

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



© 1988 by Prentice Hall

A Division of Simon & Schuster, Inc.

Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

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*

EDITORIA 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.

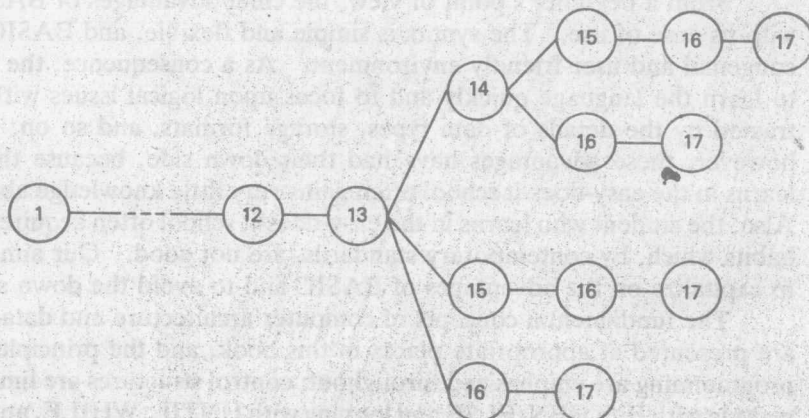
From a beginner's point of view, the chief advantages of BASIC have to do with its ease of use. The syntax is simple and flexible, and BASIC operates in a congenial and user-friendly environment. As a consequence, the student is able to learn the language quickly and to focus upon logical issues without 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

PREFACE xv

Part 1 THE FUNDAMENTALS

Chapter 1 COMPUTERS AND COMPUTER PROGRAMMING 1

- 1-1. A Model of a Typical Computer System 1
- 1-2. Software and Programming Languages 5
 - 1-2A. Machine Language 6
 - 1-2B. Other Low-Level Languages: Assembly Languages 6
 - 1-2C. Further Improvement: High-Level Languages 7
 - 1-2D. Compilers and Interpreters 9
 - 1-2E. Applications Software and Systems Software 10
- 1-3. Survey of High-Level Languages 11
 - 1-3A. FORTRAN 12
 - 1-3B. LISP 12
 - 1-3C. COBOL 12
 - 1-3D. PL/I 13
 - 1-3E. BASIC 13
 - 1-3F. Pascal 13
 - 1-3G. PROLOG 14
 - 1-3H. Ada and MODULA-2 14

| | | |
|------------------|---|-----------|
| 1-4. | Contemporary BASIC | 15 |
| 1-5. | The PDP-11 and VAX-11 Computers | 16 |
| Chapter 2 | GETTING STARTED | 18 |
| 2-1. | Log-On | 18 |
| 2-1A. | Log-On for PDP-11 Users | 18 |
| 2-1B. | Log-On for VAX-11 Users | 20 |
| 2-2. | Log-Off | 22 |
| 2-2A. | Log-Off for PDP-11 Users | 22 |
| 2-2B. | Log-Off for VAX-11 Users | 23 |
| 2-3. | Example: How to Create and Execute a Simple Program | 23 |
| 2-3A. | Entering the Program at Your Keyboard | 23 |
| 2-3B. | Correcting Errors | 26 |
| 2-3C. | The LIST Command | 29 |
| 2-4. | Expressions: Simple and Compound | 30 |
| 2-4A. | Example Program with Compound Expressions | 31 |
| 2-4B. | The Order of Operations | 32 |
| 2-4C. | Parentheses | 35 |
| 2-5. | Variables | 38 |
| 2-5A. | Storing Values in Variables: The LET Statement | 38 |
| 2-5B. | What Is a Variable? | 40 |
| 2-5C. | Printing the Value of a Variable | 41 |
| 2-6. | Finding the Roots of Numbers: Another Use of Parentheses | 42 |
| Chapter 3 | SYSTEM COMMANDS | 52 |
| 3-1. | Workspace and Permanent Storage | 54 |
| 3-2. | The SAVE Command | 54 |

