Pascal for the IBM PC

IBM DOS Pascal
and
UCSD p-System Pascal

Kevin W. Bowyer
Sherryl J. Tomboulian

Pascal for the IBM PC

IBM DOS Pascal
and
UCSD p-System Pascal

Kevin W. Bowyer
Sherryl J. Tomboulian

Robert J. Brady Co.
A Prentice-Hall Publishing and
Communications Company
Bowie, MD 20715

Pascal for the IBM PC: IBM DOS Pascal and UCSD p-System Pascal

Executive Editor: David T. Culverwell
Production Editor/Text Designer: Michael J. Rogers
Art Director/Cover Designer: Don Sellers
Assistant Art Director: Bernard Vervin
Cover photography: George Dodson
Typefaces: Melior (text), Fritz Quadrata (display), Typewriter (computer programs)
Typesetting: Creative Communications Corporation, Cockeysville, MD
Printed by: R.R. Donnelley & Sons Company, Harrisonburg, VA
Pascal portrait and Pascal computer courtesy of IBM Archives

Library of Congress Cataloging in Publication Data

Bowyer, Kevin, 1955-
Pascal for the IBM Personal Computer.

Includes index.
1. IBM Personal Computer—Programming. 2. PASCAL (Computer program language) I. Tomboulian, Sherryl, 1962- II. Title.
QA76.8.I2594B68 1983 001.64'2 83-3921
ISBN 0-89303-280-8

Prentice-Hall International, Inc., London
Prentice-Hall Canada, Inc., Scarborough, Ontario
Prentice-Hall of Australia, Pty., Ltd., Sydney
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, Petone, New Zealand
Editora Prentice-Hall Do Brasil LTDA., Rio de Janeiro

Printed in the United States of America

83 84 85 86 87 88 89 90 91 92 93 1 2 3 4 5 6 7 8 9 10

PREFACE

This book is an introductory text to Pascal programming on the IBM Personal Computer. It is meant for people who want to learn Pascal from the ground up. You should be able to use this book successfully even if you have never written a program before. If you do have programming experience, the first few chapters may seem to move slowly, but there is still much here for you. Many features specific to Pascal on the IBM PC are illustrated here in complete example programs. These features include graphics, sound, file handling, and other extensions to standard Pascal.

In addition to the numerous example programs, there are exercises and problems in the text to aid self-study. The exercises are meant to reinforce basic concepts, and the answers to the exercises follow them. The problems are a bit more ambitious and do not have accompanying answers.

There are already several dialects of Pascal available for the PC. The most popular is DOS Pascal, which runs under the IBM DOS operating system for the PC. The second most popular is UCSD p-System Pascal, which runs under the p-System. Neither of these is the same as what is called "standard" Pascal. We have consciously decided to cover both DOS Pascal and p-System Pascal in this text. There are several reasons for this. One is simply that it makes the book useful to a wider audience than if only one dialect of Pascal is covered. Another is that many of the most useful features of any Pascal are "nonstandard." By discussing two different approaches to common extensions of standard Pascal, we hope to make the reader aware of its limitations. The areas where DOS Pascal and p-System Pascal differ are things that may have to be relearned when switching to Pascal on any other operating system or computer.

Chapter 1 should get you oriented to where you and your configuration of the PC fit into the world of Pascal. Read this chapter carefully. Chapters 2 and 3 cover the very basic features of Pascal, including input and output. It is here that you will run your first programs. Chapter 4 covers control structures for selection and Chapter 5 covers looping constructs. Selection and looping are the major methods for altering the flow of control in a program. Chapter 6 looks at arrays and Chapter 7 covers procedures and functions. At this point all the fundamental features of Pascal have been discussed. Chapter 8 discusses program development. Chapters 1 through 8 should be covered in a one-semester introductory course in programming. Chapter 9 covers an advanced topic in Pascal, user-defined types. This chapter should be read in detail to truly appreciate the flavor of Pascal's differences from other languages. Chapter 10 covers file-oriented I/O. Chapter 11 discusses the generation of sound and graphics. Both of these chapters introduce a variety of nonstandard features that will be of more use as your programming ability grows.

Trademarks of material mentioned in this text

IBM is a trademark of International Business Machines Corporation. UCSD p-System PASCAL is a trademark of the regents of the University of California. CP/M is a registered trademark of Digital Research, Inc.

Limits of Liability and Disclaimer of Warranty

The author and publisher of this book have used their best efforts in preparing this book and the programs contained in it. 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.

