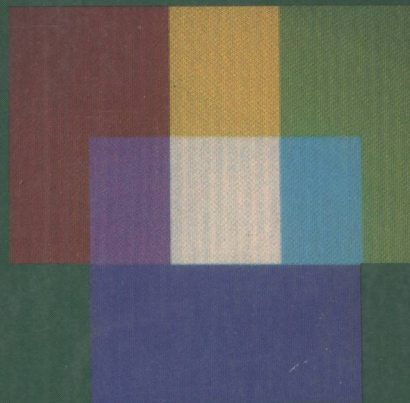


Wilson T. Price

Programming the IBM Personal Computer:

Business BASIC



TP31
P13

8564671



E8564671

Programming the IBM Personal Computer:

BUSINESS BASIC



Wilson T. Price

HOLT, RINEHART AND WINSTON

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

150000

IBM[®] is a registered trademark of International Business Machines Corporation.

The cover illustration is The Additive Color Model from Miles Color Art, Tallahassee, Florida, prepared using the Digital Facsimiles process (patent pending), Center for Color Graphics, Florida State University, Tallahassee, Florida.

Copyright © 1984 CBS College Publishing
All rights reserved.
Address correspondence to:
383 Madison Avenue, New York, NY 10017

Library of Congress Cataloging in Publication Data

Price, Wilson T.
Programming the IBM personal computer.

Includes index.

1. IBM Personal Computer—Programming. 2. Basic
(Computer program language) 3. Business—Data Processing. I. Title. II. Title:
Programming the I.B.M. personal computer.
QA76.8.I2594P75 1984 001.64'2 83-22797

ISBN 0-03-063746-5

Printed in the United States of America
Published simultaneously in Canada

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

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

Preface

Virtually every Basic book that comes on the marketplace is described as using structured programming techniques. For most of them the use of structured methods is relegated to lip service. A few do a good job of sticking to the three basic structures (sequence, selection, and looping) in presenting solutions to programming examples. However, the nature of Basic makes it difficult to write programs that are truly structured. For instance, a Basic program of any substance that does not use a **GOTO** is difficult to imagine. Yet it *is* possible to incorporate use of the **GOTO** into a framework that achieves the goals of good program design and functions within the three basic components of structured programming. That is the top priority in this book.

This book is not one on learning Basic; it is one on *how to systematically solve business-related problems while learning Basic*. From the first page of the book the stress is on organizing a problem into logical components: *program modularization*. Heavy emphasis is put on breaking problems into components and

then programming the components relatively independently of one another. Programming methods used in this book draw heavily on proven techniques now widely used in structured Cobol programming. For instance, in business data processing, there is a wide class of problems that can be broken down to three main modules:

- Initialization module
- Processing module
- Termination module

Normally, the processing module (which may consist of multiple submodules) is executed repeatedly until there is no more data to process. To this end, the **PERFORM-UNTIL** is commonly used in Cobol. Programs in this book use precisely the same approach with the **WHILE/GOSUB** combination. In fact, the **GOSUB** is a central focus from almost the very beginning.

This significantly reduces the need for using the **GOTO** statement. Overall, specific constraints are placed on use of the **GOTO**. In a nutshell, it must be used within the limit that every routine or sequence of statements has a single entry point and a single exit. Of course, there are a few exceptions to this, necessitated by the nature of IBM PC Basic. (System error trapping is the main exception.) Needless to say, good judgment must be used in every situation; after all, the objective is to learn good programming methods—not to contrive program structure solely for the purpose of being able to say, “Look, it is structured.”

Overall, the attempt has been to make this a reader’s book. Some of its important features are:

1. In all examples, modularization of programs is emphasized. This is especially important to the beginning student when covering advanced topics where example programs necessarily become much larger than earlier examples. Not only does modularization provide the student with examples that are easy to follow but it illustrates programming techniques that are critical to writing good, easy to maintain, and readable programs.
2. Emphasis is placed on using structured techniques and minimizing use of the **GOTO** statement. Because IBM Personal Computer Basic does not include a true block **IF-THEN-ELSE**, this concept is simulated to maintain good, structured techniques.
3. Each new topic is introduced through and oriented around a simple example. Each example is described in detail, its implications are discussed, it is solved, and important features of the solution are discussed. The progression is always from the concrete to the abstract.
4. As a rule, example programs are kept short—with primary focus on the topic being introduced. For the beginning reader the value of an example program or sequence in illustrating a concept is felt to be inversely proportional to the amount of code that is extraneous to that particular concept.

