Appendices F, G and H are PDF documents posted online at the book’s Companion Website (located at www.pearsonhighered.com/deitel).

## Preface

<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>I Introduction to Computers, the Internet and the Web</td>
</tr>
<tr>
<td>1.1 Introduction</td>
</tr>
<tr>
<td>1.2 Hardware and Software</td>
</tr>
<tr>
<td>1.2.1 Moore’s Law</td>
</tr>
<tr>
<td>1.2.2 Computer Organization</td>
</tr>
<tr>
<td>1.3 Data Hierarchy</td>
</tr>
<tr>
<td>1.4 Machine Languages, Assembly Languages and High-Level Languages</td>
</tr>
<tr>
<td>1.5 The C Programming Language</td>
</tr>
<tr>
<td>1.6 C Standard Library</td>
</tr>
<tr>
<td>1.7 C++ and Other C-Based Languages</td>
</tr>
<tr>
<td>1.8 Object Technology</td>
</tr>
<tr>
<td>1.8.1 The Automobile as an Object</td>
</tr>
<tr>
<td>1.8.2 Methods and Classes</td>
</tr>
<tr>
<td>1.8.3 Instantiation</td>
</tr>
<tr>
<td>1.8.4 Reuse</td>
</tr>
<tr>
<td>1.8.5 Messages and Method Calls</td>
</tr>
<tr>
<td>1.8.6 Attributes and Instance Variables</td>
</tr>
<tr>
<td>1.8.7 Encapsulation and Information Hiding</td>
</tr>
<tr>
<td>1.8.8 Inheritance</td>
</tr>
<tr>
<td>1.9 Typical C Program-Development Environment</td>
</tr>
<tr>
<td>1.9.1 Phase 1: Creating a Program</td>
</tr>
<tr>
<td>1.9.2 Phases 2 and 3: Preprocessing and Compiling a C Program</td>
</tr>
<tr>
<td>1.9.3 Phase 4: Linking</td>
</tr>
<tr>
<td>1.9.4 Phase 5: Loading</td>
</tr>
<tr>
<td>1.9.5 Phase 6: Execution</td>
</tr>
<tr>
<td>1.9.6 Problems That May Occur at Execution Time</td>
</tr>
<tr>
<td>1.9.7 Standard Input, Standard Output and Standard Error Streams</td>
</tr>
<tr>
<td>1.10 Test-Driving a C Application in Windows, Linux and Mac OS X</td>
</tr>
<tr>
<td>1.10.1 Running a C Application from the Windows Command Prompt</td>
</tr>
<tr>
<td>1.10.2 Running a C Application Using GNU C with Linux</td>
</tr>
</tbody>
</table>

1.10.3 Running a C Application Using the Terminal on Mac OS X  24
1.11 Operating Systems  27
1.11.1 Windows—A Proprietary Operating System  27
1.11.2 Linux—An Open-Source Operating System  27
1.11.3 Apple’s Mac OS X; Apple’s iOS for iPhone®, iPad® and iPod Touch® Devices  28
1.11.4 Google’s Android  28
1.12 The Internet and World Wide Web  29
1.12.1 The Internet: A Network of Networks  29
1.12.2 The World Wide Web: Making the Internet User-Friendly  29
1.12.3 Web Services  30
1.12.4 Ajax  32
1.12.5 The Internet of Things  32
1.13 Some Key Software Terminology  32
1.14 Keeping Up-to-Date with Information Technologies  34

2  Introduction to C Programming  39
2.1 Introduction  40
2.2 A Simple C Program: Printing a Line of Text  40
2.3 Another Simple C Program: Adding Two Integers  44
2.4 Memory Concepts  48
2.5 Arithmetic in C  49
2.6 Decision Making: Equality and Relational Operators  53
2.7 Secure C Programming  57

3  Structured Program Development in C  69
3.1 Introduction  70
3.2 Algorithms  70
3.3 Pseudocode  70
3.4 Control Structures  71
3.5 The if Selection Statement  73
3.6 The if…else Selection Statement  74
3.7 The while Iteration Statement  78
3.8 Formulating Algorithms Case Study 1: Counter-Controlled Iteration  79
3.9 Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 2: Sentinel-Controlled Iteration  82
3.10 Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 3: Nested Control Statements  88
3.11 Assignment Operators  92
3.12 Increment and Decrement Operators  93
3.13 Secure C Programming  95

