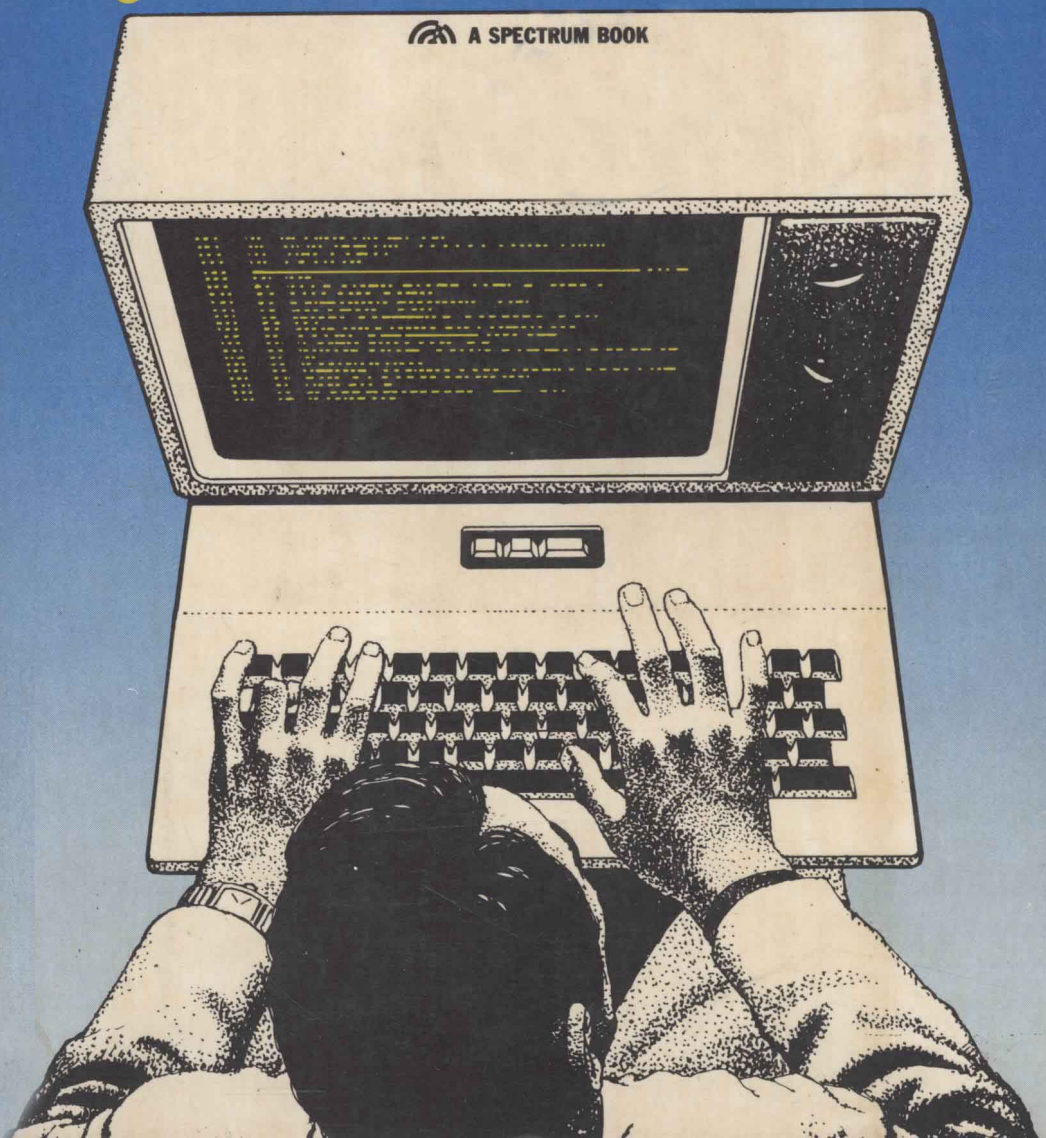
Albert L. Peabody/Richard H.C. Seabrook

dBASE II®
PROGRAMMING

Making dBASE II Work for Your Small Business

A SPECTRUM BOOK



dBASE II PROGRAMMING

Making dBASE II Work for Your Small Business

Albert L. Peabody

Richard H. C. Seabrook

Prentice-Hall, Inc., A SPECTRUM BOOK Englewood Cliffs, N.J.