5. Each chapter includes “mind-jogger” exercises within the chapter; the answers to these exercises are at the end of the chapter. The intent is to provide a reinforcement vehicle for important concepts. In many cases they relate the significant point(s) concerning an example program and give the student a better insight to some of the fine points. The reader should be urged to complete each exercise as it is encountered and then refer to the answer to be certain of comprehending it.

6. Chapter 11 is devoted to random file handling. The intent of this chapter is to present the general topic of random processing in a simple fashion. To this end, blocking of records and other sophisticated techniques are ignored. However, the main focus—basic principles of random processing—is intact.

7. The book contains eight appendixes, which include reserved words; the ASCII character set; summaries of IBM PC Basic commands, statements, functions, and error messages.

Concerning the topical organization of this book, the detailed table of contents speaks for itself. However, some important points may not be evident from this list.

First, many of the early concepts of Basic are presented in the context of a simple inventory-reporting application. Chapter 1 involves the processing of a *one-record* data file (using the **READ** and **DATA** statements) with a five-line program. Thus the reader is given a first exposure to something complete and meaningful, albeit limited and contrived. Fundamental concepts introduced involve both entering and running a program and how input, output, and calculation operations work in Basic. Chapter 2 expands upon Chapter 1; use of Basic commands is stressed. The most significant aspect of this chapter is the introduction of modularization of programs. It is here that the simple inventory problem (program) is broken down to a main module and three submodules: initialization, processing, and termination. Execution of these modules is achieved using the **GOSUB** statement.

Second, the concept of files is stressed throughout the book. After all, the user of a microcomputer is quite accustomed to the notion of “things” being stored on a disk. Thus basic principles of processing sequential files is introduced in Chapter 3. In addition to the very important early introduction of using sequential files, this chapter provides the automatic loop control vehicle essential to structured programming. The precise vehicles for this are the **WHILE-WEND** looping structure and the automatic end-of-file (EOF) detection built into the EOF variable. Thus a complete data file can be processed with a form such as

```
WHILE NOT EOF(1)
  GOSUB 3000           'Process routine
WEND
```

Third, the **GOTO** statement is downplayed and is used only within certain constraints. In fact, it is not even introduced until Chapter 4. This late introduction is not the result of contriving in the design of the book but rather because *it is simply not needed* until that point.

Fourth, because this is a *business* Basic book, emphasis is given to report generation and totaling—in particular, to group totals and level breaks.

Chapter 7 is designed to allow the reader simply to touch the surface (use the first portion of the chapter then get out) or to go into it in detail.

Acknowledgments

My first appreciation must go to my students, who have provided me with many of the ideas in this book and who have endured some of my “less than successful” experiments. Certainly, I also owe a real debt to my many colleagues, who have offered their support, ideas and criticism. I wish to express my special appreciation to Brete Harrison, senior editor, who has been outstanding in suggesting ideas for creative writing projects. I thank Steve Brown regional manager of Software Centres International, who offered numerous remarks and observations and provided me with valuable access to the Centres’ facilities. Regarding preparation of this manuscript, I had the good fortune of making the acquaintance of Jane Nishi, who has done a superb typing job for me. She has and richly deserves my very special thanks.

Finally, I wish to thank users of my books who have taken their valuable time to send me their reactions, observations, and critical remarks; many of them are reflected in this book. I will sincerely appreciate any and all comments about this book. They may be sent to me at Merritt College, 12500 Campus Drive, Oakland, CA 94619.