4  C Program Control  113
4.1 Introduction  114
4.2 Iteration Essentials 114
4.3 Counter-Controlled Iteration 115
4.4 for Iteration Statement 116
4.5 for Statement: Notes and Observations 119
4.6 Examples Using the for Statement 120
4.7 switch Multiple-Selection Statement 123
4.8 do…while Iteration Statement 129
4.9 break and continue Statements 130
4.10 Logical Operators 132
4.11 Confusing Equality (=) and Assignment (==) Operators 135
4.12 Structured Programming Summary 137
4.13 Secure C Programming 142

5 C Functions 157
5.1 Introduction 158
5.2 Modularizing Programs in C 158
5.3 Math Library Functions 159
5.4 Functions 161
5.5 Function Definitions 161
  5.5.1 square Function 162
  5.5.2 maximum Function 165
5.6 Function Prototypes: A Deeper Look 166
5.7 Function Call Stack and Stack Frames 168
5.8 Headers 172
5.9 Passing Arguments By Value and By Reference 173
5.10 Random Number Generation 174
5.11 Example: A Game of Chance; Introducing enum 178
5.12 Storage Classes 182
5.13 Scope Rules 184
5.14 Recursion 187
5.15 Example Using Recursion: Fibonacci Series 190
5.16 Recursion vs. Iteration 194
5.17 Secure C Programming 195

6 C Arrays 214
6.1 Introduction 215
6.2 Arrays 215
6.3 Defining Arrays 217
6.4 Array Examples 217
  6.4.1 Defining an Array and Using a Loop to Set the Array’s Element Values 217
  6.4.2 Initializing an Array in a Definition with an Initializer List 218
  6.4.3 Specifying an Array’s Size with a Symbolic Constant and Initializing Array Elements with Calculations 220

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4.4</td>
<td>Summing the Elements of an Array</td>
<td>221</td>
</tr>
<tr>
<td>6.4.5</td>
<td>Using Arrays to Summarize Survey Results</td>
<td>222</td>
</tr>
<tr>
<td>6.4.6</td>
<td>Graphing Array Element Values with Histograms</td>
<td>224</td>
</tr>
<tr>
<td>6.4.7</td>
<td>Rolling a Die 60,000,000 Times and Summarizing the Results in an Array</td>
<td>225</td>
</tr>
<tr>
<td>6.5</td>
<td>Using Character Arrays to Store and Manipulate Strings</td>
<td>225</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Initializing a Character Array with a String</td>
<td>226</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Initializing a Character Array with an Initializer List of Characters</td>
<td>226</td>
</tr>
<tr>
<td>6.5.3</td>
<td>Accessing the Characters in a String</td>
<td>226</td>
</tr>
<tr>
<td>6.5.4</td>
<td>Inputting into a Character Array</td>
<td>226</td>
</tr>
<tr>
<td>6.5.5</td>
<td>Outputting a Character Array That Represents a String</td>
<td>227</td>
</tr>
<tr>
<td>6.5.6</td>
<td>Demonstrating Character Arrays</td>
<td>227</td>
</tr>
<tr>
<td>6.6</td>
<td>Static Local Arrays and Automatic Local Arrays</td>
<td>228</td>
</tr>
<tr>
<td>6.7</td>
<td>Passing Arrays to Functions</td>
<td>230</td>
</tr>
<tr>
<td>6.8</td>
<td>Sorting Arrays</td>
<td>234</td>
</tr>
<tr>
<td>6.9</td>
<td>Case Study: Computing Mean, Median and Mode Using Arrays</td>
<td>236</td>
</tr>
<tr>
<td>6.10</td>
<td>Searching Arrays</td>
<td>241</td>
</tr>
<tr>
<td>6.10.1</td>
<td>Searching an Array with Linear Search</td>
<td>241</td>
</tr>
<tr>
<td>6.10.2</td>
<td>Searching an Array with Binary Search</td>
<td>242</td>
</tr>
<tr>
<td>6.11</td>
<td>Multidimensional Arrays</td>
<td>246</td>
</tr>
<tr>
<td>6.11.1</td>
<td>Illustrating a Double-Subscripted Array</td>
<td>246</td>
</tr>
<tr>
<td>6.11.2</td>
<td>Initializing a Double-Subscripted Array</td>
<td>247</td>
</tr>
<tr>
<td>6.11.3</td>
<td>Setting the Elements in One Row</td>
<td>249</td>
</tr>
<tr>
<td>6.11.4</td>
<td>Totaling the Elements in a Two-Dimensional Array</td>
<td>249</td>
</tr>
<tr>
<td>6.11.5</td>
<td>Two-Dimensional Array Manipulations</td>
<td>250</td>
</tr>
<tr>
<td>6.12</td>
<td>Variable-Length Arrays</td>
<td>253</td>
</tr>
<tr>
<td>6.13</td>
<td>Secure C Programming</td>
<td>256</td>
</tr>
</tbody>
</table>

