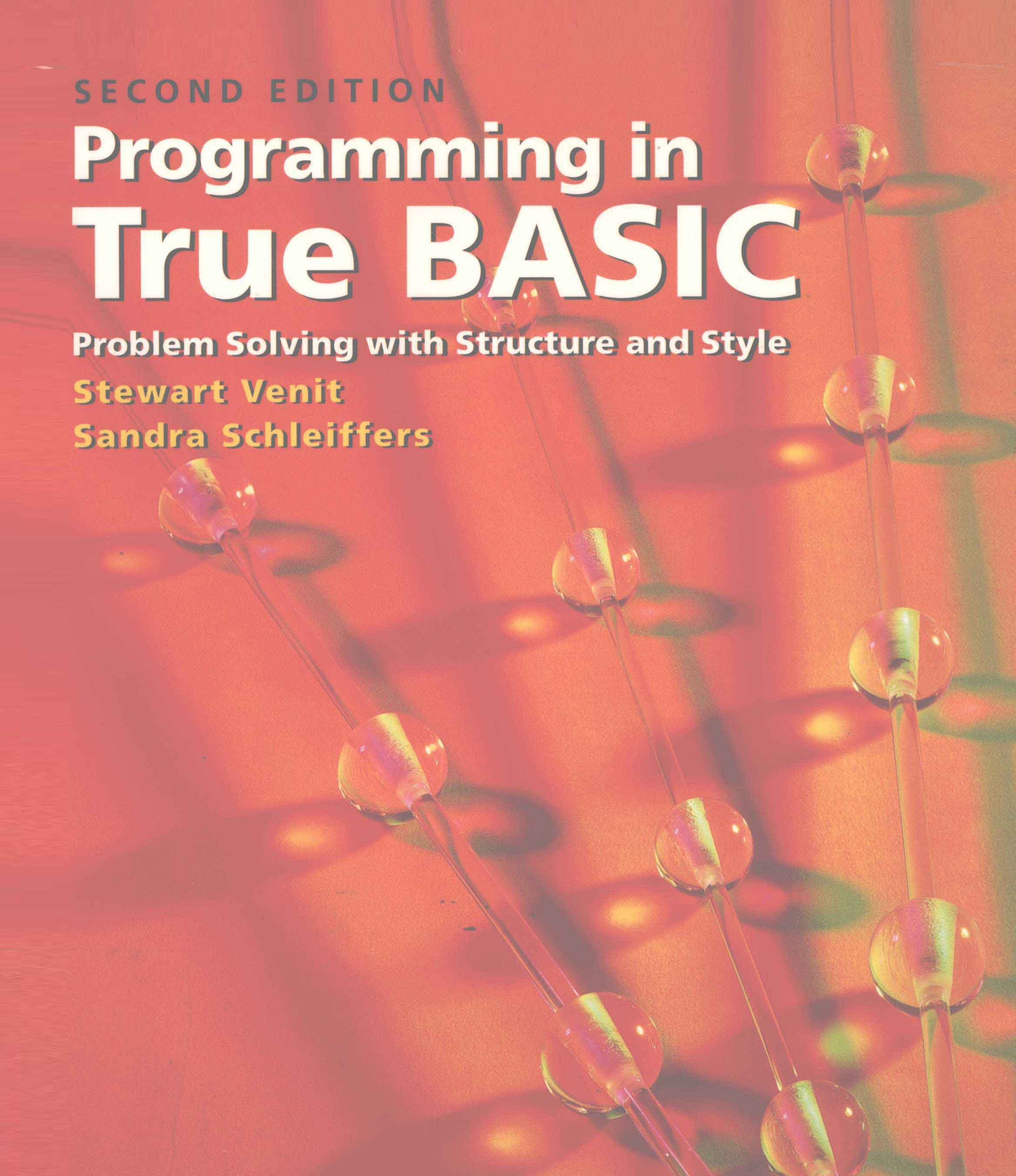
SECOND EDITION

Programming in True BASIC

Problem Solving with Structure and Style

Stewart Venit

Sandra Schleiffers



PROGRAMMING IN TRUE BASIC

PROBLEM SOLVING WITH STRUCTURE AND STYLE

SECOND EDITION

STEWART M. VENIT
California State University, Los Angeles

and

SANDRA M. SCHLEIFFERS
Colorado State University, Fort Collins

PWS Publishing
An Imprint of Brooks/Cole Publishing Company
I(T)P® An International Thomson Publishing Company

Pacific Grove • Albany • Belmont • Bonn • Boston • Cincinnati • Detroit • Johannesburg • London • Madrid
Melbourne • Mexico City • New York • Paris • Singapore • Tokyo • Toronto • Washington

