BASIC BY DESIGN

Structured Computer Programming in BASIC

Andrew Kitchen

BASIC BY DESIGN
Structured Computer Programming in BASIC

ANDREW KITCHEN
Rochester Institute of Technology
Rochester, N.Y.

Prentice-Hall, Inc.
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging in Publication Data

Kitchen, Andrew.
BASIC by design.

Includes index.
1. Basic (Computer program language) 2. Structured programming. I. Title.
QA76.73.B3K55 1983 001.64'24 82-21562
ISBN 0-13-060269-8

Editorial/production supervision: *Aliza Greenblatt/Barbara Palumbo*
Interior design: *Aliza Greenblatt*
Cover design: *Ray Lundgren*
Manufacturing buyer: *Gordon Osbourne*

Printed in the United States of America

10 9 8 7 6 5 4 3

ISBN 0-13-060269-8

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
EDITORA PRENTICE-HALL DO BRASIL, LTDA., *Rio de Janeiro*
PRENTICE-HALL CANADA INC., *Toronto*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., *Singapore*
WHITEHALL BOOKS LIMITED, *Wellington, New Zealand*

Preface

There are two major pitfalls on the way to writing an introduction to BASIC programming.

The first is the difficulty of deciding which features of the language should be covered, for the truth is that BASIC is not just one language but numerous dialects, all of which lay claim to the name "BASIC." There is a national standard for the language, the ANSI standard for minimal BASIC.† Unfortunately, this does not cover character string manipulation and file handling, two important areas of computer programming. Although most recent versions of BASIC provide facilities for handling these operations, the approaches taken vary widely from one dialect to another. This situation is made worse by the fact that some BASIC dialects do not conform to the ANSI standard, even where it does apply. This problem is resolved in *BASIC by Design* by discussing a particular version of BASIC, Digital Equipment Corporation's BASIC-PLUS, which observes the national standard wherever it applies, and by using the minimum extensions necessary to give a solid introduction to character strings and files. The discussion is kept general enough, and sufficient guidelines are provided, to make the conversion from this BASIC to any other relatively painless. Comparison of certain features of selected dialects are presented in the appendices.

The second pitfall surrounds the emphasis of the presentation. The beginning programmer does not want to be faced with a heavy treatise on programming style, particularly at a stage when his major concern is "How on earth do I get this *!?**? machine to do anything useful and not just sit there spitting error messages at me?" On the other hand, it does the student no service to introduce the features of the language one after another, cookbook style, without providing some instruction on how to write programs—clear, readable, error-free, working programs!

†Standard specifications for the syntax and semantics of a minimal core of BASIC statement types approved by the American National Standards Institute in 1978.

Like well-written prose, clearly organized programs are easier to understand and therefore easier to check and debug. The ramifications, however, go deeper than this because the structure of a program is intimately related to the approach that is taken in writing it. One cannot graft good style onto a badly planned program. So the question becomes: Does one tell the student about BASIC or teach him how to program? The bias of this text is, not surprisingly, toward the latter.

Initially, a sufficient number of statement types (PRINT, LET, INPUT) are discussed to allow the reader to write simple straight-line programs. Subsequently, new statement types (READ, DATA, IF-THEN, etc.) are introduced progressively as tools for solving the problems that arise as more sophisticated projects are attempted. The elementary building blocks (REPEAT loops and IF blocks), are introduced, first informally and later as part of a structured programming approach. In the interests of laying a solid foundation of good programming habits, the logical constructions introduced are kept as few and as simple as possible. For example, the IF-THEN-ELSE structure is not discussed nor is the ON-GO TO statement, both of which add extra power at the cost of confusing the beginning student just when he is coming to terms with REPEAT loops and IF blocks. Once the foundations are laid, other topics are tackled: the processing of lists and tables of data, text manipulation, the use of subroutines and data files and the application of the random number generator, RND.

