# Programming with Macintosh® Pascal

Richard A. [illegible]

Vance B. Wisenbaker

Richard G. Vance

# Programming with Macintosh® Pascal

Richard A. Rink

Associate Professor of Computer Science
Eastern Kentucky University

Vance B. Wiseman

Dean of the College of Social and Behavioral Science
Eastern Kentucky University

Richard G. Vance

Professor of Political Science
Eastern Kentucky University



PRENTICE HALL, Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Rink, Richard A.,

Programming with Macintosh Pascal / Richard A. Rink, Vance B. Wisenbaker, Richard G. Vance

Includes index.

ISBN 0-13-730540-0

1. Macintosh (Computer)--Programming. 2. Pascal (Computer program language) I. Wisenbaker, Vance B. II. Vance, Richard G. III. Title.

QA76.8.M3R56 1989

005.265—dc19 88-22404

Editorial/production supervision and interior design: John Fleming

Cover design: Photo Plus Art

Manufacturing buyer: Mary Noonan





Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-730540-0

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

# Preface

The primary objective of this book is to provide a solid textbook on the Pascal language for users of the Macintosh® computer. The authors believe that Macintosh Pascal provides a stimulating environment for learning programming because of the combination of power, unique graphics qualities, and an interpreter that provides easy access to the Macintosh's capabilities.

This book has been written in such a way that it will also serve as a useful reference work and as a self-study guide. As a textbook, it is directed toward the beginning courses on Pascal for computer science programs as well as toward special-interest courses on Pascal for Macintosh users. This level is elementary in the first six chapters but shifts to an intermediate level in chapters 7 through 12. This book includes all the usual topics of a beginning textbook on the Pascal language, plus some added topics on Macintosh Pascal that are special to the Macintosh computer. Topics included are pointers, files, and the manipulation of strings; procedures supported by both the QuickDraw and SANE libraries.

To use this book as a self-study guide, we suggest the following steps: First, read the chapter objectives and review questions before beginning to read the first section of any chapter. Second when you have completed reading a section within a chapter, return to the review questions and see if you can answer any questions. Third, enter the examples in each section that are full programs and see if they will execute. If you have any syntax or execution errors, check the program listing and make corrections. If you are successful at executing the example, try to modify the example to perform one or more other actions, again testing your program to see if it will execute. After completing a

chapter, again read the review questions. Once you feel that you have answered those questions correctly, choose several programming exercises from the chapter so that you can try to enhance your abilities at writing Macintosh Pascal programs. Note that some of the programs lised in this book may appear different from a listing in the program window of the Macintosh computer. This was done so that the program can be read with more ease.

The Macintosh is a true graphics machine. A person using the Macintosh Pascal language can write computer programs in a high-level language that is able to interact with the ROM-based graphics capabilities of the Macintosh computer. For example, a student in computer science using the Macintosh computer and Macintosh Pascal has a tool for understanding some of the basic graphics routines required of a workstation. By applying the QuickDraw Library routines, several windows can be overlaid on the screen, only one of them being active at any time. In any active window, lines, curves, regions, pictures, polygons, or text can be drawn, with control limited to actions of the mouse.

Each chapter introduces the basic principles of Pascal by defining the syntax, and the presentation of complete Macintosh Pascal programs shows the semantic actions. Having full Pascal programs provides the opportunity to enter the programs into a Macintosh computer and see the results of execution. You are not only able to verify the discussion in each chapter but you can alter the example to try your own ideas. This reinforces learning of the material and also provides an opportunity to build self-confidence. As a program is entered, errors can be immediately corrected and the results observed. All chapters include programming exercises in addition to the complete examples, providing practice with the material presented. Altogether there are more than 150 programming exercises, which range in difficulty from simple to challenging. Stepwise refinement is applied in the development of all structured programming concepts, the later chapters employing structure charts to enable effective top-down programming design.

The authors have chosen to write this book using a version of Pascal known as Macintosh Pascal (Version 2.0) published by THINK Technologies, Inc., and Apple Computer, Inc. Macintosh Pascal is an interpreted version of Pascal, and because of its use of windows and other features, it is an excellent teaching language. Although several Pascal compilers are now available, thay are primarily intended as development systems for writing sophisticated software systems. These compilers also assume that the user is an experienced Pascal programmer with a complete understanding of the Macintosh system. Macintosh Pascal allows you to build the self-confidence required of a programmer before tackling the complete Macintosh system.

