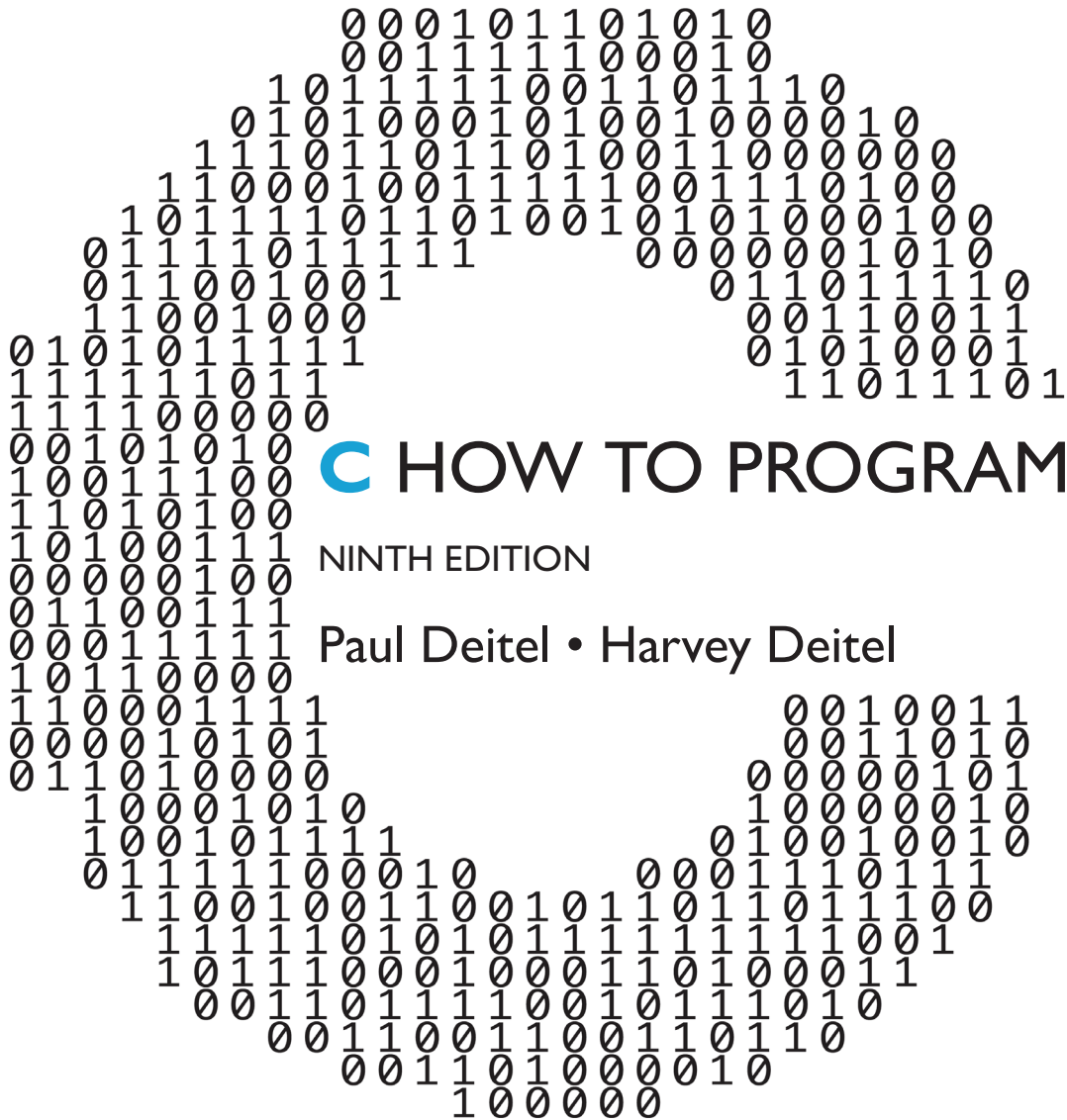


GLOBAL
EDITION



with case studies introducing

Applications Programming and
Systems Programming





DEITEL®

HOW TO PROGRAM

NINTH
EDITION
GLOBAL
EDITION

with
Case Studies Introducing

**Applications
Programming** and

**Systems
Programming**

PAUL DEITEL
HARVEY DEITEL

C How to Program: With Case Studies in Applications and Systems Programming, Global Edition

