

PASCAL, PROGRAMMING, AND PROBLEM SOLVING

A SYSTEMATIC APPROACH

MARIO GONZALEZ

KAY ROBBINS

PASCAL, PROGRAMMING, AND PROBLEM SOLVING: A SYSTEMATIC APPROACH

Mario J. Gonzalez, Jr.
and
Kay A. Robbins

HOLT, RINEHART AND WINSTON, INC.
New York Chicago San Francisco Philadelphia
Mexico City Rio de Janeiro Madrid
Montreal Toronto London Sydney Tokyo

Copyright © 1988 by Holt, Rinehart and Winston, Inc.

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

Requests for permission to make copies of the work should be mailed to: Permissions, Holt, Rinehart and Winston, Inc. 111 Fifth Avenue, New York, New York 10003.

Printed in the United States of America

Library of Congress Cataloging-in-Publication Data

Gonzalez, Mario J.

Pascal, programming, and problem solving: A Systematic Approach.

Includes index.

1. PASCAL (Computer program language) 2. Structured programming. I. Robbins, Kay A. II. Title.

QA76.73.P2G675 1988 005.13'3 87-19634

ISBN 0-03-060307-2

8 9 0 039 9 8 7 6 5 4 3 2

Printed in the United States of America

Holt, Rinehart and Winston, Inc.
The Dryden Press
Saunders College Publishing

PREFACE

This book is intended for use in an introductory course in computer science. We emphasize the development of problem-solving skills in our introduction to the basic concepts of programming. Although Pascal is used for the actual implementation of programs, the book contains many examples in which structured pseudocode is used to represent preliminary solutions. We have taken an approach that is consistent with the guidelines given by the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers for an introductory course. The International Standards Organization (ISO) standard Pascal is used throughout the text.

Whereas most elementary texts only examine small programs and program segments, this book focuses on the complete solution of problems—from the initial problem statement, through a Requirements Analysis and the development of a test plan, to a final program with documentation. We feel that if one is truly to learn top-down design, one must actually see how large problems are analyzed. By the end of a first course in computer science, a student should be able to analyze a problem of reasonable complexity and produce a well-organized working program of a few hundred lines.

Throughout most of this book, we develop two instructional threads that should be studied concurrently. Each thread includes chapters that examine basic Pascal features that can be studied in a one-semester course. Each of these chapters is matched with a chapter that deals with fundamental issues in problem solving, software engineering, the design of algorithms, and good programming practice. The material in these chapters should be introduced when the material in the corresponding Pascal chapter is studied.

In the first chapter, we introduce some fundamental concepts of com-

puting, including the structure of a simple computer system and of a Pascal program. We also introduce the notation we use to design algorithms throughout the book. At the end of Chapter 1, we give a sample Pascal program that can be used as a copying assignment by the students during the first week of class. In Chapter 2 we begin our formal discussion of Pascal. We introduce simple data types, arithmetic, the assignment statement, and simple input and output. By the end of this chapter, students should be able to write complete programs to do simple calculations.

Chapters 3 and 4 introduce our two-thread approach. In Chapter 3 we examine Pascal decision structures, including boolean expressions, relational operators, and simple and nested **if-then-else** statements. Chapter 4, the companion to Chapter 3, introduces the formal steps in problem solving and discusses the initial formulation of problems.

Chapter 5 introduces functions and procedures. The topics in the first half of this chapter permit students to replace algorithmic solution steps with functions or procedures. The second half of the chapter examines somewhat more advanced topics: scope, side effects, external procedures, and recursion. In Chapter 6, the partner to Chapter 5, we examine the development of general solution strategies and perform a detailed Requirements Analysis for two sample problems.

Chapter 7 covers looping and program control (**while**, **for**, and **repeat**), and Chapter 8 presents a detailed treatment of the design of algorithms using a top-down approach. This chapter includes an implementation of some of the solution strategies developed in Chapter 6. In Chapter 8 we also look at various techniques that can be used to debug programs when errors occur during translation, linking, and execution. Chapter 9 stands by itself and provides an in-depth look at different types of files and at the differences between terminal and nonterminal I/O.

