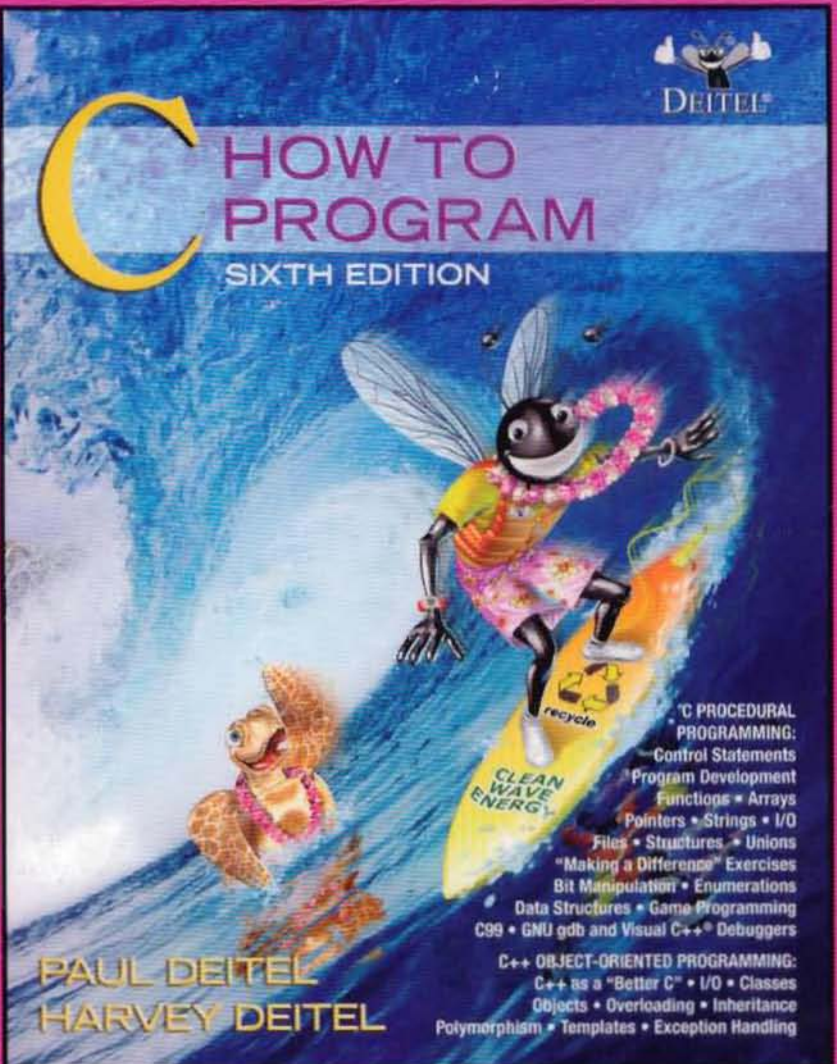
International Edition

C How to Program

Sixth Edition

Paul Deitel
Harvey Deitel


DEITEL®
C HOW TO PROGRAM
SIXTH EDITION
recycle
CLEAN WAVE ENERGY
C PROCEDURAL PROGRAMMING:
Control Statements
Program Development
Functions • Arrays
Pointers • Strings • I/O
Files • Structures • Unions
"Making a Difference" Exercises
Bit Manipulation • Enumerations
Data Structures • Game Programming
C99 • GNU gdb and Visual C++® Debuggers
C++ OBJECT-ORIENTED PROGRAMMING:
C++ as a "Better C" • I/O • Classes
Objects • Overloading • Inheritance
Polymorphism • Templates • Exception Handling
PAUL DEITEL
HARVEY DEITEL

PEARSON

Contents

Appendices E through I are PDF documents posted online at the book's Companion Website (located at `www.pearsonhighered.com/deitel`).

