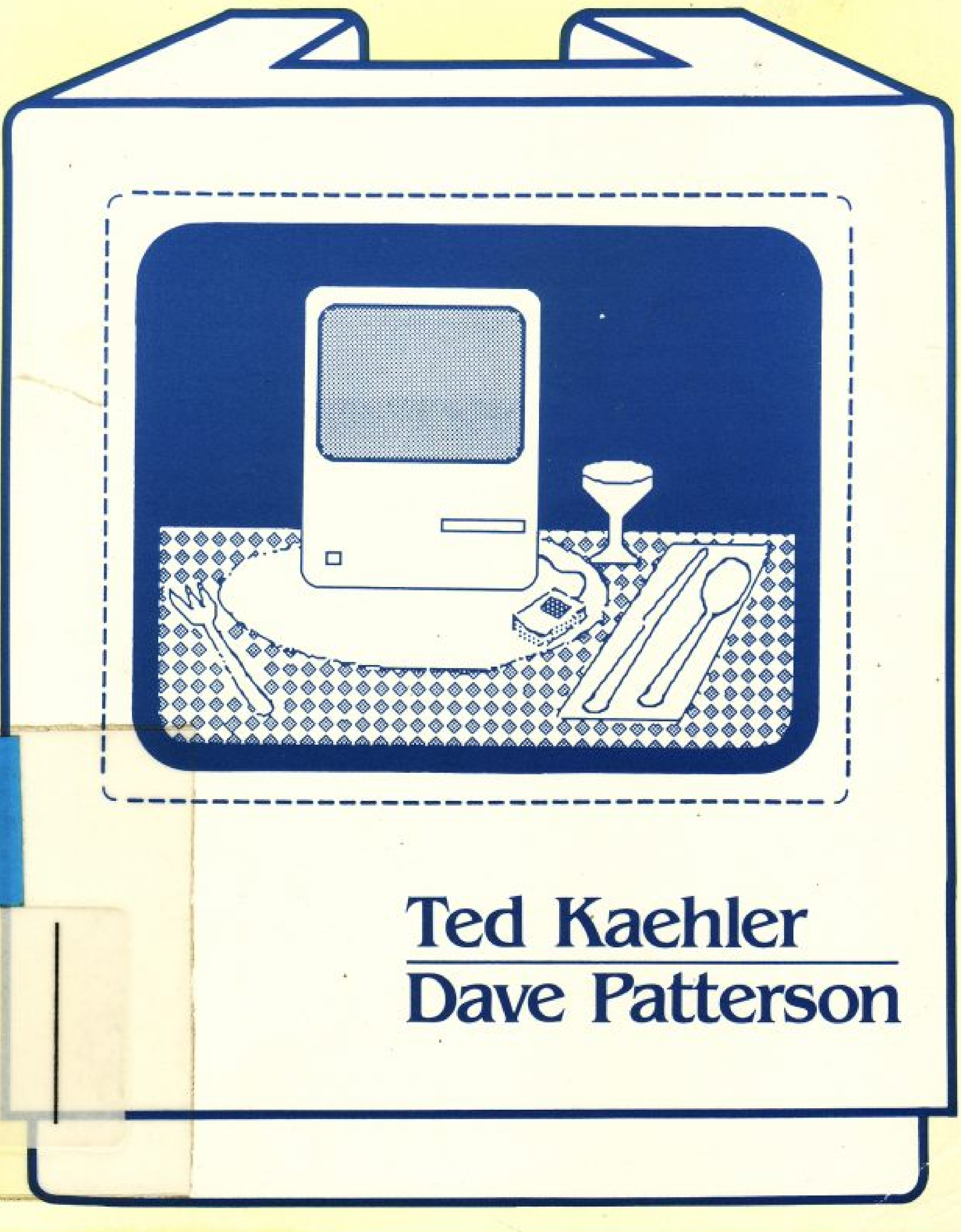
A TASTE OF SMALLTALK



Ted Kaehler

Dave Patterson

A TASTE OF SMALLTALK

Seeing is deceiving
It's eating that's believing
JAMES THURBER

TED KAEHLER
Xerox PARC and Apple

DAVE PATTERSON
University of California, Berkeley

W W NORTON & COMPANY NEW YORK LONDON

FIRST EDITION

ISBN D-393-95505-2

W. W. Norton & Company, Inc.,
500 Fifth Avenue, New York, N. Y. 10110
W. W. Norton & Company Ltd.,
37 Great Russell Street, London WC1B 3NU

1 2 3 4 5 6 7 8 9 0

CONTENTS

Preface
- The Book
- Relationship to the Smalltalk Reference Manuals
- A Language and an Environment
- Smalltalk Roots
- Acknowledgements

