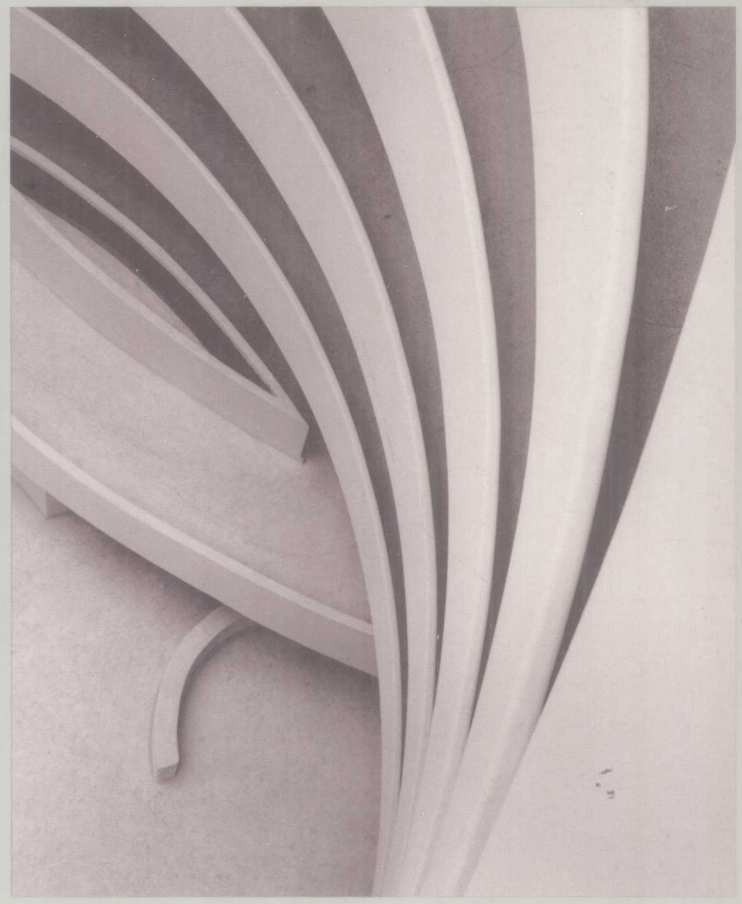
# PASCAL



SAMUEL L. MARATECK

# PASCAL



Samuel L. Marateck
*Courant Institute of Mathematical Sciences, New York University*

John Wiley & Sons, Inc.
*New York* *Chichester* *Brisbane* *Toronto* *Singapore*

*To my mother, Rita*
*and the loving memory of my father Harold Marateck*

Cover photo by George Cserna.



***Library of Congress Cataloging in Publication Data:*** Marateck, Samuel L.

Marateck, Samuel L.
Pascal / Samuel L. Marateck.
p. cm.
Includes bibliographical references.
ISBN 0-471-60546-8
1. Pascal (Computer program language) I. Title.
QA76.73.P2M335 1991
005.26'2—dc20 90-40175
CIP

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

| ASCII Code | Character | ASCII Code | Character |
|---|---|---|---|
| 0 | NULL | 76 | L |
| 7 | BELL | 77 | M |
| 8 | BACKSPACE | 78 | N |
| 9 | HORIZONTAL TAB | 79 | O |
| 10 | LINE FEED | 80 | P |
| 11 | VERTICAL TAB | 81 | Q |
| 12 | FORM FEED | 82 | R |
| 13 | CARRIAGE RETURN | 83 | S |
| 32 | SPACE | 84 | T |
| 33 | ! | 85 | U |
| 34 | " | 86 | V |
| 35 | # | 87 | W |
| 36 | $ | 88 | X |
| 37 | % | 89 | Y |
| 38 | & | 90 | Z |
| 39 | ' | 91 | [ |
| 40 | ( | 92 | \ |
| 41 | ) | 93 | ] |
| 42 | * | 94 | ^ |
| 43 | + | 95 | _ |
| 44 | , | 96 | ` |
| 45 | - | 97 | a |
| 46 | . | 98 | b |
| 47 | / | 99 | c |
| 48 | 0 | 100 | d |
| 49 | 1 | 101 | e |
| 50 | 2 | 102 | f |
| 51 | 3 | 103 | g |
| 52 | 4 | 104 | h |
| 53 | 5 | 105 | i |
| 54 | 6 | 106 | j |
| 55 | 7 | 107 | k |
| 56 | 8 | 108 | l |
| 57 | 9 | 109 | m |
| 58 | : | 110 | n |
| 59 | ; | 111 | o |
| 60 | < | 112 | p |
| 61 | = | 113 | q |
| 62 | > | 114 | r |
| 63 | ? | 115 | s |
| 64 | @ | 116 | t |
| 65 | A | 117 | u |
| 66 | B | 118 | v |
| 67 | C | 119 | w |
| 68 | D | 120 | x |
| 69 | E | 121 | y |
| 70 | F | 122 | z |
| 71 | G | 123 | { |
| 72 | H | 124 | \| |
| 73 | I | 125 | } |
| 74 | J | 126 | ~ |
| 75 | K | 127 | DEL |

