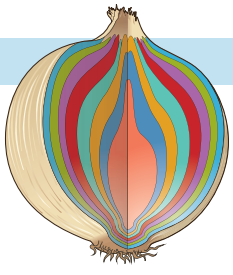
17.1 Security at All Levels 564

Information Security 564

Onion: © matka_Wariatka/ShutterStock, Inc.

17.2 Preventing Unauthorized Access 566
Passwords 567
CAPTCHA 569
Fingerprint Analysis 570
17.3 Malicious Code 571
Antivirus Software 572
Security Attacks 573
17.4 Cryptography 575
17.5 Protecting Your Information Online 578
Security and Portable Devices 580
WikiLeaks 581
Summary 583
Ethical Issues: Blogging 586
Key Terms 587
Exercises 588
Thought Questions 591

8 In Conclusion 592
Chapter 18 Limitations of Computing 593
18.1 Hardware 594
Limits on Arithmetic 594
Limits on Components 600
Limits on Communications 601
18.2 Software 602
Complexity of Software 602
Current Approaches to Software Quality 603
Notorious Software Errors 608
18.3 Problems 610
Comparing Algorithms 611
Turing Machines 618
Halting Problem 621
Classification of Algorithms 623
Summary 625
Ethical Issues: Therac-25: Anatomy of a Disaster 626



Key Terms 627
Exercises 627
Thought Questions 630

Glossary 631
Endnotes 657
Index 667

PREFACE

Choice of Topics

In putting together the outline of topics for this CS0 text, we used many sources. We looked at course catalogue descriptions and book outlines, and we administered a questionnaire designed to find out what you, our colleagues, thought should be included in such a course. We asked you and ourselves to do the following:

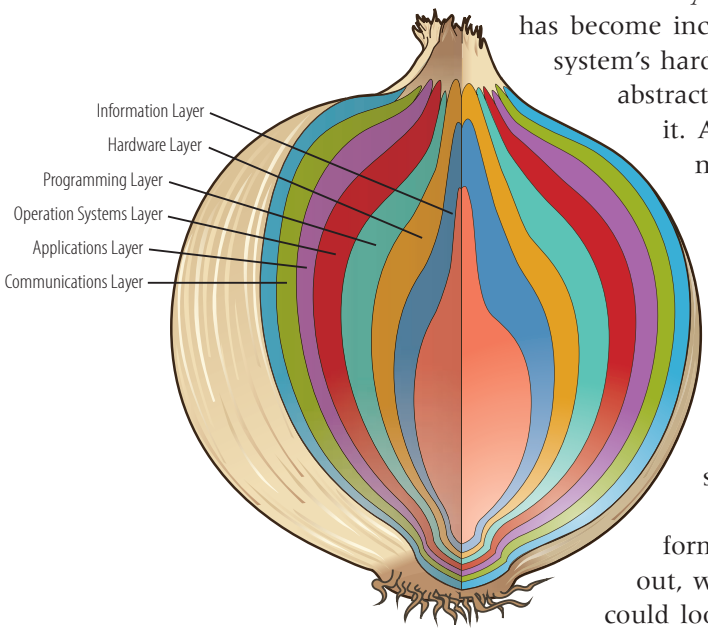
- Please list four topics that you feel students should master in a CS0 course if this is the only computer science course they will take during their college experience.
- Please list four topics that you would like students entering your CS1 course to have mastered.
- Please list four additional topics that you would like your CS1 students to be familiar with.

The strong consensus that emerged from the intersections of these sources formed the working outline for this book. Students who master this material before taking CS1 have a strong foundation upon which to build their knowledge of computer science. Although our intention was to write a CS0 text, our reviewers have pointed out that the material also forms a strong breadth-first background that can also serve as a companion to a programming-language introduction to computer science.

Rationale for Organization

This book begins with the history of hardware and software, showing how a computer system is like an onion. The processor and its machine language form the heart of the onion, and layers of software and more sophisticated hardware have been added around this heart, layer by layer. At the next layer, higher-level languages such as FORTRAN, Lisp, Pascal, C, C++, and Java were introduced parallel to the ever-increasing exploration of the programming process, using such tools as top-down design and object-oriented design. Over time, our understanding of the role of abstract data types and their implementations matured. The operating system, with its resource-management techniques—including files on ever-larger, faster secondary storage media—developed to surround and manage these programs.

The next layer of the computer system "onion" is composed of sophisticated general-purpose and special-purpose software systems that overlay the operating system. Development of these powerful programs was stimulated by theoretical work in computer science, which makes such programs possible. The final layer comprises networks and network software—that is, the tools needed for computers to communicate with one another. The Internet and the World Wide Web put the finishing touches to this layer, and this text culminates with a discussion of security issues affecting our interaction online.



As these layers have grown over the years, the user has become increasingly insulated from the computer system's hardware. Each of these layers provides an abstraction of the computing system beneath it. As each layer has evolved, users of the new layer have joined with users of inner layers to create a very large workforce in the high-tech sector of the global economy. This book is designed to provide an overview of the layers, introducing the underlying hardware and software technologies, in order to give students an appreciation and understanding of all aspects of computing systems.