CONTENTS

1

About Your IBM PC

Operating Systems and Pascal Dialects

As this book is being written, IBM already markets three different operating systems for the PC: DOS, the UCSD p-System, and CP/M-86. Also, other vendors are marketing competing versions of the UCSD p-System and CPM, as well as UNIX operating system look-alikes. The number of operating systems available for the PC is destined to grow, and most of them will likely have their own individual dialect of Pascal. Of course there is a "standard" Pascal, but it may be impossible to find a compiler that faithfully implements it and only it.

In this book we are concerned only with the IBM DOS Pascal compiler and the Pascal compiler that comes with the UCSD p-System. Sections of the book that describe differences in the two Pascals are clearly marked in the Table of Contents.

One fundamental difference between DOS and p-System Pascal is that p-System Pascal is not fully compiled. A compiler translates programs in a high level language such as Pascal into the actual language used by the processor inside the computer. The IBM PC is based on the INTEL 8Ø88 microprocessor, and the DOS Pascal compiler translates Pascal into machine code that can be directly executed by the 8Ø88. The p-System compiler translates Pascal into something called "P-code." P-code is a language closer to the machine level than Pascal but more general than any particular processor. The P-code version of a Pascal program is translated into the exact language of the INTEL 8Ø88 at the time the program is executed. (This process is referred to as "interpretive" execution.) As a result, programs written under DOS Pascal

will generally execute faster than those written under the p-System. However, using the "Native Code Generator" of the p-System will recover some of the speed. This option to the p-System compiler is explained in the User's Guide for the p-System.

File formats on floppy disk are not compatible between DOS and the p-System. That is, a file created under one of the operating systems is not directly accessible by a program running under the other operating system.

Display Devices

The choice of a display device for your PC actually begins with choosing between the monochrome monitor adapter and the color/graphics adapter. Choosing the monochrome monitor adapter automatically limits you to the IBM monochrome monitor for a display device. (It is called "monochrome" rather than "black and white" because the phosphor in the display is green.) The primary advantage of this option is the especially high quality text display of the IBM monochrome monitor. Its primary disadvantage is that it can only do text, no graphics.

With a color/graphics adapter, your choices for a display device are more numerous. You could use

1. a black-and-white home television
2. a color home television
3. a standard black-and-white monitor, or
4. a standard color monitor, either composite or RGB variety

Both choices 1 and 2 require that you purchase an RF modulator to go between the color/graphics adapter and the TV set. Choice 4 may require a cable to go with the monitor (at extra cost). You can do graphics with any of choices 1 through 4. The ability to do graphics lies in the adapter, not the monitor, but the display does dictate the visual quality of the graphics. Color graphics can only be done with options 2 and 4, as the ability to generate color is a property of the display device.

Sound and Graphics

If you are using p-System Pascal, then commands to generate sound and graphics with the PC are available in the IBMSPECIAL and TURTLEGRAPHICS libraries. If you are using DOS Pascal, there is nothing standard with the system to make it easy to use sound and graphics.

In Chapter 11, we describe a set of sound and graphics commands that are detailed in a companion book, *Games, Graphics, and Sound for the IBM PC*. Most of these commands are already available to p-System Pascal. The full set of commands described in that book are detailed in its appendices, and information is given on how to implement them for DOS Pascal. As an

alternative to implementing the commands yourself from that book, you can order a diskette of the prepared routines to link to your Pascal programs.

Text Editing, Compiling, and Executing

A text editor is a program used to enter text from the keyboard and save it in a file on disk. The file of Pascal text entered through the text editor is translated by the compiler into a form the processor of the system can actually use.

If you are using the UCSD p-System a fairly nice screen-oriented editor is a standard part of the system. A "screen-oriented" editor is distinct from a "line-oriented" editor. Screen editors allow you to make changes in the text anywhere on the screen, whereas line editors require that you locate a particular line and give editing commands specific to that line. The trouble it would take to replace the standard UCSD p-System screen editor with a nicer one is probably not worth the effort.

If you are using DOS version 1.Ø or 1.1, then the standard editor that comes with the system is a relatively simple line editor (EDLIN). You may want to seriously consider purchasing a screen editor, which would give you much more power and flexibility in creating program text. It should show a noticeable increase in your productivity. Literally dozens of screen editors are available. One entirely reasonable screen editor for DOS is the MINCE editor, marketed by Mark of the Unicorn. At least make a trip to your local computer store and see what is available in this area.

Appendices A and B detail the processes necessary to edit, compile, and execute a program. The editors described there are the standard UCSD p-System screen editor and the DOS EDLIN editor. If you have not prepared and executed programs on the PC before, you should skip to the appropriate appendix now; read it to get an idea of what is involved. You will also need to refer back to the appendix as you work the first few example programs. You should eventually refer to the documentation that came with your system to learn the advanced features of the editor and compiler.

