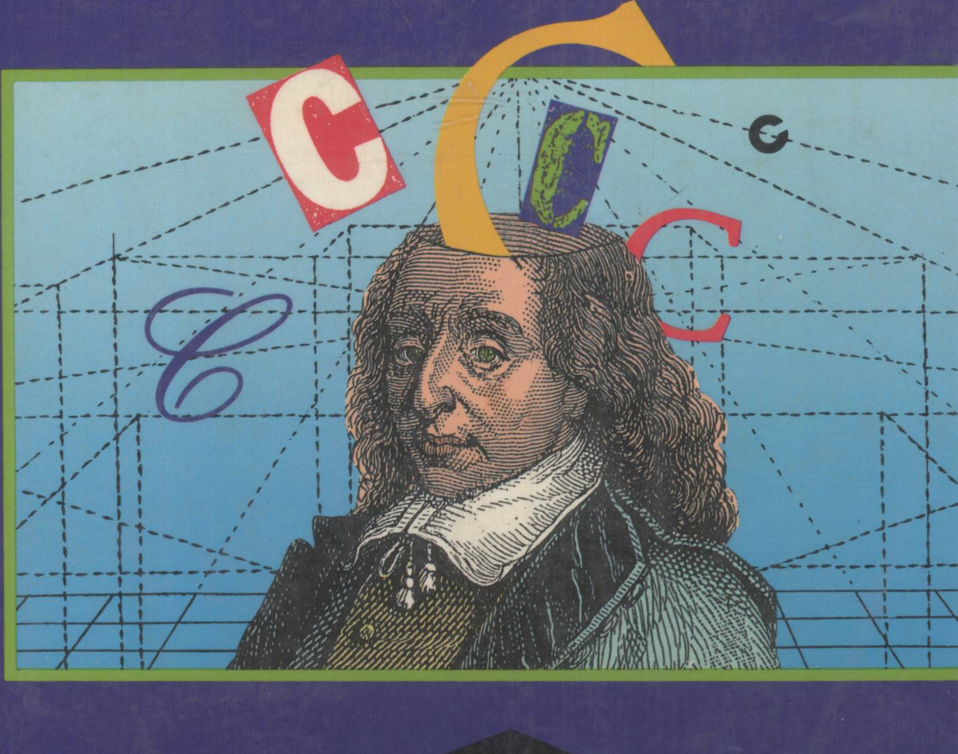
WILEY

INTRODUCING C TO PASCAL PROGRAMMERS



NAMIR SHAMMAS

P312
528

9060021

Introducing C To Pascal Programmers

NAMIR SHAMMAS

E9060021

WILEY

John Wiley & Sons, Inc.
New York • Chichester • Brisbane • Toronto • Singapore

Publisher: Stephen Kippur
Editor: Therese A. Zak
Managing Editor: Corinne McCormick
Electronic Book Production: Publishers Network, Morrisville, PA

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If legal advice or other expert assistance is required, the services of a competent proffesional person should be sought. FROM A DECLARATION OF PRINCIPLES JOINTLY ADOPTED BY A COMMITTEE OF THE AMERICAN BAR ASSOCIATION AND A COMMITTEE OF PUBLISHERS.

Copyright © 1988 by John Wiley & Sons, Inc.

All rights reserved. Published simultaneously in Canada.

Reproduction or translation of any part of this work beyond that permitted by section 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to the Permission Department, John Wiley & Sons, Inc.

Librry of Congress Cataloging in Publication Data:

Shammas, Namir Clement, 1954–
Introducing C to pascal programmers / Namir Clement Shammas.
p. cm.

ISBN 0–471–609080 (pbk.)
1. C (Computer program language) I. Title.
QA76.73.C1547 1988
005.13'3—dc19

Printed in the United States of America

88 89 10 9 8 7 6 5 4 3 2 1

Introducing C To Pascal Programmers

Related Titles of Interest

Turbo Pascal® DOS Utilities, *Alonso*

Programming with Macintosh Turbo Pascal®, *Swan*

Turbo C® Survival Guide, *Miller and Quilici*

Advanced Turbo C® Programmer's Guide, *Mosich, Shammas, Flamig*

Turbo C® DOS Utilities, *Alonso*

Turbo C® and Quick C® Functions: Building Blocks for Efficient Code, *Barden*

The Turbo Programmer's Reference: Language Essentials, *Weiskamp*

Quick C® DOS Utilities, *Alonso*

C Programming Reference: An Applied Prospective, *Miller and Quilici*

C Wizard's Programming Reference, *Schwaderer*

Introducing C to Pascal Programmers, *Shammas*

DOS Productivity Tips and Tricks, *Held*

此为试读，需要完整PDF请访问: www.ertongbook.com

*To my former colleague Riadh Al-Sabti,
wherever he may be, who taught me
that learning FORTRAN, and any other
programming language is fun.*

TRADEMARKS

Ada is a registered trademark of the U.S. Government, Ada Joint Program Office.