Having used history to describe the formation of the onion from the inside out, we were faced with a design choice: We could look at each layer in depth from the inside out or the outside in. The outside-in approach was very tempting. We could peel the layers off one at a time, moving from the most abstract layer to the concrete machine. However, research has shown that students understand concrete examples more easily than abstract ones, even when the students themselves are abstract thinkers. Thus, we have chosen to begin with the concrete machine and examine the layers in the order in which they were created, trusting that a thorough understanding of one layer makes the transition to the next abstraction easier for the students.

Onion: © matka_Wariatka/ShutterStock, Inc.

Changes in the Sixth Edition

As always when planning a revision, we asked our colleagues, including many current users of the text, to give us feedback. We appreciate the many thoughtful and insightful responses we received.

Updates in the *Sixth Edition* include a considerable overhaul of Chapters 15 and 16, which are about networks and the World Wide Web. We include new information about wireless networks, as well as updates to the top-level domains (TLDs) that are now available. In light of recent developments in U.S. oversight, we added a discussion about who controls the Internet. Screenshots and discussions of ping and traceroute utilities are now included, as well as an enhanced discussion about mobile computing. We completely rewrote the section on HTML in Chapter 16 to reflect the most up-to-date practices and the use of Cascading Style Sheets (CSS). We updated the section on social networks and added a new discussion of web-based analytics.

In addition to these and other updates, the common features throughout the book have been completely revised and augmented. The "Ethical Issues" sections at the end of each chapter have been brought up to date. The "Did You Know?" sidebars have been updated throughout the book as well, with the addition of several more that reflect new and novel topics. Finally, the biographical sections throughout have been updated.

The *Sixth Edition* features a brand new design and layout, with all figures redrawn and photos updated throughout.

Of course, we also made minor revisions throughout the book to improve and update the coverage, presentation, and examples.

Synopsis

Chapter 1 lays the groundwork, as described in the "Rationale for This Book's Organization" section above. **Chapters 2** and **3** step back and examine a layer that is embodied in the physical hardware. We call this the "information layer" because it reflects how data is represented in the computer. Chapter 2 covers the binary number system and its relationship to other number systems such as decimal (the one we humans use on a daily basis). Chapter 3 investigates how we take the myriad types of

data we manage—numbers, text, images, audio, and video—and represent them in a computer in binary format.

Chapters 4 and **5** discuss the hardware layer. Computer hardware includes devices such as transistors, gates, and circuits, all of which control the flow of electricity in fundamental ways. This core electronic circuitry gives rise to specialized hardware components such as the computer's central processing unit (CPU) and memory. Chapter 4 covers gates and electronic circuits; Chapter 5 focuses on the hardware components of a computer and how they interact within a von Neumann architecture.

Chapters 6 through **9** examine aspects of the programming layer. Chapter 6 explores the concepts of both machine language and assembly language programming using Pep/8, a simulated computer. We discuss the functionality of pseudocode as a way to write algorithms. The concepts of looping and selection are introduced here, expressed in pseudocode, and implemented in Pep/8.

Chapter 7 examines the problem-solving process as it relates to both humans and computers. George Polya's human problem-solving strategies guide the discussion. Top-down design is presented as a way to design simple algorithms. We choose classic searching and sorting algorithms as the context for the discussion of algorithms. Because algorithms operate on data, we examine ways to structure data so that it can be more efficiently processed. We also introduce subalgorithm (subprogram) statements.

Chapter 8 takes a step further toward abstraction, exploring abstract data types and containers: composite structures for which we know only properties or behaviors. Lists, sorted lists, stacks, queues, binary search trees, and graphs are discussed. The section on subalgorithms is expanded to include reference and value parameters and parameter passing.

Chapter 9 covers the concepts of high-level programming languages. Because many prominent high-level languages include functionality associated with object-oriented programming, we detour and first present this design process. Language paradigms and the compilation process are discussed. Pseudocode concepts are illustrated in brief examples from four programming languages: Python, Visual Basic .NET, C++, and Java.

Chapters 10 and **11** cover the operating system layer. Chapter 10 discusses the resource management responsibilities of the operating sys-

tem and presents some of the basic algorithms used to implement these tasks. Chapter 11 focuses on file systems, including what they are and how they are managed by the operating system.

Chapters 12 through **14** cover the application layer. This layer is made up of the general-purpose and specialized application programs that are available to the public for solving programs. We divide this layer into the sub-disciplines of computer science upon which these programs are based. Chapter 12 examines information systems, Chapter 13 examines artificial intelligence, and Chapter 14 examines simulation, graphics, gaming, and other applications.

Chapters 15 through **17** cover the communication layer. Chapter 15 presents the theoretical and practical aspects of computers communicating with each other. Chapter 16 discusses the World Wide Web and the various technologies involved. Chapter 17 examines computer security and keeping information protected in the modern information age.

Chapters 2 through 17 are about what a computer can do and how. **Chapter 18** concludes the text with a discussion of the inherent limitations of computer hardware and software, including the problems that can and cannot be solved using a computer. We present Big-O notation as a way to talk about the efficiency of algorithms so that the categories of algorithms can be discussed, and we use the Halting problem to show that some problems are unsolvable.

