Preface xix

I Introduction to Computers, the Internet and the Web 1

1.1 Introduction 2

1.2 Computers and the Internet in Industry and Research 2

1.3 Hardware and Software 5

1.3.1 Moore’s Law 6

1.3.2 Computer Organization 6

1.4 Data Hierarchy 7

1.5 Programming Languages 9

1.6 The C Programming Language 10

1.7 C Standard Library 12

1.8 C++ and Other C-Based Languages 13

1.9 Object Technology 14

1.10 Typical C Program Development Environment 16

1.10.1 Phase 1: Creating a Program 16

1.10.2 Phases 2 and 3: Preprocessing and Compiling a C Program 16

1.10.3 Phase 4: Linking 18

1.10.4 Phase 5: Loading 18

1.10.5 Phase 6: Execution 18

1.10.6 Problems That May Occur at Execution Time 18

1.10.7 Standard Input, Standard Output and Standard Error Streams 18

1.11 Test-Driving a C Application in Windows, Linux and Mac OS X 19

1.11.1 Running a C Application from the Windows Command Prompt 20

1.11.2 Running a C Application Using GNU C with Linux 22

1.11.3 Running a C Application Using GNU C with Mac OS X 25

1.12 Operating Systems 27

1.12.1 Windows—A Proprietary Operating System 28

1.12.2 Linux—An Open-Source Operating System 28

1.12.3 Apple’s Mac OS X; Apple’s iOS for iPhone®, iPad® and iPod Touch® Devices 29

1.12.4 Google’s Android 29
Contents

1.13 The Internet and World Wide Web 30
1.14 Some Key Software Development Terminology 31
1.15 Keeping Up-to-Date with Information Technologies 33
1.16 Web Resources 34

2 Introduction to C Programming 40
2.1 Introduction 41
2.2 A Simple C Program: Printing a Line of Text 41
2.3 Another Simple C Program: Adding Two Integers 45
2.4 Memory Concepts 49
2.5 Arithmetic in C 50
2.6 Decision Making: Equality and Relational Operators 54
2.7 Secure C Programming 58

3 Structured Program Development in C 70
3.1 Introduction 71
3.2 Algorithms 71
3.3 Pseudocode 71
3.4 Control Structures 72
3.5 The if Selection Statement 74
3.6 The if…else Selection Statement 75
3.7 The while Repetition Statement 79
3.8 Formulating Algorithms Case Study 1: Counter-Controlled Repetition 80
3.9 Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 2: Sentinel-Controlled Repetition 82
3.10 Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 3: Nested Control Statements 89
3.11 Assignment Operators 93
3.12 Increment and Decrement Operators 93
3.13 Secure C Programming 96

