FORTRAN with Problem Solving: A Structured Approach

Robert J. Bent
George C. Sethares

FORTRAN with Problem Solving: A Structured Approach

Robert J. Bent
George C. Sethares
Bridgewater State College

Brooks/Cole Publishing Company
Monterey, California

Dedicated to Costas H. Sethares
and to the memory of
Mary C. Sethares
Catherine A. Bent
John J. Bent

Brooks/Cole Publishing Company
A Division of Wadsworth, Inc.

© 1981 by Wadsworth, Inc., Belmont, California 94002.
All rights reserved.
No part of this book may be reproduced, stored in a retrieval system,
or transcribed, in any form or by any means—electronic,
mechanical, photocopying, recording, or otherwise—
without the prior written permission of the publisher,
Brooks/Cole Publishing Company, Monterey, California 93940,
a division of Wadsworth, Inc.

Printed in the United States of America

10 9 8 7 6 5 4 3 2

Library of Congress Cataloging in Publication Data

Bent, Robert J 1934–
FORTRAN with problem solving.

Includes index.
1. FORTRAN (Computer program language) 2. Problem solving. I. Sethares, George C., 1930– joint author. II. Title.
QA76.73.F25B46 001.64'24 80–28581
ISBN 0–8185–0436–6

Acquisition Editor: *James F. Leisy, Jr.*
Manuscript Editor: *Kirk M. Sargeant*
Production Editor: *Cece Munson*
Interior Design: *Katherine Minerva*
Cover Design and Photo: *Stan Rice*
Illustrations: *VMH Visual Communications*
Typesetting: *Graphic Typesetting Service, Los Angeles, California*

Preface

This book is intended for an audience with no prior programming experience. A mathematics background of elementary algebra is sufficient for most of the material. The many worked-out examples illustrate how the FORTRAN programming language can be used both in a data-processing and in a scientific environment. The examples and the numerous problem sets are drawn from a wide range of application areas, including business, economics, personal finance, the social and natural sciences, mathematics, and statistics.

We wrote this book with two principal goals in mind. First, we felt it important to present the elements of FORTRAN so that meaningful computer programs could be written at the earliest possible time. We adhere to the notion that one learns by doing. As a result, problem solving is emphasized from the very beginning, and the various aspects of the FORTRAN language are introduced only as needed. Our second goal was to write a book that would serve as a general introduction to computer programming, not just to a programming language. Simply describing a variety of computer applications and ways to go about writing FORTRAN programs for these applications does not constitute an introduction to programming. What is required is a consideration of the entire programming process. The approach we have taken toward this objective is to introduce programming principles only as they can be understood and appreciated in the context of the applications being considered. For example, a beginner can easily appreciate the need to identify what values must be input and what form the output should take. Hence, these steps are introduced early. On the other hand, the value of modularization—that is, breaking down a long and possibly complex task into more manageable subtasks—is not so easily grasped in the context of the straightforward programming problems first encountered. As natural as modularization may appear to an experienced programmer, a beginner must "see" its usefulness before being convinced of its value. Thus, although several of the early examples illustrate the method of problem segmentation in very simple programming situations, it is not discussed as a general programming principle until later in the book.

8880858

All FORTRAN programs appearing in this text conform to the American National Standards Institute (ANSI) document, ANSI X3.9-1978, commonly referred to as FORTRAN 77. We have taken considerable care to avoid those parts of FORTRAN 77 that are not widely implemented or that are not essential to writing "good" programs. We *have* used the IF–THEN and IF–THEN–ELSE statements. Although they are not implemented on older FORTRAN systems, they represent a major improvement in the FORTRAN language. Their use not only simplifies the coding process but can significantly improve the readability of FORTRAN programs. As shown in Chapter 6, any program containing IF–THEN and IF–THEN–ELSE statements is easily modified to run on FORTRAN systems that do not allow these statements.