We wish to acknowledge the help of several people who have contributed to this project. First we wish to thank Dr. Marijo LeVan and Dr. Jerry LeVan for contributing ideas for examples as well as comments. We wish to thank Phyllis Gabbard, Suzanne Tipton, Kellie Lynn, Lisa Raines, and Pauline Colemen for their help in the preparation of the manuscript. We would also like to thank Herman Gollwitzer of Drexel University, Barry S. Marx of Wake Technical College, Denise Kiser of the University of California at Berkeley, Henry Etlinger of Rochester Institute of Technology, and Christine Kay of

DeVry Institute of Technology for the helpful suggestions they made in the preparation of this book and John Fleming, our production editor with Prentice Hall. Macintosh is a trademark licensed to Apple Computer, Inc.; Macintosh Pascal is a product of SYMANTEC Corporation (THINK Technologies) and Apple Computer; WORD, Excel, and Multiplan are products of Microsoft, Inc.; Jazz is a product of Lotus Development Corporation; TML Pascal is a product of TML Systems; and Turbo Pascal is a registered trademark of Borland International, Inc.

# Contents

# chapter 1

# introduction

## OBJECTIVES

**After completing Chapter 1, you will know the following:**

1. What is meant by the concepts "computer" and "computer program."
2. That computer programming is a form of problem solving that involves a set of steps, as discussed and illustrated.
3. That a computer program is written by first developing an algorithm. The nature of algorithms is discussed and illustrated.
4. The menus of Macintosh Pascal, with each option of each menu.

## 1.1 COMPUTERS AND COMPUTER PROGRAMS

A computer is an automated machine that can process information and, in processing this information, solve one or more problems. What type of information can a computer process? This usually depends on the type of problem to be solved and the individuals making use of the computer. For example, an engineer may use a computer to solve a mathematical problem, in which case the solution is a mathematical model of an engineering process. A business executive may use a computer to provide information on sales, gross profits, costs, and projected future profits and sales, with the computer

providing information in both tabular and graphic formats. A student may use a computer to link with a database located thousands of miles from where he or she is sitting and submit questions to retrieve factual and deduced information from some particular set of information. In all these examples, the information to be processed by the computer may be numeric data (numbers) or symbolic data (characters) representing processes taking place at a particular instant of time.

When the computer is being used to solve a problem, the format or the steps necessary for solving this problem have previously been entered into the computer by means of a computer program. The word processor, used for composing these paragraphs and printing each character, word, line, and paragraph of text, is a computer program. A computer program is also information to the computer, but information of a special form. We will define a computer program as an ordered set of instructions written in some type of *artificial* language (such as Macintosh Pascal). The instructions composing the computer program represent the solution to a problem. Computer languages are often referred to as artificial languages because they are more restrictive in the application of syntax (grammar) and semantic (meanings) rules, attempting to avoid the ambiguity found in natural languages. Another reason for the term *artificial* is the origin of these languages in a laboratory environment. While at present Macintosh Pascal may appear to be an extensive language, you will soon see that it is much simpler to understand than any of the natural languages such as English, German, French, or Russian. It is a highly structured language with specific syntax rules and semantic definitions for each of its commands. It is best if you begin by learning some of the easier syntax rules for composing data objects (nouns to Pascal), commands that provide action (sometimes referred to as verbs), and the rules necessary to form acceptable sentences that provide semantic meaning to the computer program.

A computer language provides a means by which an individual can communicate the solution of a problem to a computer. Why is this so important? First, computers like the Macintosh execute at a very primitive level, a machine level where all commands and information are binary. This means that they are composed of sequences of one and zero bits. Writing programs at machine level can be done, but only if the person writing such programs has a thorough knowledge of the machine being used, has extensive programming experience, and pays considerable attention to the storage of information at specific locations (addresses) in the memory of the computer. An understanding of how basic operations are to be performed is also needed. Using a high-level language such as Pascal frees the programmer from such concerns and allows concentration on defining the steps for solving the problem. The binary machine instructions are generated when the Pascal program is executed by the computer, using a special program called a translator.

## 1.2 BASIC COMPUTER ORGANIZATION

When describing a computer, it is useful to consider its basic organization from the viewpoint of input, output, memory, and execution. The Macintosh computer can be viewed as a von Neumann machine, named for the mathematician John von Neumann, who is credited with the stored program concept of computing. As shown in Figure 1.1—it has five basic units: the central processor, memory, input, output, and bus.