Library of Congress Cataloging in Publication Data

PEABODY, ALBERT L.
dBase II programming.

"A Spectrum Book."
Includes index.
1. dBase II (Computer program) 2. Data base management. I. Seabrook, Richard H. C. II. Title. III. Title: dBase two programming. IV. Title: dBase 2 programming.
QA76.9.D3P38 1984 001.64'2 83-27240
ISBN 0-13-196148-9 (pbk.)
ISBN 0-13-196130-6 (pbk. and disk)

To the two Elizabeths

This book is available at a special discount when ordered in bulk quantities. Contact Prentice-Hall, Inc., General Publishing Division, Special Sales, Englewood Cliffs, N.J. 07632.

A SPECTRUM BOOK

10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

Editorial/production supervision by Jane Zalenski
Cover design by Hal Siegel
Manufacturing buyer: Joyce Levatino

ISBN 0-13-196148-9

ISBN 0-13-196130-6

Prentice-Hall International, Inc., *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
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*
Editora Prentice-Hall do Brasil Ltda., *Rio de Janeiro*

PREFACE

This book will teach you how to write programs in dBASE II's powerful internal programming language. It is designed for people who need to write small business program systems—for themselves or others. The book covers the general principles of program design, which are applicable to any language, as well as the features and instructions of dBASE II language.

You will find that the book is written in a conversational style to make it easy to read, but it is rigorously accurate in its descriptions. Examples are taken from a real, working Order Entry/Inventory system to make the principles come alive.

For the first time, this book applies the principles of good program design and coding to dBASE II language, showing with concrete examples how you can write readable, accurate programs to really make dBASE II work for you.

Our objective in writing this book is to present dBASE II's features and instructions in a logical way, sharing with you what we have learned in developing commercial programs in this exciting new language. Each of the first 4 chapters presents an important aspect of the programming process; Chapter 5 describes the characteristics of dBASE II; and chapters 6-11 teach you how to apply the principles of chapters 1-4 to dBASE II in a logical manner. When you finish reading the book, you will be ready to write your own useful systems of dBASE II programs.

INTRODUCTION

dBASE is the most powerful data base manager available for microcomputers today. Using dBASE II's simple commands, you can create data bases of information of interest to you, edit and sort that information, and generate reports based on that information.

This "immediate mode" operation from the terminal, however, represents only half of the capability of dBASE II: *almost hidden within dBASE II is a powerful programming language* which most dBASE II owners never learn to use.

The purpose of this book is to take you beyond the immediate mode of dBASE II and to teach you how to produce programs and systems of programs in dBASE II's internal language but it is more than a dBASE II programming handbook. It discusses the programming process in general, including programming techniques applicable to any programming project in any language.

When you finish this book, you will be ready to write programs in dBASE II language. You will understand data bases, their uses in practical programming, and the special ways dBASE II uses them to help you make your computer work for you, especially in small business applications.

DATA BASE MANAGEMENT SYSTEMS

The concept of a *data base* (sometimes written *database*) is in the process of evolution. Some people use it to refer to all the information that an organization owns and consider it an asset of the organization—as furniture, inventory of goods, and cash in the bank are assets. Other refer to any collection of information that is in some way related as a "data base."

We will use the term *data base* to refer to all the information that has been entered into a computer. Therefore, when we say information is "included in the data base," we mean that information has at some time been entered and is now available to the computer.

The term *data base file* will be used to refer to a collection of information in the computer related to a single subject. Examples include:

a customer list

a parts list, with part numbers and descriptions

the itemized sales for the past week (or hour)

All of these examples share certain properties. They all contain information important to you. They all contain items of information (or data items) such as "10 in. Crescent wrench," "stock # CW 1043-9" and "retail $12.98" which are somehow related to the other data items in the same data base. In fact, in virtually every case, the data items in a data base could be represented in the form of a table, often called a *relation.*

A TABLE OR "RELATION" OF DATA ITEMS

Rec #	*Part #*	*Description*	*Retail*	*On Hand*
1	1047	6" Crescent Wrench	12.98	12
2	3395	16 Oz. Hammer	6.49	9
3	5698	4" Phillips screwdriver	3.98	25

A data base file consists of *records,* shown as lines in our table example, each of which contains a certain number of *fields,* columns in the table. The fields might be labeled "Part Number" or "Description" or "Retail Price" and might define what sort of information will be contained in that particular part of each record. The records are numbered and contain the information on a particular stock item, person, or invoice.

