BASIC

An Introduction to Computer Programming Using the BASIC Language

Third Edition

William F. Sharpe and Nancy L. Jacob

BASIC

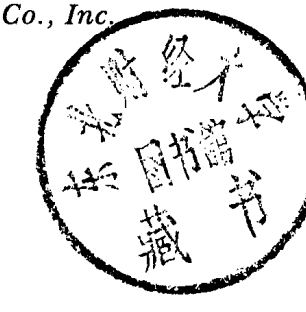
An Introduction to Computer Programming Using the BASIC Language

Third Edition

William F. Sharpe
Nancy L. Jacob

THE FREE PRESS
A Division of Macmillan Publishing Co., Inc.
NEW YORK

Collier Macmillan Publishers
LONDON



The Free Press
A Division of Macmillan Publishing Co., Inc.
866 Third Avenue, New York, N. Y. 10022

Collier Macmillan Canada, Ltd.

Library of Congress Catalog Card Number: 78-72148

Printed in the United States of America

printing number

6 7 8 9 10

Library of Congress Cataloging in Publication Data

Sharpe, William F
BASIC : an introduction to computer programming using the BASIC language.

Includes index.
1. Basic (Computer program language)
I. Jacob, Nancy L., joint author. II. Title.
QA76.73.B3S45 1979 001.6'424 78-72148
ISBN 0-02-928380-9
ISBN 0-02-928390-6 **pbk.**

BASIC

Preface

In recent years the electronic digital computer has been transformed from a device understood by members of a small cult of worshipers to an indispensable part of the life of every student, scientist, and businessman. Universities have taken account of its importance with courses on computers, systems analysis, computer programming, and more esoteric aspects of the field now known as computer science.

The key to an understanding of computers, and an essential skill for using them, is the knowledge of at least one computer programming language. There are many candidates. Some were developed when computers were both far more expensive and far less sophisticated than they are now. Such languages reduce the effort expended by the computer, but they often require substantial expenditures of time and effort by the user. Other languages take account of current technology and costs but were designed for limited classes of problems. Still others were designed for professional programmers willing and able to learn and remember rather complex sets of rules and restrictions.

This book describes a language that combines a number of desirable features with relatively few undesirable ones. The language is BASIC, developed at Dartmouth College under the direction of Professor J. G. Kemeny. It was designed for use with one of the first computer systems that allowed a number of people concurrently to communicate directly with the machine, using typewriter-like consoles. Since then, such *time-shared* computers have become commonplace. Many are available commercially, with costs as low as a few dollars per hour of use. It is literally true that a computer of this type is as near as the nearest telephone, because communication by means of standard tele-

phone lines is the normal procedure for must commercial systems and many time-shared computers used by universities and private firms.

Recent advances in technology have led to the production of a number of relatively low cost desk-top computers, many of which can be programmed using the BASIC language. It is likely that past trends will continue; the day of the inexpensive hand-held computer that can be programmed in BASIC may not be far off.

BASIC is the major language used with many time-shared computer systems and desk-top computers. But it is also found in more conventional environments, with the user preparing instructions and data on punched cards and submitting them for processing as part of a batch of such jobs at a central computer facility.

This book is designed to allow the user to think of a computer as if the latter "understands" the BASIC language. Thus no details of any of the supporting systems are included. This makes the book particularly suited for those unable to use a computer system, because it allows them to concentrate on the computational and logical properties of a high-level computer language with relatively little concern for clerical details.

Very little is said about computers per se; the interested student is referred to any of the almost countless introductory texts or manuals on the subject.

BASIC can be approached on two levels—first, as a simple, yet powerful language for standard numeric processing; then as a rich language for efficiently describing a wide range of advanced applications involving both the manipulation of numbers and the manipulation of textual material. This book is organized to reflect the dichotomy. Part I provides a full description of the essential features of the language. It should suffice for those persons who are interested in obtaining an appreciation of computers, the ability to use them in their academic work, and/or sufficient understanding to communicate with computer people in later life. Part II is designed for those persons with deeper interests and/or the need to use computers for more complex tasks. It introduces procedures particularly useful to the mathematician or scientist (for example, matrix commands) as well as those particularly useful to the business student or social scientist (for example, string manipulation).