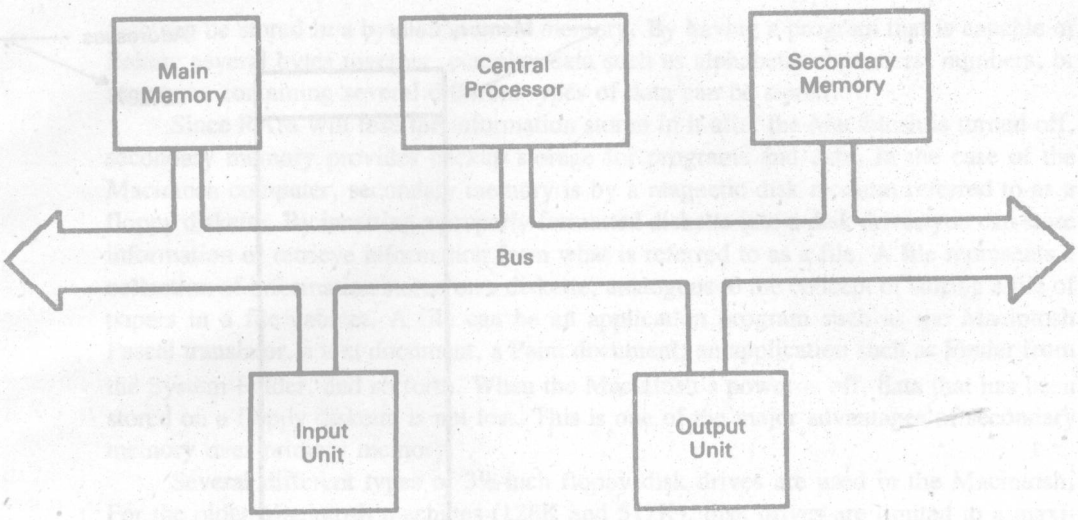


**Figure 1.1** Basic Organization of a Computer.

For the von Neumann machine the function of the central processing unit (CPU) is to fetch an instruction of a computer program from main memory, decode this instruction, and then execute it; this same process is then repeated. During the execution of an instruction, the CPU is capable of performing basic arithmetic operations such as addition, subtraction, multiplication, and division; basic logical operations such as a comparison of values; and the merging and masking of data. The CPU is also capable of controlling the actions of other units such as memory, input, and output. For the Macintosh, the central processor is the MC68000 microprocessor.

Memory is divided into two distinct classes: primary or main memory and secondary memory for storage. The purpose of main memory is to store programs necessary for performing one or more specific functions as well as the data needed at the time that a program is executed. For example, the word processing program used for writing this paragraph, Microsoft Word, is contained in main memory along with the text as it is entered from the keyboard. Having both the program and data in main memory provides for fast response when entering additional words or making corrections to text previously entered. In the Macintosh computer, main memory is divided into two basic structures, random-access memory (RAM) and read-only memory (ROM). RAM offers flexibility since it can both be written to and read from as long as the Macintosh remains powered. Once the Macintosh is turned off, any information stored in RAM is lost. ROM contains information needed by the Macintosh to "boot" or start up when the power is turned on and also provides numerous instructions for interfacing with RAM memory, disk drives, the printer, windows, and menus. ROM is different from RAM in that it can be read from but not written to by the program being executed. ROM is also permanent; its instruction will not vanish when the Macintosh is turned off.

Main memory can be viewed as a large one-dimensional table of cells, each cell storing one byte of information. As Figure 1.2 shows, each cell has a unique address represented by a whole number. The byte is composed of eight individual bits, single binary digits, with each bit location containing either a zero or a one. Whereas humans