A data base file, then, contains an indefinite (and usually growing) number of records, each of which relates to a single part or person or whatever; each record contains a definite number of fields, each of which contains one data item.

Computer data base management systems (DBMS) are used because they can perform the tasks related to information management with ease, they are highly adaptable, and they guarantee accurate results.

Guaranteed accuracy is a startling and important concept. It does not mean that the use of a DBMS will guarantee that your programs will contain no errors. It means that a good DBMS handles the most troublesome and error-prone parts of computation within itself, requiring only that the user use the proper words to ask it to perform these tasks.

For example, the very process of storing information on disk in a computer is fraught with opportunity for error. In a non-DBMS language, the programmer must decide how much disk space will be allocated for each data item and where it will be. Furthermore, he must recall the various field names, lengths, and location each time a new program is written that will use the same data base file.

When a DBMS language is used, the programmer need only recall the file and field names; the DBMS language handles the details of storing and finding the data on disk. This is a major difference between DBMS languages and traditional programming languages such as COBOL or BASIC.

In addition to guaranteeing accuracy at the lowest and most tedious level, the DBMS greatly speeds up the development of applications, the programs that actually do the specific jobs that computers perform.

Computer data base management systems operate in various ways: some are entirely operated by picking selections from menus that appear on the screen. You can select "create file" from the menu, then tell the computer, item by item, what fields you would like to have in each record of the file.

Next, you can choose "append" from the same menu, and the DBMS program lets you add data to the file you have created. Likewise, the "edit" selection lets you change information already in the file, and the "report" selection asks you which data base file to use, which fields from each record in the file to include in your report, whether any fields are to be totalled or subtotalled, and so on.

These "menu driven" DBMS programs are generally quite easy to use, but limited in flexibility—only the pre-programmed functions included by their authors can be performed. Because they allow you to store information away, to sort and select it, and to print a limited range of types of reports, these programs can be very useful if your needs are very simple and straightforward.

Other DBMS programs are *command driven.* To use these programs, you must type in commands to perform the same functions you can select from the menu in a menu-driven DBMS. However, the best of these programs have a rich repertoire of commands, amounting to an entire high-level programming language of their own. With these commands, you can create entire *command files* which manipulate data in ways that could not be foreseen by any set of standard, menu-selected functions.

dBASE II

Ashton-Tate of Los Angeles, California, distributes what has become the "standard" command-driven microcomputer data base manager—dBASE II.

dBASE II is written in assembly language, which makes it quite fast. Originally written as an imitation of a large mainframe DBMS, dBASE II incorporates most of the features of the biggest and best DBMS programs while running on all but the smallest microcomputers. Though it is most useful when run on a computer with a hard disk, it will work (with speed and file size limitations) on a dual-floppy-disk computer.

dBASE II operates in two modes. In the *immediate* mode, you can enter simple commands to create, edit, sort, and update files, and then produce relatively simple reports from them, including only selected records which meet any desired criteria, with subtotals and totals of any

numeric fields. For example, you might ask the computer to "produce a list of all persons who owe more than $100.00 with original date of invoice more than 60 days ago, and give the grand total of money involved, with subtotals for each person."

In the *program* mode, the same commands that can be entered in the immediate mode are recorded in disk files, using either the editor provided with dBASE II or any word processing program. For example, the entire series of commands needed to process a list of orders and produce invoices and packing slips might be placed in a disk file called "orders." Then, in the immediate mode, you could type in "do orders," and the commands would be read from the disk file as if they had been typed in from the terminal.

Furthermore, in the program mode, command files (programs) can "call" other command files, so that a dBASE II programmer can build up whole systems of programs that simulate the kind of menu-driven DBMS described above, except that the functions performed are those selected by the programmer, not the more limited selection provided by the authors of menu-driven DBMS. This amounts to a "third level" of use that does not require you to know even dBASE II's relatively simple commands. This, in effect, transforms dBASE II into a sort of menu-driven file manager, while retaining the power of a command-language-driven, true data base manager.

