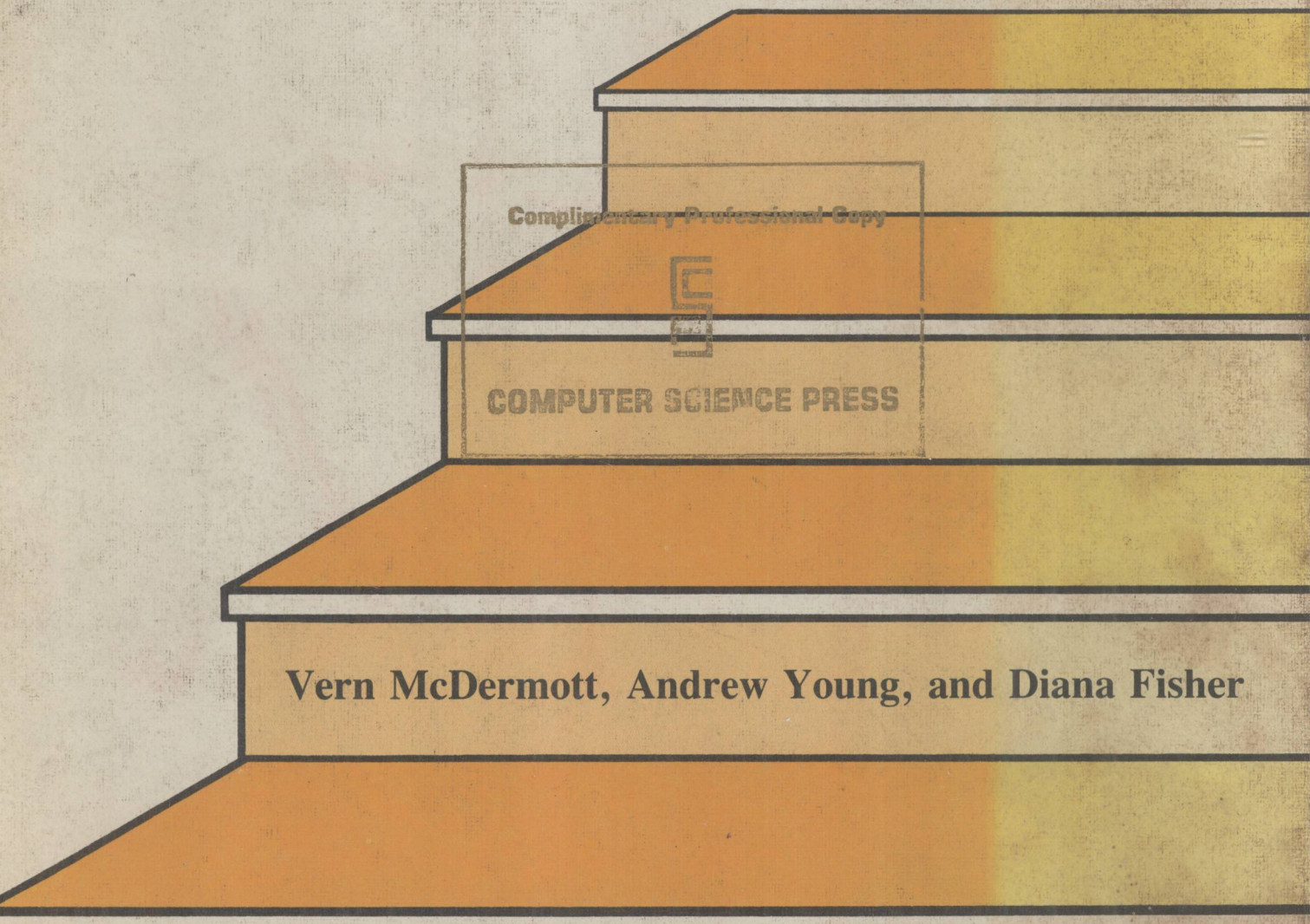
Learning Pascal Step by Step



Vern McDermott, Andrew Young, and Diana Fisher

Machine Specific for APPLE II Plus • APPLE IIe • IBM PC • TRS-80

COMPUTER SCIENCE PRESS

Learning Pascal Step by Step

Vern McDermott
Andrew Young
Diana Fisher

Machine Specific for APPLE II Plus• APPLE IIe•IBM PC•TRS-80

COMPUTER SCIENCE PRESS

Computer Science Press, Inc.
1803 Research Blvd.
Rockville, Maryland 20850

1 2 3 4 5 6 Printing

Year 90 89 88 87 86 85

Library of Congress Cataloging in Publication Data

McDermott, Vern, 1934–
Learning Pascal step by step.