The approach is kept on an elementary level. For example, a knowledge of high school algebra is the extent of the mathematical background required for all but a handful of the exercises, and the formalism of structured programming is kept to the bare essentials. However, this is not to say that the book will not challenge the more sophisticated reader. The exercises play an important role. Those in the early chapters lead the reader in easy steps through the process of program development, but he is increasingly left on his own in the later chapters. There are many extensions of the material in the book which would provide interesting classroom discussion or could be pursued independently by the reader (e.g., sorting and searching techniques; numerical methods; other program structures, IF-THEN-ELSE, CASE blocks, etc.; extensions to BASIC, PRINT-USING, MAT statements, etc.). Chapters 2 through 6 represent the core of the book. They should, as far as possible, be read in sequence, as each one builds on the one before. The second half of the book is more loosely organized, as each chapter covers a separate topic. Here are a few comments on the contents of some of the chapters.

Chapters 1 and 2 provide a leisurely introduction to communication with the computer. Their purpose is to help the beginning user to see the computer as a nonthreatening tool.

Chapter 3 introduces a principal theme of the book: the production of clear and readable programs, both as far as the output is concerned as well as

in the structure of the code. The INT function is introduced in this chapter. It appears this early primarily because it is such a useful tool; it appears repeatedly in the examples and exercises of later chapters. Furthermore, it is needed here to provide some control over output format (the PRINT USING statement gives better results but is system dependent). The semicolon, as used in the PRINT statement, is also introduced here. The use of the comma to produce columns is deferred until Chapter 4, where the printing of tables is discussed. For the more sophisticated reader, Chapters 1 through 3 will be easy reading and can be covered quite rapidly.

In Chapters 4 and 5, the two major program structures, loops (REPEAT loops) and conditional blocks (IF blocks), are introduced informally. The nonstandard form of the IF statement,

IF condition THEN statement

is used to a limited extent in Chapter 5. It is restricted to a few examples and alternative approaches are discussed at length. The reason for discussing it all, apart from the fact that it is available in most popular versions of BASIC, is that it provides a simple introduction to conditional statements and blocks. By contrasting its use with the traditional "jump around" approach, the action of the latter can be clarified.

Chapter 6 summarizes the presentation of the earlier chapters. A structured approach to program development is formally introduced and is illustrated with worked examples.

In Chapter 9, the operators required for character string manipulation are introduced. Many versions of BASIC use functions almost identical to those discussed here. Examples of other approaches are illustrated in Appendix III.

As mentioned above, file handling in BASIC has not been standardized. Chapter 11 attempts to address this problem by emphasizing general concepts of sequential file handling, those common to all systems. The examples given are written in BASIC-PLUS. However, guidelines for file handling are discussed which should enable the reader to convert these examples to run on different systems without much difficulty. Again, a survey of other approaches is provided in Appendix IV.

BASIC by Design is intended for use in a wide range of introductory courses, from a semester-long introduction to programming to an intensive (perhaps, self-study) course for graduate students in the social sciences and management. However, it is not aimed solely at the college student. Any person who has a microcomputer at home and would like to learn how to control the little monster should find this book a solid and, hopefully, entertaining introduction to programming in BASIC.

Andrew Kitchen

Acknowledgements

Many people have contributed to the form and content of this book. Unfortunately, the complete list far exceeds the capacity of my rather unreliable memory and so my thanks and apologies go to all those not mentioned.

I am most grateful for the careful analysis and suggestions provided by each of my reviewers, Drs. Brian Kernighan, Kenneth Long, Philip Raymond, and Spotswood Stoddard, and the comments of my colleagues, Nancy Watts and Marjorie Schmieder, as well as the response of my students to classroom presentation of this material. Dr. Larry Coon corrected erroneous information in the appendices and Karen Lattierre and her word processor lightened the load of writing letters to numerous software companies. The editorial staff at Prentice-Hall was always helpful and efficient. Particular thanks go to editors Stephen Cline, James Fegen, Aliza Greenblatt, and Barbara Palumbo as well as to Paulette Christie.

However, the one person who deserves the major credit for the completion of this book is my wife, Judith. Not only did she take time away from her own writing career to type the manuscript, but, also, her watchful eye helped to keep my style relatively pure. Her cynical view of the world of computers and computerese has eliminated much meaningless verbiage from the text and helped to make the book, as they say in the trade, user-friendly.