Table of Contents

Cover

Title Page

Copyright

Dedication

Contents

Preface

Before You Begin

Chapter 1. Introduction to Computers and C

1.1 Introduction

1.2 Hardware and Software

1.2.1 Moores Law

1.2.2 Computer Organization

1.3 Data Hierarchy

1.4 Machine Languages, Assembly Languages and High-Level Languages

1.5 Operating Systems

1.6 The C Programming Language

1.7 The C Standard Library and Open-Source Libraries

1.8 Other Popular Programming Languages

1.9 Typical C Program-Development Environment

1.9.1 Phase 1: Creating a Program

1.9.2 Phases 2 and 3: Preprocessing and Compiling a C Program

1.9.3 Phase 4: Linking

1.9.4 Phase 5: Loading

1.9.5 Phase 6: Execution

Table of Contents

1.9.6 Problems That May Occur at Execution Time

1.9.7 Standard Input, Standard Output and Standard Error Streams

1.10 Test-Driving a C Application in Windows, Linux and macOS

1.10.1 Compiling and Running a C Application with Visual Studio 2019 Community Edition on Windows 10

1.10.2 Compiling and Running a C Application with Xcode on macOS

1.10.3 Compiling and Running a C Application with GNU gcc on Linux

1.10.4 Compiling and Running a C Application in a GCC Docker Container Running Natively over Windows 10, macOS or Linux

1.11 Internet, World Wide Web, the Cloud and IoT

1.11.1 The Internet: A Network of Networks

1.11.2 The World Wide Web: Making the Internet User-Friendly

1.11.3 The Cloud

1.11.4 The Internet of Things

1.12 Software Technologies

1.13 How Big Is Big Data?

1.13.1 Big-Data Analytics

1.13.2 Data Science and Big Data Are Making a Difference: Use Cases

1.14 Case Study A Big-Data Mobile Application

1.15 At the Intersection of Computer Science and Data Science

Chapter 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

Chapter 3. Structured Program Development

3.1 Introduction

3.2 Algorithms

Table of Contents

3.3 Pseudocode

3.4 Control Structures

3.5 The if Selection Statement

3.6 The ifelse 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

Chapter 4. Program Control

4.1 Introduction

4.2 Iteration Essentials

4.3 Counter-Controlled Iteration

4.4 for Iteration Statement

4.5 Examples Using the for Statement

4.6 switch Multiple-Selection Statement

4.7 dowhile Iteration Statement

4.8 break and continue Statements

4.9 Logical Operators

4.10 Confusing Equality (==) and Assignment (=) Operators

4.11 Structured-Programming Summary

4.12 Secure C Programming

Chapter 5. Functions

5.1 Introduction

5.2 Modularizing Programs in C

5.3 Math Library Functions

Table of Contents

5.4 Functions

5.5 Function Definitions

5.5.1 square Function

5.5.2 maximum Function

5.6 Function Prototypes: A Deeper Look

5.7 Function-Call Stack and Stack Frames

5.8 Headers

5.9 Passing Arguments by Value and by Reference

5.10 Random-Number Generation

5.11 Game Simulation Case Study: Rock, Paper, Scissors

5.12 Storage Classes

5.13 Scope Rules

5.14 Recursion

5.15 Example Using Recursion: Fibonacci Series

5.16 Recursion vs. Iteration

5.17 Secure C ProgrammingSecure Random-Number Generation

Random-Number Simulation Case Study: The Tortoise and the Hare

Chapter 6. Arrays

6.1 Introduction

6.2 Arrays

6.3 Defining Arrays

6.4 Array Examples

6.4.1 Defining an Array and Using a Loop to Set the Arrays Element Values

6.4.2 Initializing an Array in a Definition with an Initializer List

6.4.3 Specifying an Arrays Size with a Symbolic Constant and Initializing Array Elements with Calculations

6.4.4 Summing the Elements of an Array

6.4.5 Using Arrays to Summarize Survey Results

6.4.6 Graphing Array Element Values with Bar Charts

6.4.7 Rolling a Die 60,000,000 Times and Summarizing the Results in an Array

6.5 Using Character Arrays to Store and Manipulate Strings

Table of Contents

- 6.5.1 Initializing a Character Array with a String
- 6.5.2 Initializing a Character Array with an Initializer List of Characters
- 6.5.3 Accessing the Characters in a String
- 6.5.4 Inputting into a Character Array
- 6.5.5 Outputting a Character Array That Represents a String
- 6.5.6 Demonstrating Character Arrays

6.6 Static Local Arrays and Automatic Local Arrays

6.7 Passing Arrays to Functions

6.8 Sorting Arrays

6.9 Intro to Data Science Case Study: Survey Data Analysis

6.10 Searching Arrays

- 6.10.1 Searching an Array with Linear Search
- 6.10.2 Searching an Array with Binary Search

6.11 Multidimensional Arrays

- 6.11.1 Illustrating a Two-Dimensional Array
- 6.11.2 Initializing a Double-Subscripted Array
- 6.11.3 Setting the Elements in One Row
- 6.11.4 Totaling the Elements in a Two-Dimensional Array
- 6.11.5 Two-Dimensional Array Manipulations