1. PASCAL (Computer program language)
I. Fisher, Diana, 1948- . II. Title.
QA76.73.P2M364 1985 001.64'24 84-19869
ISBN 0-88175-045-X {Student Textbook}
ISBN 0-88175-046-8 {Teacher's Edition}

The cartoon characters were drawn by Laura Bishop.

Cover design by Ruth Ramminger.

Learning Pascal Step by Step

OTHER BOOKS OF INTEREST FROM COMPUTER SCIENCE PRESS

Lowell A. Carmony
Problem Solving in Apple Pascal

Donald J. Geenen
Learning Apple Fortran

Rachelle S. Heller, C. Dianne Martin, and June L. Wright
LOGOWORLDS

Jo Lynn Talbott Jones
Applesoft BASIC Primer for the Apple II Plus, IIe, and IIc

Tom Logsdon
Computers Today and Tomorrow: The Microcomputer Revolution

Jeffrey Marcus and Marvin Marcus
Computing Without Mathematics: BASIC, Pascal and Applications

Marvin Marcus
An Introduction to Pascal and Precalculus

W. Douglas Maurer
Apple Assembly Language

Vern A. McDermott and Diana Fisher
Learning BASIC Step by Step

Vern A. McDermott and Diana Fisher
Advanced BASIC Step by Step

Derick Wood
Paradigms and Programming in Pascal

PREFACE

While developing a course in Pascal programming, the authors recognized a need for an introductory Pascal textbook that satisfied the following criteria:

The text be written with a style and a reading level that is suitable for high school or junior college students.

The text be suitable for use in an independent study program as well as in a lecture-oriented classroom environment.

The text be complete so it can be used for formal programming classes without supplementing or altering; that is, it contains material that is well organized, and new concepts that are supported by demonstration problems and ample skill development exercises.

Therefore, in developing *Learning Pascal Step by Step* the following features were implemented:

A problem-solving approach that is used throughout the text.

A diagram approach to presenting new concepts that lends itself to easy referencing.

Material that is presented at a pace to encourage progress without frustration.

Ample opportunity for students to become proficient in using the new concepts before moving on to new topics.

Careful organization of new concepts so that no program requires a concept that has not been introduced.

An easy-to-understand format that presents concepts which illustrate strengths of Pascal, such as sets, record structures, type declarations, and passing parameters to procedures.

In order to place maximum emphasis on new concepts being introduced, less emphasis was placed on comments both in the demonstration and skill development programs; however, the use of comments is strongly encouraged in places where they will enhance the clarity and readability of the programs.

ACKNOWLEDGMENTS

Special thanks go to Mike LaChapelle, Mike Lanzarotta, Tim Carlson, and Gary Schroeder, who, while using the preliminary manuscripts, pointed out many errors and organizational problems.

Andy Young would like to personally thank his grandparents, parents, and friends, Dave and Brian, whose support and encouragement were so greatly appreciated that they could never be repaid.

INTRODUCTION

Niklaus Wirth designed the language Pascal, which was named after the 17th century mathematician Blaise Pascal. One of the principal reasons for designing the language Pascal was to create a programming language that could be used to teach a well-organized and disciplined approach to programming and problem solving. As a result Pascal has become the major instructional language used in colleges and universities.

Reasons why Pascal is becoming the most popular language in use are:

1. Pascal is widely available on many different types of computers, ranging from large mainframe computers to microcomputers. That is, the different computers have Pascal compilers available that will translate programs written in Pascal into the machine language of the computer being used.
2. Pascal lends itself to good programming techniques. Thus, students of Pascal learn to program effectively.
3. Pascal is a structured programming language. Programs written in Pascal are well-organized and easy to follow.
4. Pascal is an excellent teaching language. Therefore, learning Pascal provides a good background for future studies in computer science.

This book was written for the Apple UCSD Pascal, Radio Shack TRS–80 version of New Classics Pascal–80, and the IBM P–SYSTEM VERSION IV.0.

GETTING STARTED

In order to complete the problems in this text, the user must become familiar with the computer's Pascal operating system. Abbreviated instructions for using the operating system are located in the appropriate appendix as indicated below.

APPENDIX I

APPLE Operating System

APPENDIX II

TRS–80 Operating System

APPENDIX III

IBM PC Operating System

Lessons

CONTENTS

Lesson 1
PASCAL EXPRESSIONS

1.1 OBJECTIVES

Objectives:
1. To understand the types of constants, variables, and operations that can be used.
2. To be able to translate expressions from algebra to Pascal.
3. To understand the correct sequence for the order of operations.

