BASIC Programming: A Structured Approach

Clark • Drum

15.8/24
C593

BASIC Programming:
A Structured Approach

James F. Clark

William O. Drum

Published by

J78 SOUTH-WESTERN PUBLISHING CO.

CINCINNATI WEST CHICAGO, ILL. DALLAS PELHAM MANOR, N.Y. PALO ALTO, CALIF.

ISBN: 0–538–10780–4
Library of Congress Catalog Card Number: 82–62636

1 2 3 4 5 6 7 8 Ki 0 9 8 7 6 5 4 3

Printed in the United States of America

Preface

Computers have become indispensable tools for businesses, professional people, and governments. Computer programming ability has thus become vastly more important, both as a career opportunity and for personal use. The advent of microcomputers has resulted in a literal explosion of computers using the BASIC language.

This text teaches the BASIC language, with an emphasis on programming business applications. The great majority of career opportunities in programming are in business. Several features of the text make the learning of programming easier.

All programming principles are presented in the context of practical applications. Each chapter emphasizes one idea. Beginning with Chapter 4, each chapter is divided into two topics. Topic I presents general principles—principles that apply regardless of the computer language being used. Topic II explains how to apply those principles using the BASIC language. As each BASIC keyword is presented, its general form and an example are given. Each chapter contains an example program using newly presented keywords. This helps integrate the new keywords with previously learned material. Each topic includes review questions. At the end of each chapter are lists of vocabulary words and BASIC keywords that were presented.

The computer diskette to accompany *Structured BASIC* contains a computer-driven tutorial that dynamically illustrates the operation of each BASIC keyword. This tutorial makes hard-to-grasp keywords much easier to understand. Each keyword is used in a program statement that remains on the screen while the output of the statement is displayed. The arguments of the keyword are changed on the screen, with the effects of each change appearing immediately in the display.

Each chapter includes four programming assignments. Assignment 1 is the easiest and can generally be completed by reference to similar examples in the chapter. Assignments 2 and 3 require the progressive application of creativity. Assignment 4 is an enrichment assignment, usually presenting new information and requiring a still higher level of output.

Proper documentation is emphasized throughout the text. The use of structured programming techniques is developed without the use of burdensome jargon. Hierarchy charts, program documentation sheets, and module documentation sheets are used instead of flowcharts. These forms of documentation are much easier to understand and use, and lead more quickly to error-free programs. Their use makes documentation much less a chore and, therefore, much more likely to be done.

Four end-of-the-book items are handy references once particular programming ideas have been learned. A glossary contains all new vocabulary words presented in the text, making review much easier. A quick guide to all BASIC keywords is included. This guide is a handy reference for refresher purposes once the keyword has been learned. Each entry in the reference guide refers to the page in the text that will give full information. An ASCII code table is convenient for use when writing programs that do character manipulation. An appendix gives suggestions for debugging programs.

The computer diskette contains a computer driven tutorial that dynamically illustrates the operation of each BASIC keyword. This tutorial makes hard-to-grasp keywords much easier to understand. The keyword is used in a program statement that remains on the screen while the output of the statement is displayed. The arguments of the keyword are changed on the screen, with the effects of each change appearing immediately in the display.

The versions of BASIC used by four popular microcomputers are emphasized throughout the text. Differences between computers are pointed out in each chapter and alternate methods of programming are presented where necessary. There is never any doubt as to which keywords apply to the particular computer and version of BASIC being used. The four computers covered are the Apple II[1], Commodore[2], IBM Personal Computer[3], and Radio Shack TRS–80 Models I and III[4]. All items that apply to the IBM Personal Computer are valid with other computers using Microsoft BASIC, version 5.x. Most IBM items also apply to the Radio Shack TRS–80 Model II. All items applying to the TRS–80 Model I are valid with most other computers using version 4.x Microsoft BASIC.

[1]Apple II and Applesoft are trademarks of the Apple Computer Corporation. Any reference to Apple II or Applesoft refers to this footnote.

