

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

Preface **19**

I Introduction to Computers, the Internet and the Web **29**

1.1	Introduction	30
1.2	Computers and the Internet in Industry and Research	30
1.3	Hardware and Software	33
1.3.1	Moore's Law	34
1.3.2	Computer Organization	34
1.4	Data Hierarchy	35
1.5	Programming Languages	37
1.6	The C Programming Language	38
1.7	C Standard Library	40
1.8	C++ and Other C-Based Languages	41
1.9	Object Technology	42
1.10	Typical C Program Development Environment	44
1.10.1	Phase 1: Creating a Program	44
1.10.2	Phases 2 and 3: Preprocessing and Compiling a C Program	44
1.10.3	Phase 4: Linking	46
1.10.4	Phase 5: Loading	46
1.10.5	Phase 6: Execution	46
1.10.6	Problems That May Occur at Execution Time	46
1.10.7	Standard Input, Standard Output and Standard Error Streams	46
1.11	Test-Driving a C Application in Windows, Linux and Mac OS X	47
1.11.1	Running a C Application from the Windows Command Prompt	48
1.11.2	Running a C Application Using GNU C with Linux	50
1.11.3	Running a C Application Using GNU C with Mac OS X	53
1.12	Operating Systems	55
1.12.1	Windows—A Proprietary Operating System	56
1.12.2	Linux—An Open-Source Operating System	56
1.12.3	Apple's Mac OS X; Apple's iOS for iPhone®, iPad® and iPod Touch® Devices	57
1.12.4	Google's Android	57

1.13	The Internet and World Wide Web	58
1.14	Some Key Software Development Terminology	59
1.15	Keeping Up-to-Date with Information Technologies	61
1.16	Web Resources	62

2 Introduction to C Programming 68

2.1	Introduction	69
2.2	A Simple C Program: Printing a Line of Text	69
2.3	Another Simple C Program: Adding Two Integers	73
2.4	Memory Concepts	77
2.5	Arithmetic in C	78
2.6	Decision Making: Equality and Relational Operators	82
2.7	Secure C Programming	86

3 Structured Program Development in C 98

3.1	Introduction	99
3.2	Algorithms	99
3.3	Pseudocode	99
3.4	Control Structures	100
3.5	The <code>if</code> Selection Statement	102
3.6	The <code>if...else</code> Selection Statement	103
3.7	The <code>while</code> Repetition Statement	107
3.8	Formulating Algorithms Case Study 1: Counter-Controlled Repetition	108
3.9	Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 2: Sentinel-Controlled Repetition	110
3.10	Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 3: Nested Control Statements	117
3.11	Assignment Operators	121
3.12	Increment and Decrement Operators	121
3.13	Secure C Programming	124

4 C Program Control 142

4.1	Introduction	143
4.2	Repetition Essentials	143
4.3	Counter-Controlled Repetition	144
4.4	<code>for</code> Repetition Statement	145
4.5	<code>for</code> Statement: Notes and Observations	148
4.6	Examples Using the <code>for</code> Statement	149
4.7	<code>switch</code> Multiple-Selection Statement	152
4.8	<code>do...while</code> Repetition Statement	158
4.9	<code>break</code> and <code>continue</code> Statements	160
4.10	Logical Operators	162
4.11	Confusing Equality (<code>==</code>) and Assignment (<code>=</code>) Operators	165

4.12	Structured Programming Summary	166
4.13	Secure C Programming	171

5 C Functions 186

5.1	Introduction	187
5.2	Program Modules in C	187
5.3	Math Library Functions	188
5.4	Functions	190
5.5	Function Definitions	190
5.6	Function Prototypes: A Deeper Look	194
5.7	Function Call Stack and Stack Frames	197
5.8	Headers	200
5.9	Passing Arguments By Value and By Reference	201
5.10	Random Number Generation	202
5.11	Example: A Game of Chance	207
5.12	Storage Classes	210
5.13	Scope Rules	212
5.14	Recursion	215
5.15	Example Using Recursion: Fibonacci Series	219
5.16	Recursion vs. Iteration	222
5.17	Secure C Programming	225

