



THE OFFICIAL BORLAND TURBO C SURVIVAL GUIDE

01950364

TCJC BOOKSTORE
USED BOOK

\$25.10

LAWRENCE H. MILLER

ALEXANDER E. QUILICI

THE OFFICIAL BORLAND TURBO C SURVIVAL GUIDE

Lawrence H. Miller

University of Southern California, Information Sciences Institute

Alexander E. Quilici

University of California, Los Angeles



WILEY

John Wiley & Sons

New York □ Chichester □ Brisbane □ Toronto □ Singapore

Another one for our families

Cover Illustration: Neil Shigley

Copyright ©1989 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 Permissions Department, John Wiley & Sons, Inc.

The programs presented in this book have been included solely for their instructional value. They have been carefully tested but are not guaranteed for any particular purpose. The publisher does not offer any warranties or representations, nor does it accept any liabilities with respect to the programs.

Library of Congress Cataloging in Publication Data

Miller, Lawrence H.

The Official Borland Turbo C survival guide / Lawrence H. Miller.
Alexander E. Quilici.

p. cm. — (Borland-Wiley Higher Education Series)

Bibliography: p.

Includes index.

ISBN 0-471-60861-0 (pbk.)

1. C (Computer program language) 2. Turbo C (Computer program)

I. Quilici, Alexander E. II. Title. III Series.

QA76.73.C15M552 1988

005.13'3—dc19

6pc 88-27608

CIP

The Official Borland Turbo C Survival Guide

Borland-Wiley Higher Education Series

FOREWORD

Within a very short span of time, C has become the programming language of choice for a large number of professional programmers. And among these users, Turbo C has demonstrated the speed, power, flexibility, and accessibility that make it a superior programming environment.

It is clear, then, that there is an urgent need for a textbook that presents Turbo C to students—introducing them to the superior technology of this environment in a logically organized, highly readable text. Now Larry Miller and Alex Quilici have met that need with *The Official Borland Turbo C Survival Guide*.

Here students will find a thorough and accurate discussion of Turbo C, beginning with a tutorial introduction to the language and its environment, and continuing on through advanced topics such as memory models, portability and efficiency, and graphics programming. Presentation of the material is enhanced by hundreds of example programs that illustrate Turbo C's features, and by a set of case studies implementing common real-world applications. Pictorial descriptions of data structures and algorithms also clarify and simplify features and functions of Turbo C.

Students at various levels of experience—whether self-taught or learning in the classroom—will find that *The Official Borland Turbo C Survival Guide* can open to them the substantial power of programming in Turbo C.



Phillipe Kahn
President and Founder
Borland International, Inc.

PREFACE

C is a powerful programming language, one that is especially popular among professionals who program small personal computers. And Turbo C—with its sophisticated programming environment, its speedy compiler, and its special language features—makes programming in C a pleasure. But unfortunately C also has a reputation for being hard to learn and even harder to master. Its conciseness and complexity can overwhelm all but the most experienced programmer. *The Official Borland Turbo C Survival Guide* will keep you from slowly sinking into a sea of confusion, and speed your journey from novice to expert C programmer. You'll find this book worthwhile if:

- *You've programmed in other high-level languages such as BASIC, FORTRAN, or Pascal, and now want to program in Turbo C.* We don't start from scratch and try to teach you how to program. Instead, we assume some prior programming experience and expend most of our effort describing how C differs from other high-level languages and investigating its unique and unfamiliar features.
- *You've programmed in C but you don't yet feel as though you've mastered it.* We cover the language completely, delving deeply into many issues that are often casually examined or completely avoided. You can't be a proficient C programmer without knowing how to put together large programs, how to produce portable and efficient code, or how to use pointers to effectively organize and access data—topics we emphasize rather than ignore.
- *You're familiar with C but not with Turbo C or ANSI C.* Turbo C implements the new ANSI C standard, along with some of its own additional features. We examine and exploit its ANSI C extensions such as function prototypes and implicit string concatenation. We explore the special Turbo C extensions, including its memory models, pointer modifiers, and graphics library. And we explain how to use the Turbo C programming environment, including the program builder and the debugger.
- *You're looking for examples of real-world C programs or are interested in obtaining a set of tools to improve your DOS environment.* We provide over 175 useful programs and functions, including a file display program, an indexed data base manager, a cross referencer, a function plotter, a file compression program, an octal dump program, a program to eliminate duplicate lines, various sorting and searching programs, and more. We also provide implementations of useful

