



DEITEL®

# HOW TO PROGRAM

NINTH  
EDITION  
GLOBAL  
EDITION

*with*  
Case Studies Introducing

**Applications  
Programming** and

**Systems  
Programming**

PAUL DEITEL  
HARVEY DEITEL

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on the appropriate page of appearance or in the Credits on pages.

Cover image by Ink Drop/ Shutterstock

Pearson Education Limited  
KAO Two  
KAO Park  
Hockham Way  
Harlow  
Essex  
CM17 9SR  
United Kingdom

and Associated Companies throughout the world

Visit us on the World Wide Web at: [www.pearsonglobaleditions.com](http://www.pearsonglobaleditions.com)

© Pearson Education Limited 2023

The rights of Paul Deitel and Harvey Deitel to be identified as the authors of this work, have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

*Authorized adaptation from the United States edition, entitled C How to Program, 9th Edition, ISBN 978-0-13-739839-3 by Paul Deitel and Harvey Deitel published by Pearson Education © 2022.*

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights and Permissions department, please visit [www.pearsoned.com/permissions/](http://www.pearsoned.com/permissions/).

This eBook is a standalone product and may or may not include all assets that were part of the print version. It also does not provide access to other Pearson digital products like MyLab and Mastering. The publisher reserves the right to remove any material in this eBook at any time.

**ISBN 10:** 1-292-43707-3 (print)

**ISBN 13:** 978-1-292-43707-1 (print)

**eBook ISBN 13:** 978-1-292-43699-9 (uPDF)

#### **British Library Cataloguing-in-Publication Data**

A catalogue record for this book is available from the British Library

1            22

Typeset in Times NR MT Pro by B2R Technologies Pvt. Ltd.

*In memory of Dennis Ritchie,  
creator of the C programming language  
and co-creator of the UNIX operating system.*

*Paul and Harvey Deitel*

## Trademarks

Apple, Xcode, Swift, Objective-C, iOS and macOS are trademarks or registered trademarks of Apple, Inc.

Java is a registered trademark of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds.

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided “as is” without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services.

The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time. Partial screen shots may be viewed in full within the software version specified.

Other names may be trademarks of their respective owners.

# Contents

Appendices E–H are PDF documents posted online at the book’s Companion Website (located at <https://www.pearsonglobal editions.com>).

<b>Preface</b>	<b>17</b>
<b>Before You Begin</b>	<b>49</b>
<b>I Introduction to Computers and C</b>	<b>53</b>
1.1 Introduction	54
1.2 Hardware and Software	56
1.2.1 Moore’s Law	56
1.2.2 Computer Organization	57
1.3 Data Hierarchy	60
1.4 Machine Languages, Assembly Languages and High-Level Languages	63
1.5 Operating Systems	65
1.6 The C Programming Language	68
1.7 The C Standard Library and Open-Source Libraries	70
1.8 Other Popular Programming Languages	71
1.9 Typical C Program-Development Environment	73
1.9.1 Phase 1: Creating a Program	73
1.9.2 Phases 2 and 3: Preprocessing and Compiling a C Program	73
1.9.3 Phase 4: Linking	74
1.9.4 Phase 5: Loading	75
1.9.5 Phase 6: Execution	75
1.9.6 Problems That May Occur at Execution Time	75
1.9.7 Standard Input, Standard Output and Standard Error Streams	76
1.10 Test-Driving a C Application in Windows, Linux and macOS	76
1.10.1 Compiling and Running a C Application with Visual Studio 2019 Community Edition on Windows 10	77
1.10.2 Compiling and Running a C Application with Xcode on macOS	81

1.10.3	Compiling and Running a C Application with GNU gcc on Linux	84
1.10.4	Compiling and Running a C Application in a GCC Docker Container Running Natively over Windows 10, macOS or Linux	86
1.11	Internet, World Wide Web, the Cloud and IoT	87
1.11.1	The Internet: A Network of Networks	88
1.11.2	The World Wide Web: Making the Internet User-Friendly	89
1.11.3	The Cloud	89
1.11.4	The Internet of Things	90
1.12	Software Technologies	91
1.13	How Big Is Big Data?	91
1.13.1	Big-Data Analytics	97
1.13.2	Data Science and Big Data Are Making a Difference: Use Cases	98
1.14	Case Study—A Big-Data Mobile Application	99
1.15	AI—at the Intersection of Computer Science and Data Science	100