The first and last chapters form bookends: Chapter 1 describes what a computing system is and Chapter 18 cautions about what computing systems cannot do. The chapters between take an in-depth look at the layers that make up a computing system.

Why Not a Language?

The original outline for this book included an "Introduction to Java" chapter. Some of our reviewers were ambivalent about including a language at all; others wondered why Java would be included and not C++. We decided to leave the choice to the user. Introductory chapters, formatted in a manner consistent with the design of this book, are available for

Onion: © matka_Wariatka/ShutterStock, Inc.

128

Chapter 5: Computing Components

John Vincent Atanasoff

John Vincent Atanasoff was born in Hamilton, New York, on October 4, 1903, one of nine children. When he was about ten, his father bought a new slide rule. After reading the instructions, John Vincent became more interested in the mathematics involved than in the slide rule itself. His mother picked up on his interest and helped him study his father's old college algebra book. He continued his interest in mathematics and science and graduated from high school in two years. His family moved to Old Chicara, Florida, where John Vincent graduated from the University of Florida in 1925 with a degree in electrical engineering because the university didn't offer a degree in theoretical physics. A year later, he received a master's degree in mathematics from Iowa State College. In 1930, after receiving his PhD in theoretical physics, he returned to Iowa State College as an assistant professor in mathematics and physics.

Dr. Atanasoff became interested in finding a machine that could do the complex mathematical work he and his graduate students were doing. He examined computational devices in existence at that time, including the Monroe [illegible]ator and the IBM tabula[illegible] machin[illegible]

In 1939, with a $650 grant from the school and a new graduate assistant named Clifford Berry, Dr. Atanasoff began work on the first prototype of the Atanasoff-Berry Computer (ABC) in the basement of the physics building. The first working prototype was demonstrated that year.

In 1941, John Mauchly, a physicist at Ursinus College whom Dr. Atanasoff had met at a conference, came to Iowa State to visit the Atanasoffs and see a demonstration of the ABC machine. After extensive discussions, Mauchly left with papers describing its design. Mauchly and J. Presper Eckert continued their work on a computation device at the Moore School of Electrical Engineering at the University of Pennsylvania. Their machine, the ENIAC, completed in 1945, became known as the first computer.

Dr. Atanasoff went to Washington in 1942 to become director of the Underwater Acoustics Program at the Naval Ordnance Laboratory, leaving the patent application for the ABC computer in the hands of the Iowa State attorneys. The patent application was [illegible]and the ABC

Courtesy of ISU Photo Service

Java, C++, JavaScript, Visual Basic. NET, Python, SQL, Ruby, Perl, Alice, and Pascal on the book's website and in hard copy through Jones & Bartlett Learning.

If the students have enough knowledge and experience to master the introductory syntax and semantics of a language in addition to the background material in this book, simply have the students download the appropriate chapter. As an alternative, one or all of these chapters can be used to enrich the studies of those who have stronger backgrounds.

Special Features

We have included three special features in this text in order to emphasize the history and breadth of computing as well as the moral obligations that come with new technology.

Biographies

Each chapter includes a short biography of someone who has made a significant contribution to computing as we know it. The people honored in these sections range from those who contributed to the data layer, such as George Boole and Ada Lovelace, to those who have contributed to the communication layer, such as Doug Engelbart and Tim Berners-Lee. These biographies give students a taste of history and introduce them to the men and women who are pioneers in the world of computing.

Did You Know

Our second feature (the "Did You Know?" sections indicated by a question mark) comprises sidebars that include interesting tidbits of information from the past, present, and future. They are garnered from history, current

?

Virtual games and national security

U.S. and British spies have infiltrated the fantasy world of virtual games. A 2008 National Security Agency (NSA) document declared that virtual games provide a "target-rich communication network" that allows intelligence suspects a way to communicate and "hide in plain sight."[4]

Onion: © matka_Wariatka/ShutterStock, Inc.

events, and the authors' personal experiences. These little vignettes are designed to amuse, inspire, intrigue, and, of course, educate.

ETHICAL ISSUES

The FISA Court

The United States Foreign Intelligence Surveillance Court is a U.S. federal court that was established under the Foreign Intelligence Surveillance Act of 1978 (FISA). The Court handles requests by federal law enforcement agencies for surveillance warrants against suspected foreign intelligence agents operating inside the United States.[4]

Before 2013, when Edward Snowden leaked that the Court had ordered a subsidiary of Verizon to provide detailed call records to the National Security Agency (NSA), most people had never heard of the FISA Court. The next chapter examines the controversy surrounding it.

The FISA Court comprises 11 judges who sit for 7-year terms. The Chief Justice of the Supreme Court appoints the judges, without confirmation. An application for an electronic surveillance warrant is made before

application is denied, the government may not take the same request to another judge. If the U.S. Attorney General determines that an emergency exists, he or she may authorize the electronic surveillance but must notify a Court judge not more than 72 hours after the authorization. The USA PATRIOT Act of 2001 expanded the time periods during which surveillance may be authorized.[5]

In December 2012, President Obama signed the FISA Amendments Act Reauthorization Act of 2012, which extends Title VII of FISA until December 31, 2017.