A few remarks are appropriate concerning the organization of this text. The first two chapters are introductory. Chapter 1 contains a brief introduction to computer science and to some of the terminology associated with computers. The term *compiler* described in this chapter is used throughout the text; hence, an understanding of what a compiler does is essential. Other than this, Chapter 1 may be read at any time. Chapter 2 contains an introduction to problem solving. The term *algorithm* is defined, and the sense in which computer programs and algorithms are equivalent is explained. This chapter must not be skipped.

Chapters 3–7 contain a description of what may be called elementary FORTRAN programming. Included are descriptions of the assignment statement, as used to perform numerical calculations; the READ, WRITE, and PRINT statements, as used for elementary input and output operations; and the control statements IF, GO TO, and DO. FORMAT statements, which often cause considerable difficulty for beginners, are described in parts. The simplest forms of the FORMAT statement needed to produce meaningful output are taken up in Chapter 3. The simplest forms needed to transfer numerical input data to the computer are described in Chapter 5. In Chapter 8, we give a more detailed description of FORMAT statements used for input/output operations involving numerical data; in Chapter 12, we describe the form needed for processing nonnumerical character data; and in Chapter 16, the forms needed for logical, complex, and double precision data are described. In each case, the material is written so that it can easily be taken up earlier, should that be desired. Also included in these chapters on elementary FORTRAN are descriptions of the two modes of operation used in FORTRAN programming: the batch processing mode and the conversational, or time-sharing, mode. These two computing environments are described in Chapter 4 and essentially all of the material following this chapter is appropriate in both environments. Situations where different methods apply are clearly identified.

The intermediate and advanced topics covered in Chapters 8–16 need not be taken up in the order of their appearance. Indeed, much of this material is presented in such a way that it can be taken up much earlier. For instance: the intrinsic functions (Sections 10.1–10.2) can be taken up any time after the logical IF statement is introduced in Chapter 6; statement functions (Sections 10.3–10.4) can be taken up just after Chapter 8; the material in Sections 12.1–12.4 on processing character data can be understood any time after the DATA statement is introduced in Chapter 8; and data files (Sections 14.1–14.4) can be taken up any time after Chapter 8. Included in these chapters on intermediate and advanced programming is a separate chapter (Chapter 15) on sorting and searching. Although some of this material is difficult, the chapter should not be omitted; at the very least, a method should be learned for sorting large files that will not fit in the computer's memory at one time.

Section 6.9, which describes the method of top-down programming, must not be skipped, for in many respects it is the most important section in the book. It not only brings together many of the programming principles learned in Chapters 2–6, but also describes the single most useful approach to programming that is known. No introduction to programming can be considered complete without an understanding of and a facility with the method of top-down programming.

We wish to take this opportunity to acknowledge the helpful comments of our reviewers: Charles Downey of the University of Nebraska at Omaha, Kendall E. Nygard of North Dakota State University, and Charles Pfleeger of the University of Tennessee. We feel that their many thoughtful suggestions have led to a greatly improved text.

A very special thanks goes to Patricia Shea, our typist, proofreader, debugger, and consultant on matters of form. Many of her suggestions concerning the text format have been adopted, and her meticulous concern for detail was indispensable. Finally, we are happy to acknowledge the fine cooperation of the staff at Brooks/Cole Publishing Company.

Robert J. Bent
George C. Sethares

此为试读，需要完整PDF请访问：www.ertongbook.com

Contents

1 Computer Systems

An electronic computer system has the ability to store large quantities of data, to process these data at very fast rates, and to present the results of this processing in ways that are meaningful to the task at hand. Thus, if the task is to prepare a payroll, employee data will be stored in the computer, the computer will process these data to calculate relevant wage statistics, and the results will be presented in printed form, possibly including paychecks. This payroll example illustrates the three principal tasks involved in any computer application: data must be presented to the computer (**INPUT**), data must be processed (**PROCESS**), and results must be presented in a meaningful way (**OUTPUT**). (See Figure 1.1.)