## **2     Intro to C Programming** **107**

2.1	Introduction	108
2.2	A Simple C Program: Printing a Line of Text	108
2.3	Another Simple C Program: Adding Two Integers	112
2.4	Memory Concepts	116
2.5	Arithmetic in C	117
2.6	Decision Making: Equality and Relational Operators	121
2.7	Secure C Programming	125

## **3     Structured Program Development** **137**

3.1	Introduction	138
3.2	Algorithms	138
3.3	Pseudocode	139
3.4	Control Structures	140
3.5	The <code>if</code> Selection Statement	142
3.6	The <code>if...else</code> Selection Statement	144
3.7	The <code>while</code> Iteration Statement	148
3.8	Formulating Algorithms Case Study 1: Counter-Controlled Iteration	149
3.9	Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 2: Sentinel-Controlled Iteration	151
3.10	Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 3: Nested Control Statements	158
3.11	Assignment Operators	162
3.12	Increment and Decrement Operators	163
3.13	Secure C Programming	166

<b>4</b>	<b>Program Control</b>	<b>185</b>
4.1	Introduction	186
4.2	Iteration Essentials	186
4.3	Counter-Controlled Iteration	187
4.4	for Iteration Statement	188
4.5	Examples Using the for Statement	192
4.6	switch Multiple-Selection Statement	196
4.7	do...while Iteration Statement	202
4.8	break and continue Statements	203
4.9	Logical Operators	205
4.10	Confusing Equality (==) and Assignment (=) Operators	209
4.11	Structured-Programming Summary	210
4.12	Secure C Programming	215
<b>5</b>	<b>Functions</b>	<b>231</b>
5.1	Introduction	232
5.2	Modularizing Programs in C	232
5.3	Math Library Functions	234
5.4	Functions	235
5.5	Function Definitions	236
5.5.1	square Function	236
5.5.2	maximum Function	239
5.6	Function Prototypes: A Deeper Look	240
5.7	Function-Call Stack and Stack Frames	243
5.8	Headers	247
5.9	Passing Arguments by Value and by Reference	249
5.10	Random-Number Generation	249
5.11	<b>Game Simulation Case Study: Rock, Paper, Scissors</b>	254
5.12	Storage Classes	260
5.13	Scope Rules	262
5.14	Recursion	265
5.15	Example Using Recursion: Fibonacci Series	269
5.16	Recursion vs. Iteration	272
5.17	Secure C Programming—Secure Random-Number Generation	275
	<b>Random-Number Simulation Case Study: The Tortoise and the Hare</b>	294
<b>6</b>	<b>Arrays</b>	<b>297</b>
6.1	Introduction	298
6.2	Arrays	298
6.3	Defining Arrays	300
6.4	Array Examples	300

6.4.1	Defining an Array and Using a Loop to Set the Array's Element Values	301
6.4.2	Initializing an Array in a Definition with an Initializer List	302
6.4.3	Specifying an Array's Size with a Symbolic Constant and Initializing Array Elements with Calculations	303
6.4.4	Summing the Elements of an Array	304
6.4.5	Using Arrays to Summarize Survey Results	304
6.4.6	Graphing Array Element Values with Bar Charts	306
6.4.7	Rolling a Die 60,000,000 Times and Summarizing the Results in an Array	307
6.5	Using Character Arrays to Store and Manipulate Strings	309
6.5.1	Initializing a Character Array with a String	309
6.5.2	Initializing a Character Array with an Initializer List of Characters	309
6.5.3	Accessing the Characters in a String	309
6.5.4	Inputting into a Character Array	309
6.5.5	Outputting a Character Array That Represents a String	310
6.5.6	Demonstrating Character Arrays	310
6.6	Static Local Arrays and Automatic Local Arrays	312
6.7	Passing Arrays to Functions	314
6.8	Sorting Arrays	318
6.9	<b>Intro to Data Science Case Study: Survey Data Analysis</b>	321
6.10	Searching Arrays	326
6.10.1	Searching an Array with Linear Search	326
6.10.2	Searching an Array with Binary Search	328
6.11	Multidimensional Arrays	332
6.11.1	Illustrating a Two-Dimensional Array	332
6.11.2	Initializing a Double-Subscripted Array	333
6.11.3	Setting the Elements in One Row	335
6.11.4	Totaling the Elements in a Two-Dimensional Array	335
6.11.5	Two-Dimensional Array Manipulations	335
6.12	Variable-Length Arrays	339
6.13	Secure C Programming	343