Using This Book

This book is built around several concepts:

- Start slowly.
- Learn by doing.
- The "natural" sequence of topics is not the same to experts as it is to novices.

The book develops topics relatively slowly. People who already consider themselves experts in Pascal may not like the sequence of topics in the book.

Some of Pascal's most important contributions, such as user-defined types, are put off until late in the book because we believe novices are not ready to appreciate things in the same way as experts.

There are lots of examples that are complete working programs. RUN THEM!! We believe in learning by running examples rather than by reading abstract descriptions of the language. (If you don't want to type in the programs yourself, order them on diskette. The listings of all the example programs are available on floppy disk.) Each example program illustrates a point (or points), and you really should work through them all. Reading over program text and understanding it intellectually is not at all the same level of knowledge as running the program and "feeling" its operation.

Even though it is oriented to beginners in Pascal, this book contains material that will be of use to seasoned Pascal veterans. Experts should be able to find their way easily around the structure of the book and skip the parts they don't need. Novices will need all the parts of the book, and its structure will be vital to them.

2

Getting Started in Pascal

Program Form and Simple Output

The Concept of a Program

A program is a group of commands and associated information that can be executed by the computer. A program can be thought of as being similar to a recipe. You start off with some basic information—the ingredients and the equipment needed; spoons, mixing bowls, measuring cups, etc.—and then you follow the steps as they are listed to get the result. Note, however, that there can be more than one way to achieve the final product; there are lots of recipes to make chocolate chip cookies. Conversely, you need to have the right recipe to do the job. If you want to make an angel food cake, you wouldn't use a recipe for cookies.

What is a Computer Language?

A computer language—in our case, Pascal—must be used to communicate with the computer. To continue the cooking analogy, a recipe uses a specialized language—a combination of standard English language and terms and abbreviations that are associated with cooking. Pascal works along the same principles. It uses English-like phrases where practical and has additional vocabulary that is specific to the computer environment.

A word of warning: Learning a language involves building a vocabulary and set of rules and structures before it can be used in any real sense. In

learning a foreign language, you start out with very simple phrases—like "Good morning, sir, how are you?," and "I am fine, thank you." You don't start out reading classic literature. The same is true in learning a computer language. You have to start with the basics and build. Because of this, many of the exercises and examples presented here may seem artificial. In fact, most only serve to illustrate the use of a particular feature. Be patient. Once the building blocks have been established you will be able to charge ahead to solve important and exciting problems that interest you.

The Structure of a Pascal Program

The structure of a program consists of a heading, which states the name of the program, the word BEGIN, which shows that the program is starting, the commands that describe the actions to be taken, and a final END statement. The heading consists of the word PROGRAM, the name of the program, and a semicolon. The BEGIN and END mark the start and finish of the commands. They are necessary because without them the computer would not know where one program started and another began. Returning to the idea of a recipe, without BEGIN/END it would be possible to combine a cake with a souffle but you would end up with a terrible mess. For example:

```
PROGRAM name_you_give_it;
BEGIN
   { statements }
END.
```

Note the use of a semicolon after the program heading. Semicolons tell the computer that a statement is finished. In the technical jargon of computer people, semicolons are said to be "delimiters" between statements. Delimiters are just a way of separating things. Delimiters are not always semicolons, but we'll get to that later.

What is Syntax?

Syntax is a term that refers to the rules that define the structure of a language. For instance, part of the syntax of an English sentence is that the first letter of the first word is capitalized and that the sentence ends in a period. Pascal syntax requires that a program be defined in the format described above.

So far, we only know the outline of a Pascal program. In order to write a program we need some statements that actually do something. The WRITELN statement will write messages on the display and then skip to the next line. The syntax of a WRITELN statement is the word WRITELN followed by an open parenthesis, a quote followed by as many characters as desired (as long as it fits on one line), and a closing quote followed by a closing parenthesis. For example:

```
WRITELN(' any text you like ');
```

WRITELN statements are placed between the BEGIN and END of a program. Type the following program (our first) into the editor. Refer to Appendix A for DOS Pascal or Appendix B for p-System Pascal, if you are uncertain how to proceed at this point.

```
program sample2_1;

BEGIN
   WRITELN('This is an example.');
   WRITELN('I am going to be a great programmer.');
END.
```

Program 2.1.