libraries, such as sets and queues, and utility functions to manage console input and output. All of these programs have been written, compiled, and executed using Turbo C version 2.0.

Special Features

This book has grown out of several years of teaching C courses to a wide range of students—from first year computer science majors to lifelong assembly language programmers, from casual computer users familiar only with BASIC to competent C hackers fine-tuning their skills. And we've concluded that there are several ways to ease the often difficult process of mastering a programming language, all of which we've incorporated here:

- *We provide complete, useful programs.* Our examples don't merely illustrate language features but illustrate them in a realistic way, as part of a useful function or program. And our examples are complete—we avoid potentially confusing program fragments and ensure that every function comes with a main program that shows how to call it and how to use its return value.
- *We provide sizeable programs.* Almost every chapter ends with a case study, a large, real-world application that cements the concepts covered in the chapter. We often construct these case studies from functions and programs written and explained earlier in the text, reinforcing our earlier examples and providing realistic examples of how bigger programs are built from smaller pieces.
- *We provide pictorial descriptions of data structures and algorithms.* Numerous illustrations help clarify complex concepts such as pointers, arrays, and dynamic allocation. These pictures simplify seemingly complicated data structures and algorithms, and ensure that our explanations are easy to follow.
- *We provide plenty of programming exercises.* We've included over 400 exercises spanning a wide range of difficulty. Some are simply modifications to our example programs that make them more efficient, more concise, more useable, or more user-friendly. Others range from small programs to sizeable programming projects, many of which are useful programs in their own right.

Organization

We divide the text into five parts. Part I is a tutorial introduction to Turbo C. Part II is detailed discussion of its basic data types (including pointers and arrays), operators, and statements. Part III covers constructing more complex programs, looking at functions, storage classes, and separate compilation. Part IV looks at data structures, from arrays of pointers to linked lists and trees. Finally, Part V covers special Turbo C features such as its graphics library and its memory models, studies real-world concerns such as portability, efficiency, and debugging.

Part I—A Tutorial Introduction to Turbo C • *Chapter 1* presents our first C program—one that computes the interest accumulating in a bank account—and uses it

to introduce the basic C data types, operators, and statements, and to explain how to use the Turbo C development environment to compile and run programs. • *Chapter 2* provides three more example programs—an extension to the interest rate accumulator, a program to reverse its input, and an implementation of insertion sort—and uses them to introduce several additional control structures and operators, and the basics behind arrays and functions. This chapter also examines input/output (I/O) redirection and explains how it turns these programs into useful tools. Together, these two chapters provide an excellent feel for what programming in Turbo C is like.

Part II—Data Types, Operators, and Statements • *Chapter 3* describes data types, placing extra emphasis on how to manipulate characters, including discussions of character input and output and character testing. It concludes with our first case study, a program to display its standard input a page at a time. • *Chapter 4* addresses operators, paying special attention to the shorthand assignment and bitwise operators, and to automatic type conversion and casting. Its case study is a program to compress and uncompress files. • *Chapter 5* studies statements, showing the most appropriate uses for each, and concluding with a case study that dumps its input in octal. • *Chapter 6* presents pointers, arrays, and strings, emphasizing the relationships between them. It also introduces the standard string functions and uses them to write a useful tool to detect duplicate input lines.