Sponsoring Editor: *Mike Sugarman*
Editorial Assistant: *Kathryn Schooling*
Project Development Editor: *Suzanne Jeans*
Marketing: *Nathan Wilbur*
Communications: *Maraget Parks*
Production Coordinator: *Kelsey McGee*
Manuscript Editor: *Julie Hollist*
Interior Illustration: *Rolin Graphics and Heather Theurer*
Production/Design/Typesetting: *WestWords, Inc.*
Cover Design: *Vernon T. Boes*
Cover Photo: *Ed Young Photography*
Cover Printing/Printing and Binding: *WebCom Limited*

For more information, contact PWS Publishing at Brooks/Cole Publishing Company:

BROOKS/COLE PUBLISHING COMPANY
511 Forest Lodge Road
Pacific Grove, CA 93950
USA

International Thompson Publishing Europe
Berkshire House 168-173
High Holborn
London WC1V 7AA
England

Nelson ITP
102 Dodds Street
South Melbourne, 3205
Victoria, Australia

Nelson Canada
1120 Birchmount Road
Scarborough, Ontario
Canada M1K 5G4

International Thomson Editores
Seneca 53
Col. Polanco
11560 México, D. F., México

International Thomson Publishing GmbH
Königswinterer Strasse 418
53227 Bonn
Germany

International Thomson Publishing Aisa
60 Albert Street
#15-01 Albert Complex
Singapore 189969

International Thomson Publishing Japan
Hirakawacho Kyowa Building, 3F
2-2-1 Hirakawacho
Chiyoda-ku, Tokyo 102
Japan

Printed in Canada

10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Venit, Stewart.
Programming in True BASIC : problem solving with structure and style / Stewart M. Venit and Sandra M. Schleiffers. — [2nd ed.]
p. cm.
Includes bibliographical references and index.
ISBN 0-534-95351-4
1. True BASIC (Computer program language) I. Schleiffers, Sandra M. II. Title.
QA76.73.T65V46 1998 98-34943
005.26′2—dc21 CIP

THIS BOOK IS PRINTED ON ACID-FREE RECYCLED PAPER

PREFACE

PURPOSE

This text has two fundamental objectives: to teach programming concepts in general and the elements of the True BASIC language in particular. We have placed great emphasis on structured programming principles: problem solving, top-down modular program design, structured coding, and programming style. Students using this text will learn to write readable, reliable, and well-documented programs, and be able to move on to more advanced topics and/or other programming languages without having to unlearn bad habits.

PREREQUISITES

The prerequisites for this text are minimal. No prior programming experience is required, and a single year of high school algebra should provide adequate preparation. A few advanced concepts are presented in the later chapters for better prepared students or for longer courses.

TRUE BASIC

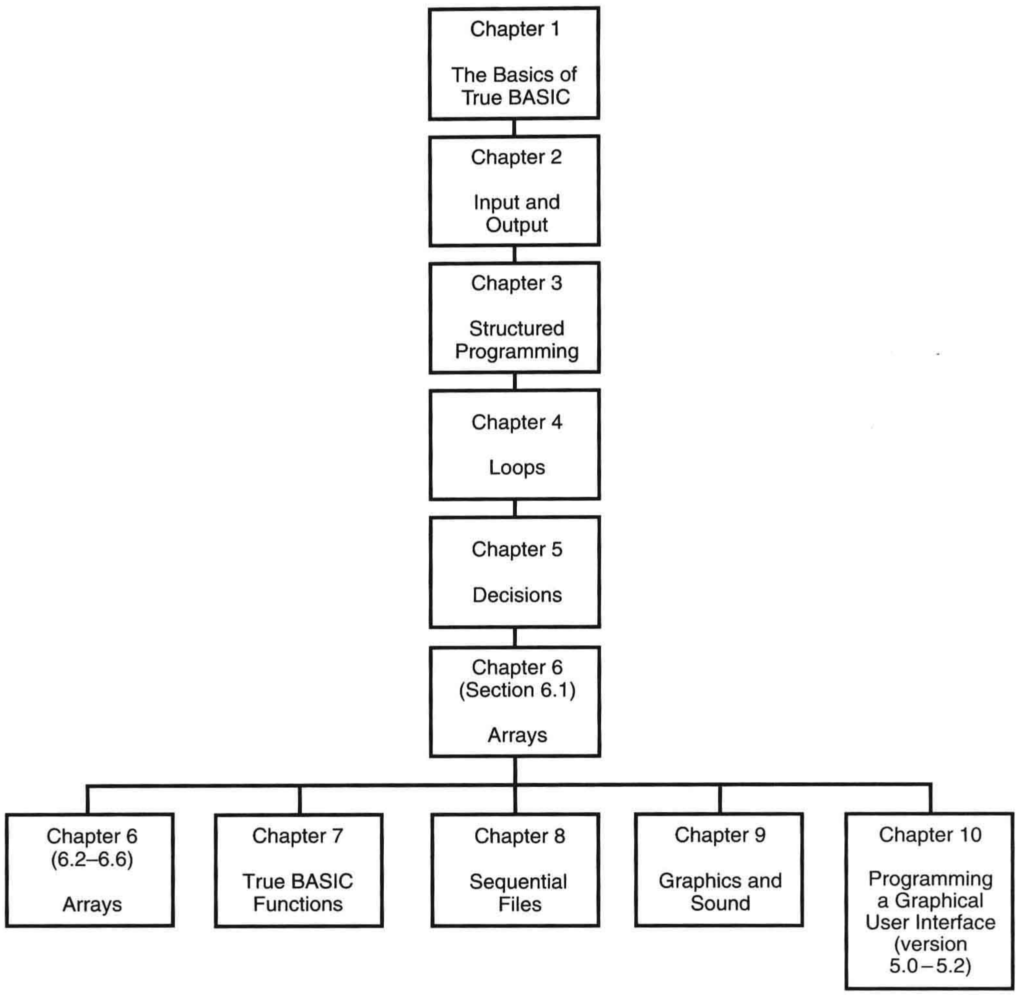
True BASIC, a version of BASIC published by True Basic Incorporated, is one of the ideal first programming languages. True to its ancestry, True BASIC is relatively easy to learn, and it has two key advantages over earlier versions of BASIC: the True BASIC dialect allows for truly structured programming and the True BASIC environment eliminates much of the frustration inherent in coding programs.

