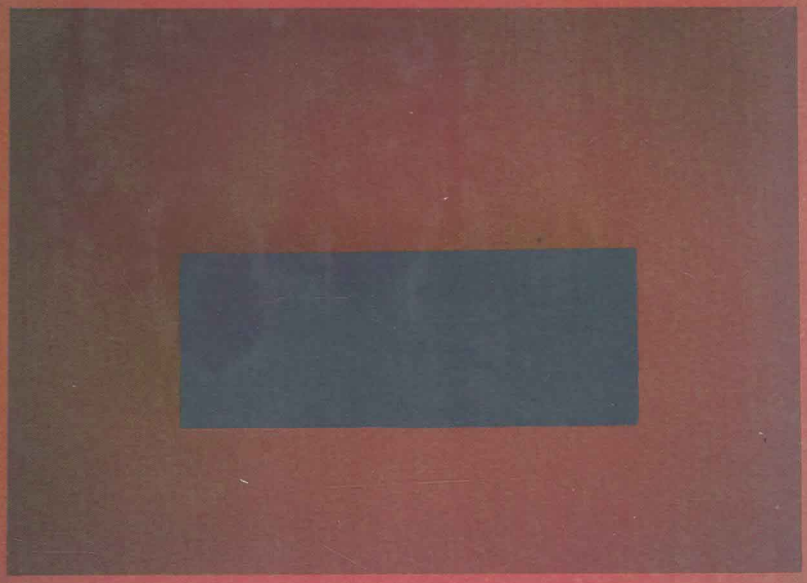
micro COMPUTER books

PASCAL

for the IBM Personal Computer

Ted G. Lewis



With IBM™ and UCSD™ Pascal

PASCAL FOR THE IBM PERSONAL COMPUTER

TED G. LEWIS

Addison-Wesley Publishing Company
Reading, Massachusetts • Menlo Park, California
London • Amsterdam • Don Mills, Ontario • Sydney

This book is in the

Addison-Wesley Microcomputer Books

Popular Series

Marshall Henrichs, *Cover Design*

Library of Congress Cataloging in Publication Data

Lewis, T. G. (Theodore Gyle), 1941–
Pascal for the IBM Personal Computer.

(Addison-Wesley microcomputer books popular series)
Includes index.
1. IBM Personal Computer—Programming. 2. PASCAL (Computer program language) I. Title. II. Title: Pascal for the I.B.M. personal computer. III. Series.
QA76.8.I2594L46 1983 001.64'2 82-22750
ISBN 0-201-05464-7

ISBN 0-201-05464-7
ABCDEFGHIJ-HA-89876543

PREFACE

Why would anyone write another book on Pascal? There are several compelling reasons why I wrote this book. First, the IBM Personal Computer provides a new vehicle for people to learn about computer operation. This means learning the extended character set of the machine, its operating system, and, in the context of this book, a new programming language such as IBM Pascal. Specifically, IBM Pascal incorporates many nonstandard features not found in other versions of Pascal. IBM Pascal, an *extended* dialect of ISO Standard Pascal, includes the new control structures otherwise, break, cycle, and then, or else, and return, and the new data types for systems programming word, byte, adr, and ads. The notion of data abstraction is pushed a small step forward by the implementation of super arrays, which alleviate some of the restrictions on passing arrays to subprograms.

Second, there is a generation of Pascal programmers who have been using UCSD Pascal on the Apple II, North Star, and other microcomputers. These programmers are now beginning to shift to the IBM Personal Computer because of its larger memory and other features and need to know the similarities and differences between UCSD Pascal and IBM Pascal. For this group, I have included many sections on UCSD Pascal. If you are one of these people, you will also be especially interested in the way IBM Pascal handles separately compiled units (Chapter 18).

Third, there are many new people entering the computer field each year who are in need of a definitive treatment of programming languages such as Pascal. Pascal is a relatively sophisticated family of languages and is not easy to learn. I have tried to break the language down into easily understood parts; each chapter describes an important part of both IBM and UCSD Pascal. Both versions of Pascal run on the IBM Personal Computer. Therefore, if you are undecided about which translator to buy, then read Chapter 1 carefully. Your decision is an important one because of the differences between the two languages.