Title VII of FISA, added by the FISA Amendments Act of 2008, created separate procedures for targeting suspected foreign intelligence agents, including non-U.S. persons and U.S. persons reasonably believed to be outside the United States.[6]

Note that the stated intent of the FISA

Ethical Issues

Our third feature is an "Ethical Issues" section that is included in each chapter. These sections illustrate the fact that along with the advantages of computing come responsibilities for and consequences of its use. Privacy, hacking, viruses, and free speech are among the topics discussed. Following the exercises in each chapter, a "Thought Questions" section asks stimulating questions about these ethical issues as well as chapter content.

Color and Typography Are Signposts

The layers into which the book is divided are color coded within the text. The opening spread for each chapter shows an image of the onion in which the outermost color corresponds to the current layer. This color is repeated in header bars and section numbers throughout the layer. Each opening spread also visually indicates where the chapter is within the layer and the book.

We have said that the first and last chapters form bookends. Although they are not part of the layers of the computing onion, these chapters are color coded like the others. Open the book anywhere and you can immediately tell where you are within the layers of computing.

To visually separate the abstract from the concrete in the programming layer, we use different fonts for algorithms, including identifiers in running text, and program code. You know at a glance whether the discussion is at the logical (algorithmic) level or at the programming-language level. In order to distinguish visually between an address and the contents of an address, we color addresses in orange.

Color is especially useful in Chapter 6, "Low-Level Programming Languages and Pseudocode." Instructions are color coded to differentiate the parts of an instruction. The operation code is blue, the register designation is clear, and the addressing mode specifier is green. Operands are shaded gray. As in other chapters, addresses are in orange.

Instructor Resources

For the instructor, slides in PowerPoint format, a test bank, and answers to the book's end-of-chapter exercises are available for free download at http://go.jblearning.com/CSI6e.

ACKNOWLEDGMENTS

Our adopters have been most helpful during this revision. To those who took the time to respond to our online survey: Thanks to all of you. We are also grateful to the reviewers of the previous editions of the text:

Tim Bower, Kansas State University
Mikhail Brikman, Salem State College
Jacques Carette, McMaster University
Howard Francis, Pikeville College
Jim Jones, Graceland University
Murray Levy, West Los Angeles College
Lew Lowther, York University
Jeffrey McConnell, Canisius College
Richard Schlesinger, Kennesaw State University
Richard Spinello, Boston College
Herman Tavani, Rivier College
Amy Woszczynski, Kennesaw State University
C. Michael Allen, UNC Charlotte
Lofton Bullard, Florida Atlantic University
Cerian Jones, University of Alberta
Calvin Ribbens, Virginia Tech
Susan Sells, Wichita State University
R. Mark Meyer, Canisius College
Tom Wiggen, University of North Dakota
Mary Dee Harris, Chris Edmonson-Yurkanan, Ben Kuipers, and Glenn Downing, The University of Texas at Austin
Dave Stauffer, Penn State
John McCormick, University of Northern Iowa
Dan Joyce, Villanova University
Mike Goldwasser, St. Louis University
Andrew Harrington, Loyola University Chicago
Daniel R. Collins, Mass Bay Community College
J. Stanley Warford, Pepperdine University
Richard C. Detmer, Middle Tennessee State University
Chip Weems, University of Massachusetts Amherst
Heather Chandler, Westwood College
Mark Holthouse, Westwood High School
Robert Vermilyer, St. Thomas Aquinas College

Bob Blucher, Lane Community College
Dale Fletter, Folsom Lake College
Dr. Jerry Westfall, Liberty University
Dwayne Towell, Abilene Christian University
Kara Nance, University of Alaska
Lisa Michaud, Merrimack College
Jeffrey Bergamini, Mendocino College
Johanna Horowitz, Siena College
Lonnie R. Nelson, Hannibal-LaGrange University
Marie Hartlein, Montgomery County Community College
Mark Terwilliger, Lake Superior State University
Patricia Roth Pierce, Southern Polytechnic State University
Quentin J. White, Sr., Palomar College
Rakesh Arya, University of Maryland Eastern Shore
William Honig, Loyola University Chicago
Barbara Zimmerman, Villanova University
Maria Jump, PhD, King's College
Joe Pistone, Palomar College
Derek Merck, Georgia Perimeter College.

Special thanks to Jeffrey McConnell of Canisius College, who wrote the graphics section in Chapter 14; Herman Tavani of Rivier College, who helped us revise the "Ethical Issues" sections; Richard Spinello of Boston College for his essay on the ethics of blogging; and Paul Toprac, Associate Director of Game Development at The University of Texas at Austin, for his contributions on gaming.

