TURBO PASCAL

FOURTH EDITION

NELL DALE • CHIP WEEMS

Introduction to

Turbo Pascal and Software Design

FOURTH EDITION

Nell Dale
University of Texas, Austin

Chip Weems
University of Massachusetts, Amherst

Jones and Bartlett Publishers
Sudbury, Massachusetts

Boston London Singapore

Editorial, Sales, and Customer Service Offices
Jones and Bartlett Publishers
40 Tall Pine Drive
Sudbury, MA 01776
508-443-5000
info@jbpub.com
http://www.jbpub.com

Jones and Bartlett Publishers International
Barb House, Barb Mews
London W6 7PA
UK

Photo credits: page 14, Eidgenossiche Technische Hochshule (ETH), Zurich, Switzerland; page 16 (top, middle r, bottom, l-r), page 17 (top, bottom l) page 21 (bottom), page 23 (top l, middle l), page 142, Courtesy IBM; page 16 (middle l), page 17 (bottom), Courtesy of Hewlett-Packard Company; page 21 (top), Courtesy Digital Equipment Corporation; page 22, Cameramann International; page 23 (top r, bottom r), Courtesy Apple Computer, Inc.; page 23 (middle r) Courtesy Sun Microsystems, Inc.; page 23 (bottom l), Courtesy Thinking Machines Corporation. Photo: Steve Grohe; page 134, Courtesy of BASF; page 189, The Image Works © Topham; page 282, Culver Pictures; page 374, The Bettmann Archive; page 471, Courtesy of the U.S. Navy.

ISBN 0-7637-0608-6

Library of Congress Catalog Card Number: 94-70364

Printed in the United States of America
00 99 98 10 9 8 7 6 5 4 3

This book is dedicated to you,
and to all of our other students for whom it was begun and
without whom it would never have been completed.

***Turbo Pascal, Fourth Edition* Program Disk**

Jones and Bartlett Publishers offers free to students and instructors a program disk with all the complete programs found in *Turbo Pascal, Fourth Edition.* The program disk is available through the Jones and Bartlett World Wide Web site on the Internet.*

Download Instructions

1. Connect to the Jones and Bartlett student diskette home page (http://www.jbpub.com/disks/).

2. Choose *Turbo Pascal, Fourth Edition.*

3. Follow the instructions for downloading and saving the *Turbo Pascal, Fourth Edition.*

4. If you need assistance downloading a Jones and Bartlett student diskette, please send email to help@jbpub.com.

Downloading the *Turbo Pascal, Fourth Edition* program disk via the Jones and Bartlett home page requires access to the Internet and a World Wide Web browser, such as the Netscape Navigator or Microsoft Internet Explorer. Instructors at schools without Internet access may call 1-800-832-0034 and request a copy of the program disk. Jones and Bartlett grants adopters of *Turbo Pascal, Fourth Edition* the right to duplicate copies of the program disk or to store the files on any stand-alone computer or network.

Preface

Since its debut in 1983, *Introduction to Pascal and Structured Design* has led the field in introducing new topics and pedagogy because, when revising the text, our philosophy is to be proactive rather than reactive. We consistently set trends rather than follow them. As always, our efforts are directed toward making the sometimes difficult concepts of computer science more accessible to all students. The fourth edition of the Turbo version contains many new topics, but the new material is valueless if it is not presented in a way that students can understand—and use.

The previous editions of this book have been widely accepted as model textbooks for the ACM-recommended curriculum for CS1 and the Advanced Placement A exam in computer science. We believe that this edition also will become a model. It reflects our view of the future direction of computer science education—with more rigor, more theory, greater use of abstraction, and the application of software engineering principles throughout the programming process.

Special Sections

In this fourth edition we continue to use boxed sections to emphasize important concepts. The *Theoretical Foundations* boxes contain material that, although not essential to achieving the goals of the text, is fundamental information that every computer science major should know. These sections include such topics as data representation, finite state machines, methods of parameter passing, and complexity of algorithms.

Even though software engineering principles are integrated throughout the book, we use Software Engineering boxes to reemphasize special points. For example, the importance of using named constants is reinforced with a boxed discussion.