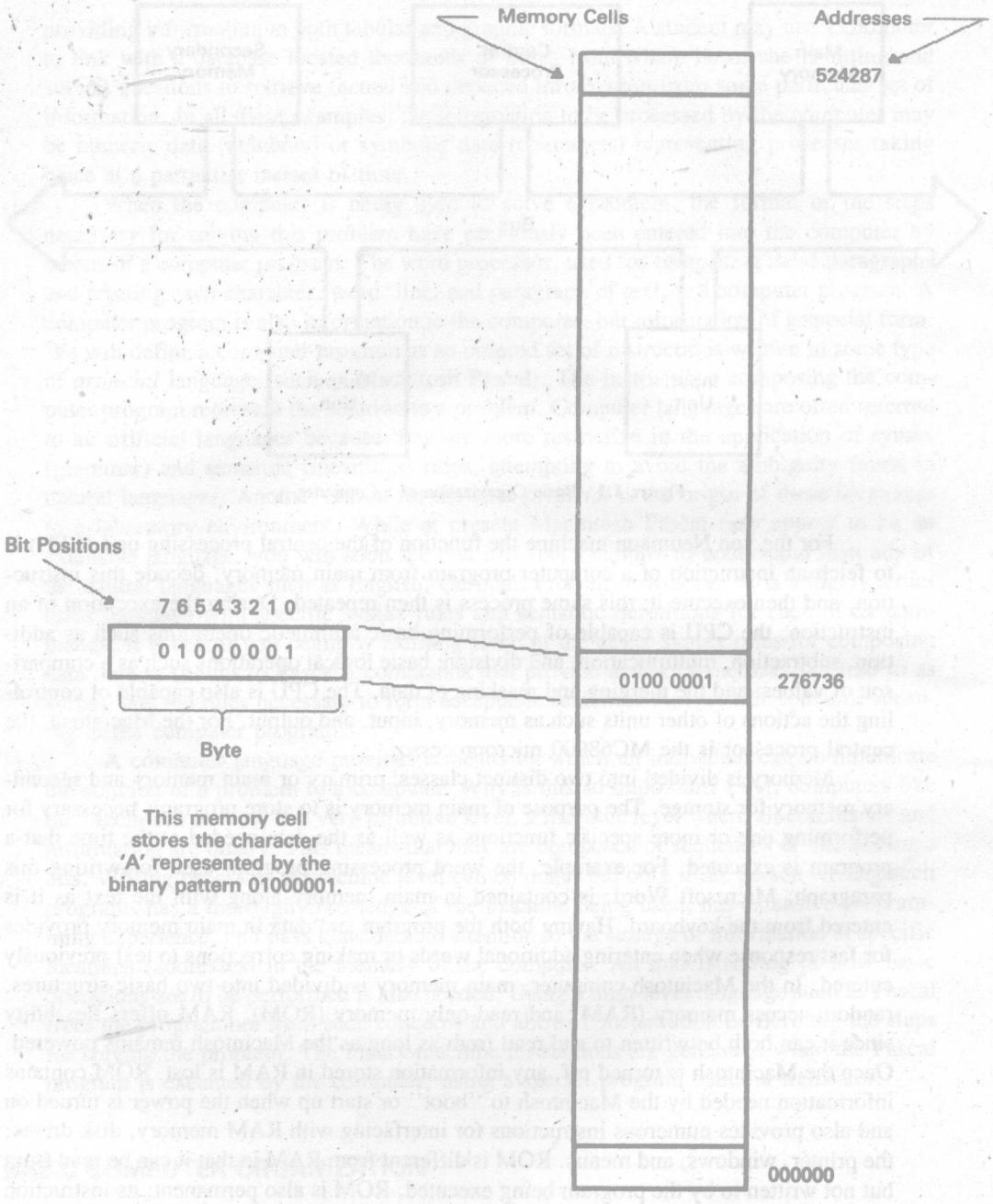


**Figure 1.2** Representation of main memory for a 512K Macintosh computer.

are accustomed to communicating in natural languages, computers deal with a more fundamental level of information where all characters are represented by these one and zero bits. In this context, an individual character shown on the keyboard of the Macin-

tosh can be stored in a byte location of memory. By having a program that is capable of linking several bytes together, complex data such as alphabetic characters, numbers, or structures containing several different types of data can be stored.

Since RAM will lose the information stored in it after the Macintosh is turned off, secondary memory provides backup storage for programs and data. In the case of the Macintosh computer, secondary memory is by a magnetic disk medium referred to as a floppy diskette. By inserting a properly formatted diskette into a disk drive, you can store information or retrieve information from what is referred to as a file. A file represents a collection of information stored on a diskette, analogous to the concept of storing a file of papers in a file cabinet. A file can be an application program such as the Macintosh Pascal translator, a text document, a Paint document, an application such as Finder from the System Folder, and so forth. When the Macintosh's power is off, data that has been stored on a floppy diskette is not lost. This is one of the major advantages of secondary memory over primary memory.

Several different types of 3½-inch floppy disk drives are used in the Macintosh. For the older Macintosh machines (128K and 512K), disk drives are limited to a maximum storage of 400K (409,600) bytes. The newer Macintosh machines (Macintosh Plus and Macintosh SE) use floppy disk drives with 800K bytes of storage capacity. Hard disk drives, with the disks permanently encased, can provide 10 megabytes (the prefix *mega-* means "million"), 20 megabytes, 40 megabytes, or 80 megabytes of disk space. Hard disk drives provide faster access to the contents of files than the 3½-inch floppy disk drives.

An input unit provides the user with a means to communicate with the computer. For the Macintosh, both the keyboard and the mouse are input devices. The keyboard allows entering information such as text in a word processing document, while by clicking the mouse button we can enter information such as our choice of options from a menu bar.

The output unit provides a means for the computer to communicate with the user. In the case of the Macintosh, the screen is an output device. Another output device is a printer such as the Apple ImageWriter. A disk drive can also serve as both an input and an output device.

The bus provides an electrical path for connecting the various units. Both commands and information can be passed along this path. In the Macintosh there are two types of bus structures, internal and external. The internal bus links the central processor with main memory, while the external bus provides a link between main memory and secondary memory as well as other input and output devices.

## 1.3 PROBLEM SOLVING

Programming can be described by the following steps: (1) having a problem that requires a solution, (2) finding the solution to the problem, (3) specifying the ordered set of steps that represent the solution, and (4) implementing these steps in a computer language. In reaching a solution to a problem, we must be concerned with analyzing the problem, finding the steps for a solution, and then formally defining the set of steps. Using a computer language such as Macintosh Pascal serves as a tool for implementing our