We appreciate and thank our reviewers and colleagues who provided advice and recommendations for the content in this *Sixth Edition*:

David Adams, Grove City College
Marie Arvi, Salisbury University
Bill Cole, Sierra College-Rocklin
Richard Croft, Eastern Oregon University
Linda Ehley, Alverno College
Janet Helwig, Dominican University
James Hicks, Los Angeles Southwest College
Aparna Mahadev, Worcester State University

Mia Moore, Clark Atlanta University
S. Monisha Pulimood, The College of New Jersey
Warren W. Sheaffer, Saint Paul College
Robert Yacobellis, Loyola University Chicago

We also thank the many people at Jones & Bartlett Learning who contributed so much, especially Laura Pagluica, Acquisitions Editor; Taylor Ferracane, Editorial Assistant; and Amy Rose, Director of Production.

I must also thank my tennis buddies for keeping me fit, my bridge buddies for keeping my mind alert, and my family for keeping me grounded.
—ND

I'd like to thank my family for their support.
—JL

SPECIAL FEATURES

Interspersed throughout *Computer Science Illuminated, Sixth Edition* are two special features of note: Ethical Issues and Biographies. A list of each is provided below for immediate access.

ETHICAL ISSUES

Digital Divide (Chapter 1, p. 30)

The FISA Court (Chapter 2, p. 49)

The Fallout from Snowden's Revelations (Chapter 3, p. 86)

Codes of Ethics (Chapter 4, p. 114)

Is Privacy a Thing of the Past? (Chapter 5, p. 148)

Software Piracy (Chapter 6, p.191)

Open-Source Software (Chapter 7, p. 238)

Workplace Monitoring (Chapter 8, p. 279)

Hoaxes and Scams (Chapter 9, p. 328)

Medical Privacy: HIPAA (Chapter 10, p. 360)

Privacy: Opt-In or Opt-Out? (Chapter 11, p. 388)

Politics and the Internet: The Candidate's View (Chapter 12, p. 417)

Initial Public Offerings (Chapter 13, p. 452)

Gaming as an Addiction (Chapter 14, p. 495)

The Effects of Social Networking (Chapter 15, p. 525)

Gambling and the Internet (Chapter 16, p. 557)

Blogging (Chapter 17, p. 586)

Therac-25: Anatomy of a Disaster (Chapter 18, p. 626)

BIOGRAPHIES

Ada Lovelace, the First Programmer (Chapter 1, p. 14)

Grace Murray Hopper (Chapter 2, p. 46)

Bob Bemer (Chapter 3, p. 84)

George Boole (Chapter 4, p. 95)

John Vincent Atanasoff (Chapter 5, p. 128)

Konrad Zuse (Chapter 6, p. 186)

George Polya (Chapter 7, p. 201)

John von Neumann (Chapter 8, p. 254)

Edsger Dijkstra (Chapter 9, p. 315)

Steve Jobs (Chapter 10, p. 357)

Tony Hoare (Chapter 11, p. 385)

Daniel Bricklin (Chapter 12, p. 403)

Herbert A. Simon (Chapter 13, p. 430)

Ivan Sutherland (Chapter 14, p. 468)

Doug Engelbart (Chapter 15, p. 509)

Tim Berners-Lee (Chapter 16, p. 544)

Mavis Batey (Chapter 17, p. 582)

Alan Turing (Chapter 18, p. 619)