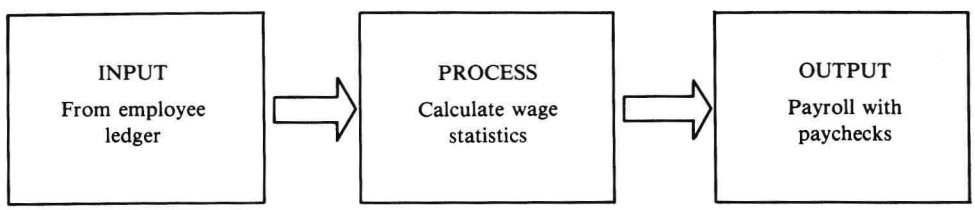


FIGURE 1.1. An INPUT–PROCESS–OUTPUT diagram.

The purpose of this chapter is not to convince you that a computer can "do" many things, nor even to indicate the computer applications you will be able to carry out after completing this text. Rather, the objectives of this chapter are to introduce you to the types of computer equipment you may encounter, to describe what a computer program is, and to introduce certain terminology that is helpful when talking about computers.

1.1 Computer Hardware

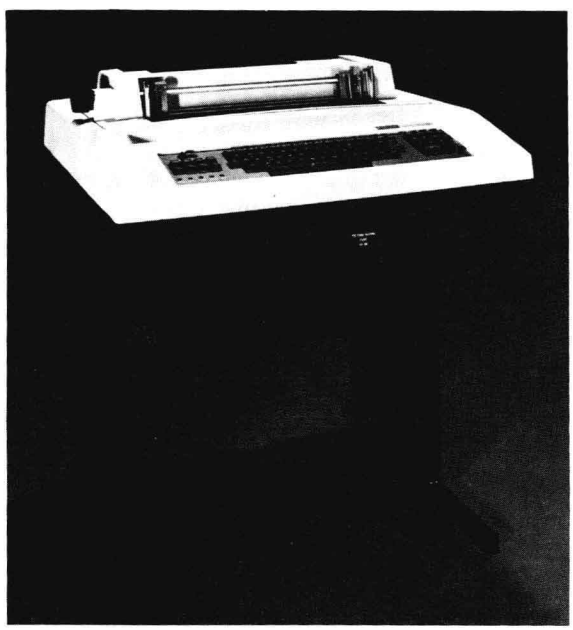
Central to every computer system is an electronic computer whose principal function is to process data. The computer component that does this is called the **central processing unit (CPU).** The CPU contains an **arithmetic unit,** consisting of circuitry that performs a variety of arithmetic and logical operations, and a **control unit,** which controls all electrical signals passing through the computer. In addition to the CPU, every computer has a **memory unit** which can store data, and from which data can be retrieved for processing. Fortunately, you need not understand how a computer processes data to make a computer work for you. The circuitry in a computer is not unlike that in an ordinary pocket calculator, and all who have used calculators know that no knowledge of their circuitry is needed to use them.

Data must be transmitted to the computer (*Input*), and results of the processing must be returned (*Output*). Devices meeting these two requirements are called **input** and **output (I/O) devices.** The I/O devices you are most likely to encounter in your introduction to computer programming are as follows.

Teletypewriter and Video Terminals: These serve as both input and output devices. On a teletypewriter (Figure 1.2) you transmit information to the computer simply by typing it at the teletypewriter keyboard and the computer transmits the results back to the teletypewriter, which produces a printed copy for you. A video terminal (Figure 1.3) works the same way except that the results are displayed on a video screen.

Card Readers: Information can be transferred to punched cards for submission to a computer. (This process is discussed in Chapter 4.) A card reader (Figure 1.4) is used to "read" the information punched on these cards and transmit it to the computer. Card readers serve only as input devices.

Line Printers: A line printer (Figure 1.5) serves only as an output device. As indicated by its name, an entire line of output is printed simultaneously. Computer systems that use card readers for input usually use line printers for output.



