FOUNDATIONS OF PROGRAMMING THROUGH BASIC

PETER MOULTON

FOUNDATIONS OF PROGRAMMING THROUGH BASIC

PETER MOULTON
University of Oregon

JOHN WILEY & SONS
New York · Chicester · Brisbane · Toronto

Copyright © 1979, by John Wiley & Sons, Inc.

All rights reserved. Published simultaneously in Canada.

Reproduction or translation of any part of
this work beyond that permitted by Sections
107 and 108 of the 1976 United States Copyright
Act without the permission of the copyright
owner is unlawful. Requests for permission
or further information should be addressed to
the Permissions Department, John Wiley & Sons.

Library of Congress Cataloging in Publication Data:

Moulton, Peter.
Foundations of programming through BASIC.

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

QA76.73.B3M673 001.6'424 78-21569
ISBN 0-471-03311-1

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

FOUNDATIONS OF PROGRAMMING THROUGH BASIC

PREFACE

This book introduces the foundations of programming through the use of the programming language BASIC. Although BASIC is reputed to be one of the most easily learned programming languages and is becoming commonly used, its design is related more to the operations of a computer than to the processes involved in solving a problem and creating a program. BASIC offers few guidelines and enforces little discipline toward the creation of clear, understandable programs. For this reason it is essential that the beginning programmer learn the foundations of good programming methodology.

This textbook presents the essential elements of the programming methodology that has been developed by computer scientists during the past decade. A consistent style of problem solving, procedure development, and programming has been adopted in all of the examples to demonstrate a methodology that explicitly relates the resultant program to the procedure and to the problem. This particular style is not mandatory and may be altered to suit the tastes of students and instructors.

An additional objective is to present all of the material within a framework of intuitively understandable problems. The material covering a particular concept and the set of related language features is presented in the following manner: first by explaining a problem together with a discussion of how that problem might be solved (the procedure), second, by exploring what sorts of tools are needed (such as the kinds of data and the operators), and finally, by determining how these tools are provided in the BASIC language. This approach—problems, then concepts, then language—is especially important in light of the many differences among programming languages, or even among the different versions of BASIC that the student might encounter.

Chapter One introduces the notions of a problem, of a procedure to compute the solution to a problem, and of the languages in which procedures can be implemented. It continues with a brief discussion of the capabilities of computers and how these capabilities relate to programming languages. Finally, it introduces the concept of an operating system and how the user accesses the computer. This lays the groundwork for using the computer while learning the fundamentals of BASIC in the next chapter, where, through an example and a few simple definitions, the approach to developing programs is introduced that is examined in detail in Chapters Three to Seven.

Chapter Two introduces all of the fundamental statements and expressions of BASIC; only file statements, arrays, formatted printing, and matrix

此为试读,需要完整PDF请访问: www.ertongbook.com

statements are deferred until later chapters. The need for each of the statements is illustrated by small problems that are then solved with short sequences of statements. In order to discourage a student from rushing ahead and thus developing poor programming habits, we avoid complex problems, long programs, and intricate control structures in this chapter.

Chapter Three introduces the three elementary control constructs of structured programming: sequential execution of statements, alternation of the selection between alternatives of the IF-THEN-ELSE statement, and repetition in the WHILE-DO form and the REPEAT-UNTIL form. These concepts are explored at an intuitive level without entering into the many theoretical arguments surrounding the pros and cons of various constructs.

Chapter Four introduces flowcharts. Because the methodology for developing procedures (presented in the text) works more naturally with English language procedures, flowcharts are seldom used in other chapters. However, flowcharts are generally used and are likely to be encountered by students, and therefore are entitled to a short chapter.

Chapter Five discusses the translation of procedures developed using the structured constructs of Chapter Three into BASIC programs. Standard patterns are developed for each of the constructs. Although in most instances these patterns are quite straightforward, some restrictions of BASIC require intricate sequences of statements. In particular, most versions of BASIC do not allow compound logical expression involving AND or OR. This means that a problem requiring control of a repetition of alternation by a compound logical expression must be implemented with a nest of BASIC IF statements, which can be frequent sources of errors. Rather than leave such constructs to a student's own ingenuity, several examples are worked out in detail using standard patterns of statements.