Wilson T. Price

8564671

Contents



Preface *xiii*

Introduction 1

BASIC CONCEPTS OF PROGRAMMING 2

 A Story of Success

 Horse Racing and Computer Programming

 Writing Computer Programs

TOP-DOWN PROGRAM DESIGN 6

 What Is Top-Down Design?

 An Example of Top-Down Analysis

STRUCTURED PROGRAMMING 9

 Introduction

 Flowchart Symbols

Flowcharting	
Structured Programming Constructs	
COMPUTER HARDWARE	15
Input/Output	
Processor	
Internal Memory	
Auxiliary Storage	
COMPUTER SOFTWARE	18
What Is Software?	
The Disk Operating System	
The Process of Programming	
1. Preparing and Running a Program	20
ACCESSING THE SYSTEM	21
The Keyboard	
Starting the Computer	
USING BASIC	24
A Bookstore Inventory	
Creating a New Program	
FUNDAMENTAL PRINCIPLES OF BASIC	26
Line Numbers	
Variables	
Variables in Memory	
INPUT/OUTPUT OPERATIONS	29
The DATA Statement	
The READ Statement	
The PRINT Statement	
OTHER PROGRAM STATEMENTS	31
Arithmetic Operations	
The LET Statement	
The END Statement	
The INPUT Statement	
Executable and Nonexecutable Statements	
2. Modularizing a Program	36
MAKING CORRECTIONS TO A PROGRAM	37
Inserting and Deleting Lines	
Listing a Program	
Error Correction	
Errors and Error Correction	
USING THE BASIC PROGRAM EDITOR	40
Editing Keys	
Making Corrections to a Program	
Error Detection and Correction	

MODULARIZING THE INVENTORY PROBLEM	43
Statement of the Problem	
A Solution	
Remarks in Basic	
String Variables	
Printing Headings	
Directing Output to the Printer	
The GOSUB Statement	
More on Arithmetic Operations	
DISTINCTION BETWEEN STATEMENTS AND COMMANDS	50
SAVING PROGRAMS	50
Temporary and Permanent Program Storage	
The SAVE Command	
The LOAD Command	
More About the LIST Command	
SUMMARY OF COMMANDS	53
Using Function Keys	
SUMMARY OF STATEMENTS	54
 3. Using Data Files	 56
BASIC CONCEPTS OF FILES	57
The Filename and Extension	
The Nature of Sequential Files	
Files and DOS	
Using Files on Drive B	
PROCESSING DATA FILES	60
A File-Processing Example	
Preparing a File For Use	
Terminating Use of a File	
The File INPUT Statement	
REPETITIVE PROCESSING	63
CALCULATING SUMMARY TOTALS	65
Incrementing a Quantity	
The Concept of Accumulating	
Expanding Example 3-1	
 4. Conditional Operations	 74
BASIC FORM OF THE IF STATEMENT	75
The Concept of the IF	
Relational Operators and Expressions	
Comparing String Quantities	
Execution of the IF Statement	
The IF-THEN-ELSE Form	

The GOTO Statement	
Combining the GOTO and the IF	
BLOCKS OF STATEMENTS	82
The IF-THEN Block Form	
An IF Within an IF	
The IF-THEN-ELSE Block Form	
ADVANCED PROGRAM LOGIC	88
Logical Operators	
Validity Checking	
Programming Techniques	
MULTIPLE STATEMENTS PER LINE	93
Program Lines and Screen Lines	
Multiple Statements per Program Line	
Multiple Statements in the IF	
USE OF IMMEDIATE MODE	95
Statements in Immediate Mode	
Interrupting Execution	
The STOP Statement	
 5. Variable Types	103
CHARACTERISTICS OF NUMERIC QUANTITIES	104
Types of Numbers	
Designating a Variable Type by its Name	
Declaring Variable Types	
MORE ON ARITHMETIC OPERATIONS	108
Mixing Data Types in Performing Calculations	
Integer Arithmetic	
INTEGERS AS LOGICAL VARIABLES	109
Saving Yes-No Information	
Using a Negation	
 6. Report Generation	113
POSITIONING FIELDS ON THE OUTPUT LINE	115
Use of the Semicolon	
The TAB Function	
EDIT MASKS	116
The Concept of the Edit Mask	
Simple Form of the PRINT USING	
Edit Images for String Fields	
EDITING A COMPLETE LINE	120
Format Planning	
A Program Segment	
Alignment of Headings and Detail Lines	