The average college student should be able to learn the material in part I with five to ten hours of classroom exposure and an equal investment in time spent preparing actual programs. In many instances it may be both feasible and desirable to forego some or all of the classroom time, relying instead on this book and the availability of helpful advice when the student encounters problems or questions. In the final analysis, the computer itself may be the best teacher.

An important feature of this edition is its conformance with the proposed American National Standard for Minimal BASIC.* Unless otherwise noted, all the material in part I is consistent with the specifications in this important document. Most computer implementations of the language conform closely to the standard, minimizing the changes required to employ the procedures described in part I on any particular machine.

At present there is no set of standards for extended elements of BASIC. In part II we present representative examples of such extensions. Although modifications may be required for some computer systems, the overall concepts are quite general. To the extent possible, we have tried to anticipate future standards and insure that the material is consistent with present standards for Minimal BASIC.

We believe that BASIC represents an excellent first (or only) computer language. It includes many of the logical and conceptual components found in other programming languages, while requiring a minimum of attention to housekeeping details. Moreover, students who have learned BASIC have little difficulty with other languages. The marketplace has provided ample evidence of the usefulness of BASIC. We know no better test.

*May 1977, American National Standards Institute, 1430 Broadway, New York 10018.

Contents

PART I

Minimal BASIC

CHAPTER 1

Introduction

The Computer and You

Imagine that you have a diligent, hard-working, and accurate, but totally unimaginative, clerical assistant. Since the assistant is so unimaginative, it is necessary for you to provide very precise sets of instructions (what to do) as well as data (what to do it to). To avoid any problems, you and the assistant have agreed upon a rather limited language—vocabulary and grammar—for stating your instructions. The language has the virtue that it admits no ambiguity. If you follow the rules when stating your desires, the assistant will do precisely what you have in mind. If any mistakes are made, they will necessarily be yours.

Now substitute *computer* for *clerical assistant.* Call the set of instructions a *program*. Call the language BASIC. Otherwise everything is the same.

The object of this book is to teach the grammar and vocabulary of BASIC. Although no computer actually "understands" BASIC directly, many computers have been taught (preconditioned) to act as if they do. The manner in which this was done need not concern the reader; for all practical purposes, one can assume that such computers understand BASIC.

Communication

To make a computer do your work, you must provide it with a *program* (indicating what to do), *data* (things to be processed), and certain information required to identify you, to tell the computer how important you are, where to send the bill, and so forth. All of this must be sent to the computer somehow,

and the computer must return information as well. Many installations provide users with typewriter-like consoles that can communicate with the computer either directly or over standard telephone lines. Others require that information be input from punched cards; output is then returned later on printed sheets. In some cases a user can communicate directly with a computer (usually a desk-top model) devoted entirely to his or her task. Detailed information about such matters obviously must be obtained from those with knowledge of the particular computer to be utilized. This book deals with the more essential and more general aspects—the writing of programs and the arrangement of data.

A program consists of a series of statements (instructions or commands); each is written on a separate line. The lines are numbered from top to bottom, with smaller numbers preceding larger numbers. Systems that allow the user to enter lines directly keep the lines in correct numeric sequence, even if they are not entered in order. When using punched-card input, the user may have to arrange the cards correctly himself, with lower-numbered cards preceding higher-numbered cards.

Data are prepared in a similar fashion. Each line containing data is numbered, and the set of lines is arranged in sequence, either automatically (for console entry systems) or by the user (for punched-card systems).

Nothing will be said here about operating teletypewriters, keypunch machines, desk-top computers, and so on. Such information is easy to obtain; the best method is simply to spend five minutes at a machine with someone who knows how to operate it.

Diagnostic Messages

As we have suggested, you can assume that the computer recognizes instructions written in the BASIC language. But what if you present it with an illegal instruction (i.e., one that violates the rules for grammar and vocabulary presented here)? In most cases the computer will be aware that it does not understand the instruction and make a reasonably well informed guess about the source of its (more properly, your) confusion. And it will tell you about its difficulty and provide its diagnosis of the problem. Don't be embarrassed by such *diagnostic messages;* most programmers learn more from them than from manuals.

Unfortunately, the computer can detect only errors arising from illegal vocabulary and/or grammar; it cannot read your mind. If your program is constructed according to the rules, the computer will happily do precisely what you tell it to do. It is up to you to make certain that what you tell it to do is what you want it to do.

