

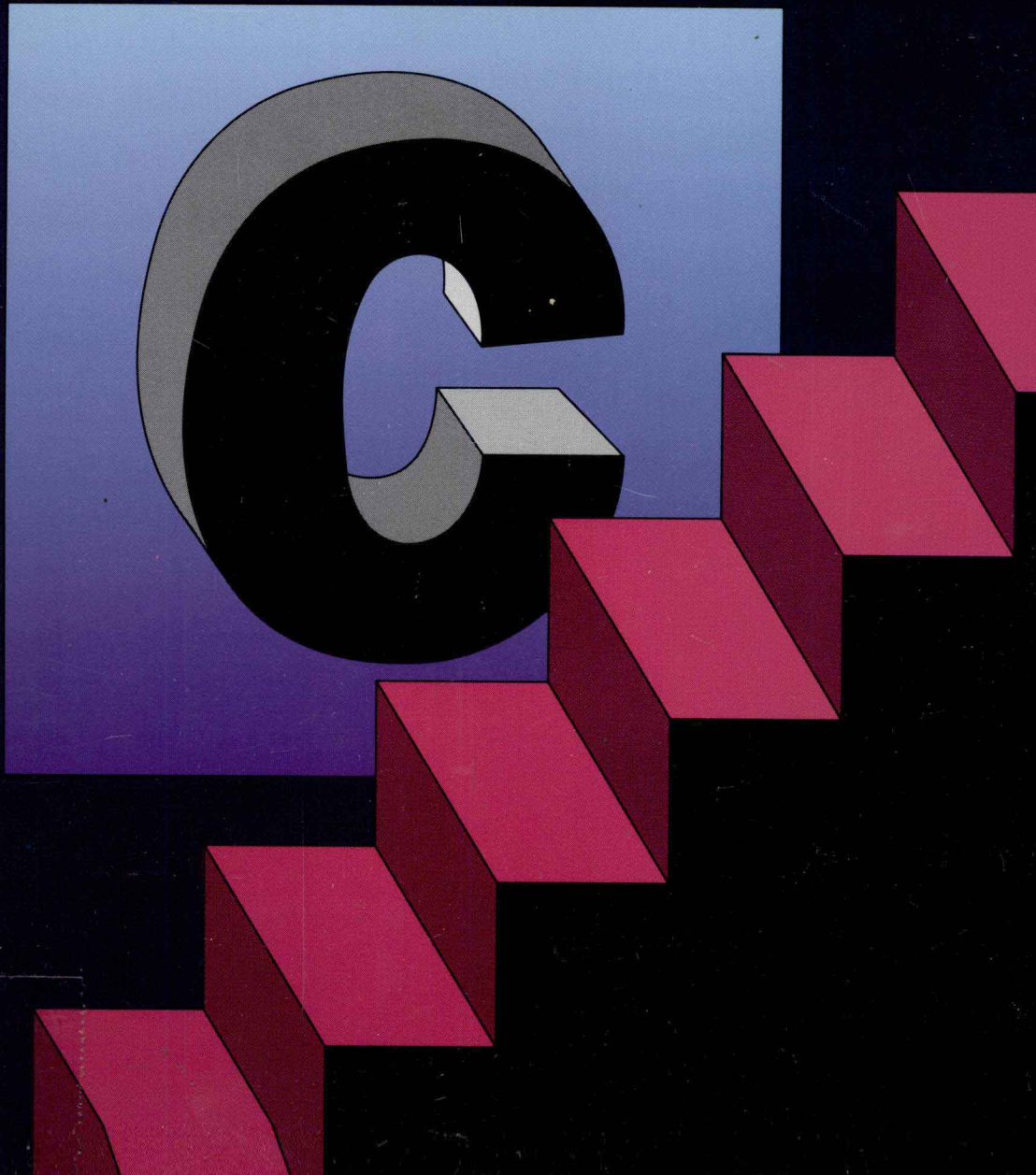
SAMS

The Waite Group®

Mitchell Waite
and Stephen Prata

C: Step-by-Step

Includes
ANSI C



C: Step-by-Step

Mitchell Waite, Stephen Prata
The Waite Group

SAMS

A Division of Prentice Hall Computer Publishing

Dedication

To Benjamin, whose warm and furry four-legged body helped me endure the rigors of writing.

— Mitchell Waite

To my wife, Kathleen.

— Stephen Prata



The text in this book is printed on recycled paper.