Finally, I wish to thank my sons, William and Matthew, for being prepared, at times, to turn down their stereo so that I could think.

Contents

Chapter

1

Starting Out

1.1 THE COMPUTER

A popular way to begin an introductory programming text is to give a brief history of the computer and a survey of the present situation in the field, with descriptions of the machines available and the uses to which they are put. Such discussions will be more fruitful when you have a better grasp of the nature of the beast. So we will dive right into the practical problem of communicating with it. Later we will return to discuss computers in more general terms and to remove some of the mystery that surrounds their operation.

There are many different makes of computer available and each manufacturer tries to design its machine to be different, and a little more attractive than those of its competitors. Specifically, every system has its own set of procedures which must be followed before the computer can be used. The computing system that you are using is likely to fall into one of two categories: either a personal microcomputer or one of a number of video or typewriter terminals serviced by a larger computer. Detailed instructions on how to use your system will be available in the manufacturer's manuals or at the installation where you are doing your computing.

This chapter describes the sign-on procedure for one time-sharing system, that provided by Digital Equipment Corporation on their PDP-11 series of minicomputers, a system popular with colleges and universities. If you are using a system of this type rather than a microcomputer, your sign-on procedure will be similar, although probably not identical. Rather than being upset by this lack of uniformity, always be prepared to ask for help if you

are having trouble. Even an expert cannot know everything about every available system. On the other hand, if you are using a microcomputer, starting up the machine may consist of little more than switching it on and typing a single command. Because of this wide variation, you may find it helpful to read Appendix I, where the procedures for two small systems are discussed.

A word of reassurance before you start: Please do not be afraid to experiment, no matter how little you feel you know. The modern computer is a relatively robust piece of equipment. It may not always respond the way you wish it would, but you cannot make it go up in smoke just by pressing the wrong key.

We used the term **time sharing** above. Let's clarify this. If you are using a microcomputer, you will have all that computing power to yourself. However, it is wasteful to devote an entire large computer to one user. The fact is that a computer operates so fast that most of the time it is idling, waiting for data to be typed in or printed out or waiting for instructions. Ingenious use is made of this fact to enable the computer to service more than one user at a time. A number of users are linked to the machine by means of video display or typewriter terminals. The machine processes instructions or data from each terminal in turn, cycling rapidly through all the users currently signed on. It spends no more than a fraction of a second with each one. Every user has all the power of the computer at his disposal, and because of the extraordinary speed of the machine, is unable to detect that this process is taking place. This technique is called **time sharing** and the system is referred to as a **multiuser** system.

If you are using such a system, it is quite possible that you will never see the computer. You will communicate with it through a **terminal,** a device similar to a typewriter, which will, in general, be in a different room from the machine itself. In fact, it is quite possible for the computer to be in a different building or even in another town.

Two types of terminal are in common use. A **printing terminal** is very much like an electric typewriter. You communicate by typing instructions at the keyboard. These are printed out for your convenience, as well as being transmitted to the machine. Responses from the computer are sent to the terminal to be printed so that you can read them. The other popular design is the **video display** or **CRT terminal** (CRT stands for cathode ray tube). This device has a typewriter keyboard but the printer is replaced by a television screen. No printed output can be produced; instead, the instructions and responses are displayed on the screen. The usefulness of such devices is limited by the fact that the user has no printed record of his work. On the other hand, they are fast and quiet and often cheaper than **hard-copy** (i.e., printing) devices.

One device you are less likely to see today is the **keypunch.** Most computer input was, until quite recently, punched onto cards in a coded form.

The cards were then fed into a **card reader,** which sensed the holes in each card electrically and transmitted the coded information to the computer. Many organizations still use such equipment, but more efficient methods have begun to supersede it.

1.2 SIGNING ON

Your terminal may be wired directly to the computer. However, if the machine is an appreciable distance from the terminal, communication will probably take place via a telephone line. Contact with the machine is made by first placing the telephone receiver face down on a device called a **modem** (modulator-demodulator) or **coupler** and then dialing the computer's number. In many installations, the dialing occurs automatically. Incidentally, the **modem** is a unit which translates the electronic code of the terminal into a code of sounds that can be transmitted by telephone, and vice versa.