Preface 21

1 Introduction to Computers, the Internet and the Web 33

1.1 Introduction 34
1.2 Computers: Hardware and Software 35
1.3 Computer Organization 36
1.4 Personal, Distributed and Client/Server Computing 37
1.5 The Internet and the World Wide Web 37
1.6 Machine Languages, Assembly Languages and High-Level Languages 38
1.7 History of C 39
1.8 C Standard Library 40
1.9 C++ 41
1.10 Java 41
1.11 Fortran, COBOL, Pascal and Ada 42
1.12 BASIC, Visual Basic, Visual C++, C# and .NET 42
1.13 Key Software Trend: Object Technology 43
1.14 Typical C Program Development Environment 44
1.15 Hardware Trends 46
1.16 Notes About C and This Book 47
1.17 Web Resources 48

2 Introduction to C Programming 55

2.1 Introduction 56
2.2 A Simple C Program: Printing a Line of Text 56
2.3 Another Simple C Program: Adding Two Integers 60
2.4 Memory Concepts 65
2.5 Arithmetic in C 66
2.6 Decision Making: Equality and Relational Operators 70

3 Structured Program Development in C 86

3.1 Introduction 87
3.2 Algorithms 87

3.3 Pseudocode 87
3.4 Control Structures 88
3.5 The `if` Selection Statement 90
3.6 The `if...else` Selection Statement 91
3.7 The `while` Repetition Statement 95
3.8 Formulating Algorithms Case Study 1: Counter-Controlled Repetition 96
3.9 Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 2: Sentinel-Controlled Repetition 98
3.10 Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 3: Nested Control Structures 105
3.11 Assignment Operators 109
3.12 Increment and Decrement Operators 110

4 C Program Control 129

4.1 Introduction 130
4.2 Repetition Essentials 130
4.3 Counter-Controlled Repetition 131
4.4 `for` Repetition Statement 132
4.5 `for` Statement: Notes and Observations 135
4.6 Examples Using the `for` Statement 135
4.7 `switch` Multiple-Selection Statement 139
4.8 `do...while` Repetition Statement 145
4.9 `break` and `continue` Statements 146
4.10 Logical Operators 148
4.11 Confusing Equality (==) and Assignment (=) Operators 151
4.12 Structured Programming Summary 153

5 C Functions 172

5.1 Introduction 173
5.2 Program Modules in C 173
5.3 Math Library Functions 174
5.4 Functions 176
5.5 Function Definitions 176
5.6 Function Prototypes 180
5.7 Function Call Stack and Activation Records 183
5.8 Headers 183
5.9 Calling Functions By Value and By Reference 184
5.10 Random Number Generation 185
5.11 Example: A Game of Chance 190
5.12 Storage Classes 193
5.13 Scope Rules 196
5.14 Recursion 199
5.15 Example Using Recursion: Fibonacci Series 202
5.16 Recursion vs. Iteration 206

6 C Arrays 227

6.1 Introduction 228
6.2 Arrays 228
6.3 Defining Arrays 230
6.4 Array Examples 230
6.5 Passing Arrays to Functions 244
6.6 Sorting Arrays 248
6.7 Case Study: Computing Mean, Median and Mode Using Arrays 250
6.8 Searching Arrays 255
6.9 Multiple-Subscripted Arrays 261

7 C Pointers 285

7.1 Introduction 286
7.2 Pointer Variable Definitions and Initialization 286
7.3 Pointer Operators 287
7.4 Passing Arguments to Functions by Reference 289
7.5 Using the `const` Qualifier with Pointers 293
7.6 Bubble Sort Using Call-by-Reference 299
7.7 `sizeof` Operator 302
7.8 Pointer Expressions and Pointer Arithmetic 305
7.9 Relationship between Pointers and Arrays 307
7.10 Arrays of Pointers 312
7.11 Case Study: Card Shuffling and Dealing Simulation 312
7.12 Pointers to Functions 317

8 C Characters and Strings 341

8.1 Introduction 342
8.2 Fundamentals of Strings and Characters 342
8.3 Character-Handling Library 344
8.4 String-Conversion Functions 349
8.5 Standard Input/Output Library Functions 354
8.6 String-Manipulation Functions of the String-Handling Library 358
8.7 Comparison Functions of the String-Handling Library 361
8.8 Search Functions of the String-Handling Library 363
8.9 Memory Functions of the String-Handling Library 369
8.10 Other Functions of the String-Handling Library 373