Output

When you give a program and data to a computer, one or two operations take place. First, the computer looks over your program. If serious errors are found, it tells you about them (with alarming candor, on occasion), and then it refuses to have anything more to do with you until you correct the errors. On the other hand, if it finds your program acceptable, it meekly begins to follow your instructions, looking at your data when told to do so and providing answers in accordance with your instructions.

CHAPTER 2

Getting Started

Here are an extremely simple program and set of data.

Program

```
 10   REM -- PAYROLL PROGRAM
 20   REM -- PROGRAMMER, WALTER P. BUNCZAK
 30   READ P
 35   READ H
 45   LET G = P * H
 60   LET W = .14 * G
 70   LET N = G - W
 80   PRINT P
 90   PRINT H
100   PRINT G
110   PRINT W
120   PRINT N
140   GO TO 30
```

Data

```
900   DATA  2.25
901   DATA 40
902   DATA  3.00
903   DATA 41
904   DATA  2.97
905   DATA 35
906   DATA  3.10
907   DATA 49
999   END
```

Format

To make a program easy to read we often insert spaces within statements. To avoid ambiguity, some separation is required. For example, key words such as READ, PRINT, and LET should be separated from numbers, letters, etc. by at least one space.[1] Moreover, such words should be typed without any intermediate spaces. So should numbers. A good rule is to simply do what comes naturally.

You may have to get used to capital letters. There are no lower-case letters on some input devices, and even if you can type them in, the computer may treat them as upper-case and type them back to you in that form. The major exception concerns letters typed between quotation marks—in such instances if you can type in lower-case letters, the computer will probably leave them alone.

Line Numbers

Notice that each statement in the program has a line number and that the numbers are arranged in order. Line numbers are required and should be between 1 and 9999. Only integers—whole numbers—are allowed. It is a good idea to leave gaps when assigning numbers (e.g., writing 10, 20, 30, and so on) in case you subsequently wish to insert additional statements.

Remarks

Every statement must begin with a legal command (after the line number). The first two commands in the program at the beginning of this chapter are remarks. A remark is used to provide information for you and/or anyone else reading your program; it provides no information to the computer. In fact, the computer ignores remarks (saying, in effect, "He or she is only talking to himself or herself, not to me"). To indicate a remark, simply use the command REM; after that you may write anything you please.

1. Key words in Minimal BASIC (all of which are described in this book) are: BASE, DATA, DEF, DIM, END, FOR, GO, GOSUB, GOTO, IF, INPUT, LET, NEXT, ON, OPTION, PRINT, RANDOMIZE, READ, REM, RESTORE, RETURN, STEP, STOP, SUB, THEN, and TO.

Instruction Sequence

A program is nothing more than a set of instructions (although remarks are instructions only in a rather academic sense). The computer is expected to follow the instructions in a particular order. If you are entering your program from a teletype or other remote terminal, the computer will execute your instructions sequentially by line number unless told to do otherwise. In other words, it will first execute the statement having the lowest line number, then the statement having the next higher line number, and so on, unless the statements themselves tell the computer to deviate from that order. (The instruction in line 140 of our sample program does just that, as you will see.) Only *one* statement can be executed at a time. All this means you must be careful to assign line numbers that accurately reflect the order in which you wish the statements in your program executed. And the order of execution is the main reason why it is important to obtain a listing of your program with the statements appearing in order by line number (as has been done with all the programs in this book). It makes the program much easier for humans to follow, since execution will proceed from the top of the page to the bottom.

Variables

The computer is provided with a number of "mailboxes," each of which can hold a number. Each mailbox has a name. There are twenty-six mailboxes with simple one-letter names: A, B, C, . . . , Z. Some other mailboxes have two-character names—a letter followed by a digit: A0, A1, . . . , A9, B0, . . . , B9, C0, . . . , Z9.

For convenience we often use the name of a mailbox to indicate the number in it. And because the number in a mailbox may be taken out and a new one put in its place, we often refer to the number in a mailbox as a variable (because it may vary as the program is executed). Thus variable A means the number currently in mailbox A; variable B3 means the number currently in mailbox B3.

Reading Data

The data on which a program operates often consist of a set of numbers. It is useful to think of the numbers as if they were in a list, with one item per row. At any given time there is a pointer indicating the next item to be read. Initially, it is set to point at the first item on the list. As the program proceeds, whenever one piece of data is used (read), the pointer moves down one item.