PURPOSE OF THIS BOOK

There are several sources of information on dBASE II: the manual that is distributed with every copy, *Everyman's Data Base Primer* by Robert A. Byers, (Ashton-Tate, 1983), *Data Base Management Systems* by David Kruglinski (Osborne/McGraw-Hill, 1983) and numerous magazine articles and reviews in the computer press. However, none of these sources adequately discusses the process of creating workable, productive systems of programs in dBASE II's programming language.

It is the purpose of this book to provide a readable but precise course in dBASE II programming. The presentation is rigorous and complete, covering all commands and their interactions with the screen, printer, and disk files. A complete alphabetical index is included to make the book more useful as a reference aid as well as a textbook.

The book is intended for all persons who need to write in dBASE II language: those who have used dBASE II at the command level and want to progress to writing true programs, those who are familiar with BASIC, FORTRAN or other programming languages and want to learn dBASE II language, and those experienced dBASE II language programmers who would like to improve their technique and productivity.

The book is intended to be both helpful and easy to use. We have written it at a level simple enough for relatively unsophisticated programmers, but we have included advanced techniques as well. We assume no previous knowledge of programming or of dBASE II.

METHOD

Programming is a complex process consisting of a series of related steps, from definition of the problem to be solved, through program design, coding, and verification (debugging and testing) to maintenance of finished programs.

In this book, we present information useful to the dBASE II programmer in all of these stages. We begin with an overview of the programming process and then proceed to describe the specifics of dBASE II programming, using the example of an actual order entry/inventory tracking system written by us in dBASE II language to illustrate all stages of the process.

Our overview of the programming process presents a methodology of software development that is applicable to any programming effort, using any language. We consider the issues raised in this section to be of equal importance with those discussed in the section on the specifics of dBASE II programming. If you follow these principles scrupulously, you will probably produce better software than the brilliant technician who ignores them.

The section on the specifics of dBASE II programming takes you through all stages of the process of development of a specific, typical DBMS application. Each step illustrates the application of the principles of program development with concrete examples. Groups of commands are introduced as they are needed in developing the application from the top down. When you finish this section, you will be able to develop your own application programs in dBASE II language.

Along the way, we point out the strengths and weaknesses of dBASE II language and note some techniques to exploit the former and minimize the latter. We analyze actual programs that embody the tricks and techniques we have devised and teach you how to develop some tricks of your own.

This book emphasizes documentation. None of us is immortal; neither can we be present in two places at once. If you are unavailable when a problem arises concerning a program you wrote last year, you will not be able to solve it. (Actually, experience teaches us that even if you are right there, if you wrote the program more than a few weeks ago, you will not remember how it works.)

Accurate and complete documentation is absolutely vital in such cases. One of our major goals is to teach you simple techniques and disciplines which will keep your documentation up to date and complete. dBASE II will help you in this effort because it is partially self-documentating and naturally structured. However, you must participate in the process!

Finally, we have organized the book in the knowledge that it must serve as both a textbook and a reference book. To this end, the book follows the natural flow of software development, introducing elements of design, coding, and verification as they are needed. Also, it is thoroughly indexed and has an informative table of contents. Our aim has been to make the book useful to you in all stages of your development as a dBASE II programmer. We hope you find it helpful, and we welcome comments and suggestions for continued improvement in future editions. Send any comments to us care of the publisher.

Instructions for Disk with dBASE II Programming

The disk which accompanies this book contains all the programs described in the book, plus several other small programs required to complete the system and make it work. The disk is intended as a learning aid to help you learn to write your own program in dBASE II language, not a substitute for a customized program system designed around your particular business; however, it is complete, and will perform its functions. It would certainly serve as the nucleus of a system suitable to any business.

The disk contains no operating system; neither does it contain dBASE II itself. You must purchase the operating system and dBASE II separately if you are actually to run the programs. However, you may use any word processor or text editor, including the one which comes with your computer, to examine and modify the programs on the disk. They are recorded in straight ASCII (American Standard Code for Information Interchange) form.

