SCHAUM'S OUTLINE OF

THEORY AND PROBLEMS of PROGRAMMING with BASIC

•

7-36

by

BYRON S. GOTTFRIED, Ph.D.

SCHAUM'S OUTLINE OF

THEORY AND PROBLEMS of PROGRAMMING with BASIC

BYRON S. GOTTFRIED, Ph. D.

Professor of Industrial Engineering
Systems Management Engineering and Operations Research
University of Pittsburgh

SCHAUM'S OUTLINE SERIES

McGRAW-HILL BOOK COMPANY

New York St. Louis San Francisco Auckland Düsseldorf Johannesbur
Kuala Lumpur London Mexico Montreal New Delhi Panama
Paris São Paulo Singapore Sydney Tokyo Toronto

07-023842-1

7 8 9 10 11 12 13 14 15 16 17 18 19 20 SH SH 7 9

Library of Congress Cataloging in Publication Data

Gottfried, Byron S

Theory and problems of programming with Basic.

(Schaum's outline series in computer science)

Bibliography: p. 221

1. Basic (Computer program language) I. Title.

QA76.73.B3G67 001.6'424 75-2066

ISBN 0-07-023842-1

Preface

Of the many programming languages that have been developed during the past 20 years, none is easier to learn than BASIC. Yet this remarkably simple language contains enough power and flexibility to be of interest to a wide variety of persons. The use of BASIC has become particularly common at the secondary school and junior college levels. Moreover, the availability of the language through most commercial timesharing systems has caused it to be widely used for many business, technical and scientific applications.

This book offers instruction in digital computer programming using the BASIC language. All of the principal features of BASIC are discussed. However, the objectives of the book are to teach the reader how to organize and write efficient computer programs and to stress the importance of good programming practice as well as to present the rules of BASIC.

The style of writing is deliberately elementary. This enables the book to be easily understood by a wide reader audience, ranging from high school students to practicing professionals. The book is particularly well suited to the advanced secondary or beginning college level, either as a textbook for an elementary programming course, a supplementary text for a more comprehensive course in analytical techniques, or as an effective self-study guide. For the most part, the required mathematical level does not go beyond high school algebra.

The text is organized into two parts of about equal length. Part I – "Basic BASIC" – contains the commonly used features of the language. A brief programming course can be taught from this material alone. Part II – "Advanced BASIC" – is concerned with more specialized features, such as subroutines, matrix statements and file manipulation.

The material is presented in such a manner that the reader can write complete, though elementary, BASIC programs as soon as possible. It is very important that the reader write such programs and run them on a computer concurrently with reading the text. This greatly enhances the beginning programmer's self-confidence and stimulates his interest in the subject. (Learning to program a computer is like learning to play the drums: it cannot be learned simply by studying a textbook!)

A large number of examples are included as an integral part of the text. These include a number of comprehensive programming problems as well as the customary drill type of exercises. In addition, a set of solved problems is included at the end of most chapters. The reader should study these examples and solved problems carefully as he reads each chapter and begins to write his own programs.

Sets of review questions, supplementary problems and programming problems are also included at the end of each chapter. The review questions enable the reader to test his recall of the material presented within the chapter. They also provide an effective chapter summary. Most of the supplementary problems and programming problems require no special mathematical or technological background. The student should solve as many of these problems as possible. (Answers to the supplementary problems are provided at the end of the text.) When using this book as a course text, it may also be advisable for the instructor to supplement the programming problems with additional assignments that reflect particular disciplinary interests.

I wish to express my gratitude to Mrs. Joanne Colella for patiently and diligently typing the entire manuscript, and to the Computer Center of the University of Pittsburgh for the computer time required to solve the numerous comprehensive programming examples. Also, I wish to thank my students who used early versions of the manuscript for their many constructive suggestions.

Finally, the reader who completes this book will have learned quite a lot about general computer programming concepts as well as the specific rules of BASIC. He should be completely convinced that programming with BASIC is not only *easy* but also *fun*!