| | | |
|-------|--|----|
| 3-3. | The DIR Command | |
| 3-3A. | The DIR Command for PDP-11 Users | 58 |
| 3-3B. | The DIR Command for VAX-11 Users | 59 |
| 3-4. | The OLD Command | 59 |
| 3-5. | The REPLACE Command | 61 |
| 3-5A. | Effects of the REPLACE Command for PDP-11 Users | 62 |
| 3-5B. | Effects of the REPLACE Command for VAX-11 Users | 63 |
| 3-6. | The RENAME Command | 65 |
| 3-7. | The DCL RENAME Command | 67 |
| 3-8. | The UNSAVE Command | 69 |
| 3-9. | System Command Summary | 70 |
| 3-10. | Common Errors in the Use of System Commands | 72 |

Chapter 4 INPUT AND OUTPUT 78

| | | |
|-------|---|----|
| 4-1. | Introduction to Data Types | 79 |
| 4-1A. | Constants (Numeric and String) | 80 |
| 4-1B. | Variables (Numeric and String) | 82 |
| 4-2. | Extended Variable-Names | 83 |
| 4-3. | The INPUT Statement | 86 |
| 4-3A. | An Example | 86 |
| 4-3B. | Printing with Semicolons | 88 |
| 4-3C. | Including the Prompt within the INPUT Statement | 91 |
| 4-3D. | Using the INPUT Statement for String Data | 91 |
| 4-3E. | Using the INPUT Statement with Multiple Data Items | 93 |
| 4-4. | More on Printing with Semicolons | 94 |
| 4-5. | String versus Numeric Operations | 95 |

| | | |
|----------------------|--|----------------|
| Chapter 5 | ADDITIONAL INPUT AND OUTPUT TECHNIQUES | 104 |
| 5-1. | FOR/NEXT Loops | 104 |
| 5-2. | Input with the READ Statement | 107 |
| 5-3. | Printing with Commas | 114 |
| 5-4. | Printing with PRINT TAB Statements | 116 |
| 5-5. | Printing Numbers with PRINT USING Statements | 121 |
| 5-6. | Printing Tables with PRINT USING Statements | 127 |
| 5-7. | Printing Character Strings with PRINT USING Statements | 130 |
| Chapter 6 | ROUNDING OUT THE FUNDAMENTALS | 147 |
| 6-1. | Numeric Data in More Detail | 147 |
| 6-1A. | Representing Numbers in Decimal Format | 147 |
| 6-1B. | Exponential Format | 148 |
| 6-1C. | More Data Types (Real Numbers and Integers) | 150 |
| 6-1D. | Options Available to VAX-11 Users | 153 |
| 6-2. | Supplied Functions | 154 |
| 6-3. | Rounding | 161 |
| 6-3A. | Rounding to the Nearest Integer | 162 |
| 6-3B. | Rounding to Other Powers of 10 | 164 |
| 6-4. | Program Development Aids | 167 |
| 6-4A. | Pseudocode | 168 |
| 6-4B. | Flowcharts | 172 |
| 6-5. | Program Documentation | 174 |

Part 2 CONTROL STRUCTURES

Chapter 7 DECISION MAKING 184

- 7-1. Decision Making with the IF/THEN/ELSE Statement 184
- 7-2. More about Conditions 186
- 7-3. Formatting Statements for Greater Readability 189
- 7-4. IFs without ELSEs 191
- 7-5. Multiple Instructions with IF Statements 194
- 7-6. Nested Decisions 200
- 7-7. Logical Conditions 212
 - 7-7A. The OR Condition 213
 - 7-7B. The AND Condition 214
 - 7-7C. The NOT Condition 216
- 7-8. Decision Making with the GOTO Statement 217

Chapter 8 LOOPS: THE FUNDAMENTAL TECHNIQUES 234

- 8-1. Counting Loops: Preliminary Examples 235
- 8-2. Controlling Loops with IF Statements and GOTO Statements 240
- 8-3. The UNTIL Loop 246
- 8-4. The WHILE Loop 249
- 8-5. The FOR/NEXT Loop 252
- 8-6. Controlling the Adjustment: The STEP Function 256
- 8-7. Accumulating Totals within a Loop 260
- 8-8. Noncounting Loops 263