© 1989 The Waite Group

97 15 14

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book the publisher assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

International Standard Book Number: 0-672-22651-0

Library of Congress Catalog Card Number: 89-60593

From The Waite Group:

Developmental Editor: *Mitchell Waite*

Managing Editor: *Scott Calamar*

From BMR:

Market Research & Editorial Development: *Susan Nelle*

Production: *Jan Granoff*

Interior Design: *Vigne Design*

Illustrations: *Kit Croucher*

Desktop Publishers: *Melanie Field and Lisa Labrecque*

From SAMS:

Acquisitions & Development Editor: *Richard Swadley*

Production Coordinator: *Marjorie Hopper*

Printed in the United States of America

Preface

When we wrote *C Primer Plus* in 1984, we aimed to create a friendly, easy-to-use, self-study guide to learning and using the C programming language. The book's subsequent success makes us think we met that goal. However, use of *C Primer Plus* has now moved from self study into the classroom as more institutions offer courses in C. This shift prompted us to develop a new edition of the book, one more directed toward the classroom environment. We surveyed instructors who use our book to find out what features they felt a textbook edition should have. Their advice and our own experiences in teaching C in the classroom have led to this book: *C Step by Step*.

Features

We've retained the major features of the original book, for they are what made the book work for many readers. These features include the following:

- The book is an introduction to programming as well as to C. We don't assume you already are proficient in some other language; however, we do assume you are not a complete computer novice. We don't discuss the history of computing or how computers work.
- We emphasize an interactive approach. In computing, you learn by doing. We often use short, easily typed examples to illustrate just one or two concepts at a time. This gives you quick feedback on how specific concepts work.
- We clarify ideas with figures and illustrations.
- We summarize the main C features in boxes highlighted with a screened background. These boxes simplify finding the summaries when you flip through the text.
- We offer occasional tidbits of information and advice in unscreened boxes.
- Each chapter contains review questions and programming exercises.

- We cover a full range of C topics, including pointers, structures, and bitwise operations.

To this tried and true foundation, we've also added several modifications and features:

- There are many more review questions and exercises. These increase your feedback on concept development and skills comprehension.
- We've increased the emphasis on programming skills by discussing structured programming, step-wise refinement, and top-down programming.
- We've increased the number of programming examples to provide more insight into programming design and analysis.
- More attention is given to developing the skills to find and avoid programming errors.
- We've integrated coverage of the ANSI C standard into the text.
- We cover loops and files earlier than before, and we've augmented the treatment of files.
- We've rewritten and expanded explanations, when necessary, to clarify the subject matter.
- The chapters are structured in a more organized fashion. Each begins with a list of contents and a list of objectives. Each ends with a summary, review questions, and exercises.
- We provide answers for half the review questions at the end of the book.

All in all, we believe we've retained the flavor of the original while adding elements that make this version more suitable for classroom use.

Hardware/Software

Nearly all the examples are "generic" C; that is, they're meant to run on any standard C implementation. We've tested the programs on a VAX 11/750 computer running under BSD 4.3 UNIX and on an IBM AT clone using Microsoft 5.1 C, Microsoft QuickC, and Borland Turbo C, all under MS-DOS. We've pointed out places where the program results may be implementation-dependent, that is, where they may depend on the particular hardware or software in use. Occasionally, we do discuss implementation-dependent matters, although we've confined our remarks to UNIX and DOS (PC-DOS or MS-DOS), which are currently the two most common C environments.

Advice to the Student

In general, learning works best as an active, not a passive process. This is particularly true for programming. Therefore, we encourage you to try out the examples in the book. Such practice will give you a better idea of how C works. If you have questions about a program, you can explore them by modifying the example. Be an active, experimenting learner, and you will learn C more quickly and in greater depth.

We wish you good fortune in learning C. We've tried to make this book meet your needs, and we hope it helps you reach your goals.

Acknowledgements

From Mitchell Waite

I would like to take this opportunity to thank the people who have helped to make *C: Step-by-Step* a reality.

First, I would like to thank Stephen Prata for his continued support and flawless writing, and ability to endure my demands for perfection.

I would like to thank the numerous college professors who responded to the original survey for *C Primer Plus*, and also those who reviewed *C: Step-by-Step's* 1200 manuscript pages. You gave us valuable feedback. Scott Calamar of The Waite Group was critical in pulling the final manuscript together and coordinating with the production company and the author. I would also like to thank Susan Nelle of Business Media Resources for her coordination of the development process and Jane Granoff for managing the production of the book.

Finally I give my thanks to the people behind the scenes at SAMS, who believed in the idea of a college textbook on C: Richard Swadley, who put us in touch with BMR and handled all the pre-sales support for the book, Jim Hill for his faith in the idea, and to all the other people at SAMS who in one way or another are involved in making *C: Step-by-Step* a success.