Chapter Six examines several strategies for testing and debugging programs.

Chapter Seven applies the material of Chapters Two, Three, Five, and Six to an actual problem that is longer than those given previously. A procedure and program are developed for the problem with considerable discussion of what is happening and why.

Chapters Eight to Eleven explore more advanced features of BASIC: files, arrays, print control, and user-defined functions. Because the implementation of files differs significantly among the many versions of BASIC, the concepts of both sequential and direct access files are first discussed in a machine-independent manner and then are examined in detail for Hewlett-Packard BASIC and DEC PDP-10 BASIC. These chapters are independent and may be covered in any order.

The remaining four chapters each present applications areas: matrices, sorting and searching, numerical methods, and modeling and simulation.

The chapter on sorting and searching covers several modifications of the insertion sort for internal sorting and the binary search algorithm. The chapter on matrices presents the matrix commands of BASIC and applies them to a problem of finding paths in a network and also to a problem of finding the solution to a set of linear equations. The chapter on numerical methods examines the bisection method for finding the zeros of a function and the trapezoidal rule for finding the area under a curve. The chapter on modeling and simulation develops a simple predator-prey model together with a program for its simulation, and develops a random walk model as an example involving the use of random numbers. All of these chapters make use of arrays, and the material on numerical methods makes use of a user-defined function. These four chapters are independent of each other and may be studied in any order.

I am most appreciative of the considerable encouragement and the many useful suggestions and criticisms offered by my editor at Wiley, Gene Davenport, and by the representatives and reviewers of Wiley. The valuable suggestions made by colleagues who patiently taught from earlier versions are gratefully acknowledged.

Peter Moulton

CONTENTS

此为试读,需要完整PDF请访问: www.ertongbook.com

ONE

INTRODUCTION

Once upon a time there was an estimator working for the Universal Floor Covering Company. The estimator's job was to talk to customers at the front desk and to provide estimates of the cost and time required to install floor covering. Because the Universal Floor Company produced a truly universal product, there was one price of $1.50 per square foot (the only option was padding at $0.25 per square foot), and installation became so routine that installation time could be estimated accurately at 100 square feet per hour. Installation charges were $7.50 per hour. Although the procedure of estimating became quite easy to remember and to follow, the estimator was a methodical person who wrote down the following sequence of steps:

1. Obtain the dimensions of the room and ask if padding is desired.
2. Compute and remember the area of the floor to be covered.
3. Compute, remember, and tell the customer the cost of the floor covering.
4. If padding is desired, compute, remember, and tell the customer the cost of the padding.
5. Compute, remember, and tell the customer the installation time.
6. Compute, remember, and tell the customer the installation charges.
7. Compute and tell the customer the total cost.

One day the estimator did not report for work, and one of the newer employees from the loading dock was recruited to fill in for the day. Fortunately, the estimation process was documented. However, the new employee did not know how to compute the area, and the costs and methods of computing the other steps were not clear. After a few minutes of talking with the new employee, the manager of the sales department sat down and wrote the following clarification of the estimator's procedure:

1. a. Obtain the LENGTH of the room from the customer.
   b. Obtain the WIDTH of the room from the customer.
   c. Ask if padding is desired.
2. Compute and remember AREA = LENGTH × WIDTH.
3. a. Compute and remember COST = AREA × 1.50.
   b. Tell the customer the COST.
4. If padding is desired
   a. Compute and remember PAD COST = AREA × 0.25.
   b. Tell the customer the PAD COST.
   Otherwise PAD COST = 0.
5. a. Compute and remember TIME = AREA/100.
   b. Tell the customer the TIME.
6. a. Compute and remember CHARGES = TIME × 7.50.
   b. Tell the customer the CHARGES.
7. Compute and tell the customer TOTAL = COST + PAD COST + CHARGES.

Fortunately, the new employee did understand the mathematical notation and was able to follow these steps. The problem was solved; the day was saved.

