BASIC-PLUS And VAX BASIC Structured Programming

EDWARD D. HARTER

ROBERT A. LEOPOLD

BASIC-PLUS And VAX BASIC Structured Programming

EDWARD D. HARTER
Boeing Computer Services

ROBERT A. LEOPOLD
Robert A. Leopold and Associates

Prentice Hall, *Englewood Cliffs, New Jersey 07632*

Library of Congress Cataloging-in-Publication Data

HARTER, EDWARD D., (date)
BASIC-PLUS and VAX BASIC structured programming.
Includes index.
1. BASIC-PLUS (Computer program language)
2. BASIC (Computer program language) 3. VAX-11
(Computer)—Programming. 4. PDP-11 (Computer)—
Programming. I. Leopold, Robert A., (date)
II. Title.
QA76.73.B3H365 1988 005.13'3 87-17457
ISBN 0-13-065905-3

Editorial/production supervision and interior design: Joan McCulley
Cover design: Lundgren Graphics, Ltd.
Manufacturing buyer: Gordon Osbourne

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-065905-3 025

PRENTICE-HALL INTERNATIONAL (UK) LIMITED, *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
PRENTICE-HALL CANADA INC., *Toronto*
PRENTICE-HALL HISPANOAMERICANA, S.A. *Mexico*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
SIMON & SCHUSTER ASIA PTE. LTD., *Singapore*
EDITORA PRENTICE-HALL DO BRASIL, LTDA., *Rio de Janeiro*

Preface

This book is intended for use in a first course on computer programming and assumes no prior knowledge of computers on the reader's part. It contains a wide range of material, however, and is suitable for either a one- or a two-semester course.

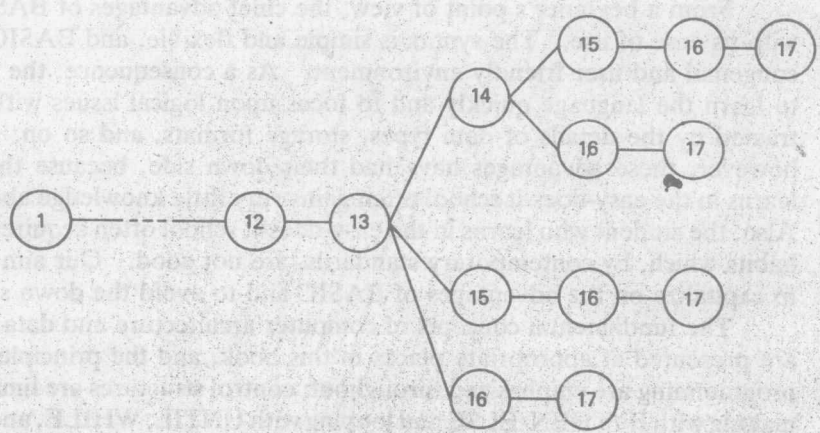
From a beginner's point of view, the chief advantages of BASIC have to do with its ease of use. The syntax is simple and flexible, and BASIC operates in a congenial and user-friendly environment. As a consequence, the student is able to learn the language quickly and to focus upon logical issues without being distracted by the details of data types, storage formats, and so on. Traditionally, however, these advantages have had their down side, because the student who learns in the easy-does-it school often gains very little knowledge about computers. Also, the student who learns in the easy-does-it school often acquires programming habits which, by contemporary standards, are not good. Our aim in this book is to capitalize on the advantages of BASIC and to avoid the down side.

The fundamental concepts of computer architecture and data representation are presented at appropriate places in this book, and the principles of structured programming are emphasized throughout: control structures are limited to decision making with IF/THEN/ELSE and looping with UNTIL, WHILE, and FOR/NEXT. We intentionally postpone any abstract discussion of the concept of structured programming until Chap. 12, however, because it is our belief that such discussions make sense only if the student has already acquired the proper habits through repeated exposure to good examples. Also, the concept of modular programming with subroutines and programmer-defined functions is introduced somewhat late (Chaps. 10 and 11)—not because we regard it as unimportant but because students are not easily sold on the idea of modular programming in BASIC until they know enough to tackle problems of appreciable size.

The book is organized into four main parts. Part I (Chaps. 1 through 6) covers the fundamentals: arithmetic, input and output, data types, and documentation techniques. Part II (Chaps. 7 through 9) deals with control structures: decision making and looping. Part III (Chaps. 10 through 12) covers the principles

of modular programming: modular design, subroutines, and programmer-defined functions. Part IV (Chaps. 13 through 17) deals with specific application areas: array handling, sorting, string handling, and file handling. For a practical reason (namely, to enable students to write interesting programs early), FOR/NEXT loops are introduced ahead of schedule—in Chap. 5.

