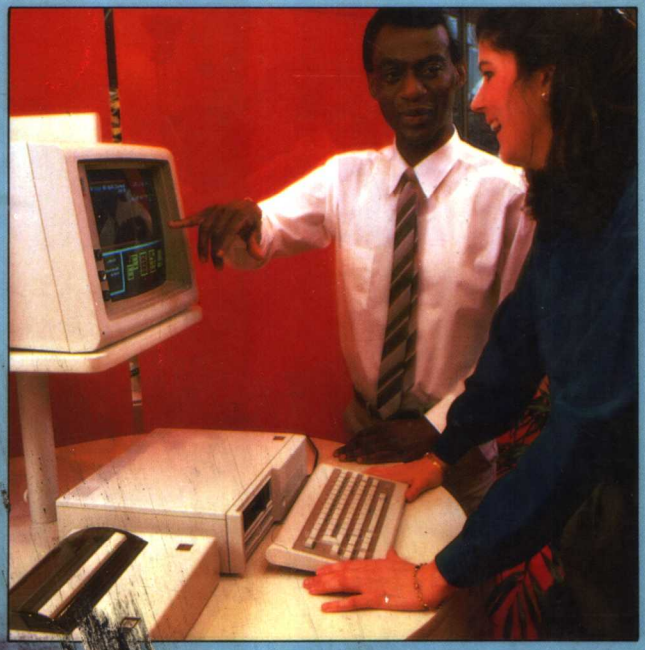
Gary G. Bitter

Paul M. Cook

# IBM BASIC for Business



# IBM BASIC for Business

Gary G. Bitter

*Arizona State University*

Paul M. Cook

*Education Consultant*

Prentice-Hall, Englewood Cliffs, NJ 07632

**Library of Congress Cataloging-in-Publication Data**

Bitter, Gary G.
IBM BASIC for business.

Includes index.
1. IBM Personal Computer—Programming. 2. Basic (Computer program language) 3. Business—Data processing. I. Cook, Paul M. II. Title.
HF5548.4.I24B57 1985 658'.055265 85-16759
ISBN 0-13-448093-7

*Editorial/production supervision:* Sophia Papanikolaou
*Cover design:* Photo Plus Art
*Manufacturing buyer:* Edward O'Dougherty
Photo Courtesy of International Business Machines



LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY:

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.



Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-448093-7 01

Prentice-Hall International, (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Whitehall Books Limited, *Wellington, New Zealand*

# Preface

Why BASIC? The acronym stands for Beginner's All-purpose Symbolic Instructional Code. If this is your first venture into the world of programming, then the keyword is "Beginner's." The BASIC language is very forgiving of the inevitable mistakes one encounters in the learning process. "Learning by mistakes" is an inherent part of this process and is particularly valid in learning computer programming.

Once the initiate has a fundamental grasp on the concepts involved in writing instructions to the computer, the techniques involved can be refined and a structured approach to programming developed.

BASIC is a very pervasive language. Most computers can be programmed in BASIC, and certainly all personal computers can.

There's a wide gap between interpreting a program, and writing one from scratch. This book attempts to bridge that gap with both the sequential presentation of materials and specific examples of concepts. The reader is led through the programming process from the initial concept of a program idea to the finished product.

Whether you're a seasoned veteran of the business world or a beginning business student, a high level executive or a small, one-person-business owner, this book is for you!

Although the IBM PC is fully capable of performing high level mathematical functions, those skills are not necessary to write sophisticated and useful programs. In fact, this book was written on the assumption that the reader has little or no experience with either a computer, or high-level math. Some mathematical concepts are necessary, but if you have no problem understanding,

$$A = 5$$
$$B = 6$$
$$A + B = 11$$

then you can program in BASIC.

Chapter 1, "Introduction" and chapter 2, "Getting Started with the IBM PC," serve to get you started by listing assumptions, explaining how to get the most out of the book, and familiarizing you with the IBM PC along with the fundamentals of operating it.

Chapter 3, "An Introduction To Programming," explains some ground rules of programming and how your IBM PC communicates with its disk drive.

Chapter 4, "Getting Information Into Your IBM PC," introduces you to two very common methods of getting data into a program. Variables are discussed, as are the rules used in naming them.

Chapter 5, "Decision Making with Your Computer," explains ways that you can have a program branch from one section to another when certain conditions are true. It is here that subroutines are introduced and their value shown.

Chapter 6, "Counters and Accumulators," shows you two very useful concepts that are applicable in many situations and procedures. Their differences are explained, as are their similarities.

Chapter 7, "How To Write a Program," introduces flowcharts and their value to the programming process. A simple program is created to show each step with a direct application.