| Ordinal Position | EBCDIC Graphic or Control | Ordinal Position | EBCDIC Graphic or Control |
|---|---|---|---|
| 0 | Null | 151 | p |
| 5 | Horizontal Tab | 152 | q |
| 7 | Delete | 153 | r |
| 11 | Vertical Tab | 161 | Tilde |
| 12 | Form Feed | 162 | s |
| 13 | Carriage Return | 163 | t |
| 37 | Line Feed | 164 | u |
| 39 | Escape | 165 | v |
| 47 | Bell | 166 | w |
| 64 | Space | 167 | x |
| 74 | ¢ | 168 | y |
| 75 | . | 169 | z |
| 76 | < | 192 | { |
| 77 | ( | 193 | A |
| 78 | + | 194 | B |
| 79 | \| | 195 | C |
| 80 | & | 196 | D |
| 90 | ! | 197 | E |
| 91 | $ | 198 | F |
| 92 | * | 199 | G |
| 93 | ) | 200 | H |
| 94 | ; | 201 | I |
| 95 | ¬ | 208 | } |
| 96 | - | 209 | J |
| 97 | / | 210 | K |
| 106 | \| | 211 | L |
| 107 | , | 212 | M |
| 108 | % | 213 | N |
| 109 | _ | 214 | O |
| 110 | > | 215 | P |
| 111 | ? | 216 | Q |
| 121 | Grave accent | 217 | R |
| 122 | : | 226 | S |
| 123 | # | 227 | T |
| 124 | @ | 228 | U |
| 125 | ' | 229 | V |
| 126 | = | 230 | W |
| 127 | " | 231 | X |
| 129 | a | 232 | Y |
| 130 | b | 233 | Z |
| 131 | c | 240 | 0 |
| 132 | d | 241 | 1 |
| 133 | e | 242 | 2 |
| 134 | f | 243 | 3 |
| 135 | g | 244 | 4 |
| 136 | h | 245 | 5 |
| 137 | i | 246 | 6 |
| 145 | j | 247 | 7 |
| 146 | k | 248 | 8 |
| 147 | l | 249 | 9 |
| 148 | m | 250 | \| |
| 149 | n | 255 | Eight ones |
| 150 | o | | |

The Extended Binary Coded Decimal Interchange Code (EBCDIC)

# PREFACE

This book is an outgrowth of the notes I have used teaching an introduction to computer science course using Pascal—i.e., CS I. It is written for students who have no prior knowledge of computers or programming. The book comes in two versions, this volume discusses Standard Pascal, and a companion volume discusses Turbo Pascal.

The design of the book has special significance and deserves comment. The right-hand pages contain programming material (programs, output, and tables) which most students will readily understand; this is described in detail in "To The Reader." The left-hand pages contain explanatory text. Because of the length of the programs in the book, it has not always been possible to strictly adhere to this design, therefore, occasionally a self-test question will appear on a right-hand page. It has been my experience, with books on Fortran 77 and Basic I wrote and designed the same way, that students who have used a terminal without having previously gone to class and who have studied only the right-hand pages have been able to write and successfully run programs at their first session at the terminal. Of course, students should also read the text on the left-hand pages, to understand all facets of the programming technique described.