If you own dBASE II and wish to try out the programs, you must either "boot up" your computer with a separate systems disk, or generate a system on the enclosed disk, following the instructions which came with your computer.

Next, start up dBASE II as you normally do. Assuming that your dBASE II disk is in drive A, the disk enclosed with this book in drive B, type the following from the dBASE II "dot prompt":

```
.SET DEFAULT TO B
.DO INVER
```

and that's all there is to it. You will see a sign-on message, then will be requested to enter the date. This will lead you to the main menu of selections of the inventory/sales tracking system. We have not set escape off, which means you can press the escape button on your computer at any time to return to the dot prompt so that you may examine or change any of the programs or data file structures.

CONTENTS

PART ONE

THE PROGRAMMING PROCESS

The process of programming includes much more than writing the actual lines of code that will make the computer do what you want. It is a complex process, consisting of the following separate but interdependent stages:

Problem Definition
Program Design
Program Coding
Program Verification

The next four chapters will analyze these steps briefly; their importance for the production of useful programs is so great, and the tendency among programmers to ignore one or more of them so strong, that it is worth the time and effort to survey each one.

chapter one
PROBLEM DEFINITION

No one of the four steps can be said to be more important than the others. All of them must be followed to produce a good program. However, it is the first two which are most frequently shortchanged in the programming process. All too often we think we understand the problem at hand, and we only learn the importance of problem definition when our programs fail to satisfy their users (even if we are those users!). And of course, since program design is not nearly as much fun as typing in programs, it is often skipped entirely.

OUTPUT REQUIREMENTS

The proper place to begin problem definition is at the end. In almost any commercial application, it is the end which determines the shape of the beginning and the middle. After all, if you didn't want your computer to produce something for you, some report or analysis, you wouldn't start programming at all. So the process of problem definition starts at the end, with the reports or other products desired.

It is important to note that different products may be desired by different people who will use the same programs, and that none of the people who are requesting the programming process may be fully aware of exactly what they want. Perhaps the warehouse supervisor feels a need for a timely listing of inventory on his shelves. He may have seen a similar report produced by another system, or may have simply decided on his own what information the report should contain.

But how does he know what information is available in the computer's data base? How does he know what alternate forms his report might take? How does he even know that what he wants is a periodic, printed report, rather than a terminal in the warehouse that could be used to query the data base and generate the same information in a more accurate and timely form?

Chances are, he doesn't. Nor does he know what information might be needed by the shipping department that might be produced simultaneously with little additional cost, or what other people in his own department might want from the same report.

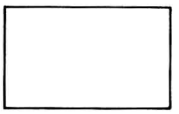
And so the process of problem definition must consider not only the products that are initially requested, but also the *needs* expressed in the original request, and it must continue with a complete analysis of other possible products that might meet these needs. It is an iterative process, consisting generally of interviews with those people who have requested that a program be written, a presentation of possible output forms to determine whether they will do the job, followed by more interviews with the same and other people who might use the products, a presentation of other possible forms of output, and so on.

The more times you go through this loop, the more you will learn about the real needs of the users. It never ceases to amaze us how much can be learned in this process and how great the distance between the initially requested report and the end product of the analysis process often is.

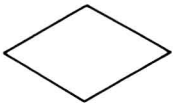
The importance of the third and subsequent iterations in this process and of the involvement of related personnel, in addition to the original requestor, cannot be overemphasized. Very frequently, the manager of a department has an excellent idea of what he needs to analyze the operations of his department, but he is not aware of the needs of lower-level personnel in day-to-day operation of the department. In fact, the lower-level personnel themselves may be unaware of their needs until they participate in the process.

Let's look at the process of designing output products, represented in a "flowchart" (Figure 1-1).

A flowchart is a document that traces the flow of action through a process. It consists of boxes, diamonds, and other symbols that represent processing steps and are connected by lines and arrows that indicate the order in which they occur. Traditionally, a box like this



represents a process, while a diamond



represents a decision point. Other symbols are used to represent data input through terminals, data storage disks, data output through printers or terminal screens, and all the other parts of the process.

Some authors believe that flowcharts are the only way the structure of a process can be represented in program development. Others believe flow charts are old fashioned and prefer other techniques such as Warnier-Orr diagrams or even topic outlines. We believe flowcharts are a convenient way