Part III—Constructing Complex Programs • *Chapter 7* focuses on functions, exploring the details of how they can communicate with each other through parameter passing and return values. It also covers the closely related topics of pointers to functions, generic functions, and functions that can take variable numbers of arguments. Its case study is a recursive implementation of binary search. • *Chapter 8* studies storage classes and separate compilation, and discusses Turbo C's program builder. Its case study shows how to create abstract data types, illustrating them with a package that implements sets. • *Chapter 9* presents the preprocessor, describing how we can use it to make our programs easier to read and debug, as well as more efficient. The case study uses macros to create a simpler interface to functions with complicated arguments.

Part IV—Organizing and Accessing Data • *Chapter 10* discusses multidimensional arrays, showing how we can use pointers to access them efficiently. It dramatically illustrates their use in an implementation of the Game of Life. • *Chapter 11* presents arrays of pointers, showing how we can use dynamic allocation to initialize them, and how we can use them to access command-line arguments. Its case study is a string sorting program that takes advantage of both. • *Chapter 12* studies structures, unions, and enumerated types, including arrays of structures, using all of them in a simple data base program. • *Chapter 13* continues our study of structures by showing how to write insertion sort using linked lists and trees, and combining both in a C program cross-referencer. • *Chapter 14* explains external files, providing complete coverage of the standard I/O library, bringing together the ideas of the chapter in an indexed data base for names, addresses, and phone numbers.

Part V—Real-World Programming Issues • *Chapter 15* shows how Turbo C's memory models and pointer modifiers support writing programs that require large amounts of memory. Its case study describes how to write programs that work well under multiple memory models. • *Chapter 16* presents common portability problems

and suggests some solutions, concluding with a set of functions for portably managing console displays. • *Chapter 17* examines efficiency, including techniques for making our programs run faster or take up less space. Its case study provides fast functions for allocating fixed-size blocks of memory and for reading in an array of integers. • *Chapter 18* is an in-depth discussion of Turbo C graphics programming that ends with a program to plot functions. • *Chapter 19* describes the debugger and uses it to debug several short programs. The book concludes with a complete set of appendices that discusses configuring Turbo C, presents the different command-line options, menu entries, and header files, and summarizes the Turbo C library functions.

Acknowledgements

Contributions by several of our friends and colleagues have greatly improved the quality of this text. We're deeply in debt to Robert Quilici for painstakingly plowing through our prepublication drafts, unearthing plenty of problems with our programs and explanations. And we're grateful to Jake Richter for his careful proofreading, to David Smallberg for his always perceptive comments and criticisms, and to Dorab Patel for his invaluable wizardry with formatting.

We would like to thank the UCLA Computer Science Department and USC's Information Science Institute for the generous use of their resources and their allowing us the time to produce this work. We would also like to thank the people at John Wiley, especially our editors, Gene Davenport and Terry Zak, who as usual have been extremely helpful and made life pleasant for their authors.

And finally, we're grateful for all of the encouragement and support given to us by our families and friends—especially Rita Grant-Miller, Tammy Merriweather, Irene Borromeo, and Doris Perl.

Lawrence H. Miller
Alexander E. Quilici

Los Angeles, California
August 1988

CONTENTS

PREFACE	vii
1 INTRODUCING TURBO C	1
1.1 OUR FIRST TURBO C PROGRAM	1
Comments, File Inclusion, and Constants	1
The main Program	3
Variable Declarations	4
Assignment Statements	4
The printf Statement	4
The while Loop	6
The return Statement	7
Program Layout	7
1.2 COMPILING AND RUNNING TURBO C PROGRAMS	7
1.3 USING THE TURBO C COMMAND-LINE ENVIRONMENT	8
Compiling with tcc	8
Dealing with Compile Errors	8
1.4 USING THE INTERACTIVE TURBO C ENVIRONMENT	10
Entering a Program in tc	11
Obtaining Help	12
Saving and Loading Files	13
Compiling and Executing Programs	14
Using Menus	16
2 SOME EXAMPLE PROGRAMS	19
2.1 EXTENDING OUR FIRST C PROGRAM	19
Using scanf	19
Handling Input Errors	21
The if Statement	21
The for Loop	23
2.2 A PROGRAM TO REVERSE ITS INPUT	24
Arrays	24
Functions	26
Input/Output Redirection	30
2.3 CASE STUDY—INSERTION SORT	33
Handling Input Errors	33
Implementing Insertion Sort	36
	xi