Laying the Groundwork

1 The Big Picture

The Information Layer

2 Binary Values and Number Systems

3 Data Representation

The Hardware Layer

4 Gates and Circuits

5 Computing Components

The Programming Layer

6 Low-Level Programming Languages and Pseudocode

7 Problem Solving and Algorithms

8 Abstract Data Types and Subprograms

9 Object-Oriented Design and High-Level Programming Languages

The Operating Systems Layer

10 Operating Systems

11 File Systems and Directories

The Applications Layer

12 Information Systems

13 Artificial Intelligence

14 Simulation, Graphics, Gaming, and Other Applications

The Communications Layer

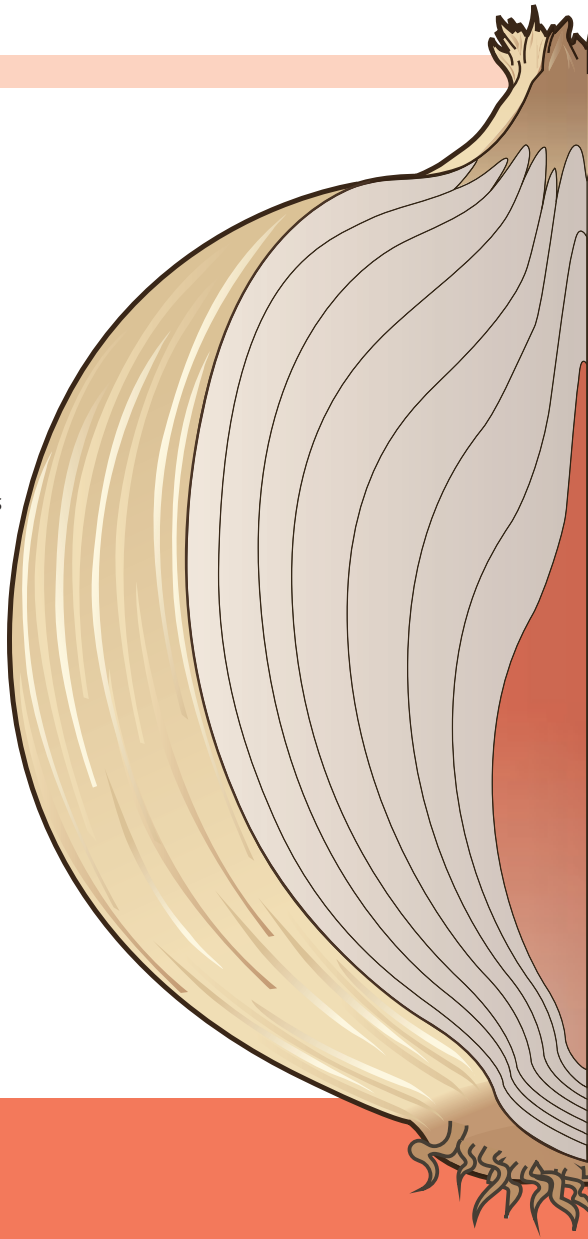
15 Networks

16 The World Wide Web

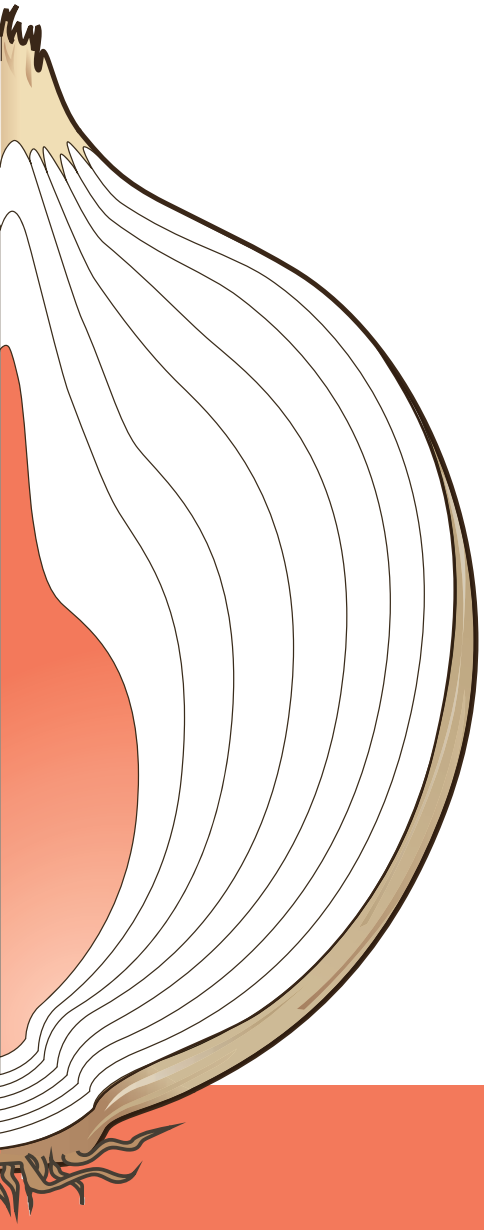
17 Computer Security

In Conclusion

18 Limitations of Computing



LAYING THE GROUNDWORK



1 THE BIG PICTURE

This book is a tour through the world of computing. We explore how computers work—what they do and how they do it, from bottom to top, inside and out. Like an orchestra, a computer system is a collection of many different elements, which combine to form a whole that is far more than the sum of its parts. This chapter provides the big picture, giving an overview of the pieces that we slowly dissect throughout the book and putting those pieces into historical perspective.

Hardware, *software*, *programming*, *web surfing*, and *email* are probably familiar terms to you. Some of you can define these and many more computer-related terms explicitly, whereas others may have only a vague, intuitive understanding of them. This chapter gets everyone on relatively equal footing by establishing common terminology and creating the platform from which we will dive into our exploration of computing.

GOALS

After studying this chapter, you should be able to:

- describe the layers of a computer system.
- describe the concept of abstraction and its relationship to computing.
- describe the history of computer hardware and software.
- describe the changing role of the computer user.
- distinguish between systems programmers and applications programmers.
- distinguish between computing as a tool and computing as a discipline.

Computing system
Computer hardware, software, and data, which interact to solve problems

Computer hardware
The physical elements of a computing system

Computer software
The programs that provide the instructions that a computer executes

1.1 Computing Systems

In this book we explore various aspects of computing systems. Note that we use the term *computing system*, not just *computer*. A computer is a device. A **computing system**, by contrast, is a dynamic entity, used to solve problems and interact with its environment. A computing system is composed of hardware, software, and the data that they manage. **Computer hardware** is the collection of physical elements that make up the machine and its related pieces: boxes, circuit boards, chips, wires, disk drives, keyboards, monitors, printers, and so on. **Computer software** is the collection of programs that provide the instructions that a computer carries out. And at the very heart of a computer system is the information that it manages. Without data, the hardware and software are essentially useless.