The easiest way for students to learn program design is to see as many examples as possible. With this in mind, we start presenting examples of programming design as early as Chapter 2. Moreover, as the text progresses, we present different programming planning techniques ranging from top-down design and stepwise refinement of pseudocode, to the bottom-up approach and including the divide and conquer, exhaustive search, induction and solution by analogy approaches. The tool box approach (in which general utility modules

are developed; see Chapter 9) is also used to write some of the longer programs. For most of the longer programs, we use stepwise refinement not only of the pseudocode, but of the programs themselves. We used this approach with two things in mind: Just as it is easier to write a complicated program in stages, it is also easier to understand one written this way. Also, representing already written modules only by the procedure heading allowed us to fit most of the longer programs on one page. Thus, the student not only sees a pragmatic strategy for writing lengthy programs, but also has an easier time of assimilating the material. But isn't that one of the reasons for using procedures and functions in the first place?

An approach unique to this text is the discussion of the input buffer to explain the intricacies of input/output (see Chapter 4). This also allows us to take the mystery out of the workings of the `eof` and `eoln` functions (see Chapter 8). Another technique students find interesting is the reappearance of a given topic throughout the course of the book. With each subsequent appearance, a more sophisticated technique is used to write the program. One such topic that appears as a *leit motif* throughout the book is the writing of a calculator, starting in Chapter 7 with the appearance of a program that simply evaluates a three-character expression, and ending in Chapter 18 with a program that uses a tree to evaluate expressions that are fully parenthesized.

The programming part of the book starts in Chapter 2 with the introduction of pseudocode. We used this as an opportunity to introduce some of Pascal syntax so that the student early on becomes accustomed to where and why semicolons and the `BEGIN` and `END` delimiters are used. This also allows us to introduce procedures and top-down design at this early stage. It is an alternative approach to that of Richard Pattis in his book *Karel the Robot.*

Chapter 3 covers the basics of Pascal programming and applies the principles of the previous chapter to program design; Chapter 4 deals with input/output and concludes with a discussion of I/O using the buffer. Chapter 5 offers a consistent treatment of `FOR` loops for all types of variables so that the similarities in the different loops are revealed. We chose to discuss the `FOR` loop before the `WHILE` because of the inherent pitfalls in the latter due to infinite loops. Armed now with statements that allow us to write non-trivial programs, in Chapter 6 we give a detailed view of procedures and the implementation of top-down design. As we learn the different facets of the `IF–THEN–ELSE` and boolean operators in Chapter 6, we develop a word processing program and use stubs to design a program that converts strings of digits into integers. Chapter 7 rounds off our study of loops with a study of the `WHILE` and `REPEAT–UNTIL` loops. We use the bottom-up paradigm to design a two-dimensional random walk program. The chapter concludes with text files, so that from here on, we can write our data as files. Chapter 8, which describes functions, finishes our study of subprograms and concludes with the design and implementation of an elementary calculator. Chapter 9 formally introduces the concept of data structures in discussing arrays. We use the tool box approach here to design and implement a program that determines the frequency of letters in a sentence. The next logical step after learning arrays is to study an application of arrays—strings.

Chapter 10 begins by presenting the standard equivalents of the Turbo Pascal string functions and procedures and ends up by using them to design a global search and replace program. Chapter 11 uses one of the unique facets of Pascal—the user-defined type to write calendar programs and a program that generates Roman numerals—and concludes with the writing of a problem we first introduced in Chapter 2 and have intermittently discussed until now. Chapter 12 plans and implements a table look-up program. Chapter 13, on records, discusses many programs, one of which forms an index of words used in a paragraph. In writing this program we see that the bubble sort is stable, whereas the selection sort is not.

Chapter 14 reviews text files and covers binary files. It uses a binary file for a program that simulates a card-playing program. Chapter 15 gives a thorough introduction to recursion using a unique approach. Each time a subprogram is recursively activated, a copy of the subprogram and the stack is shown. These copies are connected so that you can trace the recursion. You will see the advantage of using this technique when you read the chapter.