DEC PDP-11 is a trademark of Digital Equipment Corporation.

MS-DOS is a trademark of Microsoft Corporation.

PC-DOS is a trademark of International Business Machines.

Turbo C is a registered trademark of Borland International.

Turbo Pascal if a registered trademark of Borland International.

UCSD-Pascal is a trademark of the regions of the University of California, San Diego.

UNIX is a trademark of AT&T.

WordStar is a registered trademark of MicroPro International Corporation.

LISTINGS

Demonstration	Pascal Listing		C Listing	
	number	page	number	page
Greetings	3.1	18	3.2	19
Greetings			3.3	20
Simple data type	3.4	21	3.5	22
Constants	3.6	24	3.7	25
Constants			3.8	26
Formatted output			3.9	30
Character I/O	3.10	32		
Character I/O	3.11	33	3.12	33
Character I/O	3.13	34	3.14	34
Arithmetic operators	4.1	37	4.2	38
Arithmetic operators	4.3	41	4.4	42
Arithmetic operators			4.5	43
Character operators	4.6	44	4.7	44
sizeof operator	4.8	45	4.9	46
Type casting	4.10	48	4.11	49
Relational operators	4.12	52	4.13	54
Bit-manipulation operators	4.14	56	4.15	56
Bit-manipulation operators			4.16	57
Using #define			5.1	63
Using #define			5.2	69
Conditional compilation			5.3	72

Demonstration	Pascal Listing		C Listing	
	number	page	number	page
if statement	6.1	77	6.2	77
if-else statement	6.3	79	6.4	80
nested if-else statements	6.5	81	6.6	82
Nested if-else statements	6.7	83	6.8	84
switch statement	6.9	87	6.10	88
switch statement			6.11	89
switch statement	6.12	90	6.13	91
switch statement	6.14	92		
for loop	7.1	96	7.2	97
for loop	7.3	98	7.4	98
Nested for loops	7.5	99	7.6	101
Open loop	7.7	103	7.8	103
Exiting for loops			7.9	105
do-while loop	7.10	106	7.11	106
do-while loop	7.12	108	7.13	109
while loop	7.14	111	7.15	112
while loop	7.16	114		
while loop	7.17	115	7.18	116
Word count	7.19	117	7.20	118
Square function	8.1	122	8.2	122
Solve root pf math function	8.3	124	8.4	125
Functions replacing macros			8.5	127
void functions			8.6	132
Recursive function	8.7	136	8.8	136
Static variables	9.1	140	9.2	141
Scope of variables	9.3	145	9.4	145
Scope of variables			9.5	146
Pointers to simple data types	9.6	151	9.7	152
Simple array	9.8	154	9.9	155
Simple array	9.10	156	9.11	157
Two-dimensional array	9.12	159	9.13	159
Initializing a C matrix			9.14	161
Accessing an array with a pointer			9.15	163
Accessing an array with a pointer			9.16	164
Sieve benchmark	9.17	165	9.18	166
Sieve benchmark			9.19	166
Accessing a matrix with a pointer			9.20	168
Translate characters of a string	9.21	170	9.22	170