[2]Commodore is a trademark of Commodore Business Machines, Inc. Any reference to the Commodore microcomputer refers to this footnote.

[3]IBM Personal Computer is a trademark of International Business Machines. Any reference to IBM or the IBM Personal Computer refers to this footnote.

[4]TRS–80 is a trademark of the Radio Shack Division of Tandy Corporation. Any reference to the TRS–80 or the Radio Shack microcomputer refers to this footnote.

Contents

Chapter 1

Getting Started in BASIC

Objectives:
1. Describe the computer and its functions.
2. Define computer program.
3. Write a simple BASIC program using the keywords REM, PRINT, and END.
4. Use commands NEW, RUN, and LIST.
5. Add, delete, and change program lines.
6. Obtain hardcopy output from the line printer.
7. Save and load programs.

Topic 1.1— INTRODUCTION TO COMPUTERS AND PROGRAMMING

One of the greatest electronic marvels of the century is the computer. Since 1945 when the first electronic computer made its appearance, technology has advanced rapidly. Perhaps no other invention has touched the lives of so many people. The size of computers continues to decrease as they work more rapidly, become less expensive, and are easier to use. Computers are in use everywhere— in businesses, schools, and homes.

WHAT IS A COMPUTER?

A computer is an information processing machine that can accept data, make comparisons, and perform calculations. Since it is electronic, it works accurately and tirelessly at high rates of speed. Computers process data that are often referred to as input. Input (raw facts, numbers, characters, etc.) is entered into the computer and stored in its memory. The input is processed and becomes output. The output is referred to as processed information. This information can be displayed on a screen, printed, or stored for future use. For example, a student's test scores during a given time period would be considered as input if they were to be entered into a computer. If they were added together and averaged, the average would be the output and considered processed information.

WHAT IS A PROGRAM?

Although many people think the computer has "intelligence," it can really do nothing without being directed step by step. These step-by-step instructions are referred to as a program. Programs are written by persons known as programmers.

Programs are usually keyed in (typed) from a keyboard and "remembered" by the computer. The computer can then carry out the instructions quickly once they have all been entered. If the computer is equipped with a storage device, the programs may be stored for future use. Storage devices are electronic units that can write data on magnetic tape or a magnetic disk. The data or programs stored on these units are called back into the computer when needed.

A computer can only "understand" programs written in machine language. Programs written in machine language contain instructions using codes. These codes have a special meaning to the computer's electronic circuitry. Writing programs in machine language is very difficult and time-consuming for the programmer. To avoid these problems, programmers generally use high-level languages. High-level languages use English-like instructions that the computer translates into machine language. An interpreter or compiler program completes the translation process. These translator programs are usually supplied by the manufacturer of the computer.

The language called BASIC (Beginner's All-purpose Symbolic Instruction Code) is a high-level language. BASIC can be used to write programs for almost all small computers as well as for many larger ones. BASIC resembles the English language. It uses certain English words called keywords. Keywords are words that have a special meaning to the translator program of the computer. In this text, each chapter concentrates on a number of keywords. Appendix A contains a summary of the more commonly used keywords. The summary may be used as a reference when writing programs.

One of the easiest ways to learn BASIC is by studying examples of various programs and then applying what has been learned to programming exercises and activities.

WRITING A SIMPLE PROGRAM

As indicated previously, a program is a sequence of instructions that tells the computer what to do. Each step in this sequence is known as a statement. Statements begin with line numbers. It is best to start with line number 10 and to increase each following line number by 10. By doing this, additional lines needed to change the program may be inserted later.

Each statement in BASIC contains one or more keywords that have a special meaning to the computer. Before examining some of the keywords in depth, observe the following short program.

LINE NUMBER

KEYWORD

```
10 REM ** THIS IS A SAMPLE PROGRAM
20 PRINT "HI!  I AM A COMPUTER."
30 PRINT "I WILL DO WHAT I AM TOLD."
40 END
```

When the computer is instructed to execute the statements of this program, the following output will be produced:

```
HI!  I AM A COMPUTER.
I WILL DO WHAT I AM TOLD.
```

Throughout this text, all examples of programs will be on a brown shaded background, and all examples of output will be on a printout page similar to the example above.

