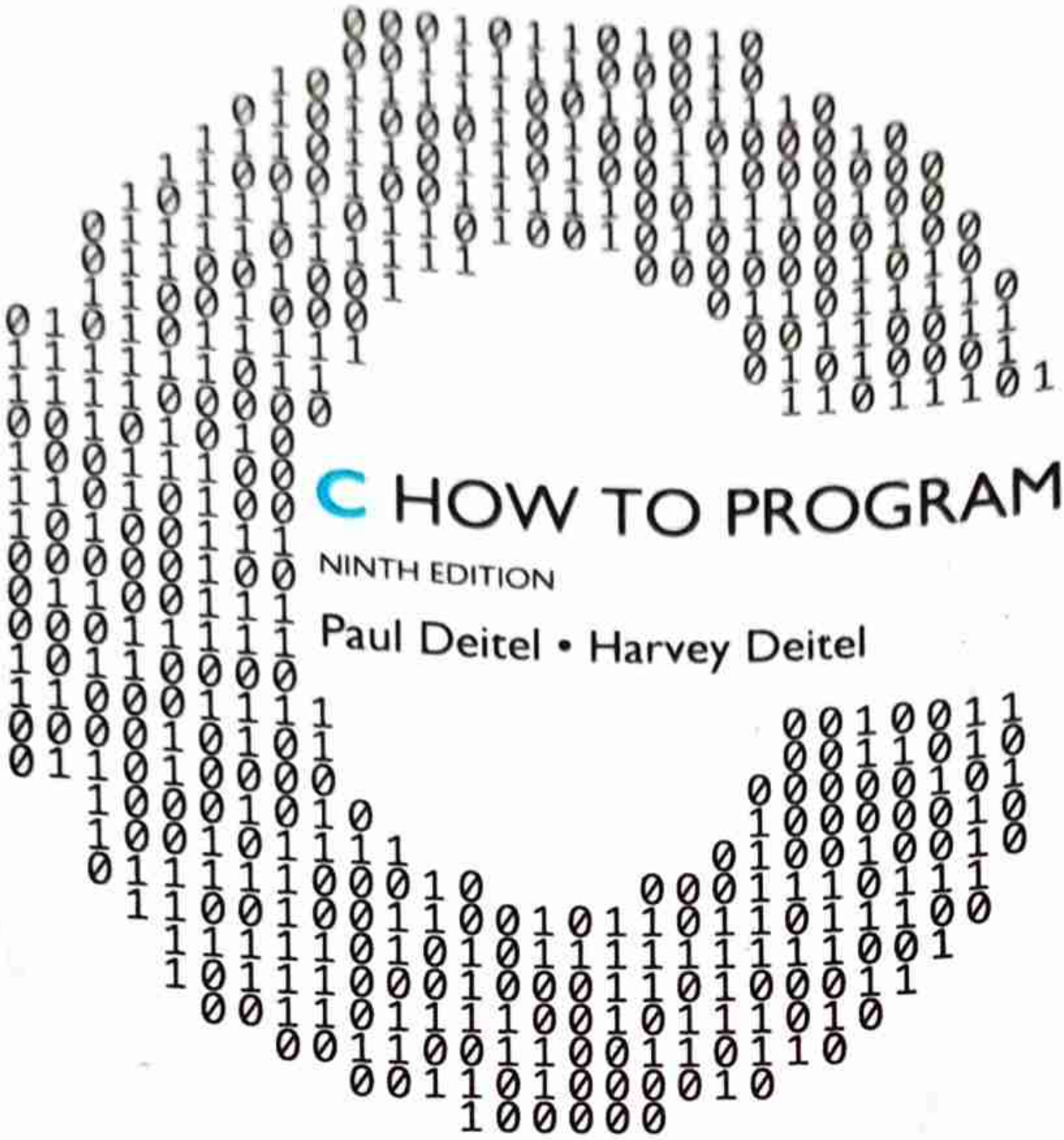
GLOBAL EDITION



C HOW TO PROGRAM

NINTH EDITION

Paul Deitel • Harvey Deitel

with case studies introducing

Applications Programming and

Systems Programming

DEITEL

C HOW TO PROGRAM

NINTH EDITION

GLOBAL EDITION

with

Case Studies Introducing

Applications Programming and

Systems Programming

PAUL DEITEL
HARVEY DEITEL

Contents

Appendices E–H are PDF documents posted online at the book's Companion Website (located at www.pearsonglobaleditions.com).