4 C Program Control 114
4.1 Introduction 115
4.2 Repetition Essentials 115
4.3 Counter-Controlled Repetition 116
4.4 for Repetition Statement 117
4.5 for Statement: Notes and Observations 120
4.6 Examples Using the for Statement 121
4.7 switch Multiple-Selection Statement 124
4.8 do…while Repetition Statement 130
4.9 break and continue Statements 132
4.10 Logical Operators 134
4.11 Confusing Equality (==) and Assignment (=) Operators 137
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.12</td>
<td>Structured Programming Summary</td>
<td>138</td>
</tr>
<tr>
<td>4.13</td>
<td>Secure C Programming</td>
<td>143</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>159</td>
</tr>
<tr>
<td>5.2</td>
<td>Program Modules in C</td>
<td>159</td>
</tr>
<tr>
<td>5.3</td>
<td>Math Library Functions</td>
<td>160</td>
</tr>
<tr>
<td>5.4</td>
<td>Functions</td>
<td>162</td>
</tr>
<tr>
<td>5.5</td>
<td>Function Definitions</td>
<td>162</td>
</tr>
<tr>
<td>5.6</td>
<td>Function Prototypes: A Deeper Look</td>
<td>166</td>
</tr>
<tr>
<td>5.7</td>
<td>Function Call Stack and Stack Frames</td>
<td>169</td>
</tr>
<tr>
<td>5.8</td>
<td>Headers</td>
<td>172</td>
</tr>
<tr>
<td>5.9</td>
<td>Passing Arguments By Value and By Reference</td>
<td>173</td>
</tr>
<tr>
<td>5.10</td>
<td>Random Number Generation</td>
<td>174</td>
</tr>
<tr>
<td>5.11</td>
<td>Example: A Game of Chance</td>
<td>179</td>
</tr>
<tr>
<td>5.12</td>
<td>Storage Classes</td>
<td>182</td>
</tr>
<tr>
<td>5.13</td>
<td>Scope Rules</td>
<td>184</td>
</tr>
<tr>
<td>5.14</td>
<td>Recursion</td>
<td>187</td>
</tr>
<tr>
<td>5.15</td>
<td>Example Using Recursion: Fibonacci Series</td>
<td>191</td>
</tr>
<tr>
<td>5.16</td>
<td>Recursion vs. Iteration</td>
<td>194</td>
</tr>
<tr>
<td>5.17</td>
<td>Secure C Programming</td>
<td>197</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>217</td>
</tr>
<tr>
<td>6.2</td>
<td>Arrays</td>
<td>217</td>
</tr>
<tr>
<td>6.3</td>
<td>Defining Arrays</td>
<td>218</td>
</tr>
<tr>
<td>6.4</td>
<td>Array Examples</td>
<td>219</td>
</tr>
<tr>
<td>6.5</td>
<td>Passing Arrays to Functions</td>
<td>232</td>
</tr>
<tr>
<td>6.6</td>
<td>Sorting Arrays</td>
<td>236</td>
</tr>
<tr>
<td>6.7</td>
<td>Case Study: Computing Mean, Median and Mode Using Arrays</td>
<td>239</td>
</tr>
<tr>
<td>6.8</td>
<td>Searching Arrays</td>
<td>244</td>
</tr>
<tr>
<td>6.9</td>
<td>Multidimensional Arrays</td>
<td>249</td>
</tr>
<tr>
<td>6.10</td>
<td>Variable-Length Arrays</td>
<td>256</td>
</tr>
<tr>
<td>6.11</td>
<td>Secure C Programming</td>
<td>259</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>278</td>
</tr>
<tr>
<td>7.2</td>
<td>Pointer Variable Definitions and Initialization</td>
<td>278</td>
</tr>
<tr>
<td>7.3</td>
<td>Pointer Operators</td>
<td>279</td>
</tr>
<tr>
<td>7.4</td>
<td>Passing Arguments to Functions by Reference</td>
<td>282</td>
</tr>
<tr>
<td>7.5</td>
<td>Using the const Qualifier with Pointers</td>
<td>284</td>
</tr>
<tr>
<td>7.5.1</td>
<td>Converting a String to Uppercase Using a Non-Constant Pointer to Non-Constant Data</td>
<td>287</td>
</tr>
</tbody>
</table>
8.9.4 Function `memchr`  363
8.9.5 Function `memset`  364

8.10 Other Functions of the String-Handling Library  365
8.10.1 Function `strerror`  365
8.10.2 Function `strlen`  365

8.11 Secure C Programming  366

9  C Formatted Input/Output  379
9.1 Introduction  380
9.2 Streams  380
9.3 Formatting Output with `printf`  380
9.4 Printing Integers  381
9.5 Printing Floating-Point Numbers  382
9.6 Printing Strings and Characters  384
9.7 Other Conversion Specifiers  385
9.8 Printing with Field Widths and Precision  386
9.9 Using Flags in the `printf` Format Control String  388
9.10 Printing Literals and Escape Sequences  391
9.11 Reading Formatted Input with `scanf`  391
9.12 Secure C Programming  398

