


INTRODUCTION TO
COMPUTER
PROGRAMMING
USING TURBO PASCAL

RICHARD E. JOHNSON

DAVID M. KEIL



Introduction to Computer Programming using Turbo Pascal

Richard E. Johnson

David M. Keil

WEST PUBLISHING COMPANY

Minneapolis/St. Paul New York Los Angeles San Francisco

▼▼▼ Production Credits ▼▼▼

Composition: Carlisle Communications

Copyediting: Loretta Palagi

Cover Image: Kirsten L. Johnson and Kristopher Hill

Text Design: Rosyln M. Stendahl, Dapper Design

Photo Credits: 5, 137, 193, 446 ©1994 Historical Pictures/Stock Montage, Inc.
160 (left and right) Courtesy of International Business Machines Corporation

WEST'S COMMITMENT TO THE ENVIRONMENT

In 1906, West Publishing Company began recycling materials left over from the production of books. This began a tradition of efficient and responsible use of resources. Today, up to 95 percent of our legal books and 70 percent of our college and school texts are printed on recycled, acid-free stock. West also recycles nearly 22 million pounds of scrap paper annually—the equivalent of 181,717 trees. Since the 1960s, West has devised ways to capture and recycle waste inks, solvents, oils, and vapors created in the printing process. We also recycle plastics of all kinds, wood, glass, corrugated cardboard, and batteries, and have eliminated the use of Styrofoam book packaging. We at West are proud of the longevity and the scope of our commitment to the environment.

Production, Prepress, Printing and Binding by West Publishing Company.



TEXT IS PRINTED ON 10% POST CONSUMER RECYCLED PAPER



British Library Cataloguing-in-Publication Data. A catalogue record for this book is available from the British Library.

Copyright © 1995 - By WEST PUBLISHING COMPANY
610 Opperman Drive
P.O. Box 64526
St. Paul, MN 55164-0526

All rights reserved

Printed in the United States of America

02 01 00 99 98 97 96 95 8 7 6 5 4 3 2 1 0

Library of Congress Cataloging-in-Publication Data

Johnson, Richard, 1932-

Introduction to computer programming using Turbo Pascal / Richard Johnson, David Keil.

p. cm.

Includes index.

ISBN 0-314-04206-7 (pbk.)

1. Pascal (computer program language) 2. Turbo Pascal (Computer file) 3. Object-oriented programming (Computer science) I. Keil, David. II. Title.

QA76.73.P2J63 1994

005.13'3--dc20

94-13901



Introduction to Computer Programming using Turbo Pascal

▼▼▼ **Dedication** ▼▼▼

*Dedicated, with love, to Jessy, Kevin, Kirsten,
Leif, Juan David, and Leonilda, and to our parents*



Preface

This text and the accompanying program disk are based on ideas and materials originally developed for two distinct but overlapping one-semester courses, Introduction to Computer Science and Pascal Programming.

Different views exist about where the boundary lies between *computer science* and *computer programming*, and to what extent the two subjects overlap. We believe that to lay claim to the title “computer scientist” you must be reasonably competent at writing programs in some language. We also believe that to program well in any language you should have a rudimentary knowledge of the architecture and mathematics of a general-purpose computer.

▼ A Note on Chapter 1

Chapter 1 describes a simplified computer and a ten-instruction assembly language for it. We feel that this introduction to the basic operation of a digital computer will enrich the subsequent study of programming in a high-level language (in this case, Pascal). When a compiled version of one of your programs executes, you will have a better idea of what occurs at the machine level. Some of the mystery of the “magic box” will have been revealed. The program example diskette that accompanies the text includes software to write, assemble, and run programs written in the assembly language of Chapter 1. Appendix B describes the use of this software.

Chapter 1 has a second objective. The three fundamental control structures of all modern languages, the *sequence*, the *decision* (branch), and the *loop*, are introduced using machine-level programming examples unencumbered by the syntax requirements of Pascal. When the **IF...THEN...ELSE** statement and the Pascal loops (**FOR**, **WHILE**, **REPEAT**) are introduced in later chapters, the *concepts* of branching and looping will be familiar to you; only the syntax will be new.

