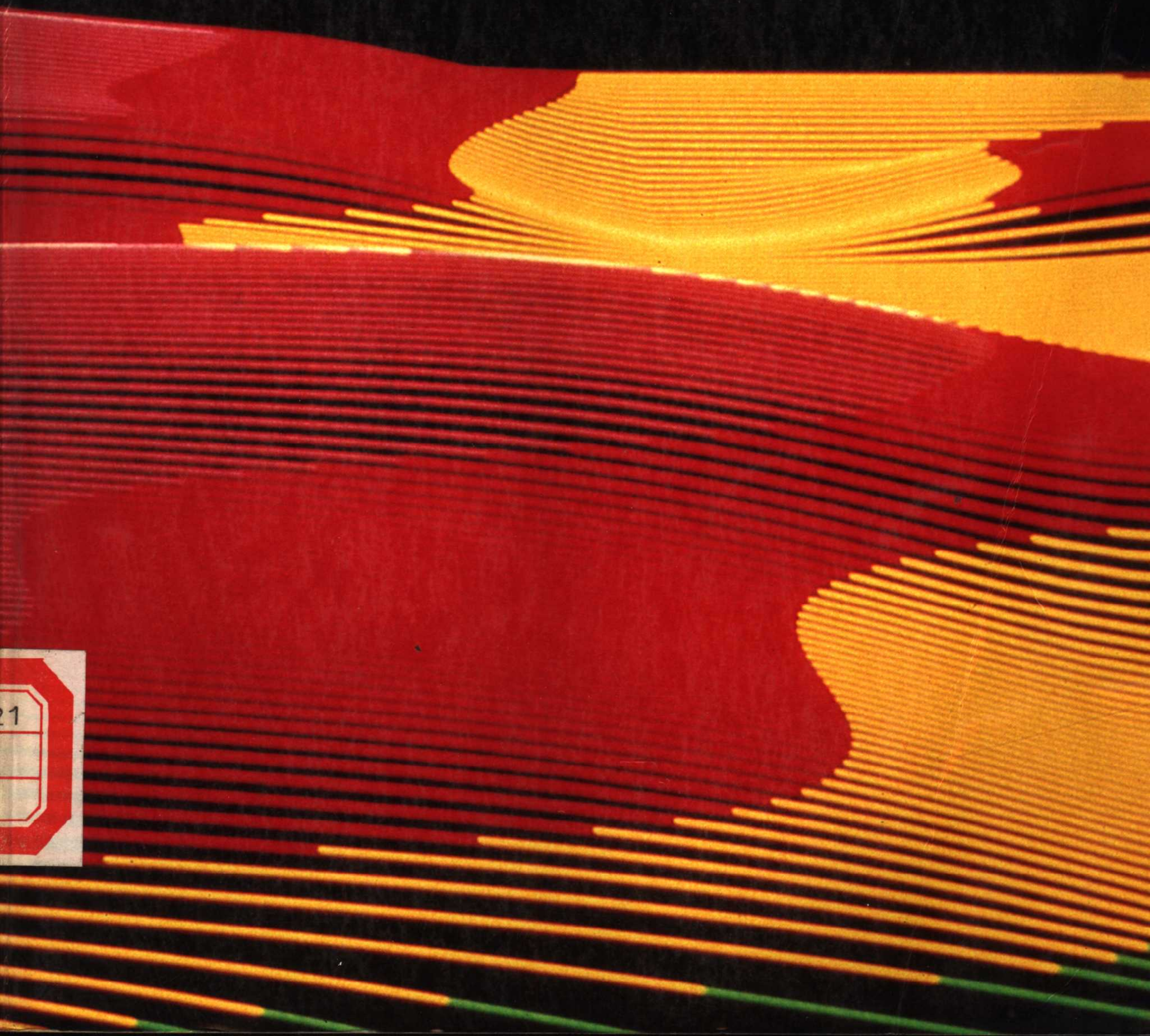
the basics of BASIC

Alfredo C. Gomez



the basics of BASIC

Alfredo C. Gomez

Broward Community College
Fort Lauderdale, Florida

Digital Data Systems, Inc.
Fort Lauderdale, Florida

Holt, Rinehart and Winston

New York • Chicago • San Francisco • Philadelphia
Montreal • Toronto • London • Sydney • Tokyo
Mexico City • Rio de Janeiro • Madrid

The computer-generated art shown on the cover was supplied by Dr. Melvin L. Prueitt, Los Alamos National Laboratory. It is a summation of exponential functions, and the colors denote altitudes.

