# BASIC

## A Guide to Structured Programming

Dwyer Critchfield Shore Kaufman

# BASIC

## A Guide to Structured Programming

**Thomas A. Dwyer**
**Margot Critchfield**
**J. Michael Shore**
University of Pittsburgh

**Michael S. Kaufman**



**Houghton Mifflin Company** **Boston**
Dallas Geneva, Illinois Hopewell, New Jersey Palo Alto

The photos in Figures 1.2 and 1.5 are courtesy of Radio Shack. All other photos, and the color photo for the cover, are by Margot Critchfield. The program that produced the cover art is described in Section 9.6 of this book.

Use of the AT&T logo is courtesy of AT&T Communications.

Use of the Commodore logo is by permission of Commodore Electronics, Limited.

Use of the Heathkit logo is courtesy of Heath, Inc.

Use of the IBM logo on pages 7, 9, 10, 16, 49, 51, 58, 135, 136, 152, 168, 209, 210, 223, 224, 230, 231, 233, 259, 264, 269 is by courtesy of International Business Machines Corporation.

Use of the Kaypro logo is courtesy of Kaypro Corporation.

Use of the Radio Shack logo is courtesy of Radio Shack, A Division of Tandy Corporation.

Use of the Zenith logo is courtesy of Zenith Electronics Corporation.



Printed in the U.S.A.

Library of Congress Catalog Card Number: 84-61356

ISBN: 0-395-35653-9

ABCDEFGHIJ-SM-898765

# Preface

In 1980, a friend of ours adopted a 5 pound "shepherd-mix" puppy from the local animal shelter. He promptly named him BASIC, and then spent the months ahead explaining to people that the pup's name wasn't all that strange, especially when you knew that it was also the name of a computer programming language.

By 1984, two noticeable changes had taken place. BASIC weighed in at 90 pounds plus, and most people who were told his name smiled and said "Oh, yes—just like the computer programming language."

The level of people's awareness of the computer culture has increased even more dramatically in recent years. In particular, the number of persons who are familiar with the programming language BASIC now numbers in the millions, and it's growing at an astonishing rate. There may soon be as many people acquainted with BASIC programming as there are with the three R's.

The purpose of this book is to help students join this group, but in a rather special way. Our goal is to guide beginning programmers in mastering the art and science of *professional* BASIC programming, but without forsaking the informal aspect of BASIC that makes its use such a satisfying experience.

This goal presents some problems, of course. How does one reconcile the precision demanded by professionalism with the informality of BASIC, an informality deliberately built into the language so that computer programming might be accessible to beginners of all ages and backgrounds? This dilemma is akin to that faced by the flight instructor who knows that fledgling pilots will never become real professionals without discipline—and lots of it. But the same instructor knows that to try and communicate the hundreds of techniques that define that discipline during the first few hours of instruction is an exercise in futility. While learning to program may not be quite as traumatic an experience as learning to fly, the same kind of futility can easily be experienced by novice programmers who are given too many techniques too soon.

Our approach to resolving this dilemma has been to organize the content and style of the book in two parts. Part I (Chapters 1 through 4) introduces the fundamental features of BASIC in a relaxed, low-key manner. These opening chapters also contain numerous short but useful examples, written for the most part in minimal BASIC. While short examples do not permit meaningful discussion of the modern concept of structured programming, they allow one to quickly gain the experience that is vital to understanding the philosophy behind this (and other) techniques of professional programming.

Part II of the book (Chapters 5 through 12) is called "Professional BASIC." In addition to introducing the many advanced features of extended

BASIC, it emphasizes structured problem solving and structured program design. These ideas appear throughout the book, but formal discussions of the techniques involved are reserved for the later chapters. By this time the student will have had time to become comfortable with the many procedural details that seem so complicated at first, but that soon become second nature. The techniques of structured program design can then be introduced, not as isolated theory, but as powerful aids in the development of new programs that go beyond the limits traditionally associated with first courses.

In particular, Part II illustrates the use of structured design methodology in connection with programs related to statistical grade analysis and string manipulation (Chapter 6); modern searching and sorting techniques (Chapters 7 and 8); simulations, games, and graphics (Chapter 9); menu-driven software, data encryption, and word processing (Chapter 10); data processing and sequential file manipulation (Chapter 11); and random access files and database programs (Chapter 12). A series of colored headings is used in the margins of these chapters to show the connection between the structured design of specific examples and the general principles of design discussed in the text.

