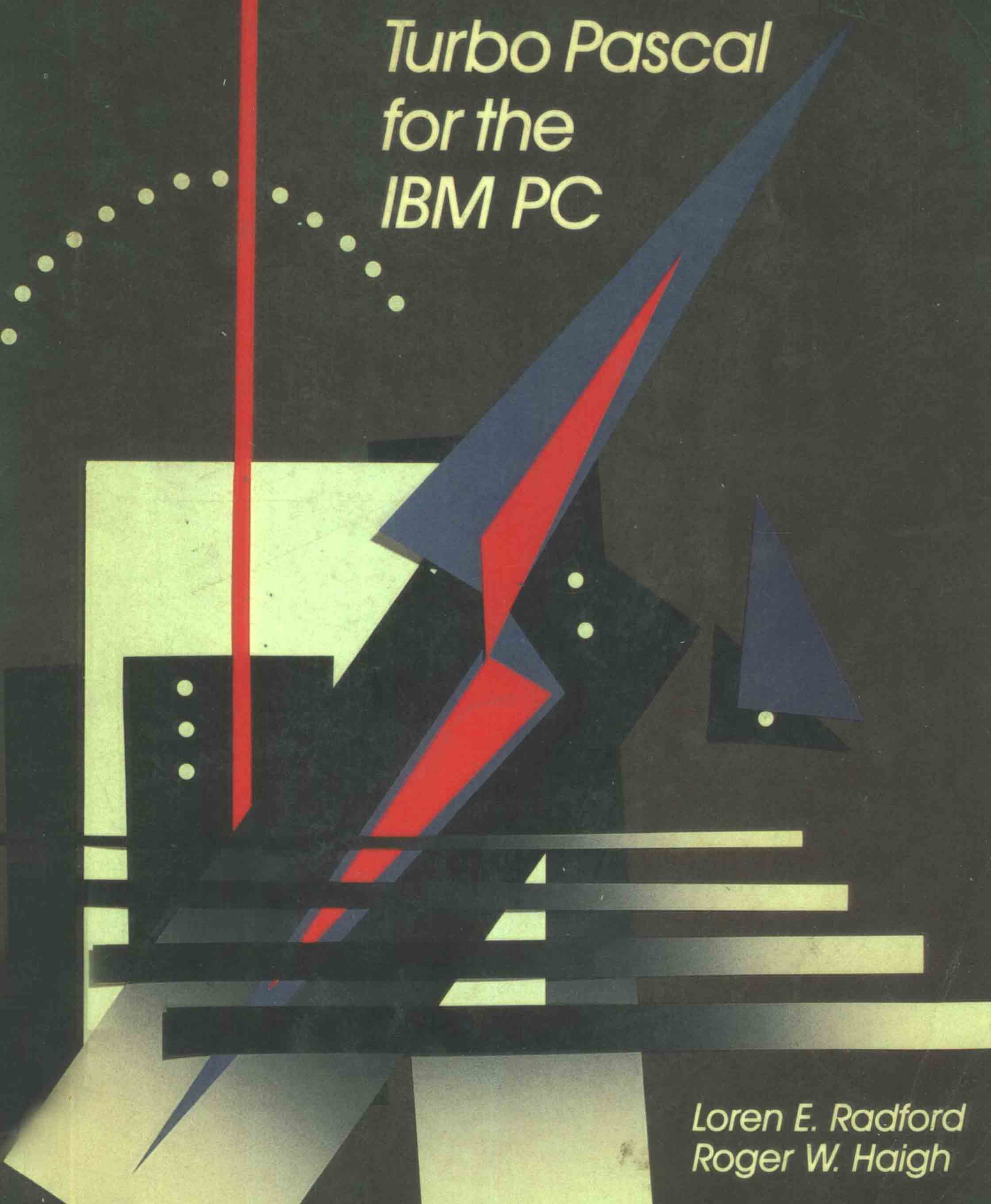
Turbo Pascal for the IBM PC



Loren E. Radford
Roger W. Haigh

Turbo Pascal for the IBM PC

Roger W. Haigh
Vermont State Colleges

PWS Computer Science
Boston

PWS PUBLISHERS

Prindle, Weber & Schmidt • Duxbury Press • PWS Engineering • Breton Publishers •
Statler Office Building • 20 Park Plaza • Boston, Massachusetts 02116

Library of Congress Cataloging-in-Publication Data

Radford, Loren E.
Turbo Pascal for the IBM PC.