1. The Example 1
- The Tower of Hanoi 1
- A Rosetta Stone: Pascal, C, LISP, and Smalltalk 3

2. Messages and Objects Everywhere 9
- The Smalltalk-80 Language 9
- Defining a Method 14
- Troubleshooting When You **accept** a Method 25

3. Running the Example 29
- Continuing the First Example 29
- Running a Program 30
- Troubleshooting Runtime Errors 35
- The Second Smalltalk Example 36
- Installing the hanoi Method 38
- So You Don't Want To Type This All in Again 41

4. Defining a Class 44
- The Class TowerOfHanoi 44
- How to Create a New Class 48
- Recapping the Smalltalk Terminology 57

5. Animating the program 64
- A Second Class 64
- Installing the Classes HanoiDisk and AnimatedTowerOfHanoi 76

6. An Algorithm for the Rest of Us 83
- Move a Disk by Moving an Object 83
- Defining the Classes HanoiDiskRules and TowerByRules 94

7. On Your Own 99
Finding What You Need 99
What Is Object-Oriented Programming, Really? 109

Appendix 1: The Smalltalk Text Editor 113
Appendix 2: How to Talk to Yourself When Reading Smalltalk 115
Appendix 3: Methods Missing from the Apple Level 0 Image 117
Appendix 4: Exercises 120
Appendix 5: Hints for the Exercises 122
Appendix 6: Answers to the Exercises 124
Index 129

Everything new meets with resistance.
RUSSIAN PROVERB

THE BOOK

The Smalltalk-80™ system* is both a civilized programming environment and the premier "object-oriented" programming language. The programming environment offers an elegant and highly integrated set of tools: overlapping windows on a diverse set of applications appear together on the screen, and a single button click moves the user from one application to another. The system makes it easy to search, view code, edit, detect syntax errors, and execute code, all from within a single window. As a language, Smalltalk offers a uniform and powerful metaphor: procedures and data that belong together are packaged in an "object." An object interacts with the rest of the system by singling out another object and sending it a message. Smalltalk's combination of good editors, a natural modularization of code, and a language based on a powerful idea forms a system that is at its best during the construction and evolution of a large applications program.

A new programming system is enticing to most programmers, especially when it promises increased productivity. Alas, this enthusiasm wanes when the programmer confronts the mountain of documentation that must be conquered before writing a first, simple program. The dilemma is compounded when the new system uses terms and ideas that are unlike any encountered before. The delightfully slothful

* Smalltalk-80 is a registered trademark of the Xerox Corporation. In other corporate news, UNIX is a trademark of AT&T, VAX is a trademark of Digital Equipment, SUN is a trademark of Sun Microsystems, IBM is a trademark of International Business Machines Corp., and Macintosh is a trademark licensed to Apple Computer, Inc.

programmer generally looks for friends to guide him. Without such guides, the programmer returns to the old familiar programming environment, feeling secure, if slightly disappointed.

We must confess that this insight comes from personal experience and not from scholarly observation. When students at the University of California at Berkeley first confronted a working Smalltalk-80 system, they found that they didn't know what to do next. Although they knew a lot about the Smalltalk language and its user interface, they didn't know where to begin. We wrote this book to tell them, and to provide a gentle introduction to Smalltalk for the well-adjusted slothful programmer. We suspect that there are many people of kindred spirit who want to learn the Smalltalk-80 system—perhaps even you.

The purpose of this tutorial is to give you a taste of programming in the Smalltalk-80 language and environment. Our approach is to guide you through several versions of a small program in a short time. If these small examples get you interested, then larger programs will get you addicted, for the power of the Smalltalk-80 system increases as programs grow.

We introduce the system a piece at a time, taking as our example your favorite recursive program, the Tower of Hanoi, and giving complete programs with explanations. Rather than starting from first principles, we use the "Rosetta Stone" approach: in Chapter 1, we show Pascal, C, and LISP versions (it will help if you know one of these languages), and then present a Smalltalk version for comparison. The next three chapters show how to enhance the program and turn it into a Smalltalk class. Along the way we tour several parts of the Smalltalk-80 system that will be useful in other applications. In Chapter 5, we'll spice up the program by making the disks fly between the towers on the graphics display, while in Chapter 6 we depart from the standard recursive solution and present a heuristic algorithm, which nevertheless uses our original data structures to solve the puzzle. Through rewriting the Tower of Hanoi program we demonstrate the value of "object-oriented" programming and show how the proper use of objects can make a program intuitive and easy to modify.