1.2 CONSTANTS, VARIABLES, AND OPERATIONS

DATA TYPES[1]

Type	*Examples*	*Description*
INTEGER	5 86 0	Whole numbers (i.e., numbers that do not contain a decimal point). The largest integer allowed is +32767 and the smallest is −32768.
REAL	5.3 18.0 0.4 6.5E3 4.32E−2	Numbers that involve a decimal. (The decimal must have at least one digit in front of the decimal point and one digit behind the decimal point.) 6.5E3 = 6500 (6.5×10^3) 4.32E−2 = 0.0432 (4.32×10^{-2})

VARIABLES (IDENTIFIERS)

Examples:

```
N
Sum
Average1
Average2
AppliedToPrin
```

Rules for Identifiers

1. They contain a sequence of the letters (A–Z) or a combination of the letters (A–Z) and the digits (0–9).
2. They must begin with one of the letters.
3. They cannot contain symbols such as /,*,−,&,$,%, or blanks.

PROBLEM IDENTIFIERS

```
Summation1
Summation2
```

Problem

Even if there is no specified limit to the number of digits that a variable can contain, it is common that only the first eight are actually used. Therefore, the two identifiers to the left will be treated as the same variable by some compilers.

[1] Other types such as `STRING`, `BOOLEAN`, and `CHAR` will be presented in later lessons.

ILLEGAL IDENTIFIERS	*Reason*
`Miles/Gal`	Only the characters (A–Z) are allowed; symbols like "/" are illegal.
`3rdGroup`	A variable must start with one of the letters (A–Z).
`Net Salary`	Blanks are not allowed.
`Real`	Reserved words cannot be used as variables. (`REAL`, `END`, `PROGRAM`, `INTEGER`, `DIV` are examples.)

OPERATIONS	*Meaning*
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division (Real type)
`DIV`	Division (Integer type)
`MOD`	Remainder Division (See explanation below.)

Note: Problems involving exponents must be treated like multiplication problems until the proper functions are presented in a later lesson. Example: X^2 must be written as (X*X).

ORDER OF EXECUTION

Rules for the order of execution of operations are the same as the rules in algebra.

First	Exponents and parentheses
Second	`*`, `/`, `DIV`, and `MOD` from left to right
Third	`+` and `-` from left to right

ASSIGNMENT STATEMENT EXAMPLES	EXPLANATION
`Y := A + B`	The value of `A + B` is assigned to `Y`.
`Ave := (Item1 + Item2) / 2`	The numbers stored in `Item1` and `Item2` are added together, the sum is divided by two, and the result is assigned to `Ave`. `Item1` and `Item2` may be real or integer type numbers.
`IntQuot := Num DIV Den`	The number stored in `Num` is divided by the number stored in `Den` and the integer part of the result is assigned to `IntQuot`. For example: `25 DIV 4 = 6` `-14 DIV 3 = -4`
`RealQuot := Num / Den`	Same as above only the real result is assigned to `RealQuot`. For example: `25 / 4 = 6.25` `-14 / 5 = -2.8`

`Rem := Num1 MOD Num2`	The value stored in `Num1` will be divided by the value stored in `Num2` and the remainder will be assigned to `Rem`. For example: `6 MOD 2 = 0`, `10 MOD 3 = 1`, `-15 MOD 4 = 3`, `4 MOD (-15) = 4`[1]

1.3 DEMONSTRATION PROBLEMS

Changing algebraic expressions to Pascal expressions:

Algebraic Expression	*Equivalent Pascal Expression*
1. $8X+3$	`8*X + 3`
2. $5(X+6)$	`5*(X + 6)`
3. $\frac{A+B}{2C}$	`(A + B) / (2*C)`
4. $A^2 + B^2$	`A*A + B*B` (temporary solution[2])
5. $\frac{XY}{Z+10}$	`X*Y / (Z + 10)` or `(X*Y) / (Z + 10)`

Note: In the above examples the identifiers were chosen as they would be in algebra. When used in programs, the identifiers should be chosen so that they are as descriptive as possible. For example, averaging three numbers in Pascal may look like this: `(Item1 + Item2 + Item3) / HowMany`.

Evaluating Pascal expressions:

Pascal Expression	*Value*
6. `8*5 + 3`	`43`
7. `6 + 4*2 - 3`	`11`
8. `15 DIV 2`	`7`
9. `0.5 * 8.0 + 1.6 * 1.2`	`5.92`
10. `50.0 / 2.5 * (4.2 + 1.8)`	`120`
11. `25 MOD 8`	`1`

[1] On some compilers the parentheses are not necessary. However, it is recommended they be used in order to make the programs more universally accepted.

[2] This is a temporary solution until the proper functions are introduced in a later lesson.