BYRON S. GOTTFRIED

Contents

Complete Programming Examples

Part I

Basic BASIC

Chapter 1

Introductory Concepts

This book offers instruction in computer programming using the programming language called BASIC (Beginner's All-purpose Symbolic Instruction Code). We will see how a problem that is initially described in words can be analyzed, outlined and finally transformed into a working BASIC program. These concepts are demonstrated in detail by the many sample problems that follow.

1.1 Computer Characteristics

What is a digital computer and how does it operate?

A computer is an *information processing* machine. In order to function it must be given a unique set of instructions, called a *program*, for each particular task it is to perform. The program is stored in the computer's internal *memory* for as long as it is needed. Once a complete program has been stored in memory it can be *executed*, causing the desired operations to be carried out.

Normally, execution of a computer program causes the following to happen:

1. The necessary information, called the *input data*, is read into the computer and stored in another sector of the computer's memory.
2. The input data is processed to produce the desired results, which are known as the *output data*.
3. Finally, the output data (and perhaps a portion of the input data) is printed out onto a sheet of paper.

Example 1.1

Suppose we wish to use a computer to calculate the area of a circle using the formula

$$A = \pi r^2$$

given a value of the radius r. The steps involved would be as follows:

1. Read a numerical value for the radius.
2. Calculate a value for the area, using the above formula.
3. Print the values of the radius and the associated area.
4. Stop.

Each of these steps might correspond to one instruction in a BASIC program. Once the complete program has been executed it can be erased from memory, or it can be saved and run again, using a different value for the radius.

We can represent the entire procedure pictorially, as shown in Fig. 1.1. This is known as a *flowchart.* Flowcharts can be helpful in assisting the reader to visualize the flow of logic in a program.

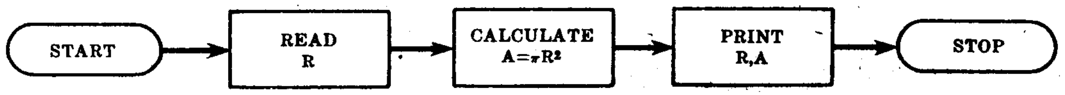

Fig. 1.1

We will consider what is meant by "reading" and "printing" in the next section. For now, however, let us take note of two important computer concepts: the computer's memory, and its ability to be programmed. Modern computers have memories that range in size from a few thousand to several hundred thousand *words*, where a word may represent an instruction, a numerical quantity or a set of characters. The size of a computer's memory is usually expressed as some multiple of $2^{10} = 1024$ words. This quantity is referred to as 1K. Thus a computer with a 48K memory will have $48 \times 1024 = 49{,}152$ words. Thus a great deal of information can be stored in the memory of even a small computer. Moreover, the unlimited number of different programs that can be written for a given computer is the principal reason for the computer's great versatility.

An additional computer characteristic is its extremely high speed. Simple tasks, such as adding two numbers, can be carried out in a fraction of a microsecond. (One microsecond, abbreviated 1 μs, is equivalent to 10^{-6}, or one millionth, of a second.) On a more practical level, the end-of-semester grades for all students in a large university can be processed in a few seconds of computer time, at a cost of about one dollar.

1.2 Modes of Operation

There are two different ways that a digital computer can be used. These are the *batch mode* and the *timesharing mode.* Both are very common. Each has its own advantages for certain types of problems.

Batch Processing

In *batch processing*, a number of jobs are read into the computer and are processed sequentially. (A *job* refers to a computer program and the sets of input data to be processed.) Usually the program and the data are recorded on punched cards. (Each character in an instruction or data item, and each digit in a numerical quantity, is represented on a punched card by an encoded set of holes. Every set of holes occupies one column of the punched card. A punched card will typically contain 80 columns. Thus as many as 80 characters or digits can be recorded on a single punched card.) Hence each job will be read into the computer via a mechanical cardreading device, and the output will be printed by means of a special printer.