Once you have made contact with the computer, the computer may respond with a message or by printing some sort of prompting character to indicate that it is ready. If you are using a small microcomputer, you are now ready to start playing with the machine, so you can skip the rest of this chapter. Larger multiuser systems require more elaborate sign-on procedures. If you are using one of these, you will now have to establish your right to use the computer. This is usually done by typing an account number and a password. So, before you attempt to use the machine, find out what the protocol is for your installation and also make sure that you have been issued an account number and have chosen a password.

All this may seem like an inconvenience, but it allows for some control over the users of the system. Also, as you will see later, it provides individual users with protection if they have private material to which they wish to restrict access.

1.3 A TYPICAL SIGN-ON PROCEDURE

Here is the **sign-on** procedure (also referred to as **log-in**) used on the PDP-11 RSTS-E time-sharing system. Switch on the terminal and, if necessary, dial the computer. Then type

```
HELLO ●
```

and press the **RETURN** key (on some keyboards this key is referred to as **ENTER**). The machine responds with (for example)

```
RSTS V7.0-07 **SJFC** Job 7 KB14 11-Feb-80 15:00
```

and continues on the next line with

```
#
```

Then it stops. It is waiting for your account number. Type it right after the # symbol:

```
#202,1 ●
```

The symbol ● indicates that you should press the RETURN (or ENTER) key. This key must be pressed after every message or command that you type. The machine responds with

```
password: ●
```

and stops again, this time waiting for your password. Type this on the same line and press RETURN. As you type, the system will hide the password by suppressing the printout, so that nothing appears on the paper. Some systems allow the password to be printed and then print over it with X's to obliterate it. The machine may now respond with a message. In any case, it ends by printing

```
Ready
```

The sign-on procedure is complete. Now you can use the system.

Just to give you a better idea of the procedure and to introduce some notation, here is the complete example as it appears at the terminal. Everything you type has been **underlined** to distinguish it from what is printed by the computer. We will continue this convention throughout the book in order to clarify the examples, whenever there is interaction between the machine and the user.

Example

The complete sign-on procedure may look something like this:

```
HELLO ●
RSTS V7.0-07 ** SJFC **  Job 5  KB20  02-Feb-82  08:48
#202,1 ●
Password: ●          (machine suppresses the printing of your secret
                      password)

WELCOME TO RSTS/E V7.0 TIME SHARING
```

```
THE USER  ROOM HOURS ARE:
--------------------------

   MON-FRI  7:00 A.M.  TO 12:00 MIDNIGHT
   SAT     10:00 A.M.  TO 12:00 MIDNIGHT
   SUN     12:00 NOON  TO 12:00 MIDNIGHT

THE DECWRITERS ARE SET TO A BAUD RATE OF 300
AND A WIDTH OF 80 CHARACTERS.

TIMESHARING WILL BE DOWN FROM 4:30 P.M. TO 5:15 P.M.
EVERY FRIDAY NIGHT TO ALLOW FOR SYSTEM BACKUP...


HOPE YOU HAD A GOOD VACATION!!!!!!!!!!!!!!!!!!!!!!

                                          THANK YOU!!!!


Ready                                                  ■
```

You will notice that almost all the printing is in uppercase letters. Computer output devices were traditionally limited to printing uppercase characters and now, although most systems will accept and print both upper- and lowercase, it is still common practice to use only capital letters in computer programming. In fact, most terminals have a key labeled **CAPS LOCK** which selects uppercase letters without affecting the other keys. Uppercase symbols such as !, (,), and * over the numbers must still be selected by holding down the **SHIFT** key.

Don't forget that, when typing data or instructions into the machine, you must always (well, almost always) follow your input by pressing the RETURN key, ● . This tells the machine that you are done and that it can start to process whatever you have typed in. On the other hand, the machine itself will indicate to you that it has finished the current task, and is ready to accept further instructions, by printing a word such as **Ready** (or **Done** or **OK**).

1.4 SIGNING OFF

When you learn to drive a car, learning to stop the thing is at least as important as starting it. The same is true for the computer. After you have finished working at the machine, resist the temptation to just switch it off