Chapter 8, "Controlled Loops," introduces and elaborates on the FOR . . . NEXT statements and how to get the most out of them in creating and using controlled loops.

Chapter 9, "Using Data Stored Within Computer Programs," guides you through the powerful concepts of READ . . . DATA statements. These concepts are elaborated on to show the value and flexibility of applying them to solve a programming problem.

Chapter 10, "Lists and Tables of Data," explains how to set up and fill one-dimensional (lists) and two-dimensional (tables) arrays. Statements necessary to set up these arrays are also covered.

Chapter 11, "The Sort Process," shows the readers techniques that will allow them to write programs that will sort lists of data into numerical or alphabetical order.

Chapter 12, "String Manipulation," explains techniques used to have the computer manipulate information by segregating specific portions of it or determining the length of a string. Ways of calling up information by its "ASCII" code are covered, as are ways of joining different pieces of information.

Chapter 13, "Library Functions," introduces the reader to statements that will cause the computer to generate random numbers and how to determine the range from which it will select. Statements used for higher-level functions are explained along with examples of their use.

Chapter 14, "Formatting Your Programs and Output," is a particularly useful chapter in that it illustrates techniques of arranging the displays produced by a BASIC program. Menus are explained, and ways of incorporating passwords into programs are demonstrated.

Chapter 15, "Creating Data Files," takes you into an advanced capability of BASIC. Creating data files will allow you to store information produced or collected

by a program into a file created on a disk for "permanent" storage and later reference. Two types of data files are explained, along with demonstrations of both.

In addition to these chapters, nine appendices are used to present such supplemental information as the use and care of disks, the function keys, ASCII codes, reserved words, and error messages. Also covered in the appendices are sections on editing and a glossary of statements, commands, and functions. Finally, a section that lists and describes the four main types of business programs that are commercially available and used by the business world is included.

One thing BASIC is *not* is a spectator sport. If this book covered how to ride a bicycle, the only way you could actually learn would be to get on a bike and peddle. No matter how comprehensive or exhaustive a book on programming is, you must get your hands on those keys and try it! On the other hand, computer programming is not a good subject for speedreading. You may jump around from chapter to chapter if you feel comfortable with the material, but concepts are presented in a sequential way and later chapters build on concepts presented earlier in the book.

Finally, learning should be fun! Not necessarily easy, but fun. As you progress in your new skills, you'll begin to see all of the pieces begin to blend into an overall, cohesive picture of BASIC programming on your IBM PC.

Now, dig in and have fun!

Gary G. Bitter
Paul M. Cook

# Acknowledgments

Many people contributed to the generation and production of this book. Although the entire list would be prohibitive to print here, the authors would like to thank Marcia Horton, our editor, who helped to create a smooth transition from initial conception, to finished product. Her help and guidance were invaluable and greatly appreciated.

We would also like to thank Kay Bitter and Carol Clark for their relentless support, encouragement, and patience during the long hours required to "bring it all together."

Thank you.

Gary G. Bitter
Paul M. Cook

# Contents

# 1

# Introduction

## OVERVIEW

Beginning any new experience is filled with both excitement and apprehensions. So let's begin at the beginning. First you'll see why this book is for you. Next a brief look at programming in BASIC and how to get the most out of this text. If you are a busy businessperson or professional, you'll want to know why you should take the time to write your own programs. Chapter 1 concludes with a look at the three major categories of computers.

## 1-1 ASSUMPTIONS

There are three basic assumptions.

1. You have little or no experience in programming a computer.

The breakthroughs and innovations of the past few years have put very powerful microcomputers within the reach of the general public. It's been said that if the automobile industry had made equal strides in the development of their technology, you could buy a Rolls-Royce for $2.54 and drive it from San Francisco to New York on one gallon of gas. An astounding analogy, but very illustrative of the advances in this area. Now, even small businesses can take advantage of the powerful capabilities of a computer to handle and process large quantities of data. We're now in the midst of the information revolution, which promises to surpass even the technological revolution that had such an impact on mankind during the last century.

Even though today's microcomputers are quite sophisticated, these computers are easy to operate. Modern programming languages make the task of programming a microcomputer relatively easy.

2. You are interested in using microcomputers to solve business problems.

Power has shifted from an industrial base to one of information. In business, operating theories work on an assumption of "perfect information," which, in reality, is very seldom obtained. Computers make possible the rapid processing and retrieval of information that puts us a little closer to that goal of "perfect." The more information you have, the better the decision-making process becomes.

Whether you are in business now or plan to enter the field of business in the near future, you will be given opportunities to take advantage of the power of a computer by learning to write your own programs.