▼ Why Turbo Pascal?

Our classroom experience is that the most significant progress in mastering a programming language occurs through designing, coding, and testing original programs. To code and test a Pascal program, you must have access to an editor and a compiler. Although simple programs written in “generic” or “standard” Pascal may compile successfully on most compilers, at some point in the process of writing more sophisticated programs it becomes advantageous, if not necessary, to use the nonstandard features of a particular compiler. For that reason, we have written this text with exclusive reference to the Turbo Pascal compiler.

▼ Pedagogy

Certain pedagogical beliefs have been kept in mind in writing the text. Among them are the following:

1. The best way to learn is by doing. Throughout the text, *research activities* and *chapter programming exercises* based on the program examples encourage you to use the program example disk that accompanies the text.
2. The level of understanding of a particular concept that satisfies another student may not satisfy your curiosity. To address these differences, we have included *optional sections* on certain topics. They may be omitted without loss of continuity. In an additional attempt to address individual differences, and to challenge gifted students, one or more problems labeled “Challenge Exercise” are included in several of the sets of chapter programming exercises.
3. Single-concept program examples, or at least those that emphasize a *single new idea*, are less confusing and hence more instructive than examples that illustrate several new ideas. Although the program examples tend to become larger as the text progresses, we have limited the number of new ideas and/or syntax elements in each new program example.

In a further attempt to anchor new ideas and concepts to a familiar base, we bring back one program example, Program PAYROLL, which is introduced in Chapter 4, in newer, more sophisticated versions, in Chapters 5, 6, 7, and 13. Each new version of Program PAYROLL adds flexibility and power by making use of newly acquired programming tools.

4. For *teaching* purposes, simple, ordinary, “little” words are preferable to fancy, unusual, “big” words. In fact, we find in our classes an ever-increasing percentage of students for whom English is a second language. Struggling with uncommon words and unwieldy English sentences can be just as tough a task as mastering programming concepts. This use of simple language does not result in a “watered down” text.
5. The text is written with first-semester computer science students in mind. There are places in the text where the temptation to offer an extremely terse and logically elegant definition or problem solution—the kind that would delight a good mathematician—is great, but where a less abstract definition or solution seems to be more appropriate for a beginning programming student.
6. We have tried to write with clarity and in adequate detail about programming logic using Pascal as a vehicle. We have *avoided* writing a complete Turbo Pascal reference manual. This text is about programming and problem-solving; it is not about a compiler.
7. Twice as many words do not necessarily make an idea or a definition twice as clear. There is something to be said for the economical and efficient use of English.

▼ A Note on Chapter 15

Object-oriented programming (OOP) is well established in industry. OOP concepts provide much of the content of programming journals, and OOP themes have, for some time, been dominant topics at software development seminars.

We have not always found it easy to learn this new way of programming, but the enthusiasm of our students encourages us. We have discovered that, while *objects* (as the word is used in the strict sense of its meaning in the vocabulary of OOP) aren't intuitive for die-in-the-wool procedure-oriented programmers, they make sense to beginning programmers. Objects tend to model the real world as most of us conceptualize it.

Chapter 15 is not a comprehensive treatment of OOP. It is, rather, an introduction to some of the underlying principles and structures of object-oriented programming. We hope that Chapter 15 will motivate you to learn more about an important new way of programming.

▼ Unit OURSTUFF

Units are fundamentally important in the programming environment of Turbo Pascal. In addition to the predefined units (*Crt*, *Printer*, *Dos*, and so on), user-defined units can be of real value to programmers (students and professionals) as a way to reduce coding time. For those of us who attempt to write object-oriented programs, user-defined units become essential to the implementation of **inheritance**.

Beginning in Chapter 7, we present a user-defined unit, *OURSTUFF*. Two useful I/O procedures are included in the initial version of *OURSTUFF*. Later, in Chapter 10, two routines used in several of the sorting and searching program examples are added to *OURSTUFF*. Finally, in Chapter 11, *OURSTUFF* is expanded to include several new I/O routines, a sort procedure, and a search function; these routines are used extensively to simplify coding in the program examples in Chapters 11 through 15.