The principal dialect of BASIC used throughout the book is Microsoft Disk Extended BASIC, which is very similar to BASIC PLUS. Microsoft BASIC is available on all the Radio Shack TRS-80 and most Commodore microcomputers; the Apple II, IIe, IIc, and Macintosh; the IBM PC and PC Jr; and microcomputer systems that use either the CP/M or MS-DOS operating systems (for example, Tandy, Zenith, Heathkit, Monroe, DEC Rainbow, AT&T, Sanyo, and Kaypro). BASIC PLUS is used on the time-sharing systems supplied by the Digital Equipment Corporation (for example, the PDP-11 and VAX series) and on the DEC 350 personal computer. When the text discusses features that apply to a specific computer, the logo of that computer manufacturer (for instance, IBM, Tandy, or Apple) appears in color in the margin.

To help support the general discussions in the text, and the more than 150 program examples used to illustrate these discussions, the book contains numerous exercises and programming problems. Those designated as *Exercises* are paper-and-pencil explorations. Those called *Lab Exercises* serve as guides to doing work directly on a computer. In addition, each chapter ends with a comprehensive summary, followed by a collection of more demanding problems and projects.

Most of the projects are connected with writing programs that have real applications. The principal piece of advice we would offer anyone is to do as many of these projects as possible on either a home or school computer, allowing plenty of time to experiment. Also talk to others, and share ideas. Then experiment some more. You can't hurt anything, but if you persevere with this stratagem, you're in for a treat. Not only will things get easier, but they'll take on a fascination that will return your investment of time and energy many times over.

## Acknowledgements

The authors wish to thank Sherrilyn Reiter, who typed much of the manuscript and special technical material in the appendices. We are grateful to the many reviewers whose valuable suggestions helped shape the book's content and peda-

gogical style. Particular thanks go to Martha Baxter of Mesa Community College, Louis A. DeAcetis of Bronx Community College, John J. DiElsi of Mercy College, Wesley Fasnacht of West Chester State College, June Fordham of Prince George's Community College, James Gips of Boston College, John B. Lane of Edinboro State College, C. Gardner Mallonee II of Essex Community College, Theodore V. Smith of Broward Community College, and Kenneth W. Veatch of San Antonio College.

T.A.D.
M.C.
J.M.S.
M.S.K.

## About the Authors

**Thomas Dwyer** is Professor of Computer Science at the University of Pittsburgh. He is coauthor with Margot Critchfield of a dozen books on computing, including *CP/M and the Personal Computer, A Bit of IBM BASIC, A BIT of Applesoft BASIC,* and *Structured Program Design with TRS-80 BASIC.*

**Margot Critchfield** holds her Ph.D. from the University of Pittsburgh where she teaches computer programming.

**J. Michael Shore** teaches computing at the University of Pittsburgh and Taylor Allderdice High School.

**Michael Kaufman** is a graduate of the Harvard Law School and is presently practicing law in New York.

# Contents

PART

# I

# Informal BASIC

# CHAPTER ONE

# Getting Acquainted with a Computer

NEW TOPICS INTRODUCED IN CHAPTER 1

- *Digital computers*
- *Microcomputer systems*
- *Central processor/memory components*
- *Floppy disks*
- *Login procedures*
- *Modems*
- *Entering and running BASIC programs*
- *Personal computers*
- *Input/output components*
- *Mass-storage components*
- *Time-sharing systems*
- *Acoustic couplers*
- *BASIC interpreters and compilers*
- *Correcting errors in programs*

## 1.1 The Plan for Part I

A good starting place for any book on computing is to try to answer the question: What can a computer do? There are thousands of answers to this question, but in general they all boil down to this: A computer can do those things that can be explained to it in terms of instruction sets called *computer programs.*

Generally speaking, there are two kinds of computer programs. First, there are customized programs written by the person using the computer (or by a consultant the user has hired) to do something special—something of unique interest to that user. Second, there are "off-the-rack" or packaged programs of a more general nature. These are of greatest value to users whose needs match the capabilities built into the packages by their original designers.

Three examples of packaged programs will be shown later in this book. One is an electronic spreadsheet program, shown in Chapter 9. The second is a word-processing program, discussed in Chapter 10 (where you'll also be shown how to write your own simplified word-processing program). The third is a database-management program, illustrated in Chapter 12.