**Chapter 9 LOOPS: ADDITIONAL TECHNIQUES
 AND APPLICATIONS 278**

- 9-1. Nested Loops 278
 - 9-1A. Introductory Examples of Nesting 278
 - 9-1B. Another Example: Compound Interest 285
- 9-2. New Methods for Looping with READ/DATA 290
 - 9-2A. The Header Record Method 292
 - 9-2B. The Trailer Record Method 293
- 9-3. Disk Files 298
 - 9-3A. Reading Data from Disk Files 299
 - 9-3B. Writing Data into Disk Files 302
- 9-4. A Useful Application: Summarizing the Data 305

Part 3 STRUCTURED PROGRAMMING

Chapter 10 Modular Program Design 320

- 10-1. An Example 321
- 10-2. High-Level Design 323
- 10-3. Lower-Level Design 324
- 10-4. Review and Revision 327

**Chapter 11 MODULAR PROGRAMMING WITH SUBROUTINES
 AND FUNCTIONS 334**

- 11-1. Fundamentals of Subroutines 335
- 11-2. Nested Subroutines 339
- 11-3. An Application: The Grade Report Program 344
- 11-4. The Grade Report Subroutines 346

- 11-5. Programmer-Defined Functions 350
 - 11-5A. A Rounding Function 351
 - 11-5B. An Averaging Function 354
- 11-6. Multiple-Line Functions 354
- 11-7. Nested Function Definitions 360
- 11-8. The Complete Program 365

Chapter 12 STRUCTURED PROGRAMMING AND MODULAR PROGRAMMING WITH EXAMPLES AND APPLICATIONS 376

- 12-1. Structured Programming 377
- 12-2. Logic Structure and Source-Language Structure 379
- 12-3. Structured Programming and Modular Programming 384
- 12-4. The ON-GOSUB Statement 385
- 12-5. Refining an Interactive Program 391
 - 12-5A. Checking Input 391
 - 12-5B. Appearance of the Program 393
- 12-6. Modifying the Grade Report Program 398
 - 12-6A. The High-Level Logic 400
 - 12-6B. The Details 404
- 12-7. Planning Printed Output 410

Part 4 APPLICATIONS

Chapter 13 ARRAYS 424

- 13-1. Introduction to Arrays 425
 - 13-1A. Subscripted Variables 426
 - 13-1B. Subscripted Variables in BASIC 427

| | | |
|-------------------|---|------------|
| 13-1C. | Using Subscripted Variables | 428 |
| 13-1D. | A Sample Program | 430 |
| 13-1E. | Another Example: Arrays of Strings | 432 |
| 13-2. | The DIM Statement | 434 |
| 13-3. | Magic Numbers | 437 |
| 13-4. | Calculations in Arrays | 440 |
| 13-5. | Searching an Array | 444 |
| 13-6. | Two-Dimensional Arrays | 449 |
| 13-6A. | First Example: A Mileage Program | 452 |
| 13-6B. | Second Example: An Improved Mileage Program | 455 |
| 13-6C. | Third Example: Sales Analysis | 457 |
| 13-6D. | Fourth Example: Improved Sales Analysis | 461 |
| Chapter 14 | SORTING AND SEARCHING TECHNIQUES | 481 |

| | | |
|-------|-----------------------------|-----|
| 14-1. | The Bubble Sort | 482 |
| 14-2. | Refining the Bubble Sort | 487 |
| 14-3. | The Insertion Sort | 490 |
| 14-4. | Refining the Insertion Sort | 496 |
| 14-5. | Sorting Records in Files | 499 |
| 14-6. | Sorting on Multiple Keys | 502 |
| 14-7. | The Binary Search | 505 |