Large quantities of information (both programs and data) can be transmitted into and out of the computer very quickly in batch processing. Therefore this mode of operation is well suited for jobs that require large amounts of computer time or are physically lengthy. On the other hand, the time required for a job to be processed in this manner usually varies from several hours to one or two days, even though the job may have required only a second or two of actual computer time. (The job must wait its turn before it can be read, processed and the results printed out.) Thus batch processing can be undesirable when it is necessary to process a simple job and return the results as quickly as possible.

Example 1.2

A student has 1000 different values for the radius of a circle and would like to calculate an area for each radius. To do so he must execute a computer program (such as the one described in Example 1.1) 1000 different times, once for each value of the radius. Batch processing will be used, since all of the calculations are to be carried out in rapid succession.

To process the data the student will read a deck of cards into the computer. The first part of the deck will contain the program, with one instruction per card. Following the program will be 1000 data cards, each of which will contain one value for the radius. The card deck is illustrated in Fig. 1.2.

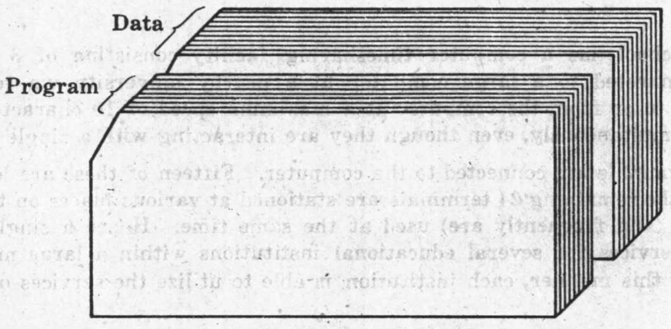

Fig. 1.2

After the deck of cards has been read into the computer there will be a delay of several hours before the program is executed and the input data processed. After the computation has been completed the results will be printed on a large sheet of paper. In this case the student will receive a listing that contains 1000 pairs of numbers. Each pair of numbers will represent a value for the radius and its corresponding area.

Timesharing

Timesharing involves the simultaneous use of the computer by several different users. Each user is able to communicate with the computer through a *typewriter terminal*, which may be connected to the computer by a telephone line or microwave circuit. (A commonly used typewriter terminal is shown in Fig. 1.3.) The terminals are usually located remotely from the computer – perhaps several hundred miles away. Because all of the terminals can be served at essentially the same time, each user will be unaware of the others and will seem to have the entire computer at his own personal disposal.

A single typewriter terminal serves as both input and output device. The program and the input data are typed into the computer via the keyboard, and the output data are transmitted from the computer to the typewriter terminal, where they are printed out. The transmission of data to and from the typewriter terminal is much slower, however, than the processing of the data by the computer. The processing may take 1 second, the transmission 5 minutes. (The cardreaders and printers used in batch processing are able to transmit data much more rapidly than are typewriter terminals.) It is

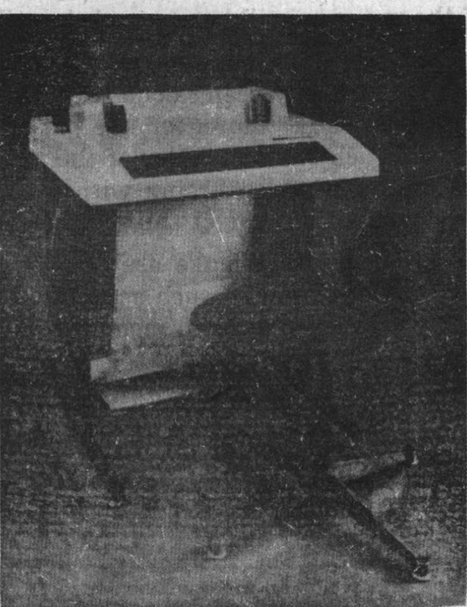

Fig. 1.3

this relative difference in speed that allows one computer to interact with several typewriter terminals simultaneously.

Timesharing is best suited for processing relatively simple jobs that do not require extensive data transmission or large amounts of computer time. Many of the computer applications that arise in schools and commercial offices have these characteristics. Such applications can be processed quickly, easily and at minimum expense using timesharing.