From Stephen Prata

Many people have helped make this book possible. I wish to thank Susan Nelle of Business Media Resources for surveying what instructors want in a C text and to thank those educators for their responses. I especially wish to thank those who reviewed the first draft: Janet Spears, Myron Kaplan, Kenneth Walker, Eliezer Dekel, David Hale, and Craig Colston. I wish to thank Scott Calamar of The Waite Group, and Jane Granoff of Business Media Resources for guiding the book through production.

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks are listed below. In addition, terms suspected of being trademarks or service marks have been appropriately capitalized. SAMS cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Apple is a registered trademark of Apple Computer, Inc.

IBM is a registered trademark of International Business Machines Corp.

Lifeboat is a trademark of Lifeboat Associates, Inc.

Macintosh is a trademark of McIntosh Laboratory, Inc., and is used by Apple Computer, Inc., with express permission of its owner.

Microsoft is a registered trademark of Microsoft Corporation.

MS-DOS is a registered trademark of Microsoft Corporation.

UNIX is a trademark of AT&T Bell Laboratories.

VAX is a registered trademark of Digital Equipment Corp.

WordPerfect is registered trademark of WordPerfect Corporation.

WordStar is a registered trademark of MicroPro International Corporation.

Table of Contents

Preface	xvii
Acknowledgements	xx
Trademarks	xxi
1 C and Programming	1
Contents	1
Objectives	1
1.1 C History	2
1.2 C Virtues	2
Design Features	2
Efficiency	3
Portability	3
Power and Flexibility	3
Programmer Orientation	3
Potentially Disadvantageous Disvirtues (Faults)	4
1.3 C Trends	4
1.4 Using C: Seven Steps	4
Step 1: Define the Program Objectives	5
Step 2: Design the Program	5
Step 3: Write the Code	6
Step 4: Compile the Code	7
Step 5: Run the Program	7
Step 6: Test and Debug the Program	7
Step 7: Maintain and Modify the Program	8
Commentary for Beginners	8
1.5 A Design Example	9
Defining the Program Objectives	9
Designing the Program	9
Writing the Program	11
Structured Programming	12
1.6 Programming Mechanics	12
Source Code Files	12
Object Code Files, Executable Files, and Libraries	13
Preparing a C Program on a UNIX System	14
Preparing a C Program on an IBM PC (Microsoft C)	16
Preparing a C Program on an IBM PC (Turbo C and Quick C)	17
Why Compile?	18

1.7 Language Standards	19
1.8 Some Typographic Conventions	19
Typeface	19
Screen Output	20
Input and Output Devices	20
1.9 Summary	20
Review Questions	21
Programming Exercises	21
2 Introducing C	21
Contents	23
Objectives	23
2.1 A Simple Sample of C	24
2.2 The Explanation	25
Pass 1: Quick Synopsis	25
Pass 2: Details	27
2.3 The Structure of a Simple Program	32
2.4 Tips on Making Your Programs Readable	32
2.5 Taking Another Step	34
Documentation	34
Multiple Declarations	34
Multiplication	35
Printing Multiple Values	35
2.6 Multiple Functions	35
2.7 Debugging	36
Syntax Errors	36
Semantic Errors	37
Program State	39
2.8 Keywords	39
2.9 Summary	39
Review Questions	40
Programming Exercises	42
3 Data and C	45
Contents	45
Objectives	45
3.1 A Sample Program	46
What's New in This Program	47
3.2 Data: Variables and Constants	48
3.3 Data: Data Types	49
Integer versus Floating-Point Types	49
The Integer	49
The Floating-Point Number	49
3.4 C Data Types	52
The <code>int</code> Type	52

Other Integer Types	55
Using Characters: Type char	59
Types float and double	63
Other Types	66
Type Sizes	67
3.5 Using Data Types	69
3.6 I/O Arguments and Pitfalls	70
3.7 One More Example	71
What Happens	71
A Possible Problem	72
3.8 Summary	73
Review Questions	74
Programming Exercises	75
4 Character Strings and Formatted Input/Output	77
Contents	77
Objectives	77
4.1 Introductory Program	78
4.2 Character Strings—An Introduction	79
Type char Arrays and the Null Character	79
Using Strings	80
String Length— strlen()	81
4.3 Constants and the C Preprocessor	83
Using #define and #include Together	86
C—A Master of Disguise: Creating Aliases	87
4.4 Exploring and Exploiting printf() and scanf()	88
printf()	89
Using printf()	89
Conversion Specification Modifiers for printf()	92
The Meaning of Conversion	96
Using scanf()	102
The scanf() View of Input	104
The * Modifier with printf() and scanf()	105
4.5 Usage Tips	107
4.6 Summary	108
Review Questions	109
Programming Exercises	111
5 Operators, Expressions, and Statements	113
Contents	113
Objectives	113
5.1 Introducing Loops	114
5.2 Fundamental Operators	116
Assignment Operator: =	116
Addition Operator: +	118

