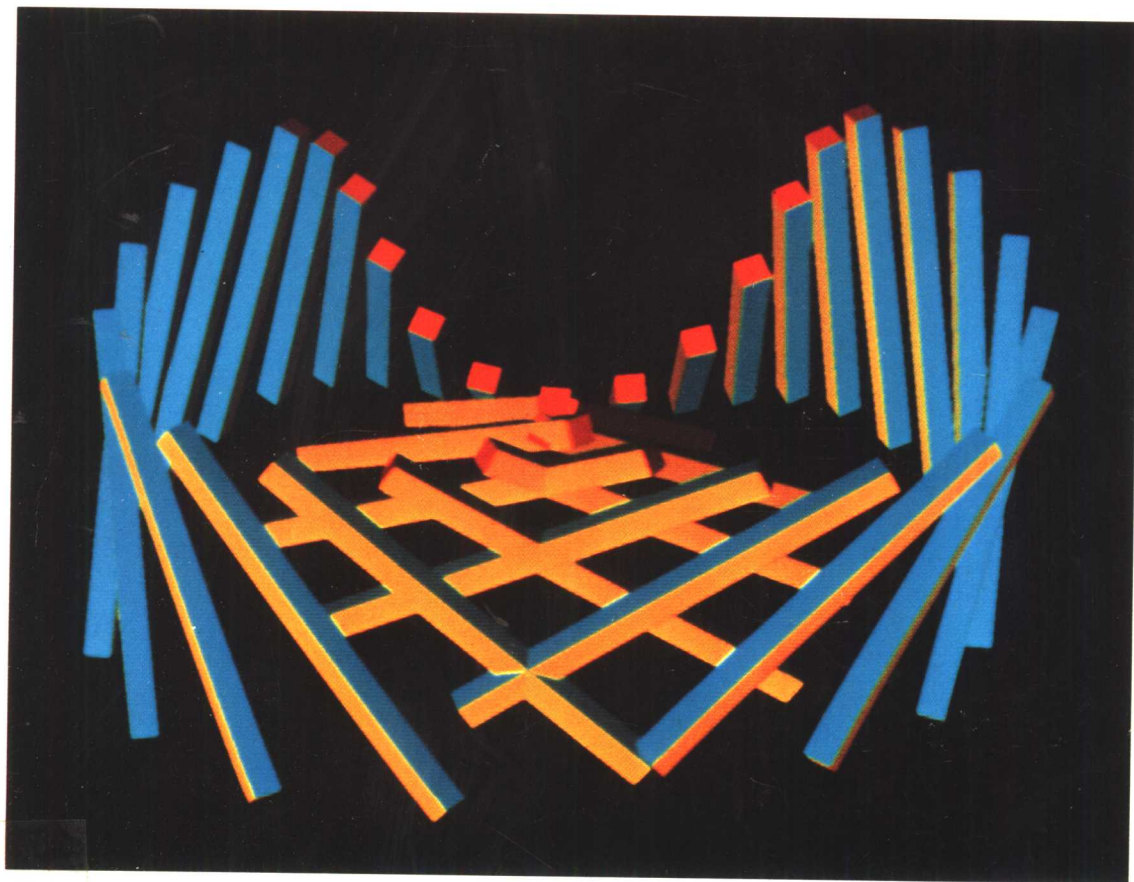


# UCSD Pascal

Featuring the  
Apple® IIe and II Plus



Haigh/Radford

# UCSD Pascal

Featuring the  
Apple® IIe and II Plus

**ROGER W. HAIGH**

*West Virginia Northern Community College*

**LOREN E. RADFORD**

*Baptist College at Charleston*

---

**PWS PUBLISHERS**  
Boston

---

# PWS PUBLISHERS

Prindle, Weber & Schmidt • Willard Grant Press • WGP • Duxbury Press •  
Statler Office Building • 20 Park Plaza • Boston, Massachusetts 02116

"UCSD" is a trademark of the Regents of the University of California. "Apple" is a registered trademark of Apple Computer, Inc.

© Copyright 1984 by PWS Publishers

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission, in writing, from the publisher.

