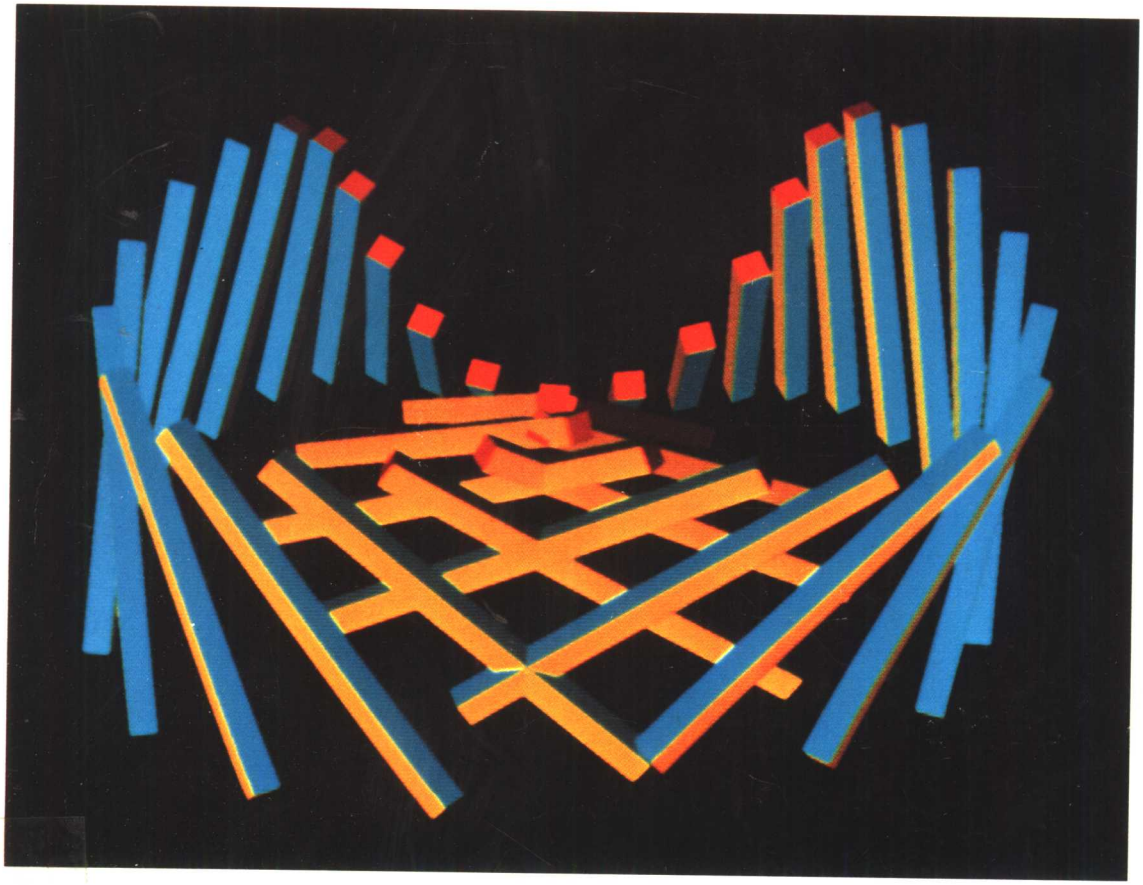
UCSD Pascal

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 • Duxbury Press •
Statler Office Building • 20 Park Plaza • Boston, Massachusetts 02116

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;
                  TOTDAYS : LONG;             } global variable declarations (pp. 57, 182)

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

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
    END                                       } CASE 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

END.
```

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

assignment statement (p. 59)

IF-THEN-ELSE statement (pp. 121–124)

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

1 THE LANGUAGE AND THE OPERATING SYSTEM

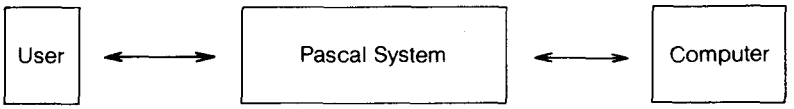
1.1 TEXT STRUCTURE AND SCOPE

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.** 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† 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

**Apple Pascal Operating System Reference Manual* (Cupertino, Calif.: Apple Computer Co., 1980).

†UCSD PASCAL is a trademark of the Regents of the University of California.