6 C Arrays 244

6.1	Introduction	245
6.2	Arrays	245
6.3	Defining Arrays	246
6.4	Array Examples	247
6.5	Passing Arrays to Functions	260
6.6	Sorting Arrays	264
6.7	Case Study: Computing Mean, Median and Mode Using Arrays	267
6.8	Searching Arrays	272
6.9	Multidimensional Arrays	277
6.10	Variable-Length Arrays	284
6.11	Secure C Programming	287

7 C Pointers 305

7.1	Introduction	306
7.2	Pointer Variable Definitions and Initialization	306
7.3	Pointer Operators	307
7.4	Passing Arguments to Functions by Reference	310
7.5	Using the const Qualifier with Pointers	312
7.5.1	Converting a String to Uppercase Using a Non-Constant Pointer to Non-Constant Data	315

7.5.2	Printing a String One Character at a Time Using a Non-Constant Pointer to Constant Data	316
7.5.3	Attempting to Modify a Constant Pointer to Non-Constant Data	318
7.5.4	Attempting to Modify a Constant Pointer to Constant Data	319
7.6	Bubble Sort Using Pass-by-Reference	319
7.7	sizeof Operator	322
7.8	Pointer Expressions and Pointer Arithmetic	325
7.9	Relationship between Pointers and Arrays	327
7.10	Arrays of Pointers	331
7.11	Case Study: Card Shuffling and Dealing Simulation	332
7.12	Pointers to Functions	337
7.13	Secure C Programming	342

8 C Characters and Strings 362

8.1	Introduction	363
8.2	Fundamentals of Strings and Characters	363
8.3	Character-Handling Library	365
8.3.1	Functions isdigit, isalpha, isalnum and isxdigit	366
8.3.2	Functions islower, isupper, tolower and toupper	368
8.3.3	Functions isspace, iscntrl, ispunct, isprint and isgraph	369
8.4	String-Conversion Functions	370
8.4.1	Function strtod	371
8.4.2	Function strtol	372
8.4.3	Function strtoul	373
8.5	Standard Input/Output Library Functions	374
8.5.1	Functions fgets and putchar	374
8.5.2	Function getchar	376
8.5.3	Function sprintf	377
8.5.4	Function sscanf	377
8.6	String-Manipulation Functions of the String-Handling Library	378
8.6.1	Functions strcpy and strncpy	379
8.6.2	Functions strcat and strncat	380
8.7	Comparison Functions of the String-Handling Library	381
8.8	Search Functions of the String-Handling Library	382
8.8.1	Function strchr	383
8.8.2	Function strcspn	384
8.8.3	Function strpbrk	385
8.8.4	Function strrchr	385
8.8.5	Function strspn	386
8.8.6	Function strstr	386
8.8.7	Function strtok	387
8.9	Memory Functions of the String-Handling Library	388
8.9.1	Function memcpy	389
8.9.2	Function memmove	390
8.9.3	Function memcmp	391

8.9.4	Function <code>memchr</code>	391
8.9.5	Function <code>memset</code>	392
8.10	Other Functions of the String-Handling Library	393
8.10.1	Function <code>strerror</code>	393
8.10.2	Function <code>strlen</code>	393
8.11	Secure C Programming	394

9 C Formatted Input/Output 407

9.1	Introduction	408
9.2	Streams	408
9.3	Formatting Output with <code>printf</code>	408
9.4	Printing Integers	409
9.5	Printing Floating-Point Numbers	410
9.6	Printing Strings and Characters	412
9.7	Other Conversion Specifiers	413
9.8	Printing with Field Widths and Precision	414
9.9	Using Flags in the <code>printf</code> Format Control String	416
9.10	Printing Literals and Escape Sequences	419
9.11	Reading Formatted Input with <code>scanf</code>	419
9.12	Secure C Programming	426