We hope that the ongoing development and extensive use of a particular user-defined unit will persuade you to take advantage of this important programming tool.

▼ ACM/IEEE

The text contents have been examined with reference to the report of the ACM/IEEE—CS Joint Curriculum Task Force, *Computing Curricula 1991*. The report lists 55 *knowledge units* under 10 *subject areas*. Taken collectively, these constitute the *common requirements* for a four-year undergraduate curriculum in computer science. The task force makes no attempt to mandate a chronological ordering of the knowledge units over eight semesters, although some sample curricula are suggested. This text addresses, at some level, at least 7 of the 10 subject areas and at least 25 of the 55 required knowledge units.

▼ End-of-Chapter Activities

We make a distinction between the *review problems* and the *chapter exercises* that appear at the end of each chapter. The review problems do not require writing of original programs, and do not, for the most part, require a significant amount of coding. The review problems are typically questions about program examples that occur in the chapter, and are often of the “What if?” variety. They might be useful as at least partial preparation for a quiz or a test. The chapter exercises, with few

exceptions, are programming exercises that give you an opportunity to implement the structures and ideas of the chapter. The exercises are not sorted in “ascending” order from easy to difficult. Rather, they tend to follow the sequence of the chapter sections.

▼ Thanks

This text and the accompanying program example diskette reflect the valuable advice and constructive criticism of a great number of our teaching colleagues from community colleges, liberal arts colleges, and universities across the country. These colleagues include:

Phil Novinger	<i>The Florida State University</i>
Helen Casey	<i>Sam Houston State University</i>
Roger E. Eggen	<i>University of North Florida</i>
Paul A. Smith	<i>South Puget Sound Community College</i>
Michael Fry	<i>Lebanon Valley College</i>
Robert Moll	<i>The University of Massachusetts, Amherst</i>
Sheau-Dong Lang	<i>University of Central Florida</i>
Peter Casey	<i>Central Oregon Community College</i>
James Payne	<i>Kellogg Community College</i>
Y. H. Harris Kwong	<i>State University of New York, Fredonia</i>
Phillip R. Bender	<i>Marquette University</i>
Thomas J. Cheatham	<i>Middle Tennessee State University</i>
Ronald A. Mann	<i>University of Louisville</i>
Evelyn W. Speiser	<i>Glendale Community College</i>
Joseph J. Waters	<i>Santa Rosa Junior College</i>
E. Terry Magel	<i>Kentucky State University</i>
Andrew Bernat	<i>The University of Texas at El Paso</i>
Mike Michaelson	<i>Palomar College</i>
Paul Shapiro	<i>Newton Centre MA</i>
Robert Sterling	<i>Tidewater Community College</i>
Peggy S. Eaton	<i>University of New Hampshire and Plymouth State University</i>
Sharon Underwood	<i>Livingston University</i>
Walter Chesbro	<i>Santa Rosa Junior College</i>
Barbara A. Gentry	<i>Parkland College</i>
Timothy Margush	<i>The University of Akron</i>
Stephen F. Weiss	<i>The University of North Carolina at Chapel Hill</i>

Murat M. Tanik	<i>Southern Methodist University</i>
Curtis R. Bring	<i>Moorhead State University</i>
Jimmie M. Purser	<i>Millsaps College</i>
Ingrid Russell	<i>University of Hartford</i>
Marguerite R. Summers	<i>Sangamon State Universtiy</i>
Michael Pelle	<i>Berkshire Community College</i>
John Stocksen	<i>Kansas City Community College</i>
Stephen C. Solosky	<i>Nassau Community College</i>
Charles M. Williams	<i>Georgia State University</i>
Norman H. Liebling	<i>San Jacinto College</i>
Carl Maltz	<i>California State University, Long Beach</i>

All of the material has been tested and refined in our classrooms at Western New England College and Massachusetts Bay Community College. The software and expanded documentation for the model processor of Chapter 1 is also being used at the Florida Center for Instructional Technology, University of South Florida. We are in great debt to our students and teaching colleagues for the many improvements they have contributed to the text and software you are about to use.