Includes index.
1. IBM Personal Computer—Programming. 2. PASCAL (Computer program language) 3. Turbo Pascal (Computer program language) I. Haigh, Roger W. II. Title.
QA76.8.I2594R33 1986 005.265 85–19355
ISBN 0–534–06426–4

ISBN 0-534-06426-4

Printed in the United States of America.
86 87 88 89 90 — 5 4 3 2 1

Sponsoring Editor: *Bob Prior*
Production Coordination: *Elise Kaiser*
Production: *Editing, Design & Production, Inc.*
Interior and Cover Design: *Elise Kaiser*
Cover Printer: *Phoenix Color Corp.*
Text Printer/Binder: *Maple-Vail Book Manufacturing Group*

Cover art © 1985 by Sanjay Kothari.

Preface

This text is intended for use in a beginning-level one-semester Pascal course and by individuals who desire to teach themselves to program. In addition to presenting Pascal, we also 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 develop a solution in a simple, but flexible, algorithmic language and then translate it into Pascal (Chapter 3). We have found such an approach indispensable in developing large-scale research and administrative applications in other languages.

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 typewriter. However, 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.

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 from business to scientific.

In presenting the Pascal language, we concentrate on the Turbo version. We also show the UCSD Pascal variations for those who use that dialect. We have been impressed by the ease of use and speed of Turbo Pascal. These features, along with cost considerations, make Turbo a very desirable piece of software that an individual can afford to own. Although the earlier versions of Turbo had limited graphics capability, the most recent version (3.0) introduces a Turtlegraphics capability that considerably enhances that feature of the software.

We assume no prior computer experience on the part of the reader. However, those with some experience may be able to move more quickly and tackle some of the more complex programming projects. In the first two chapters we introduce the reader to the functional parts of the Turbo Pascal system. In presenting material on the Pascal systems we have attempted to lead the user through the essential housekeeping tasks using a system of dialogues. (A version of these dialogues for UCSD Pascal is presented in Appendix A.) Our intent is to minimize the confusion that often results when one is required to cope with an unfamiliar operating system and a new language at the same time.

In Chapter 3 we introduce programming in the context of problem solving. In this chapter a reasonably complex problem is solved and the basic elements of the algorithmic language are developed. Some attention is given to program documentation and structure. Chapter 4 includes a discussion of the input and output procedures, the basic ordinal data types, and arithmetic operations. In Chapter 5 the first structured statement (the while loop) is used in conjunction with a structured data type (the file). From this point, the developments of statements and data structures parallel one another.

Chapter 8 provides a thorough treatment of functions and procedures with attention to passage of information and variable scope. With this chapter the concept of program modularity is fully developed. Recursion is also introduced in this chapter. Chapters 9 and 10 complete the development of data structures with a treatment of arrays, records, sets, and linked structures. In these chapters certain lengthier programs are developed as applications. These applications include computing statistical measures of a data set, producing a credit record system, determining a word-frequency distribution, and writing a program documenter. The last two chapters (Chapters 11 and 12) deal with some of the graphics features of the language. The programs in these chapters provide examples of many of the programming techniques introduced throughout the text.

Program listings have consistently been presented with line numbers for ease of reference. These listings along with the program output have been reproduced from computer output. We have chosen this option to remove errors introduced during the production process. Programs have been selected to illustrate the use of statements and data structures as simply as possible. Often, prototype procedures or functions are presented that the student may use in other applications. The programs have not been extensively commented. We have relied upon modularity and selection of identifier names to make the programs reasonably self-documenting. The programs from the text are available on diskettes for the IBM PC from the publisher.

Finally, certain information has been gathered at the end of each chapter. This includes syntax diagrams, a data tree that displays the relations between the data types as they are developed, a glossary of terms in which the new concepts and terms are expressed precisely, and a listing of the new Pascal syntax. We intend for this material to provide a perspective on the parallel development of data structures and Pascal statements and to enhance the reader's ability to discuss the language with some precision.

Acknowledgments

In preparing this text we have benefited from the assistance of a number of people. We wish to thank our colleagues and reviewers, Robert Holloway of the University of Wisconsin and Evelyn Speiser of Glendale Community College. Thanks also to Nancy G. Haigh, who read and reread drafts of the manuscript in order to improve its clarity and style, and to the staff of PWS Publishers, especially Elise Kaiser and Bob Prior, and Karen Zack of EDP, Inc.