The two-thread approach is resumed in Chapter 10, where we introduce arrays and look at character data and strings. In Chapter 11, the companion chapter, we examine programming style and documentation and the role of these topics in the implementation of correct and reliable computer programs. Chapter 12 looks at user-defined data types, and the companion chapter, Chapter 13, examines test plans and the manner in which data should be prepared for program testing.

The remaining three chapters stand by themselves. Chapter 14 extends the subject of Chapter 10 by looking at multi-dimensional arrays. Chapter 15 gives an introduction to data structures by examining the use of pointers and linked lists. The final chapter, Chapter 16, presents searching and sorting as an introduction to the analysis of algorithms.

Each chapter in this book begins with a statement of objectives and a list of keywords. Exercises with answers are provided after each idea is in-

roduced. The problems at the ends of the chapters are generally more difficult than the exercises in the body of the text and often include extensions of the ideas developed in the chapters. In some of the later chapters, there are major projects. An instructor should try to select one of these projects for the students to complete during the semester.

The Pascal keywords are given in boldface type, and the predefined identifiers are shown in italics. Pascal syntax and important programming concepts are shown in shaded boxes. We have used standard Pascal throughout this book. Any usage of Pascal that may deviate from the ISO Standard is clearly prefaced with a few words of warning. The following appendices are included at the end of the book:

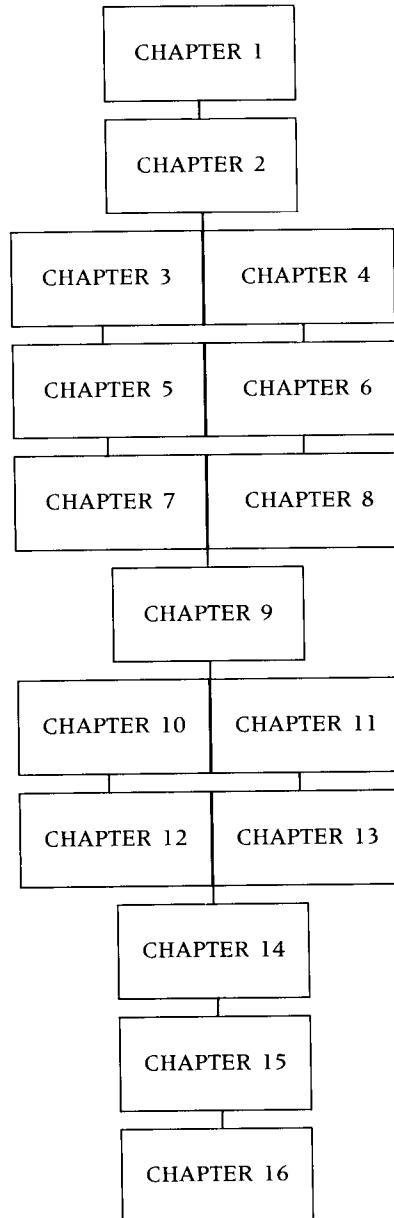
- A** ASCII Codes
- B** Pascal Keywords
- C** Pascal Operator Precedence
- D** Pascal Predefined Functions and Procedures

The following special features of this book are particularly significant in an introductory computer science sequence:

- 1** The two-thread organization allows an instructor to integrate current techniques in problem solving, software engineering, and good programming practice with the introduction of the Pascal syntax. This provides flexibility and allows the instructor to tailor the course to particular objectives.
- 2** Structured pseudocode is used consistently and systematically as part of problem solving throughout the book.
- 3** Well-chosen Pascal examples and good problem sets help students gain confidence and skill as they proceed through the course.
- 4** Shaded boxes for exercises, key points, definitions, syntax, and chapter objectives emphasize the important points in each chapter.
- 5** Chapter keywords are listed at the beginning of each chapter, and a glossary at the end of the chapter provides a definition for each of these terms.
- 6** The text provides a wide range of exercises and problems, from short and simple ones to those that are suitable for term projects.

-
- 7 Complete programs, including documentation, provide a model for students to emulate.
 - 8 The book follows ACM guidelines for CS-1. It uses standard Pascal, but points out aspects of the language that are implementation dependent or for which many systems are likely to vary from the standard.
 - 9 A complete chapter is devoted to program testing. The approach suggested in this chapter can be easily understood and implemented by students in a first course in computer science.