We feel strongly that our students should be aware of the contributions made to our field by previous generations. Thus we have included biographies of such people as Blaise Pascal, Ada Lovelace, and Grace Murray Hopper in sections called *May We Introduce.*

Changes in the Fourth Edition

The changes in this revision have been driven by three major factors.

- Turbo Pascal has become more of a standard than the ISO standard.
- Programmers are using a more modern approach to the classification of data types.

- Our belief that it is imperative to introduce and emphasize abstraction to the next generation of computer science students early in the curriculum.

In previous editions of *Turbo Pascal*, we introduced the Turbo extensions with the warning that it is better to use the standard when possible. Our philosophy has changed with this edition—the nonstandard Turbo features, the unit and the string, are essential to modern computer science theory and software engineering practice. It is ironic that almost all Pascal compilers have these extensions; only the ISO standard does not. Therefore we emphasize these features rather than warn against their use, although we continue to point out which constructs are not ISO standard.

We have grouped the *nonstructured* composite types (the record and the set) together. Records traditionally have been viewed as a structured type and covered following arrays. Records are structured from the compiler writer's view because a base address and offset method are used to implement them. However to the *user* of the record data type it is unstructured. The order in which the fields are declared can be changed, and it does not affect the code because the fields are accessed by name. An advantage of introducing records before arrays is that access by name is easier to comprehend than the indirect access used with arrays.

We have grouped the *structured* composite types (the file and the array) together. Files and arrays are conceptually very similar; they differ only in their access methods. For some reason students have trouble with files. (One of us grades advanced placement computer science exams and can attest to this fact.) We hope that by grouping files with arrays and making their similarities and differences explicit, we can clarify this problem area for students.

We also have moved the chapter on units from the end of the book to the chapter immediately following the introduction of the built-in Pascal composite types. Thus instructors now can use units to demonstrate information hiding and data encapsulation throughout the balance of the text.

The modifications in this edition are so sweeping that we have changed the name of the book to *Introduction to Turbo Pascal and Software Design.* The new title reflects the change in emphasis in the second half of the book from solving one-time problems using structured design to creating general-purpose software modules that can be used in a variety of problem solutions. The top-down design of a problem now interfaces with these previously created and tested modules.

Synopsis

Chapter 1 is designed to create a comfortable rapport between the student and the subject. The basics of hardware and software are presented, and problem-solving techniques are discussed and reinforced in a Problem-Solving Case Study. We have added a section on ethics to round out this introduction.

Because of the increased emphasis on Turbo Pascal features, the old Chapter 2 has been reorganized into two chapters. In the new Chapter 2, we introduce strings as the first data type that students use and show them how to concatenate and output strings. Then in Chapter 3 we introduce numeric data types and operations, thus giving the students a powerful set of tools that allows them to write more interesting programs from the beginning.

Top-down design methodology and input are covered in Chapter 4. To reinforce the discussion of strings, Turbo string input is discussed in detail. We introduce Boolean expressions and the IF statement in Chapter 5 and looping with the WHILE statement in Chapter 6. Although many instructors prefer to cover all the looping statements together, we believe that the theory of loops is more important than the syntax and that the theory can best be emphasized by using only one syntactic construct. In both Chapters 5 and 6, we stress the theory of flow of control represented in the selection and iteration constructs rather than the syntax of the constructs themselves. Pre- and post-conditions are covered in Chapter 5 and loop invariants in Chapter 6.

The next three chapters are devoted to designing and writing subprograms. Built-in procedures and functions are introduced in Chapter 1 and referred to in each subsequent chapter. By Chapter 7, students are quite comfortable with using subprograms and receptive to the idea of writing their own. Chapter 7 covers flow of control in procedures, formal and actual parameters, local variables, and interface design. Chapter 8 expands the discussion to consider value parameters, nested scope, stubs and drivers, and more on interface design. Chapter 9 describes user-defined functions, with a brief discussion of recursion. Because of Chapter 9's numerical orientation, we also discuss the problems of representation and precision associated with real numbers.

