

Peter A. Darnell Philip E. Margolis

Software Engineering in C

C 语言及软件工程 [英]

Springer-Verlag
World Publishing Corp

Peter R. Dornell Philip E. Margolis

Software Engineering in C

Springer-Verlag
World Publishing Corp

Peter A. Darnell
Senior Software Engineer
Stellar Computer Inc.
95 Wells Avenue
Newton, MA 02159, USA

Philip E. Margolis
Senior Writer
Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824, USA

Library of Congress Cataloging-in-Publication-Data
Margolis, Philip E.

Software engineering in C.
(Springer books on professional computing)
Bibliography: p.
Includes index.

1. Computer software--Development. 2. C (Computer
program language) I. Darnell, Peter. II. Title.
III. Series.

QA76.76.D47M364 1988 005.13'3 87-27515

© 1988 by Springer-Verlag New York Inc.

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer-Verlag, 175 Fifth Avenue, New York, NY 10010, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use of general descriptive names, trade names, trademarks, etc. in this publication, even if the former are not especially identified, is not to be taken as a sign that such names, as understood by the Trade Marks and Merchandise Marks Act, may accordingly be used freely by anyone.

Text prepared by the authors using Interleaf/Apollo DN 3000 workstation.

Reprinted by World Publishing Corporation, Beijing, 1990
for distribution and sale in The People's Republic of China only
ISBN 7-5062-0784-2

ISBN 0-387-96574-2 Springer-Verlag New York Berlin Heidelberg
ISBN 3-540-96574-2 Springer-Verlag Berlin Heidelberg New York

Preface

This book describes the C programming language and software engineering principles of program construction. The book is intended primarily as a textbook for beginning and intermediate C programmers. It does not assume previous knowledge of C, nor of any high-level language, though it does assume that the reader has some familiarity with computers. While not essential, knowledge of another programming language will certainly help in mastering C.

Although the subject matter of this book is the C language, the emphasis is on software engineering—making programs readable, maintainable, portable, and efficient. One of our main goals is to impress upon readers that there is a huge difference between programs that merely work, and programs that are well-engineered, just as there is a huge difference between a log thrown over a river and a well-engineered bridge.

The book is organized linearly so that each chapter builds on information provided in the previous chapters. Consequently, the book will be most effective if chapters are read sequentially. Readers with some experience in C, however, may find it more useful to consult the table of contents and index to find sections of particular interest.

Each chapter is autonomous inasmuch as it covers a single, well-defined area of the C language, such as scalar data types or control flow. Moreover, the chapters themselves are organized linearly, so that each section uses information provided in earlier sections. Again, experienced C programmers may want to skim introductory sections.

Although this book covers all C features, it makes no claim to being a reference manual. The organization and pace is designed for those learning the language rather than those who already know the language. If you plan to do extensive programming in C, we recommend that you supplement our book with *C: A Reference Manual*, by Harbison and Steele.

This book describes all features of the C language defined by Kernighan and Ritchie (known as the K&R standard), as well as all features defined in the C Standard proposed by the American National Standards Institute (ANSI). Where the two versions differ, we highlight the difference either by explicitly describing each version in the text or by describing the ANSI feature in a shaded box. A list of differences between the two standards appears in Appendix E. For more information about the ANSI Standard, you should read the official specification, which you can obtain by writing to:

American National Standards Institute
1430 Broadway
New York, NY 10018

In addition to using shaded boxes to describe ANSI extensions, we also use boxes to highlight common errors made by C programmers. These “Bug Alerts” are intended as buoys to mark places where we and others have run aground.

The examples in this book have all been tested on three machines: A PC-compatible Zenith Z-151 running the Microsoft Version 3.0 C compiler, an Apollo DN3000 running the DOMAIN C compiler (Version 4.78), and a Sun Microsystems 3/50 computer running Version 3.1 of the Sun compiler. Whenever possible, we have tried to use realistic examples gleaned from our own experiences. Occasionally we refer to “our machine”, which means any of these three computers. The most significant aspect of “our machine” is that it allocates four bytes for ints.