3	BASIC DATA TYPES	39
3.1	IDENTIFIERS AND NAMING CONVENTIONS	39
3.2	DATA TYPES	40
	Integers	40
	Reals	41
	Characters	43
3.3	CHARACTERS AND INTEGERS	45
3.4	CHARACTER INPUT AND OUTPUT	47
	getchar and putchar	48
	Unbuffered Character Input	49
3.5	CHARACTER TESTING FUNCTIONS	52
3.6	CONVERTING CHARACTERS INTO NUMBERS	56
3.7	CASE STUDY—A FILE DISPLAY PROGRAM	60
4	OPERATORS AND CONVERSIONS	63
4.1	OPERATORS, OPERANDS, AND PRECEDENCE	63
4.2	ARITHMETIC OPERATORS	65
	Integer Arithmetic	65
	Floating Point Arithmetic	66
	Unsigned Arithmetic	66
4.3	RELATIONAL AND LOGICAL OPERATORS	67
4.4	BITWISE OPERATORS	69
	Bit Shifts	69
	Bitwise Logical Operators	70
	Getting and Setting Bits	71
	Using Exclusive OR	74
4.5	ASSIGNMENT OPERATORS	77
	Shorthand Assignment Operators	77
	Postfix and Prefix Assignment	78
4.6	OTHER OPERATORS	79
	The Comma Operator	79
	The sizeof Operator	80
	The Conditional Operator	81
4.7	TYPE CONVERSIONS	82
	Automatic Type Conversions	82
	Casts	84
4.8	CASE STUDY—DATA COMPRESSION	85
5	STATEMENTS	89
5.1	EXPRESSION AND COMPOUND STATEMENTS	89
5.2	SIMPLE DECISIONS—if	90
5.3	MULTIWAY DECISIONS—else-if	91
5.4	MULTIWAY DECISIONS—switch	94
5.5	LOOPS	96
	The do-while Statement	98

	The for Statement	98
	Concise Control Expressions	100
5.6	THE NULL STATEMENT	102
5.7	ALTERING CONTROL FLOW	104
	The break Statement	104
	The continue Statement	105
	The goto Statement	105
5.8	CASE STUDY—OCTAL DUMP	107
6	ARRAYS POINTERS AND STRINGS	111
6.1	POINTERS	111
	Declaring and Obtaining Pointers	112
	Dereferencing Pointer Variables	112
6.2	TRAVERSING ARRAYS USING POINTERS	114
	Pointer Arithmetic	115
	Pointer Comparisons	118
6.3	ARRAY PARAMETERS AND POINTERS	121
6.4	CHARACTER ARRAYS—STRINGS	123
	Building Strings from the Input	124
	String Functions from the Standard Library	125
	Additional String Functions	127
	Using Pointers to Traverse Strings	128
6.5	DYNAMICALLY ALLOCATING ARRAYS	132
	Allocating Storage with malloc	134
	Deallocating Storage with free	136
6.6	CASE STUDY—ELIMINATING DUPLICATE LINES	136
7	FUNCTIONS	139
7.1	FUNCTION PROTOTYPES	139
	Automatic Parameter Type Conversions	139
	Specifying the Types of Return Values	139
7.2	OLD-STYLE DECLARATIONS AND DEFINITIONS	143
	Type Checking Problems	144
	Parameters and Portability	145
	Problems with Mixing Styles	147
	Prototypes versus Old-Style Declarations	147
7.3	THE return STATEMENT	148
7.4	SIMULATING CALL-BY-REFERENCE	150
7.5	PASSING VARIABLE NUMBERS OF ARGUMENTS	154
	Built-in Variable Argument Functions	157
7.6	POINTERS TO FUNCTIONS	158
	Built-in Generic Functions	161
	Writing a Generic Function	162
7.7	RECURSION	163
7.8	CASE STUDY—BINARY SEARCH	167