PWS Publishers is a division of Wadsworth, Inc

## Library of Congress Cataloging in Publication Data

Haigh, Roger W.

UCSD Pascal.

Includes index.

1. Apple IIe (Computer)—Programming. 2. Apple II Plus (Computer)—Programming. 3. UCSD Pascal (Computer program language) I. Radford, Loren E. II. Title.  
III. Title: U.C.S.D. Pascal.

QA76.8.A6623H35 1984 001.64'2 83-21974

ISBN 0-87150-457-X

ISBN 0-87150-457-X

Cover photo created by Dave Kamins, Computer Graphics Laboratory, Boston University.

Cover Design by Trisha Hanlon. Text design by Sara Waller. Composition by Science Press, Inc. Artwork by Atlantic Offset Company. Text printed and bound by The Maple-Vail Book Manufacturing Group. Covers printed by New England Book Components.

Printed in the United States of America

84 85 86 87 88 — 10 9 8 7 6 5 4 3 2 1

```

PROGRAM AGEINDAYS;
CONST YEARDAYS = 365; constant declaration (pp. 56-57)
TYPE MONTHS = (JAN,FEB,MAR,APR,MAY,JUN,
               JUL,AUG,SEP,OCT,NOV,DEC); } user-defined ordinal
                                         type declaration (pp. 167-169)
      LONG = INTEGER [6];
VAR BY,CY,BYDAYS,CYDAYS : INTEGER; } global variable declarations (pp. 57, 182)
    TOTDAYS : LONG;

FUNCTION LEAPYR(YR:INTEGER):BOOLEAN;
BEGIN
    LEAPYR := FALSE;
    IF (YR MOD 4 = 0) AND (YR <> 0) THEN
        LEAPYR := TRUE
    END;
} user-defined Boolean
  function (pp. 181-183)

PROCEDURE FINDDAYSINYEAR
(WHICHYEAR:STRING;VAR DAYS,YR:INTEGER); formal parameter list (p. 182)
VAR I1 : MONTHS;
    MO : INTEGER; } local variable declarations (pp. 193-195)

BEGIN
    WRITE('ENTER ',WHICHYEAR,' IN THE FORM 10 14 83 ');
    READLN(MO,DAYS,YR);
    FOR I1 := JAN TO DEC DO BEGIN FOR loop header (pp. 154-156)
        IF (ORD(I1) + 1 = MO) THEN BEGIN
            IF (LEAPYR(YR) AND (MO > 2)) THEN DAYS := DAYS + 1;
            EXIT(FINDDAYSINYEAR)
        END;
        CASE I1 OF
            JAN,MAR,MAY,JUL,AUG,OCT,DEC : DAYS := DAYS + 31;
            APR,JUN,SEP,NOV : DAYS := DAYS + 30;
            FEB : DAYS := DAYS + 28
        } CASE
        END; statement
        (pp. 129-133)
    END;
    IF (LEAPYR(YR) AND (MO > 2)) THEN compound Boolean expression (pp. 127-129)
        DAYS := DAYS + 1
    END;

PROCEDURE SUMDAYS(BY,CY:INTEGER;VAR T:LONG);
VAR I2 : INTEGER;
BEGIN
    FOR I2 := BY TO CY - 1 DO BEGIN
        T := T + YEARDAYS; use of accumulator (pp. 148-152)
        IF LEAPYR(I2) THEN T := T + 1
    END
END;
} user-defined
  procedure
  (pp. 181-190)

```

**BEGIN**

```
  FINDDAYSINYEAR('BIRTH DATE',BYDAYS,BY);  
  FINDDAYSINYEAR('CURRENT DATE',CYDAYS,CY);  
  TOTDAYS := CYDAYS - BYDAYS;  
  SUMDAYS(BY,CY,TOTDAYS);  
  IF ((BY > CY) OR (TOTDAYS < 0)) THEN  
    WRITELN('YOU ARE NOT YET BORN.')  ELSE BEGIN  
    WRITE('CONGRATULATIONS, YOU ARE ');  
    WRITELN(TOTDAYS,' DAYS OLD TODAY.')  END
```