Example 1.3

A suburban high school has a computer timesharing facility consisting of 3 typewriter terminals. These terminals are connected to a large computer at a nearby university via telephone lines. Each terminal transmits data to or from the computer at a maximum speed of 10 characters per second. All 3 terminals can be used simultaneously, even though they are interacting with a single computer.

An additional 39 terminals are connected to the computer. Fifteen of these are located at 5 other high schools in the area, and the remaining 24 terminals are stationed at various places on the university campus. All 42 terminals can be (and frequently are) used at the same time. Hence a single computer is able to provide computational services for several educational institutions within a large metropolitan area. By sharing the computer in this manner, each institution is able to utilize the services of a large computer at a reasonable cost.

Example 1.4

A student has written a BASIC program that he wishes to execute in the timesharing mode. To do so, he must type his program into a typewriter terminal, one instruction at a time. Each instruction is typed on a separate line. The instructions are transmitted to the computer, where they are stored in memory as soon as they are typed.

After the entire program has been typed and stored in memory, the student can execute his program simply by typing the word RUN into the typewriter terminal. This will cause the input data to be read and the output data to be computed and then transmitted directly to the typewriter terminal.

An important feature of timesharing is the ability of the computer and the user to *interact* with one another during program execution. Thus a relatively small portion of a program may be executed and a message printed out, indicating the preliminary results that have been computed and requesting additional data from the user. Further execution of the program will be suspended until the additional data have been supplied.

In supplying the requested data, the user may be influenced by the preliminary results that have already been calculated. Thus he may wish to study the output before supplying any additional data. Moreover, the particular data supplied by the user may influence the manner in which the remainder of the program is executed. Hence the computer and the user are, in a sense, conversing with each other, since the information supplied by one of the participants may affect the subsequent actions of the other.

Programs that make use of this idea are said to be written in a *conversational mode*. Computerized games, such as tic-tac-toe, checkers and chess, are excellent examples of interactive (or conversational mode) programs.

Example 1.5

A student wishes to use a timesharing system to calculate the radius of a circle whose area has a value of 100. An interactive program is available which will calculate the area of a circle, given the radius. (Note that this is just the opposite of what the student wishes to do.) Therefore the student will proceed by trial-and-error, guessing a value for the radius and determining whether or not this value corresponds to the desired area. If not, the student will assume another value for the radius and calculate a new area, and so on. This trial-and-error procedure will continue until the student has found a value for the radius that yields an area sufficiently close to 100.

The student will communicate with the computer by means of a typewriter terminal. Once the communication link has been established and program execution begins, the message

```
RADIUS=?
```

will be printed on the typewriter terminal. The student then enters a value for the radius. Let us assume that the student enters a value of 5 for the radius. The typewriter terminal will then respond by printing

AREA= 78.5398

DO YOU WISH TO REPEAT THE CALCULATION?

The student then types either YES or NO. If the student types YES, then the message

RADIUS=?

will again be printed, and the entire procedure is repeated. If the student types NO, then the message

GOODBYE

is printed on the typewriter terminal, and the communication link to the computer is automatically disconnected.

In Fig. 1.4 we see the information that is printed during a typical timesharing session, using the program described above. The information typed by the student has been underlined. An approximate value of $r = 5.6$ was determined after only 3 calculations.

Notice the manner in which the student and the computer appear to be conversing with one another. Also, note that the student waits until he sees a calculated area before deciding whether or not to carry out another calculation. If so, the new value that the student supplies for the radius will be influenced by the previous calculated results.

```
RADIUS=? 5
AREA= 78.5398

DØ YØU WISH TØ REPEAT THE CALCULATIØN? YES

RADIUS=? 6
AREA= 113.097

DØ YØU WISH TØ REPEAT THE CALCULATIØN? YES

RADIUS=? 5.6
AREA= 98.5204

DØ YØU WISH TØ REPEAT THE CALCULATIØN? NØ

GØØDBYE
```

Fig. 1.4

1.3 Introduction to BASIC

Many different kinds of languages have been developed for programming a digital computer. These languages differ considerably in their degree of difficulty, their general availability and their intended purpose.