Chapters 1 through 9 emphasize control abstraction. Chapters 11 through 17 feature data abstraction. Chapter 10 provides the transition by formalizing the concept of a data type, discussing ordinal and scalar data types along with the operations provided for them, and showing students how to create user-defined data types. Because the CASE statement and the FOR statement require the use of ordinal data types, we have introduced the rest of the control structures in this chapter. CASE, FOR and REPEAT/UNTIL are the "ice cream and cake" of the control structures—nice to have, but not necessary.

We begin Chapter 11 with a discussion of composite data types and the distinction between unstructured and structured data types. We introduce the set and the record—Pascal's built-in, unstructured composite data types. We believe that the array traditionally has been introduced before the record because FORTRAN didn't have records. When the first Pascal texts came out, they were modeled on FORTRAN texts, and the new Pascal material was put at the end. This placement is artificial; access by name is more natural for students than access by index.

We group the structured composite data types, the file and the array, together in Chapter 12. Files and arrays are conceptually very similar; they differ only in their access methods. We discuss their similarities and differences in detail and give guidelines for when each is appropriate.

After the introduction of arrays the focus shifts from the syntax and semantics of the composite data types to their use in defining abstract data types. Because we already have introduced record and array, we can clearly differentiate the concepts of array and list from the beginning. The array is a built-in, fixed-sized data structure. The list is a user-defined, variable-sized structure represented by a length and an array of items bound together in a record. The elements in the list are those elements in the array from the first position through the length position.

In the last decade the topic of data structures has become subsumed under the broader topic of *abstract data types (ADTs)*—the study of classes of objects whose logical behavior is defined by a set of values and a set of operations. The term *data structures* refers to the implementation of data objects within a program; that is, the implementation of structured relationships. The term *abstract data type* refers to a collection of data values and operations in which operations are described by their logical behavior, not their implementation. The shift in emphasis from data structures to abstract data types is representative of the move towards more abstraction in computer science education. We believe that this shift is a crucial one, and it has driven the reorganization of the second half of this book.

Thus we define abstract data types in Chapter 13 and emphasize them throughout the balance of the book. We use the design of the unordered list as a way to guide the presentation of the algorithms on lists. In Chapter 14 we introduce the unit as a way for encapsulating our abstract data types, procedural parameters as a vehicle for more generality, and use an auxiliary unit as a technique for creating generic abstract data types. It is ironic that Standard Pascal always has allowed procedures and functions to be passed as parameters, but because Turbo Pascal—the compiler that most students use—has not had this feature, we (and most other authors) have not covered it. Turbo Pascal now supports procedural parameters.

In keeping with the move toward more abstraction, we introduce the object data type in great detail in Chapter 16 as an additional encapsulation mechanism. The progression from abstract data type encapsulated within a unit to the object data type in which the operations are bound within the object is smooth and logical. We end this chapter with a discussion of object-oriented design and programming.

The pointer data type, referenced variables, and dynamic data structures are covered in Chapter 17. To reinforce the concepts of abstraction, we replace the array-based implementation of the ordered list ADT with a linked implementation without changing the interface section of the unit encapsulating it. We also discuss a linked list of objects.

Chapter 18 covers recursion. There is no consensus as to the best place to cover this subject. We believe that it is better to wait until at least the second semester to cover this topic. However, we have included recursion for those instructors who have requested it. We divide the examples into two parts: those that require only simple data types, and those that require structured data types. Professors can cover the first part after Chapter 9. The second part contains examples from simple lists to binary trees. Instructors may use these examples individually after the appropriate chapter (for example, simple arrays after Chapter 12) or as a group after Chapter 17.

Chapter 19 covers graphics, the subject that is often the most popular with students. As with recursion, no consensus exists on when to introduce graphics or even if the topic should be covered in an introductory course. We thus have written Chapter 19 in a similar manner to Chapter 18. Most of the material may be assigned at any point after Chapter 2. The remainder of the graphics coverage depends on an understanding of two-dimensional arrays and may be used at any point after Chapter 15.

Additional Features

Goals Each chapter begins with a list of learning objectives for the student. These goals are reinforced and tested in the end-of-chapter exercises.

Problem-Solving Case Studies Problem solving is best demonstrated through case studies. In each case study we present a problem and use problem-solving techniques to develop a manual solution. Next we expand the solution to an algorithm, using modular design methodology, and then we code the algorithm in Pascal. We show simple test data and output and follow up with a discussion of what is involved in thoroughly testing the program.

