RICHARD V. ANDREE
JOSEPHINE P. ANDREE
DAVID D. ANDREE

Computer Programming:

Techniques, Analysis, and Mathematics

COMPUTER PROGRAMMING

Techniques, Analysis, and Mathematics

RICHARD V. ANDREE
Professor of Mathematics
Professor of Information and Computing Sciences
The University of Oklahoma

JOSEPHINE P. ANDREE

DAVID D. ANDREE

Prentice-Hall, Inc., Englewood Cliffs, New Jersey

10 9 8 7 6 5 4 3 2 1

ISBN: 0-13-166082-9
Library of Congress Catalog Card Number: 72-5254

Printed in the United States of America

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA, PTY. LTD., *Sydney*
PRENTICE-HALL OF CANADA, LTD., *Toronto*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*

PREFACE TO THE INSTRUCTOR

This book is designed to be read by students. Experience shows that students can and will read, understand, and enjoy it. It is designed not merely to teach FORTRAN IV coding, but to teach *efficient programming techniques*.

Perhaps the most important part of this volume is the outstanding collection of interesting problems. Users of the preliminary editions indicate real student enthusiasm. These problems are *not* the result of chance. A good problem, well stated, is indeed a gem. A vast fortune in such problem-gems is contained in this one volume. We hope you enjoy each one.

There is a tendency among some students to try problems first; then to examine the illustrative examples; and finally, as a last resort, to read the text. Knowing this, a good deal of additional instructional material is included in the answer section, particularly for the beginning chapters. Each point in the text is also covered in the problems. Each problem is designed to relate directly to knowledge which the student needs. There are more problems than any one student or even one class can be expected to work completely, but don't be surprised if you have questions on problems that were not assigned.

Our experience is that Chapter 1, which is designed to teach the student to *read* simple FORTRAN programs, not to write them, should be covered rapidly. In our own classes we usually work most of Problem Set 1-4 in class during the first lecture, and assign Problem Sets 1-6 and 1-7 to be worked for the second class meeting. Individual problems of Set 1-10 are assigned to be worked for the third class session. Since the answers for all of the problems in Chapter 1 are available in the back of the text, extensive class discussion is unnecessary. As the student progresses further into the text, his need for answers should diminish. Answers to odd-numbered problems are given for Chapters 2, 3, and 4. Some problems require discussion and opinion rather than programs or numerical answers.

This text may be used with any fixed-word-length, binary computer, with or without terminals, as long as a FORTRAN IV (or WATFOR or WATFIV) compiler is available. The last half of Chapter 8 uses the IBM 1130 assembly language as an example.

Throughout this text the student is encouraged to think and understand on his own. The aim of this text is *problem solving*, not FORTRAN coding. Mathematics is the study of basic structure, and this is a mathematics book. Without an understanding of the basic structures of the techniques involved, no real computing skill can be developed. Surprisingly, the mathematical prerequisites for this course are usually met by the second course in high school algebra. Some of the material in Chapters 1 to 4 has been used with ninth grade students. So skillfully are the additional mathematical concepts evolved that, when the student completes the course, the vital mathematical concepts have been developed along with the theory of computing. Although your three authors have worked hard to accomplish this blend, the real credit goes to the more than 100 college professors and high school teachers who have used these notes during the last four years in summer institutes. Their frank and occasionally jeering comments have resulted in myriad revisions, which we hope have produced a book worthy of their cooperation.

One serious drawback in teaching computer programming in an academic situation is that the student feels that computer time is "free" and hence has little incentive to learn *efficient* computing techniques. Few employers favor this attitude. Our society can no longer afford the inefficiency of computer programs which waste 40% of the valuable computer time to obtain inadequate answers. This book emphasises efficient techniques throughout.

This book provides the computing techniques and computer-related mathematics that a beginner needs to continue in information and computing sciences. It does not lull him into the belief that he now fully understands what computing is all about. Many important doors are opened enroute which the student may either explore or ignore as his own talents and interests, and those of his teacher, dictate. In short, it is the type of book from which we sincerely wish we had been able to study.

Many years of teaching experience both in computing science and in related mathematics have helped in the selection, arrangement, and presentation of the material in this text. Chapters 1, 2, 3, and 4, should probably be presented in the order given. Any of Chapters 5, 6, 7, and 8, may be presented in any order, and then the brief Chapter 9 provides an excellent closing for the course with summaries and a review of the techniques for *efficient* programming stressed in Chapters 2, 3, and 4.

D.D.A.
J.P.A.
R.V.A.

PREFACE TO THE STUDENT

This text was written for *you*. Your three authors have done their best to teach you what you really need to know about computing—not merely FORTRAN coding, but also the much more important topics that make the difference between a dilettante and an expert.

It is easy to write computer programs that will run and produce results on a modern computer. We have taught 7th and 8th graders to do so. Within a week **you** will be able to write useful programs which will help solve problems that you might well consider impossible or at least unreasonable without a computer. Here are two samples:

1. Without using existing tables, create a table showing the values of

$$x,\quad x^2,\quad x^3,\quad x^5,\quad x^{10},\quad \sqrt{x},\quad \sqrt[3]{x},\quad \sqrt[4]{x},\quad \sin x,\quad \log x$$

for $x = 0.1, 0.2, 0.3, \ldots, 9.9, 10.0$

2. Obtain at least one real solution of the equation

$$x\sqrt{\sin x + 4.2} - 7.6 = 0$$

Either of these problems could require hours of computation, if solved without the use of a computer—yet with a computer each can be solved in ten minutes (within thirty seconds, once the computer instructions are written).