HOW TO USE THIS BOOK



CONTENTS

1 INTRODUCTION TO COMPUTING	1
1.1 PROBLEM SOLVING	2
1.2 THE COMPUTER AS A PROBLEM SOLVER	2
1.3 SOME TERMINOLOGY	3
1.4 COMPUTER ORGANIZATION	4
1.5 INTRODUCING THE COMPUTER PROGRAM	8
1.6 HOW TO RUN A PROGRAM	13
1.7 THE STEP NOTATION FOR ALGORITHM DESIGN	14
1.7.1 <i>The Sequence Operation</i>	15
1.7.2 <i>Selection</i>	16
1.7.3 <i>Iteration</i>	17
1.8 SUMMARY	19
GLOSSARY	20
ANSWERS TO EXERCISES	22
 2 THE BUILDING BLOCKS FOR SIMPLE PROGRAMS	 23
2.1 INTRODUCTION	24
2.2 IDENTIFIERS	26
2.3 NUMERIC DATA	28
2.4 CHARACTER AND BOOLEAN DATA	29
2.5 CONSTANTS	30
2.6 DECLARATION OF VARIABLES	31

2.7	ARITHMETIC AND ARITHMETIC EXPRESSIONS	32
2.8	ASSIGNMENT STATEMENTS	35
2.9	PASCAL PREDEFINED FUNCTIONS	39
2.10	SIMPLE INPUT AND OUTPUT	44
2.11	FORMATTED OUTPUT	49
2.12	PUTTING A PROGRAM TOGETHER	53
2.13	SUMMARY	58
	GLOSSARY	59
	ADDITIONAL PROBLEMS	61
	ANSWERS TO EXERCISES	63

3 DECISION STRUCTURES IN PASCAL **66**

3.1	INTRODUCTION TO RELATIONAL OPERATIONS AND CONDITIONS	67
3.2	BOOLEAN EXPRESSIONS	68
3.3	THE SIMPLE if-then-else STATEMENT	72
3.4	NESTED if-then-else	75
3.5	THE case STATEMENT	78
3.6	COMPOUND STATEMENTS AND THE if-then-else	80
3.7	SOME COMPLETE PROGRAMS	81
3.8	SUMMARY	86
	GLOSSARY	86
	ADDITIONAL PROBLEMS	87
	ANSWERS TO EXERCISES	91

4 FROM PROBLEM TO PROGRAM—A PERSPECTIVE **93**

4.1	INTRODUCTION	94
4.2	AN OVERALL TIMETABLE FOR PROBLEM SOLVING AND PROGRAMMING	94
4.3	PROBLEM ANALYSIS	97
4.4	PROGRAM DEVELOPMENT	99
4.5	IMPLEMENTATION	102
4.6	TESTING AND DEBUGGING	105
4.7	DOCUMENTATION	106
4.8	OVERVIEW OF THE PROBLEM-SOLVING PROCESS	109
4.9	THE EVOLUTION OF COMPUTING	110

4.10	SUMMARY	112
	GLOSSARY	112
	ADDITIONAL PROBLEMS	113

5 FUNCTIONS AND PROCEDURES 115

5.1	INTRODUCTION	116
5.2	USER-DEFINED FUNCTIONS	116
5.3	THE FUNCTION CALL	118
5.4	USER-DEFINED PROCEDURES	123
5.5	THE PROCEDURE CALL	126
5.6	VALUE PARAMETERS AND VARIABLE PARAMETERS	128
5.7	MORE ON PARAMETER PASSING	134
5.8	GLOBAL VARIABLES, LOCAL VARIABLES, AND THE SCOPE OF AN IDENTIFIER	136
5.9	SIDE EFFECTS	141
5.10	EXTERNAL PROCEDURES AND FORWARD DIRECTIVE (Optional Topic)	142
5.11	RECURSION (Optional Topic)	144
5.12	SUMMARY	147
	GLOSSARY	148
	ADDITIONAL PROBLEMS	149
	ANSWERS TO EXERCISES	152

6 PROBLEM ANALYSIS 157

6.1	INTRODUCTION	158
6.2	UNDERSTANDING THE PROBLEM	158
6.3	INPUT DATA	161
6.4	OUTPUT DATA	166
6.5	SPECIAL CASES	167
6.6	SOLUTION APPROACH	167
	6.6.1 <i>Selection of a General Strategy</i>	168
	6.6.2 <i>Is a Computer Necessary?</i>	171
	6.6.3 <i>Does a Solution Already Exist?</i>	172
6.7	EXAMPLES	174
6.8	SUMMARY	179
	GLOSSARY	179