Address correspondence to:
383 Madison Avenue, New York, NY 10017

Library of Congress Cataloging in Publication Data

Gomez, Alfredo, 1939–
The basics of BASIC.

Includes index.
1. Basic (Computer program language). I. Title.
QA76.73.B3G645 1983 001.64'24 83-8431

ISBN 0-03-063069-X

Printed in the United States of America

Published simultaneously in Canada

3 4 5 6 039 9 8 7 6 5 4 3 2 1

CBS COLLEGE PUBLISHING
Holt, Rinehart and Winston
The Dryden Press
Saunders College Publishing

the basics of BASIC

To my father and mother
who made it possible

Preface

The question is often asked: Why another BASIC programming book? This text meets the needs of college students and professionals by providing a comprehensive, machine-independent manual. Most of the programming examples are business oriented, but some math examples have been used to illustrate specific concepts.

The book contains a large number of programming examples. A two-step technique has been used to illustrate each concept. Following one or two simple programs a more realistic, longer program is given. This allows the reader to become familiar with practical programs.

Each BASIC statement or programming concept is first illustrated using the most elementary form of BASIC. However, what can be done with more advanced versions of BASIC is also explained.

Chapter 4 covers an introduction to the programming process, flowcharts, and structured programming. I feel these concepts introduced early in the book are of considerable value. Flowcharts and structured programming techniques are illustrated throughout the book.

A question and answer section follows most chapters. The purpose of these sections is to cover some of the more subtle items that often come up when learning programming. The questions are based on the questions students normally ask in my classes.

Realizing the importance of file handling, a separate chapter has been included. Chapter 11 includes examples of file handling in three major versions of BASIC: BASIC PLUS by Digital Equipment Corporation, TRS-80 Level II by Radio Shack, and AppleSoft by Apple Computers.

For those students who have never taken a course in introductory data processing, Appendix A covers most of the elementary concepts that are needed prior to taking a course in BASIC programming. The appendixes also include a summary of all BASIC statements and a description of several of the most popular versions of BASIC.

Alfredo C. Gomez

Contents

1

Getting Started in BASIC

The earlier computer programming languages are somewhat difficult to learn and apply. As the use of computers became more widespread, it was decided that a new simple language was needed. This formed the basis for BASIC, an acronym for Beginner's All-Purpose Symbolic Instruction Code. BASIC's very simple grammatical rules, wide availability, and the fact that it can be learned in a short period of time have made it a very popular programming language.

BASIC is a general purpose language equally suitable for numerical and non-numerical applications in business and science. Most computer manufacturers provide BASIC for their machines and systems. In the microcomputer field BASIC is almost universally used. This widespread use has almost made it required knowledge for anyone in the computational fields in business or science.

Statement Numbers

A program in BASIC consists of a series of statements. These statements, which can be one line or more in length, are individual instructions. Each statement in the program must start with a statement number. These numbers have two purposes:

1. The computer uses them to assign a memory location for the statement.
2. The numbers tell the computer the sequence of the program statements, which are executed in numerical order.

A short program that illustrates the statement numbers is shown below:

```
10 REM A BASIC PROGRAM TO ADD TWO NUMBERS
20 READ A,B
30 LET C=A+B
40 PRINT C
50 DATA 2,3
60 END
```

Note that in this program each individual statement is numbered, and the numbers go from low to high. In most versions of BASIC the statement numbers can go from 1 to 99999. Note that in the example the first line was numbered 10, the second one was numbered 20, the third one 30, etc. They were not numbered 1, 2, 3, 4, 5 because quite often programmers change their minds and decide to insert new statements in between the ones that have already been written. Line numbering by one does not allow changes or additions to the program. It is strongly recommended that the lines be numbered to allow spaces in between so that additions can be included. For example, if the statement numbers were 10, 20, 30, 40, and 50, then an additional statement number could be added between existing lines. As an example, 45 would be executed right after 40 and just before 50. Statement numbers must be unsigned integer constants.