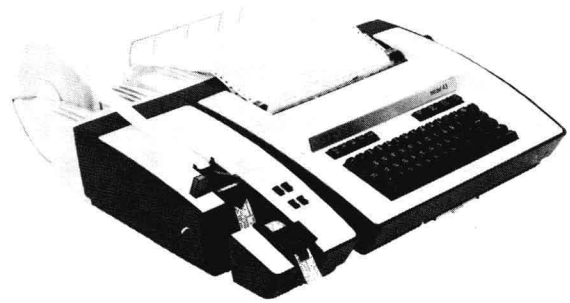


Figure 1.2 (above) ASR Model 43 Data Terminal with paper tape unit. (Courtesy of Teletype Corporation.) (left) DECwriter LA-36 Terminal. (Courtesy of Digital Equipment Corporation.)

此为试读,需要完整PDF请访问: www.ertongbook.com

FIGURE 1.3. Digital VT100 Video display with keyboard. (Courtesy of Digital Equipment Corporation.)

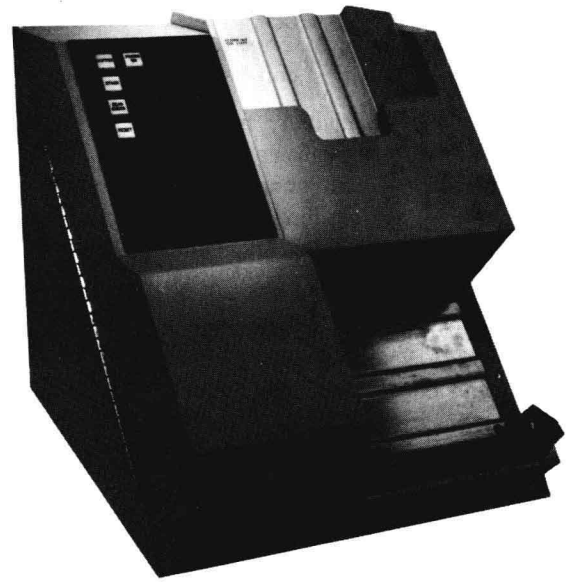


FIGURE 1.4. Card reader. (Courtesy of Control Data Corporation.)