### 7 C Pointers

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>275</td>
</tr>
<tr>
<td>7.2</td>
<td>Pointer Variable Definitions and Initialization</td>
<td>276</td>
</tr>
<tr>
<td>7.3</td>
<td>Pointer Operators</td>
<td>277</td>
</tr>
<tr>
<td>7.4</td>
<td>Passing Arguments to Functions by Reference</td>
<td>279</td>
</tr>
<tr>
<td>7.5</td>
<td>Using the const Qualifier with Pointers</td>
<td>283</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>284</td>
</tr>
<tr>
<td>7.5.2</td>
<td>Printing a String One Character at a Time Using a Non-Constant Pointer to Constant Data</td>
<td>285</td>
</tr>
<tr>
<td>7.5.3</td>
<td>Attempting to Modify a Constant Pointer to Non-Constant Data</td>
<td>287</td>
</tr>
<tr>
<td>7.5.4</td>
<td>Attempting to Modify a Constant Pointer to Constant Data</td>
<td>288</td>
</tr>
<tr>
<td>7.6</td>
<td>Bubble Sort Using Pass-by-Reference</td>
<td>289</td>
</tr>
<tr>
<td>7.7</td>
<td>sizeof Operator</td>
<td>292</td>
</tr>
<tr>
<td>7.8</td>
<td>Pointer Expressions and Pointer Arithmetic</td>
<td>295</td>
</tr>
<tr>
<td>7.8.1</td>
<td>Allowed Operators for Pointer Arithmetic</td>
<td>295</td>
</tr>
<tr>
<td>7.8.2</td>
<td>Aiming a Pointer at an Array</td>
<td>295</td>
</tr>
</tbody>
</table>
## 7.8.3 Adding an Integer to a Pointer