Demonstration	Pascal Listing		C Listing	
	number	page	number	page
Translate characters of a string	9.23	171		
String manipulation	9.24	172	9.25	173
Enumerated types	10.1	189	10.2	190
Enumerated types			10.3	192
Sorting with structures	10.4	196	10.5	197
Sorting with nested structures	10.6	201	10.7	202
Bitfields			10.8	204
Sorting with pointers to structures	10.9	206	10.10	208
Sorting with pointers to structures			10.11	209
Complex math with unions	10.12	213	10.13	214
Complex math with unions			10.14	216
Far pointers and direct video output			10.15	219
Far pointers and direct video output			10.16	219
Passing arrays to functions	11.1	224	11.2	225
Passing arrays to functions	11.3	226		
Passing strings to functions	11.4	227	11.5	228
Passing structures to functions	11.6	230	11.7	231
Passing arrays by reference	11.8	232	11.9	233
Passing strings to functions	11.10	236	11.11	237
Passing structures	11.12	238	11.13	239
Matrix inversion benchmark	11.14	241	11.15	242
Binary tree benchmark	11.16	245	11.17	246
Binary tree benchmark	11.18	248	11.19	249
Quick sort	11.20	251	11.21	252
Interactive Shell sort	11.22	254	11.23	256
Accessing command line arguments	11.24	260	11.25	261
Pointer to a function			11.26	263
Pointer to a function			11.27	268
Pointer to a function			11.28	272
Variable number of arguments			11.29	276
WordStar to ASCII converter (character I/O)	12.1	283	12.2	285
File printer (string I/O)	12.3	289	12.4	290
File Lister (string I/O)	12.5	292	12.6	293
Write strings to sequential file	12.7	296	12.8	298
Read strings from sequential file	12.9	302	12.10	303

Demonstration	Pascal Listing		C Listing	
	number	page	number	page
Write structure to sequential file	12.11	308	12.12	310
Read structure from sequential file	12.13	314	12.14	315
Write union to sequential file	12.15	318	12.16	320
Read union from sequential file	12.17	324	12.18	326
Stream I/O error			12.19	328
Copy file	12.20	333	12.21	334

此为试读，需要完整PDF请访问：www.ertongbook.com

INTRODUCTION

This introductory book is written for the Pascal programmer who wants to learn C using microcomputer implementations. While the material caters to these two languages in general, specifics and examples are presented based on two popular Pascal and C implementations: Turbo Pascal (version 4) and Turbo C (version 1.5 and up).

The reader is assumed to be at least moderately familiar with programming in Pascal. The basic presentation strategy employs listings in Pascal and their equivalent versions in C. Learning by comparing similar listings of the two languages enables the reader to draw on his or her experience as a Pascal programmer. This permits the reader to learn about the similarities and differences between the two languages and gradually develop a working knowledge of C. To accomplish this goal, simple (but not too trivial), short, and easy-to-read Pascal programs are generally used. The Pascal source code allows the reader to understand in more depth the task of the equivalent C listing. Throughout the chapters there are special notes for programming in C, as well as Pascal-to-C translation hints.

CONTENTS

Chapter 1: Why Learn C? 1

The Origin of C 1

Chapter 2: A Quick Tour of C 4

General C Program Components 4
Basic Data Types and Variables 5
Operators 6
Constants 7
Basic Console I/O 7
Decision-Making 8
Loops 10
Arrays 11
Strings 12
User-Defined Data Types 13
Functions 14
File I/O 15

Chapter 3: Getting Started 18

A Simple C Program 18
Simple Data Types in C 20
Constants In C 24
Basic Console I/O in C 26
Chapter Summary 35

Chapter 4: C Operators 36

Using Various Operators to Create Expressions 36
Arithmetic Operators 36
Increment and Assignment Operators 39

Character Operators 43
Sizing Data Objects 45
Type Casting 47
Relational Operators and Conditional Expressions 51
Bit-Manipulation Operators 55
Comma Operator 58
Chapter Summary 60

Chapter 5: The C Preprocessor andCompiler Directives 62

The C Preprocessor 62
Predefined Macros 71
Compiler Directives 71
Chapter Summary 73

Chapter 6: Decision-Making 75

The if Statement 75
The switch Statement 85
Chapter Summary 93

Chapter 7: Loops 95

Loops: An Overview 95
The for Loop 96
Exiting Loops 104
The do-while Loop 105
The While Loop 111
Chapter Summary 119

Chapter 8: Simple Functions 120

Overview 120
C Functions 120
Making a C Function Work as a Procedure 131
Recursive Functions 135
Exiting Functions 137
Chapter Summary 137

Chapter 9: Pointers, Arrays, and Strings 139

Storage Classes 139
Scope of Variables 144
Pointers to Simple Data Types 148
Arrays in C: An Overview 153
One-Dimensional Arrays 153
MultiDimensional Arrays 157
Accessing Arrays with Pointers 161
Strings in C 169
Chapter Summary 182