Preface 17

Before You Begin 49

1 Introduction to Computers and C 53
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

2 Intro to C Programming

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

3 Structured Program Development

3.1 Introduction
3.2 Algorithms
3.3 Pseudocode
3.4 Control Structures
3.5 The `if` Selection Statement
3.6 The `if...else` Selection Statement
3.7 The `while` Iteration Statement
3.8 **Formulating Algorithms Case Study 1: Counter-Controlled Iteration**
3.9 **Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 2: Sentinel-Controlled Iteration**
3.10 **Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 3: Nested Control Statements**
3.11 Assignment Operators
3.12 Increment and Decrement Operators
3.13 Secure C Programming

4 Program Control 183
4.1 Introduction 184
4.2 Iteration Essentials 184
4.3 Counter-Controlled Iteration 185
4.4 for Iteration Statement 188
4.5 Examples Using the for Statement 191
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

5 Functions 231
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 **Game Simulation Case Study: Rock, Paper, Scissors** 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
Random-Number Simulation Case Study: The Tortoise and the Hare 294

6 Arrays 297
6.1 Introduction 298
6.2 Arrays 298
6.3 Defining Arrays 300
6.4 Array Examples 300

4 Program Control

4.1 Introduction 185
4.2 Iteration Essentials 186
4.3 Counter-Controlled Iteration 186
4.4 for Iteration Statement 187
4.5 Examples Using the for Statement 188
4.6 switch Multiple-Selection Statement 193
4.7 do...while Iteration Statement 196
4.8 break and continue Statements 202
4.9 Logical Operators 203
4.10 Confusing Equality (==) and Assignment (=) Operators 205
4.11 Structured Programming Summary 209
4.12 Secure C Programming 210
215

5 Functions 231

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 **Game Simulation Case Study: Rock, Paper, Scissors** 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
Random-Number Simulation Case Study: The Tortoise and the Hare 294

6 Arrays 297

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 310
6.5.5 Outputting a Character Array That Represents a String 310
6.5.6 Demonstrating Character Arrays 312
6.6 Static Local Arrays and Automatic Local Arrays 314
6.7 Passing Arrays to Functions 318
6.8 Sorting Arrays 321
6.9 **Intro to Data Science Case Study: Survey Data Analysis** 326
6.10 Searching Arrays 326
6.10.1 Searching an Array with Linear Search 328
6.10.2 Searching an Array with Binary Search 332
6.11 Multidimensional Arrays 332
6.11.1 Illustrating a Two-Dimensional Array 333
6.11.2 Initializing a Double-Subscripted Array 335
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 339
6.12 Variable-Length Arrays 343
6.13 Secure C Programming

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 const 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 174
7.5.3 Attempting to Modify a Constant Pointer to Non-Constant Data 176
7.5.4 Attempting to Modify a Constant Pointer to Constant Data 177
7.6 Bubble Sort Using Pass-By-Reference 178
7.7 sizeof Operator 182
7.8 Pointer Expressions and Pointer Arithmetic 184
7.8.1 Pointer Arithmetic Operators 185
7.8.2 Aiming a Pointer at an Array 185
7.8.3 Adding an Integer to a Pointer 185
7.8.4 Subtracting an Integer from a Pointer 186
7.8.5 Incrementing and Decrementing a Pointer 186
7.8.6 Subtracting One Pointer from Another 186
7.8.7 Assigning Pointers to One Another 186
7.8.8 Pointer to `void` 186
7.8.9 Comparing Pointers 187
7.9 Relationship between Pointers and Arrays 187
7.9.1 Pointer/Offset Notation 187
7.9.2 Pointer/Subscript Notation 188
7.9.3 Cannot Modify an Array Name with Pointer Arithmetic 188
7.9.4 Demonstrating Pointer Subscripting and Offsets 188
7.9.5 String Copying with Arrays and Pointers 190
7.10 Arrays of Pointers 192
7.11 **Random-Number Simulation Case Study: Card Shuffling and Dealing** 193
7.12 Function Pointers 198
7.12.1 Sorting in Ascending or Descending Order 198
7.12.2 Using Function Pointers to Create a Menu-Driven System 201
7.13 Secure C Programming 203
Special Section: Building Your Own Computer as a Virtual Machine 217
Special Section—Embedded Systems Programming Case Study: Robotics with the Webots Simulator 224

