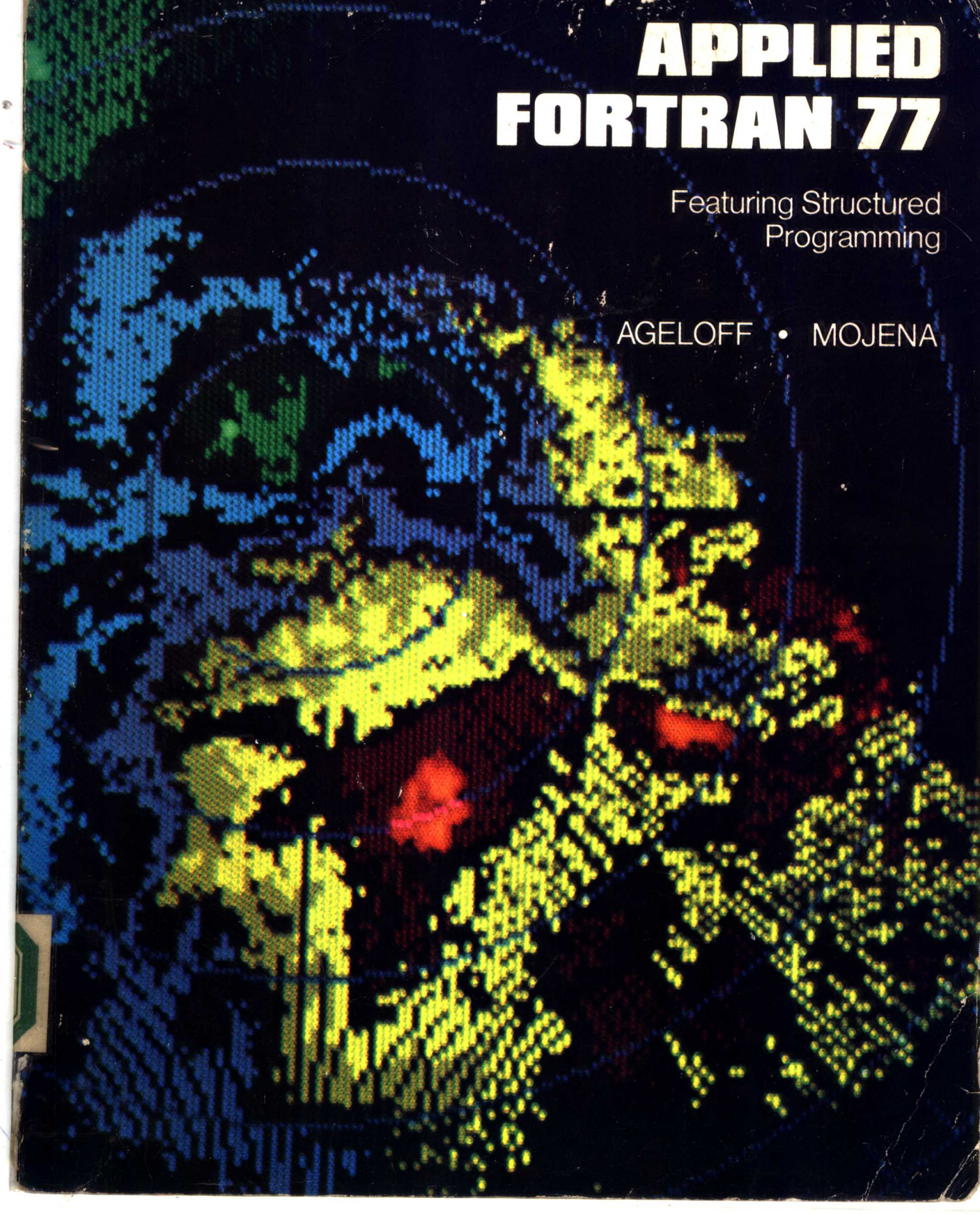
APPLIED FORTRAN 77

Featuring Structured Programming

AGELOFF • MOJENA



Applied
FORTRAN 77
featuring
Structured Programming

ROY AGELOFF
University of Rhode Island

RICHARD MOJENA
University of Rhode Island

WADSWORTH PUBLISHING COMPANY
Belmont, California
A division of Wadsworth, Inc.