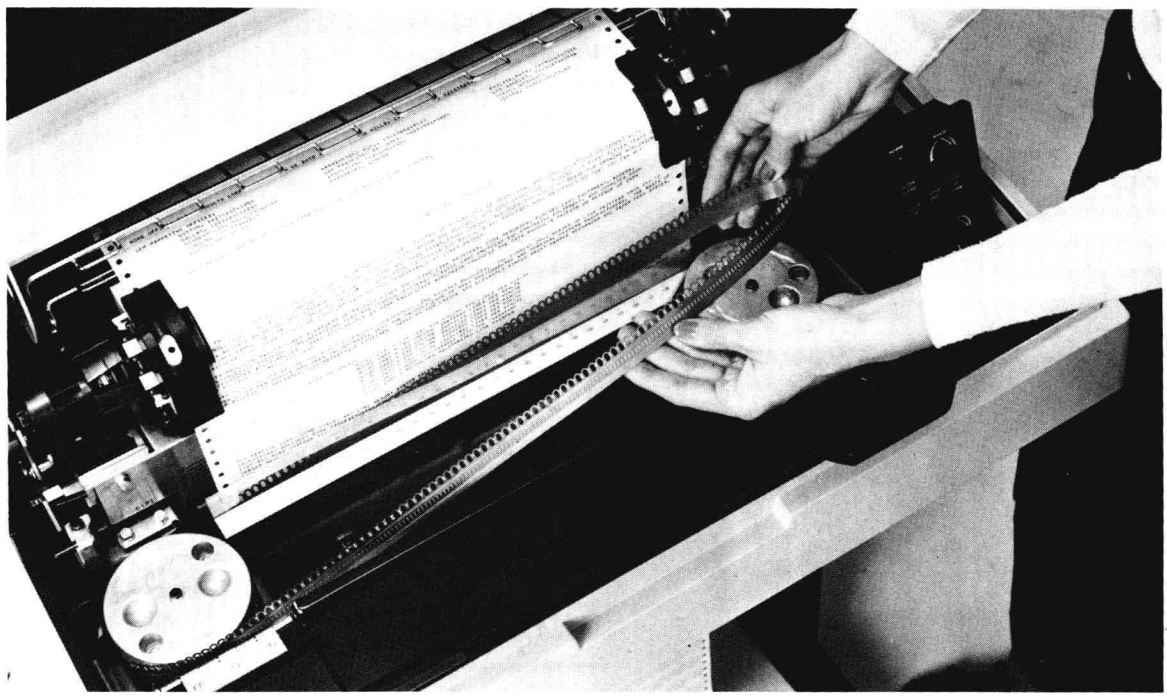


FIGURE 1.5. Dataproducts' B-Series band printer. (Courtesy of Dataproducts Corporation.)

Most modern computer systems are equipped with storage devices other than the memory unit. They are called *external* (or *secondary*) storage devices because they are not a part of the computer as is the memory unit. The most common external storage devices are as follows.

Magnetic-tape units: Information is stored on magnetic tapes as sequences of magnetized "spots." Although tape units can be rather "large" (Figure 1.6), some computer systems (especially microcomputer systems) use ordinary cassette tape recorders. Data are "read" from a tape by reading through the tape sequentially until the desired data are found. For this reason, tape units are called sequential access devices.

Disk-storage units: (Figure 1.7) Information is stored on rotating disks that resemble phonograph records. However, the disks have no grooves; the data are stored as sequences of magnetized spots appearing on concentric circles. A disk unit will contain one or more disks, each with one or more read/write heads. Disk units are called random access devices. This term indicates that data stored on any part of a disk can be accessed directly without having to read through the entire disk to find the desired data.

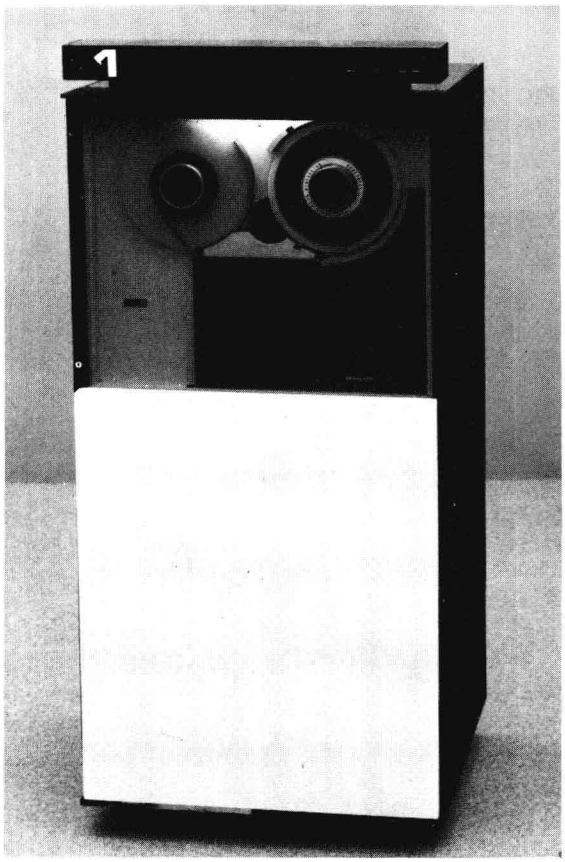


FIGURE 1.6. Magnetic tape unit. (Courtesy of Honeywell Information Systems.)