10 C Structures, Unions, Bit Manipulation and Enumerations  405
10.1 Introduction  406
10.2 Structure Definitions  406
10.2.1 Self-Referential Structures  407
10.2.2 Defining Variables of Structure Types  407
10.2.3 Structure Tag Names  408
10.2.4 Operations That Can Be Performed on Structures  408
10.3 Initializing Structures  409
10.4 Accessing Structure Members  409
10.5 Using Structures with Functions  411
10.6 `typedef`  411
10.7 Example: High-Performance Card Shuffling and Dealing Simulation  412
10.8 Unions  415
10.8.1 Union Declarations  415
10.8.2 Operations That Can Be Performed on Unions  415
10.8.3 Initializing Unions in Declarations  416
10.8.4 Demonstrating Unions  416
10.9 Bitwise Operators  417
10.9.1 Displaying an Unsigned Integer in Bits  418
10.9.2 Making Function `displayBits` More Scalable and Portable  420
10.9.3 Using the Bitwise AND, Inclusive OR, Exclusive OR and Complement Operators  420
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.9.4  Using the Bitwise Left- and Right-Shift Operators</td>
<td>423</td>
</tr>
<tr>
<td>10.9.5  Bitwise Assignment Operators</td>
<td>425</td>
</tr>
<tr>
<td>10.10   Bit Fields</td>
<td>426</td>
</tr>
<tr>
<td>10.11   Enumeration Constants</td>
<td>429</td>
</tr>
<tr>
<td>10.12   Secure C Programming</td>
<td>431</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11      C File Processing</td>
<td>441</td>
</tr>
<tr>
<td>11.1    Introduction</td>
<td>442</td>
</tr>
<tr>
<td>11.2    Files and Streams</td>
<td>442</td>
</tr>
<tr>
<td>11.3    Creating a Sequential-Access File</td>
<td>443</td>
</tr>
<tr>
<td>11.4    Reading Data from a Sequential-Access File</td>
<td>448</td>
</tr>
<tr>
<td>11.5    Random-Access Files</td>
<td>452</td>
</tr>
<tr>
<td>11.6    Creating a Random-Access File</td>
<td>453</td>
</tr>
<tr>
<td>11.7    Writing Data Randomly to a Random-Access File</td>
<td>455</td>
</tr>
<tr>
<td>11.8    Reading Data from a Random-Access File</td>
<td>458</td>
</tr>
<tr>
<td>11.9    Case Study: Transaction-Processing Program</td>
<td>459</td>
</tr>
<tr>
<td>11.10   Secure C Programming</td>
<td>465</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12      C Data Structures</td>
<td>476</td>
</tr>
<tr>
<td>12.1    Introduction</td>
<td>477</td>
</tr>
<tr>
<td>12.2    Self-Referential Structures</td>
<td>478</td>
</tr>
<tr>
<td>12.3    Dynamic Memory Allocation</td>
<td>478</td>
</tr>
<tr>
<td>12.4    Linked Lists</td>
<td>479</td>
</tr>
<tr>
<td>12.4.1  Function insert</td>
<td>485</td>
</tr>
<tr>
<td>12.4.2  Function delete</td>
<td>487</td>
</tr>
<tr>
<td>12.4.3  Function printList</td>
<td>488</td>
</tr>
<tr>
<td>12.5    Stacks</td>
<td>488</td>
</tr>
<tr>
<td>12.5.1  Function push</td>
<td>492</td>
</tr>
<tr>
<td>12.5.2  Function pop</td>
<td>492</td>
</tr>
<tr>
<td>12.5.3  Applications of Stacks</td>
<td>493</td>
</tr>
<tr>
<td>12.6    Queues</td>
<td>494</td>
</tr>
<tr>
<td>12.6.1  Function enqueue</td>
<td>498</td>
</tr>
<tr>
<td>12.6.2  Function dequeue</td>
<td>499</td>
</tr>
<tr>
<td>12.7    Trees</td>
<td>500</td>
</tr>
<tr>
<td>12.7.1  Function insertNode</td>
<td>504</td>
</tr>
<tr>
<td>12.7.2  Traversals: Functions inOrder, preOrder and postOrder</td>
<td>504</td>
</tr>
<tr>
<td>12.7.3  Duplicate Elimination</td>
<td>505</td>
</tr>
<tr>
<td>12.7.4  Binary Tree Search</td>
<td>505</td>
</tr>
<tr>
<td>12.7.5  Other Binary Tree Operations</td>
<td>505</td>
</tr>
<tr>
<td>12.8    Secure C Programming</td>
<td>506</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13      C Preprocessor</td>
<td>517</td>
</tr>
<tr>
<td>13.1    Introduction</td>
<td>518</td>
</tr>
<tr>
<td>13.2    #include Preprocessor Directive</td>
<td>518</td>
</tr>
</tbody>
</table>

©Copyright 1992-2013 by Pearson Education, Inc. All Rights Reserved.
Contents

13.3  #define Preprocessor Directive: Symbolic Constants  519
13.4  #define Preprocessor Directive: Macros  519
13.5  Conditional Compilation  521
13.6  #error and #pragma Preprocessor Directives  522
13.7  # and ## Operators  523
13.8  Line Numbers  523
13.9  Predefined Symbolic Constants  523
13.10  Assertions  524
13.11  Secure C Programming  524

