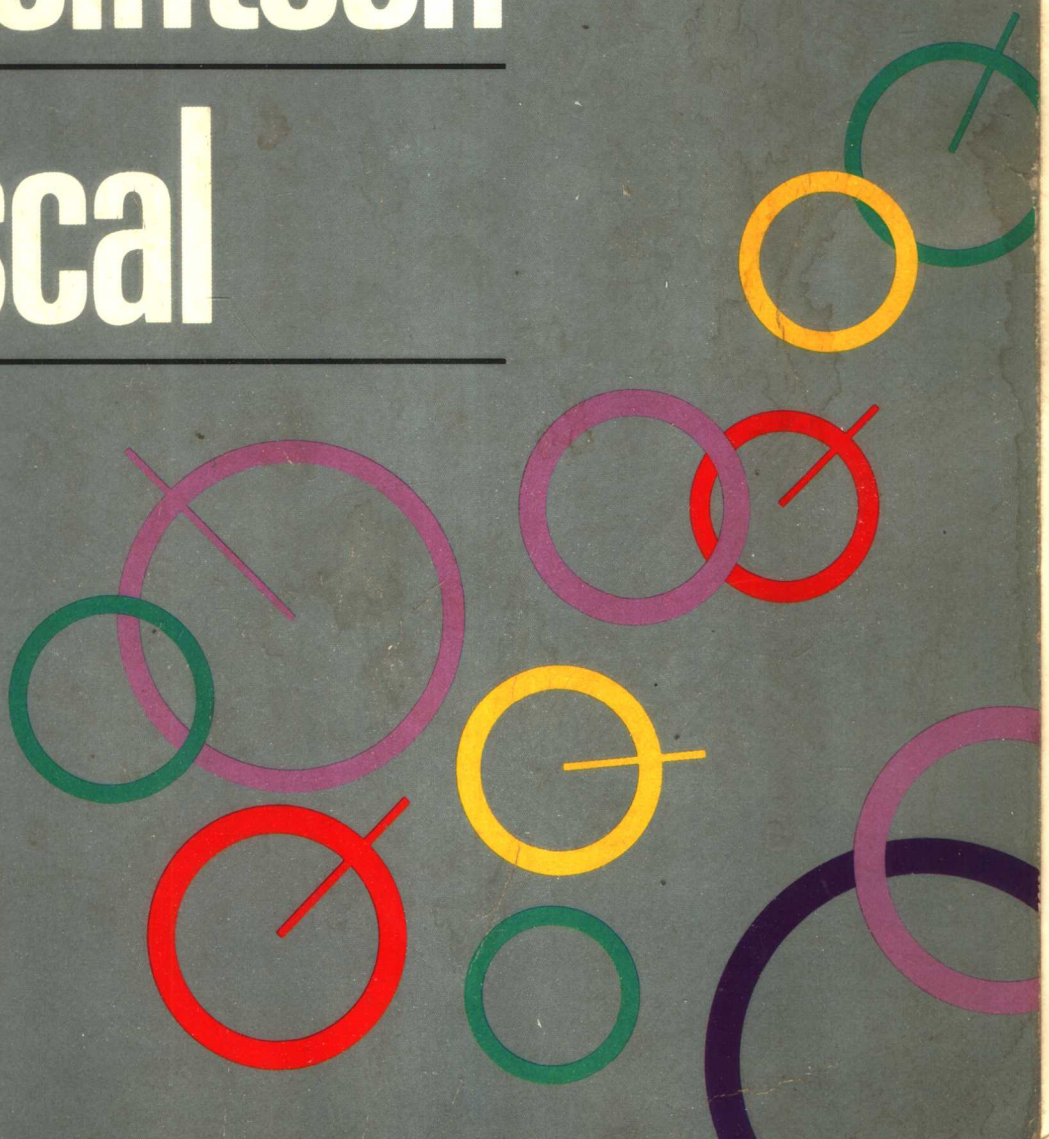
OsborneMcGraw-Hill

THE FIRST BOOK OF Macintosh™ Pascal



PAUL A. SAND

THE FIRST BOOK OF MACINTOSH™ PASCAL

Paul A. Sand

Osborne McGraw-Hill
Berkeley, California

Published by
Osborne **McGraw-Hill**
2600 Tenth Street
Berkeley, California 94710
U.S.A.

For information on translations and book
distributors outside of the U.S.A., please write to
Osborne **McGraw-Hill** at the above address.