We assume you know how to program and have seen how the mouse interacts with windows on a bit-mapped display. This tutorial is best if you have a Smalltalk-80 system to try it on. The book is also valuable if you don't have access to a running system but just want to find out what Smalltalk is really like. To give you a feeling for Smalltalk, the system, we have included many pictures of the screen. By using this book with any of several commercially available Smalltalk-80 systems, you can write an interactive, graphical Smalltalk program in a day or less. You will then be in a better position to decide if you want to learn more about Smalltalk, and, perhaps more significant, you can amaze your programming friends with phrases like, "Well, I've *written* a Smalltalk-80 program, and I think ..."

RELATIONSHIP TO THE SMALLTALK REFERENCE MANUALS

Some books are to be tasted,
others to be swallowed,
and some few to be chewed and digested.
FRANCIS BACON, *Of Studies*

1983 was a landmark year in Smalltalk's history, as the number of Smalltalk books went from zero to three. The first book, *Smalltalk-80: The Language and Its Implementation,* by Adele Goldberg and David Robson (Reading, Massachusetts: Addison-Wesley, 1983), plays such an important role that we will refer to it as the "Blue Book." It describes the language and interpreter in detail, thereby defining the Smalltalk-80 language. In addition, the Blue Book gives an example of the design and implementation of a moderate-sized program.

Smalltalk-80: The Interactive Programming Environment, by Adele Goldberg (Reading, Massachusetts: Addison-Wesley, 1984), documents the manner in which Smalltalk presents itself on the screen. It catalogs and explains the various windows, menus, and editors in the Smalltalk-80 programming environment, and what the user must do to run them. We refer to it as the "User's Guide."

Glenn Krasner edited *Smalltalk-80: Bits of History, Words of Advice* (Reading, Massachusetts: Addison-Wesley, 1983), a collection of papers describing several implementations of Smalltalk. His book is for people interested in building Smalltalk-80 systems, and is not intended for those who only want to use such systems.

In contrast to these titles, *A Taste of Smalltalk* does not tackle the syntax and semantics of the language head-on, but eases you into the Smalltalk system by translating a familiar program from another language. The example is then expanded to show off the interactive nature of the system. In short, this book is for programmers who want to build some example programs in Smalltalk with as little preparation as possible.

A LANGUAGE AND AN ENVIRONMENT

The Smalltalk language is based on a small number of consistent abstractions. Like LISP, Smalltalk seeks to provide uniform treatment of different kinds of information—text, graphics, symbols, and numbers. By packaging the behavior of each form of information with the actual data, the information can be shared between programs without changing representations. Often, code is general enough to be "reused," reducing the size of programs.

The source code for every part of the Smalltalk-80 system is available in System Browser windows. A Smalltalk programmer typically uses an editor to build upon an existing part of the system by modification. Overlapping windows make it possible to see several pieces of your code and several pieces of system code at once while you are debugging (Smalltalk was the first system to allow users to do this).

Smalltalk is an exploratory system, in which it is easy to make changes and test prototypes. Pieces of Smalltalk code tend to be small; they are compiled incrementally and take effect instantly. Such a system invites capricious change, which often leads to innovation and improvement. The goal is to provide an environment in which you can take a fall and get up again faster than you could plan a careful, conservative course of action. The text editor and many other parts of the Smalltalk-80 system were built from within the running system—an exhilarating and surprisingly safe experience. Smalltalk is at its best in large applications. The tools and editors are ideal for the frequent changes that are necessary in large software projects.

SMALLTALK ROOTS

Smalltalk was developed in the Learning Research Group at Xerox's Palo Alto Research Center in the early 1970s. The major ideas in Smalltalk belong to Alan Kay, who traces their roots to Simula, LISP, and SketchPad. Early implementations were masterminded by Dan Ingalls (the first person to write code for multiple overlapping windows, opaque pop-up menus, and BitBlt*). 1972 to 1979 were Smalltalk's formative years, and the Smalltalk-80 system is a "presentable" and documented version of Smalltalk-76.

Smalltalk runs on a personal computer with a high-resolution display and a pointing device. While the language is fairly concise, the environment is rather large, though not nearly as large as UNIX™.

* BitBlt, also called "raster-op," is a general-purpose operation for moving rectangles of bits from any position to any other position on a bit-mapped display.

Smalltalk has taxed every machine on which it has run, including Xerox Altos, VAXes, SUNs, and Xerox Dorados, but the recent growth in power and speed of microcomputers has made it practical to implement Smalltalk on low-cost machines. From engineering workstations Smalltalk has spread to increasingly inexpensive personal computers. The vendors offering Smalltalk-80 systems as of January 1986 are given in the table below.