ISBN 0-534-00961-1

Library of Congress Cataloging in Publication Data

Ageloff, Roy, 1943–
Applied FORTRAN 77.

Includes index.
1. FORTRAN (Computer program language)
2. Structured programming. I. Mojena, Richard, joint author. II. Title.
QA76.73.F25A33 001.64'24 80-27418
ISBN 0-534-00961-1

Editorial production services by Cobb/Dunlop Publisher Services, Inc.

About the Cover: Cover shows Doppler radar, the latest method for detecting storms in early stages. Colors represent varying wind velocities. Photograph courtesy Severe Storms Laboratory, NOAA, Norman, Oklahoma.

Printed in the United States of America

10 11 12 13 14 15—90 89 88 87

Preface

This textbook is designed for a first course in FORTRAN programming. No prerequisites are required other than a willingness to develop problem-solving skills.

The combination of features described below distinguishes this book from others in the field.

FORTRAN 77 implementation. FORTRAN 77 is the basis of the FORTRAN material, as established by the document labeled ANSI X3.9-1978. By FORTRAN 77 we mean "full" FORTRAN 77 rather than "subset" FORTRAN 77. Statements and functions used in this book are summarized and illustrated inside the front and back covers.

Evolutionary design. Following a comprehensive overview of the field in Chapter 1, coverage of programming proceeds from simple to difficult, with the student writing complete programs by the end of Chapter 2 and running programs by the end of Chapter 3.

By design, the pace of Chapters 2–4 builds slowly, to encourage confidence and to develop a sound foundation. This approach necessarily discards the complete treatment of a topic in one place. For example, loop structures are first overviewed in Chapter 4, and a DO-loop implementation is illustrated. General DO-WHILE, DO-UNTIL, and Last-Record-Check loops, however, are not implemented until Chapter 6.

Structured programming. The structured programming philosophy is presented early in Chapter 4, elaborated upon in Chapters 4–6, and adhered to throughout the book.

Chapter 4 implements the loop structure through DO/CONTINUE statements. The early implementation of the loop structure based on DO/CONTINUE statements has several advantages: it allows the early introduction of realistic problems and the power of repetitive processing; it's "cleaner" than the IF/GO TO approach; and it minimizes breadth of coverage (thereby student confusion) in one chapter by postponing the introduction of IF-type statements.

Program design and style. Good program design and style are emphasized throughout the book. For example, in Chapter 2 we present a four-step program-writing procedure that includes flowcharts and pseudocode. Other style considerations include documen-

tation (from Chapter 3 on), indentation (from Chapter 4 on), structured programming (Chapters 4–11), the "proper" use of GO TO statements (Chapter 6), top-down design (Module E), and modular programming (Module F).

Additionally, Module G summarizes a number of issues in good program design, including the interrelationships among structured, top-down, and modular programming; the readability versus efficiency tradeoff; the design of general versus specific programs; the use of documentation aids; and considerations regarding debugging, transportability, reliability, and management practices.

Use of Modules. We have attempted to design flexibility into the use of this book by placing certain topics in modules at the book's end. Intrinsic functions (Module A) are first used in Chapter 2; I/O with formats (Modules B–D) is not used specifically until Chapter 7, but can be assigned at any time after Chapter 4; top-down design, including stepwise refinement (Module E), can be assigned either during or after Chapter 5; modular programming (Module F) can be assigned at the end of Chapter 9; and Module G (On Designing Better Programs) can be assigned any time after Chapter 6.

Emphasis on meaningful applications. The word "applied" in the title of the book is used to suggest our emphasis throughout on meaningful applications of the computer. Applications include those relating to *information processing* and those relating to *mathematical modeling*. They are described in a wide variety of contexts, including areas in business, economics, mathematics, statistics, and the sciences, as well as emerging areas in the public sector (such as health care delivery, emergency response systems, and allocation of public resources).