Loren E. Radford
Roger W. Haigh

Contents

1 The Language and the Operating System 1

1.1 Text Structure and Scope 1
1.2 Pascal as a Language 3
1.3 The Pascal System 4
1.4 Using the Keyboard 5
1.5 Using Turbo Pascal: Guided Exercises 6
1.6 Hierarchy in the Turbo Pascal System 12

2 Running Simple Programs 17

2.1 Elements of a Complete Program 17
2.2 Entering a Program 20
2.3 Compiling a Program 21
2.4 Executing a Program 23
2.5 Changing a Program 23
2.6 Dealing with Errors 25
2.7 Debugging 27
2.8 A Word About Program Style 28
2.9 Saving a Program 28
2.10 Configuring Your System Diskette 29

3 Program Development and Problem-Solving 33

3.1 Programming and Problem-Solving 33
3.2 A Problem-Solving Model 35
3.3 The Nature of an Algorithm 35
3.4 Pseudocode Illustrated 37
3.5 A More Complex Algorithm 43
3.6 Coding the Algorithm 48
3.7 Completing the Job: Documentation 52

4 Fundamentals of Data Manipulation 59

4.1 Performing Operations Upon Data 59
4.2 Unstructured Data Types 60
4.3 Storing Data in Memory: Variables and Constants 61
4.4 Arithmetic Operations with Numeric Constants 65
4.5 Mixed-Mode Expressions 70
4.6 Input/Output Procedures 74
4.7 Creating a Data Type 79
4.8 Pascal Identifiers 81
4.9 Syntax Diagrams 82

5 Loops and Files 93

5.1 The Structured Statement 93
5.2 Defining a Data Set 94
5.3 Using Text Files for Data Sets 95
5.4 While Loops in Pseudocode 97
5.5 Relational Operators and Boolean Values 98
5.6 While Loops in Pascal 99
5.7 Counters and Accumulators in While Loops 104
5.8 Creating Text Files with Programs 109
5.9 Files That Contain Numeric Data 112

6 Conditional Statements 125

6.1 Program Branching 125
6.2 Single Alternative Decision Structures 126
6.3 Double Alternative Decision Structures 128
6.4 Nested Decision Structures 132
6.5 Compound Boolean Expressions 134
6.6 Multiple Alternative Decision Structures 135
6.7 User-Defined Ordinal Types 141

7 Loops and Arrays 155

7.1 Algorithms and Data 155
7.2 Loops in General 156
7.3 Syntax for Repeat Loops 159
7.4 Indexed Loops 164
7.5 Using the Loop Index 167
7.6 Structured Data Types 172
7.7 Subscripts and Arrays 173
7.8 Using Arrays 173
7.9 Sorting with an Array 177
7.10 Strings 180
7.11 String Length 182
7.12 Text File Input/Output Using Strings 183

8 Functions and Procedures 195

8.1 Modular Programs 195
8.2 Built-in Functions 197
8.3 User-Defined Procedures and Functions 201
8.4 Choosing Between Functions and Procedures 210
8.5 A Template for Modular Programs 213
8.6 Designing a Modular Program 217
8.7 Random Numbers 221
8.8 Other String Operations 225
8.9 Recursion 229

9 Arrays and Records 245

9.1 More on Arrays and Data Structures 245
9.2 Subrange Types 246
9.3 Arrays and User-Defined Ordinal Types 247
9.4 Using an Array as an Accumulator 249
9.5 Application: Elementary Statistical Measures Using Arrays 252
9.6 Two-Dimensional Arrays 260
9.7 Matrix Operations 262
9.8 Application: Order Exploding Using Arrays 264
9.9 The Record and the With Statement 269
9.10 Application: Designing a Small Record System 278

10 Sets and Linked Structures 293

10.1 Completing the Data Tree 293
10.2 Sets 294
10.3 Set Operations 297
10.4 Base Conversions 301
10.5 Application: Word Frequency Distribution 303
10.6 A New Variable Type: The Pointer 315
10.7 Constructing a Linked List 316
10.8 Modifying a Singly Linked List 322
10.9 Binary Trees 328
10.10 Application: Program Documentation 337

11 Introduction to Graphics 349

11.1 Turtlegraphics 349
11.2 The Graphics Screen 351
11.3 Color Options 354
11.4 Drawing Simple Geometric Figures 355
11.5 Using Recursion to Produce Figures 364
11.6 Application: Building an Interpreter 366

12 Graphics Applications 387

12.1 Graphics in Data Analysis 387
12.2 Application: Plotting a Time Series 389
12.3 Application: Producing X-Y Plots 392
12.4 Application: Producing Histograms 396
12.5 Application: Computer Mapping 401
12.6 Map Resources 412