The program above illustrates three keywords; they will be used in almost all programs. They will be explained in more detail in the following sections.

Using the Keyword REM. The keyword **REM** is a shortened form of the word REMark. It allows comments to be placed in a program. It may be placed on any line in the program where the programmer wishes to make comments. REM has no effect on the computer. In fact, the computer ignores any statement beginning with REM. Any character on the keyboard may be used in a REMark statement. The asterisks (*) in the example merely separate the comment from the keyword REM. REMark statements are useful for identifying a program, labeling different sections, or making explanations. The general form of the REM statement is as follows:

General form: *line number* REM *text*

Examples:

```
10 REM ** COMMISSION REPORT
::
::
90 REM THIS SECTION CALCULATES 7% COMMISSION
```

Note: In this text the general form of each new keyword will be illustrated in this manner. Notice that the zeros have a slash through them (Ø). This is done so they will not be confused with the letter "O." The double colons (::) used between lines 10 and 90 indicate that additional statements have been omitted.

Using the Keyword PRINT. The keyword **PRINT** causes printing or output to appear. If a **CRT** (cathode ray tube, which is like a TV screen) is being used, the output will appear on the screen. If a printer is being used, the output may appear on the paper.

General form: *line number* PRINT *items to be printed*

Example:
```
1Ø PRINT "THIS IS AN EXAMPLE"
```

Printing Literals. One of the items that can be printed is a **literal.** A literal is a message enclosed in quotation marks. In the example program on page 3, the PRINT statements use literals.

Example:
```
1Ø PRINT "HI!  I AM A COMPUTER."
```

Output:
```
HI!  I AM A COMPUTER.
```

The PRINT statement will display any characters (including blanks) that have been keyed in between quotes. Quotation marks may not be used inside another quotation mark. If there is a need for quotation marks inside, use single quotation marks.

Example:
```
1Ø PRINT "THIS IS A 'WILD' PROGRAM"
```

Output:
```
THIS IS A 'WILD' PROGRAM
```

If a blank line is desired, PRINT with nothing following it may be used.

Example:

```
10 PRINT "THIS IS AN EXAMPLE"
20 PRINT
30 PRINT "OF DOUBLE SPACING"
```

Output:

```
THIS IS AN EXAMPLE

OF DOUBLE SPACING
```

Printing Constants. Another type of item that may follow the keyword PRINT is a constant. A constant is an actual number. It is not placed within quotation marks. It may not include commas, dollar signs, or any other special characters. However, it may contain a minus sign to indicate a negative number.

Example:

```
10 PRINT 75.3
20 PRINT -54.67
```

Output:

```
 75.3
-54.67
```

CONTROLLING SPACING WITH COMMAS AND SEMICOLONS

Combinations of literals or constants may appear after the PRINT statement. The items may be separated by a comma or a semicolon.

Spacing with Commas. If commas are used to separate print items, the results are printed in zones. The size and number of these zones vary with the computer used. As can be seen in the example below, using a comma to print items in zones is a method of setting up the output in columns.

Example:

```
10 PRINT "ZONE 1","ZONE 2","ZONE 3"
20 PRINT 10,20,30
```

Output:

```
ZONE 1          ZONE 2            ZONE 3
 10              20                30
```

In line 10 above, the commas between the literals cause the spacing illustrated in the output. Each comma tells the computer to move to the next zone before printing the literal. If the literal itself contains more characters than will fit into a print zone, two zones will be used. If the program attempts to print more literals than the number of zones available on a line, the output will wrap around (continue printing) on the next line.

Note also that the constants printed by line 20 are printed with a blank space before them because they are positive numbers. If any number had been negative, the negative sign (–) would have appeared in the first space. An Apple computer does not reserve the space for the negative sign.