9 C Formatted Input/Output 388

9.1 Introduction 389
9.2 Streams 389
9.3 Formatting Output with `printf` 389
9.4 Printing Integers 390
9.5 Printing Floating-Point Numbers 391

9.6 Printing Strings and Characters 393
9.7 Other Conversion Specifiers 394
9.8 Printing with Field Widths and Precision 395
9.9 Using Flags in the `printf` Format Control String 398
9.10 Printing Literals and Escape Sequences 400
9.11 Reading Formatted Input with `scanf` 401

10 C Structures, Unions, Bit Manipulations and Enumerations 414

10.1 Introduction 415
10.2 Structure Definitions 415
10.3 Initializing Structures 418
10.4 Accessing Structure Members 418
10.5 Using Structures with Functions 420
10.6 `typedef` 420
10.7 Example: High-Performance Card Shuffling and Dealing Simulation 421
10.8 Unions 423
10.9 Bitwise Operators 426
10.10 Bit Fields 435
10.11 Enumeration Constants 438

11 C File Processing 449

11.1 Introduction 450
11.2 Data Hierarchy 450
11.3 Files and Streams 452
11.4 Creating a Sequential-Access File 453
11.5 Reading Data from a Sequential-Access File 458
11.6 Random-Access Files 462
11.7 Creating a Random-Access File 463
11.8 Writing Data Randomly to a Random-Access File 465
11.9 Reading Data from a Random-Access File 468
11.10 Case Study: Transaction-Processing Program 469

12 C Data Structures 486

12.1 Introduction 487
12.2 Self-Referential Structures 488
12.3 Dynamic Memory Allocation 488
12.4 Linked Lists 490
12.5 Stacks 498
12.6 Queues 504
12.7 Trees 510

13 C Preprocessor 527

13.1 Introduction 528

13.2 #include Preprocessor Directive 528
13.3 #define Preprocessor Directive: Symbolic Constants 528
13.4 #define Preprocessor Directive: Macros 529
13.5 Conditional Compilation 531
13.6 #error and #pragma Preprocessor Directives 532
13.7 # and ## Operators 532
13.8 Line Numbers 533
13.9 Predefined Symbolic Constants 533
13.10 Assertions 534

14 Other C Topics 539

14.1 Introduction 540
14.2 Redirecting I/O 540
14.3 Variable-Length Argument Lists 541
14.4 Using Command-Line Arguments 543
14.5 Notes on Compiling Multiple-Source-File Programs 544
14.6 Program Termination with exit and atexit 546
14.7 volatile Type Qualifier 547
14.8 Suffixes for Integer and Floating-Point Constants 548
14.9 More on Files 548
14.10 Signal Handling 550
14.11 Dynamic Memory Allocation: Functions calloc and realloc 552
14.12 Unconditional Branching with goto 553

15 C++ as a Better C; Introducing Object Technology 560

15.1 Introduction 561
15.2 C++ 561
15.3 A Simple Program: Adding Two Integers 562
15.4 C++ Standard Library 564
15.5 Header Files 565
15.6 Inline Functions 567
15.7 References and Reference Parameters 569
15.8 Empty Parameter Lists 574
15.9 Default Arguments 574
15.10 Unary Scope Resolution Operator 576
15.11 Function Overloading 577
15.12 Function Templates 580
15.13 Introduction to Object Technology and the UML 583
15.14 Wrap-Up 586

16 Introduction to Classes and Objects 592

16.1 Introduction 593
16.2 Classes, Objects, Member Functions and Data Members 593

16.3 Defining a Class with a Member Function 594
16.4 Defining a Member Function with a Parameter 598
16.5 Data Members, *set* Functions and *get* Functions 601
16.6 Initializing Objects with Constructors 608
16.7 Placing a Class in a Separate File for Reusability 611
16.8 Separating Interface from Implementation 615
16.9 Validating Data with *set* Functions 621
16.10 Wrap-Up 626