MORE ON THE PRINT USING	123
Inserting Commas in Numeric Quantities	
Trailing Minus Sign	
Floating Dollar Sign	
The Asterisk Fill	
FUNCTIONS	126
The Concept of the Function	
The DATE\$ and TIME\$ Functions	
Breaking a String into Parts	
The LEN Function	
CONCATENATION OF STRINGS	129
Editing a Social Security Number	
A TYPICAL REPORT	130
Problem Statement	
Program Planning—Example 6-2	
A Program—Example 6-2	
PRINTER PAGE CONTROL	135
Planning for Output	
Program Planning—Example 6-3	
A Program—Example 6-3	
 7. Group Totals	 141
PAGE TOTALS	142
Problem Definition	
Using Accumulators	
Program Planning	
A Program—Example 7-1	
GROUP TOTAL PRINCIPLES	148
Control Groups	
Group Total Logic	
A Group Total Example	
A Sample Program—Example 7-2	
PAGE CONTROL	159
A Variable Spacing Example	
A Sample Program—Example 7-3	
 8. Program Loops	 169
THE WHILE-WEND LOOP STRUCTURE	170
WHILE-WEND Loop Control	
General Form of the WHILE	
THE FOR-NEXT LOOP STRUCTURE	172
Counted Loops	
Single Entry—Single Exit Principles	

The FOR-NEXT Statements	
General Form of the FOR Statement	
A Flowchart Representation of the FOR-NEXT Loop	
PROGRAMMING EXAMPLES WITH FOR-NEXT	178
Calculating a Table	
Program Planning and Solution	
Nested Loops	
EARLY EXIT FROM A FOR-NEXT LOOP	185
Searching a Table	
The RESTORE Statement	
An Example Program	
RULES REGARDING FOR-NEXT LOOPS	188
 9. Subscripted Variables and Arrays	197
BASIC PRINCIPLES OF ARRAYS	198
Calculation of the Mean	
The Notion of Subscripting	
Subscripted Variables in Basic	
Selecting Array Names	
PROCESSING AN ARRAY	201
Program Planning—Example 9-1	
Reading and Storing Data Values—Example 9-1	
Searching the Array	
TABLE SEARCHING	205
A Table Search Example	
The Concept of Arguments and Functions	
Program Planning—Example 9-2	
Loading the Table	
Searching the Table	
A Program—Example 9-2	
TWO-DIMENSIONAL ARRAYS	210
Rows and Columns of Data	
Two-Dimensional Arrays in Basic	
A Program—Example 9-3	
 10. Advanced String Operations	219
THE LINE INPUT STATEMENT AND THE INSTR FUNCTION	220
The Concept of Word Processing	
The LINE INPUT Statement	
The INSTR Function	
A TEXT-FORMATTING EXAMPLE	223
Problem Definition	
A Program—Example 10-1	

OTHER STRING-PROCESSING OPERATIONS	229
The SPACE\$ and STRING\$ Functions	
Distinctions Between Numeric Data and String Data	
The STR\$ and VAL Functions	
MANIPULATION OF INDIVIDUAL BYTES	234
The ASCII Character Set	
The CHR\$ and ASC Functions	
An Example of Character Manipulation	
The MID\$ Statement	
Control and Other Characters	
 11. Random File Processing	242
BASIC PRINCIPLES OF FILES	243
Introduction	
The Nature of Sequential Files	
ASCII and Non-ASCII Files	
Storage Buffers	
Fixed-Length Records	
SOME ADDITIONAL LANGUAGE CAPABILITIES	248
Opening a Random File	
The FIELD Statement	
The LSET and RSET Statements	
The GET and PUT Statements	
RANDOM I/O PROCESSING	252
Problem Definition	
Program Planning	
Creating a Random File	
Building a Random File — Sequential Processing	
Accessing a Random File — Random Processing	
OTHER RANDOM FILE TECHNIQUES	259
Providing for Sequential Access	
Defining the Record Length	
The Concept of a File Directory	
BINARY DATA IN FILES	261
Mapping Numeric Quantities to String	
Single-Byte Numeric Quantities	
A Sample Application	
 12. Additional Topics	268
MENU CONTROL	269
The ON-GOSUB Statement	
A Simple Menu	