Appendix A describes all of the runtime library functions defined in the ANSI standard. Many of these functions are derived from UNIX functions and are present in current C runtime libraries. Be careful, though, because some ANSI functions behave differently from identically-named functions in older libraries.

Appendix B shows the syntax of the ANSI C language in the form of “railroad diagrams.” Each rectangular box in a diagram represents another diagram defined elsewhere. Items that appear in ovals are C keywords and predefined names that must appear exactly as they are written. Circles are used to represent punctuation tokens. Unless stated otherwise, it is always legal to insert spaces and newlines between one item and another.

Appendix C lists all names reserved by the ANSI standard. This includes keywords, library function names, and type definitions used by the li-

brary. You should avoid declaring variables that conflict with these names.

Appendix D lists certain ranges that an ANSI-conforming compiler must support. This includes, for example, the range of values that must be representable in a floating-point number.

Appendix E lists the major differences between the ANSI Standard and the K&R standard. Each entry in this list contains a reference to the section in the book where the difference is described. Note that this list is not exhaustive.

Appendix F contains the source listings for a C interpreter. In Chapter 12, we refer to this program as an example of using good engineering techniques to produce a large software product.

Acknowledgements

First and foremost, we wish to acknowledge our debt to the authors of the two most influential books about C: Samuel Harbison, Brian Kernighan, Dennis Ritchie, and Guy Steele.

In addition to the books by these authors, we also leaned heavily on the Draft Proposed ANSI Standard, and we thank all of the members of the ANSI X3J11 Subcommittee for their efforts in creating this document.

Many people reviewed various parts of this book at various stages. We are indebted to all of them, particularly David Boyce, Gary Bray, Clem Cole, Karen Darnell, Norman Garfinkle, John Humphrys, Ben Kingsbury, Diane Margolis, Doug McGlathery, Beth O'Connell, John Peyton, Bill Plauger, Barry Rosenberg, Jim Van Sciver, Kincade Webb, Bob Weir, and John Weiss. We are also indebted to the software development team at Dynatech Data Systems, especially Elizabeth Stark and Jonathan Edney. Special thanks go to Chuck Connell, Sam Harbison, and Tom Pennello, who read the manuscript in its entirety and offered numerous invaluable suggestions. We would also like to thank Kathy Ford for her assistance in preparing the artwork, and Andrea Morris for her expert editorial advice. Naturally, we accept responsibility for any flaws that remain.

Finally we would like to thank Apollo Computer Inc. and Stellar Computer Inc. for providing the working environments in which to produce this book. The entire book was formatted using the Interleaf Version 3.0 electronic publishing system running on an Apollo DN3000 workstation.

Suggested Reading

We have found the following books extremely helpful in mastering C and in absorbing general software engineering principles.

Aho, Alfred V., and Jeffrey P. Ullman. *Principles of Compiler Design*. Addison-Wesley, 1972.

Brooks, Frederick P., Jr. *The Mythical Man Month: Essays on Software Engineering*. Addison-Wesley, 1974.

Date, C. J. *An Introduction to Database Systems*. 4th ed. Addison-Wesley, 1986.

Foley, J. D., and A. Van Dam. *Fundamentals of Computer Graphics*. Addison-Wesley, 1980.

Harbison, Samuel P., and Guy L. Steele Jr. *C: A Reference Manual*. 2d ed. Prentice Hall, 1984.

Kernighan, Brian W., and P. J. Plauger. *Software Tools*. Addison-Wesley, 1976.

Kernighan, Brian W., and P. J. Plauger. *Elements of Programming Style*. McGraw-Hill, 1978.

Kernighan, Brian W., and Dennis M. Ritchie. *The C Programming Language*. Prentice-Hall, 1978.

Knuth, Donald E. *The Art of Computer Programming*. Addison-Wesley, 1973.

Shore, John. *The Sacher Torte Algorithm*. Penguin Books, 1986.