## **7 Pointers 363**

7.1	Introduction	364
7.2	Pointer Variable Definitions and Initialization	365
7.3	Pointer Operators	366
7.4	Passing Arguments to Functions by Reference	369
7.5	Using the <code>const</code> Qualifier with Pointers	373
7.5.1	Converting a String to Uppercase Using a Non-Constant Pointer to Non-Constant Data	374



7.5.2	Printing a String One Character at a Time Using a Non-Constant Pointer to Constant Data	374
7.5.3	Attempting to Modify a Constant Pointer to Non-Constant Data	376
7.5.4	Attempting to Modify a Constant Pointer to Constant Data	377
7.6	Bubble Sort Using Pass-By-Reference	378
7.7	sizeof Operator	382
7.8	Pointer Expressions and Pointer Arithmetic	384
7.8.1	Pointer Arithmetic Operators	385
7.8.2	Aiming a Pointer at an Array	385
7.8.3	Adding an Integer to a Pointer	385
7.8.4	Subtracting an Integer from a Pointer	386
7.8.5	Incrementing and Decrementing a Pointer	386
7.8.6	Subtracting One Pointer from Another	386
7.8.7	Assigning Pointers to One Another	386
7.8.8	Pointer to void	386
7.8.9	Comparing Pointers	387
7.9	Relationship between Pointers and Arrays	387
7.9.1	Pointer/Offset Notation	387
7.9.2	Pointer/Subscript Notation	388
7.9.3	Cannot Modify an Array Name with Pointer Arithmetic	388
7.9.4	Demonstrating Pointer Subscripting and Offsets	388
7.9.5	String Copying with Arrays and Pointers	390
7.10	Arrays of Pointers	392
7.11	<b>Random-Number Simulation Case Study: Card Shuffling and Dealing</b>	393
7.12	Function Pointers	398
7.12.1	Sorting in Ascending or Descending Order	398
7.12.2	Using Function Pointers to Create a Menu-Driven System	401
7.13	Secure C Programming	403
	<b>Special Section: Building Your Own Computer as a Virtual Machine</b>	417
	<b>Special Section—Embedded Systems Programming Case Study: Robotics with the Webots Simulator</b>	424

## 8 Characters and Strings 441

8.1	Introduction	442
8.2	Fundamentals of Strings and Characters	442
8.3	Character-Handling Library	444
8.3.1	Functions isdigit, isalpha, isalnum and isxdigit	445
8.3.2	Functions islower, isupper, tolower and toupper	447
8.3.3	Functions isspace, iscntrl, ispunct, isprint and isgraph	448
8.4	String-Conversion Functions	450
8.4.1	Function strtod	450