MULTIPLE PROGRAMS AND THE CHAIN STATEMENT	273
Basic Principles of the CHAIN Statement	
Additional Features of the CHAIN	
SCREEN CONTROL	277
The CLS and LOCATE Statements	
The COLOR Statement	
USER-DEFINED FUNCTIONS	280
BASIC ERROR-HANDLING CAPABILITIES	282
Basic Errors	
The ON ERROR Statement	
The RESUME Statement	
Automatic End-of-File Detection	
The ERR and ERL Variables	
On Error Routines in General	
 Appendixes	 291
Appendix I	Reserved Words 292
Appendix II	The ASCII Character Set 294
Appendix III	Summary of Basic Statements 296
Appendix IV	Summary of Basic Commands 303
Appendix V	Summary of Basic Functions 304
Appendix VI	Summary of DOS Commands 306
Appendix VII	Using EDLIN to Create Data Files 309
Appendix VIII	Selected Error Messages 310
 Index	 313

Introduction

BASIC CONCEPTS OF PROGRAMMING

A Story of Success

Horse Racing and Computer Programming

Writing Computer Programs

TOP-DOWN PROGRAM DESIGN

What Is Top-Down Design?

An Example of Top-Down Analysis

STRUCTURED PROGRAMMING

Introduction

Flowchart Symbols

Flowcharting

Structured Programming Constructs

COMPUTER HARDWARE

Input/Output

Processor

Internal Memory

Auxiliary Storage

COMPUTER SOFTWARE

What Is Software?

The Disk Operating System

The Process of Programming

OBJECTIVES

To many, the computer and its terminology are very foreign. Furthermore, the rigorous way of thinking imposed by programming has a tendency to throw many people for a loss. The purpose of this introduction is to provide the basis on which to begin learning the discipline of programming. Concepts and topics discussed in this chapter are as follows:

1. The basic notion of what programming a computer is all about. This is illustrated by a simple analogy.
2. The principles of top-down design. Very large and complex problems can often be nearly impossible to comprehend. However, if broken down into small, independent components (modules), they are easily understood.
3. The fact that structured programming is basically a set of rules for solving programming problems. The use of flowcharts, which are pictorial representations of the program logic, is described.
4. The computer itself, called *hardware*. The modern computer consists of input and output devices (for getting information in and out of the computer), memory, a control unit, and auxiliary storage.
5. Programs that make the computer work for us, commonly referred to as *software* systems. Specialized systems for controlling the computer are called *operating systems* and provide for automatic control of the computer.

BASIC CONCEPTS OF PROGRAMMING

A Story of Success

To gain a little insight into the world of the computer, let us consider the story of a very bright college student who had a strong dislike for conventional “work.” When faced with the prospect of financing his college education, he decided to do so by playing the horses—a means of earning an income that he considered infinitely superior to pumping gas or washing dishes. Since his hobby had long been horse racing, he had an extensive file of information on the horses that generally raced at the local tracks. This file consisted of a set of 5×7 cards: one for each horse. Each card showed the past record of the horse (including best running times), information on how well the horse did under various weather conditions, and so on.

The information file was the key to the student’s success. Prior to each weekend, he would select from his file the record of each horse running that weekend. Armed with this stack of cards, a summary form, a pencil, his brilliant mind, and his system, he would sit down to work (Figure 1). His system consisted of a series of calculations that gave each horse a performance score between 0 and 100. Past experience had shown that the horse with the highest score in each race was usually the winner.

The procedure the student followed was simple: