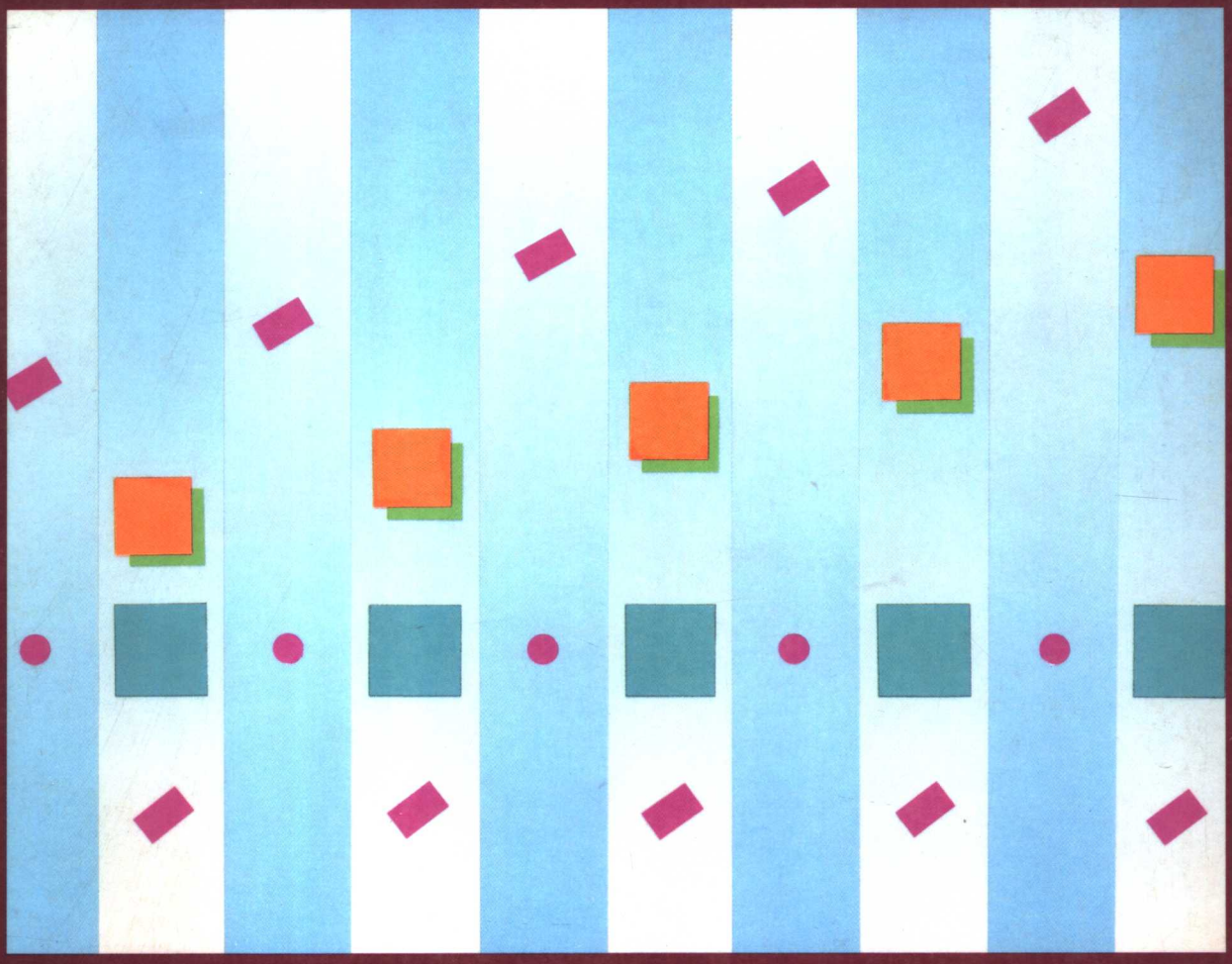


PASCAL, PROGRAMMING, AND PROBLEM SOLVING

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

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-

troduced. 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.

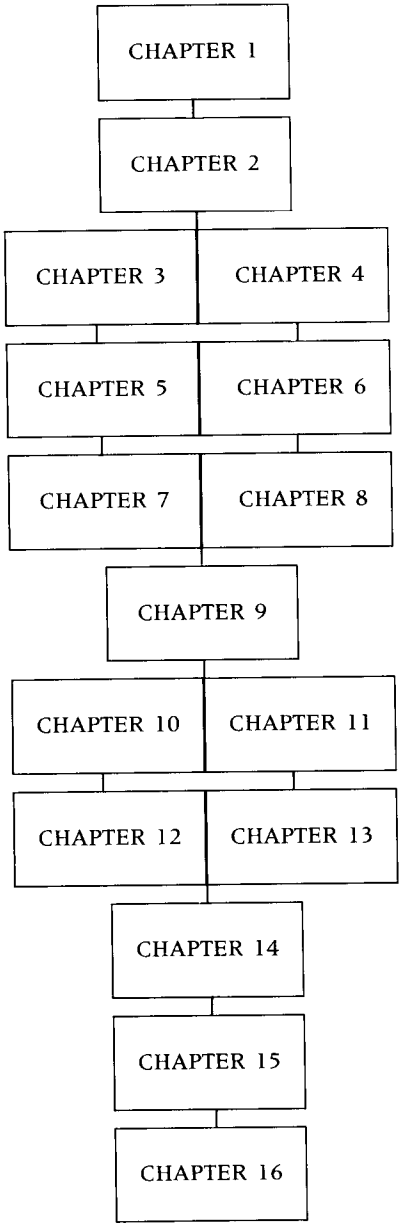
6 The text provides a wide range of exercises and problems, from short and simple ones to those that are suitable for term projects.

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

CHAPTER 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.