Once the program has been entered, you'll want to execute it, but this can't be done yet. First it is necessary to put the program into a different form, through the use of a compiler. The compiler translates Pascal into a simpler language that the machine can actually execute. (Use of the compiler is also described in Appendices A and B.) Pascal is meant to be understood by people, not by machines. Your Pascal program actually has to be translated into a form the computer can use.

The compiler will go through each line of the program and try to translate it. Unfortunately, the compiler is "stupid." It can only do exactly what you tell it to, nothing more and nothing less. Because of this, it is common to get syntax errors while compiling. A syntax error means that the compiler doesn't understand something in the program. For instance, if you omit the period after END you will get a syntax error. When such an error occurs, you must go back into the editor, fix the error, and try to compile again.

Two words of warning. One is that when you start out you are going to get a lot of syntax errors. Everybody does. It'll be a bit frustrating, but things will improve as you gain more experience. Second, the message you get concerning the type of error and the line that the compiler says the error is on may not in fact directly describe the error you made. For example, if you leave a semicolon off a statement, it won't be discovered until the statement afterward. When an error is found, you know only that something is wrong in the program *somewhere up to* the point where the compiler indicates an error.

Once the program has been compiled it can be executed. The part of the computer that executes the program is called the processor. The processor takes each line of the program, in order, and carries out the specified task. When Program 2.1 is run, the following lines will be displayed on the screen:

```
This is an example.
I am going to be a great programmer.
```

Comment Statements

As mentioned, the compiler carefully checks each line to be sure that it conforms with Pascal syntax, but there is an exception to this. Anything that

you place in curly brackets { } or in parentheses-asterisks (* *) will be ignored by the compiler. This is a way of inserting "comments" into the text of the program. You can use comments to make notes to yourself or to provide information to anyone who may read the program later. The following example program produces the exact same result as Program 2.1.

```
program sample2_2;
{  this is a program that demonstrates the use of comments }
(* author: jane programmer          written: feb 2, 1982 *)

BEGIN

   WRITELN('This is an example.');  { comment after statement }
   WRITELN('I am going to be a great programmer.');

END.
```

Program 2.2.

We use curly brackets because they involve fewer keystrokes than the parentheses-asterisks. Either is legal in Pascal, but the parentheses/asterisk version is primarily for computers that do not have curly brackets on the keyboard. You should develop the habit of commenting all your programs in the same general manner. Use comments to make the purpose and operation of the program as obvious as possible to a reader. We have already introduced the convention of documenting who wrote the program and when it was written. Later we will introduce more standards for using comments to document your program.

You can also put blank lines and blank spaces in your program to make it easier to read. Notice that we indent all the statements grouped together in the BEGIN/END block.

Overview

1. The syntax of a program is:
```
PROGRAM name_you_make_up;
BEGIN
   { statements }
END.
```
2. WRITELN is used to output messages to the display. Whatever is within the quotes gets printed out, and further output statements will begin on the next line. The syntax is:

```
WRITELN('Anything you like as long as it fits on one line.');
```

3. To run a program you need to compile it. A compiler translates the Pascal statements into a form the computer can use. You can execute the translated (compiled) version of the program.
4. A syntax error means a statement is not grammatically correct in the Pascal language. Syntax errors are detected by the compiler. The most

common errors you will encounter at this point are omitting semicolons at the end of the line, leaving out the BEGIN or END, leaving out the period after END, or leaving off one or both quotes in a WRITELN.

Exercises

1. Modify the example program of this section so that the first line of the output is

   ```
   "This is another example."
   ```

2. Modify the example to write out a third line:

   ```
   "This is fun. RAH! RAH! RAH!"
   ```

3. Create a program that writes out the first line of the Pledge of Allegiance. Put a comment in the program that describes what it does.

Solutions

1. ```
   program solution;
   BEGIN
      WRITELN('This is another example.');
      WRITELN('I am going to be a great programmer.');
   END.
   ```
2. ```
   program solution;
   BEGIN
      WRITELN('This is an example.');
      WRITELN('I am going to be a great programmer.');
      WRITELN('This is fun. RAH! RAH! RAH!');
   END.
   ```
3. ```
   program solution;

   { This program prints the first line of the pledge of
     allegiance. }

   BEGIN
      WRITELN('I pledge allegiance to the flag');
      WRITELN('of the United States of America.');
   END.
   ```

More About Output

There are two very similar output statements in Pascal: the WRITELN and the WRITE statements. We have already used the WRITELN statement to print out messages from our programs. Our messages have all been strings of text enclosed by single quotes. There has only been one string in each statement, and the message from each statement has appeared on its own line. In this
   ```