3. You have access to an IBM Personal Computer with a single disk drive and the IBM disk known as *PC DOS*.

You could read a multitude of books on how to drive a car, but until you get into one and start driving, you'll never develop the necessary skills to put it to use. The same goes for programming and getting the most out of a microcomputer.

You will need at least one disk drive because most business applications require the storage and retrieval of programs and data. Once a program is typed into your computer, you must have some means of saving it for quick and reliable retrieval. If you had to type in the program each time you wished to use it, you would waste more time than the computer can save you. That's the real power of a computer: the fast manipulation, processing, and retrieval of information (data).

## 1-2 WRITING A PROGRAM

To make the computer do some of those astounding tasks, it must be programmed. What is a program?

> PROGRAM—An explicit set of instructions for performing certain tasks or for solving a certain problem, written and communicated in a form understood by the computer. *T

These instructions tell the computer what to do and in what order to do them. One instruction tells the computer to print values of calculations, while another allows

new data to be entered from the keyboard. Humans communicate with each other using a language that each understands in a given society. To communicate with the computer, you must use one of the many languages that the computer can translate into a form it understands. The language we will use is BASIC.

Why BASIC? The acronym BASIC stands for Beginner's All-purpose Symbolic Instruction Code. The language was originally designed to teach beginners programming logic. BASIC is easy to learn, yet powerful enough to handle the tasks that most small business computer users will want the computer to do. BASIC may not be quite as fast as machine language, but machine language is too difficult for the average business owner or employee to learn in a short period of time. Other languages have been, and will continue to be, used in the business community. Yet, BASIC remains the number one programming language used on microcomputers.

BASIC can be used on many different computers. Different computers use different versions of BASIC. Some computers can use several different versions of BASIC. Yet, most programs written in one version of BASIC will not RUN on a computer using a different version of BASIC, whether the same computer or a different one is used. Once you learn one version on BASIC, learning another version is usually quite easy. Programming logic and techniques are the same, with the difference being mostly in the syntax of the language. We will use the IBM-PC version of BASIC.

## 1-3 GETTING THE MOST OUT OF THIS BOOK

Most of the material is presented in the following format. Chapters will begin with an overview. Then, concepts will be presented at a level that will enable you to understand the required programming skills. Once a concept has been discussed, questions will be presented, followed by programming exercises.

You will also encounter boxed-in definitions throughout the book. These definitions will be followed by one or more symbols to indicate what type of definition they are. The symbols used and their meaning are

*C = A command
*S = A statement
*T = A term

Some definitions may be applied as either a statement or command. These will be indicated by both symbols.

In the back of the book, you will find the following appendices:

**A.** Additional exercises
**B.** Use and care of disks
**C.** Editing
**D.** The Function Keys

**E.** ASCII codes
**F.** Reserved words
**G.** Glossary of Statements, Functions, and Commands
**H.** Error messages
**I.** Brief descriptions of four types of business programs and their capabilities

In order to give examples throughout the book, certain conventions must be observed. When you encounter an example such as

```
LOAD "<program name>"      [ ENTER ]
```

the "less than" symbol (<) and the "greater than" symbol (>) will be used to show a note and should not be entered on the keyboard. The note will indicate the general form of what you are to type. In this example, <program name> indicates you are to type the name of a program. Don't confuse this with the normal use of < (less than) and > (greater than).

The squared brackets ([ and ]) indicate that an actual key is to be pressed on the keyboard. In the example above, [ ENTER ] means to press the ENTER key on the keyboard.

Follow the sequence of the chapters, because later chapters assume you have learned previously presented concepts. This doesn't mean that you have to understand every detail of each chapter before proceeding to the next. When you feel that you have a firm grasp on the material in one chapter, do the exercises, and go on. As you progress through the book, you'll begin to see how each of the chapters fits together into an overall picture of BASIC programming.

Answer the questions, and do the programming exercises. They are designed to reinforce what you have just learned. Programming can only be learned by actively writing your own solutions.

No two programmers will write a program the same way to accomplish the same purpose. That's one of the exciting facets of programming: there's no one right way to write a program.

## 1-4 WHY WRITE YOUR OWN PROGRAMS?

One justifiable argument asks why someone should learn to program at all. There are a multitude of commercial programs already written to handle most needs. This is a fair argument, and you may be able to find what you need commercially available, but there are several points that need to be considered.

1. Most commercial programs are written for a general market. You will need some of these programs. The accounting packages will handle standardized accounting practices. Yet, many businesses have specialized needs and/or particular ways in running the business. These "special" business operations are often not profitable for commercial companies that sell programs.