Chapter 15 STRING PROCESSING 516

| | | |
|--------|------------------------------|-----|
| 15-1. | Substring Processing | 517 |
| 15-1A. | The MID Function | 518 |
| 15-1B. | Extraction of Substrings | 521 |
| 15-1C. | The LEFT and RIGHT Functions | 522 |
| 15-1D. | The LEN Function | 522 |

| | | |
|-------------------|--|------------|
| 15-1E. | The INSTR Function | 523 |
| 15-1F. | Things To Know | 526 |
| 15-2. | Conversion Operations | 527 |
| 15-2A. | The VAL Function | 528 |
| 15-2B. | Comparison of Strings | 532 |
| 15-2C. | The NUM1\$ Function | 535 |
| 15-2D. | The CHR\$ and ASCII Functions | 536 |
| 15-2E. | The CVT\$\$ Function | 540 |
| 15-3. | String Arithmetic | 542 |
| 15-3A. | The SUM\$ Function | 542 |
| 15-3B. | The DIF\$ Function | 543 |
| 15-3C. | The PROD\$ Function | 543 |
| 15-3D. | The QUO\$ Function | 543 |
| 15-4. | Compound Logical Conditions | 544 |
| Chapter 16 | SEQUENTIAL FILE PROCESSING | 556 |
| 16-1. | Sequence Checking | 557 |
| 16-2. | Control Breaks | 559 |
| 16-2A. | Single-Level Control Breaks | 561 |
| 16-2B. | Double-Level Control Breaks | 564 |
| 16-2C. | Control Headings | 570 |
| 16-3. | File Merges | 574 |
| 16-4. | Master File Updates | 583 |
| 16-5. | Master File Updates with Insertions | 590 |
| Chapter 17 | DIRECT ACCESS FILE PROCESSING | 604 |
| 17-1. | Introduction to Direct Access Files | 605 |
| 17-2. | Relative Files in VAX BASIC | 607 |
| 17-2A. | The MAP and OPEN Statements | 607 |
| 17-2B. | Writing Records into a Relative File with the PUT Statement | 609 |
| 17-2C. | Reading Records from a Relative File with the GET Statement | 610 |

| | | |
|--------|--|-----|
| 17-2D. | Initializing a Relative File | 611 |
| 17-2E. | Writing Records with the UPDATE Statement | 612 |
| 17-2F. | The Program | 615 |
| 17-3. | Block I/O Files in BASIC-PLUS | 621 |
| 17-3A. | The OPEN and FIELD Statements | 621 |
| 17-3B. | Writing Records into a Block I/O File with the PUT Statement | 623 |
| 17-3C. | Reading Records from a Block I/O File with the GET Statement | 625 |
| 17-3D. | Initializing a Block I/O File | 626 |
| 17-3E. | Putting Records into an Initialized File | 628 |
| 17-3F. | The Program | 629 |
| 17-3G. | Processing Numeric Data in the I/O Buffer | 633 |
| 17-4. | Hashing | 636 |
| 17-4A. | Sequential Processing of a Hashed File | 638 |
| 17-4B. | Collisions | 639 |
| 17-4C. | Implications | 641 |

APPENDICES

| | | |
|------------|-------------|-----|
| Appendix A | ASCII CODES | 647 |
|------------|-------------|-----|

| | | |
|------------|----------|-----|
| Appendix B | KEYWORDS | 648 |
|------------|----------|-----|

| | |
|------------|-----|
| BASIC-PLUS | 648 |
|------------|-----|

| | |
|-----------|-----|
| VAX BASIC | 649 |
|-----------|-----|

| | |
|-------|-----|
| INDEX | 653 |
|-------|-----|

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

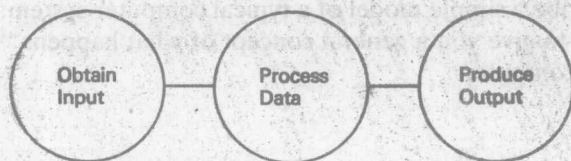


Figure 1-1