ADDITIONAL PROBLEMS	180
ANSWERS TO EXERCISES	183
 7 LOOPING STRUCTURES IN PASCAL	 184
7.1 THE while STATEMENT	185
7.2 THE for STATEMENT	188
7.3 THE repeat STATEMENT	195
7.4 NESTED LOOPS	196
7.5 TERMINATION OF INPUT DATA	198
7.6 PRINTING A CALENDAR	201
7.7 SUMMARY	209
GLOSSARY	210
ADDITIONAL PROBLEMS	211
ANSWERS TO EXERCISES	218
 8 PROGRAM DEVELOPMENT, IMPLEMENTATION, AND DEBUGGING	 221
8.1 INTRODUCTION	222
8.2 ALGORITHMS	222
8.3 TOP-DOWN DESIGN	223
8.4 PROGRAM DEBUGGING	239
8.4.1 <i>Translation Errors</i>	239
8.4.2 <i>Linkage Errors</i>	242
8.4.3 <i>Run Time Errors</i>	243
8.4.4 <i>Tracing</i>	245
8.4.5 <i>Instrumenting a Program</i>	249
8.5 SUMMARY	254
GLOSSARY	254
ADDITIONAL PROBLEMS	256
ANSWERS TO EXERCISES	258
 9 INPUT AND OUTPUT	 259
9.1 INTRODUCTION	260
9.2 TEXT FILES	267

9.3	TEXT FILES AND NUMERIC DATA	269
9.4	FILE MARKERS	274
9.5	INPUT AND OUTPUT USING FILE MARKERS (Optional Topic)	277
9.6	INPUT AND OUTPUT FROM THE TERMINAL	280
9.7	EXTERNAL FILES	281
9.8	A FILE MERGE	283
9.9	SUMMARY	286
	GLOSSARY	286
	ADDITIONAL PROBLEMS	287
	ANSWERS TO EXERCISES	289
10	ARRAYS	293
10.1	INTRODUCTION TO ARRAYS	294
10.2	ARRAY DEFINITIONS	297
10.3	ARRAYS AND LOOPS	298
10.4	THE type SECTION	305
10.5	ARRAYS AS PARAMETERS	307
10.6	FILES, FUNCTIONS, AND PROCEDURES AS PARAMETERS	310
10.7	CHARACTER DATA AND THE STRING TYPE	312
10.8	SUMMARY	317
	GLOSSARY	318
	ADDITIONAL PROBLEMS	319
	SUGGESTED PROJECT: COMPUTERIZED FORM LETTERS (A MAIL MERGE)	325
	ANSWERS TO EXERCISES	327
11	PROGRAMMING STYLE AND DOCUMENTATION	330
11.1	INTRODUCTION	331
11.2	DOCUMENTATION	333
	11.2.1 <i>Program and Procedure Headers</i>	333
	11.2.2 <i>Meaningful Identifiers</i>	334
	11.2.3 <i>Intermediate Comments</i>	336
11.3	READABILITY	339
	11.3.1 <i>Blanks and Blank Lines</i>	339
	11.3.2 <i>Indentation</i>	341
	11.3.3 <i>Listing Control</i>	342

11.4	RELIABILITY	343
11.5	RELIABILITY FEATURES OF PASCAL	348
11.6	MODULARITY	351
11.7	PORTABILITY	354
11.8	SUMMARY	355
	GLOSSARY	356
	ANSWERS TO EXERCISES	357
12	USER-DEFINED DATA TYPES	359
12.1	STRICT TYPING AND ASSIGNMENT COMPATIBILITY	360
12.2	SUBRANGE SPECIFICATIONS AND ENUMERATED TYPES	362
12.3	SETS	366
12.4	RECORDS	372
12.5	TRANSACTION PROCESSING: AN APPLICATION OF FILES AND RECORDS	378
	12.5.1 <i>Problem Statement</i>	380
	12.5.2 <i>Input and Output Data</i>	381
	12.5.3 <i>A Basic Test Plan</i>	382
	12.5.4 <i>Algorithm Design</i>	383
12.6	ABSTRACT DATA TYPES	389
12.7	SUMMARY	390
	GLOSSARY	390
	ADDITIONAL PROBLEMS	391
	SUGGESTED PROJECT: TRANSACTION PROCESSING	393
	ANSWERS TO EXERCISES	394
13	PROGRAM TESTING	396
13.1	INTRODUCTION	397
13.2	DEVELOPING A TEST PLAN	397
13.3	IMPLEMENTATION AND TESTING ALTERNATIVES	399
13.4	IDENTIFICATION OF THE CRITICAL POINTS	411
13.5	TEST DATA FOR PROGRAM TESTING	415
13.6	SUMMARY	418
	GLOSSARY	418
	ADDITIONAL PROBLEMS	419
	ANSWERS TO EXERCISES	420