8	CONSTRUCTING LARGER PROGRAMS	173
8.1	LOCAL VARIABLES	173
	The Storage Class <code>auto</code>	173
	The Storage Class <code>register</code>	177
	The Storage Class <code>static</code>	179
8.2	GLOBAL VARIABLES	182
8.3	EXTERNAL DECLARATIONS	184
8.4	SEPARATE COMPILATION	187
	Separate Compilation with <code>tcc</code>	190
	Separate Compilation with <code>tc</code>	191
	A Bigger Project—Queues	192
	More Advanced Project Files	196
8.5	PRIVATE GLOBAL VARIABLES AND FUNCTIONS	199
8.6	TYPE MODIFIERS	200
8.7	CASE STUDY—ABSTRACT DATA TYPES	201
	Implementing Sets	201
	Defining the set Type	201
	The set Operations	202
	Using Sets	203
	Some Notes On Abstract Data Types	203
9	THE PREPROCESSOR	209
9.1	PREPROCESSOR DIRECTIVES	209
9.2	SIMPLE MACRO SUBSTITUTION	210
	Syntactic Replacement	212
9.3	MACRO SUBSTITUTION WITH PARAMETERS	214
	Some Useful Macros	215
	Using Macros in Macro Definitions	217
	Potential Problems	219
	Macro Operators	220
	Undefining Names	221
9.4	FILE INCLUSION	223
	Including Data Type Definitions	225
9.5	CONDITIONAL COMPILATION	226
	Testing Name Definition	227
	Defining Names	229
	Predefined Names	230
	More General Compile-Time Tests	232
9.6	OTHER PREPROCESSOR DIRECTIVES	235
9.7	CASE STUDY—IMPROVING FUNCTION INTERFACES	236
10	MULTIDIMENSIONAL ARRAYS	239
10.1	TWO-DIMENSIONAL ARRAYS	239
	Internal Representation of Two-Dimensional Arrays	241
	Initializing Two-Dimensional Arrays	243
10.2	POINTERS AND TWO-DIMENSIONAL ARRAYS	245

Traversing Columns	245
Traversing Rows	249
Accessing Rows	250
10.3 THREE-DIMENSIONAL ARRAYS	252
<i>N</i> -Dimensional Arrays and Parameters	253
<i>N</i> -Dimensional Arrays and Pointers	254
Initializing Two-Dimensional Arrays	257
10.4 CASE STUDY—THE GAME OF LIFE	257
An Array-Subscripting Version	258
A New Version Using Pointers	260
11 ARRAYS OF POINTERS	267
11.1 ARRAYS OF POINTERS—RAGGED ARRAYS	267
Accessing Individual Items	269
Dynamic String Allocation	271
11.2 POINTERS AND ARRAYS OF POINTERS	276
11.3 COMMAND-LINE ARGUMENTS	281
Command-Line Options	282
11.4 ARRAYS OF POINTERS TO FUNCTIONS	286
11.5 TYPE DECLARATIONS	289
Declarators	289
Type Specifications	291
11.6 CASE STUDY—SORTING STRINGS	292
12 CONSTRUCTED TYPES	297
12.1 STRUCTURES	297
Defining Structures	297
Field Selection	298
Compile-Time Initialization	299
Structures as Function Arguments	299
Structures and Operators	300
Arrays of Structures	302
Bitfields	307
12.2 UNIONS	308
12.3 ENUMERATED TYPES	311
12.4 CASE STUDY—A DATA BASE APPLICATION	316
Storing the Data Base	317
The Data Base Program	317
13 LINKED LISTS AND TREES	325
13.1 LINKED LISTS	325
Building a Sorted Linked List	326
Traversing a Linked List	330
A Note on Insertion Sort's Efficiency	330
13.2 SORTING STRINGS USING LINKED LISTS	331
13.3 GENERALIZED LINKED LISTS	332