The primary purpose of this book, however, is to show you how to write your own customized programs, using a computer programming language called BASIC. It's something like showing a hi-fi enthusiast how to assemble a customized stereo system. Selecting and interfacing the components for such a system takes professional know-how, and not everyone will want (or be able) to go this route. But when you consider the level of insight, personalized control, and pride of accomplishment associated with the customized approach, this option has a lot more going for it than anyone might suspect.

The goal of Part I (the first four chapters of the book) is to help a beginner get started programming in BASIC with a minimum of fuss. For that reason, these chapters say very little about the theory of program design (that's coming in Part II). The idea is to get you acquainted with small, but real, programs that are geared to taking the mystery out of programming as soon as possible.

The best strategy for making this plan work will be to mix "off-line" preparation with "on-line" practice. The term *on-line* means working at the keyboard of a computer (or a terminal connected to a computer), entering and executing programs *interactively,* that is, in a manner that lets you immediately see the results of your work on the computer's display device.

To guide you in preparing for on-line work, the text is sprinkled with self-study sections called *Exercises.* For example, Exercise 2.1 is the first exercise of Chapter 2. It is based on material in the first part of Chapter 2, and it helps prepare for the actual computing work that follows.

Directions for on-line experimentation are given in the form of *Lab Exercises.* Each lab exercise is associated with a program that (like all the other programs in the book) has been given a unique file name. For example, Lab Exercise 2.1 (file name ARITH) is the first lab exercise in Chapter 2, and it guides you in working on-line with a program called ARITH. (The rules for inventing file names will be explained in Section 2.5.)

There is also a collection of programming projects at the end of each chapter. In general, these are best approached as a combination of off-line planning and on-line experimentation, repeated in a cycle that emphasizes both aspects of program design.

Off-line planning and documentation

On-line testing and experimentation

## 1.2 Types of Computers

The full name for the kind of computer we will study is "general-purpose digital computer." From now on we'll refer to such machines simply as *computers,* which is what everybody does anyway. Although at one time there was considerable attention given to nondigital computers, called *analog* machines (including slide rules), today just about all computing activity is done on digital machines. (The reason why the word digital is used will be seen in Section 4.1.)

Digital computers come in many sizes and shapes, but there are two general types you are most likely to encounter: microcomputers and time-sharing computers.