Table A summarizes and references the applications described in the book through examples and exercises. This table clearly illustrates our philosophy that problems should be presented in an *evolutionary* context. As new material is learned, many examples and exercises improve upon previous versions of the same problem. This approach not only is pedagogically sound but also is consistent with (but not identical to) the evolutionary nature of program design in the "real world."

Extensive examples and exercises. The learning of FORTRAN is greatly facilitated by numerous and carefully designed examples and exercises. More than 60 *complete* programs are illustrated in this book; entirely new programs are accompanied by flowcharts and/or pseudocode. Exercises are found both within chapters (Follow-up Exercises) and at the end of chapters (Additional Exercises). The book has 493 exercises, many with multiple parts. The chapters on programming (Chapters 2–11) average better than 40 exercises per chapter.

Follow-up exercises serve to reinforce, integrate, and extend preceding material. This feature gives the book a "programmed learning" flavor without the regimentation of such an approach. Additionally, we have found that such exercises create an excellent basis for planning many classroom lectures. *Answers to selected follow-up exercises are provided at the back of the textbook.* Solutions to those follow-up exercises identified by either a single or double asterisk are given in the *Instructor's Manual.*

The chapter-end exercises offer opportunities for review and the development of new programming problems. All programming problems include test data. Examples

TABLE A Page References for Applications Programs

Information Processing

Mathematical Modeling

and exercises are generally framed in a "real world" context that will interest and motivate the student. Exercises are ordered from least difficult to most difficult. The more difficult exercises are designed to challenge the good student, and are identified by a double asterisk. The *Instructor's Manual* gives answers to all chapter-end exercises.

Common errors. The necessary process of debugging is time consuming, frustrating, and difficult to master by beginning programmers. In our experience, students commit certain programming errors more commonly than others. Accordingly, the book features sections on debugging procedures and common errors at the end of *each* programming chapter, beginning with Chapter 3 and ending with Chapter 11.

I/O with formats. Our treatment of I/O with formats is carefully designed to give instructors as much flexibility as possible regarding the choice of list-directed I/O versus format-directed I/O. The topic (I/O with formats) is set off in modules (B–D), which can be assigned as early as the end of Chapter 4. Within chapters, the use of format specifications is postponed until Chapter 7; thereafter the use of format specifications is either clearly labeled (and easily skipped) or incidental to the material.

In effect, the book is essentially independent of I/O with formats, but is designed to facilitate timing and integration of the topic. Our own preference is to delay its introduction until after Chapter 6, to provide ample time first for algorithmic design.

Batch and time sharing. Since the computer systems at institutions of higher education vary, we illustrate both batch and time-sharing approaches to computing. Time sharing is given more than a superficial treatment in keeping with its increased use in universities, governmental agencies, and companies.

To accommodate both batch and time-sharing users, most programs and discussions of FORTRAN are independent of processing environments. For example, we consciously use the term input record instead of card.

Where appropriate (Chapters 3 and 11) we design programs for either batch or time-sharing environments. This allows us to describe features of each system. In general, however, the book can be used without bias by either batch or time-sharing users. Better yet, as in our own treatment of programming, it can be used by courses that require exposure to both processing environments. In particular we favor the growing use of *batch interface*.

A book on programming and problem solving, not a programming manual. We believe that a FORTRAN course should be much more than just a course that teaches the FORTRAN language. It should teach the process of programming as a creative activity, from conceptualization of the problem to implementation of the computer program.

In keeping with this belief, we sacrifice some scope in the FORTRAN language by devoting more space to pedagogy through patient explanations and extensive examples and exercises. Additionally, we emphasize the development of problem solving by formalizing the writing of programs through a four-step procedure first introduced in Chapter 2, which subsequently is integrated with top-down design and modular programming.

A programming course also should broaden a student's perspective. Accordingly, Chapter 1 presents a more thorough overview of the field than does the typical introductory chapter, and Module G summarizes style issues of keen topical interest.