True BASIC is also a very powerful language, suitable for producing relatively large and efficient programs. This text does not attempt to cover every facet of the language, but instead provides students with a firm foundation that will enable them to explore it further on their own if they wish.

Several versions of True BASIC are currently in use (Macintosh versions, DOS versions, and Windows versions). All programs written in this text should run in the various versions of True BASIC with the exception of the programs in Chapter 10 (Programming a Graphical User Interface). Chapter 10 was written specifically for the latest Windows 95 and Macintosh versions (5.0–5.2) of True BASIC. The syntactical differences among the various versions are slight. Therefore, with the exception of the Chapter 10 programs, all programs will run under the latter versions of the language. However, some programs may need slight modification when using the Macintosh versions and when using PC versions of True BASIC earlier than 5.0. Specifically, programs will have to be modified slightly in Chapter 8 (Sequential Files), and in Chapter 9 (Graphics and Sound). Notes that describe these modifications are provided where applicable.

FEATURES OF THE TEXT

1. The text introduces the True BASIC language quickly—the first program appears at the very beginning of the first chapter.
2. New statements are introduced through short programs or program segments, avoiding *at this point* elaborate program design and long explanations. Display boxes that show the form, action, and examples of newly introduced statements are provided for easy reference.
3. Structured programming principles are emphasized throughout the text. Program design, style, and maintenance are introduced in Chapter 1 and are reinforced in every subsequent chapter.
4. All complete programs are written with good programming style. *Style Pointers* are presented throughout the text.
5. All chapters contain *applications*—longer programs that illustrate the complete program development process of analysis, design, coding, and testing. (Most applications appear in the *Focus on Problem Solving* sections.) Pseudocode and hierarchy charts are the primary program design tools.
6. The text contains a few relatively long illustrative programs (several hundred lines each). These help students to see the *need* for program design, structured coding, and good programming style.
7. Short *self-test* exercises (with answers in Appendix C) appear at the end of each section. More detailed *Review Exercise* sets which contain short answer, debugging, and skill builder exercises (with answers to the odd-numbered ones in Appendix C) are supplied at the end of each chapter. *Programming Problems,* which offer a wide range of difficulty, and *Laboratory Projects,* which consist of debugging and maintenance problems, also appear at the end of every chapter.
8. A disk that contains the program files for the Laboratory Projects is packaged with the text. Chapter 10 lab project programs will run only in True BASIC 5.2 SILVER or GOLD Versions, unless users have access to the True Control or True Dial libraries from a previous 5.* edition. Macintosh adopters of the text also should be able to use the Laboratory Projects disk that accompanies the text. Some minor modification may be needed as discussed previously.
9. Each chapter concludes with a useful summary that includes (among other things) a list of the key terms, new True BASIC statements, and style pointers introduced in that chapter. The key terms are boldfaced within the text.
10. *Programming Pointers,* which discuss subtleties in the language and common programming errors, appear throughout the text.
11. The last four chapters of the text are relatively short and, except in one case, independent of one another (see the Dependency flowchart on the following page). This makes it easy for an instructor to select extra topics to fill out a course once the core material has been covered.
12. The text contains a chapter on graphics and sound. These topics are not only important in their own right, but also arouse the interest of almost all students.