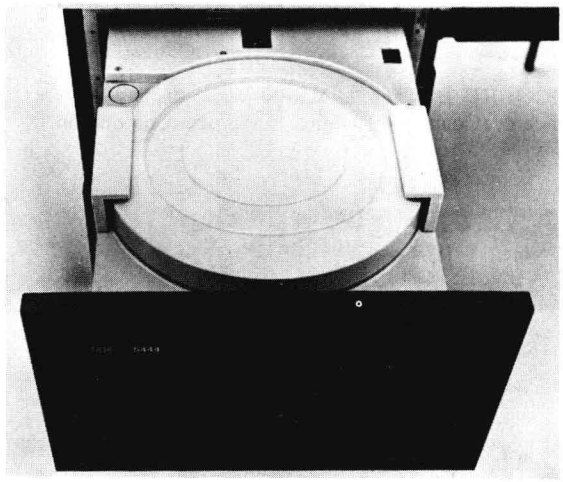


FIGURE 1.7. IBM 5444 Disk unit. (Courtesy of IBM Corporation.)

Data terminals, card readers, tape units, disk units, and all other mechanical and electrical devices other than the computer itself are referred to as **computer peripherals.** The computer and all peripherals constitute what is called the **hardware** of the computer system. Figure 1.8 illustrates the flow of information between a computer and its peripherals.

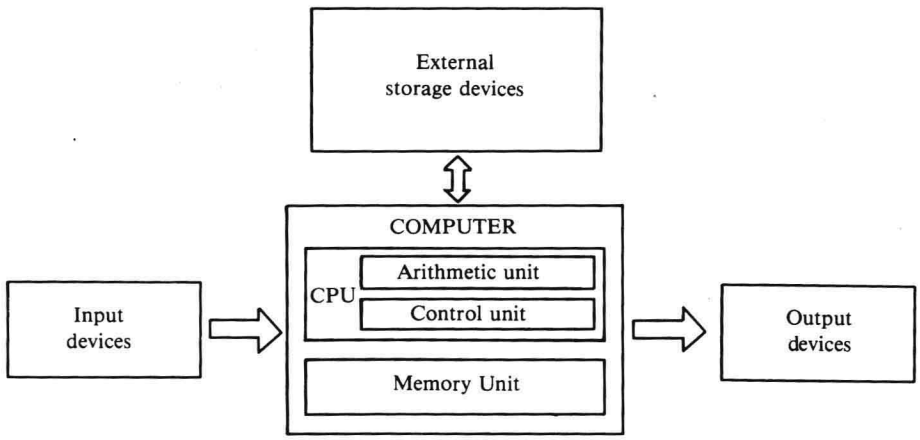


FIGURE 1.8. Flow of information through a computer system.

1.2 Computer Software

The physical components, or hardware, of a computer system are inanimate objects. They cannot prepare a payroll or perform any other task, however simple, without human assistance. This assistance is given in the form of instructions to the computer. A sequence of such instructions is called a **computer program,** and a person who determines what these instructions should be is called a programmer.

The precise form that instructions to a computer must take depends on the particular computer system being used. **FORTRAN** (FORmula TRANslation) is a carefully constructed English-like language used for writing computer programs. Instructions in the FORTRAN language are designed to be understood by people as well as by the computer. Even the uninitiated will understand the meaning of this simple FORTRAN program:

```
TERM1=3
TERM2=9
SUM=TERM1+TERM2
PRINT SUM
STOP
END
```

A computer is an electronic device and understands an instruction such as TERM1 = 3 in a very special way. An electronic device can distinguish between two distinct electrical states. Consider, for instance, an ordinary on/off switch for a light fixture. When the switch is in the "on" position, current is allowed to flow and the light bulb glows. If we denote the "on" position by the number 1 and the "off" position by the number 0, we can say that the instruction 1 causes the bulb to glow and the instruction 0 causes it not to glow. In like manner, we could envision a machine with two switches whose positions are denoted by the four codes 00, 01, 10, and 11, such that each of these four codes causes a different event to occur. It is this ability to distinguish between two distinct electrical states that has led to the development of modern computers. Indeed, modern computers are still based on this principle. Each computer is designed to "understand" a certain set of primitive instructions. On some computers these instructions take the form of sequences of 0's and 1's, but their precise form is not important to the beginner. All such primitive

instructions that are meaningful to a particular computer are together called the **machine language** for that computer.