Testing and Debugging Following the case studies in each chapter, this section considers in depth the implications of the chapter material with regard to thorough testing of programs. The section concludes with a list of testing and debugging hints.

Quick Checks At the end of each chapter are questions that test the student's recall of major points associated with each chapter goal. Upon reading each question, the student immediately should know the answer, which he or she can then verify by glancing at the answers at the end of the section. The page number on which the concept is discussed appears at the end of each question so that the student can review the material in the event of an incorrect response.

Exam Preparation Exercises These questions help the student prepare for tests. The questions have objective answers and are designed to be answerable with a few minutes of work. Answers to selected questions are given in the back of the book, and the remaining questions are answered in the *Instructor's Guide*.

Programming Warm-Up Exercises This section provides the student with experience in writing Pascal code fragments or procedures. The student can practice the syntactic constructs in each chapter without the burden of writing a complete program. Solutions to selected questions from each chapter appear in the back of the book; the remaining solutions may be found in the *Instructor's Guide*.

Programming Problems We have included specifications for problems from a wide range of disciplines in these exercises, which require the student to write complete programs.

Case Study Follow-Up A new kind of exercise in the fourth edition, Case Study Follow-Up questions require the student to analyze or modify the case studies in the chapter. These exercises afford experience in reading and understanding the documentation and code for an existing program.

Supplements

Instructor's Guide Prepared by the authors, the *Instructor's Guide* features teaching notes, answers to the balance of the exercises, a carefully worked-out solution and discussion for one programming problem per chapter, and an example of an advanced placement exam question, with a sample solution and the actual grading rubrics used by the AP exam graders.

Test Item File The *Test Item File* includes more than 1,300 test questions patterned after those in the Exam Preparation Exercises. Also available on request from the publisher is an electronic version of the *Test Item File*, in IBM PC format.

Transparency Masters Enlargements of many figures in the text and the code for the case studies are available for use in lecture presentations.

Program Disk The program disk available through the Jones and Bartlett web site contains all the programs in the text. Please see the instructions for downloading this disk on the page preceding the Preface.

A Laboratory Course for Turbo Pascal Available separately, this laboratory manual parallels *Introduction to Turbo Pascal and Software Design* in either an open or closed lab setting. It was written to help students master Turbo Pascal syntax through guided exercises, each within the context of a complete program. Having mastered the language syntax, students can then concentrate on problem-solving and algorithmic design. Each chapter is divided into three parts: *Prelab, Inlab,* and *Postlab*. The accompanying disk contains the programs, program shells (partial programs), and data files.

Casebook for Turbo Pascal This supplement contains a variety of new case studies that are of interest to students such as designing a recording studio, analyzing text, simulating traffic at an intersection, and forecasting a card game.

Acknowledgments

We would like to thank the many individuals who have helped us in the preparation of this fourth edition. We are indebted to the members of the faculties of the Computer Sciences Departments at the University of Texas at Austin and at the University of Massachusetts at Amherst.

From among our colleagues, we are especially grateful to Suzy Gallagher and Diane Law at Austin, and W. Richards Adrion, Carla Bradley, and Conrad Wogrin at Amherst. We extend special thanks to Jeff Brumfield for developing the syntax template metalanguage and allowing us to use it in the text.

For their many helpful suggestions, we thank the lecturers, teaching assistants, consultants, and student proctors who run the courses for which this book was written, and the students themselves.

We are grateful to the following people who took the time to review our manuscript: Dr. William E. Ayen, Colorado Tech; Larry Crockett, Augsburg College; Jack N. Donato, Jefferson Community College; Merrell R. Jones, Southern Utah University; James Roberts, Charleston Southern University; David R. Thomas, Texas A&M—Corpus Christi; and Andrew Townsend, Mira Costa College.

Anyone who has ever written a book—or is related to someone who has—can appreciate the amount of time involved in such a project. To our families—all the

Dale clan and extended Dale family (too numerous to name), and to Lisa, Charlie, and Abby—thanks for your tremendous support and indulgence.

N.D.
C.W.

Brief Contents

Table of Contents