13.4	BINARY TREES	337
	A Note on the Efficiency of Tree Sort	342
13.5	CASE STUDY—A CROSS-REFERENCE PROGRAM	343
	Cross-Referencer Data Structure	343
	The Program	345
14	EXTERNAL FILES	355
14.1	ACCESSING EXTERNAL FILES	355
	Character File I/O	356
	Formatted I/O to Files	357
	Line-Oriented I/O	361
14.2	THE STANDARD FILES	363
14.3	RANDOM FILE ACCESS	366
14.4	BLOCK INPUT AND OUTPUT	368
14.5	FILE UPDATING	372
14.6	CASE STUDY—AN INDEXED DATA BASE	375
15	MEMORY MODELS AND POINTER MODIFIERS	383
15.1	POINTERS REVISITED	383
	The 8086 Memory Architecture	383
	Memory Models	385
	Displaying Pointers	387
15.2	POINTER MODIFIERS	390
	near Data Pointers	391
	far Pointers	397
	huge Pointers	402
15.3	FUNCTION MODIFIERS	405
	near Functions	406
	far Functions	406
	huge Functions	409
15.4	CASE STUDY—REIMPLEMENTING QUEUES	412
16	PORTABILITY	415
16.1	PRINCIPLES OF PORTABILITY	415
16.2	PORTABILITY ACROSS COMPILERS	416
	Using Special Features	416
	Using Function Prototypes	417
	Identifier Names	418
	Libraries	420
16.3	PORTABILITY ACROSS OPERATING SYSTEMS	420
16.4	MACHINE DEPENDENCIES	423
	Data Type Sizes	423
	Pointer Problems	424
	Character Set Differences	425
	Sign Problems	426
	Byte Ordering	426
16.5	CASE STUDY—PORTABLE I/O DEVICE HANDLING	428

17 EFFICIENCY	435
17.1 SOME BASICS	435
17.2 COMMON SOURCES OF INEFFICIENCY	436
Unnecessary Conversions	436
Unnecessary Arithmetic	437
17.3 C EFFICIENCY AIDS	439
Compile-Time Initialization	439
Using Pointers	440
Using Macros	442
17.4 SHRINKING SPACE REQUIREMENTS	444
Eliminating Unused Storage	445
Eliminating Extra Information	446
17.5 CASE STUDY—REPLACING LIBRARY ROUTINES	446
Managing Our Own Free Storage Pool	446
Writing Our Own Input Handler	449
18 TURBO C'S GRAPHICS LIBRARY	453
18.1 INTRODUCTION	453
Using the Graphics Library	455
Error Handling	456
Obtaining Screen Information	457
18.2 DRAWING OBJECTS	459
Lines and Polygons	459
Arcs and Circles	462
Filled Objects	463
18.3 COLORS	465
cga Color Control—Low Resolution	468
cga Color Control—High Resolution	469
ega / vga Color Control	469
18.4 STYLES	472
Setting Line Drawing Styles	473
Getting and Setting Fill Patterns	474
18.5 SCREEN AND VIEWPORT MANIPULATION	476
18.6 TEXT OUTPUT IN GRAPHICS MODE	481
18.7 TEXT OUTPUT IN TEXT MODE	483
Manipulating Text	484
Character Attributes	485
18.8 CASE STUDY—A FUNCTION PLOTTER	486
Overview of the Function Plotter	486
The Plotter Itself	488
19 DEBUGGING TURBO C PROGRAMS	493
19.1 INTRODUCTION	493
19.2 DEBUGGER BASICS	494
Starting the Debugger	495
Executing Statements	495