Appendices

A The UCSD p-System 417

A.1 Exercises for UCSD Pascal 417
A.2 Hierarchy in the UCSD System 429
A.3 Entering a Program 430
A.4 Compiling a Program 434
A.5 Executing a Program 435
A.6 Changing a Program 437
A.7 Dealing with Errors 439
A.8 Saving a Program 443
A.9 Configuring Your System Diskettes 444

B Library of Programs 447

PROGRAM STATISTICS 447
PROGRAM MATRIX_OPERATIONS 450
PROGRAM TANDEM_SORT 453
PROGRAM PM_CORR 455
PROGRAM BINARY_TREE 458

C ASCII Codes and Their Meanings 463

Answers to Selected Exercises 467

Index 475

1 The Language and the Operating System

1.1 Text Structure and Scope

This text is an introduction to problem-solving using the Pascal language. Our primary goal 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 specific software to be used. We will consider versions of the language that run on the IBM Personal Computer and compatible machines.

Our attention will focus on Turbo* Pascal. Turbo Pascal was developed by Borland International. It is now in its third version, which includes extended graphics capabilities. As its name implies, Turbo is characterized by a rapid compilation time. This feature is particularly valuable while learning the language and during program development.

*Turbo Pascal is licensed by Borland International, Scotts Valley, California.

The second software system, the UCSD* p-System, will be discussed in Appendix A. The letters UCSD refer to the University of California at San Diego where this dialect of Pascal was developed. Pascal programs in the text will show any variations required to run in UCSD Pascal. We will refer to the two software systems generically as the Pascal system.

The Pascal system acts as an interface between the computer and you (Figure 1.1). While the user communicates 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 the housekeeping tasks associated with developing and manipulating programs.

When using Pascal, the housekeeping tasks require more explicit attention than when writing BASIC programs, for example. This fact may complicate early efforts to learn the language. In particular, the UCSD p-System has a rather complicated hierarchical structure. The Turbo Pascal system is considerably less complex than the UCSD p-System but to use it effectively requires some knowledge of the IBM PC disk operating system. We recognize this obstacle and attempt to minimize your frustration by providing step-by-step directions for performing the essential tasks.

You can obtain more details on the system you are using from the documentation provided with that system. In particular, the *Turbo Pascal Reference Manual* provides a description of that implementation of Pascal. Users of the p-System are referred to the *Beginner's Guide for the UCSD p-System,*** the *User's Guide for the UCSD p-System,**** and the *UCSD Pascal Reference for the UCSD p-System.*†

Our emphasis will be on writing well-structured programs in Pascal. No previous programming experience is assumed. If you have already programmed, you may need to give up some habits developed

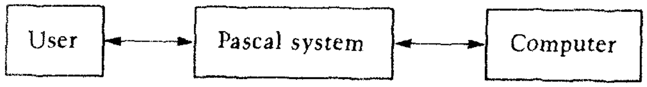


Figure 1.1
Relation of the Computer to You

*UCSD Pascal is a trademark of the Regents of the University of California.

**Beginner's Guide for the UCSD p-System (Boca Raton, FL: IBM Corp., 1982) and (Softech Microsystems, Inc., 1981).

***User's Guide for the UCSD p-System (Boca Raton, FL: IBM Corp., 1982) and (Softech Microsystems, Inc., 1981).

†UCSD Pascal Reference for the UCSD p-System (Boca Raton, FL: IBM Corp., 1982) and (Softech Microsystems, Inc., 1981).

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, it is assumed 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 of Pascal can be found in the *Pascal User Manual and Report.** In the report section of the publication, Wirth states that:

> The desire for a new language for the purpose of teaching programming is due to my dissatisfaction with the presently used languages whose features and constructs too often cannot be explained logically and convincingly and which too often defy systematic reasoning. Along with this dissatisfaction goes my conviction that the language in which the student is taught to express his ideas profoundly influences his habits of thought and invention, and that the disorder governing these languages directly imposes itself onto the programming style of the students.

Pascal is often referred to as a *highly-structured language.* The meaning of the term structured language is not easily explained at this point. For now, we will quote Yourdon, who has written

*K. Jensen and N. Wirth, Pascal User Manual and Report (New York: Springer-Verlag, 1978), p. 133.

that a structured language lends itself to "writing programs according to a set of rigid rules in order to decrease testing problems, increase productivity, and increase the readability of the resulting programs."* We will say more about structured programming in later chapters.

The names of many computer languages are acronyms. For example, BASIC means Beginner's All-purpose Symbolic Instruction Code, and FORTRAN stands for FORmula TRANslation. In contrast, Pascal is named for Blaise Pascal, a seventeenth-century French mathematician, philosopher, and inventor of an adding machine that performed calculations with wheels and cogs.