The general goals of this book are threefold:

- To give you a solid, broad understanding of how a computing system works
- To develop an appreciation for and understanding of the evolution of modern computing systems
- To give you enough information about computing so that you can decide whether you wish to pursue the subject further

The rest of this section explains how computer systems can be divided into abstract layers and how each layer plays a role. The next section puts the development of computing hardware and software into historical context. This chapter concludes with a discussion about computing as both a tool and a discipline of study.

Layers of a Computing System

A computing system is like an onion, made up of many layers. Each layer plays a specific role in the overall design of the system. These layers are depicted in FIGURE 1.1 and form the general organization of this book. This is the "big picture" that we will continually return to as we explore different aspects of computing systems.

You rarely, if ever, take a bite out of an onion as you would an apple. Rather, you separate the onion into concentric rings. Likewise, in this book we explore aspects of computing one layer at a time. We peel each layer separately and explore it by itself. Each layer, in itself, is not that complicated. In fact, a computer actually does only very simple tasks—it just does them so blindingly fast that many simple tasks can be combined to

accomplish larger, more complicated tasks. When the various computer layers are all brought together, each playing its own role, amazing things result from the combination of these basic ideas.

Let's discuss each of these layers briefly and identify where in this book these ideas are explored in more detail. We essentially work our way from the inside out, which is sometimes referred to as a bottom-up approach.

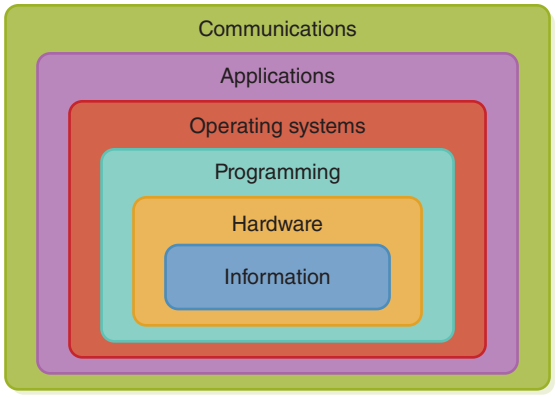


FIGURE 1.1 The layers of a computing system

The innermost layer, information, reflects the way we represent information on a computer. In many ways this is a purely conceptual level. Information on a computer is managed using binary digits, 1 and 0. So to understand computer processing, we must first understand the binary number system and its relationship to other number systems (such as the decimal system, the one humans use on a daily basis). Then we can turn our attention to how we take the myriad types of information we manage—numbers, text, images, audio, and video—and represent them in a binary format. Chapters 2 and 3 explore these issues.

The next layer, hardware, consists of the physical hardware of a computer system. Computer hardware includes devices such as gates and circuits, which control the flow of electricity in fundamental ways. This core electronic circuitry gives rise to specialized hardware components such as the computer's central processing unit (CPU) and memory. Chapters 4 and 5 of the book discuss these topics in detail.

The programming layer deals with software, the instructions used to accomplish computations and manage data. Programs can take many forms, be performed at many levels, and be implemented in many languages. Yet, despite the enormous variety of programming issues, the goal remains the same: to solve problems. Chapters 6 through 9 explore many issues related to programming and the management of data.

Every computer has an operating system (OS) to help manage the computer's resources. Operating systems, such as Windows XP, Linux, or Mac OS, help us interact with the computer system and manage the way hardware devices, programs, and data interact. Knowing what an operating system does is key to understanding the computer in general. These issues are discussed in Chapters 10 and 11.

The previous (inner) layers focus on making a computer system work. The applications layer, by contrast, focuses on using the computer to solve specific real-world problems. We run application programs

to take advantage of the computer's abilities in other areas, such as helping us design a building or play a game. The spectrum of area-specific computer software tools is far-reaching and involves specific subdisciplines of computing, such as information systems, artificial intelligence, and simulation. Application systems are discussed in Chapters 12, 13, and 14.

Computers no longer exist in isolation on someone's desktop. We use computer technology to communicate, and that communication is a fundamental layer at which computing systems operate. Computers are connected into networks so that they can share information and resources. The Internet, for example, evolved into a global network, so there is now almost no place on Earth that you cannot communicate with via computing technology. The World Wide Web makes that communication relatively easy; it has revolutionized computer use and made it accessible to the general public. Chapters 15 and 16 discuss these important issues of computing communication.

The use of computing technology can result in increased security hazards. Some issues of security are dealt with at low levels throughout a computer system. Many of them, though, involve keeping our personal information secure. Chapter 17 discusses several of these issues.

Most of this book focuses on what a computer can do and how it does it. We conclude with a discussion of what a computer *cannot* do, or at least cannot do well. Computers have inherent limitations on their ability to represent information, and they are only as good as their programming makes them. Furthermore, it turns out that some problems cannot be solved at all. Chapter 18 examines these limitations of computers.

Sometimes it is easy to get so caught up in the details that we lose perspective on the big picture. Try to keep that in mind as you progress through the information in this book. Each chapter's opening page reminds you of where we are in the various layers of a computing system. The details all contribute a specific part to a larger whole. Take each step in turn and you will be amazed at how well it all falls into place.