13. The text includes several appendices (chapter appendices and text appendices). Chapter appendices discuss relevant aspects of Windows 95/98 and DOS and the command environments inherent in the different versions of True BASIC, coding programs to output results directly to the printer, an introduction to library usage, and utilizing MAT statements with arrays, additional numeric functions and programming dialog boxes. Text appendices include using the Internet, the entire 256-character IBM-extended ASCII code schematic and answers to selected practice exercises.

SUPPLEMENTARY MATERIALS

An *Instructor's Manual* containing teaching suggestions, additional exercises (with answers), answers to the even-numbered Chapter Review exercises, and solutions to selected Programming Problems is available to those who adopt the text.

ACKNOWLEDGMENTS

We would like to thank the many people who helped bring this project to fruition. The following reviewers greatly improved this text through their thoughtful comments and useful suggestions:

William Beaver	City College of San Francisco
Curtis Bring	Moorhead State University
John Castek	University of Wisconsin at La Crosse
W. O. Crain	Seattle Central Community College
Louise Darcey	Texas A & M University
James Ingraham	Phoenix College
Nancy Harrington	Johnson County Community College
Robert Harrington	Utah Valley Community College (retired)
Taylor Hollist	State University of New York at Oneonta
Dana Johnson	North Dakota State University
Firooz Khosraviyani	University of Texas-Permian Basin
Harold Kollmeier	Glassboro State University
Norman Lindquist	Western Washington University
Marilyn Meyer	Fresno City College
George Miller	North Seattle Community College
Lewis Miller	Cañada College
Chris Nikolopoulos	Bradley University
J. Douglas Robertson	Bentley College
Pasha Rostov	California Polytechnic State University, San Luis Obispo
R. Waldo Roth	Taylor University
Ingrid Russell	University of Hartford
Jean Simutis	California State University at Hayward
Alfred Weaver	University of Virginia
Jack Weir	Rock Valley Community College
Michael Willis	Montgomery College

We are indebted to Richard Mixter who suggested and helped shape this project, and to Jennifer Maughan and Kelsey McGee who skillfully guided it through production.

I would like to thank my wife, Corinne, and my daughter, Tamara, for their continued encouragement and support.

S.V.

I would like to thank Stewart for his flexibility with this project and for his support. For their continued support and encouragement, I would also like to thank the members of my department and the people who have made my life very special.

S.S.

INTRODUCTION

Sixty years ago, electronic computers did not exist. Just 30 years ago, there were fewer than 25,000 of them. Today, millions of computers are in use around the world.

In their early days, computers were used almost exclusively by large businesses, engineering firms, universities, and government institutions. At that time, they were expensive, somewhat temperamental machines, kept isolated in their own air conditioned rooms, and operated by specially trained personnel.

Today, computers are everywhere. You can find them in homes, schools, and offices; in supermarkets and fast food restaurants; and on airliners and the space shuttle. They are used by the young and the old, by filmmakers and farmers, and by bankers and baseball managers. We use computers in almost limitless ways: for entertainment, education, money management, product design and manufacture, and to run our businesses and institutions. There are now few human endeavors that are not somehow touched by the use of the electronic computer.

In this introduction, we will discuss various aspects of the computer and how it is used. Although you can certainly learn to program in True BASIC without this background information, it may help to clarify certain points and should increase your computer literacy.

COMPUTERS

Everyone who uses a computer on a daily basis becomes accustomed to dealing with special computer-related terminology. Yet to a beginner, many terms can be confusing and even intimidating—for example, microdisks and hard disks, kilobytes and megabytes, mice and monitors, and much more. In this section we will try to take some of the mystery out of computer terminology.

What Is a Computer?

As with any evolving technology, precisely defining the term *computer* is not easy. *Computers* can have many forms and their capabilities are constantly expanding. Yet, all computers do the same basic things. Every **computer** can input, store, manipulate, and output vast quantities of data at very high speeds. Moreover, all computers are *programmable*—that is, they can follow a list of instructions (a **program**) and act upon intermediate results without human intervention.

A **personal computer (PC)**—also called a **microcomputer**—is a relatively small type of computer, usually intended for use by one person at a time. (Larger computers—known as *minicomputers, mainframes,* and *supercomputers,* in order of increasing size and power—can be simultaneously shared by many users, who are electronically connected to the larger computer by cables or telephone lines.) All personal computers are small enough to fit on a desktop. Portable PCs (**Laptops**) are even smaller, usually no larger than a loose-leaf binder. A typical PC is shown in Figure 1.