1.3 The Pascal System

It is best to think of the Pascal system as the environment in which Pascal programs are written. Keep in mind that the purpose of the Pascal system is to assist in the performance of the housekeeping tasks described in Section 1.1. Knowing the Pascal system allows you to develop and process programs more rapidly, but it does not improve the structure or the content of those programs.

In this section, the following operations performed by the Pascal system are discussed:

1. Editing
2. Compiling
3. Executing
4. Filing

The system performs other operations, but for now our attention is restricted to these four. A brief description of each operation follows.

Editing usually conveys the notion of correcting or modifying an existing document. In the Pascal system, editing refers to the process of creating, as well as modifying, documents. The result of the editing is the creation or the modification of a text file. The text file is a representation of a document (program or data) in a form that humans can read.

Compiling refers to the process of converting a text file consisting of a Pascal program into a form readable by the machine

*Edward Yourdon, Techniques of Program Structure and Design (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1975), p. 144.

(or more nearly readable in the case of UCSD Pascal). To execute a program that exists only in text form, the compiling must occur first. During the process of compilation, errors in the text version of the program that will prevent its execution are detected. Such errors are identified to the programmer, who must correct them (using the Editor) before a machine-readable version of the program can be produced. Thus, two versions of a program exist—one that the programmer produces (a text file) and one that the compiler produces (a code file).

Executing refers to causing the computer to perform the instructions in a program. As noted in the previous paragraph, a program must be in code form before it can be executed. During the execution of a program, each instruction in the code version of the program is executed by the central processing unit.

Filing refers to a set of processes which may be considered as record-keeping. These processes include saving programs, retrieving programs, inquiring about disk contents, and so forth. Some filing processes can destroy programs or data sets, and they must be used carefully.

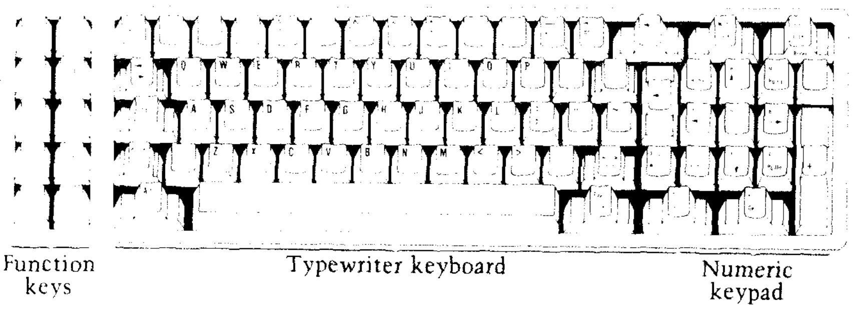
1.4 Using the Keyboard

The keyboard is the primary mechanism for communicating with the computer. Therefore, it is important to use it effectively. In particular, certain special keys will be used frequently to interact with the Pascal system. First note that there are two shades of keys. The lighter keys in the center are those normally found on a typewriter. The lighter keys at the right are those found on an adding machine or electric typewriter keypad. The darker keys are used to invoke special functions relating to the operation of the computer. Somc of these darker keys, such as the shift keys, backspace key, caps lock key, and so forth, also perform normal typewriter functions. Some keyboards may not have the shading described here. Some of the special keys are marked as follows:

Ctrl	control key
↵	enter key
Esc	escape key
Alt	alternate key
Del	delete key
←	cursor left
↑	cursor up
↓	cursor down
→	cursor right

Figure 1.2 is a diagram of the IBM PC keyboard. Locate the special keys on this diagram.

Figure 1.2
IBM PC Keyboard



1.5 Using Turbo Pascal: Guided Exercises

In this section, we lead you through a set of exercises designed to acquaint you with Turbo Pascal using an IBM PC. These exercises include:

1. Starting Turbo
2. Examining the Turbo files
3. Running a Pascal program
4. Backing up the Turbo diskette
5. Creating private diskettes

Starting Turbo

In describing the start-up of Turbo Pascal it is assumed that you have the original diskette furnished with the software or a backup copy of that diskette. We also assume that you know how to boot the **disk operating system** (DOS) using your DOS diskette. Thus, you may start with the screen showing the following display:

```
Current date is Tue  1-01-1980
Enter new date: 9-15-85
Current time is  0:00:31.91
Enter new time: 14:00


The IBM Personal Computer DOS
Version 2.10  (C)Copyright IBM Corp 1981, 1982, 1983

A>
```