6.12 Variable-Length Arrays

6.13 Secure C Programming

Chapter 7. Pointers

7.1 Introduction

7.2 Pointer Variable Definitions and Initialization

7.3 Pointer Operators

7.4 Passing Arguments to Functions by Reference

7.5 Using the const Qualifier with Pointers

- 7.5.1 Converting a String to Uppercase Using a Non-Constant Pointer to Non-Constant Data
- 7.5.2 Printing a String One Character at a Time Using a Non-Constant Pointer to Constant Data
- 7.5.3 Attempting to Modify a Constant Pointer to Non-Constant Data

Table of Contents

7.5.4 Attempting to Modify a Constant Pointer to Constant Data

7.6 Bubble Sort Using Pass-By-Reference

7.7 sizeof Operator

7.8 Pointer Expressions and Pointer Arithmetic

7.8.1 Pointer Arithmetic Operators

7.8.2 Aiming a Pointer at an Array

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/Subscript Notation

7.9.3 Cannot Modify an Array Name with Pointer Arithmetic

7.9.4 Demonstrating Pointer Subscripting and Offsets

7.9.5 String Copying with Arrays and Pointers

7.10 Arrays of Pointers

7.11 Random-Number Simulation Case Study: Card Shuffling and Dealing

7.12 Function Pointers

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

Special Section: Building Your Own Computer as a Virtual Machine

Special Section Embedded Systems Programming Case Study: Robotics with the Webots Simulator