17 Classes: A Deeper Look, Part 1 633

17.1 Introduction 634
17.2 `Time` Class Case Study 635
17.3 Class Scope and Accessing Class Members 641
17.4 Separating Interface from Implementation 643
17.5 Access Functions and Utility Functions 644
17.6 `Time` Class Case Study: Constructors with Default Arguments 647
17.7 Destructors 652
17.8 When Constructors and Destructors are Called 653
17.9 `Time` Class Case Study: A Subtle Trap—Returning a Reference to a `private` Data Member 656
17.10 Default Memberwise Assignment 659
17.11 Wrap-Up 661

18 Classes: A Deeper Look, Part 2 667

18.1 Introduction 668
18.2 `const` (Constant) Objects and `const` Member Functions 668
18.3 Composition: Objects as Members of Classes 677
18.4 `friend` Functions and `friend` Classes 683
18.5 Using the `this` Pointer 686
18.6 `static` Class Members 691
18.7 Data Abstraction and Information Hiding 696
18.8 Wrap-Up 698

19 Operator Overloading 704

19.1 Introduction 705
19.2 Fundamentals of Operator Overloading 706
19.3 Restrictions on Operator Overloading 707
19.4 Operator Functions as Class Members vs. Global Function 708
19.5 Overloading Stream Insertion and Stream Extraction Operators 710
19.6 Overloading Unary Operators 713
19.7 Overloading Binary Operators 714
19.8 Dynamic Memory Management 714
19.9 Case Study: `Array` Class 716
19.10 Converting between Types 728

19.11 Building a String Class 729
19.12 Overloading ++ and -- 730
19.13 Case Study: A Date Class 732
19.14 Standard Library Class string 736
19.15 explicit Constructors 740
19.16 Proxy Classes 743
19.17 Wrap-Up 747

20 Object-Oriented Programming: Inheritance 759

20.1 Introduction 760
20.2 Base Classes and Derived Classes 761
20.3 protected Members 764
20.4 Relationship between Base Classes and Derived Classes 764
20.4.1 Creating and Using a CommissionEmployee Class 765
20.4.2 Creating a BasePlusCommissionEmployee Class Without Using Inheritance 770
20.4.3 Creating a CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy 775
20.4.4 CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data 780
20.4.5 CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using private Data 787
20.5 Constructors and Destructors in Derived Classes 794
20.6 public, protected and private Inheritance 802
20.7 Software Engineering with Inheritance 803
20.8 Wrap-Up 804

21 Object-Oriented Programming: Polymorphism 810

21.1 Introduction 811
21.2 Polymorphism Examples 812
21.3 Relationships Among Objects in an Inheritance Hierarchy 813
21.3.1 Invoking Base-Class Functions from Derived-Class Objects 814
21.3.2 Aiming Derived-Class Pointers at Base-Class Objects 821
21.3.3 Derived-Class Member-Function Calls via Base-Class Pointers 822
21.3.4 Virtual Functions 824
21.3.5 Summary of the Allowed Assignments Between Base-Class and Derived-Class Objects and Pointers 830
21.4 Type Fields and switch Statements 831
21.5 Abstract Classes and Pure virtual Functions 831
21.6 Case Study: Payroll System Using Polymorphism 833
21.6.1 Creating Abstract Base Class Employee 835
21.6.2 Creating Concrete Derived Class SalariedEmployee 838
21.6.3 Creating Concrete Derived Class HourlyEmployee 840
21.6.4 Creating Concrete Derived Class CommissionEmployee 843

21.6.5 Creating Indirect Concrete Derived Class BasePlusCommissionEmployee 845
21.6.6 Demonstrating Polymorphic Processing 846
21.7 (Optional) Polymorphism, Virtual Functions and Dynamic Binding "Under the Hood" 850
21.8 Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, dynamic_cast, typeid and type_info 854
21.9 Virtual Destructors 858
21.10 Wrap-Up 858