} procedure calls with  
actual parameters (pp. 183–190)

} assignment statement (p. 59)

} IF-THEN-ELSE statement (pp. 121–124)

**END.**

---

---

# PREFACE

We wrote this book hoping that it would help a wide variety of people to learn to program a computer using the Pascal language. We were motivated partly by the belief that the ability to program a computer will become increasingly important in the future—even for people who do not consider themselves to be computer scientists. There is a tendency for many people to view learning how to program as a vocational skill—much like the ability to use a calculator or a typewriter. But there is a growing body of opinion that suggests that the ability to develop and debug a computer program contributes to the development of human problem-solving skills and to the improvement of the thinking process itself. An essential aspect of our approach to teaching Pascal involves problem-solving techniques that are language independent.

The text is intended for use in a beginning-level, one-semester Pascal course and for use by individuals who desire to teach themselves to program. In addition to teaching Pascal, we also attempt to introduce the learner to problem-solving techniques, which include the use of a simple, flexible algorithmic language. In creating even a moderately complex program, the process is always complicated by the idiosyncrasies of the dialect of the language available. For that reason, we prefer to teach one how to develop a solution in a simple, but flexible, algorithmic language and then to translate it into Pascal (Chapter 3). We have found such an approach indispensable in developing large-scale research and administrative applications in other languages.

Of the one hundred or more existing computer languages, Pascal has been growing in importance. This is probably because the language is highly structured and quite powerful, yet not particularly difficult. Thus, Pascal is suitable for a wide variety of applications ranging from business to the scientific.

In presenting the Pascal language, we concentrate on the Apple version of the UCSD dialect of Pascal, which is a complete environment for program development. Since the Apple modifications to UCSD Pascal are few, the text can be used effectively with any implementation of UCSD Pascal. In the first two chapters we introduce the reader to the UCSD Pascal system and its major components: the Filer, the Editor and the Compiler. A more detailed treatment of the Editor can be found in Appendix D.

We assume that readers have no prior computing experience; however, those with some experience may be able to move more quickly and tackle some of the more complex programming projects. In so far as mathematical skills are concerned, we

assume only that readers of this book have an understanding of arithmetic. Since this text is intended to be useful to a wide audience, including people with backgrounds and interests in the sciences as well as those interested in business, the social sciences, and humanities, some may find occasional sections that require more detailed mathematical treatments. Chapter 14 involves several statistical techniques and may be omitted. Also, some mathematically oriented exercises are scattered throughout the book. Those so inclined can avoid such exercises in favor of others closer to their interest.

The first ten chapters are intended for use in linear order. The remaining chapters may be selected in any order or even omitted, if time constraints require it. The only exception is the fact that Chapter 14 and the first part of Chapter 15 assume basic familiarity with Turtlegraphics, which is presented in Chapter 12. In these later chapters, we present various programming applications likely to be of interest to students of differing backgrounds. Chapter 10 introduces the CHAR and STRING data types and provides several useful examples of programming with textual data. A word frequency technique often of interest to those in linguistics is also presented. Appendix D expands considerably upon the earlier treatment of character data. In that appendix, we invite the student to join us in creating a word-processing printer routine, which, when used with the system Editor, constitutes a functioning word processor. Students are encouraged to develop that software more fully themselves as a programming project. Chapter 11 presents information on pointers and files. Chapter 12 introduces Turtlegraphics with a wide variety of application programs. In Chapter 13, we build a LOGO Turtlegraphics interpreter, which is both an interesting project for extending one's knowledge of programming in Pascal and an interesting piece of software to use. Chapter 14 presents a number of basic statistical techniques and the graphic display of their results. The first half of Chapter 15 presents various techniques for drawing maps using Turtlegraphics. The last half of that chapter demonstrates techniques for playing melodies and even harmonies on the Apple.

Finally, in Appendix C, we present a number of useful procedures and functions that we use from time to time in various chapters. This library is available on diskette from the the authors at cost. A second diskette with mapping software and data sets is also available, and its contents are explained more fully in Section 15.9 (Chapter 15).