ACKNOWLEDGMENTS

We wish to express our deep appreciation to many who have contributed to this project: to Jon Thompson, our editor, for unflagging encouragement, support, and expert advice; to Jerry Holloway of Wadsworth and Lila Gardner of Cobb/Dunlop for editorial/production magic and liaison par excellence; to Richard R. Weeks, Dean, University of Rhode Island, for administrative support; to David Wishart, CLUSTAN, and A. Jack Cole, Computational Science Department, University of St. Andrews, Scotland, for their generous and kind support; to the Computer Laboratories at the University of Rhode Island and the University of St. Andrews, for obvious reasons; to our reviewers, Nell Dale, University of Texas, Austin, Donald R. Chand, Georgia State University, Mark Luker, University of Minnesota, Duluth, Dale Grosvenor, Iowa State University, Kenneth Joy, Northern Michigan University, Thomas J. Murray, University of Missouri, St. Louis, and Donald Woods, Texas A & M University who provided invaluable corrections and suggestions for manuscript revisions; to Fred Wild, Andrew Ehrlich, and Susan Rose for "grunt" work on end-of-chapter solutions; to Fran Mojena and Marjorie Nield for their skill, patience, and steadiness in typing the manuscript; to our students, who always teach us something about teaching; and to our immediate families, who are wondering when it will all end.

January, 1981
Kingston, Rhode Island

ROY AGELOFF
RICHARD MOJENA

Contents

CHAPTER 1

Orientation

The electronic computer is one of humankind's foremost technological inventions; for good or for bad, its presence affects each of us, and its future holds even more potential to affect our lives.

This chapter is an orientation to the course you are about to take. We first define the computer and discuss its impact. Thereafter we provide a relatively complete, nontechnical overview of what makes up a computer system and a preview of how to communicate with the computer. Finally, we outline how you will benefit from this course.

If you are warm-blooded and living in the twentieth century, then we suspect that you are curious about the computer. By the time this course is over we hope that we (together with your instructor) will have helped you translate that curiosity into a continuing, productive, and rewarding experience.

1.1
WHAT IS A COMPUTER?

A **computer** can be defined most generally as *a device which is capable of manipulating data to achieve some task.* Given this definition, adding machines, cash registers, gasoline pumps, and electronic calculators all qualify as simple computers. The machine we usually think of as a computer, however, can be identified by four significant characteristics.

1. It's electronic.
2. It's fast.
3. It can store large amounts of data.
4. It can execute stored instructions.

Characteristics of Electronic Computers

The great speed of today's electronic computers is a direct result of miniaturization in solid-state electronics. To give you a rough idea of the speed capabilities of large electronic computers, consider the following estimates. One minute of computer time is equivalent to approximately 6700 hours of skilled labor by a person using a calculator.

In other words, a person using a calculator would take one hour to accomplish what a computer can accomplish in less than one hundredth of a second. In fact, the electronic transfers within computers are so fast that computer designers use a basic unit of time equal to one billionth of a second (called a *nanosecond*)—quite a feat when you consider that the basic unit of time for us mortals is one second.

Another significant characteristic of electronic computers is their capacity to store large amounts of data and instructions for later recall. In other words, much like the human brain, the computer has "memory." For example, computers at most universities can store several million characters of data in primary storage and hundreds of millions of characters in secondary storage.

Finally, an electronic computer is differentiated from most other computing devices by its ability to store instructions in memory. By this we mean that the computer can execute a set of instructions without interference from human beings. This characteristic makes the computer efficient: it can carry on automatically while we do something else. Of course, the computer cannot completely do without us, but more about that later.

Computer Classifications

To further narrow the definition of an electronic computer, we make the following distinctions: analog versus digital computers and special-purpose versus general-purpose computers.

The **analog computer** manipulates data representing continuous physical processes such as temperature, pressure, and voltage. The fuel injection system of an automobile, for example, deals with physical processes as it regulates the fuel/air ratio in the carburetor on the basis of engine speed, temperature, and pressure; the gasoline pump converts the flow of fuel into price (dollars and cents) and volume (gallons to the nearest tenth). Not surprisingly, therefore, analog computers are used primarily to control such processes. For example, analog computers now control the production of products such as steel and gasoline, provide on-board guidance for aircraft and spacecraft, regulate the peak energy demands of large office buildings or factories, and monitor the vital life signs of patients in critical condition.