My approach in this text is to start out with many examples that will coax you into using the computer as soon as possible. The chapters become increasingly advanced as you move from beginning to end, the last chapter showing the full power of UCSD Pascal in implementing real-life programs. These programs, which are ready to be compiled and run, ordinarily would cost you several times the price of this book if you were to buy them separately—they are listed in the chapters for you to copy, modify, read, etc. These programs may be purchased on diskette, but you can easily enter them manually into your IBM Personal Computer.

I am indebted to many people for their help and guidance in this project. Tom Bell and Tom Dwyer at Addison-Wesley were great to work with. Kent Byerley at Computerland gave free advice on hardware and software problems. Abbas Birjandi, Edmund Wu, and Larry England contributed many programs that appear in the book. Ann Puig and Donna Lee Norvell-Race gave me the benefit of their expertise in manuscript preparation. Thanks to these people, the book was completed in a timely manner, and its effectiveness was enhanced.

Corvallis, Oregon T.G.L.
August 1983

Other books in the Microcomputer Books Series are available from your local computer store or bookstore. For more information write:

General Books Division
Addison-Wesley Publishing Company, Inc.
Reading, Massachusetts 01867
(617) 944-3700

(10341) **Pascal: A Problem Solving Approach**
Elliot B. Koffman

(03115) **A Bit of BASIC**
Thomas A. Dwyer and Margot Critchfield

(01589) **BASIC and the Personal Computer**
Thomas A. Dwyer and Margot Critchfield

(05247) **The Little Book of BASIC Style**
John M. Nevison

(05248) **Executive Computing**
John M. Nevison

(06192) **The Computer Image**
Donald Greenburg, Aaron Marcus, Allan Schmidt, and Vernon Gorter

(10158) **Pascal from BASIC**
Peter Brown

(10243) **Executive VisiCalc for the IBM Personal Computer**
Roger E. Clark

(06577) **Pascal for BASIC Programmers**
Charles Seiter and Robert Weiss

(16455) **The Addison-Wesley Book of IBM Software 1984**
Robert Wells, Sandra Rochowansky, and Michael Mellin

(13047) **1-2-3 Go!**
Julie Bingham

(10256) **A Guide to the Best Business Software for the IBM PC**
Richard C. Dorf

(08848) **The Business Guide to the UNIX System**
Jean Yates, Sandra Emerson, James Talty, and Rebecca Thomas

(08847) **The Business Guide to the XENIX System**
Rebecca Thomas, Jean Yates, Sandra Emerson, Joe Campbell, and James Talty

(06896) **The IBM PC from the Inside Out**
Murray Sargent III and Richard L. Shoemaker

CONTENTS

CHAPTER 1

INTRODUCTION TO PASCAL ON THE IBM PERSONAL COMPUTER

This book is about Pascal programming on the IBM Personal Computer. In the chapters that follow, you will be gradually introduced to the power and flexibility of two Pascal dialects. The IBM Pascal dialect is an extension to the ISO (International Standards Organization) standard Pascal that emphasizes the "systems" features needed for machine-level programming. You can think of these extensions as a type of high-level assembly language. The UCSD Pascal dialect is an extension to the ISO standard Pascal that emphasizes the needs of application programming. You can do extended arithmetic, graphics, and screen editing programs with the UCSD Pascal dialect.

Unfortunately, this simple division between these two dialects is a gross oversimplification. Both languages can be applied to systems programming, and with the addition of some assembly language routines, both can be used for application programming. To really understand

the differences, you will have to read the remaining chapters. But for a quick overview, we can examine these two languages in the large.

IBM PASCAL

IBM Pascal was developed by Microsoft, Inc., to be a highly efficient, optimized code translator. Considerable effort went into making programs run fast after being compiled into machine language. One consequence of this improved efficiency is that three passes are made over your original source program in order to translate it into machine language. An IBM Pascal program is translated into machine instructions that execute under the IBM DOS operating system. The result is a machine-language program that can be combined with other machine-language programs. Once translated, an IBM Pascal program cannot be distinguished from any other machine-language program.