Subtraction Operator: -	.118
Sign Operators: - and +	.118
Multiplication Operator: *	.119
Division Operator: /	.121
Operator Precedence	.122
Precedence and the Order of Evaluation	.124
5.3 Some Additional Operators	.126
The sizeof Operator	.126
Modulus Operator: %	.126
The Incrementing Operator: ++	.128
The Decrementing Operator: --	.131
Precedence	.132
Don't Be Too Clever	.133
5.4 Expressions and Statements	.134
Expressions	.134
Statements	.135
Compound Statements (Blocks)	.136
5.5 Type Conversions	.138
The Cast Operator	.141
5.6 Function Arguments and Type Conversions	.142
5.7 An Example Program	.143
5.8 Summary	.145
Review Questions	.146
Programming Exercises	.149

6 C Control Statements: Looping **151**

Contents	.151
Objectives	.151
6.1 An Initial Example	.152
Program Comments	.153
C-Style Reading Loop	.154
6.2 The while Statement	.155
Terminating a while Loop	.155
When a Loop Terminates	.156
while : An Entry-Condition Loop	.157
Syntax Points	.157
6.3 Which Is Bigger: Using Relational Operators and Expressions	.159
What Is Truth?	.160
So What Else Is True?	.161
Troubles with Truth	.162
Precedence of Relational Operators	.164
6.4 Indefinite Loops and Counting Loops	.166
6.5 The for Loop	.167
Using for for Flexibility	.168
The Comma Operator	.174

Zen0 Meets the for Loop	176
6.6 An Exit-Condition Loop: do while	178
6.7 Which Loop?	180
6.8 Nested Loops	181
Discussion	182
A Nested Variation	182
6.9 Arrays	183
Using a for Loop with an Array	184
6.10 A Loop Example using a Function Return Value	186
Program Discussion	188
Using Functions with Return Values	189
6.11 Summary	190
Review Questions	190
Programming Exercises	194
7 C Control Statements: Branching and Jumps	197
Contents	197
Objectives	197
7.1 The if Statement	198
if Basics	199
7.2 Adding else to the if Statement	200
Another Example: Introducing getchar() and putchar()	201
Multiple Choice: else if	204
Pairing elses with ifs	207
More Nested ifs	207
7.3 Let's Get Logical	212
Precedence	213
Order of Evaluation	214
7.4 A Word-Count Program	215
7.5 The Conditional Operator: ?	217
7.6 Multiple Choice: switch and break	220
Using the switch Statement	221
Reading Only the First Character of a Line	223
Multiple Labels	223
switch and if else	225
7.7 Other Control Statements: break , continue , goto	225
The break Statement	225
The continue Statement	226
The goto Statement	228
7.8 Summary	232
Review Questions	232
Programming Exercises	236
8 Character Input/Output and Redirection	239
Contents	239
Objectives	239

8.1 Single-Character I/O: getchar() and putchar()	.240
8.2 Buffers	.241
8.3 Terminating Keyboard Input	.242
Files, Streams, and Keyboard Input	.242
The End of File	.244
8.4 Redirection and Files	.246
UNIX and DOS Redirection	.247
Comment	.250
8.5 A Graphic Example	.250
8.6 Creating a More Friendly User Interface	.252
Working with Buffered Input	.252
Mixing Numeric and Character Input	.255
8.7 Character Sketches	.258
Analyzing the Program	.260
8.8 Menu Browsing	.263
Tasks	.263
Toward a Smoother Execution	.263
Mixing Character and Numeric Input	.265
8.9 Summary	.268
Review Questions	.269
Programming Exercises	.270

9 Functions 273

Contents	.273
Objectives	.273
9.1 Review	.274
Creating and Using a Simple Function	.275
Function Arguments	.278
Defining a Function with an Argument: Formal Arguments	.279
Calling a Function with an Argument: Actual Arguments	.280
The Black-Box Viewpoint	.281
Returning a Value from a Function with return	.281
Function Types	.283
9.2 ANSI C Function Prototyping	.285
The Problem	.285
The ANSI Solution	.286
No Arguments and Unspecified Arguments	.287
ANSI-Style Function Definitions	.288
9.3 Finding Addresses: The & Operator	.288
9.4 Altering Variables in the Calling Program	.290
9.5 Pointers: A First Look	.292
The Indirection Operator: *	.292
Declaring Pointers	.293
Using Pointers to Communicate Between Functions	.294
9.6 Recursion	.297