Placing two commas together causes the computer to move over two zones.

Example:

```
10 PRINT "ZONE 1",,"ZONE 3"
20 END
```

Output:

```
ZONE 1                          ZONE 3
```

Spacing with Semicolons. When a semicolon is used to separate literals, no space is inserted between the items printed. However, on computers other than an Apple, if positive numbers are being printed, there will be one space. These computers always provide one space before a number to allow for printing the negative sign. The difference between the results of using a semicolon and a comma can be seen in the following example:

Example:

```
10 PRINT "NO","SPACE"
20 PRINT "NO";"SPACE"
30 PRINT "TOTAL",120
40 PRINT "TOTAL";120
```

Output:

```
NO              SPACE
NOSPACE
TOTAL            120
TOTAL 120
```

USING THE KEYWORD END

The END statement causes the computer to stop executing a program. Some computers require the END statement to be the last statement in a program. Then it must contain the highest line number. The END statement is optional for the Apple, Commodore Pet, IBM Personal Computer, and the Radio Shack TRS-80 microcomputers emphasized in this book.

General form: *line number* END

Example:
```
90 END
```

REVIEW QUESTIONS

1. What is a computer? (Obj. 1)
2. What is the difference between input and output? (Obj. 1)
3. What is a program? (Obj. 2)
4. Explain the difference between a program written in machine language and one written in a high-level language. (Obj. 2)
5. What is a keyword? Give some examples. (Obj. 3)
6. Why is it desirable to increase each line number in a BASIC program by 10? (Obj. 3)
7. What is the purpose of the keyword REM? (Obj. 3)
8. What is a literal? Give an example of a statement that causes a literal to be displayed. (Obj. 3)
9. What is a constant? Give an example of a statement that causes a constant to be displayed. (Obj. 3)
10. Explain how the output of a program is different when a comma is used and when a semicolon is used. (Obj. 3)
11. What is the purpose of the END statement? (Obj. 3)
12. Does the computer on which you are working require an END statement to be the last line of each program? (Obj. 3)

Topic 1.2— ENTERING AND RUNNING A PROGRAM

The previous section introduced a simple program. This section introduces commands used for entering (typing) and running (executing) programs. A command tells the computer to take immediate action. Commands are not numbered.

PROCEDURE FOR ENTERING AND RUNNING A PROGRAM

Entering a Program. Before a program is keyed in, the computer should be told to "forget" anything that may already be in its memory. The command NEW (or SCRatch on some time-shared computers) clears the memory of the computer and should always be used before a new program is entered. Simply key in the command and press the ENTER or RETURN key. In addition to clearing the memory of the computer, the NEW command may also clear the screen and move the cursor to the screen's left edge. The cursor, which usually is a dash, an underscore, or a solid block, marks the printing position at any point. As any character or space is keyed in, the cursor moves from left to right. Once the NEW command has been entered, key in the lines of the program. Remember, each line must begin with a line number. Press the ENTER or RETURN key at the end of each line. (Note: in the remainder of this text this key will be indicated by ENTER/RETURN.)

Running a Program. After keying a program into the memory of the computer, a command rather than a statement is needed to tell the computer to execute the program. The command to execute a program is RUN. Simply key in RUN and press the ENTER/RETURN key. Remember, do not key in a line number before the command. If the command RUN is entered for the program on page 3, the following output will appear:

```
HI!  I AM A COMPUTER.
I WILL DO WHAT I AM TOLD.
```

RUN could be entered again, and the same output would appear because the program has been entered in the computer's memory. It will stay in memory until it is altered (changed), removed with the NEW command, or the machine is turned off.

MAKING CHANGES IN A PROGRAM

Changes may be made in a program because the programmer desires different results or needs to correct errors. If there is an error in the program—for example, a misspelled keyword or a missing quotation mark—an error message will appear when the program is RUN (executed).