### 7.8.4 Subtracting an Integer from a Pointer

### 7.8.5 Incrementing and Decrementing a Pointer

### 7.8.6 Subtracting One Pointer from Another

### 7.8.7 Assigning Pointers to One Another

### 7.8.8 Pointer to `void`

### 7.8.9 Comparing Pointers

### 7.9 Relationship between Pointers and Arrays

#### 7.9.1 Pointer/Offset Notation

#### 7.9.2 Pointer/Index Notation

#### 7.9.3 Cannot Modify an Array Name with Pointer Arithmetic

#### 7.9.4 Demonstrating Pointer Indexing and Offsets

#### 7.9.5 String Copying with Arrays and Pointers

### 7.10 Arrays of Pointers

### 7.11 Case Study: Card Shuffling and Dealing Simulation

### 7.12 Pointers to Functions

#### 7.12.1 Sorting in Ascending or Descending Order

#### 7.12.2 Using Function Pointers to Create a Menu-Driven System

### 7.13 Secure C Programming

## 8 C Characters and Strings

### 8.1 Introduction

### 8.2 Fundamentals of Strings and Characters

### 8.3 Character-Handling Library

#### 8.3.1 Functions `isdigit`, `isalpha`, `isalnum` and `isxdigit`

#### 8.3.2 Functions `islower`, `isupper`, `tolower` and `toupper`

#### 8.3.3 Functions `isspace`, `iscntrl`, `ispunct`, `isprint` and `isgraph`

### 8.4 String-Conversion Functions

#### 8.4.1 Function `strtol`

#### 8.4.2 Function `strtol`

#### 8.4.3 Function `strtol`

### 8.5 Standard Input/Output Library Functions

#### 8.5.1 Functions `fgets` and `putchar`

#### 8.5.2 Function `getchar`

#### 8.5.3 Function `printf`

#### 8.5.4 Function `scanf`

### 8.6 String-Manipulation Functions of the String-Handling Library

#### 8.6.1 Functions `strcpy` and `strncpy`

#### 8.6.2 Functions `strcat` and `strncat`

### 8.7 Comparison Functions of the String-Handling Library

### 8.8 Search Functions of the String-Handling Library
# C Formatted Input/Output

## 9 Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.8.5</td>
<td>Function <code>strspn</code></td>
<td>357</td>
</tr>
<tr>
<td>8.8.6</td>
<td>Function <code>strstr</code></td>
<td>357</td>
</tr>
<tr>
<td>8.8.7</td>
<td>Function <code>strtok</code></td>
<td>358</td>
</tr>
<tr>
<td>8.9</td>
<td>Memory Functions of the String-Handling Library</td>
<td>359</td>
</tr>
<tr>
<td>8.9.1</td>
<td>Function <code>memcpy</code></td>
<td>360</td>
</tr>
<tr>
<td>8.9.2</td>
<td>Function <code>memmove</code></td>
<td>361</td>
</tr>
<tr>
<td>8.9.3</td>
<td>Function <code>memcmp</code></td>
<td>362</td>
</tr>
<tr>
<td>8.9.4</td>
<td>Function <code>memchr</code></td>
<td>362</td>
</tr>
<tr>
<td>8.9.5</td>
<td>Function <code>memset</code></td>
<td>363</td>
</tr>
<tr>
<td>8.10</td>
<td>Other Functions of the String-Handling Library</td>
<td>363</td>
</tr>
<tr>
<td>8.10.1</td>
<td>Function <code>strerror</code></td>
<td>364</td>
</tr>
<tr>
<td>8.10.2</td>
<td>Function <code>strlen</code></td>
<td>364</td>
</tr>
<tr>
<td>8.11</td>
<td>Secure C Programming</td>
<td>365</td>
</tr>
</tbody>
</table>

## 9 C Formatted Input/Output

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>Introduction</td>
<td>378</td>
</tr>
<tr>
<td>9.2</td>
<td>Streams</td>
<td>378</td>
</tr>
<tr>
<td>9.3</td>
<td>Formatting Output with <code>printf</code></td>
<td>378</td>
</tr>
<tr>
<td>9.4</td>
<td>Printing Integers</td>
<td>379</td>
</tr>
<tr>
<td>9.5</td>
<td>Printing Floating-Point Numbers</td>
<td>380</td>
</tr>
<tr>
<td>9.5.1</td>
<td>Conversion Specifiers <code>e</code>, <code>E</code> and <code>f</code></td>
<td>381</td>
</tr>
<tr>
<td>9.5.2</td>
<td>Conversion Specifiers <code>g</code> and <code>G</code></td>
<td>381</td>
</tr>
<tr>
<td>9.5.3</td>
<td>Demonstrating Floating-Point Conversion Specifiers</td>
<td>382</td>
</tr>
<tr>
<td>9.6</td>
<td>Printing Strings and Characters</td>
<td>382</td>
</tr>
<tr>
<td>9.7</td>
<td>Other Conversion Specifiers</td>
<td>383</td>
</tr>
<tr>
<td>9.8</td>
<td>Printing with Field Widths and Precision</td>
<td>384</td>
</tr>
<tr>
<td>9.8.1</td>
<td>Specifying Field Widths for Printing Integers</td>
<td>384</td>
</tr>
<tr>
<td>9.8.2</td>
<td>Specifying Precisions for Integers, Floating-Point Numbers and Strings</td>
<td>385</td>
</tr>
<tr>
<td>9.8.3</td>
<td>Combining Field Widths and Precisions</td>
<td>386</td>
</tr>
<tr>
<td>9.9</td>
<td>Using Flags in the <code>printf</code> Format Control String</td>
<td>387</td>
</tr>
<tr>
<td>9.9.1</td>
<td>Right and Left Justification</td>
<td>387</td>
</tr>
<tr>
<td>9.9.2</td>
<td>Printing Positive and Negative Numbers with and without the <code>+</code> Flag</td>
<td>388</td>
</tr>
<tr>
<td>9.9.3</td>
<td>Using the Space Flag</td>
<td>388</td>
</tr>
<tr>
<td>9.9.4</td>
<td>Using the <code>#</code> Flag</td>
<td>389</td>
</tr>
<tr>
<td>9.9.5</td>
<td>Using the <code>0</code> Flag</td>
<td>389</td>
</tr>
<tr>
<td>9.10</td>
<td>Printing Literals and Escape Sequences</td>
<td>390</td>
</tr>
<tr>
<td>9.11</td>
<td>Reading Formatted Input with <code>scanf</code></td>
<td>390</td>
</tr>
<tr>
<td>9.11.1</td>
<td><code>scanf</code> Syntax</td>
<td>391</td>
</tr>
<tr>
<td>9.11.2</td>
<td><code>scanf</code> Conversion Specifiers</td>
<td>391</td>
</tr>
<tr>
<td>9.11.3</td>
<td>Reading Integers with <code>scanf</code></td>
<td>392</td>
</tr>
<tr>
<td>9.11.4</td>
<td>Reading Floating-Point Numbers with <code>scanf</code></td>
<td>393</td>
</tr>
<tr>
<td>9.11.5</td>
<td>Reading Characters and Strings with <code>scanf</code></td>
<td>393</td>
</tr>
<tr>
<td>9.11.6</td>
<td>Using Scan Sets with <code>scanf</code></td>
<td>394</td>
</tr>
</tbody>
</table>
9.11.7 Using Field Widths with scanf 395
9.11.8 Skipping Characters in an Input Stream 396
9.12 Secure C Programming 397