As the program gets more complex, programmers like to increment by 100. It is common to see programs in which the lines are numbered 100, 200, 300, 400, 500, etc. In this manner the programmer can make substantial changes between lines.

The BASIC Character Set

The characters permissible in BASIC are grouped into three types:

1. *Numeric*. The numeric characters are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
2. *Alphabetic*. The alphabetic characters are A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, and Z.
3. *Special characters*. The special characters are . , ; " + – * / = ( ) > < ↑ $, and the blank character, which is equivalent to a space on the keyboard. Some versions of BASIC use other special characters.

Constants

A constant is a value that does not change; its numerical value is fixed and stated explicitly. Constants appear in many computations. Plus or minus signs as well as decimal points may be used in conjunction with constants.

Some examples of valid constants are

```
250     -8.56
78549   +0.44
-25
```

Imbedded blanks between the first and last digit do not affect the value of the constant; as an example,

```
832 =  8    32 =  8    3    2
```

Some examples of invalid constants are

	Reason
$50.68	$ is not permitted
65,860	No commas are allowed
689.69	Only one decimal point is allowed
123-46-6492	- is invalid

Variables

Variables, unlike constants, may assume different values. In almost every BASIC program variable names need to be used since the values associated with a name can be different every time new data is made available.

In the simplest form of BASIC, the naming of variables is very restricted. The most elementary form of BASIC only allows a single letter or a letter followed by a digit. A maximum of two characters is allowed, and no special characters are permitted. Some examples of legal variable names are

A	B	X	Y
B3	Z8	M4	P6

Some examples of unacceptable variable names are

	Reason
B55	Only a single digit is allowed
$ X	No special characters are allowed
A -	No special characters are allowed
5B	The letter must precede the digit
FORCE	Only a single alphabetic is allowed

With these rules only 286 possible distinct variable names can be obtained since there are 26 single letters plus 10×26 combinations of letters followed by the numbers 0 through 9.

Most BASIC compilers allow variables to have many more characters than described previously. In fact it is common to find that variable names can be as long as 29 characters. In this manner variable names can be more descriptive of what they represent. It is suggested that the student check the manual of the system being used in order to determine the rules for naming variables.

Expressions

An expression is a combination of constants and variables linked by arithmetic operation symbols. Expressions are used to perform arithmetic operations. Parentheses may be included to perform certain groups of operations as an entity. The allowed symbols are

+	Addition
−	Subtraction
*	Multiplication
↑	Exponentiation or raising to a power (** in some systems)
/	Division
(	Left parentheses
)	Right parentheses

Some examples of valid expressions are

Operation	BASIC
ab	`A*B`
$\frac{ab}{c}$	`A*B/C`
$a + bc$	`A+B*C`
$\frac{a+b}{c+d}$	`(A+B)/(C+D)`
$(a + b)^2$	`(A+B)↑2`

When parentheses are not present, operations are performed according to the following priority:

1. Exponentiations
2. Multiplications and divisions from left to right
3. Additions and subtractions from left to right

Parentheses are cleared first, then operations with the higher precedent are performed before operations with lower precedent. If the operations have equal priority, they are executed from left to right.

Input-Output Statements

Before operations can be performed, the input data to be processed must be read into the computer; this is done by means of input statements such as READ and INPUT. Data can also be entered by an assignment statement such as LET.

After the input data has been entered into the computer and the operations performed, the output results have to be printed onto an output device such as a printer or cathode-ray tube (CRT). This is done by means of the PRINT statement. An example of a complete BASIC program with READ, DATA, LET, and PRINT statements is shown in Figure 1.1. A more detailed analysis of input-output statements will be given in Chapter 3. However, the reader should follow the program shown in Figure 1.1. In statement 10 of this program, two variables are read, A and B. The variables are read from DATA statement 40, which contains the values of A and B, which in this case are 2 and 3. When the READ statement is used in order to input data, the DATA must also be included in the program, and the DATA statement must contain the values of the variables that are being read. After statement 10 is executed, the value of A becomes equal to 2 and the value of B becomes equal to 3. In statement 20 the value of a third variable C is calculated as a sum of A and B. At this point, A, B, and C are defined and available for future processing. In statement 30 the values of A, B, and C are printed on the output device. Note that the output appears under the program listing and contains the values of A, B, and C.