The material in Chapters 17 and 18 deals with dynamic pointers. These chapters cover linked lists, stacks, trees, and queues as completely as they are treated in the CS II course.

It is a pleasure to thank Professor Max Goldstein for making available to me all the equipment of the Courant Institute Academic Computer Facility and, as always, to thank Professor Jacob T. Schwartz and Professor Goldstein for their friendship and constant support. I offer my thanks to my friends and colleagues, Professor Marsha Berger and Dr. David McQueen, for useful discussions, and a special thank you to Dr. Jeffery H. Gordon; to Professors Martin Davis, Ralph Grishman, and Olaf Widlund, on whose watch as chairmen I taught Pascal; to Professor Robert Dewar, for making available to me a copy of his screen editor DVED; to Gary Rosenblum, for his help in printing the manuscript in near camera-ready form in order to ease the reviewers' task; and to Lorenza Prignano, for taking notes on my original lectures.

Thanks go to my good friend Eugene Rodolphe for reviewing the original draft of the manuscript. I am also indebted to the following professors for reviewing either the Pascal or the Turbo Pascal manuscript: George Beekman and John Bertani, Oregon State University; C. Mark Bilodeau, United States Military Academy, West Point; Larry Crockett, Augsberg College; Amrik Dhillon, Borland International; Henry Etlinger, Rochester Institute of Technology; Charles E. Frank, Northern Kentucky University; Dale Grosvenor, Iowa State University; Taylor D. Hanna, Portland Area Community College; Herbert Koller, San Francisco University; Rose M. Laird, Northern Virginia Community College; Phyllis Lefton, Manhattanville College; Lewis Miller, Cañada College; Benedict Pollina, University of Hartford; Catherine Riccardo, Iona College; Robert G. Reynolds, Wayne State; Waldo Roth, Taylor University; Patricia Sterr, Joliet Junior College; and Delaine Timney, University of South Carolina. I am particularly indebted to Professors Bertani, Koller, Riccardo, and Sterr for invaluable phone conversations.

Samuel L. Marateck

# TO THE READER

This book has been written on the premise that it is at times easier to learn a subject from pictorial representations supported by text than from text supported by pictorial representations. With this in mind, in Chapters 3–14 we have used a double-page format for our presentation. On the left-hand page (we call it the text page) appears the text, and on the right-hand page (we call it the picture page) appears the pictorial representation, consisting mostly of programs and tables.

Each picture page was written to be as self-contained as possible, so that readers (if they so desire) may read that page first and absorb the essence of the contents of the entire double page before going on to read the text. The text page consists of a very thorough discussion of the programming techniques presented on the picture page. It refers to parts of the programs and tables on the picture page; when reference is made on the text page to a given line of print on the picture page, that line—whenever it is feasible to do so—is reproduced in the text to promote readability. Students with a previous background in programming languages and others who understand the picture page completely may find that in some chapters they can skip the text (left-hand) pages and concentrate on the picture pages.

The following techniques are used as aids in making the picture page self-contained:

1. As many as possible of the ideas discussed in the text are illustrated in the programs and tables. The captions beneath these encapsulate much of what is said in the text.

2. Shaded words in the captions describe lines underlined in the figures. To illustrate this, a version of Figure 3.7*a* is reproduced below.

```
Salary 1: = 11.4;
Salary 2: = 20.2;
WriteIn (Salaryl, Salary2)
```

FIGURE 3.7*a*. In the program, the number 11.4 is assigned to the variable `Salary 1` and 20.2 to `Salary 2` in the assignment statement

The statements `Salary 1: = 11.4;` and `Salary 2: = 20.2` are shaded to show that they are described by the words shaded in the caption. Thus they are both assignment statements.

3. To the right of most programs appears a table describing what effect certain statements in the program have on the computer's memory. For instance, the following table describes the effect that `Salaryl: = 11.4` has on the memory:

| Description | Salary 1 |
|---|---|
| `Salary 1: = 11.4;` | 11.4 |

We see from the table that this statement caused the value 11.4 to be associated with `Salary 1` in the computer's memory. The line-by-line analysis afforded by these tables should help the reader understand the program.

# CONTENTS