10 C Structures, Unions, Bit Manipulation and Enumerations 404
10.1 Introduction 405
10.2 Structure Definitions 405
10.2.1 Self-Referential Structures 406
10.2.2 Defining Variables of Structure Types 407
10.2.3 Structure Tag Names 407
10.2.4 Operations That Can Be Performed on Structures 407
10.3 Initializing Structures 408
10.4 Accessing Structure Members with . and -> 408
10.5 Using Structures with Functions 410
10.6 typedef 411
10.7 Example: High-Performance Card Shuffling and Dealing Simulation 411
10.8 Unions 414
10.8.1 Union Declarations 414
10.8.2 Operations That Can Be Performed on Unions 415
10.8.3 Initializing Unions in Declarations 415
10.8.4 Demonstrating Unions 415
10.9 Bitwise Operators 416
10.9.1 Displaying an Unsigned Integer in Bits 417
10.9.2 Making Function displayBits More Generic and Portable 419
10.9.3 Using the Bitwise AND, Inclusive OR, Exclusive OR and Complement Operators 420
10.9.4 Using the Bitwise Left- and Right-Shift Operators 423
10.9.5 Bitwise Assignment Operators 424
10.10 Bit Fields 425
10.10.1 Defining Bit Fields 425
10.10.2 Using Bit Fields to Represent a Card’s Face, Suit and Color 426
10.10.3 Unnamed Bit Fields 428
10.11 Enumeration Constants 428
10.12 Anonymous Structures and Unions 430
10.13 Secure C Programming 430

11 C File Processing 441
11.1 Introduction 442
11.2 Files and Streams 442
11.3 Creating a Sequential-Access File 443
11.3.1 Pointer to a FILE 445
11.3.2 Using fopen to Open the File 445
11.3.3 Usingfeof to Check for the End-of-File Indicator 445

## Contents

11.3.4 Using fprintf to Write to the File  446  
11.3.5 Using fclose to Close the File  446  
11.3.6 File Open Modes  447  
11.4 Reading Data from a Sequential-Access File  449  
11.4.1 Resetting the File Position Pointer  450  
11.4.2 Credit Inquiry Program  450  
11.5 Random-Access Files  454  
11.6 Creating a Random-Access File  454  
11.7 Writing Data Randomly to a Random-Access File  456  
11.7.1 Positioning the File Position Pointer with fseek  458  
11.7.2 Error Checking  459  
11.8 Reading Data from a Random-Access File  459  
11.9 Case Study: Transaction-Processing Program  461  
11.10 Secure C Programming  466  

### 12 C Data Structures 477

12.1 Introduction  478  
12.2 Self-Referential Structures  479  
12.3 Dynamic Memory Allocation  479  
12.4 Linked Lists  480  
12.4.1 Function insert  486  
12.4.2 Function delete  487  
12.4.3 Function printlist  489  
12.5 Stacks  489  
12.5.1 Function push  493  
12.5.2 Function pop  494  
12.5.3 Applications of Stacks  494  
12.6 Queues  495  
12.6.1 Function enqueue  499  
12.6.2 Function dequeue  500  
12.7 Trees  501  
12.7.1 Function insertNode  504  
12.7.2 Traversals: Functions inorder, preorder and postorder  505  
12.7.3 Duplicate Elimination  506  
12.7.4 Binary Tree Search  506  
12.7.5 Other Binary Tree Operations  506  
12.8 Secure C Programming  506  

### 13 C Preprocessor 518

13.1 Introduction  519  
13.2 #include Preprocessor Directive  519  
13.3 #define Preprocessor Directive: Symbolic Constants  520  
13.4 #define Preprocessor Directive: Macros  521  
13.4.1 Macro with One Argument  521
13.4.2 Macro with Two Arguments  522
13.4.3 Macro Continuation Character  522
13.4.4 #undef Preprocessor Directive  522
13.4.5 Standard Library Functions and Macros  522
13.4.6 Do Not Place Expressions with Side Effects in Macros  523
13.5 Conditional Compilation  523
13.5.1 #if…#endif Preprocessor Directive  523
13.5.2 Commenting Out Blocks of Code with #if…#endif  523
13.5.3 Conditionally Compiling Debugging Code  524
13.6 #error and #pragma Preprocessor Directives  524
13.7 # and ## Operators  524
13.8 Line Numbers  525
13.9 Predefined Symbolic Constants  525
13.10 Assertions  526
13.11 Secure C Programming  526