Table of Contents

Chapter 1

Introduction To Programming	1
1.1 High-Level Programming Languages	3
1.2 History of C	5
1.3 ANSI Standard	6
1.4 Nature of C	8

Chapter 2

C Essentials	9
2.1 Program Development	9
2.2 Functions	14
2.3 Anatomy of a C Function	18
2.4 Formatting Source Files	26
2.5 The <i>main()</i> Function	28
2.6 The <i>printf()</i> Function	31
2.7 The <i>scanf()</i> Function	33
2.8 The Preprocessor	34
Exercises	38

Chapter 3

Scalar Data Types	39
3.1 Declarations	40
3.2 Different Types of Integers	42
3.3 Different Kinds of Integer Constants	47
3.4 Floating-Point Types	52
3.5 Initialization	55
3.6 Finding the Address of an Object	56
3.7 Introduction to Pointers	57
3.8 Typedefs	61
3.9 Mixing Types	63
3.10 Explicit Conversions — Casts	72
3.11 Enumeration Types	73
3.12 The void Data Type	74
Exercises	76

Chapter 4

Control Flow	78
4.1 Conditional Branching	79
4.2 The switch Statement	89
4.3 Looping	95
4.4 Nested Loops	107
4.5 A Simple Calculator Program	110
4.6 The break and continue Statements	111
4.7 The goto Statement	113
4.8 Infinite Loops	114
Exercises	116

Chapter 5

Operators and Expressions	118
5.1 Precedence and Associativity	121
5.2 Unary Minus Operator	124
5.3 Binary Arithmetic Operators	125
5.4 Arithmetic Assignment Operators	128
5.5 Increment and Decrement Operators	132
5.6 Comma Operator	137
5.7 Relational Operators	138
5.8 Logical Operators	140
5.9 Bit-Manipulation Operators	144
5.10 Bitwise Assignment Operators	152
5.11 Cast Operator	152
5.12 sizeof operator	154
5.13 Conditional Operator (? :)	155
5.14 Memory Operators	156
Exercises	157

Chapter 6

Arrays and Pointers	159
6.1 Declaring an Array	160
6.2 How Arrays Are Stored in Memory	162
6.3 Initializing Arrays	163
6.4 Array Example: Encryption and Decryption	166
6.5 Pointer Arithmetic	168
6.6 Passing Pointers as Function Arguments	169
6.7 Accessing Array Elements Through Pointers	172

6.8	Passing Arrays as Function Arguments	174
6.9	Sorting Algorithms	176
6.10	Strings	179
6.11	Multidimensional Arrays	195
6.12	Arrays of Pointers	202
6.13	Pointers to Pointers	206
	Exercises	212

Chapter 7

	Storage Classes	215
7.1	Fixed vs. Automatic Duration	216
7.2	Scope	220
7.3	Global Variables	225
7.4	The register Specifier	230
7.5	Summary of Storage Classes	235
7.6	Dynamic Memory Allocation	237
	Exercises	241

Chapter 8

	Structures and Unions	243
8.1	Structures	243
8.2	Linked Lists	264
8.3	Unions	271
8.4	enum Declarations	279
	Exercises	280

Chapter 9

	Functions	281
9.1	Passing Arguments	281
9.2	Declarations and Calls	284
9.3	Pointers to Functions	295
9.4	Recursion	306
9.5	The <i>main()</i> Function	309
9.6	Complex Declarations	310
	Exercises	314

Chapter 10

The C Preprocessor	316
10.1 Macro Substitution	317
10.2 Conditional Compilation	330
10.3 Include Facility	335
10.4 Line Control	336
Exercises	339

Chapter 11

Input and Output	340
11.1 Streams	341
11.2 Buffering	343
11.3 The <code><stdio.h></code> Header File	345
11.4 Error Handling	345
11.5 Opening and Closing a File	347
11.6 Reading and Writing Data	350
11.7 Selecting an I/O Method	357
11.8 Unbuffered I/O	359
11.9 Random Access	360
Exercises	372