The Components of a Computer

As its definition implies, a computer must have the ability to input, store, manipulate, and output data. These functions are carried out by the following five main components of a computer system.

1. Central Processing Unit (CPU)
2. Internal Memory
3. Secondary Storage Devices
4. Input Devices
5. Output Devices

In a personal computer, the first two components (and usually the third) are housed in the **system cabinet** (see Figure 1). Input devices like the keyboard and mouse and output devices like the monitor are housed in their own enclosures and are connected to the system unit by cables. Figure 2 illustrates the relationship among these components; the arrows show the direction of data and information flow.

All of the physical components that make up the computer system are known as **hardware.** Devices that are used by a computer but located outside the system unit are sometimes called **peripherals.** The term **software** refers to the programs used by a computer system.

FIGURE 1 A Typical Microcomputer System

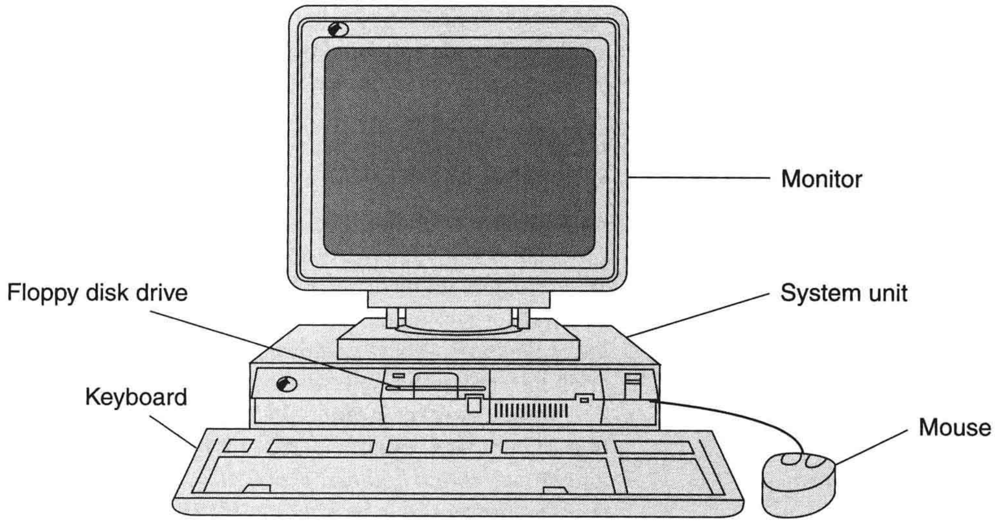
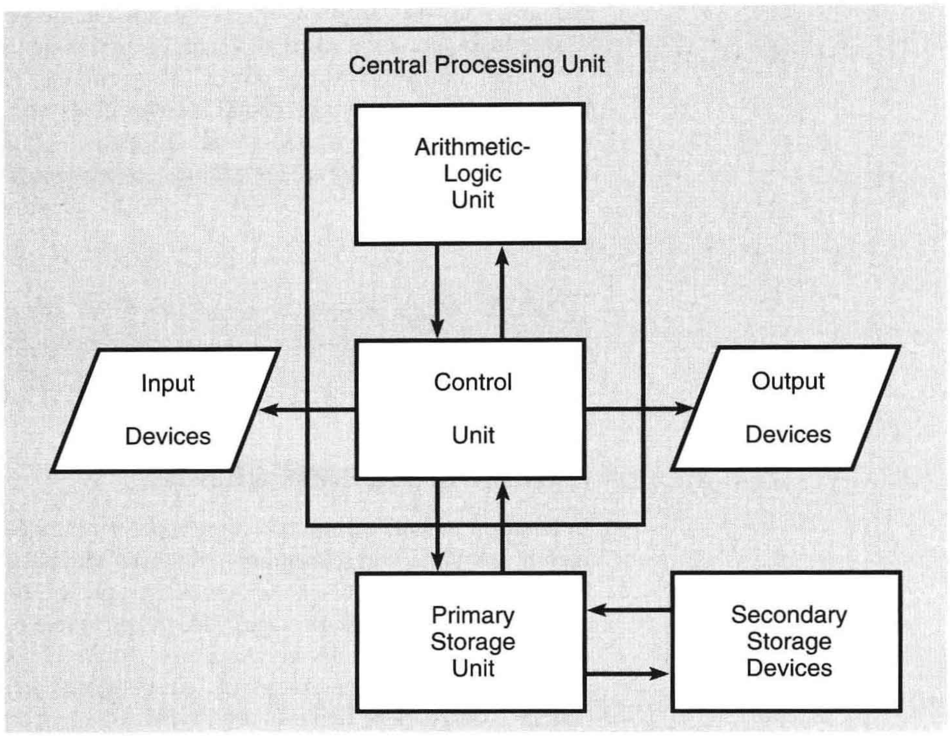