Throughout the book, we present information that is displayed on the computer screen in a special bold type face for your convenience. However, often we cannot display an entire line as it will appear on the screen because we cannot fit all the characters in one line of text. Therefore, you will notice that the display of the UCSD Pascal system command line and subsystem option lines will often be displayed in two lines. Sometimes, in displaying the text of a program, a similar problem occurs. When we cannot fit an entire line of the program on a single line in the book, we break the line at a convenient place and put the remaining text on the next line. In order to be able to distinguish such second lines, we have right-justified them. We think you will adapt easily to these two situations. You might want to reread this paragraph after you have read four or five chapters of the text.

In the process of writing this book we have benefited from the assistance of a number of people. We are glad to have an opportunity to express our gratitude to William Teoh of the University of Alabama, Huntsville, and Evelyn Speiser of Glendale Community College, who read the manuscript and offered numerous useful suggestions; to Nancy G. Haigh, who read and reread drafts of the manuscript in order to improve its clarity and style; to Betty O'Bryant of PWS, who orchestrated the various design and production tasks that transform a manuscript into a book and made it look easy; and to Karín Hammann, who meticulously edited the copy. We are also happy to acknowledge the assistance of Robert Holloway, University of Wisconsin; Phillip D. Jackson, Appalachian Microsystems; William F. Ward, Indian River Community College; and F. J. Lopez-Lopez, Southwestern College.



---

---

# CONTENTS

<b>1</b>	<b>THE LANGUAGE AND THE OPERATING SYSTEM</b>	<b>1</b>
<b>1.1</b>	Text Structure and Scope	<b>1</b>
<b>1.2</b>	Pascal as a Language	<b>2</b>
<b>1.3</b>	The Pascal System	<b>3</b>
<b>1.4</b>	Using the Pascal System: Guided Exercises	<b>4</b>
<b>1.5</b>	Hierarchy in the Pascal System	<b>20</b>
	Review	<b>22</b>
<b>2</b>	<b>RUNNING SIMPLE PROGRAMS</b>	<b>25</b>
<b>2.1</b>	Elements of a Complete Program	<b>25</b>
<b>2.2</b>	Entering a Program	<b>27</b>
<b>2.3</b>	Compiling a Program	<b>31</b>
<b>2.4</b>	Executing a Program	<b>32</b>
<b>2.5</b>	Changing a Program	<b>34</b>
<b>2.6</b>	Dealing with Errors	<b>36</b>
<b>2.7</b>	Debugging	<b>40</b>
<b>2.8</b>	A Word about Program Style	<b>41</b>
<b>2.9</b>	Saving a Program	<b>41</b>
	Review	<b>43</b>

<b>3</b>	<b>PROGRAM DEVELOPMENT AND PROBLEM SOLVING</b>	<b>45</b>
<b>3.1</b>	Programming and Problem Solving	<b>45</b>
<b>3.2</b>	A Problem-Solving Model	<b>46</b>
<b>3.3</b>	The Nature of an Algorithm	<b>47</b>
<b>3.4</b>	Illustrating Pseudocode	<b>49</b>
<b>3.5</b>	Coding the Algorithm	<b>56</b>
<b>3.6</b>	Completing the Job: Documentation	<b>60</b>
	Review	<b>65</b>
<b>4</b>	<b>FUNDAMENTALS OF DATA MANIPULATION</b>	<b>67</b>
<b>4.1</b>	Performing Operations upon Data	<b>67</b>
<b>4.2</b>	Storing Data in Memory: Variables and Constants	<b>68</b>
<b>4.3</b>	Arithmetic Operations with Numeric Constants	<b>71</b>
<b>4.4</b>	Mixed-Mode Expressions	<b>75</b>
<b>4.5</b>	Input/Output Procedures	<b>79</b>
<b>4.6</b>	Creating a Data Type	<b>83</b>
<b>4.7</b>	Pascal Identifiers	<b>84</b>
<b>4.8</b>	Syntax Diagrams	<b>85</b>
	Review	<b>88</b>
	Programming Exercises	<b>92</b>
<b>5</b>	<b>LOOPS AND FILES</b>	<b>94</b>
<b>5.1</b>	The Structured Statement	<b>94</b>
<b>5.2</b>	Defining a Data Set	<b>95</b>
<b>5.3</b>	Using Text Files for Data Sets	<b>96</b>
<b>5.4</b>	WHILE Loops in Pseudocode	<b>97</b>