In the next example, two variables are read by statement 10, A and B, which become 10 and 2, respectively. In statement 20 a third variable C is calculated as equal to A divided by B. In statment 30, variable D is calculated as equal to A times B. In statement 40 the program asks for the value of C and D to be printed. The computer responds and prints the value of C and D following the listing.

Strings

In BASIC programs the data is not always numerical in nature. Often the programmer needs to read a series of alphabetic and special characters representing names, addresses, social security numbers, etc. These are called character strings and are used to produce output reports, to generate labels and headings, or to print messages reflecting a particular set of conditions.

```
10 READ A,B
20 LET C = A+B
30 PRINT A,B,C
40 DATA 2,3
50 END

 2             3             5

10 READ A,B
20 LET C = A/B
30 LET D = A*B
40 PRINT C,D
50 DATA 10,2
60 END

 5             20
```

Figure 1.1

Character strings or alphanumeric data are defined by a series of letters, numbers, and special characters in a string. Some examples of character strings are

"JOHN SMITH"
"7440 NW 15TH ST"
"123-48-7744"
"JACK"
"PAYROLL REPORT"

In most cases character strings are enclosed in quotes. The maximum number of characters allowed in a string varies from system to system, but typically is 256 characters.

In most systems when variable names are used to denote strings rather than numbers, the dollar sign ($) is placed after the variable name. As an example, A$, B$, C5$, Z9$ are valid character string variable names.

The REM Statement

The REM statement contains descriptive comments or remarks used to aid the programmer in identifying or documenting the purpose of each section of the program. REM statements are ignored by the compiler or interpreter during translation and only become part of the program listing.

The programmer should use the REM statement wisely, starting sections of the program that need to be clarified with concise descriptions or titles. Excessive use of REM statements can be damaging, since then it becomes difficult to separate the REM statements from the real program. REM statements can clutter the program and make it difficult for the programmer to read. This difficulty can be overcome by setting off comments with asterisks.

All programs should start with REM statements indicating the name of the program, defining some of the variable names in a more descriptive manner, indicating the name of the programmer and the date the program was written, as well as other appropriate information. Keep in mind that REM statements only appear in the program listing and have nothing to do with the output of the program. A good rule of thumb in the use of REM statements is to use them whenever a new part of the program begins. Some examples of valid REM statements are

```
10 REM PROBLEM 1
10 REM *** JOHN SMITH PB 1 PROGRAMMING 101***
10 REM *** STATEMENTS 300-400 COMPUTE DEPRECIATION ***
```

END and STOP Statements

Most BASIC programs must end with an END statement. The END statement is the last BASIC statement in any program and should always be the highest numbered statement.

If any other statements are typed beyond the END statement, they may be ignored by the compiler. Some versions of BASIC do not require an END statement.

The STOP statement indicates where in the program the programmer wants to stop processing. There may be more than one STOP statement in any one BASIC program. The STOP statement can be considered as transferring control to the END statement.

The general forms of the END and STOP statements are

statement number END
statement number STOP

Figure 1.2 shows several short BASIC programs that illustrate the use of the REM, LET, READ, DATA, and PRINT statements. The reader should analyze the examples carefully.

Some Important BASIC Commands

BASIC commands are not the same as statements. Statements have a statement number and are written within a program, whereas commands are not accompanied by a number

```
]LIST

10  REM  PERFORMS ARITHMETIC CALCULATION
20  READ A,B,C
30 D = (A + B) / (C + D)
40  PRINT A,B,D
50  DATA  2,5,6,3
60  END

]RUN
2              5              1.16666667


10 REM READS TWO STRING VARIABLES
20 READ A$,B$,C,D1
30 M = C*D1
40 PRINT A$,B$,C,M
50 DATA JOHN,SMITH,38,5.3
60 END

 JOHN          SMITH          38             201.4


]LIST

10  REM  PERFORMS ARITHMETIC CALCULATION
20  READ A,B,C
30 D = (A + B) / (A + C)
40  PRINT A,B,D
50  DATA  4,2,1
60  END

]

]RUN
4              2              1.2
```

Figure 1.2