10 C Structures, Unions, Bit Manipulation and Enumerations 433

10.1	Introduction	434
10.2	Structure Definitions	434
10.2.1	Self-Referential Structures	435
10.2.2	Defining Variables of Structure Types	435
10.2.3	Structure Tag Names	436
10.2.4	Operations That Can Be Performed on Structures	436
10.3	Initializing Structures	437
10.4	Accessing Structure Members	437
10.5	Using Structures with Functions	439
10.6	<code>typedef</code>	439
10.7	Example: High-Performance Card Shuffling and Dealing Simulation	440
10.8	Unions	443
10.8.1	Union Declarations	443
10.8.2	Operations That Can Be Performed on Unions	443
10.8.3	Initializing Unions in Declarations	444
10.8.4	Demonstrating Unions	444
10.9	Bitwise Operators	445
10.9.1	Displaying an Unsigned Integer in Bits	446
10.9.2	Making Function <code>displayBits</code> More Scalable and Portable	448
10.9.3	Using the Bitwise AND, Inclusive OR, Exclusive OR and Complement Operators	448

10.9.4	Using the Bitwise Left- and Right-Shift Operators	451
10.9.5	Bitwise Assignment Operators	453
10.10	Bit Fields	454
10.11	Enumeration Constants	457
10.12	Secure C Programming	459

11 C File Processing 469

11.1	Introduction	470
11.2	Files and Streams	470
11.3	Creating a Sequential-Access File	471
11.4	Reading Data from a Sequential-Access File	476
11.5	Random-Access Files	480
11.6	Creating a Random-Access File	481
11.7	Writing Data Randomly to a Random-Access File	483
11.8	Reading Data from a Random-Access File	486
11.9	Case Study: Transaction-Processing Program	487
11.10	Secure C Programming	493

12 C Data Structures 504

12.1	Introduction	505
12.2	Self-Referential Structures	506
12.3	Dynamic Memory Allocation	506
12.4	Linked Lists	507
12.4.1	Function insert	513
12.4.2	Function delete	515
12.4.3	Function printList	516
12.5	Stacks	516
12.5.1	Function push	520
12.5.2	Function pop	520
12.5.3	Applications of Stacks	521
12.6	Queues	522
12.6.1	Function enqueue	526
12.6.2	Function dequeue	527
12.7	Trees	528
12.7.1	Function insertNode	532
12.7.2	Traversals: Functions inOrder, preOrder and postOrder	532
12.7.3	Duplicate Elimination	533
12.7.4	Binary Tree Search	533
12.7.5	Other Binary Tree Operations	533
12.8	Secure C Programming	534

13 C Preprocessor 545

13.1	Introduction	546
13.2	#include Preprocessor Directive	546

13.3	<code>#define</code> Preprocessor Directive: Symbolic Constants	547
13.4	<code>#define</code> Preprocessor Directive: Macros	547
13.5	Conditional Compilation	549
13.6	<code>#error</code> and <code>#pragma</code> Preprocessor Directives	550
13.7	<code>#</code> and <code>##</code> Operators	551
13.8	Line Numbers	551
13.9	Predefined Symbolic Constants	551
13.10	Assertions	552
13.11	Secure C Programming	552

14 Other C Topics 557

14.1	Introduction	558
14.2	Redirecting I/O	558
14.3	Variable-Length Argument Lists	559
14.4	Using Command-Line Arguments	561
14.5	Notes on Compiling Multiple-Source-File Programs	562
14.6	Program Termination with <code>exit</code> and <code>atexit</code>	564
14.7	Suffixes for Integer and Floating-Point Literals	565
14.8	Signal Handling	566
14.9	Dynamic Memory Allocation: Functions <code>calloc</code> and <code>realloc</code>	568
14.10	Unconditional Branching with <code>goto</code>	569

15 C++ as a Better C; Introducing Object Technology 575

15.1	Introduction	576
15.2	C++	576
15.3	A Simple Program: Adding Two Integers	577
15.4	C++ Standard Library	579
15.5	Header Files	580
15.6	Inline Functions	582
15.7	References and Reference Parameters	584
15.8	Empty Parameter Lists	589
15.9	Default Arguments	589
15.10	Unary Scope Resolution Operator	591
15.11	Function Overloading	592
15.12	Function Templates	595
15.13	Introduction to C++ Standard Library Class Template <code>vector</code>	598
15.14	Introduction to Object Technology and the UML	604
15.15	Wrap-Up	607