<b>5.5</b>	<b>Relational Operators and Boolean Values</b>	<b>99</b>
<b>5.6</b>	<b>WHILE Loops in Pascal</b>	<b>100</b>
<b>5.7</b>	<b>Creating Text Files with Programs</b>	<b>106</b>
<b>5.8</b>	<b>Files That Contain Numeric Data</b>	<b>108</b>
	Review	<b>112</b>
	Programming Exercises	<b>116</b>
 <b>6</b>	 <b>CONDITIONAL STATEMENTS</b>	 <b>118</b>
<b>6.1</b>	<b>Program Branching</b>	<b>118</b>
<b>6.2</b>	<b>Single Alternative Decision Structure</b>	<b>118</b>
<b>6.3</b>	<b>Double Alternative Decision Structure</b>	<b>121</b>
<b>6.4</b>	<b>Nested Decision Structures</b>	<b>126</b>
<b>6.5</b>	<b>Compound Boolean Expressions</b>	<b>127</b>
<b>6.6</b>	<b>Multiple Alternative Decision Structures</b>	<b>129</b>
	Review	<b>135</b>
	Programming Exercises	<b>138</b>
 <b>7</b>	 <b>LOOPS AND ARRAYS</b>	 <b>140</b>
<b>7.1</b>	<b>Algorithms and Data</b>	<b>140</b>
<b>7.2</b>	<b>Loops in General</b>	<b>140</b>
<b>7.3</b>	<b>Syntax for Repeat Loops</b>	<b>144</b>
<b>7.4</b>	<b>Counters and Accumulators</b>	<b>148</b>
<b>7.5</b>	<b>Indexed Loops</b>	<b>153</b>
<b>7.6</b>	<b>Using the Loop Index</b>	<b>156</b>
<b>7.7</b>	<b>Structured Data Types</b>	<b>160</b>
<b>7.8</b>	<b>Subscripts and Arrays</b>	<b>161</b>
<b>7.9</b>	<b>Using Arrays</b>	<b>161</b>
<b>7.10</b>	<b>Sorting with an Array</b>	<b>164</b>

<b>7.11</b>	User-Defined Ordinal Types	<b>167</b>
	Review	<b>171</b>
	Programming Exercises	<b>173</b>

## **8** FUNCTIONS AND PROCEDURES **176**

<b>8.1</b>	Modular Programs	<b>176</b>
<b>8.2</b>	Built-In Functions	<b>177</b>
<b>8.3</b>	User-Defined Procedures and Functions	<b>181</b>
<b>8.4</b>	Choosing between Functions and Procedures	<b>190</b>
<b>8.5</b>	A Template for Modular Programs	<b>193</b>
<b>8.6</b>	Designing a Modular Program	<b>197</b>
<b>8.7</b>	Random Numbers	<b>200</b>
<b>8.8</b>	Recursion	<b>205</b>
	Review	<b>212</b>
	Programming Exercises	<b>215</b>

## **9** ARRAYS AND SETS **218**

<b>9.1</b>	More Data Structures	<b>218</b>
<b>9.2</b>	Elementary Statistical Measures Using Arrays	<b>218</b>
<b>9.3</b>	Using an Array as an Accumulator	<b>226</b>
<b>9.4</b>	Two-Dimensional Arrays	<b>228</b>
<b>9.5</b>	Matrix Operations	<b>230</b>
<b>9.6</b>	Order Exploding Using Arrays	<b>233</b>
<b>9.7</b>	Subrange Types	<b>236</b>
<b>9.8</b>	Sets	<b>237</b>
<b>9.9</b>	Set Operations	<b>241</b>
	Review	<b>244</b>
	Programming Exercises	<b>246</b>