As a strict computational device, however, the analog computer lacks the precision one needs with counting. Place yourself in the role of a computer that has the task of adding the numbers 1 and 2. Your props are a ruler, pencil, paper, and a jar of beads. You might proceed with your task as follows: First, you take one bead from the jar and place it on the paper. Next, you take two beads from the jar and place them on the paper. Finally, you count the number of beads you have on the paper. *Exactly* three, right? Now, be an analog computer. With pencil, paper, and ruler, draw a line 1 inch in length. Next, draw a 2-inch line at the end of the 1-inch line you drew earlier. Now measure the length of this overall line. Is your line exactly 3 inches long? Not really, only *approximately 3*—as the accuracy of your answer depends on the precision of the scale on the ruler, the steadiness of your hand, the acuteness of your eyesight, and the sharpness of your pencil point. When it comes to calculating, the counting approach based on beads is more accurate than the approach based on measurement.

You will be using the **digital computer,** which operates by counting digits. This type of computer manipulates data (numbers in our decimal system, letters in our alphabet,

and special characters) by counting binary (two-state or 0–1) digits. **Hybrid computers,** which combine the features of digital and analog computers, have been designed for certain types of applications, such as the analysis of aircraft designs that are tested in wind tunnel experiments.

We have been classifying computers by how they process data, but we can also classify them according to their function. **Special-purpose computers** are designed to accomplish a single task, whereas **general-purpose computers** are designed to accept programs of instruction for carrying out different tasks. For example, one special-purpose computer has been designed strictly to do navigational calculations for ships and aircraft. The instructions for carrying out this task are built into the electronic circuitry of the machine so that the navigator simply keys in data and receives the answer. Other special-purpose computers include those used in color television sets to improve color reception; those used in PBX (Private Branch Exchange) telephones to perform various functions, such as automatic placement of a call at a preset time and simplified dialing of frequently used telephone numbers; and those used in automobiles to calculate such items as "miles of fuel left" and "time of destination," and to monitor and read out instantaneously the status of oil level, gasoline level, engine temperature, brake-lining wear, and other operating conditions.

In contrast, a general-purpose computer used by a corporation might accomplish tasks relating to the preparation of payrolls and production schedules and the analyses of financial, marketing, and engineering data all in one day. Similarly, the academic computer you are about to use might run a management simulation one minute and analyze the results of a psychology experiment the next minute, or it might even accomplish both of these tasks (and more) concurrently.

In general, compared with the special-purpose computer, the general-purpose computer has the flexibility of satisfying the needs of a variety of users, but at the expense of speed and economy. In this textbook, we focus strictly on the *electronic, digital, general-purpose computer.*

1.2 IMPACT OF THE COMPUTER

Since the first sale of an electronic computer by Remington Rand in 1951, the computer industry has grown to such an extent that by the mid-1970s it had generated over $75 billion in sales and provided at least 700,000 jobs. The computer has revolutionized the operations of many governmental agencies, private enterprises, and public institutions, and many experts agree that the computer industry is a "young child," if not still an "infant." In this section we present a brief historical sketch of the development of the computer, provide you with a sample of computer applications, and end with an assessment of the computer's impact.

Historical Sketch

Many conceptions and inventions dating back to the early nineteenth century were necessary precedents to the development of the computer. The first digital general-purpose computer was completed in 1944 when Howard Aiken at Harvard University designed the **Mark I** to generate mathematical tables. Unlike electronic computers, the Mark I was a mechanical computer that operated by a system of telephone relays,

mechanized wheels, and tabulating equipment. By current standards, it was *very* large, *very* unreliable, *very* slow, and *very* limited in its scope of applications. In 1946 the team of J. W. Mauchly and J. P. Eckert, Jr., from the University of Pennsylvania, completed the first *electronic* computer. This computer was named **ENIAC,** for the intimidating title Electronic Numerical Integrator And Calculator. Essentially, ENIAC was an electronic version of Mark I, in which vacuum tubes replaced the function of telephone relays; this replacement resulted in an increase of computing speed by a factor of nearly 200. Commissioned by the U.S. Army, it did an incomparable job (for the times) of generating artillery trajectory tables, but weighed 30 tons, and filled a 150-square-meter room.