Both the estimator and the new employee faced the same problem—estimating the cost of floor covering for a customer. In both cases the solution to the problem lay in having a *procedure* to follow, a sequence of steps which the person could understand and carry out. A more general and more abstract definition of a procedure is:

> *A procedure is a finite set of steps or instructions which can be mechanically interpreted and carried out by some agent.*

The estimator's procedure consisted of seven steps which could be mechanically carried out by the estimator as the agent. However, these steps were not a procedure for the new employee who was not able to interpret them. The original procedure had to be refined and clarified for the new agent. We will be concerned with solving problems by creating procedures

for which the computer will be the agent. This will require refining and clarifying the procedures to a level of detail and into a language which can be mechanically interpreted and carried out by the computer. A procedure which is to be carried out by a computer is a *program*.

We encounter procedures all around us: instructions on the front of vending machines, on coin telephones, steps to be taken in order to enroll in a course, detailed instructions on how to complete a tax form, recipes in a cookbook. In each case, a finite set of instructions has been prepared for a particular class of agents with the expectation that these agents will be able to interpret and carry out the instructions mechanically, that is, in the absence of the creator of the procedure. If the procedure is an appropriate one, the agent will not have to ask questions and request clarification. Of course in the real world the unexpected frequently arises, as evidenced by the presence of tax consultants, registration advisors, telephone operators, and angry notes taped to the front of malfunctioning vending machines.

1.1 LANGUAGES FOR PROCEDURES

Both the estimator and the sales manager described procedures using English together with some elementary mathematical notation. Most everyday sets of instructions may be communicated in our natural languages. However, there are many times when our everyday language is found to be ambiguous, imprecise, or cumbersome. Compare the language of the following procedure for preparing beef stew with standardized recipe language of today.*

> BEEF STEW: Take very good Beef, and slice it very thin; and beat it with the back of a knife: Put it to the gravy of some meat, and some wine or strong broth, sweet-herbs a quantity, let it stew till it be very tender; season it to your liking; and garnish your dish with Marigold-flowers or Barberries.
>
> *The Closet Opened*, 1667

The recipe language of today contains standardized terms and abbreviations which must be learned by aspiring cooks.

Other examples of languages that have been developed to avoid the difficulties of natural language are those used for knitting instructions, moves in chess, and diagrams of football plays. Each of these languages has been developed for describing procedures for particular sets of agents: cooks, knitters, chess players, football players.

*From *Early American Herb Recipes*, by Alice Cooke Brown with permission of Charles E. Tuttle Co., Inc.

A language designed for describing programs is a programming language.

1.2 CAPABILITIES OF COMPUTERS

Before we begin to study a programming language, it will be useful to spend some time on the nature of computers. At the present time we would not expect to tell our computer, "Go to the store and buy a quart of milk." This is not something we would expect to be able to say in a programming language. Just what can a computer do?

One answer might be that a computer computes. A more general answer is that a computer manipulates symbols or information. But just what does this mean? A very simple but good example of a problem to which a computer might be applied is that of the floor covering estimator; and we might investigate the operations involved in solving that problem. Realizing that a computer is more simple-minded and mechanical than a new employee from the loading dock, let's look at the more detailed procedure.

First of all the estimator does compute. It is necessary to compute area, hours required for installation, cost of installation, and so forth. A computer has the capability to perform arithmetic computations: addition, substraction, multiplication, and division. One of the basic components of a computer is called the *arithmetic unit*, and it is this component that performs these operations. This unit is also called *arithmetic-logic unit* because it can also compare values in order to compute logical relations such as equal to, greater than, and so on. These logical operations are necessary when questions arise, such as "Does the customer want padding?" This is equivalent to "Is the customer's response equal to 'yes'?." 'Yes' is an example of nonnumeric information.

Secondly, the procedure indicates that certain values are to be remembered and used in the computations at later steps. A computer has a component called the *memory* or *storage* which contains a large number of *memory units* or *storage units*, each of which can hold one value or one item of information. Values computed in the arithmetic unit can be stored in a memory unit. This storage operation is called *assignment*. In the more detailed form of the estimator's procedure, words in capital letters, such as LENGTH, are used to name items of information. Each such item of information would have an associated storage unit which contains the value of the item of information. The storage unit can be imaged as a box, enabling us to illustrate it as in Figure 1.1 to indicate that the value of the LENGTH is 10.