<b>10</b>	<b>PROCESSING CHARACTER DATA</b>	<b>248</b>
<b>10.1</b>	The ASCII Coding System	<b>248</b>
<b>10.2</b>	Character and String Data Types	<b>249</b>
<b>10.3</b>	Sets of Character Data	<b>250</b>
<b>10.4</b>	Base Conversions	<b>253</b>
<b>10.5</b>	Random Words	<b>255</b>
<b>10.6</b>	Strings	<b>256</b>
<b>10.7</b>	String Length	<b>257</b>
<b>10.8</b>	Using String and Character Data Together	<b>258</b>
<b>10.9</b>	Text File Input/Output Using Strings	<b>259</b>
<b>10.10</b>	Application: Word-Frequency Distribution	<b>262</b>
<b>10.11</b>	Converting Long Integers to Strings	<b>273</b>
<b>10.12</b>	Lower-Case Conversion	<b>274</b>
<b>10.13</b>	Other String Operations	<b>275</b>
	Review	<b>277</b>
	Programming Exercises	<b>279</b>
<b>11</b>	<b>DATA STRUCTURES: RECORDS, FILES, AND LINKED LISTS</b>	<b>282</b>
<b>11.1</b>	Completing the Data Tree	<b>282</b>
<b>11.2</b>	The Record and the WITH Statement	<b>283</b>
<b>11.3</b>	Files in General	<b>285</b>
<b>11.4</b>	Designing a Record System	<b>285</b>
<b>11.5</b>	A New Variable Type: The Pointer	<b>295</b>
<b>11.6</b>	Constructing a Linked List	<b>296</b>
<b>11.7</b>	Modifying a Singly Linked List	<b>301</b>
<b>11.8</b>	Binary Trees	<b>304</b>
	Review	<b>311</b>
	Programming Exercises	<b>313</b>

<b>12</b>	<b>INTRODUCTION TO TURTLEGRAPHICS</b>	<b>315</b>
<b>12.1</b>	Turtlegraphics	<b>315</b>
<b>12.2</b>	The Graphics Screen	<b>316</b>
<b>12.3</b>	Drawing Simple Geometric Figures	<b>320</b>
<b>12.4</b>	Using Recursion to Produce Graphics Figures	<b>328</b>
<b>12.5</b>	Printing Graphic Images	<b>329</b>
<b>12.6</b>	Controlling Printer Output	<b>331</b>
<b>12.7</b>	Plotting a Time Series	<b>334</b>
	Review	<b>339</b>
	Programming Exercises	<b>340</b>
<b>13</b>	<b>BUILDING A LOGO TURTLEGRAPHICS INTERPRETER</b>	<b>342</b>
<b>13.1</b>	Introduction to LOGO	<b>342</b>
<b>13.2</b>	Turtlegraphics in LOGO	<b>343</b>
<b>13.3</b>	Creating a Turtlegraphics Interpreter	<b>345</b>
<b>13.4</b>	Parsing	<b>349</b>
<b>13.5</b>	Other TGI Subprocedures	<b>352</b>
<b>13.6</b>	Using the TGI Program	<b>360</b>
	Review	<b>361</b>
	Programming Exercises	<b>362</b>
<b>14</b>	<b>MORE DATA MANIPULATION</b>	<b>364</b>
<b>14.1</b>	Entry of Numerical Data	<b>364</b>
<b>14.2</b>	Output of Numerical Data	<b>368</b>
<b>14.3</b>	Producing X-Y Plots	<b>372</b>
<b>14.4</b>	Producing Bar Graphs	<b>376</b>
<b>14.5</b>	Producing Histograms	<b>379</b>

<b>14.6</b>	Fitting Data to a Straight Line	<b>385</b>
	Review	<b>390</b>
	Programming Exercises	<b>391</b>

## **15 MAKING MAPS AND MUSIC 392**

<b>15.1</b>	Computer Mapping	<b>392</b>
<b>15.2</b>	Creating a Map Coordinates File	<b>397</b>
<b>15.3</b>	Finding the Midrange of the Coordinates	<b>399</b>
<b>15.4</b>	Drawing a Map	<b>401</b>
<b>15.5</b>	Printing Graphic Displays	<b>405</b>
<b>15.6</b>	Using the Apple to Make Music	<b>406</b>
<b>15.7</b>	Making Melodies	<b>407</b>
<b>15.8</b>	Making Harmonies	<b>410</b>
<b>15.9</b>	Map Resources	<b>414</b>
	Review	<b>416</b>
	Programming Exercises	<b>417</b>