Abstraction

Abstraction A mental model that removes complex details

The levels of a computing system that we just examined are examples of abstraction. An **abstraction** is a mental model, a way to think about something, that removes or hides complex details. An abstraction leaves only the information necessary to accomplish our goal. When we are dealing with a computer on one layer, we don't need to be thinking

about the details of the other layers. For example, when we are writing a program, we don't have to concern ourselves with how the hardware carries out the instructions. Likewise, when we are running an application program, we don't have to be concerned with how that program was written.

Numerous experiments have shown that a human being can actively manage about seven (plus or minus two, depending on the person) pieces of information in short-term memory at one time. This is called *Miller's Law*, based on the psychologist who first investigated it.[1] Other pieces of information are available to us when we need them, but when we focus on a new piece, something else falls back into secondary status.

This concept is similar to the number of balls a juggler can keep in the air at one time. Human beings can mentally juggle about seven balls at once, and when we pick up a new one, we have to drop another. Seven may seem like a small number, but the key is that each ball can represent an abstraction, or a chunk of information. That is, each ball we are juggling can represent a complex topic as long as we can think about it as one idea.

We rely on abstractions every day of our lives. For example, we don't need to know how a car works to drive one to the store. That is, we don't really need to know how the engine works in detail. We need to know only some basics about how to interact with the car: how the pedals and knobs and steering wheel work. And we don't even have to be thinking about all of those things at the same time. See FIGURE 1.2.

FIGURE 1.2 A car engine and the abstraction that allows us to use it

Information hiding
A technique for isolating program pieces by eliminating the ability for one piece to access the information in another

Even if we do know how an engine works, we don't have to think about it while driving. Imagine if, while driving, we had to constantly think about how the spark plugs ignite the fuel that drives the pistons that turn the crankshaft. We'd never get anywhere! A car is much too complicated for us to deal with all at once. All the technical details would be too many balls to juggle at the same time. But once we've abstracted the car down to the way we interact with it, we can deal with it as one entity. The irrelevant details are ignored, at least for the moment.

Information hiding is a concept related to abstraction. A computer programmer often tries to eliminate the need or ability of one part of a program to access information located in another part. This technique keeps the pieces of the program isolated from each other, which reduces errors and makes each piece easier to understand. Abstraction focuses on the external view—the way something behaves and the way we interact with it. Information hiding is a design feature that gives rise to the abstractions that make something easier to work with. Information hiding and abstraction are two sides of the same coin.

FIGURE 1.3 Marcel Duchamp discussing his abstract painting *Nude Descending a Staircase*

Abstract art, as the name implies, is another example of abstraction. An abstract painting represents something but doesn't get bogged down in the details of reality. Consider the painting shown in FIGURE 1.3, entitled *Nude Descending a Staircase*. You can see only the basic hint of the woman and the staircase, because the artist is not interested in the details of exactly how the woman or the staircase looks. Those details are irrelevant to the effect the artist is trying to create. In fact, the realistic details would get in the way of the issues that the artist thinks are important.

Abstraction is the key to computing. The layers of a computing system embody the idea of abstraction. And abstractions keep appearing within individual layers in various ways as well. In fact, abstraction can be seen throughout the entire evolution of computing systems, which we explore in the next section.

1.2 The History of Computing

The historical foundation of computing goes a long way toward explaining why computing systems today are designed as they are. Think of this section as a story whose characters and events have led to the place we are now and form the foundation of the exciting future to come. We examine the history of computing hardware and software separately because each has its own impact on how computing systems evolved into the layered model we use as the outline for this book.

This history is written as a narrative, with no intent to formally define the concepts discussed. In subsequent chapters, we return to these concepts and explore them in more detail.

A Brief History of Computing Hardware

The devices that assist humans in various forms of computation have their roots in the ancient past and have continued to evolve until the present day. Let's take a brief tour through the history of computing hardware.

Early History

Many people believe that Stonehenge, the famous collection of rock monoliths in Great Britain, is an early form of a calendar or astrological calculator. The *abacus*, which appeared in the sixteenth century BC, was developed as an instrument to record numeric values and on which a human can perform basic arithmetic.

In the middle of the seventeenth century, Blaise Pascal, a French mathematician, built and sold gear-driven mechanical machines, which performed whole-number addition and subtraction. Later in the seventeenth century, a German mathematician, Gottfried Wilhelm von Leibniz, built the first mechanical device designed to do all four whole-number operations: addition, subtraction, multiplication, and division. Unfortunately, the state of mechanical gears and levers at that time was such that the Leibniz machine was not very reliable.

In the late eighteenth century, Joseph Jacquard developed what became known as *Jacquard's loom*, used for weaving cloth. The loom used a series of cards with holes punched in them to specify the use of specific colored thread and therefore dictate the design that was woven into the cloth. Although not a computing device, Jacquard's loom was the first to make use of an important form of input: the punched card.

?

Beyond all dreams

"Who can foresee the consequences of such an invention? The Analytical Engine weaves algebraic patterns just as the Jacquard loom weaves flowers and leaves. The engine might compose elaborate and scientific pieces of music of any degree of complexity or extent."

—Ada, Countess of Lovelace, 1843[2]