14 Other C Topics  531
14.1 Introduction  532
14.2 Redirecting I/O  532
14.2.1 Redirecting Input with <  532
14.2.2 Redirecting Input with |  533
14.2.3 Redirecting Output  533
14.3 Variable-Length Argument Lists  533
14.4 Using Command-Line Arguments  535
14.5 Compiling Multiple-Source-File Programs  537
14.5.1 extern Declarations for Global Variables in Other Files  537
14.5.2 Function Prototypes  537
14.5.3 Restricting Scope with static  538
14.5.4 Makefiles  538
14.6 Program Termination with exit and atexit  538
14.7 Suffixes for Integer and Floating-Point Literals  540
14.8 Signal Handling  540
14.9 Dynamic Memory Allocation: Functions calloc and realloc  543
14.10 Unconditional Branching with goto  543

15 C++ as a Better C; Introducing Object Technology  549
15.1 Introduction  550
15.2 C++  550
15.3 A Simple Program: Adding Two Integers  551
15.3.1 Addition Program in C++  551
15.3.2 <iostream> Header  552
15.3.3 main Function  552
15.3.4 Variable Declarations  552
## Contents

15.3.5 Standard Output Stream and Standard Input Stream Objects 552  
15.3.6 `std::endl` Stream Manipulator 553  
15.3.7 `std::` Explained 553  
15.3.8 Concatenated Stream Outputs 553  
15.3.9 `return` Statement Not Required in `main` 553  
15.3.10 Operator Overloading 553  
15.4 C++ Standard Library 554  
15.5 Header Files 554  
15.6 Inline Functions 556  
15.7 C++ Keywords 558  
15.8 References and Reference Parameters 559  
  15.8.1 Reference Parameters 559  
  15.8.2 Passing Arguments by Value and by Reference 560  
  15.8.3 References as Aliases within a Function 562  
  15.8.4 Returning a Reference from a Function 563  
  15.8.5 Error Messages for Uninitialized References 564  
15.9 Empty Parameter Lists 564  
15.10 Default Arguments 564  
15.11 Unary Scope Resolution Operator 566  
15.12 Function Overloading 567  
15.13 Function Templates 570  
  15.13.1 Defining a Function Template 570  
  15.13.2 Using a Function Template 571  
15.14 Introduction to Object Technology and the UML 573  
  15.14.1 Basic Object Technology Concepts 573  
  15.14.2 Classes, Data Members and Member Functions 574  
  15.14.3 Object-Oriented Analysis and Design 575  
  15.14.4 The Unified Modeling Language 576  
15.15 Introduction to C++ Standard Library Class Template `vector` 576  
  15.15.1 Problems Associated with C-Style Pointer-Based Arrays 576  
  15.15.2 Using Class Template `vector` 577  
  15.15.3 Exception Handling: Processing an Out-of-Range Index 581  
15.16 Wrap-Up 583  

### 16 Introduction to Classes, Objects and Strings 589  
16.1 Introduction 590  
16.2 Defining a Class with a Member Function 590  
16.3 Defining a Member Function with a Parameter 593  
16.4 Data Members, `set` Member Functions and `get` Member Functions 597  
16.5 Initializing Objects with Constructors 602  
16.6 Placing a Class in a Separate File for Reusability 606  
16.7 Separating Interface from Implementation 610  
16.8 Validating Data with `set` Functions 615  
16.9 Wrap-Up 620
17 Classes: A Deeper Look; Throwing Exceptions 627
17.1 Introduction 628
17.2 Time Class Case Study 629
17.3 Class Scope and Accessing Class Members 635
17.4 Access Functions and Utility Functions 636
17.5 Time Class Case Study: Constructors with Default Arguments 637
17.6 Destructors 643
17.7 When Constructors and Destructors Are Called 643
17.8 Time Class Case Study: A Subtle Trap—Returning a Reference or a Pointer to a private Data Member 647
17.9 Default Memberwise Assignment 650
17.10 const Objects and const Member Functions 652
17.11 Composition: Objects as Members of Classes 654
17.12 friend Functions and friend Classes 660
17.13 Using the this Pointer 662
17.14 static Class Members 668
17.15 Wrap-Up 673

18 Operator Overloading; Class string 683
18.1 Introduction 684
18.2 Using the Overloaded Operators of Standard Library Class string 685
18.3 Fundamentals of Operator Overloading 688
18.4 Overloading Binary Operators 689
18.5 Overloading the Binary Stream Insertion and Stream Extraction Operators 690
18.6 Overloading Unary Operators 694
18.7 Overloading the Unary Prefix and Postfix ++ and -- Operators 695
18.8 Case Study: A Date Class 696
18.9 Dynamic Memory Management 701
18.10 Case Study: Array Class 703
18.10.1 Using the Array Class 704
18.10.2 Array Class Definition 708
18.11 Operators as Member vs. Non-Member Functions 716
18.12 Converting Between Types 716
18.13 explicit Constructors and Conversion Operators 718
18.14 Overloading the Function Call Operator () 720
18.15 Wrap-Up 721