Chapter 12

Software Engineering	373
12.1 Product Specification	376
12.2 Software Design	382
12.3 Project Management and Cost Estimation	389
12.4 Software Tools for Software Production	395
12.5 Debugging	397
12.6 Testing	400
12.7 Performance Analysis	401
12.8 Documentation	401
Exercises	403

Appendix A The ANSI Runtime Library 405

A.1 Function Names	406
A.2 Header Files	406
A.3 Synopses	407
A.4 Functions vs. Macros	408

A.5 Error Handling	409
A.6 Diagnostics	410
A.7 Character Handling	411
A.8 Setting Locale Parameters	413
A.9 Mathematics	415
A.10 Non-Local Jumps	420
A.11 Signal Handling	422
A.12 Variable Argument Lists	425
A.13 I/O Functions	427
A.14 General Utilities	455
A.15 String-Handling Functions	465
A.16 Date and Time Functions	472
 Appendix B Syntax of ANSI C	 478
 Appendix C Implementation Limits	 495
C.1 Translation Limits	495
C.2 Numerical Limits	496
 Appendix D Differences Between the ANSI and K&R Standards	 500
D.1 Source Translation Differences	500
D.2 Data Type Differences	502
D.3 Statement Differences	505
D.4 Expression Differences	505
D.5 Storage Class and Initialization Differences	507
D.6 Preprocessor Differences	509
 Appendix E Reserved Names	 512
 Appendix F C Interpreter Listing	 519
 Appendix G ASCII Codes	 584
 Index	 586

List of Figures

Figure 1-1. Language Spectrum	2
Figure 1-2. Different Compilers for Different Machines	4
Figure 2-1. Stages of Program Development	10
Figure 2-2. Compiling and Linking.	11
Figure 2-3. Software Hierarchy	15
Figure 2-4. Elements of a Function	18
Figure 2-5. Anatomy of the <i>square()</i> Function	19
Figure 2-6. Memory after $j = 5 + 10$	22
Figure 2-7. Syntax of an Assignment Statement	26
Figure 3-1. Hierarchy of C Data Types	40
Figure 3-2. Dereferencing a Pointer Variable	59
Figure 3-3. Hierarchy of C Scalar Data Types	65
Figure 4-1. Syntax of an if...else Statement	79
Figure 4-2. Braces Ensure Correct Control Flow	86
Figure 4-3. Logic of a Nested if Statement	88
Figure 4-4. Syntax of a switch Statement	91
Figure 4-5. Syntax of a while Statement	95
Figure 4-6. Flow Control of a while Statement	96
Figure 4-7. Syntax of a do...while Statement	98
Figure 4-8. Syntax of a for Statement	99
Figure 5-1. Evaluation of an Expression Enclosed By Parentheses	122
Figure 5-2. Representation of an Expression as an Inverted BinaryTree	123
Figure 5-3. Arithmetic Assignment Operator Equivalences	131
Figure 6-1. Syntax of an Array Declaration	160
Figure 6-2. Storage of an Array	162
Figure 6-3. Initialization of Arrays	164
Figure 6-4. Passing the Wrong Pointer Type	171
Figure 6-5. Storage of a String	181
Figure 6-6. Storage of a Multidimensional Array	196
Figure 6-7. Array of Pointers	203
Figure 6-8. Storage of an Array of Pointers to Strings	205
Figure 6-9. A Pointer to a Pointer	207
Figure 7-1. Hierarchy of Active Regions (Scopes)	221
Figure 8-1. Memory Storage for the vs Structure	245
Figure 8-2. Allocation Without Alignment Restrictions	255
Figure 8-3. Allocation with Alignment Restrictions	255