FIGURE 2 The Components of a Computer System



THE CENTRAL PROCESSING UNIT

The **central processing unit** (or CPU), also called the microprocessor, is the brain of the computer. It consists of two major components:

1. A **control unit** that processes the instructions and directs the flow of information throughout the computer system.
2. An **arithmetic-logic unit** that performs the necessary arithmetic (addition, subtraction, and so on) and logical operations (like comparing two numbers).

THE INTERNAL STORAGE UNIT

The **internal storage unit** stores the information to be processed by the CPU. This information consists of the program being executed, as well as the **data**—numbers, words, and other symbols—manipulated by it. The internal storage unit is known by several other names: **internal memory, main memory, and RAM** (for Random-Access Memory).

In modern computers, the internal memory consists of a set of microchips connected by very fine wires. The CPU interacts with its internal storage unit at a very high rate of speed; it takes only a fraction of a millionth of a second for it to access memory. Unfortunately, any information stored there is lost when the computer's power is turned off.

Internal memory is partitioned into storage locations called **memory cells.** Each memory cell is capable of storing one **character** (letter, digit, comma, asterisk, and so on), also known as one **byte** of information. On most computers, one byte is made up of a combination of eight zeros and ones, each of which is called a bit (for *binary* digit).

One indication of the "power" of a computer is the number of storage locations it contains, the size of its internal memory. This number is usually expressed in *kilobytes* (KB) or *megabytes* (MB); one kilobyte is equal to 1,024 ($=2^{10}$) bytes and one megabyte is 1,024 kilobytes. For example, one megabyte of RAM contains 1,048,576 storage locations (1 MB = 1,024 KB = 1,024 × 1,024 bytes) in its internal memory.

NOTE We sometimes speak of the CPU and internal memory as being *the* computer. The secondary storage and input/output devices are referred to as **peripherals.** All these components taken together make up a **computer system.**

EXTERNAL STORAGE DEVICES

In addition to internal memory, a computer needs **external storage,** another form of memory, which stores programs and data semipermanently. Unlike the contents of RAM, which are lost when the computer is turned off, information remains in external storage until you decide to erase it. However, to make use of any data or software stored on an external storage device, you must first *load* (or copy) the information into RAM.

The primary type of external storage device on a PC is the **disk drive.** Most PCs contain a **hard disk drive** housed within the system unit, and at least one **external** (microdisk), also housed within the system unit but accessible from the outside (refer to Figure 1). The **hard disk** is a magnetic platter that is sealed within the drive. **Microdisks** (or **diskettes**), on the other hand, are stored away from the computer and are inserted into a drive when needed. Hard disks store massive amounts of data and information with storage capacities ranging from 800 MB to 4.5 GB or more. Microdisks, however, are portable disks 3 1/2 inches in diameter with a storage capacity of 1.44 MB.

Most PCs are also equipped with additional types of external storage devices such as a **CD-ROM drive** and/or a **Zip drive.** CD-ROM drives use disks that are similar to audio compact disks (hence the "CD" in the name). Like microdisks, these disks are removable and portable, but unlike microdisks, they hold huge amounts of information (about 600 MB). Because CD-ROM drives are Read-Only Memory (**ROM**) devices, you cannot write (save) data on these disks. They are used primarily to store reference material (such as an encyclopedia) and sophisticated computer games. Zip drives use disks call Zip disks that are similar in size to microdisks. These disks are portable and have a storage capacity of 100 MB. Zip disks are an excellent storage medium for transporting large amounts of data and programs to and from other PCs that have access to a Zip drive.

INPUT DEVICES

An **input device** enables us to communicate with the computer. It accepts information in a form that is understandable to people (such as typed or spoken words), transforms it into a machine-readable form, and transmits it to the computer.

The typewriter-like **keyboard** is by far the most common input device in use today. To enter information into the computer, you simply type it at the keyboard in much the same way you would if you were using an ordinary typewriter. (The characters you type at the keyboard simultaneously appear on the computer's display screen.) Computer keyboards contain a few more keys than a typewriter; the extra ones facilitate communication with the machine.