22 Templates 864

22.1 Introduction 865
22.2 Function Templates 865
22.3 Overloading Function Templates 869
22.4 Class Templates 869
22.5 Nontype Parameters and Default Types for Class Templates 876
22.6 Notes on Templates and Inheritance 877
22.7 Notes on Templates and Friends 877
22.8 Notes on Templates and static Members 878
22.9 Wrap-Up 878

23 Stream Input/Output 883

23.1 Introduction 884
23.2 Streams 885
23.2.1 Classic Streams vs. Standard Streams 885
23.2.2 iostream Library Header Files 886
23.2.3 Stream Input/Output Classes and Objects 886
23.3 Stream Output 889
23.3.1 Output of char * Variables 889
23.3.2 Character Output Using Member Function put 889
23.4 Stream Input 890
23.4.1 get and getline Member Functions 890
23.4.2 istream Member Functions peek, putback and ignore 893
23.4.3 Type-Safe I/O 893
23.5 Unformatted I/O Using read, write and gcount 893
23.6 Introduction to Stream Manipulators 894
23.6.1 Integral Stream Base: dec, oct, hex and setbase 895
23.6.2 Floating-Point Precision (precision, setprecision) 896
23.6.3 Field Width (width, setw) 897
23.6.4 User-Defined Output Stream Manipulators 898
23.7 Stream Format States and Stream Manipulators 900
23.7.1 Trailing Zeros and Decimal Points (showpoint) 900
23.7.2 Justification (left, right and internal) 901
23.7.3 Padding (fill, setfill) 903
23.7.4 Integral Stream Base (dec, oct, hex, showbase) 904

23.7.5 Floating-Point Numbers; Scientific and Fixed Notation (scientific, fixed) 905
23.7.6 Uppercase/Lowercase Control (uppercase) 906
23.7.7 Specifying Boolean Format (boolalpha) 906
23.7.8 Setting and Resetting the Format State via Member Function flags 907
23.8 Stream Error States 909
23.9 Tying an Output Stream to an Input Stream 911
23.10 Wrap-Up 911

24 Exception Handling 921

24.1 Introduction 922
24.2 Exception-Handling Overview 923
24.3 Example: Handling an Attempt to Divide by Zero 923
24.4 When to Use Exception Handling 929
24.5 Rethrowing an Exception 930
24.6 Exception Specifications 932
24.7 Processing Unexpected Exceptions 933
24.8 Stack Unwinding 933
24.9 Constructors, Destructors and Exception Handling 935
24.10 Exceptions and Inheritance 936
24.11 Processing new Failures 936
24.12 Class auto_ptr and Dynamic Memory Allocation 939
24.13 Standard Library Exception Hierarchy 941
24.14 Other Error-Handling Techniques 943
24.15 Wrap-Up 944

A Operator Precedence Charts 951

B ASCII Character Set 955

C Number Systems 956

C.1 Introduction 957
C.2 Abbreviating Binary Numbers as Octal and Hexadecimal Numbers 960
C.3 Converting Octal and Hexadecimal Numbers to Binary Numbers 961
C.4 Converting from Binary, Octal or Hexadecimal to Decimal 961
C.5 Converting from Decimal to Binary, Octal or Hexadecimal 962
C.6 Negative Binary Numbers: Two's Complement Notation 964

D Game Programming: Solving Sudoku 969

D.1 Introduction 969
D.2 Deitel Sudoku Resource Center 970
D.3 Solution Strategies 970

D.4 Programming Sudoku Puzzle Solvers 974
D.5 Generating New Sudoku Puzzles 975
D.6 Conclusion 977

Appendices on the Web 978

Appendices E through I are PDF documents posted online at the book's Companion Website (located at `www.pearsonhighered.com/deitel`).