8.4.2	Function <code>strtol</code>	451
8.4.3	Function <code>strtoul</code>	452
8.5	Standard Input/Output Library Functions	453
8.5.1	Functions <code>fgets</code> and <code>putchar</code>	453
8.5.2	Function <code>getchar</code>	455
8.5.3	Function <code>sprintf</code>	455
8.5.4	Function <code>sscanf</code>	456
8.6	String-Manipulation Functions of the String-Handling Library	457
8.6.1	Functions <code>strcpy</code> and <code>strncpy</code>	458
8.6.2	Functions <code>strcat</code> and <code>strncat</code>	459
8.7	Comparison Functions of the String-Handling Library	460
8.8	Search Functions of the String-Handling Library	462
8.8.1	Function <code>strchr</code>	463
8.8.2	Function <code>strcspn</code>	464
8.8.3	Function <code>strpbrk</code>	464
8.8.4	Function <code>strrchr</code>	465
8.8.5	Function <code>strspn</code>	465
8.8.6	Function <code>strstr</code>	466
8.8.7	Function <code>strtok</code>	467
8.9	Memory Functions of the String-Handling Library	468
8.9.1	Function <code>memcpy</code>	469
8.9.2	Function <code>memmove</code>	470
8.9.3	Function <code>memcmp</code>	470
8.9.4	Function <code>memchr</code>	471
8.9.5	Function <code>memset</code>	471
8.10	Other Functions of the String-Handling Library	473
8.10.1	Function <code>strerror</code>	473
8.10.2	Function <code>strlen</code>	473
8.11	Secure C Programming	474
	<b>Pqyoaf X Nylfomigrob Qwbbfmh Mndogvk: Rboqlrut yua</b>	
	<b>Boklnxhmywex</b>	488
	<b>Secure C Programming Case Study: Public-Key Cryptography</b>	494

## **9 Formatted Input/Output** **503**

9.1	Introduction	504
9.2	Streams	504
9.3	Formatting Output with <code>printf</code>	505
9.4	Printing Integers	506
9.5	Printing Floating-Point Numbers	507
9.5.1	Conversion Specifiers <code>e</code> , <code>E</code> and <code>f</code>	508
9.5.2	Conversion Specifiers <code>g</code> and <code>G</code>	508
9.5.3	Demonstrating Floating-Point Conversion Specifiers	509
9.6	Printing Strings and Characters	510

9.7	Other Conversion Specifiers	511
9.8	Printing with Field Widths and Precision	512
9.8.1	Field Widths for Integers	512
9.8.2	Precisions for Integers, Floating-Point Numbers and Strings	513
9.8.3	Combining Field Widths and Precisions	514
9.9	<code>printf</code> Format Flags	515
9.9.1	Right- and Left-Alignment	515
9.9.2	Printing Positive and Negative Numbers with and without the + Flag	516
9.9.3	Using the Space Flag	516
9.9.4	Using the # Flag	517
9.9.5	Using the 0 Flag	517
9.10	Printing Literals and Escape Sequences	518
9.11	Formatted Input with <code>scanf</code>	519
9.11.1	<code>scanf</code> Syntax	520
9.11.2	<code>scanf</code> Conversion Specifiers	520
9.11.3	Reading Integers	521
9.11.4	Reading Floating-Point Numbers	522
9.11.5	Reading Characters and Strings	522
9.11.6	Using Scan Sets	523
9.11.7	Using Field Widths	524
9.11.8	Skipping Characters in an Input Stream	525
9.12	Secure C Programming	526

## 10 Structures, Unions, Bit Manipulation and Enumerations 535

10.1	Introduction	536
10.2	Structure Definitions	537
10.2.1	Self-Referential Structures	537
10.2.2	Defining Variables of Structure Types	538
10.2.3	Structure Tag Names	538
10.2.4	Operations That Can Be Performed on Structures	538
10.3	Initializing Structures	540
10.4	Accessing Structure Members with . and ->	540
10.5	Using Structures with Functions	542
10.6	<code>typedef</code>	542
10.7	Random-Number Simulation Case Study: High-Performance Card Shuffling and Dealing	543
10.8	Unions	546
10.8.1	union Declarations	547
10.8.2	Allowed unions Operations	547
10.8.3	Initializing unions in Declarations	547
10.8.4	Demonstrating unions	548