Error messages state the kind of error, such as the wrong usage or spelling of a keyword. The computer stops executing the program when an error is detected. Since there is such a difference in the way error messages are stated, the list that comes with the computer must be used to interpret these messages. Any errors have to be corrected before the program can be executed successfully. The program may have to be examined to locate the line that contains an error. This procedure is explained in the next section.

The TRS-80 and IBM microcomputers will follow the error message with the line number of the statement that contains the error. At this point, simply press ENTER, and the line will be displayed. It can then be replaced by reentering the line with the correction.

Listing a Program. The command LIST will display the entire program in the computer's memory. If the program is long and sections have to be displayed one at a time, LIST may be followed by the range of line numbers to be displayed. For example:

LIST 10–100 would display lines 10 through 100.
LIST 50 would display line 50.

Adding Lines. New lines may be added to a program simply by keying them. The line number assigned to the new line will control its placement in the program. For example, if a new line is to be placed between lines 30 and 40, the new one might be numbered 35. (Note: the computer will automatically put the line in its proper place; it does not have to be physically keyed in between the two existing lines. In fact, a program may be entered in any order; the computer will arrange the statements in numerical order.)

Deleting Lines. A single line may be deleted from a program simply by keying in the line number and pressing the ENTER/RETURN key. On some computers the command DELETE must be used along with the line number. A group of lines may be deleted by typing DELETE followed by the line number range. For example, the command DELETE 40–90 will remove all lines from number 40 through number 90.

Modifying Existing Lines. One way to modify an existing program line is to reenter the line number and new line. The new line simply replaces the old line.

OBTAINING HARDCOPY

When microcomputers or terminals are used, the output appears on the screen. If the computer has a printer attached to it, hardcopy output may be desirable—that is, output in printed form. On the TRS-80 and the IBM, LPRINT will cause the output to go to the printer when the program is RUN. To print a blank line on the TRS-80, you need a statement that prints blanks.

Example:

```
10 LPRINT "END-OF-YEAR REPORT"
20 LPRINT " "←——This would print a blank line after the heading.
```

If hardcopy output is desired on the Apple, PR#1 must be used somewhere before the first PRINT statement. This statement causes all of the output to be sent to the printer. Later in the program, if the output is not to be sent to the printer, the statement PR#0 is used.

Example:

```
40 PRINT CHR$(4);"PR#1"
50 PRINT "THIS WILL BE HARDCOPY OUTPUT"
::
::
90 PRINT CHR$(4);"PR#0"
```

On the Commodore a routine (series of statements) must be written before the first PRINT statement. The following example shows the routine that is used:

Example:

```
40 OPEN 4,4
50 CMD4
60 PRINT "THIS WILL BE HARDCOPY OUTPUT"
::
::
90 PRINT #4 ←——————————————————— turns off CMD
100 CLOSE 4
```

To obtain a hardcopy listing of the program itself, simply use the command LLIST on the TRS-80 and the IBM computers. On the Apple the following lines are entered:

```
PR#1
LIST
PR#0
```

The Commodore uses the following routine:

```
OPEN#4,4
CMD4, "PROGRAM LISTING" ←—— can be any name desired
LIST
PRINT#4
CLOSE#4
```

Note: Both of the above routines for hardcopy listings on the Apple and Commodore are not numbered since they are considered commands and not statements.

SAVING A PROGRAM

When a program is entered into the main memory of the computer, it stays there until the power goes off or it is purposely removed with the NEW command. If the program is needed for future use, it can be saved if a storage device is available. The storage device may be a **disk drive** (recording device) that uses a **floppy diskette** for storage. A floppy diskette is an oxide-coated plastic disk, either 5¼ or 8 inches in diameter, enclosed in a protective covering. It is used for magnetically storing data. The procedure to be used for saving a program on disk varies with computers. SAVE, followed by the name of the program enclosed in quotation marks, frequently is used to save the program on diskette.

Example: SAVE "EXCH1"

Note: The quotation marks are not used on the Apple.

The example illustrates a program name that is used frequently in this text. The EXCH1 means example for Chapter 1.