14 MULTI-DIMENSIONAL ARRAYS	421
14.1 ARRAY DIMENSIONS	422
14.2 TWO-DIMENSIONAL ARRAYS IN PASCAL	423
14.3 THE GRADE-BOOK PROBLEM	426
14.3.1 <i>Data Entry</i>	429
14.3.2 <i>A Revised Design</i>	431
14.3.3 <i>Computing Each Student's Average</i>	434
14.3.4 <i>Computing the Class Average for Each Project</i>	437
14.3.5 <i>Array Cross-Sections</i>	438
14.4 PLOTTING DATA VALUES ON A GRID	439
14.5 ARRAYS IN MORE THAN TWO DIMENSIONS	442
14.6 SUMMARY	444
GLOSSARY	445
ADDITIONAL PROBLEMS	446
SUGGESTED PROJECT: COMPUTERIZED DATING	450
ANSWERS TO EXERCISES	451
 15 POINTERS AND LINKED LISTS	 453
15.1 INTRODUCTION TO DATA STRUCTURES	454
15.2 PASCAL-SUPPORTED DATA STRUCTURES	455
15.3 THE LIST	456
15.4 POINTER VARIABLES	458
15.5 REPRESENTATION OF LISTS USING POINTERS	460
15.6 ACCESSING A DYNAMIC LIST	462
15.7 INSERTION AND DELETION IN AN ORDERED LINKED LIST	469
15.8 REPRESENTATIONS OF POLYNOMIALS BY LINKED LISTS	473
15.9 SUMMARY	474
GLOSSARY	475
ADDITIONAL PROBLEMS	476
SUGGESTED PROJECT: POLYNOMIAL MANIPULATION	478
ANSWERS TO EXERCISES	479
 16 PRINCIPLES OF SEARCHING AND SORTING	 482
16.1 INTRODUCTION	483
16.2 THE LINEAR SEARCH ALGORITHM	483

16.3	EFFICIENCY OF THE LINEAR SEARCH	486
16.4	THE BINARY SEARCH ALGORITHM	487
16.5	A COMPARISON OF THE LINEAR AND BINARY SEARCH ALGORITHMS	493
16.6	THE BUBBLE SORT	495
16.7	ANALYSIS OF THE BUBBLE SORT	499
16.8	THE QUICK SORT	499
16.9	SUMMARY	507
	GLOSSARY	508
	ADDITIONAL PROBLEMS	508
	ANSWERS TO EXERCISES	514

APPENDICES

A:	ASCII CODES	515
B:	PASCAL KEYWORDS	517
C:	PASCAL OPERATOR PRECEDENCE	518
D:	PASCAL PREDEFINED FUNCTIONS AND PROCEDURES	519

INDEX	523
--------------	------------

C H A P T E R 1

INTRODUCTION

TO

COMPUTING

By the end of this chapter, you should be ready to run your first computer program. A copying exercise is provided near the end of the chapter to help you become familiar with the mechanics of running a computer program on your system.

OBJECTIVES

1. To introduce some basic terminology.
2. To analyze the operation of a simple computer as a motivation for our discussion on programming.
3. To give an overview of computing and problem solving.
4. To introduce the step notation used in this text.

KEYWORDS: Bug, compiler, debugging, documentation, execution, hardware, high-level language (HLL), high-order language (HOL), input operation, machine language, memory, object program, operating system, output operation, procedure-oriented language, processing unit, program, programmer, programming language, prompt, software, software engineering, source program, tracing, translator.