Another type of input device is used to "point" at items on the display screen and thus initiate an action. The most popular of these devices is the **mouse,** a handheld object containing one, two, or three buttons that moves a pointer on the screen when you roll the mouse around on the desk top. Pointing devices such as the mouse can speed up some input operations, but they lack the versatility of a keyboard.

OUTPUT DEVICES

Whereas input devices allow us to communicate with the computer, **output devices** make it possible for the computer to "talk" to us. The most common output devices are *monitors* and *printers.*

The computer's primary output device is the monitor, a high resolution television-like screen enclosed in a case and controlled by circuitry—the *video adapter* card—within the computer. The output from the monitor is called **soft copy.**

To make a permanent copy of the computer's output (**hard copy**) on paper, we use a printer. These come in several varieties including ink jet, bubble jet, and laser jet printers with black and white and color capabilities. Among these, the laser jet printer is the premier printer in use today. It combines the print quality of type-written text with the speed of Superman.

PROGRAMMING LANGUAGES AND SOFTWARE

The computer's hardware (its CPU, memory, and peripherals) is useless without instructions that tell it what to do. The first computers were given these instructions (the *program*) by actually rewiring some of their circuits. Needless to say, this was a painstaking task. Then, in the late 1940s, the brilliant mathematician John von Neumann came up with the idea of storing the instructions in the computer's internal memory, and this is the way it has been done ever since.

Types of Programming Languages

A **programming language** is a set of symbols and the rules governing their use, employed in constructing programs. Programming languages are of four fundamental types:

1. Machine languages
2. Assembly languages
3. High-level languages
4. Event-driven languages

A **machine language** program consists of a sequence of zeros and ones. The numbers specified and the order in which they appear tell the computer what to do. Machine language, which differs considerably from one type of computer to another, is the only language the machine can understand directly. However, as you might imagine, it is very difficult for humans to read or write. For this reason, **programmers** write their programs in either *assembly* or *high-level* languages.

Assembly language is a symbolic representation of machine language. There is usually a one-to-one correspondence between the two; each assembly language instruction translates into one machine language instruction. However, assembly language uses easily recognizable codes, which make it a lot easier for people to understand. For example, the following instruction adds two numbers on a certain minicomputer:

Machine Language Instruction

```
0110110111110111 0000000000000010 0000000000000010
```

Assembly Language Equivalent

```
ADD   A, B
```

Before a computer can carry out an assembly language program, it must be translated (by the computer) into machine language. This is done by a special program called an **assembler**.

High-level languages (procedural languages) usually contain English words and phrases; their symbols and structure are far removed from those of machine language. High-level languages have several advantages over machine or assembly languages. They are easier to learn and use, and the resultant programs are easier to read and modify. A single instruction in a high-level language usually translates into many instructions in machine language. Moreover, a given high-level language does not differ much from computer to computer; a program written on one machine can usually be modified relatively easily for use on another. On the negative side, programs written in a high-level language are usually less efficient than their assembly language counterparts. High-level languages, like assembly languages, must be translated into machine language before their instructions can be carried out. This is done by programs known as *interpreters* and *compilers.*

The first high-level language, FORTRAN (which stands for FORmula TRANslation), was developed in the mid-1950s for engineering and scientific applications. Since then, there has been a flood of high-level languages. A few of these are:

Ada (named after the Countess of Lovelace)—mostly for military applications

BASIC (Beginner's All-purpose Symbolic Instruction Code)—the language you will learn in this text

C—for efficient programming of many different types of applications

COBOL (COmmon Business Oriented Language)—for business related programming

Pascal (named after Blaise Pascal, a 17th-century philosopher and mathematician)—for teaching programming concepts and microcomputer applications

An **event-driven language** is a high-level language that allows the programmer, among other things, to create a **graphical user interface (GUI)** by writing code to add forms (windows/dialog boxes), command buttons, text boxes, etc. to a program. In this way, the executing program's environment appears similar to the Windows 3.1 or Windows 95 software environments. True BASIC (versions 5.0–5.2) and Visual BASIC are both examples of event-driven languages.

True BASIC

True BASIC is a particular version of the BASIC programming language. BASIC was created by John Kemeny and Thomas Kurtz at Dartmouth College in the mid-1960s for the specific purpose of providing students with a powerful yet easy-to-learn means of writing programs. Since the advent of microcomputers, which are often sold with BASIC included as part of the package, it has become the most popular programming language in use today.

As its popularity increased, so did the number of BASIC **dialects,** slightly different versions of the language. The most popular version of BASIC in use today is published by Microsoft Corporation and is available for most microcomputers in several different versions; it may be called Microsoft BASIC, QBASIC, or Visual-BASIC. True BASIC, also created by John Kemeny and Thomas Kurtz, is an implementation of BASIC that closely conforms to national standards, and is very well suited for both beginning and experienced programmers. True BASIC is simple enough to be easily learned, yet powerful enough to be used in practical applications.