BASIC is a digital computer programming language whose instructions resemble elementary algebraic formulas, augmented by certain English words, such as LET, GO TO, READ, PRINT, IF, THEN, etc. Thus BASIC is a simple, "people-oriented" language, in contrast to some other programming languages that are more cryptic and therefore more difficult to learn and use. Moreover, many people find that BASIC programming can be a lot of fun, in much the same way that some people enjoy solving crossword puzzles.

Because of its similarity to elementary algebra BASIC is particularly well suited to solving problems in science, mathematics and engineering. However, use of the language is by no means restricted to these areas. BASIC is also applied to a wide variety of problems in business, economics, psychology, medicine, library science – virtually any area that might require extensive manipulation of numerical data or character information. (By *character information* we mean nonnumerical symbols, letters, words, etc.) We will see a variety of elementary BASIC applications in the programming examples included in this book.

BASIC was originally developed at Dartmouth College by John Kemeny and Thomas Kurtz in the mid 1960's. The simplicity of the language attracted the attention of a number of commercial timesharing services, and it was not long before several of these services had adopted BASIC for the use of their customers. Most timesharing services now offer some version of BASIC, though there are some variations in the more advanced features. Nevertheless, the language has become sufficiently standardized so that anyone learning one version of BASIC (as presented in this book, for example) should have little difficulty in adapting to any other version.

As might be expected, BASIC is commonly thought of as a timesharing language. It is, in fact, the most widely used of all currently available timesharing languages. We will present the language from a timesharing point of view in this book. The reader should understand, however, that BASIC is also an effective batch-processing language, and its use should therefore be considered for batch as well as timesharing applications.

Structure of a BASIC Program

Each instruction in a BASIC program is written as a separate *statement*. Thus a complete BASIC program will be composed of a sequence of statements. These statements must appear in the order in which they will be executed unless a deliberate "jump" (i.e., a transfer of control) is indicated.

The following rules apply to all BASIC statements:

1. Every statement must appear on a separate line.†
2. A statement cannot exceed one line in length (i.e., cannot be "continued" from one line to the next).
3. Each statement must begin with a positive integer quantity, known as a *statement number* (or *line number*). No two statements can have the same statement number.
4. Successive statements must have increasing statement numbers.
5. Each statement number must be followed by a BASIC *keyword*, which indicates the type of instruction that is to be carried out.
6. Blank spaces may be inserted wherever desired in order to improve the readability of the statement.

It is also possible to include blank lines in a BASIC program. Every blank line must have a unique line number, however, and the line number must be followed by at least one blank space. (If a line number is followed immediately by a carriage return, then that line will be *deleted* from the program. We will discuss this further in Chapter 3.)

Example 1.6 Area of a Circle

Figure 1.5 presents a simple BASIC program to calculate the area of a circle whose radius is specified. The logic used to carry out the computation has already been discussed in Example 1.1. The program is so elementary, however, that its logical basis can be determined by simple inspection.

```
10 INPUT R
20 LET A=3.14159*R↑2
30 PRINT R,A
40 END
```

Fig. 1.5

We see that the program consists of four statements, each of which appears on a separate line. Every statement has its own statement number (or line number). These numbers increase successively from the beginning (top) to the end (bottom) of the program. The statements contain the BASIC keywords INPUT, LET, PRINT and END, respectively.

The purpose of the first statement (INPUT) is to enter a numerical value for the radius (R) from a typewriter terminal. The second statement (LET) causes the quantity πR^2 to be evaluated. This quantity will then be represented by the letter A. The third statement (PRINT) causes the numerical values for R and A to be transmitted to the typewriter terminal, where they are then printed. Finally, tne last statement (END) is required in order to identify the end of the program.