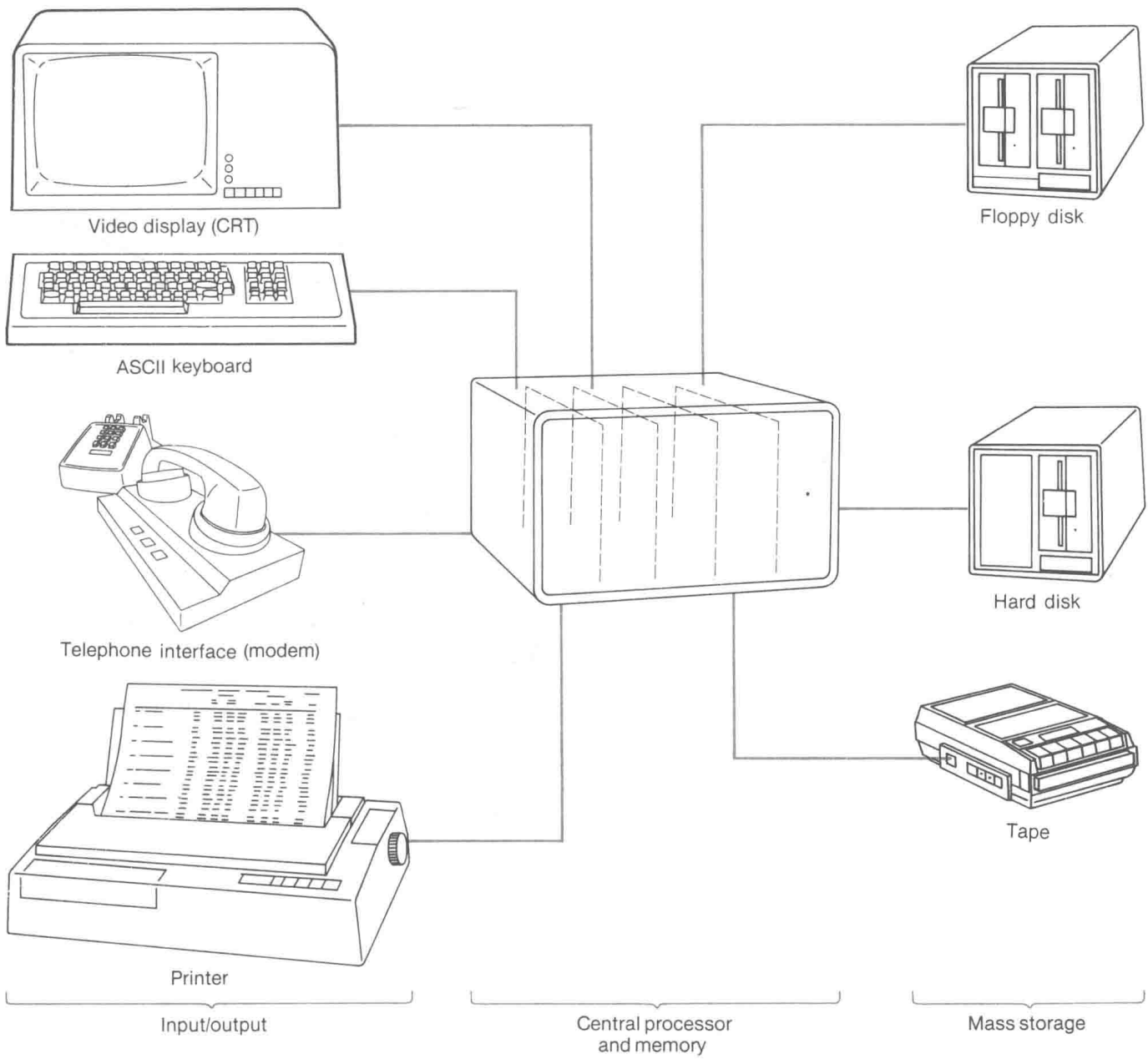



**FIGURE 1.1** Example of a microcomputer system

## Microcomputer Systems

Microcomputers (also known as personal computers) first appeared in the early 1970s. Today they are used by literally millions of people. A better term for micro (or personal) computer is *microcomputer system.* It is called a system because it consists of several parts, or *components,* that work together. As Figure 1.1 shows, there can be quite a number of these components, but they can all be grouped into three categories: *input/output* components, *central-processor/ memory* components, and *mass storage* components.

The input/output (also called I/O) components allow the user to communicate with the machine. For input, you usually use a keyboard to "talk" to the computer, typing in the programs (sets of instructions) that tell the computer what you want it to do. In this book, you will learn how to express such instructions as statements in a language called BASIC (*Beginner's All-purpose Symbolic Instruction Code*). BASIC instructions are stored in the computer's memory, along with any *data* (numbers or alphabetical symbols) that the program is to work on. The central processing unit (CPU) of the machine then manipulates this data* according to the program's instructions. The results of this processing are displayed on an output device—usually a video display or a printer.

Section 1.3 will show photographs of several microcomputer systems. They can all be called *microcomputer* systems, since the central processing unit of each uses a *microprocessor chip.* This is a thin slice of silicon on which thousands of circuits are engraved, placing all the power of general-purpose computing within low-cost desk-top units. You'll also notice that the microcomputer systems shown in these photos have *floppy disks* for mass storage. Floppy disks are circular pieces of magnetically coated plastic on which both programs and data can be stored for future use. We will have more to say about disks in Section 2.5 and Chapter 11.