The core of the subject matter is presented in Chaps. 1 through 12, and it is assumed that all these chapters will be covered by the instructor. Chaps. 1 through 6 can be covered fairly quickly, but Chaps. 7 through 12 might require more time. The material in Chaps. 13 through 17 is somewhat more advanced in nature, and it is not assumed that all these chapters will be covered—especially in a one-semester course. It is assumed that Chap. 13 (on arrays) will be covered, however, because all the subsequent chapters presuppose a knowledge of arrays. Chap. 14 (on sorting and searching) can be regarded as optional, but it is strongly recommended, especially if Chaps. 16 and 17 (on file handling) are to be covered. Chapter 15 (on string handling) also can be considered optional. Chapters 16 and 17 (on file handling) can be regarded as optional, but Chap. 17 presupposes Chap. 16. The most likely paths to be taken through this book, therefore, are those indicated below.



We wish to express our sincere gratitude to the following colleagues, each of whom read our text in manuscript form and offered many valuable suggestions: Mr. Jim Bublitz (Waukesha County Technical Institute), Dr. Edgar B. Lewis (Arizona State University), Dr. Robert I. Matthews (University of Puget Sound), Dr. Ralph A. Morrelli (Trinity College of Hartford, Connecticut), and Dr. Keith B. Olson (Montana College of Mineral Science and Technology). We also wish to convey our thanks to Pacific Lutheran University of Tacoma, Washington, for the use of a VAX-11/750; to Robert Morris College of Chicago, Illinois, for the use of a PDP-11/44; and to Ms. Frances Greenleaf for the use of the telephone modem with which most of the example programs were tested. Finally, special thanks go to Ms. Joan McCulley and Ms. Marcia Horton of Prentice Hall for their kind and professional assistance during the preparation of the manuscript and the production of this book.

Contents

Part 2 CONTROL STRUCTURES

APPENDICES

1

Computers and Computer Programming

Contrary to popular belief, computers are not "smart." In fact, they cannot do anything unless we tell them exactly what to do. The main reason we use computers is that they can follow our instructions very quickly and accurately. They cannot think for themselves.

Moreover, the basic operations that computers can perform are few in number and very simple in nature: Computers can do simple arithmetic. They can compare numbers and determine whether the first is less than or equal to the second. They can move numbers from place to place within their "memories." And so on. Virtually all of the basic actions that computers can perform are as simple as these.

To **program** a computer means to prepare a set of instructions for the computer to follow. The more complicated tasks we often associate with computers, such as preparing paychecks, guiding rockets to the moon, and playing arcade games, are accomplished by programming, or instructing, computers to perform sequences of the simple types of actions described above. Imagine, for example, a person who does not know how to multiply but does know how to add. We could "program" this person to multiply 3×4 by instructing him or her to perform a sequence of additions:

$$3 + 3 + 3 + 3$$

This, by the way, is how some computers multiply.

This textbook is an introduction to the art of programming computers. More specifically, it is an introduction to programming computers in the language called **BASIC**. The first chapter is intended to provide a general knowledge of this art and an understanding of how BASIC fits into the overall landscape of computer programming.

1-1 A MODEL OF A TYPICAL COMPUTER SYSTEM

In this section we shall describe a simple model of a typical computer system. The purpose of this discussion is to give you a general concept of what happens "under the hood" when you use a computer.

In a general way a computer system is similar to a factory. Like a factory, it does the following:

- It receives both raw materials and instructions which tell it what to do with the materials. The materials and the instructions are called the **input** to the system.
- It transforms the materials according to the instructions. This is called **processing**.
- It produces something useful for the outside world. This is called the **output** from the system.

Notice that the **input** consists of both raw materials and instructions. The raw materials are called **data**, and the instructions are called **programs**. The three activities just mentioned—obtaining input, processing data, and producing output—are the three major functions of any computer system. This is depicted in Fig. 1-1.

In a computer system these three activities are handled by physical devices called input devices, the processor, and output devices, respectively. This is depicted in Fig. 1-2. Typical **input devices** include terminal keyboards, card readers, magnetic tape drives, and magnetic disk drives. These devices are used to send data and programs to the processor. Typical **output devices** include printers, television-like cathode ray tubes (CRTs), magnetic tape drives, and magnetic disk drives. These devices are used to receive output from the processor. Note that the same device can be used both for input and for output. Tape drives and disk drives, for example, can be used both to send input to the processor and to receive output from the processor. Dual-purpose devices such as these are often called **input/output devices**.

The **processor** is the heart of the system. This is where the input is sent, where the data are processed, and where the output comes from. A typical processor includes three physically distinct components—a **control unit**, a **memory unit**, and an **arithmetic-logic unit**—as depicted in Fig. 1-3. The **arithmetic-logic unit** consists of circuits that perform arithmetic operations (such as addition and subtraction) and logical operations (such as the comparison of numbers). The **memory unit**—often called **main memory**—consists of circuits that store both instructions (programs) and data. The **control unit** consists of circuits that copy and read the instructions from main memory, one at a time. The control unit is wired in such a way that it responds to these instructions by causing the appropriate actions to take place in other parts of the system. When the control unit reads an instruction to add two numbers, for example, it causes the numbers to be copied

Figure 1-1