THE FIRST BOOK OF MACINTOSH™ PASCAL

234567890 DODO 898765

ISBN 0-07-881165-1

Cindy Hudson, Acquisitions Editor
Paul Hoffman, Technical Editor
Catherine Pearsall, Copy Editor
Cheryl Creager, Composition
Yashi Okita, Cover Design

INTRODUCTION

This book is an introduction to the Pascal programming language and to the version of that language on the Apple Macintosh computer. You can use this book regardless of your level of expertise in computer programming. If you are an absolute beginner, don't worry; it is our assumption that you know nothing about either programming in general or Pascal in particular. We will explain everything you need to know in order to write, enter, and run programs in Pascal on the Macintosh.

Even if you already have some familiarity with the Pascal programming language, you will find this book useful. Macintosh Pascal is like no other version of Pascal you have ever used—just as the Macintosh itself is like no other computer you have used. We will attempt to explain and demonstrate many of the major features of Macintosh Pascal, especially those that aren't found in "Standard" Pascal.

Book Philosophy

Computer programming is a skill that is learned by doing. No book yet written can take the place of the actual experience you will gain in designing and writing your own programs. Realizing this, we will not promise that simply reading this book will make you a seasoned expert in Macintosh Pascal; our goals are more modest. This book will

- Provide you with a description of Macintosh Pascal's features and how these features are used in programs.

- Give you the *rules* of the language: what is legal, what isn't legal.
- Show you many examples of Macintosh Pascal programs that demonstrate how the language can be used to access many of the hardware and software features of the Macintosh itself.
- Provide you with some suggestions for changing the programs presented here as well as ideas for your own programs. Ideally, this should get you started in learning how to program on your own.

In general, the emphasis in this book will be on the final two points: to show you how Macintosh Pascal works and to spark your own curiosity enough to discover more on your own. We will lean heavily on example programs throughout the text to accomplish this.

Somewhat less emphasis will be placed on the more "formal" aspects of Pascal: the precise rules that make one Pascal statement legal and another one unacceptable. Many of the formal rules of Pascal are designed to make sense of very unusual program constructions, ones unlikely to occur in everyday programming. We will forego explaining such rules in all their mind-numbing detail. Instead, we will concentrate on developing your intuitive sense for what's right and what's wrong in Pascal.

As a user of Macintosh Pascal, you own or have access to the documentation provided with the Macintosh Pascal software. You should look upon this book as a *supplement* to the Macintosh Pascal documentation, not as a replacement. The Macintosh Pascal reference manuals are complete, concise, and rigorous. This approach has the advantage that you may look there for the exact rules on what you may or may not do in your Pascal programs. The disadvantage is that it is difficult for programming novices and other programmers not familiar with Macintosh Pascal to sort through the descriptions of every last nit-picking rule and restriction to extract the truly useful information present in the reference manuals: the knowledge you will need *every* time you write a Pascal program.

In this text, we provide a more leisurely and less formal introduction to Macintosh Pascal. We concentrate on the "good parts" of Macintosh Pascal: those that can be used easily by both programmers unfamiliar with either Pascal and

programmers who know Pascal but may not be acquainted with the Macintosh's version of the language.

Why Learn to Program?

As you probably know, everything the Macintosh (or any other computer) does is controlled by its software: programs that tell the computer what to do in various situations and how to accomplish useful tasks. Learning to program is, simply stated, discovering how to tell the computer to "do things."

For many people, computer use is restricted to using programs someone else has written. Given the increasing sophistication and usefulness of commercially available computer software, this is a perfectly workable strategy for many, if not most, computer users. You have an inalienable right *not* to learn to program your computer, if you so choose.

Having said that, however, here are some reasons why you might, after all, want to undertake the task of writing your own programs:

- Learning to program will give you a better idea of how the computer works and what it can and cannot do.
- Programming is, like mathematics, an intellectual discipline that is inherently worth knowing.
- Knowing how to program can stand you in good stead when a computer problem arises at work or at home for which no available prewritten software applies. Depending on your expertise, you may be able to write a program to solve the problem, often with less time and expense than if you had sought commercially available software for the same purpose.
- Programming can be profitable. People who don't know how (or don't have time) to program will pay you money to make their computer jump through designated hoops.
- Last, but not least, programming is fun—a truly endless source of amusement. If you enjoy intellectual challenge, thinking a problem through to its solution, and the sense of accomplishment that occurs when something works as it should, you will find programming to be immensely enjoyable. In a sense, programming is the ultimate computer game.

Why Learn Pascal?