Figure 8-4. Syntax of Bit Field Declarations	256
Figure 8-5. Storage of Three Consecutive Bit Fields	257
Figure 8-6. Alternative Storage of Three Consecutive Bit Fields.	257
Figure 8-7. Storage of Two Consecutive Bit Fields	258
Figure 8-8. Storage of the <i>DATE</i> Structure with Bit Fields	259
Figure 8-9. Alternative Storage of the <i>DATE</i> Structure with Bit Fields	259
Figure 8-10. A Singly Linked List	265
Figure 8-11. Linked-List Insertion	269
Figure 8-12. Linked-List Deletion	269
Figure 8-13. Example of Union Memory Storage	271
Figure 8-14. Storage in example Union After Assignment	272
Figure 9-1. Pass By Reference vs. Pass By Value	282
Figure 9-2. Syntax of a Function Definition	285
Figure 9-3. Syntax of a return Statement	287
Figure 9-4. Syntax of a Function Allusion	289
Figure 9-5. Syntax of a Function Call	291
Figure 9-6. Recursion	308
Figure 10-1. Syntax of a Function-Like Macro	320
Figure 10-2. Syntax of Conditional Compilation Directives	330
Figure 10-3. Syntax of the #line Directive	336
Figure 11-1. Streams.	341
Figure 12-1. Balanced Binary Tree Implementation of a Symbol Table	385
Figure 12-2. Inbalanced Binary Tree Implementation of a Symbol Table	386
Figure 12-3. Typical Software Development Curve	391
Figure 12-4. Time Lines and Milestones for the cint Project. ...	393
Figure 12-5. Directory Structure for C Interpreter Project Containing Debug and Edit Subsystems	394

List of Tables

Table 2-1. Legal and Illegal Variable Names	23
Table 2-2. Reserved C Keywords	23
Table 3-1. Scalar Type Keywords	41
Table 3-2. Size and Range of Integer Types on Our Machine. .	44
Table 3-3. Integer Constants	48
Table 3-4. Types of Integral Constants	49
Table 3-5. C Escape Sequences	50
Table 3-6. ANSI Trigraph Sequences	51
Table 3-7. Legal and Illegal Floating-Point Constants	54
Table 4-1. Relational Operators	82
Table 4-2. Relational Expressions	83
Table 5-1. Precedence and Associativity of C Operators	120
Table 5-2. Unary Arithmetic Operators	124
Table 5-3. Binary Arithmetic Operators	125
Table 5-4. Examples of Expressions Using Arithmetic Operators.	126
Table 5-5. Arithmetic Assignment Operators	128
Table 5-6. Examples of Expressions Using Arithmetic Assignment Operators	132
Table 5-7. The Increment and Decrement Operators	133
Table 5-8. Examples of Expressions Using the Increment and Decrement Operators	135
Table 5-9. The Comma Operator	137
Table 5-10. The Relational Operators	138
Table 5-11. Examples of Expressions Using the Relational Operators	139
Table 5-12. The Logical Operators	140
Table 5-13. Truth Table for C's Logical Operators.	141
Table 5-14. Examples of Expressions Using the Logical Operators	142
Table 5-15. The Bit-Manipulation Operators	144
Table 5-16. Examples Using the Shift Operators	145
Table 5-17. Shifting Negative Numbers	145
Table 5-18. Decimal, Hexadecimal, Binary, and Octal Versions of the Integers Zero Through 15	147
Table 5-19. The Bitwise AND Operator	147
Table 5-20. Examples Using the Bitwise Inclusive OR Operator .	147
Table 5-21. Example Using the XOR Operator	148
Table 5-22. Example Using the Bitwise Complement Operator ..	148
Table 5-23. The Bitwise Assignment Operators	152