14  Other C Topics  529
14.1  Introduction  530
14.2  Redirecting I/O  530
14.3  Variable-Length Argument Lists  531
14.4  Using Command-Line Arguments  533
14.5  Notes on Compiling Multiple-Source-File Programs  534
14.6  Program Termination with exit and atexit  536
14.7  Suffixes for Integer and Floating-Point Literals  537
14.8  Signal Handling  538
14.9  Dynamic Memory Allocation: Functions calloc and realloc  540
14.10  Unconditional Branching with goto  541

15  C++ as a Better C; Introducing Object Technology  547
15.1  Introduction  548
15.2  C++  548
15.3  A Simple Program: Adding Two Integers  549
15.4  C++ Standard Library  551
15.5  Header Files  552
15.6  Inline Functions  554
15.7  References and Reference Parameters  556
15.8  Empty Parameter Lists  561
15.9  Default Arguments  561
15.10  Unary Scope Resolution Operator  563
15.11  Function Overloading  564
15.12  Function Templates  567
15.13  Introduction to C++ Standard Library Class Template vector  570
15.14  Introduction to Object Technology and the UML  576
15.15  Wrap-Up  579

16  Introduction to Classes, Objects and Strings  586
16.1  Introduction  587
16.2  Defining a Class with a Member Function  587
### Contents

16.3 Defining a Member Function with a Parameter 590  
16.4 Data Members, *set* Functions and *get* Functions 593  
16.5 Initializing Objects with Constructors 599  
16.6 Placing a Class in a Separate File for Reusability 603  
16.7 Separating Interface from Implementation 606  
16.8 Validating Data with *set* Functions 612  
16.9 Wrap-Up 617

17 Classes: A Deeper Look, Part 1 623  
17.1 Introduction 624  
17.2 *Time* Class Case Study 625  
17.3 Class Scope and Accessing Class Members 632  
17.4 Separating Interface from Implementation 633  
17.5 Access Functions and Utility Functions 634  
17.6 *Time* Class Case Study: Constructors with Default Arguments 637  
17.7 Destructors 642  
17.8 When Constructors and Destructors Are Called 643  
17.9 *Time* Class Case Study: A Subtle Trap—Returning a Reference to a *private* Data Member 646  
17.10 Default Memberwise Assignment 649  
17.11 Wrap-Up 652

18 Classes: A Deeper Look, Part 2 658  
18.1 Introduction 659  
18.2 *const* (Constant) Objects and *const* Member Functions 659  
18.3 Composition: Objects as Members of Classes 667  
18.4 *friend* Functions and *friend* Classes 673  
18.5 Using the *this* Pointer 675  
18.6 *static* Class Members 680  
18.7 Proxy Classes 685  
18.8 Wrap-Up 689

19 Operator Overloading; Class *string* 695  
19.1 Introduction 696  
19.2 Using the Overloaded Operators of Standard Library Class *string* 697  
19.3 Fundamentals of Operator Overloading 700  
19.4 Overloading Binary Operators 701  
19.5 Overloading the Binary Stream Insertion and Stream Extraction Operators 702  
19.6 Overloading Unary Operators 706  
19.7 Overloading the Unary Prefix and Postfix ++ and -- Operators 707  
19.8 Case Study: A *Date* Class 708  
19.9 Dynamic Memory Management 713
20 Object-Oriented Programming: Inheritance 743
  20.1 Introduction 744
  20.2 Base Classes and Derived Classes 744
  20.3 protected Members 747
  20.4 Relationship between Base Classes and Derived Classes 747
    20.4.1 Creating and Using a CommissionEmployee Class 748
    20.4.2 Creating a BasePlusCommissionEmployee Class Without Using Inheritance 752
    20.4.3 Creating a CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy 758
    20.4.4 CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data 763
    20.4.5 CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using private Data 766
  20.5 Constructors and Destructors in Derived Classes 771
  20.6 public, protected and private Inheritance 771
  20.7 Software Engineering with Inheritance 772
  20.8 Wrap-Up 773

21 Object-Oriented Programming: Polymorphism 778
  21.1 Introduction 779
  21.2 Introduction to Polymorphism: Polymorphic Video Game 780
  21.3 Relationships Among Objects in an Inheritance Hierarchy 780
    21.3.1 Invoking Base-Class Functions from Derived-Class Objects 781
    21.3.2 Aiming Derived-Class Pointers at Base-Class Objects 784
    21.3.3 Derived-Class Member-Function Calls via Base-Class Pointers 785
    21.3.4 Virtual Functions 787
  21.4 Type Fields and switch Statements 793
  21.5 Abstract Classes and Pure virtual Functions 793
  21.6 Case Study: Payroll System Using Polymorphism 795
    21.6.1 Creating Abstract Base Class Employee 796
    21.6.2 Creating Concrete Derived Class SalariedEmployee 800
    21.6.3 Creating Concrete Derived Class CommissionEmployee 802
    21.6.4 Creating Indirect Concrete Derived Class BasePlusCommissionEmployee 804
    21.6.5 Demonstrating Polymorphic Processing 806
Contents