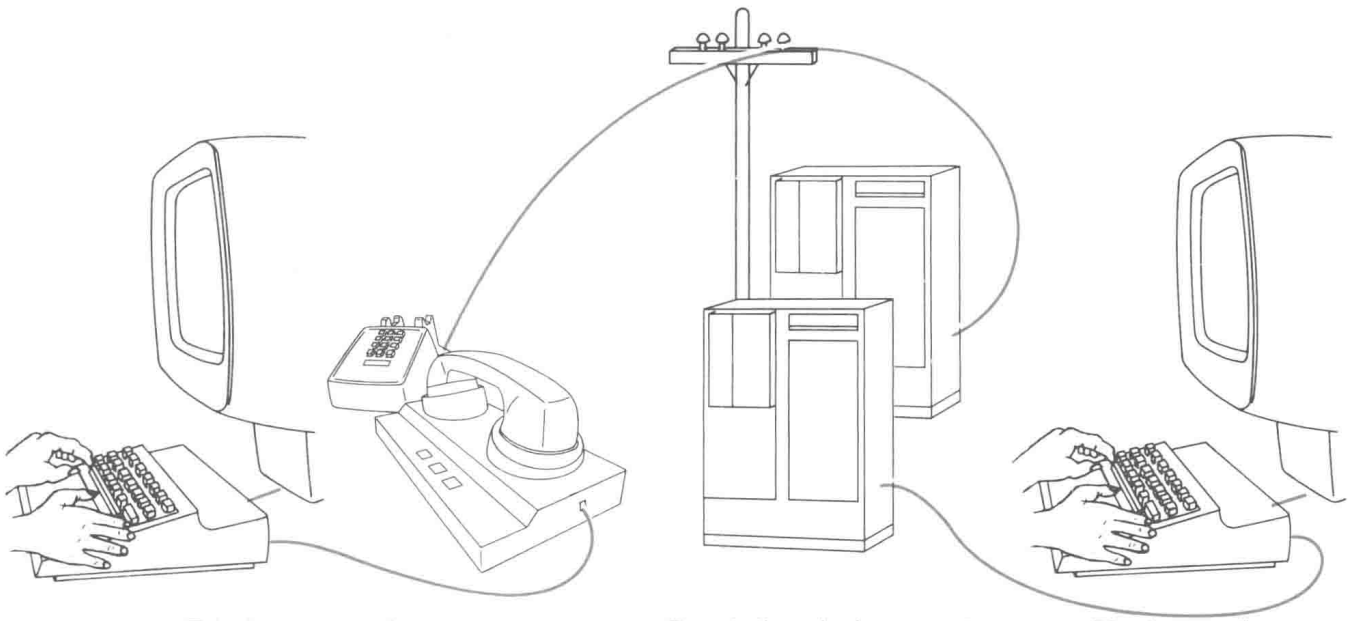
## Time-Sharing Computer Systems

The second type of computer that you may use is a machine large enough to require a room all to itself. The machine may be close at hand, or it may be miles away. Such machines can be controlled by several users, each one working at a separate terminal. (The word *terminal* refers to a combination of a keyboard input device and a video or printer-like output device.) However, the terminals are hardly ever in the same room as the computer. This is no problem, since two-way communication with a computer can take place over long cables or telephone lines. The setup looks something like that shown on page 7.

With such an arrangement, many people can be given the illusion that they are simultaneously communicating with the central computer. The process that makes this possible is called *time sharing.*

How does time sharing work? The computer carries out its operations at such tremendous speed that it can give you enough computing time to keep you busy in a fraction of a minute. The rest of that minute can go to the other users (*user* means anyone working at an on-line terminal). The situation is something like that of a stockbroker taking telephone orders from several customers at the same time. If the stockbroker could switch back and forth from one telephone to

**Data* is the plural of *datum.* However, through usage, the word data has become accepted as being both singular *and* plural: *this data* and *these data* can both be used. This is one of the many changes that computers have brought to the English language.




another fast enough, each customer would think that he or she was getting the stockbroker's full attention. The computer *is* that fast; you think it's talking only to you.

To make things clearer, let's consider the two types of computer systems separately. You need read only the section that deals with the type of computer you have (Section 1.3 for microcomputers, Section 1.4 for time-sharing computers).

## 1.3 Using a Microcomputer: Procedures for the TRS-80, Apple, and IBM Micros

Radio Shack
APPLE
IBM

Getting a microcomputer ready for BASIC programming is usually a simple procedure. It consists of two steps.

1. Turn on the power to all components in the system.
2. Load a special program called the BASIC interpreter.

Step 1 can be made even easier by having all the components plugged into a switched multiple-outlet box (110-volt ac). Then, if you leave the power switches for all the components in their ON position, you can turn the entire system on or off with the switch on the outlet box. (Some manufacturers may advise against this. If so, follow their recommendations.)

The exact procedure for step 2 depends on whether you are using a computer system with floppy disks or not. If you are, then a BASIC *system* disk—a disk that contains the BASIC interpreter—must be inserted into the computer's primary disk drive. On some systems, if you insert the system disk and close the drive door before turning the power on, step 2 will take place automatically after step 1. If you *don't* use disks, BASIC is usually stored in a part of the computer called ROM (*Read Only Memory*). In this case, it's not necessary to take any special action in step 2. In all cases, be sure to read the manual for your specific system before you do any of this.