16 Introduction to Classes, Objects and Strings 614

16.1	Introduction	615
16.2	Defining a Class with a Member Function	615

17.1	Introduction	652
17.2	Time Class Case Study	653
17.3	Class Scope and Accessing Class Members	660
17.4	Separating Interface from Implementation	661
17.5	Access Functions and Utility Functions	662
17.6	Time Class Case Study: Constructors with Default Arguments	665
17.7	Destructors	670
17.8	When Constructors and Destructors Are Called	671
17.9	Time Class Case Study: A Subtle Trap—Returning a Reference to a private Data Member	674
17.10	Default Memberwise Assignment	677
17.11	Wrap-Up	680

18.1	Introduction	687
18.2	const (Constant) Objects and const Member Functions	687
18.3	Composition: Objects as Members of Classes	695
18.4	friend Functions and friend Classes	701
18.5	Using the this Pointer	703
18.6	static Class Members	708
18.7	Proxy Classes	713
18.8	Wrap-Up	717

19.1	Introduction	724
19.2	Using the Overloaded Operators of Standard Library Class <code>string</code>	725
19.3	Fundamentals of Operator Overloading	728
19.4	Overloading Binary Operators	729
19.5	Overloading the Binary Stream Insertion and Stream Extraction Operators	730
19.6	Overloading Unary Operators	734
19.7	Overloading the Unary Prefix and Postfix <code>++</code> and <code>--</code> Operators	735
19.8	Case Study: A Date Class	736
19.9	Dynamic Memory Management	741

19.10	Case Study: Array Class	743
19.10.1	Using the Array Class	744
19.10.2	Array Class Definition	747
19.11	Operators as Member Functions vs. Non-Member Functions	755
19.12	Converting between Types	755
19.13	explicit Constructors	757
19.14	Building a String Class	759
19.15	Wrap-Up	760

20 Object-Oriented Programming: Inheritance 771

20.1	Introduction	772
20.2	Base Classes and Derived Classes	772
20.3	protected Members	775
20.4	Relationship between Base Classes and Derived Classes	775
20.4.1	Creating and Using a <code>CommissionEmployee</code> Class	776
20.4.2	Creating a <code>BasePlusCommissionEmployee</code> Class Without Using Inheritance	780
20.4.3	Creating a <code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy	786
20.4.4	<code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy Using protected Data	791
20.4.5	<code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy Using private Data	794
20.5	Constructors and Destructors in Derived Classes	799
20.6	public, protected and private Inheritance	799
20.7	Software Engineering with Inheritance	800
20.8	Wrap-Up	801

21 Object-Oriented Programming: Polymorphism 806

21.1	Introduction	807
21.2	Introduction to Polymorphism: Polymorphic Video Game	808
21.3	Relationships Among Objects in an Inheritance Hierarchy	808
21.3.1	Invoking Base-Class Functions from Derived-Class Objects	809
21.3.2	Aiming Derived-Class Pointers at Base-Class Objects	812
21.3.3	Derived-Class Member-Function Calls via Base-Class Pointers	813
21.3.4	Virtual Functions	815
21.4	Type Fields and switch Statements	821
21.5	Abstract Classes and Pure virtual Functions	821
21.6	Case Study: Payroll System Using Polymorphism	823
21.6.1	Creating Abstract Base Class <code>Employee</code>	824
21.6.2	Creating Concrete Derived Class <code>SalariedEmployee</code>	828
21.6.3	Creating Concrete Derived Class <code>CommissionEmployee</code>	830
21.6.4	Creating Indirect Concrete Derived Class <code>BasePlusCommissionEmployee</code>	832
21.6.5	Demonstrating Polymorphic Processing	834

21.7	(Optional) Polymorphism, Virtual Functions and Dynamic Binding “Under the Hood”	838
21.8	Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, <code>dynamic_cast</code> , <code>typeid</code> and <code>type_info</code>	841
21.9	Virtual Destructors	845
21.10	Wrap-Up	845