8 Characters and Strings 241
8.1 Introduction 242
8.2 Fundamentals of Strings and Characters 242
8.3 Character-Handling Library 244
8.3.1 Functions `isdigit`, `isalpha`, `isalnum` and `isxdigit` 245
8.3.2 Functions `islower`, `isupper`, `tolower` and `toupper` 247
8.3.3 Functions `isspace`, `iscntrl`, `ispunct`, `isprint` and `isgraph` 248
8.4 String-Conversion Functions 250
8.4.1 Function `strtod` 250

8.4.2 Function strtol 45
8.4.3 Function strtoul 45
8.5 Standard Input/Output Library Functions 45
8.5.1 Functions fgets and putchar 45
8.5.2 Function getchar 45
8.5.3 Function sprintf 45
8.5.4 Function sscanf 45
8.6 String Manipulation Functions of the String-Handling Library 45
8.6.1 Functions strcpy and strncpy 45
8.6.2 Functions strcat and strncat 46
8.7 Comparison Functions of the String-Handling Library 46
8.8 Search Functions of the String-Handling Library 46
8.8.1 Function strchr 46
8.8.2 Function strcspn 46
8.8.3 Function strpbrk 46
8.8.4 Function strrchr 46
8.8.5 Function strspn 46
8.8.6 Function strstr 46
8.8.7 Function strtok 46
8.9 Memory Functions of the String-Handling Library 47
8.9.1 Function memcpy 47
8.9.2 Function memmove 47
8.9.3 Function memcmp 47
8.9.4 Function memchr 47
8.9.5 Function memset 47
8.10 Other Functions of the String-Handling Library 47
8.10.1 Function strerror 47
8.10.2 Function strlen 47
8.11 Secure C Programming 48
Pqyoaf X Nylfomigrob Qwbbfmh Mndogvk: Rboqlrut yua Boklnxhmywex 49
Secure C Programming Case Study: Public-Key Cryptography 49

9 Formatted Input/Output 50
9.1 Introduction 50
9.2 Streams 50
9.3 Formatting Output with printf 50
9.4 Printing Integers 50
9.5 Printing Floating-Point Numbers 50
9.5.1 Conversion Specifiers e, E and f 50
9.5.2 Conversion Specifiers g and G 50
9.5.3 Demonstrating Floating-Point Conversion Specifiers 51
9.6 Printing Strings and Characters

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 printf Format Flags 515
9.9.1 Right- and Left-Justification 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 scanf 519
9.11.1 scanf Syntax 520
9.11.2 scanf 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 typedef 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

Special Section: Raylib Game-Programming Case Studies
Game-Programming Case Study Exercise: SpotOn Game
Game-Programming Case Study: Cannon Game
Visualization with raylib—Law of Large Numbers Animation
Case Study: The Tortoise and the Hare with raylib—a Multimedia "Extravaganza"
Random-Number Simulation Case Study: High-Performance Card Shuffling and Dealing with Card Images and raylib

11 File Processing

11.1 Introduction
11.2 Files and Streams
11.3 Creating a Sequential-Access File
  11.3.1 Pointer to a `FILE`
  11.3.2 Using `fopen` to Open a File
  11.3.3 Using `feof` to Check for the End-of-File Indicator
  11.3.4 Using `fprintf` to Write to a File
  11.3.5 Using `fclose` to Close a File
  11.3.6 File-Open Modes
11.4 Reading Data from a Sequential-Access File
  11.4.1 Resetting the File Position Pointer
  11.4.2 Credit Inquiry Program
11.5 Random-Access Files
11.6 Creating a Random-Access File
11.7 Writing Data Randomly to a Random-Access File
  11.7.1 Positioning the File Position Pointer with `fseek`
  11.7.2 Error Checking
11.8 Reading Data from a Random-Access File

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

15 Other Topics
15.1 Introduction
15.2 Variable-Length Argument Lists
15.3 Using Command-Line Arguments
15.4 Compiling Multiple-Source-File Programs
15.4.1 extern Declarations for Global Variables in Other Files
15.4.2 Function Prototypes
15.4.3 Restricting Scope with static
15.5 Program Termination with exit and atexit

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 bool 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 `restrict` 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 `__func__` Predefined Identifier 786
C.7.8 `va_copy` Macro 787
C.8 C11/C18 Features 787
C.8.1 C11/C18 Headers 787
C.8.2 `quick_exit` Function 787
C.8.3 Unicode® Support 787
C.8.4 `_Noreturn` 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

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

Index

Online Appendices

E Number Systems

F Using the Visual Studio Debugger

G Using the GNU gdb Debugger

H Using the Xcode Debugger