Chapter 8. Characters and Strings

8.1 Introduction

8.2 Fundamentals of Strings and Characters

8.3 Character-Handling Library

Table of Contents

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 strtod

8.4.2 Function strtol

8.4.3 Function strtoul

8.5 Standard Input/Output Library Functions

8.5.1 Functions fgets and putchar

8.5.2 Function getchar

8.5.3 Function sprintf

8.5.4 Function sscanf

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

8.8.1 Function strchr

8.8.2 Function strcspn

8.8.3 Function strpbrk

8.8.4 Function strrchr

8.8.5 Function strspn

8.8.6 Function strstr

8.8.7 Function strtok

8.9 Memory Functions of the String-Handling Library

8.9.1 Function memcpy

8.9.2 Function memmove

8.9.3 Function memcmp

8.9.4 Function memchr

8.9.5 Function memset

8.10 Other Functions of the String-Handling Library

8.10.1 Function strerror

8.10.2 Function strlen

Table of Contents

8.11 Secure C Programming

Pqyoaf X Nylfomigrob Qwbbfmh Mndogvk: Rboqlrut yua Boklnxhmywex

Secure C Programming Case Study: Public-Key Cryptography

Chapter 9. Formatted Input/Output

9.1 Introduction

9.2 Streams

9.3 Formatting Output with printf

9.4 Printing Integers

9.5 Printing Floating-Point Numbers

9.5.1 Conversion Specifiers e, E and f

9.5.2 Conversion Specifiers g and G

9.5.3 Demonstrating Floating-Point Conversion Specifiers

9.6 Printing Strings and Characters

9.7 Other Conversion Specifiers

9.8 Printing with Field Widths and Precision

9.8.1 Field Widths for Integers

9.8.2 Precisions for Integers, Floating-Point Numbers and Strings

9.8.3 Combining Field Widths and Precisions

9.9 printf Format Flags

9.9.1 Right- and Left-Alignment

9.9.2 Printing Positive and Negative Numbers with and without the + Flag

9.9.3 Using the Space Flag

9.9.4 Using the # Flag

9.9.5 Using the 0 Flag

9.10 Printing Literals and Escape Sequences

9.11 Formatted Input with scanf

9.11.1 scanf Syntax

9.11.2 scanf Conversion Specifiers

9.11.3 Reading Integers

9.11.4 Reading Floating-Point Numbers

9.11.5 Reading Characters and Strings

9.11.6 Using Scan Sets

Table of Contents

9.11.7 Using Field Widths

9.11.8 Skipping Characters in an Input Stream

9.12 Secure C Programming

Chapter 10. Structures, Unions, Bit Manipulation and Enumerations

10.1 Introduction

10.2 Structure Definitions

10.2.1 Self-Referential Structures

10.2.2 Defining Variables of Structure Types

10.2.3 Structure Tag Names

10.2.4 Operations That Can Be Performed on Structures

10.3 Initializing Structures

10.4 Accessing Structure Members with . and ->

10.5 Using Structures with Functions

10.6 typedef

10.7 Random-Number Simulation Case Study: High-Performance Card Shuffling and Dealing

10.8 Unions

10.8.1 union Declarations

10.8.2 Allowed unions Operations

10.8.3 Initializing unions in Declarations

10.8.4 Demonstrating unions

10.9 Bitwise Operators

10.9.1 Displaying an Unsigned Integers Bits

10.9.2 Making Function displayBits More Generic and Portable

10.9.3 Using the Bitwise AND, Inclusive OR, Exclusive OR and Complement Operators

10.9.4 Using the Bitwise Left- and Right-Shift Operators

10.9.5 Bitwise Assignment Operators

10.10 Bit Fields

10.10.1 Defining Bit Fields

10.10.2 Using Bit Fields to Represent a Cards Face, Suit and Color

10.10.3 Unnamed Bit Fields

10.11 Enumeration Constants

Table of Contents

10.12 Anonymous Structures and Unions

10.13 Secure C Programming

Special Section: Raylib Game-Programming Case Studies

Game-Programming Case Study Exercise: SpotOn Game

Game-Programming Case Study: Cannon Game

Visualization with raylibLaw of Large Numbers Animation

Case Study: The Tortoise and the Hare with rayliba Multimedia Extravaganza

Random-Number Simulation Case Study: High-Performance Card Shuffling and Dealing
with Card Images and raylib

Chapter 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

11.10 Secure C Programming

AI Case Study: Intro to NLPWho Wrote Shakespeares Works?

AI/Data-Science Case StudyMachine Learning with GNU Scientific Library

Table of Contents

AI/Data-Science Case Study: Time Series and Simple Linear Regression

Web Services and the Cloud Case Studylibcurl and OpenWeatherMap

Chapter 12. Data Structures

12.1 Introduction

12.2 Self-Referential Structures

12.3 Dynamic Memory Management

12.4 Linked Lists

12.4.1 Function insert

12.4.2 Function delete

12.4.3 Functions isEmpty and printList

12.5 Stacks

12.5.1 Function push

12.5.2 Function pop

12.5.3 Applications of Stacks

12.6 Queues

12.6.1 Function enqueue

12.6.2 Function dequeue

12.7 Trees

12.7.1 Function insertNode

12.7.2 Traversals: Functions inOrder, preOrder and postOrder

12.7.3 Duplicate Elimination

12.7.4 Binary Tree Search

12.7.5 Other Binary Tree Operations

12.8 Secure C Programming

Special Section: Systems Software Case StudyBuilding Your Own Compiler

Chapter 13. Computer-Science Thinking: Sorting Algorithms and Big O

13.1 Introduction

13.2 Efficiency of Algorithms: Big O

13.2.1 $O(1)$ Algorithms

13.2.2 $O(n)$ Algorithms

13.2.3 $O(n^2)$ Algorithms

Table of Contents

13.3 Selection Sort

13.3.1 Selection Sort Implementation

13.3.2 Efficiency of Selection Sort

13.4 Insertion Sort

13.4.1 Insertion Sort Implementation

13.4.2 Efficiency of Insertion Sort

13.5 Case Study: Visualizing the High-Performance Merge Sort

13.5.1 Merge Sort Implementation

13.5.2 Efficiency of Merge Sort

13.5.3 Summarizing Various Algorithms Big O Notations

Chapter 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

Table of Contents

Chapter 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

15.6 Suffices for Integer and Floating-Point Literals

15.7 Signal Handling

15.8 Dynamic Memory Allocation Functions calloc and realloc

15.9 goto: Unconditional Branching

A. Operator Precedence Chart

B. ASCII Character Set

C. Multithreading/Multicore and Other C18/C11/C99 Topics

C.1 Introduction

C.2 Headers Added in C99

C.3 Designated Initializers and Compound Literals

C.4 Type bool

C.5 Complex Numbers

C.6 Macros with Variable-Length Argument Lists

C.7 Other C99 Features

C.7.1 Compiler Minimum Resource Limits

C.7.2 The restrict Keyword

C.7.3 Reliable Integer Division

C.7.4 Flexible Array Members

C.7.5 Type-Generic Math

C.7.6 Inline Functions

C.7.7 __func__ Predefined Identifier

C.7.8 va_copy Macro

Table of Contents

C.8 C11/C18 Features

- C.8.1 C11/C18 Headers
- C.8.2 quick_exit Function
- C.8.3 Unicode® Support
- C.8.4 _Noreturn Function Specifier
- C.8.5 Type-Generic Expressions
- C.8.6 Annex L: Analyzability and Undefined Behavior
- C.8.7 Memory Alignment Control
- C.8.8 Static Assertions
- C.8.9 Floating-Point Types

C.9 Case Study: Performance with Multithreading and Multicore Systems

- C.9.1 Example: Sequential Execution of Two Compute-Intensive Tasks
- C.9.2 Example: Multithreaded Execution of Two Compute-Intensive Tasks
- C.9.3 Other Multithreading Features

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