Finally, we wish to thank the editorial staff at West Educational Publishing, including Sharon Adams, Peter Gordon, Lucy Paine Kezar, and Jay Ricci, for their assistance and encouragement, and Amy Gabriel, production editor, for her painstaking attention to detail in putting the whole thing together.

We wish you great success and happy computing!

Richard E. Johnson and David M. Keil
August 1994



Introduction to Computer Programming using Turbo Pascal



Contents

- ▼ **CHAPTER 1** A Model Computer, Machine Language Programming, and Algorithm Design 1
 - Introduction 2
 - 1.1 A Model Computer for Numeric Problem-Solving 3
 - 1.2 A Set of Operations 7
 - 1.3 The Fetch-Execute Cycle 10
 - 1.4 Data Statements 16
 - 1.5 Flowcharts, Decisions, and Branch Structures 17
 - 1.6 Loop Structures, Counters, and Sentinel Values 22
 - 1.7 The Binary Number System (Optional) 29
 - 1.8 Storing Integers in Memory (Optional) 35
 - Summary 39
 - Review Problems 40
 - Exercises 43

- ▼ **CHAPTER 2** Introduction to the Pascal Language 49
 - Introduction 50
 - 2.1 Programming Languages 50
 - 2.2 Describing a Programming Language 54
 - 2.3 Shifting to Pascal: Input, Output, and Assignment 56
 - 2.4 Order of Precedence of Arithmetic Operations 64
 - 2.5 The Arithmetic Operators +, -, *, DIV, and MOD 66
 - 2.6 ASCII Characters and the Data Types Char and String 67
 - 2.7 Redirecting Program Output: Text Files 73
 - 2.8 Viewing Screen Output 75
 - Summary 76
 - Review Problems 77
 - Exercises 78

- ▼ **CHAPTER 3 The Data Type Real 83**
 - Introduction 84
 - 3.1 What Is a Real Number? 84
 - 3.2 Declaring and Using Real Data 85
 - 3.3 The Arithmetic Operator / 91
 - 3.4 Declaring Constants 92
 - 3.5 Formatting Output 94
 - 3.6 Some Standard Arithmetic Functions 97
 - 3.7 How Real Numbers Are Stored in RAM (Optional) 100
 - Summary 105
 - Review Problems 106
 - Exercises 107

- ▼ **CHAPTER 4 Program Design, Debugging, and a Peek at Procedures 111**
 - Introduction 112
 - 4.1 Program Documentation 112
 - 4.2 Consistency in Style 114
 - 4.3 Top-Down Design and Stepwise Refinement: A Method for Program Development 116
 - 4.4 Modular Program Design: A First Look at Procedures 122
 - 4.5 The Turbo Debugger and Windows (Optional) 126
 - Summary 131
 - Review Problems 131
 - Exercises 133

- ▼ **CHAPTER 5 Pascal Decision Statements 135**
 - Introduction 136
 - 5.1 Relational and Logical Operators and Boolean Expressions 136
 - 5.2 IF ... THEN ... ELSE Statements 141
 - 5.3 Types of Errors and Error Messages 145
 - 5.4 Boolean Variables 147
 - 5.5 Compound Statements 150
 - 5.6 Nested IF ... THEN ... ELSE Statements 153
 - 5.7 The CASE Statement 156
 - 5.8 Logic Gates and Binary Addition (Optional) 160
 - Summary 165
 - Review Problems 166
 - Exercises 168

- ▼ **CHAPTER 6 Pascal Loop Statements 173**
 - Introduction 174
 - 6.1 Scalar and Ordinal Data Types 174
 - 6.2 FOR Loops 176
 - 6.3 WHILE Loops 185
 - 6.4 REPEAT Loops 190
 - 6.5 Comparing the Three Loop Statements 198
 - 6.6 More About Text Files 199
 - 6.7 Nested Loops 208
 - 6.8 Random Number Generation (Optional) 211
 - Summary 219
 - Review Problems 220
 - Exercises 221