19 Object-Oriented Programming: Inheritance 732
19.1 Introduction 733
19.2 Base Classes and Derived Classes 733
19.3 Relationship between Base and Derived Classes 736
19.3.1 Creating and Using a CommissionEmployee Class 736
19.3.2 Creating a BasePlusCommissionEmployee Class Without Using Inheritance 741
20 Object-Oriented Programming: Polymorphism 767
20.1 Introduction 768
20.2 Introduction to Polymorphism: Polymorphic Video Game 769
20.3 Relationships Among Objects in an Inheritance Hierarchy 769
  20.3.1 Invoking Base-Class Functions from Derived-Class Objects 770
  20.3.2 Aiming Derived-Class Pointers at Base-Class Objects 773
  20.3.3 Derived-Class Member-Function Calls via Base-Class Pointers 774
  20.3.4 Virtual Functions and Virtual Destructors 776
20.4 Type Fields and switch Statements 783
20.5 Abstract Classes and Pure virtual Functions 783
20.6 Case Study: Payroll System Using Polymorphism 785
  20.6.1 Creating Abstract Base Class Employee 786
  20.6.2 Creating Concrete Derived Class SalariedEmployee 790
  20.6.3 Creating Concrete Derived Class CommissionEmployee 792
  20.6.4 Creating Indirect Concrete Derived Class BasePlusCommissionEmployee 794
  20.6.5 Demonstrating Polymorphic Processing “Under the Hood” 796
20.7 (Optional) Polymorphism, Virtual Functions and Dynamic Binding 800
20.8 Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, dynamic_cast, typeid and type_info 803
20.9 Wrap-Up 807

21 Stream Input/Output: A Deeper Look 812
21.1 Introduction 813
21.2 Streams 814
  21.2.1 Classic Streams vs. Standard Streams 814
  21.2.2 istream Library Headers 815
  21.2.3 Stream Input/Output Classes and Objects 815
21.3 Stream Output 817
  21.3.1 Output of char * Variables 818
  21.3.2 Character Output Using Member Function put 818
21.4 Stream Input 819
  21.4.1 get and getline Member Functions 819
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.4.2</td>
<td>istream Member Functions peek, putback and ignore</td>
<td>822</td>
</tr>
<tr>
<td>21.4.3</td>
<td>Type-Safe I/O</td>
<td>822</td>
</tr>
<tr>
<td>21.5</td>
<td>Unformatted I/O Using read, write and gcount</td>
<td>822</td>
</tr>
<tr>
<td>21.6</td>
<td>Introduction to Stream Manipulators</td>
<td>823</td>
</tr>
<tr>
<td>21.6.1</td>
<td>Integral Stream Base: dec, oct, hex and setbase</td>
<td>824</td>
</tr>
<tr>
<td>21.6.2</td>
<td>Floating-Point Precision (precision, setprecision)</td>
<td>824</td>
</tr>
<tr>
<td>21.6.3</td>
<td>Field Width (width, setw)</td>
<td>826</td>
</tr>
<tr>
<td>21.6.4</td>
<td>User-Defined Output Stream Manipulators</td>
<td>827</td>
</tr>
<tr>
<td>21.7</td>
<td>Stream Format States and Stream Manipulators</td>
<td>828</td>
</tr>
<tr>
<td>21.7.1</td>
<td>Trailing Zeros and Decimal Points (showpoint)</td>
<td>829</td>
</tr>
<tr>
<td>21.7.2</td>
<td>Justification (left, right and internal)</td>
<td>830</td>
</tr>
<tr>
<td>21.7.3</td>
<td>Padding (fill, setfill)</td>
<td>832</td>
</tr>
<tr>
<td>21.7.4</td>
<td>Integral Stream Base (dec, oct, hex, showbase)</td>
<td>833</td>
</tr>
<tr>
<td>21.7.5</td>
<td>Floating-Point Numbers; Scientific and Fixed Notation (scientific, fixed)</td>
<td>834</td>
</tr>
<tr>
<td>21.7.6</td>
<td>Uppercase/Lowercase Control (uppercase)</td>
<td>835</td>
</tr>
<tr>
<td>21.7.7</td>
<td>Specifying Boolean Format (boolalpha)</td>
<td>835</td>
</tr>
<tr>
<td>21.7.8</td>
<td>Setting and Resetting the Format State via Member Function flags</td>
<td>836</td>
</tr>
<tr>
<td>21.8</td>
<td>Stream Error States</td>
<td>837</td>
</tr>
<tr>
<td>21.9</td>
<td>Tying an Output Stream to an Input Stream</td>
<td>840</td>
</tr>
<tr>
<td>21.10</td>
<td>Wrap-Up</td>
<td>840</td>
</tr>
</tbody>
</table>