Table 5-24. The Cast Operator	152
Table 5-25. The <code>sizeof</code> Operator	154
Table 5-26. The Conditional Operator	155
Table 5-27. The Memory Operators	156
Table 6-1. String Functions in the Standard Library	194
Table 7-1. Semantics of Storage Class Specifiers	236
Table 9-1. Legal and Illegal Declarations in C	313
Table 11-1. <code>fopen()</code> Text Modes	347
Table 11-2. File and Stream Properties of <code>fopen()</code> Modes	348
Table 11-3. I/O to <code>stdin</code> and <code>stdout</code>	369
Table 11-4. I/O to files	369
Table 11-5. Error-Handling Functions	371
Table 11-6. File Management Functions	371
Table 12-1. Summary of Programming Style Issues	375
Table 12-2. List of Modules in the C Interpreter	383
Table A-1. Header Files for the Runtime Library	407
Table A-2. Character Testing Functions	412
Table A-3. Trigonometric and Hyperbolic Functions	417
Table A-4. Signal Handling Macros	423
Table A-5. The <code>fopen()</code> Modes	430
Table A-6. <code>printf()</code> Conversion Characters	438
Table A-7. <code>printf()</code> Flag Characters	441
Table A-8. <code>scanf()</code> Conversion Characters	447
Table A-9. Format Specifiers for the <code>ctime()</code> Function	476
Table E-1. Reserved Names	518

List of Boxes

Box 2-1	Compiling and Linking in a UNIX Environment	12
Box 2-2	The Mailbox Analogy	24
Box 2-3	Bug Alert — No Nested Comments	28
Box 3-1	ANSI Feature — signed Qualifier	45
Box 3-2	ANSI Feature — unsigned Constants	49
Box 3-3	ANSI Feature — Trigraph Sequences	51
Box 3-4	ANSI Feature — long double Type	53
Box 3-5	ANSI Feature — float and long double constants	54
Box 3-6	Bug Alert — Confusing typedef with #define	62
Box 3-7	ANSI Feature — Unsigned Conversions	68
Box 4-1	Bug Alert — Confusing = with ==	83
Box 4-2	Bug Alert — Missing Braces	87
Box 4-3	Bug Alert — The Dangling else	89
Box 4-4	Bug Alert — Off-By-One Errors	104
Box 4-5	Bug Alert — Misplaced Semicolons	106
Box 5-1	Bug Alert — Integer Division and Remainder	129
Box 5-2	Bug Alert — Side Effects	136
Box 5-3	Bug Alert — Comparing Floating-Point Values	139
Box 5-4	Bug Alert — Side Effects in Relational Expressions	142
Box 6-1	ANSI Feature — Initialization of Arrays	165
Box 6-2	Bug Alert — Walking Off the End of an Array	177
Box 6-3	ANSI Feature — String Concatenation	187
Box 6-4	Bug Alert — Referencing Elements in a Multidimensional Array	199
Box 7-1	ANSI Note — Scope of Function Arguments	224
Box 7-2	Bug Alert — The Dual Meanings of static	225
Box 7-3	Non-ANSI Strategies for Declaring Global Variables	228
Box 7-4	ANSI Feature — The const Storage-Class Modifier	231
Box 7-5	ANSI Feature — The volatile Storage-Class Modifier	233
Box 7-6	ANSI Feature — Generic Pointers	240
Box 8-1	ANSI Feature — struct and union Name Spaces	253
Box 8-2	Bug Alert — Passing Structures vs. Passing Arrays	261
Box 8-3	ANSI Feature — Initializing Unions	275
Box 9-1	ANSI Feature — Function Prototypes	292
Box 10-1	ANSI Feature — Flexible Formatting of Preprocessor Lines	317
Box 10-2	Bug Alert — Ending a Macro Definition With a Semicolon	317
Box 10-3	Bug Alert — Using = to Define a Macro	321
Box 10-4	Bug Alert — Space Between Left Parenthesis and Macro Name	322
Box 10-5	ANSI Feature — Using a Macro Name in Its Own Definition	323
Box 10-6	Bug Alert — Side Effects in Macro Arguments	325
Box 10-7	Bug Alert — Binding of Macro Arguments	325
Box 10-8	ANSI Feature — String Producer	327
Box 10-9	ANSI Feature — Token Pasting	329
Box 10-10	ANSI Feature — The #error Directive	337
Box 10-11	ANSI Feature — The #pragma Directive	338
Box 11-1	Bug Alert — Opening a File	349