If you decide to learn to program, you need to learn at least one programming language. In this book, obviously, we are suggesting you learn the programming language called *Pascal.* Pascal was developed in the late 1960s by Niklaus Wirth. His aims were to produce a language suitable for teaching programming concepts clearly and systematically and to make the language usable on a large number of computers. His success is obvious: Pascal's popularity has increased rapidly since its introduction, and versions of the language are available on many computers, ranging from small, inexpensive personal computers to large mainframe systems.

Why is Pascal so popular? The primary reason is that Pascal makes the job of writing, reading, and modifying computer programs easier than do many other programming languages.

Here are some ways in which Pascal is a convenient language for programmers. (You need not worry if you don't understand all or any items on this list, by the way. We'll be seeing how all these things work later.)

- Pascal has "structured" control statements (**while**, **repeat**, **for**, **case**, and **if-then-else**) that allow the programmer to write clear and concise code with the flow of control proceeding from top to bottom. The control flow in programs written in languages lacking these structured control statements often contains complex webs of **if** tests and **goto** statements. Such programs are often called "spaghetti code" because they are so difficult for even experienced programmers to untangle.
- Pascal permits the programmer to break up a large program into smaller, relatively independent procedures and functions, each one of which performs a single, easily understood task. Each procedure or function can have its own set of "private" variables that are only used when that procedure or function is executed. Each procedure or function has well-defined input and output parameters used for communicating with its calling routine. This decomposition of a large program into modules greatly aids the programmer in both the initial design of the program and also in any subsequent modifications to the program; the modules can also be reused

in subsequent programs, decreasing the overall programming effort.

- Pascal allows programmers to define their own data types and data structures in addition to those already built into the language. Judicious use of this feature can make a program more compact and easy to understand.
- Pascal allows the use of long identifiers for variables, procedures, and functions. This allows the programmer to use names with mnemonic significance, another aid in understanding a program.

Although Pascal is a good computer language, it is not a perfect one. And it isn't suitable for all applications. Rather than go into Pascal's deficiencies here, however, we'll merely note that a good deal of serious software development is carried out in Pascal. And even if Pascal isn't your programming language of choice, you will find that learning Pascal will make learning and using other programming languages much easier.

Why Learn Macintosh Pascal?

As we will see in the first chapter, the "programming environment" provided by the Macintosh Pascal software is extraordinarily easy to use. In order to write Pascal programs on a typical computer system, you have to learn how to use a number of different programs, each with its own set of hard-to-remember commands and rules. This, fortunately, isn't the case with Macintosh Pascal: there are no complex command sequences to memorize, nor are there separate "editor," "compiler," or "debugger" programs to master. You need only learn to use the Macintosh Pascal software in order to be able to enter, run, and modify your own Pascal programs.

Macintosh Pascal uses most of the conventional elements of the Macintosh user interface: pull-down menus, multiple overlapping windows on the screen, mouse selection, and so on. In fact, if you know how to use a word processing program like MacWrite or Microsoft Word, you already know nearly everything you need in order to enter your Pascal programs into the computer.

Macintosh Pascal is therefore an excellent way to learn

the Pascal language. But that's not all: built into Macintosh Pascal is the capability for you to control nearly all aspects of your computer's operations. These capabilities include some not offered with languages costing many times more than Macintosh Pascal. (As before, you shouldn't worry if you don't understand everything—or anything—on this list as yet.)

- Macintosh Pascal provides a variety of numeric data types that make it possible to write precise and reliable computational programs.
- Macintosh Pascal supplies a number of extensions to the Standard Pascal language that make writing common application programs easier: there is a built-in **string** data type, an **otherwise** clause on the **case** statement, sophisticated memory management, and direct-access file I/O routines, to name a few. (These final two topics are relatively advanced, however, and won't be considered in this text.)
- Macintosh Pascal allows programs to access most of the capabilities of the Macintosh; for example, built-in routines callable from Pascal programs allow your programs to manipulate windows, use the mouse, the clock, and the sound generator.
- Probably the most interesting feature is Macintosh Pascal's ability to call the QuickDraw routines and other software contained in the Macintosh's read-only memory. QuickDraw is an extensive "library" of routines that allows your programs to perform awesome feats of graphics magic.

Macintosh Pascal, while an excellent learning environment, is not suited to writing large application programs. In computer jargon, Macintosh Pascal is an interpreted language, rather than a compiled one. Compared to programs written in other languages, you may observe that your Macintosh Pascal programs are rather slow. (They may very well be fast enough for your purposes, however.) Also, your Pascal programs can't run "by themselves"; the Macintosh Pascal system must be present in the computer's memory at the same time. This imposes a relatively restrictive upper limit on the size of your Pascal programs.