21.7  (Optional) Polymorphism, Virtual Functions and Dynamic Binding
      “Under the Hood”  810
21.8  Case Study: Payroll System Using Polymorphism and Runtime Type
      Information with Downcasting, dynamic_cast, typeid and type_info  813
21.9  Virtual Destructors  817
21.10 Wrap-Up  817

22  Templates  823
22.1  Introduction  824
22.2  Function Templates  824
22.3  Overloading Function Templates  827
22.4  Class Templates  828
22.5  Nontype Parameters and Default Types for Class Templates  834
22.6  Wrap-Up  835

23  Stream Input/Output  839
23.1  Introduction  840
23.2  Streams  841
      23.2.1  Classic Streams vs. Standard Streams  841
      23.2.2  istream Library Headers  842
      23.2.3  Stream Input/Output Classes and Objects  842
23.3  Stream Output  845
      23.3.1  Output of char * Variables  845
      23.3.2  Character Output Using Member Function put  845
23.4  Stream Input  846
      23.4.1  get and getline Member Functions  846
      23.4.2  istream Member Functions peek, putback and ignore  849
      23.4.3  Type-Safe I/O  849
23.5  Unformatted I/O Using read, write and gcount  849
23.6  Introduction to Stream Manipulators  850
      23.6.1  Integral Stream Base: dec, oct, hex and setbase  851
      23.6.2  Floating-Point Precision (precision, setprecision)  851
      23.6.3  Field Width (width, setw)  853
      23.6.4  User-Defined Output Stream Manipulators  854
23.7  Stream Format States and Stream Manipulators  856
      23.7.1  Trailing Zeros and Decimal Points (showpoint)  856
      23.7.2  Justification (left, right and internal)  857
      23.7.3  Padding (fill, setfill)  859
      23.7.4  Integral Stream Base (dec, oct, hex, showbase)  860
      23.7.5  Floating-Point Numbers; Scientific and Fixed Notation
             (scientific, fixed)  861
      23.7.6  Uppercase/Lowercase Control (uppercase)  862
      23.7.7  Specifying Boolean Format (boolalpha)  862
      23.7.8  Setting and Resetting the Format State via Member
             Function flags  863
23.8 Stream Error States 864
23.9 Tying an Output Stream to an Input Stream 866
23.10 Wrap-Up 867

24 Exception Handling: A Deeper Look 876
24.1 Introduction 877
24.2 Example: Handling an Attempt to Divide by Zero 877
24.3 When to Use Exception Handling 883
24.4 Rethrowing an Exception 884
24.5 Processing Unexpected Exceptions 885
24.6 Stack Unwinding 886
24.7 Constructors, Destructors and Exception Handling 888
24.8 Exceptions and Inheritance 888
24.9 Processing new Failures 889
24.10 Class unique_ptr and Dynamic Memory Allocation 892
24.11 Standard Library Exception Hierarchy 894
24.12 Wrap-Up 896

A Operator Precedence Charts 902

B ASCII Character Set 906

C Number Systems 907
C.1 Introduction 908
C.2 Abbreviating Binary Numbers as Octal and Hexadecimal Numbers 911
C.3 Converting Octal and Hexadecimal Numbers to Binary Numbers 912
C.4 Converting from Binary, Octal or Hexadecimal to Decimal 912
C.5 Converting from Decimal to Binary, Octal or Hexadecimal 913
C.6 Negative Binary Numbers: Two’s Complement Notation 915

D Game Programming: Solving Sudoku 920
D.1 Introduction 920
D.2 Deitel Sudoku Resource Center 921
D.3 Solution Strategies 921
D.4 Programming Sudoku Puzzle Solvers 925
D.5 Generating New Sudoku Puzzles 926
D.6 Conclusion 928

Appendices on the Web 929

Index 930
Contents

Appendices E through H are PDF documents posted online at the book’s Companion Website (located at www.pearsonhighered.com/deitel).

E  Sorting: A Deeper Look

F  Introduction to the New C Standard

G  Using the Visual Studio Debugger

H  Using the GNU Debugger