Not only does the IBM Pascal compiler make three time-consuming passes over your source program, but it also requires 128K of main memory. You should keep this memory requirement in mind if you have not yet purchased a machine with that much capacity.

An IBM Pascal object program (machine-language version of your program) usually requires approximately 25K of run-time support routines. A run-time support routine is a subprogram that resides in memory while your program is running. It helps your program do input and output (I/O), arithmetic, and so on.

UCSD PASCAL

UCSD Pascal was developed at the University of California at San Diego under the guidance of Dr. Kenneth Bowles. Later it was licensed to SofTech Microsystems for commercial distribution. The original version has evolved to version IV, which runs on the IBM Personal Computer and on many other microcomputers.

One of the strong features of the UCSD Pascal language is that it is part of a portable operating system called the p-system. The p-system includes a filer for managing disk files, an editor for composing programs, and translators for converting assembly-language programs, FORTRAN programs, and UCSD Pascal programs into executable instructions.

Portability is obtained at a price, however, because programs in the p-system are translated into p-code, not machine language. A p-code program is a program containing hypothetical machine-language instructions called p-code instructions. These p-code instructions are more compact than native machine-language instructions, but they execute more slowly. The entire p-system will execute on a 64K machine; thus only 64K of memory is required. The p-system interprets p-code programs in much the same way as a BASIC interpreter directly executes BASIC instructions. The p-code interpreter, sometimes called a p-code simulator, simulates a p-code machine. The p-code simulator is a machine-language program. The p-system runs UCSD Pascal programs much more slowly than they would run if they were translated into machine instructions. Thus the price of portability is slower execution of your program.

Version IV of the UCSD p-system includes a native-code generator, which can translate most p-code instructions into equivalent machine-language instructions. Since the translation is not 100% complete, you must continue to run your programs under the p-system. The advantage of native-code translation, however, is increased speed. Much (not all) of the lost performance speed is regained, but the resultant program is considerably larger. So, you must decide which is more important to you: size or speed. (You can selectively translate subprograms into native code.)

The UCSD Pascal compiler does its translation in one pass. This means you can expect fairly quick translations. Very little in the way of run-time support is needed by your p-code program because the run-time support routines are built into the p-code simulator.

Portable p-code programs can be run on any microcomputer that has the p-system software. For the software developer, this means that once a program is written for the IBM Personal Computer, it will also run on a number of other microcomputers.

A COMPARISON OF SYSTEM REQUIREMENTS

Table 1.1 summarizes the differences between UCSD and IBM Pascal system requirements. This is a very broad comparison, but it does tell you what kind of hardware you will need in either case.

TABLE 1.1 System Requirements of the Two Pascal Dialects

	IBM	*UCSD*
Diskettes to compile	3	1
Manuals	1	2 (language only)
Memory required	128K	64K
Disk drives recommended	2	2
Number of printers supported	up to 2	up to 1
Display	black-and-white or color	black-and-white or color
Operating system	DOS	p-system

The IBM dialect requires three diskettes, one for each of its three passes. Since the UCSD version is a single-pass compiler, it requires only one diskette. Both versions can run on a single diskette system (especially on double-sided drives), but this is not recommended.

Notice the difference in main memory requirements. The UCSD p-system is very compact, but your Pascal programs may be very large. Therefore, you may need 128K of main memory in either case. This memory difference should not be a consideration in choosing one version of Pascal over the other. You will soon find that 128K is desirable for all but the most trivial programming.

Both dialects support the monochrome display and the color adapter. However, UCSD Pascal includes turtlegraphics, a library of programs that allows you to do color and black-and-white graphics. If

you plan to do graphics at all, the UCSD Pascal language is the best choice.

Programs written for DOS can be easily transferred to other IBM Personal Computers running under control of DOS. However, programs written for the p-system can run on any other computer under control of the p-system. (A licensing fee allows you to distribute the p-system with your programs.)

If you plan to do system-level programming and you want machine-language object programs as your final result, then the IBM Pascal system is probably your best choice. What you must remember is that you are trading portability for performance.