E Game Programming with the Allegro C Library I

E.1 Introduction II
E.2 Installing Allegro II
E.3 A Simple Allegro Program III
E.4 Simple Graphics: Importing Bitmaps and Blitting IV
E.5 Animation with Double Buffering IX
E.6 Importing and Playing Sounds XVI
E.7 Keyboard Input XX
E.8 Fonts and Displaying Text XXV
E.9 Implementing the Game of Pong XXXI
E.10 Timers in Allegro XXXVII
E.11 The Grabber and Allegro Datafiles XLII
E.12 Other Allegro Capabilities LI
E.13 Allegro Resource Center LII

F Sorting: A Deeper Look LVIII

F.1 Introduction LIX
F.2 Big O Notation LIX
F.3 Selection Sort LX
F.4 Insertion Sort LXIV
F.5 Merge Sort LXVII

G Introduction to C99 LXXVIII

G.1 Introduction LXXIX
G.2 Support for C99 LXXIX
G.3 New C99 Headers LXXX
G.4 // Comments LXXX
G.5 Mixing Declarations and Executable Code LXXXI
G.6 Declaring a Variable in a `for` Statement Header LXXXII
G.7 Designated Initializers and Compound Literals LXXXIV
G.8 Type `bool` LXXXVII
G.9 Implicit `int` in Function Declarations LXXXVIII
G.10 Complex Numbers LXXXIX
G.11 Variable-Length Arrays XC

G.12 The snprintf Function: Helping Avoid Hacker Attacks XCIII
G.13 Additions to the Preprocessor XCV
G.14 Other C99 Features XCVI
G.15 Web Resources XCIX

H Using the Visual Studio Debugger CIV

H.1 Introduction CV
H.2 Breakpoints and the **Continue** Command CV
H.3 **Locals** and **Watch** Windows CIX
H.4 Controlling Execution Using the **Step Into**, **Step Over**, **Step Out** and **Continue** Commands CXII
H.5 **Autos** Window CXIV
H.6 Wrap-Up CXVI

I Using the GNU Debugger CXVIII

I.1 Introduction CXIX
I.2 Breakpoints and the `run`, `stop`, `continue` and `print` Commands CXIX
I.3 `print` and `set` Commands CXXIV
I.4 Controlling Execution Using the `step`, `finish` and `next` Commands CXXVI
I.5 `watch` Command CXXVIII
I.6 Wrap-Up CXXX

Index 979

Preface

Welcome to the C programming language—and to C++, too! This book presents leading-edge computing technologies for students, instructors and software development professionals.

At the heart of the book is the Deitel signature "live-code approach." Concepts are presented in the context of complete working programs, rather than in code snippets. Each code example is immediately followed by one or more sample executions. All the source code is available at `www.deitel.com/books/chtp6/`.

We believe that this book and its support materials will give you an informative, interesting, challenging and entertaining introduction to C.

As you read the book, if you have questions, send an e-mail to `deitel@deitel.com`; we'll respond promptly. For updates on this book and its supporting C and C++ software, and for the latest news on all Deitel publications and services, visit `www.deitel.com`.

New and Updated Features

Here are the updates we've made for *C How to Program, 6/e*:

- *"Making a Difference" Exercises Set.* We encourage you to use computers and the Internet to research and solve problems that really matter. These new exercises are meant to increase awareness of important issues the world is facing. We hope you'll approach them with your own values, politics and beliefs.
- *Tested All Code on Windows and Linux.* We've tested every program (the examples and the exercises) using both Visual C++ 2008 and GNU GCC 4.3. The code examples and exercise code solutions were also tested using Visual Studio 2010 Beta.
- *New Design.* The book has a new interior design that graphically serves to organize, clarify and highlight the information, and enhances the book's pedagogy.
- *Improved Terminology Sections.* We've added page numbers for the defining occurrences of all terms in the terminology lists for easy reference.
- *Updated Coverage of C++ and Object-Oriented Programming.* We updated Chapters 15–24 on object-oriented programming in C++ with material from our just published *C++ How to Program, 7/e*.
- *Titled Programming Exercises.* We've titled all the programming exercises. This helps instructors tune assignments for their classes.
- *New Web Appendices.* Chapters 15–17 from the previous edition are now searchable PDF Appendices E–G, available on the Companion Website (see the access card at the front of the book).