Applied BASIC Programming

Roy Ageloff

Richard Mojen

Applied **BASIC** *Programming*

ROY AGELOFF
University of Rhode Island

RICHARD MOJENA
University of Rhode Island

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

ISBN 0-534-00808-9

Library of Congress Cataloging in Publication Data

Ageloff, Roy, 1943–
Applied BASIC programming.

Includes index.
1. Basic (Computer program language)
I. Mojena, Richard, 1943– joint author.
II. Title.
QA76.73.B3A36 001.6'424 79-21142
ISBN 0-534-00808-9

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

About the Cover: The cover of this book began as a black and white photograph of plastic daisy printwheels. The color was computer-generated through a high-resolution video camera, and the resulting image was converted to a 35mm transparency. Photograph courtesy of AGT Computer Products, Inc., 914 Westwood Blvd., Los Angeles, California 90024.

Printed in the United States of America

9 10——85

To the best part of us
SHANA *and* YARA
with love

TABLE 0.1 Page References for Applications Programs

Information Processing

Mathematical Modeling

They are described in a wide variety of contexts, including areas in business, economics, mathematics, statistics and emerging areas in the public sector (e.g., health care delivery, emergency response systems, allocation of public resources, etc.).

Table 0.1 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."

Module C describes three extensive applications using the *case method* of study popularized by graduate programs in business administration. These cases are designed as capstone programming assignments using a proven technique that simulates reality more effectively than end-of-chapter programming assignments.

Extensive examples and exercises. The learning of BASIC is greatly facilitated by numerous and carefully designed examples and exercises. More than forty *complete* programs are illustrated within the book; about half of these are accompanied by program flowcharts. Exercises are found both within chapters (Follow-up Exercises) and at the end of chapters (Additional Exercises). The book has more than 300 exercises, many with multiple parts. The chapters on minimal BASIC programming (Chapters 3–9) average 34 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 they create an excellent basis for planning many classroom lectures. Answers to selected follow-up exercises are provided at the end of the textbook. Answers to those follow-up exercises marked with either a single (*) or double asterisk (**) are given in the *Instructor's Manual.*

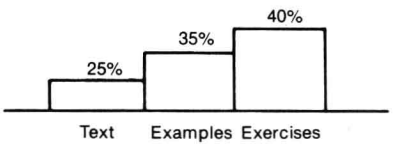
The chapter-end exercises offer opportunities for review and the development of new programming problems. All programming problems include test data. Examples and exercises are generally framed in business, economic, and public sector scenarios to interest and motivate the student. Exercises are ordered from least to most difficult. The more difficult exercises are designed to challenge the good student, and are identified by a double asterisk (**).

The histogram in Figure 0.1 summarizes the breakdown of space devoted to textual matter, examples, and exercises within Chapters 3–12. This further highlights the strong emphasis (75%) placed on examples and exercises within the primary programming chapters.

Evolutionary approach. Coverage of programming proceeds from simple to difficult, with students running complete programs by the end of Chapter 3. By design, the pace of chapters builds slowly, to encourage confidence and to develop a sound foundation. Necessarily, this approach discards the complete treatment of a topic in one place. For example, transfer of control is broken up into Chapters 5 and 6, and I/O is specifically discussed in Chapters 3, 4, 7, 8, 10, 11, and 12.

In addition, many sample programs and exercises are introduced early, and then improved and expanded in later chapters as new features of the BASIC language are presented.

FIGURE 0.1 Percent Breakdown of Chapters 3–12 by Category



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 in *each* programming chapter, beginning with Chapter 4 and ending with Chapter 12.

Top down design and structured programming. The topics pseudocode, top down design, and structured programming in *Module B can be assigned anytime after Chapter 5* by those instructors who wish their students to design and write programs using these techniques. The assignment of Module B before Chapter 7 would serve to review style issues and to consolidate design philosophy at a receptive point in time.

The early assignment of structured programming is especially desirable if the local system supports structured BASIC statements. It is less desirable if minimal BASIC statements are used to simulate control structures to the letter, since forcing an inherently unstructured language (minimal BASIC) into pure structure requires "sleight of hand."

Our personal preference is to live with the unstructured faults of minimal BASIC while keeping in mind the spirit of structured programming (for example, top down execution and minimal use of GO TOs). This tradeoff will remain, of course, until the ANSI committee and the industry see fit to standardize a set of structured BASIC statements, a development we strongly endorse.

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

Our emphasis on applications, examples, and exercises is in keeping with this belief. Additionally, we give structure to the programming process by introducing a four-step procedure in Chapter 2 that facilitates the design and documentation of programs.

A programming course also should broaden a student's perspective. Accordingly, Chapter 1 overviews the field more thoroughly than the typical introductory chapter, and Module B discusses style issues of keen topical interest.

Acknowledgments

We wish to express our deep appreciation to many who have contributed to this project: to Mike Snell and Jon Thompson, our editors, for unflagging encouragement, support,

and expert advice; to Jenny Sill and Paula Delucchi, for liaison par excellence; to Warren Rogers, Chairman, and Richard R. Weeks, Dean, both of the University of Rhode Island, for consistent administrative help; to Diane Marcotte, for overseeing the preparation of hundreds of copies of the manuscript for class teaching; to our students, who suffered through "ditto" copies of the manuscript, yet managed to teach us something about teaching; to our reviewers, Larry Cook, Auburn University, Dennis Coleman, University of Hawaii, Henry Etlinger, Rochester Institute of Technology, Myron Goldberg, Pace University, Richard Hatch, San Diego State University, Richard Lott, Bentley College, R. Waldo Roth, Taylor University, and Clinton Smullen, University of Tennessee, who provided invaluable corrections and suggestions for subsequent revisions; to Fred Wild, for help with the manuscript and "grunt" work on the *Instructor's Manual;* to Fran Mojena, for her skill, patience, and steadiness in typing through several drafts of the manuscript, while claiming that she actually learned something; and to our immediate families, who remained cheerfully supportive in spite of our frequent absence.

January, 1980
Kingston, Rhode Island

ROY AGELOFF
RICHARD MOJENA

Contents

PART IV MODULES

PART I

Foundations

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. Hopefully, by the time this course is over, 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. Electronic
2. Speed
3. Storage capability
4. Ability to execute stored instructions

Characteristics of Electronic Computers

The great speed of today's 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.

A second 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 do its thing 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 represented by 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 based on engine speed, temperature, and pressure; the gasoline pump converts the flow of fuel into price (dollar 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 which 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 out one bead from the jar and place it on the paper; next, you take out 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 one inch in length; next, draw a two-inch line at the end of the one-inch line you drew earlier; now measure the length of this overall line. Is your line exactly three inches long? Not really, only *approximately three,* for 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 which 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 personal business exchange (PBX) telephones to perform various functions, such as automatic placement of a call at a preset time and simplified dialing of frequently used phone numbers; and those used in automobiles to calculate items such 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, breakline 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