Chapter 10: Enumerated and Structured Data Types 185

Type Redefinition in C 185
Enumerated Data Types 187
Structured Data Types 193
Accessing Arrays of Structures 196
Bitfields in C 203

Using Pointers to Structures 205
Unions and Pointers to Unions 211
Far Pointers 218
Chapter Summary 220

Chapter 11: Advanced Functions 223

Using Arrays as Arguments 223
Using Strings as Arguments 227
Using Structures as Arguments 229
Passing Arguments by Reference Using Pointers 231
Passing Simple Variables and Simple Arrays 232
Passing Strings 236
Passing Structures 237
Passing Numeric Matrices 240
Passing Pointers to Dynamic Structures 244
More Array Sorting 251
Accessing the Command Line Arguments 259
Pointers to Functions 262
Functions with a Variable Number of Arguments 275
Chapter Summary 277

Chapter 12: Basic File I/O 279

Modes of File I/O in C 279
Character I/O 282
String I/O 286
Writing and Reading Numeric Data Using Sequential Files 295
Binary Stream I/O 306
Using Structures to Write and Read Numeric Data in Binary Streams 306
Using Unions with Random Access Binary Streams 316
Stream I/O Error 328
Basic Low-Level File I/O 331
Chapter Summary 335

Appendix A: C Escape Sequences 339

Appendix B: Formatted I/O String Control 340

Appendix C: Predefined Data Types in Turbo C 341

Appendix D: Operators in C 342

Appendix E: Memory Models for Turbo C 344

CHAPTER

1

Why Learn C?

THE ORIGIN OF C

C is a language that has come of age. Its roots go back to the BCPL language, developed by Martin Richards, and the B language, developed by Kenneth Thompson in 1970. C itself was developed for and implemented under UNIX™, by Dennis Ritchie, at Bell Laboratories, and first ran on a DEC PDP-11™ in the early 1970s. C was the first high-level assembler (that is, a cross between an assembler and a high-level language) that was successfully used to port UNIX over to different machines.

The ANSI Standard for C

In 1978, Prentice-Hall published *The C Programming Language* by, Brian Kernighan and Dennis Ritchie. This book described the C version accompanying the UNIX version 5. Dubbed the K&R definition, the book provided a de facto language reference, despite the fact that no ANSI standard existed for C in the seventies. In 1983, an ANSI standard committee was formed to look into the issue of defining a standard for C. In 1987, the committee completed its work, introducing a number of modifications over the K&R definition. This book looks at the ANSI standard and not the K&R definition.

The Dual Nature of C

Using C to write operating systems (like UNIX and MS-DOS®, to name a few) draws from its powerful features as a high-level assembler. Essentially, C is a small-core language with no predefined I/O routines whose compilers are notorious for producing fast and tight code. As a structured high-level assembler, C enjoys two natures, depending on the type of application for which it is used.

C as a High-Level Structured Language

You can use C as a high-level language and take advantage of its support for extended numeric precision, user-definable record structures, powerful operators, loops, and decision-making constructs. Consequently, high-level applications can be developed in various fields, such as statistics, engineering design, accounting, and database management. As a high-level language, C is compared with other similar, well-known languages, such as Pascal, Modula-2, and Ada.

C: The High-Level Assembler

On the other hand, you can employ the power of C to perform advanced data manipulation and low-level access and to implement some unusual programming tricks. C gives you the freedom to perform these tasks, assuming that you know what you are doing. Compared to Pascal, C removes programming guard rails and puts more responsibility on the programmer's shoulders. Thus, C can be used to develop many low-level applications, such as operating systems, compilers, interpreters, and word processors.

The Journey From Pascal To C

Why migrate from Pascal to C? Why change from one structured language to another? Is it worth it? These are some of the questions that Pascal programmers might ask in contemplating learning C.

As a Pascal programmer, you developed the skill of crafting your programs in a structured, modular manner. The more a software developer programs in a language, the more programming tricks he or she discovers. However, there is usually an asymptotic limit that is reached, beyond which any language cannot be pushed. You can rewrite Pascal programs to run faster or compile into smaller code, until further refinement is either not possible or not feasible. The above reason points to one of the major reasons why high-level programmers migrate to C: to develop programs that have more speed and/or smaller code.

The good news for you as a Pascal programmer is that you already have the experience of using a structured language. This makes learning C easier

此为试读，需要完整PDF请访问： www.ertongbook.com