BASIC is usually translated into machine language by means of a built-in program, either an *interpreter* or a *compiler.* An **interpreter** translates each instruction of your program and then the translated instruction is executed by the computer. A **compiler,** on the other hand, translates the entire program before the computer executes any of the instructions. Interpreters allow you to execute partial programs, thus enabling you to see how your program is working before you complete it. However, interpreted programs execute considerably more slowly than their compiled counterparts. Today the current versions of BASIC,7 like True BASIC and Visual BASIC, use compilers to translate the programs you write.

Computer Software

Programming languages are used to create **software,** the programs used by a computer system. Software can be divided into two main categories:

1. Systems software
2. Applications software

Systems software, which is written by the computer manufacturer or specifically for that manufacturer, coordinates the actions of the components of the computer system. It includes compilers, interpreters, text editors, and the computer's **operating system** (Windows 95/98 on IBM-compatible microcomputers), the master control program for the system. When you load a program into memory or use your printer, you do so with the aid of the operating system. Generally speaking, systems software supplies the tools that enable us to write and to run our applications programs.

Applications software refers to the programs that solve the computer user's problems. It includes, as just a few examples, word processors, spreadsheets, teaching programs, and games. (Games solve the problem of what to do with your spare time.) Applications software might be written by its user, but it is much more likely to be created by a software publishing house.

Writing Your Own Programs

In this text you will learn to write True BASIC programs: your own applications software. Here are some suggestions to help the learning process.

1. Read the text carefully. The computer is a literal-minded machine; it requires that you follow exactly the rules governing the use of True BASIC. For example, a misspelled word may confuse the computer even though it's perfectly clear (to a human being) what you *meant* to say.
2. Take things one step at a time. After you've read a section, work its self-test; after you've completed a chapter, read the summary and work the review exercises. This will help reinforce the material you've just learned.
3. Practice, practice, practice. To learn programming, you must program. It is important to practice new concepts by doing some of the programming problems at the end of each chapter.
4. Above all, be patient. Everyone makes mistakes in writing programs, so don't let your mistakes get you down.

If you follow these suggestions, you should find programming to be fun and a rewarding experience. Good luck!

SUMMARY

Key Terms*

computer	central processing unit (CPU)
microcomputer	arithmetic-logic unit (ALU)
hardware	data
control unit	main memory
internal storage unit	memory cells
internal memory	byte
RAM	peripherals
character	external storage
bit	disk drive
computer system	CD-ROM disk
diskette (microdisk)	Zip disk
hard disk	input device
keyboard	mouse
output device	monitor
hard copy	printer
soft copy	machine language
programming language	assembly language
programmer	high-level language
assembler	event-drive language
dialects (of BASIC)	graphical user interface (GUI)
compiler	interpreter
systems software	software
application software	operating system
program	personal computer

*Key terms are listed in order of introduction.

Components of a Computer

Central processing unit
Internal storage unit: internal memory (main memory or RAM)
External storage devices (disk drives, types of storage media)
Input devices (such as a keyboard or mouse)
Output devices (such as a monitor or printer)

Types of Programming Languages

Machine languages
Assembly languages
High-level languages
Event-driven languages

REVIEW EXERCISES

1. In your own words, what is a computer?
2. According to your definition which of the following qualify as computers?
   - **a.** Simple calculators
   - **b.** Programmable calculators
   - **c.** Microwave ovens
   - **d.** Digital watches
3. According to the definition given in the text, which of the items in exercise 2 are computers?
4. Name the five major components of a computer system?
5. How many characters can be stored by a computer that has:
   - **a.** 64 KB of RAM
   - **b.** 2 MB of RAM
6. Why are *both* internal and secondary storage necessary in a computer system?
7. Name the four general categories of computer languages.
8. What is the difference between an interpreter and a compiler?
9. What is the difference between hardware and software?
10. What is the difference between application software and system software?

CONTENTS IN BRIEF

Introduction Computers I-1
Chapter 1 The Basics of True BASIC 1
Chapter 2 Input and Output 67
Chapter 3 Structured Programming 109
Chapter 4 Loops 161
Chapter 5 Decisions 211
Chapter 6 Arrays 261
Chapter 7 True BASIC Functions 315
Chapter 8 Sequential Files 363
Chapter 9 Graphics and Sound 401
Chapter 10 Programming a Graphical User Interface 445
Appendix A Using the Internet A-3
Appendix B The ASCII Code A-9
Appendix C Answers to Selected Exercises A-11