You will not be required to write programs in machine language. The computer you use will contain a **compiler,** which automatically translates your FORTRAN instructions into equivalent machine-language instructions that are then executed by the computer (Figure 1.9). Thus, when the six-line FORTRAN program shown above is presented to the computer, the compiler produces an equivalent machine-language program that instructs the computer to perform the specified task. The FORTRAN program is called the **source program** and the machine-language program is called the **object program.**

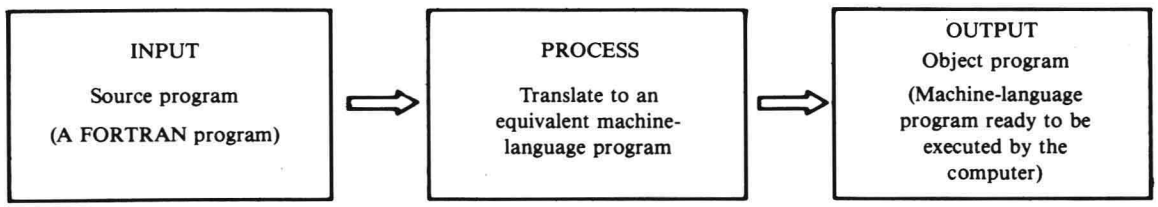


FIGURE 1.9. INPUT–PROCESS–OUTPUT diagram for a FORTRAN compiler.

FORTRAN compilers are themselves computer programs. They are called **system programs** because they are an integral part of the computer system being used. The FORTRAN programs appearing in this text, as well as the programs you will write, are called **application programs.** They are not an integral part of the computer system, so they are not called system programs. All computer programs, both system programs and application programs, are called **computer software.** The term *software* refers not only to computer programs, but also to any documentation, such as manuals and circuit diagrams, concerned with the operation of computers.

In addition to a FORTRAN compiler, your system will most likely include other system programs. These include programs to produce printed listings of your programs, to "save" your programs on secondary storage devices for later use, to assist you in finding errors in the programs you write, and, most important of all, a program called the **operating system** that exercises general control over the entire system. The operating system allows you to issue commands to the computer to "call up" and execute any of the other system programs provided.

The emergence of computer science as a new discipline has been accompanied by a proliferation of new words and expressions. They are useful for talking about computers but are, for the most part, absolutely unnecessary if your objective is to learn a computer language such as FORTRAN to assist you in solving problems. In our discussion of computer hardware and software, we have attempted to introduce only fundamental concepts and frequently used terminology. If this is your first exposure to computers, you may feel lost in this terminology. Don't be disheartened: much of the new vocabulary has already been introduced. You will become more familiar with it and recognize its usefulness as you study the subsequent chapters. You will also find it helpful to reread this chapter after you have written a few computer programs.

1.3 Review True-or-False Quiz

1. The principal function of a computer is to process data. T F
2. The term *arithmetic unit* is another name for the CPU. T F
3. I/O devices, external storage devices, and the central processing unit are called computer peripherals. T F
4. Card readers and video terminals can be used as input devices but cannot be used as output devices. T F
5. Disk storage units are called *random access devices* because information stored on a disk is accessed by randomly searching portions of the disk until the desired data are found. T F
6. Tape units are called *sequential access devices* because information is read from a tape by reading through the tape until the desired data are found. T F

7. The function of a FORTRAN compiler is to translate FORTRAN programs into machine language. T F
8. A FORTRAN compiler is part of the hardware of a computer system. T F
9. A FORTRAN program written to solve a particular problem may accurately be called a system program. T F
10. The machine-language program obtained when a FORTRAN program is compiled is called the object program. T F
11. An operating system is a computer system. T F
12. The expressions *computer software* and *computer program* are synonymous. T F

此为试读,需要完整PDF请访问: www.ertongbook.com