### 22 Exception Handling: A Deeper Look | 849 |
| 22.1 | Introduction | 850 |
| 22.2 | Example: Handling an Attempt to Divide by Zero | 850 |
| 22.3 | Rethrowing an Exception | 856 |
| 22.4 | Stack Unwinding | 857 |
| 22.5 | When to Use Exception Handling | 859 |
| 22.6 | Constructors, Destructors and Exception Handling | 860 |
| 22.7 | Exceptions and Inheritance | 861 |
| 22.8 | Processing new Failures | 861 |
| 22.9 | Class unique_ptr and Dynamic Memory Allocation | 864 |
| 22.10 | Standard Library Exception Hierarchy | 867 |
| 22.11 | Wrap-Up | 868 |

### 23 Introduction to Custom Templates | 874 |
| 23.1 | Introduction | 875 |
| 23.2 | Class Templates | 875 |
| 23.3 | Function Template to Manipulate a Class-Template Specialization Object | 880 |
| 23.4 | Nontype Parameters | 882 |
| 23.5 | Default Arguments for Template Type Parameters | 882 |
| 23.6 | Overloading Function Templates | 883 |
| 23.7 | Wrap-Up | 883 |

Contents

A  C and C++ Operator Precedence Charts  886

B  ASCII Character Set  890

C  Number Systems  891
   C.1  Introduction  892
   C.2  Abbreviating Binary Numbers as Octal and Hexadecimal Numbers  895
   C.3  Converting Octal and Hexadecimal Numbers to Binary Numbers  896
   C.4  Converting from Binary, Octal or Hexadecimal to Decimal  896
   C.5  Converting from Decimal to Binary, Octal or Hexadecimal  897
   C.6  Negative Binary Numbers: Two’s Complement Notation  899

D  Sorting: A Deeper Look  904
   D.1  Introduction  905
   D.2  Big O Notation  905
   D.3  Selection Sort  906
   D.4  Insertion Sort  910
   D.5  Merge Sort  913

E  Multithreading and Other C11 and C99 Topics  924
   E.1  Introduction  925
   E.2  New C99 Headers  926
   E.3  Designated Initializers and Compound Literals  927
   E.4  Type bool  929
   E.5  Implicit int in Function Declarations  931
   E.6  Complex Numbers  932
   E.7  Additions to the Preprocessor  933
   E.8  Other C99 Features  934
      E.8.1  Compiler Minimum Resource Limits  934
      E.8.2  The restrict Keyword  935
      E.8.3  Reliable Integer Division  935
      E.8.4  Flexible Array Members  935
      E.8.5  Relaxed Constraints on Aggregate Initialization  936
      E.8.6  Type Generic Math  936
      E.8.7  Inline Functions  936
      E.8.8  Return Without Expression  937
      E.8.9  __func__ Predefined Identifier  937
      E.8.10  va_copy Macro  937
   E.9  New Features in the C11 Standard  937
      E.9.1  New C11 Headers  938
      E.9.2  Multithreading Support  938
      E.9.3  quick_exit function  946

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.9.4 Unicode® Support</td>
<td>946</td>
</tr>
<tr>
<td>E.9.5 _Noreturn Function Specifier</td>
<td>946</td>
</tr>
<tr>
<td>E.9.6 Type-Generic Expressions</td>
<td>946</td>
</tr>
<tr>
<td>E.9.7 Annex L: Analyzability and Undefined Behavior</td>
<td>947</td>
</tr>
<tr>
<td>E.9.8 Memory Alignment Control</td>
<td>947</td>
</tr>
<tr>
<td>E.9.9 Static Assertions</td>
<td>947</td>
</tr>
<tr>
<td>E.9.10 Floating-Point Types</td>
<td>948</td>
</tr>
<tr>
<td>E.10 Web Resources</td>
<td>948</td>
</tr>
</tbody>
</table>

### Appendices on the Web

#### F Using the Visual Studio Debugger

#### G Using the GNU gdb Debugger

#### H Using the Xcode Debugger

Appendices F, G and H are PDF documents posted online at the book’s Companion Website (located at www.pearsonhighered.com/deitel).