## **APPENDICES 419**

<b>A</b>	Pascal Reserved Words	<b>420</b>
<b>B</b>	Pascal Compiler Error Messages	<b>422</b>
<b>C</b>	Library of Programs	<b>427</b>
<b>D</b>	Word Processing	<b>444</b>
<b>E</b>	ASCII Codes and Their Meaning	<b>474</b>
<b>F</b>	Diagnosing Run-Time Errors	<b>476</b>
<b>G</b>	Handling Input/Output Errors within Programs	<b>480</b>

## **INDEX 483**

# 1

---

# THE LANGUAGE AND THE OPERATING --- SYSTEM

## 1.1 TEXT STRUCTURE AND SCOPE

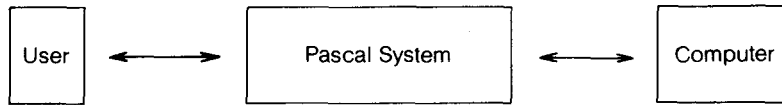
This text provides an introduction to problem solving in the environment of the Apple UCSD Pascal system. Our primary goal throughout the text is to help you develop both problem-solving skills and proficiency in the Pascal programming language. It is necessary, however, to spend some time discussing the Apple UCSD Pascal operating system, hereafter called the *Pascal system*. The letters *UCSD* refer to the University of California at San Diego, at which a popular dialect of Pascal was developed. This dialect runs on a number of different computers. In this book, we will concentrate on the Apple implementation of UCSD Pascal, although we hasten to point out that the Apple version has quite a bit in common with other UCSD Pascal dialects.

The Pascal system acts as an interface between the computer and the user (Figure 1.1). While we are communicating with the Pascal system, it in turn is communicating with the computer on a more fundamental level. The Pascal system responds to the user's instructions in order to perform all the necessary housekeeping tasks that are associated with the development and manipulation of programs.

People who have programmed in BASIC may find the details of a new language and a complex operating system burdensome. In BASIC, the housekeeping tasks (entering, editing, saving, listing, and running programs) are simple and are quickly learned, and the user can concentrate on the language. With Apple Pascal, the housekeeping tasks require more explicit attention, which fact may complicate early efforts to learn the language. We recognize this obstacle, and attempt to minimize your frustration by providing in the early chapters step-by-step directions for performing the essential tasks. You can obtain more details on the Apple UCSD Pascal system from



Figure 1.1  
Relation of User to Computer



the *Apple Pascal Operating System Reference Manual*.<sup>\*</sup> However, like other computer manuals, this document assumes that you have considerable familiarity with the system. We will introduce you to the essentials of the Pascal system as you need to know them. After that, you may delve into the system manual for further guidance.

Our emphasis will be on writing well-structured programs in UCSD Pascal as implemented on the Apple microcomputer. This implementation includes UCSD Pascal<sup>†</sup> and certain extensions to the language that were developed by Apple. No previous programming experience will be assumed. If you have already programmed, you may need to give up some habits you developed while programming with other languages. If you have never programmed, you are fortunate that your first experience is with a structured language such as Pascal.

In this chapter, we assume that you are familiar with certain basic terms used in computing, such as

- Computer language
- Program
- Computer memory
- Central processing unit (CPU)
- Input/output (I/O) devices
- File
- Software

If you are not familiar with these terms, look up their definitions in the review section at the end of this chapter. Additional computer terminology is defined as it is introduced in each section.

## 1.2 PASCAL AS A LANGUAGE

The Pascal language was developed in 1968 by Professor Niklaus Wirth at the Eidgenossische Technische Hochschule in Zurich, Switzerland. The official description

<sup>\*</sup>*Apple Pascal Operating System Reference Manual* (Cupertino, Calif.: Apple Computer Co., 1980).

<sup>†</sup>UCSD PASCAL is a trademark of the Regents of the University of California.