22 Templates **851**

22.1	Introduction	852
22.2	Function Templates	852
22.3	Overloading Function Templates	855
22.4	Class Templates	856
22.5	Nontype Parameters and Default Types for Class Templates	862
22.6	Wrap-Up	863

23 Stream Input/Output **867**

23.1	Introduction	868
23.2	Streams	869
23.2.1	Classic Streams vs. Standard Streams	869
23.2.2	<code>iostream</code> Library Headers	870
23.2.3	Stream Input/Output Classes and Objects	870
23.3	Stream Output	873
23.3.1	Output of <code>char *</code> Variables	873
23.3.2	Character Output Using Member Function <code>put</code>	873
23.4	Stream Input	874
23.4.1	<code>get</code> and <code>getline</code> Member Functions	874
23.4.2	<code>istream</code> Member Functions <code>peek</code> , <code>putback</code> and <code>ignore</code>	877
23.4.3	Type-Safe I/O	877
23.5	Unformatted I/O Using <code>read</code> , <code>write</code> and <code>gcount</code>	877
23.6	Introduction to Stream Manipulators	878
23.6.1	Integral Stream Base: <code>dec</code> , <code>oct</code> , <code>hex</code> and <code>setbase</code>	879
23.6.2	Floating-Point Precision (<code>precision</code> , <code>setprecision</code>)	879
23.6.3	Field Width (<code>width</code> , <code>setw</code>)	881
23.6.4	User-Defined Output Stream Manipulators	882
23.7	Stream Format States and Stream Manipulators	884
23.7.1	Trailing Zeros and Decimal Points (<code>showpoint</code>)	884
23.7.2	Justification (<code>left</code> , <code>right</code> and <code>internal</code>)	885
23.7.3	Padding (<code>fill</code> , <code>setfill</code>)	887
23.7.4	Integral Stream Base (<code>dec</code> , <code>oct</code> , <code>hex</code> , <code>showbase</code>)	888
23.7.5	Floating-Point Numbers; Scientific and Fixed Notation (<code>scientific</code> , <code>fixed</code>)	889
23.7.6	Uppercase/Lowercase Control (<code>uppercase</code>)	890
23.7.7	Specifying Boolean Format (<code>boolalpha</code>)	890
23.7.8	Setting and Resetting the Format State via Member Function flags	891

23.8	Stream Error States	892
23.9	Tying an Output Stream to an Input Stream	894
23.10	Wrap-Up	895

24 Exception Handling: A Deeper Look 904

24.1	Introduction	905
24.2	Example: Handling an Attempt to Divide by Zero	905
24.3	When to Use Exception Handling	911
24.4	Rethrowing an Exception	912
24.5	Processing Unexpected Exceptions	913
24.6	Stack Unwinding	914
24.7	Constructors, Destructors and Exception Handling	916
24.8	Exceptions and Inheritance	916
24.9	Processing new Failures	917
24.10	Class <code>unique_ptr</code> and Dynamic Memory Allocation	920
24.11	Standard Library Exception Hierarchy	922
24.12	Wrap-Up	924

A Operator Precedence Charts 930

B ASCII Character Set 934

C Number Systems 935

C.1	Introduction	936
C.2	Abbreviating Binary Numbers as Octal and Hexadecimal Numbers	939
C.3	Converting Octal and Hexadecimal Numbers to Binary Numbers	940
C.4	Converting from Binary, Octal or Hexadecimal to Decimal	940
C.5	Converting from Decimal to Binary, Octal or Hexadecimal	941
C.6	Negative Binary Numbers: Two's Complement Notation	943

D Game Programming: Solving Sudoku 948

D.1	Introduction	948
D.2	Deitel Sudoku Resource Center	949
D.3	Solution Strategies	949
D.4	Programming Sudoku Puzzle Solvers	953
D.5	Generating New Sudoku Puzzles	954
D.6	Conclusion	956

Appendices on the Web 957

Index 958

Appendices E through H are PDF documents posted online at the book's Companion Website (located at www.pearsoninternationaleditions.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