Such facility may seem a highly desirable goal itself—perhaps it is. However, it is merely a *byproduct* of our true goal, which is to teach you the simple mathematical analysis necessary to solve problems *efficiently*, using both your intelligence and the computer's accuracy and speed to obtain the solution. Even though a modern computer is an almost unbelievably reliable electrical device (see page 7), it often produces unexpected answers for those who use it. Sometimes these "incorrect" answers result from an error in the instructions

(program) given to the computer, but more frequently they occur because the computer does not use the rational number system with which most human beings compute. Throughout this book, we, your authors, have worked hard to help you avoid the pitfalls which plague most neophytes. Please let us guide you along the charted path and teach you to avoid these difficulties.

It is easier to *read* any language than to write it. Chapter 1 teaches you to *read* Standard FORTRAN IV computer language and introduces concepts and vocabulary needed throughout the course. Chapter 2 teaches you to write programs, and gets you started in the important new study of algorithms (methods). It also introduces some of the peculiar (nonfield) properties of the floating-point number system. Knowing *how* to read and write (code) instructions for a computer is *not enough.* You also need an understanding of *what* to code—and of *what not to code*—and of the related mathematical algorithms. Top salaries in the growing computer field are reserved for employees who are able to *analyze problems* and *devise valid mathematical techniques* for their **efficient** *computer solution*, rather than those who merely write computer instructions, based on the analyses provided by others. This book has enabled many students to acquire the basic knowledge of *Computer Programming: Techniques, Analysis, and Mathematics* needed to enter the fascinating new world which computers are creating. It can do the same for you if you will let it.

One of your coauthors, David Andree, served as a student assistant during the more than four years this material was used in mimeographed form. He rewrote those portions of the text that gave students trouble. This does not mean the text will present no difficulties. A problem worthy of computer solution is often *inherently difficult* (as well as time-consuming and too tedious to do by hand). It would be false teaching to restrict problems to easy ones. Chapters 3 and 4 contain many problems worthy of serious consideration.

A few minutes examination of the Table of Contents will disclose the general organization of the text. After that, you are ready to read Chapter 1.

ACKNOWLEDGEMENTS

It is impossible to recognize individually the hundreds of teachers and students who participated in the evolution of this text. However, it would be negligent not to express our sincere thanks to two particular confrères:

Jeanne MacKay—who faithfully supervised the production of the preliminary manuscript from initial rough drafts to final proofs. Her common sense and competence have made our task much easier.

Paula V. May—who started learning about computers using the first draft of this book, and is now an instructor. Her suggestions and cooperative assistance both as student and instructor have made our lives as well as this text richer and more fun.

CONTENTS

Each instructor will, of course, emphasize those portions of this book that most closely meet the needs and interests of his students. The original material has been taught by several instructors in various schools. A suggested number of lessons for each chapter based on this experience is presented as a possible guide for teachers using the material for the first time. Somewhere between 33 and 48 class periods, plus laboratory and testing periods, should provide adequate coverage of the material for most students.

of varied interests and ability. The many extras of extended ANSI FORTRAN are discussed.

Simulation provides a powerful aid to modern research in many areas. Simulation often demands the use of pseudo-random numbers, which are introduced in this chapter and used in several ways. The Dance Program (Section 5-4) provides a somewhat typical case history of how computer programs actually develop. Students should recognize that programs may not emerge full-blown from the programmer's mind, but often evolve rather slowly. Without dwelling on the sophisticated nature of the statistical and physical theory involved, this chapter introduces the reader to the use of pseudo-random numbers in several important types of simulation.

from one computer to another. The general concept of a single-address machine is used to introduce the concepts, and then specialized using the IBM 1130 assembly language as an example. The object of the chapter is to introduce the reader to the power of assembly language programming, not to provide an entire course in it.

Chapter 1

INTRODUCTION AND THE READING OF FORTRAN IV

1-1 Introduction

The computer is no longer a strange machine to be approached with awe. Instead it is a useful, exasperating, and often indispensable servant which will do *exactly* what it is told to do, providing you speak its language, but not one bit more. It is this latter property that is so exasperating. There is no button labeled, "Oh, you know what I mean, go ahead and do it." A computer—even a very modern computer—*is not* a giant brain. It is a marvelous piece of equipment—a logical engine.

An automobile jack will permit a frail girl to lift a heavy automobile, but we do not say the jack has muscles. Nevertheless, it is true that the jack permits a girl to accomplish a feat which, if done without assistance, would require superhuman muscular strength. In an analogous fashion, a computer can help man do many things that would require superhuman intelligence—but the computer itself is *not* intelligent—not yet, anyway. It has been said that human beings are slow, error-prone geniuses while computers are fast, accurate morons. The statement contains much truth.

Some people do not wish to learn all about computers—they wish merely to be able to use the power of the computer; others wish to learn, in various degrees, about the finer points of computer programming. Basic FORTRAN IV, which this book discusses first, is a powerful language with ingenious translation programs. After only a few hours of study you can place moderately difficult problems on a computer and obtain results. After you have learned FORTRAN, we hope that you will be interested in learning the much more versatile "assembly language" for your computer so you can use the full power of the computer on your problems. Experience in FORTRAN provides a foundation for all programming. The basic concepts involved remain unchanged. FORTRAN is available, with minor variations, on almost