Recursion Revealed	297
Recursion Fundamentals	299
Tail Recursion	300
Recursion and Reversal	301
9.7 All C Functions are Created Equal	304
9.8 Compiling Programs with Two or More Functions	305
UNIX	306
Microsoft C 4.0.–5.1	306
QuickC	306
Turbo C	306
Using Header Files	306
9.9 Summary	310
Review Questions	310
Programming Exercises	311

10 Arrays and Pointers

313

Contents	313
Objectives	313
10.1 Arrays	314
Initialization and Storage Classes	314
More Array Initialization	316
Assigning Array Values	319
10.2 Pointers to Arrays	320
10.3 Functions, Arrays, and Pointers	323
Array Names as Arguments	324
Using Pointer Arguments	325
Pointers and Arrays: A Comment	326
10.4 Pointer Operations	327
10.5 Another Example	329
10.6 Multidimensional Arrays	330
Computation Scheme	333
Initializing a Two-Dimensional Array	333
10.7 Pointers and Multidimensional Arrays	334
Functions and Multidimensional Arrays	337
10.8 Planning a Program	42
General Plan	342
The read_array() Function	343
The show_array() Function	344
The mean() Function	345
The Result	346
10.9 Summary	347
Review Questions	348
Programming Exercises	350

11 Character Strings and String Functions

353

Contents	.353
Objectives	.353
11.1 Defining Strings within a Program	.355
Character String Constants	.355
Character String Arrays and Initialization	.356
Array Versus Pointer	.357
Specifying Storage Explicitly	.359
Arrays of Character Strings	.360
Pointers and Strings	.361
11.2 String Input	.362
Creating Space	.362
The gets() Function	.362
The scanf() Function	.365
11.3 String Output	.366
The puts() Function	.367
The printf() Function	.367
11.4 The Do-It-Yourself Option	.368
11.5 String Functions	.370
The strlen() Function	.370
The strcat() Function	.371
The strcmp() Function	.373
The strcpy() Function	.376
The sprintf() Function	.377
Other String Functions	.379
11.6 A String Example: Sorting Strings	.379
Sorting	.381
11.7 The ctype.h Character Functions	.382
11.8 Command-Line Arguments	.383
Command-Line Arguments in Integrated Environments	.385
Command-Line Options	.385
11.9 String-to-Number Conversions	.386
11.10 Summary	.388
Review Questions	.389
Programming Exercises	.392

12 File Input/Output

395

Contents	.395
Objectives	.395
12.1 Communicating with Files	.396
What Is a File?	.396
Levels of I/O	.397
Standard Files	.398
12.2 Low-Level I/O	.398
Berkeley (BSD) UNIX	.400

MS-DOS	400
Standard File Descriptors	401
Using a Buffer	401
Performance Considerations	402
Commentary	403
12.3 Standard I/O	403
The fopen() Function	404
The getc() and putc() Functions	405
The fclose() Function	405
Standard Files	406
12.4 A Simple File-Condensing Program	406
12.5 File I/O: fprintf() , fscanf() , fgets() , and fputs()	408
The fprintf() and fscanf() Functions	408
The fgets() and fputs() Functions	409
12.6 Random Access: fseek() and ftell()	411
How fseek() and ftell() Work	412
Binary Mode Versus Text Mode	413
Portability	414
12.7 Using Random Access in a Text Mode	414
12.8 Behind the Scenes with Standard I/O	416
Comment	417
12.9 Other Standard I/O Functions	417
int ungetc(int c, FILE *fp)	417
int fflush(FILE *fp)	418
int setvbuf(FILE *fp, char *buf, int mode, size_t size)	418
Binary I/O: fread() and fwrite()	418
size_t fwrite(void *ptr, size_t size, size_t nmemb, FILE *fp)	420
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *fp)	420
int feof(FILE *fp) and int ferror(FILE *fp)	421
An Example	421
12.10 Summary	424
Review Questions	424
Programming Exercises	426

13 Storage Classes and Program Development 429

Contents	429
Objectives	429
13.1 Storage Classes and Scope	430
Automatic Variables	431
External Variables	432
Definitions and Declarations	434
Static Variables	435
External Static Variables	436
Multiple Files	437
Scope and Functions	437