On balance, however, Macintosh Pascal is an excellent learning and programming tool for all but the most demanding applications.

Hardware Requirements

Macintosh Pascal will run on any Apple Macintosh computer; it will also run on an Apple Macintosh XL (Lisa) computer system set up with the MacWorks Macintosh-emulation software. You may run Macintosh Pascal on a bare-minimum Macintosh system with 128K bytes of memory and the single built-in disk drive.

As with most software for the Macintosh, additional hardware will make working with Macintosh Pascal easier. A second disk drive, while not required, will greatly decrease the time you spend on the drudgery of removing and inserting disks. If you add a printer to the system, you'll be able to get listings of your Pascal programs on paper. Expansion to 512K bytes of memory will permit you to write and use much larger programs under Macintosh Pascal than is possible with a 128K system. (Memory expansion will also allow your other Macintosh software to handle large amounts of data more easily.) All programs in this book will fit easily in a 128K Macintosh, however.

How to Read the Rest of the Book

As previously indicated, computer programming is one of those subjects that can't be learned by simply reading a book (even a *good* book). So in order to get the most out of this book, you should read it in front of your computer. Try the example programs in each chapter as you read. Don't be afraid to experiment with the example programs; we'll even suggest some possible experiments to you as we go along.

Even more important to learning a computer language is the ability to take the descriptions and examples of the language elements given here and apply them when the time comes to write programs of your own. This, too, is an ability that only comes with practice. In addition to the experiments you can try out on our example programs, we'll suggest some problems you can try to solve by writing your own programs "from scratch."

If you are a novice to programming, simply start at the beginning. You will find that the material in most chapters depends heavily on information from previous chapters, so

skipping ahead to a seemingly more interesting chapter is generally a bad idea.

If you already know some other version of the Pascal language, you will probably not want to plod through the chapters of this book that contain material you already know. Macintosh Pascal *is* Pascal, after all, and if you know Pascal, you already know an appreciable fraction of the material covered in this book. You might find it more interesting to take a fast track through "Macintosh Pascal-only" parts of the book, skimming or skipping sections that cover material you already know.

We will make it easy for you to do just that. Material in this book that applies only to Macintosh Pascal will usually be confined to individual sections and chapters, not scattered throughout the text. For easy identification, these sections and chapters will all have the word "Macintosh" in their titles.

Chapter One will introduce you to Macintosh Pascal, taking you through a step-by-step example of writing and running a simple program. All readers should probably work through this chapter.

Chapter Two introduces the fundamental concepts of Pascal: constants, variables, integer, real, and Boolean types, assignment statements, expressions, operator precedence, simple input and output operations, and looping control structures. This chapter contains very little Macintosh-specific material.

Chapter Three discusses advanced editing techniques you'll use in Macintosh Pascal: selecting, cutting, pasting, and copying blocks of text, finding and replacing pieces of your program, and general disk housekeeping. Much of this material will be familiar to those acquainted with other Macintosh software, but some Macintosh Pascal-only rules are discussed.

Chapter Four discusses Pascal's decision control structures: **if-then-else**, **case**, and **goto**; it contains very little Macintosh-only material.

Chapter Five covers Macintosh Pascal's marvelous debugging aids: the Observe and Instant windows, program breakpoints, and step-by-step program execution.

Chapter Six explores six additional Macintosh Pascal data types: characters, strings, long integers, and three additional kinds of real numbers: computational, double, and extended.

Of these six data types, only characters are present in Standard Pascal.

Chapter Seven is an introduction to built-in or *library* functions available for use in your Pascal programs. This includes both standard functions built into nearly every version of Pascal, and Macintosh Pascal functions to accomplish tasks such as string manipulation, binary arithmetic, and other special Macintosh operations.

Chapter Eight considers built-in library procedures in Macintosh Pascal, including the first description of QuickDraw routines.

Chapter Nine discusses how to go about programming your own procedures and functions. This is mostly Standard Pascal material.

Chapter Ten introduces the concept of defining your own data types and illustrates the principle using enumerated and subrange types.

Chapter Eleven covers Pascal's "structured" types: arrays, sets, and records, including variant records.

Chapter Twelve discusses many of Macintosh Pascal's predefined types and how they are used to access additional QuickDraw capabilities, as well as other Macintosh features.