- ▼ **CHAPTER 7 Procedures, Parameters, and Local Variables 229**
 - Introduction 230
 - 7.1 Value and Variable Parameters 230
 - 7.2 Local Variables, Scope, and Nested Procedures 243
 - 7.3 Turbo Pascal Units 253
 - Summary 261
 - Review Problems 261
 - Exercises 263

- ▼ **CHAPTER 8 Functions and Recursion 269**
 - Introduction 270
 - 8.1 The Difference Between a Function and a Procedure 270
 - 8.2 Functions of Various Data Types 276
 - 8.3 Indexing a String 282
 - 8.4 String Functions and Procedures 285
 - 8.5 An Introduction to Recursion 290
 - Summary 300
 - Review Problems 300
 - Exercises 302

- ▼ **CHAPTER 9 Arrays and Other User-Defined Data Types 307**
 - Introduction 308
 - 9.1 The Type Declaration 308
 - 9.2 Enumerated Types 309
 - 9.3 Subrange Types 313

- 9.4 Declaring and Manipulating One-Dimensional Arrays 317
- 9.5 A File Maintenance Program 332
- 9.6 Multidimensional Arrays 337
- Summary 345
- Review Problems 345
- Exercises 346

▼ CHAPTER 10 Sorting and Searching Arrays 351

- Introduction 352
- 10.1 Timing an Instruction Sequence 352
- 10.2 A Program to Compare Sort Algorithms 355
- 10.3 The Bubble Sort 359
- 10.4 The Shell Sort 362
- 10.5 The Quicksort (Optional) 369
- 10.6 Memory Management and the Stack (Optional) 374
- 10.7 A Program to Compare Search Algorithms 376
- 10.8 The Linear Search 380
- 10.9 The Binary Search 383
- Summary 387
- Review Problems 387
- Exercises 389

▼ CHAPTER 11 Building a Larger Program 393

- Introduction 394
- 11.1 The Include Directive, .LIB Files, and a Driver 394
- 11.2 The Window Procedure 397
- 11.3 Beefing Up OURSTUFF 404
- 11.4 Using Stubs in Top-Down Design 407
- 11.5 Putting It All Together 414
- Summary 425
- Review Problems 425
- Exercises 426

▼ CHAPTER 12 Sets (Optional) 429

- Introduction 430
- 12.1 Data Structures and Abstract Data Types 430
- 12.2 Set Notation and Set Operations 430
- 12.3 Pascal Implementation of Sets 433
- 12.4 Two Programs that Use Sets 440

12.5 Memory Allocation for a Set Variable (Optional) 448

Summary 450

Review Problems 451

Exercises 452

▼ CHAPTER 13 Records and Files 455

Introduction 456

13.1 Records 456

13.2 Arrays of Records 461

13.3 Typed Files and Files of Records 476

13.4 Maintaining a File of Records 500

Summary 512

Review Problems 512

Exercises 514

▼ CHAPTER 14 Pointer Variables and Dynamic Structures 519

Introduction 520

14.1 Dynamic Variables and Pointers 520

14.2 Allocating and Deallocating Memory 522

14.3 Pointers and the Heap 524

14.4 Linked Lists Using Pointers 529

14.5 Binary Trees 548

Summary 572

Review Problems 573

Exercises 574

▼ CHAPTER 15 Introduction to Object-Oriented Programming 579

Introduction 580

15.1 Why Have Another Revolution? 580

15.2 What Is an Object? 581

15.3 Adding Capabilities to an Object Class 585

15.4 Some Graphics Objects 587

15.5 A Linked List Object 593

15.6 Inheritance: Passing Characteristics from One Type to Another 598

15.7 Virtual Methods: Changing How Descendants Behave 603

Summary 611

Review Problems 612

Exercises 612

Appendices

- A. MS-DOS Essentials A-1**
- B. Model Processor Programming B-1**
- C. Turbo Pascal Reserved Words and Standard Identifiers C-1**
- D. Using Turbo Pascal D-1**
- E. Solutions to Odd-Numbered Chapter Review Problems E-1**
- F. Glossary F-1**
- G. Unit OURSTUFF G-1**
- H. ASCII Table H-1**

Index I-1