10.9	Bitwise Operators	549
10.9.1	Displaying an Unsigned Integer's Bits	550
10.9.2	Making Function <code>displayBits</code> More Generic and Portable	551
10.9.3	Using the Bitwise AND, Inclusive OR, Exclusive OR and Complement Operators	552
10.9.4	Using the Bitwise Left- and Right-Shift Operators	555
10.9.5	Bitwise Assignment Operators	557
10.10	Bit Fields	558
10.10.1	Defining Bit Fields	558
10.10.2	Using Bit Fields to Represent a Card's Face, Suit and Color	559
10.10.3	Unnamed Bit Fields	561
10.11	Enumeration Constants	561
10.12	Anonymous Structures and Unions	563
10.13	Secure C Programming	564
	<b>Special Section: Raylib Game-Programming Case Studies</b>	574
	<b>Game-Programming Case Study Exercise: SpotOn Game</b>	580
	<b>Game-Programming Case Study: Cannon Game</b>	581
	<b>Visualization with raylib—Law of Large Numbers Animation</b>	583
	<b>Case Study: The Tortoise and the Hare with raylib— a Multimedia “Extravaganza”</b>	585
	<b>Random-Number Simulation Case Study: High-Performance Card Shuffling and Dealing with Card Images and raylib</b>	587

## 11 File Processing 593

11.1	Introduction	594
11.2	Files and Streams	594
11.3	Creating a Sequential-Access File	596
11.3.1	Pointer to a FILE	597
11.3.2	Using <code>fopen</code> to Open a File	597
11.3.3	Using <code>feof</code> to Check for the End-of-File Indicator	597
11.3.4	Using <code>fprintf</code> to Write to a File	598
11.3.5	Using <code>fclose</code> to Close a File	598
11.3.6	File-Open Modes	599
11.4	Reading Data from a Sequential-Access File	601
11.4.1	Resetting the File Position Pointer	602
11.4.2	Credit Inquiry Program	602
11.5	Random-Access Files	606
11.6	Creating a Random-Access File	607
11.7	Writing Data Randomly to a Random-Access File	609
11.7.1	Positioning the File Position Pointer with <code>fseek</code>	611
11.7.2	Error Checking	612
11.8	Reading Data from a Random-Access File	612

11.9	Case Study: Transaction-Processing System	614
11.10	Secure C Programming	620
	AI Case Study: Intro to NLP—Who Wrote Shakespeare’s Works?	630
	AI/Data-Science Case Study—Machine Learning with GNU Scientific Library	636
	AI/Data-Science Case Study: Time Series and Simple Linear Regression	642
	Web Services and the Cloud Case Study—libcurl and OpenWeatherMap	643

## **12 Data Structures 649**

12.1	Introduction	650
12.2	Self-Referential Structures	651
12.3	Dynamic Memory Management	652
12.4	Linked Lists	653
	12.4.1 Function insert	657
	12.4.2 Function delete	659
	12.4.3 Functions isEmpty and printList	661
12.5	Stacks	662
	12.5.1 Function push	666
	12.5.2 Function pop	667
	12.5.3 Applications of Stacks	667
12.6	Queues	668
	12.6.1 Function enqueue	673
	12.6.2 Function dequeue	674
12.7	Trees	675
	12.7.1 Function insertNode	678
	12.7.2 Traversals: Functions inOrder, preOrder and postOrder	679
	12.7.3 Duplicate Elimination	680
	12.7.4 Binary Tree Search	680
	12.7.5 Other Binary Tree Operations	680
12.8	Secure C Programming	681
	Special Section: Systems Software Case Study—Building Your Own Compiler	690

## **13 Computer-Science Thinking: Sorting Algorithms and Big O 711**

13.1	Introduction	712
13.2	Efficiency of Algorithms: Big O	713
	13.2.1 $O(1)$ Algorithms	713
	13.2.2 $O(n)$ Algorithms	713
	13.2.3 $O(n^2)$ Algorithms	713

## **14** Contents

13.3	Selection Sort	714
13.3.1	Selection Sort Implementation	715
13.3.2	Efficiency of Selection Sort	718
13.4	Insertion Sort	719
13.4.1	Insertion Sort Implementation	719
13.4.2	Efficiency of Insertion Sort	722
13.5	<b>Case Study: Visualizing the High-Performance Merge Sort</b>	722
13.5.1	Merge Sort Implementation	723
13.5.2	Efficiency of Merge Sort	727
13.5.3	Summarizing Various Algorithms' Big O Notations	728