Because each memory unit can hold only one value, when a value is assigned to a memory unit, any previous value in that memory unit is over-

Figure 1.1. The storage unit associated with the name LENGTH containing the value 10.

written. When a value is used in a later computation, it is brought from memory to the arithmetic unit. This operation does not remove the value from the memory unit, it only copies it into the arithmetic unit. The value remains unchanged in the memory.

But how do values get into the computer in the first place? How does the estimator get values? In two ways. Some values, such as the price of floor covering, are included in the procedure. Just so, some values will be included in a program. These values are called *constants*. Other values, such as the length and width of a room, are obtained from outside sources such as a customer. A computer must be able to obtain information from the outside world and does so through its *input unit*. The input unit connects with a variety of input devices. Some of the commoner devices are:

1. *Card readers*, which are capable of translating configurations of rectangular holes in paper cards into patterns of electronic signals.
2. *Paper tape readers*, which are capable of translating configurations of round holes in paper tape into electronic signals.
3. *Teletypes*, which are capable of translating a mechanical displacement of a key into electronic signals.
4. *Magnetic tape units*, which are capable of translating magnetized patterns on iron oxide coated plastic tape into electronic signals. In each case these electronic signals are directed to the memory where the information is stored and made available to the other components of the computer.

In order to tell the customer the result of a computation, the computer must send electronic signals through its *output unit* to a device that will translate these signals into some physical medium such as holes in cards or paper tape, or an impact of a type ball onto a page, or magnetized patterns on a magnetic tape.

Finally, the computer must have some way of proceeding through the steps of a program. The component that provides this capability is the *control unit*. The individual instructions of a program are encoded and stored in memory. The control unit keeps track of the location of the next instruction to be performed, reads that instruction from memory, interprets it, and directs the other components of the computer to perform the indicated operation. The function of the control unit distinguishes a computer from a cal-

culator. Most calculators require a human operator to provide the sequencing from step to step. In order for the instructions of the program to be accessible to the control unit, they must be stored within the computer itself. This requires encoding each type of instruction as a unique number. Then the sequences of numbers corresponding to the sequences of instructions of the program are placed in sequential storage locations in the computer's memory. This concept was very significant in the development of computers and led to the term *stored program computer*. Of course any value placed in a storage unit may be changed by a sequence of instructions assigning a new value to that storage unit. Thus, an important corollary of the stored program concept is that a program stored in memory can be read and be altered by operations of the computer itself.

A second very important feature of the control unit is that it is capable of detecting conditions resulting from arithmetic or logical operations in the arithmetic unit and then conditionally selecting the next instruction to be performed. It is this feature that allows a program to test the equality of the customer's choice of padding against "yes" or "no," and then to add the cost of padding or not according to the result of the test.

The components of a typical computer may be diagrammed as:

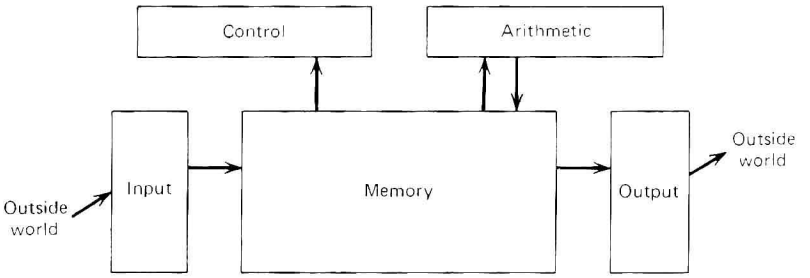

Figure 1.2. The major components of a computer. The arrows indicate flow of information, both data values, and instructions.

Computers are really very simple devices. Their power comes from three very important characteristics.

1. They are very fast. Even though a single machine instruction represents a relatively simple operation, a typical computer can perform a million such operations in 1 second.
2. They are very accurate and reliable. Although it is possible for operations to be performed incorrectly and for information to be lost, such errors are extremely infrequent, and many internal checks are made for most possible errors.

此为试读,需要完整PDF请访问: www.ertongbook.com