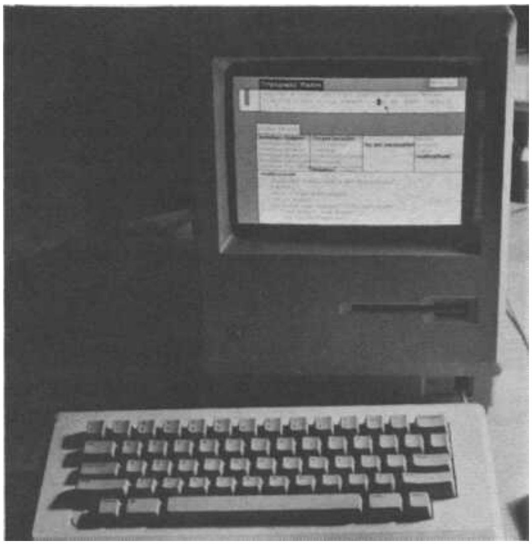
Machine	Where to Get the Software	License
Xerox 1100, 1108x, 1132	Xerox	2
Tektronix 4404, 4405, 4406	Tektronix	2
Sun 2	U.C. Berkeley*	2
IBM PC AT	Softsmarts, Inc.**	2
Apple Macintosh 1MB	Apple***	1 (Level 1)
Apple Macintosh 512K	Apple	1 (Level 0)

* Berkeley distributes an interpreter written in C that runs under UNIX. Before you can buy Berkeley's interpreter, you must obtain a license for the Smalltalk-80 system from Xerox.
** In case you haven't heard of them, Softsmarts, Incorporated is located in Woodside, California. They offer the genuine License 2 Smalltalk-80 system from Xerox on an IBM PC AT.
***The Apple Level 1 Smalltalk system runs on a Macintosh Plus, a Macintosh that has been upgraded to 1MB, or a Macintosh XL running MacWorks. The Apple Level 1 and Level 0 systems are available from Apple as unsupported releases and are not official products as of this writing.

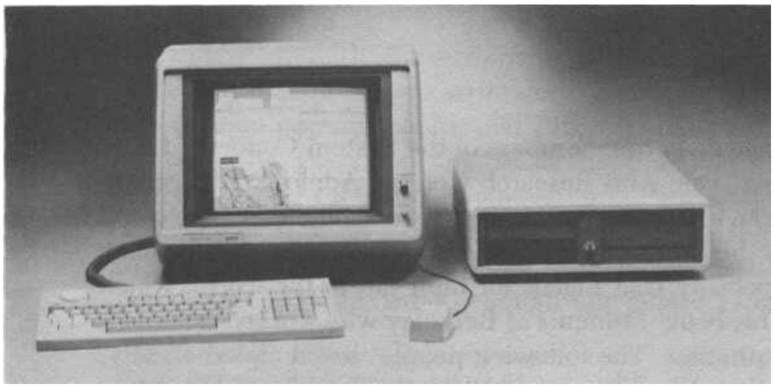
Two different versions of the Smalltalk-80 system are licensed by Xerox. License 2 is the standard system documented in the reference manuals, and is the system that Xerox is actively licensing to computer manufacturers and universities. License 1 is an older system that is available from Apple. A *Taste of Smalltalk* primarily describes License 2 systems, but does include instructions for using License 1 systems whenever possible.

ACKNOWLEDGMENTS

We wish to thank all the members of the System Concepts Laboratory at the Xerox Palo Alto Research Center. Adele Goldberg and Dave Robson wrote the reference manuals for Smalltalk, and we could not have written this without their previous work. We wish to thank Alan Kay and especially Dan Ingalls for their gift to the world, which we celebrate in this book. Students at Berkeley who worked on Smalltalk provided inspiration. The following people, listed alphabetically, tested the tutorial and gave us valuable feedback: Bill Baldwin, Ron



Clockwise from above: The complete License 2 version of Smalltalk-80 running on the IBM PC AT (photo courtesy of Softsmarts, Inc.); a 512K Macintosh with one disk drive running Apple's Level 0 Smalltalk system (photo by Mark Embury, Palo Alto, California); Smalltalk-80 on the Tektronix 4404 Artificial Intelligence System (photo courtesy of Tektronix, Inc.).



Carter, Yuchee Chen, Jeanetta Cooper, Bill Croca, Al Hoffman, Peter K. Lee, Mark Lentczner, Frank Ludolph, Randy Smith, and Marvin Zauderer. Lissy Bland wrote an early version of the answers to the exercises, and Mark Lentczner wrote the programs that answered a later version. Adele Goldberg, Glenn Krasner, Larry Tesler, SueAnn Ambron, and Carol Kaehler reviewed the manuscript and made many vital suggestions. Todd MacMillan provided technical assistance. We also thank John Hawkins of W. W. Norton for his profoundly steadying influence on this project. Carol Kaehler (the real writer in one family) and Linda Patterson (the real artist in the other) provided inspiration, and this book is dedicated to them.