Notice the symbols that are used in line 20 to represent arithmetic operations. Multiplication is indicated by an asterisk (*), and a vertical arrow (↑) is used to raise a quantity to a power. (This latter operation is known as *exponentiation*.) The other arithmetic operations, namely addition, subtraction and division, are represented in BASIC by a plus sign (+), a minus sign (−) and a slash (/), respectively.

†On most typewriter terminals, a line is equivalent to 72 characters. However, some typewriter terminals allow as many as 132 characters per line.

Execution of a BASIC Program

Once a BASIC program has been written it can be run on a great many different types and brands of computers. Except for a few minor differences that may exist from one version of BASIC to another, the language is machine-independent. This has been a significant factor in the widespread acceptance of BASIC.

The reason for this independence from a machine is that execution of a BASIC program is actually a two-step procedure. The first step, called *compilation,* causes the BASIC statements to be converted to the computer's own *machine-language* instruction set. (*Machine language* consists of a large number of very detailed instructions that are related to the computer's internal, electronic circuitry. Every computer has its own machine language. Programming in machine language is much more difficult than programming in a higher-level language, such as BASIC. Moreover, machine-language programs are not transferable from one type of computer to another.) Accordingly, the BASIC program is referred to as the *source* program. In the second step the resulting machine-language program, called the *object* program, is executed, causing the input data to be processed in the desired manner.

Usually the object program will be executed immediately after it has been compiled. The programmer will not be aware that two distinct steps have taken place, since he sees only the source program, the input data and the computed output data.

Some Advantages of BASIC

1. BASIC is "people-oriented." It is easy to learn and fun to use. Any well-organized person can learn to program in BASIC. An extensive background in mathematics is not necessary.
2. The language is flexible, allowing the programmer to alter an existing program with very little effort.
3. BASIC is well suited for use in a timesharing system, which offers the programmer the use of a large computer at minimum expense.
4. Except for minor differences between one version of BASIC and another, the language is machine-independent. Hence a BASIC program can be run on many different computers.

Review Questions

1.1 A digital computer can be thought of as what kind of a machine?

1.2 What is meant by a computer program?

1.3 What, in general, happens when a computer program is executed?

1.4 What is a computer memory? What kinds of information are stored in a computer's memory?

1.5 What terms are used to describe the size of a computer's memory? What are some typical memory sizes?

1.6 What time unit is used to express the speed with which elementary tasks are carried out by a computer?

1.7 What is meant by batch processing and timesharing? Describe the advantages and disadvantages of each.

1.8 In what sense can a computer and a user interact with one another? What is meant by a conversational mode program?

1.9 What does the word BASIC stand for? What are the general characteristics of the BASIC language?

1.10 What is a BASIC statement? In what order must the statements appear in a BASIC program?

1.11 Summarize the 6 rules that apply to all BASIC statements.

1.12 In BASIC, what are the symbols that are used to indicate addition, subtraction, multiplication and division?

1.13 What is meant by exponentiation? What symbol is used in BASIC to represent exponentiation?

1.14 What is meant by compilation of a BASIC program? What is a source program? Object program? Why are these concepts important?

1.15 Summarize the principal features of the BASIC language.

Solved Problems

1.16 Several elementary BASIC programs are presented below. Explain the purpose of each program.

(*a*)

```
10 INPUT L,W
20 LET A=L*W
30 PRINT L,W,A
40 END
```

To calculate the area of a rectangle whose length and width are given.

(*b*)

```
10 INPUT A,B,C,D,E
20 LET S=A+B+C+D+E
30 PRINT A,B,C,D,E
40 PRINT S
50 END
```

To calculate the sum of 5 numbers. Note that the 5 numbers will be printed on one line, and the calculated sum on the next line. (Each PRINT statement begins on a new line.)

(*c*)

```
10 INPUT A,B,C
20 LET X1=(-B+(B↑2-4*A*C)↑.5)/(2*A)
30 LET X2=(-B-(B↑2-4*A*C)↑.5)/(2*A)
40 PRINT A,B,C,X1,X2
50 END
```