A COMPARISON OF LANGUAGE FEATURES

Standard (ISO) Pascal is very similar to the original language invented by Niklaus Wirth. The original language was designed to help students learn proper programming skills and to be easily implemented on a variety of computers. These priorities have long since been forgotten in the rush toward modern structured programming languages. Most "real-world" versions of Pascal go beyond the pedagogical standard. They attempt to do everything that a professional programmer wants of them. IBM Pascal and UCSD Pascal are no exceptions. Table 1.2 summarizes the overall features of these two dialects of Pascal.

Both languages support character strings as a built-in data type. IBM Pascal actually supports two kinds of strings: an lstring (length string) and a simple string. An lstring can vary in length, whereas a simple string cannot. The UCSD string and the IBM lstring are similar.

Both languages support string-processing intrinsic functions. An *intrinsic function* is a built-in function. IBM Pascal is a little confusing, however, because of its lstring and simple-string data types. These are covered in great detail in later chapters.

Graphics (turtlegraphics) and sound reproduction are supported by a library of pretranslated routines in UCSD Pascal. You should use UCSD Pascal if you plan to program applications in music, games, or graphic design.

TABLE 1.2 Features of the Two Pascal Dialects

	IBM	*UCSD*
Strings	yes	yes
Graphics	no	yes
Sound	no	yes
Units (separate compilation)	yes	yes
Modules (separate compilation)	yes	no
Concurrent processes	no	yes
Program chaining	no	yes
Direct files	yes	yes
Systems programming	yes	no
Packed data	no	yes
Initial values	yes	no
Structured constants	yes	no
Conrol break	yes	yes
Control cycle	yes	no
Case-otherwise	yes	no
Procedures as actual parameters	yes	no

IBM Pascal does not directly support screen cursor control, but a machine-language routine for this operation is included in this book. The UCSD language, on the other hand, includes the intrinsic function GOTOXY for cursor control.

Both languages support separately compiled subprograms. A *unit* is a cluster of procedures, functions, and data that can be separately compiled, stored on disk, and then linked into any other program. Units can be useful when one is building large systems. IBM Pascal modules are also separately compiled programs. They are similar to units.

UCSD Pascal provides a way for you to dynamically overlay pieces of a large program into memory. This means you can run programs that are too large to fit into memory in one piece. This feature may be an important one to consider if you plan to build very large programs. If so, then the UCSD Pascal language may be your only choice.

UCSD Pascal also provides concurrent processes. A *concurrent process* is a program that executes side by side with another program. Both programs execute in short bursts—first one and then the other. For example, you might want to write a program that will print the contents of a file while another program is doing word processing (on another file). The print program is called a *spooler*, and it executes in short bursts in between the times the word-processor program is executing. The spooler and the word-processor program are part of a system of (two) concurrent processes. Since it provides a way for you to write concurrent programs, UCSD Pascal may be a better language for you to use for systems programming than IBM Pascal. In fact, if you want to learn more about concurrency in computer systems, use UCSD Pascal.

Both languages exceed the ISO standard for file processing. A *direct-access file* is a file containing data that can be directly accessed in one "seek" to the diskette. This feature is essential in most data-processing applications. Because of its importance, a full chapter of this book is devoted to this topic.

IBM Pascal includes extensions to ISO Pascal for doing systems programming. In particular, IBM Pascal includes bit-level data types, **byte** and **word,** that let you access binary-encoded cells in memory. Other data types allow you to control the memory segmentation register of the 8088 processor, and so forth. These features break down the strong typing of Pascal so that you can do mixed-type operations on memory.

UCSD Pascal allows you to compress data into the smallest possible memory space. This *packed attribute* causes data to be squeezed into memory in the most efficient way possible. IBM Pascal, on the other hand, does not pack data. Remember, the goal of IBM Pascal is to be fast and efficient. If you want packed data, then you must pack it yourself using the system-level data types.

UCSD Pascal provides an exit procedure that lets you break out of a procedure or program before reaching the end. This is an "early termination" intrinsic that is useful in structured programming.