UNIVAC I (Universal Automatic Computer), developed by Remington (now Sperry) Rand in 1951, was the first commercial computer. Unlike its predecessors, it computed using binary arithmetic and allowed the storage of instructions in internal computer memory. During this **first-generation** period computers were developed by RCA, Philco, GE, Burroughs, Honeywell, NCR, and IBM. The first computer to achieve dominance in the industry was the IBM 650, which became the commercial leader during the period 1954–1959. These first-generation machines used vacuum tubes, required air conditioning, had relatively small amounts of internal memory, and were slow by today's standards.

Subsequent generations of computers resulted in dramatic reductions in *size* and relative *cost* and increases in *speed, reliability,* and the capacity for *storage*. **Second-generation** computers during the period 1959–1965 replaced the vacuum tubes of the first-generation computers with transistors. The most widely used second-generation computers were the IBM 1620, the IBM 1401, and the IBM 7094.

The **third-generation** computers (1965–1970) that followed made use of the emerging field of microelectronics (miniaturized circuits), which increased the packing densities of transistorized circuits by a factor of 100. The third-generation computers were more reliable, faster, and more sophisticated than earlier computers. They also had the ability to handle several programs concurrently (multiprogramming), resulting in a more efficient use of the computer. The most prominent family of computers in this generation was the IBM System/360.

During the 1970s, a series of refinements and improvements to third-generation machines were marketed. These computers utilized large-scale integrated circuitry (LSI) and other microminiaturization features, resulting in further reductions in size and power requirements as compared with earlier computers.

Another significant development in the 1970s was the use of small (in physical size and memory capacity), inexpensive, yet powerful computers referred to as **minicomputers** and **microcomputers.** The use of minicomputers is common in small to medium companies, colleges, hospitals, governmental agencies, and other organizations. In the past decade many organizations have decentralized their computer processing activities by implementing what is called **distributed processing,** whereby a network of computers links together geographically remote locations. A typical configuration is a minicomputer at each location for either local processing tasks or communication to a large central computer for major processing tasks.

Microcomputers, which are smaller than minicomputers, currently are marketed by consumer retail outlets such as Radio Shack. Today's desk-top microcomputers are

20 times faster, have larger memory, are thousands of times more reliable, consume the power of a light bulb rather than a locomotive, occupy one 30,000th of the volume and cost one 10,000th ($500 versus $5 million) as much as the massive first-generation machines that filled an entire room. Their use in small organizations and homes is expected to increase dramatically in the next decade.

Applications

The computer represents a revolutionary technological tool for extending our applied capabilities. The diversity of the sample applications listed in Table 1.1 should give you

TABLE 1.1 Sample Applications of the Computer

Information Processing

Preparation of payroll and billings
Maintenance of inventory information
Maintenance of customer accounts
Technical processing of reference information by media and public libraries
Calculation of income taxes by the IRS
Maintenance of student records by universities
Maintenance of flight and reservation information by airlines
Cataloguing of blood supplies by regional blood banks
Maintenance of checking accounts by banks
Editing and reproduction of typed manuscripts
Maintenance of criminal records by the FBI
Maintenance of property tax records by a municipality
Budgeting by organizations and individuals
Recording of monetary distributions by state and federal welfare agencies

Mathematical Modeling

Statistical analyses of census data, biological data, engineering data, etc.
Production scheduling and inventory control
Medical diagnosis
Orbital analysis for satellites
Management of financial portfolios
Location of fire stations in an urban area
Simulation of economic decay in a city
Dietary meal planning in institutions
Statistical forecasting
Educational planning and school bus scheduling
Design of airway and highway traffic systems
Chemical analysis
Design of solar energy systems
Planning, scheduling, and controlling of complex projects (such as construction of a submarine, office building, or sports stadium)