## **14** Preprocessor **735**

14.1	Introduction	736
14.2	<code>#include</code> Preprocessor Directive	737
14.3	<code>#define</code> Preprocessor Directive: Symbolic Constants	737
14.4	<code>#define</code> Preprocessor Directive: Macros	738
14.4.1	Macro with One Argument	739
14.4.2	Macro with Two Arguments	740
14.4.3	Macro Continuation Character	740
14.4.4	<code>#undef</code> Preprocessor Directive	740
14.4.5	Standard-Library Macros	740
14.4.6	Do Not Place Expressions with Side Effects in Macros	741
14.5	Conditional Compilation	741
14.5.1	<code>#if...#endif</code> Preprocessor Directive	741
14.5.2	Commenting Out Blocks of Code with <code>#if...#endif</code>	742
14.5.3	Conditionally Compiling Debug Code	742
14.6	<code>#error</code> and <code>#pragma</code> Preprocessor Directives	743
14.7	<code>#</code> and <code>##</code> Operators	744
14.8	Line Numbers	744
14.9	Predefined Symbolic Constants	745
14.10	Assertions	745
14.11	Secure C Programming	746

## **15** Other Topics **753**

15.1	Introduction	754
15.2	Variable-Length Argument Lists	754
15.3	Using Command-Line Arguments	756
15.4	Compiling Multiple-Source-File Programs	758
15.4.1	<code>extern</code> Declarations for Global Variables in Other Files	758
15.4.2	Function Prototypes	759
15.4.3	Restricting Scope with <code>static</code>	759
15.5	Program Termination with <code>exit</code> and <code>atexit</code>	760

15.6	Suffixes for Integer and Floating-Point Literals	762
15.7	Signal Handling	762
15.8	Dynamic Memory Allocation Functions <code>calloc</code> and <code>realloc</code>	765
15.9	<code>goto</code> : Unconditional Branching	767

## **A** Operator Precedence Chart 773

## **B** ASCII Character Set 775

## **C** Multithreading/Multicore and Other C18/C11/C99 Topics 777

C.1	Introduction	778
C.2	Headers Added in C99	779
C.3	Designated Initializers and Compound Literals	779
C.4	Type <code>bool</code>	781
C.5	Complex Numbers	782
C.6	Macros with Variable-Length Argument Lists	784
C.7	Other C99 Features	784
C.7.1	Compiler Minimum Resource Limits	784
C.7.2	The <code>restrict</code> Keyword	784
C.7.3	Reliable Integer Division	785
C.7.4	Flexible Array Members	785
C.7.5	Type-Generic Math	786
C.7.6	Inline Functions	786
C.7.7	<code>__func__</code> Predefined Identifier	786
C.7.8	<code>va_copy</code> Macro	787
C.8	C11/C18 Features	787
C.8.1	C11/C18 Headers	787
C.8.2	<code>quick_exit</code> Function	787
C.8.3	Unicode® Support	787
C.8.4	<code>_Noreturn</code> Function Specifier	788
C.8.5	Type-Generic Expressions	788
C.8.6	Annex L: Analyzability and Undefined Behavior	788
C.8.7	Memory Alignment Control	789
C.8.8	Static Assertions	789
C.8.9	Floating-Point Types	789
C.9	Case Study: Performance with Multithreading and Multicore Systems	790
C.9.1	Example: Sequential Execution of Two Compute-Intensive Tasks	793
C.9.2	Example: Multithreaded Execution of Two Compute-Intensive Tasks	795
C.9.3	Other Multithreading Features	799

<b>D</b>	<b>Intro to Object-Oriented Programming Concepts</b>	<b>801</b>
D.1	Introduction	801
D.2	Object-Oriented Programming Languages	801
D.3	Automobile as an Object	802
D.4	Methods and Classes	802
D.5	Instantiation	802
D.6	Reuse	802
D.7	Messages and Method Calls	803
D.8	Attributes and Instance Variables	803
D.9	Inheritance	803
D.10	Object-Oriented Analysis and Design (OOAD)	804
 <b>Index</b>		 <b>805</b>

## **Online Appendices**

<b>E</b>	<b>Number Systems</b>
<b>F</b>	<b>Using the Visual Studio Debugger</b>
<b>G</b>	<b>Using the GNU gdb Debugger</b>
<b>H</b>	<b>Using the Xcode Debugger</b>