Not covered in this text are "advanced" Pascal topics such as recursion, file I/O, pointers, and handles. Our coverage of QuickDraw, while sufficient to allow you to write useful programs that generate sophisticated graphics, doesn't cover many powerful aspects of QuickDraw that are less easy to use. Advanced sound generation using the four-voice synthesizer, event management, direct "in-line" calls to the Macintosh's ROM Toolbox routines, and use of the Standard Apple Numeric Environment (SANE) library are also not discussed in this book.

Recommended Reading

The original definition of the Pascal language was described in *Pascal User Manual and Report* by Kathleen Jensen and Niklaus Wirth (Springer-Verlag, 1978). This book contains a concise but well-written description of the legalities of the Pascal language. It remains a good introduction to Pascal for those who are familiar with another computer language.

Recently both the American National Standards Institute (ANSI) and the International Standards Organization (ISO) have approved Pascal standards that clear up the minor ambiguities contained in the original Jensen and Wirth Pascal. The "Level 0" ISO Standard is equivalent to the ANSI standard. A good, readable description is found in *Standard Pascal User Reference Manual* by Doug Cooper (W.W. Norton, 1983).

In nearly all cases there is no difference between ANSI/ISO Pascal and Jensen and Wirth Pascal. When we use the term "Standard Pascal," we will be referring to either one. In cases where there is a difference, we will explicitly say which interpretation we are using.

CONTENTS

GETTING STARTED WITH MACINTOSH PASCAL 1

The first program to write is the same in all languages...

—*B. Kernighan and D. Ritchie*
The C Programming Language
(Prentice-Hall, 1978)

Our goal in this chapter is to acquaint you with most of the basic operations you'll be performing every time you program in Macintosh Pascal. You will learn how to enter programs into the computer and how to correct the inevitable mistakes you will make in the process. The chapter discusses how to run your program once you have typed it in and how to make further modifications to your program. You'll find out how to save programs on disk and how to print them on your printer.

A REVIEW OF MACINTOSH FUNDAMENTALS

This book will assume that you have at least some experience in operating the Macintosh. Since Macintosh Pascal and most Macintosh applications use the same basic operations, if you have used any other Macintosh program, you already know

just about everything you need to know to use Macintosh Pascal.

If you are a complete novice to the Macintosh, we suggest that you take time to explore the features of the Macintosh, either by using the excellent "Guided Tour" casette tapes provided with your computer or by following the more traditional method of reading the Macintosh user manual.

In order to use Macintosh Pascal, you should be acquainted with the following topics and terms:

- *Moving the mouse.* By moving the mouse around your desktop, you move the *pointer* around the Macintosh screen. Usually the pointer is an arrow pointing north by northwest, but its shape changes depending on where it is pointing and what the Macintosh is doing at the time. (When necessary, the text will refer to the pointer's shape, for example the I-beam pointer.)
- *Clicking.* Many operations involve moving the pointer to a certain object and then pressing and quickly releasing the mouse button. In Macintosh jargon, this is called *clicking* the object. For example, to "click the Pascal disk icon" means to move the pointer to the picture of the Pascal disk on the screen and then press and quickly release the mouse button. Like most operations on the Macintosh, this is far easier done than said.
- *Double-clicking.* This operation only differs from clicking in that you press and release the mouse button twice in quick succession after positioning the pointer.
- *Pressing.* To *press* something means to move the pointer to it and then hold down the mouse button without moving the mouse.
- *Dragging.* To *drag*, you position the pointer on an object; press the mouse button and, holding it down, move the mouse to another position; then release the button. The object to which you originally pointed will move to the new pointer location.
- *Menu selection.* A *menu* (a list of possible command choices) is displayed when you press one of the words or phrases in the *menu bar* at the top of the screen. To choose one of the commands, drag to the command you want and release the mouse button. For example, if you were instructed to "choose Go from the Run menu," you would (1) move the pointer to the word "Run" in the

menu bar, (2) press and hold the mouse button, (3) move the pointer downward in the displayed menu until the word "Go" was highlighted, and (4) release the mouse button. Again, this sounds more complex than it actually is.

- *Opening icons.* Application programs, your own programs and word processing documents, pictures, and other material stored on the disks are often represented as little pictures called *icons* on the Macintosh screen. Some icons are shown in Figure 1-1. Disks themselves are also represented as disk icons. The most common operation on these icons is to *open* them. Opening an icon creates a window through which you can view the contents of the icon. (What precisely happens depends on what the icon represents.) To open an icon, you *select* the icon by clicking it and then choose Open from the File menu. A faster method is simply to double-click the icon.

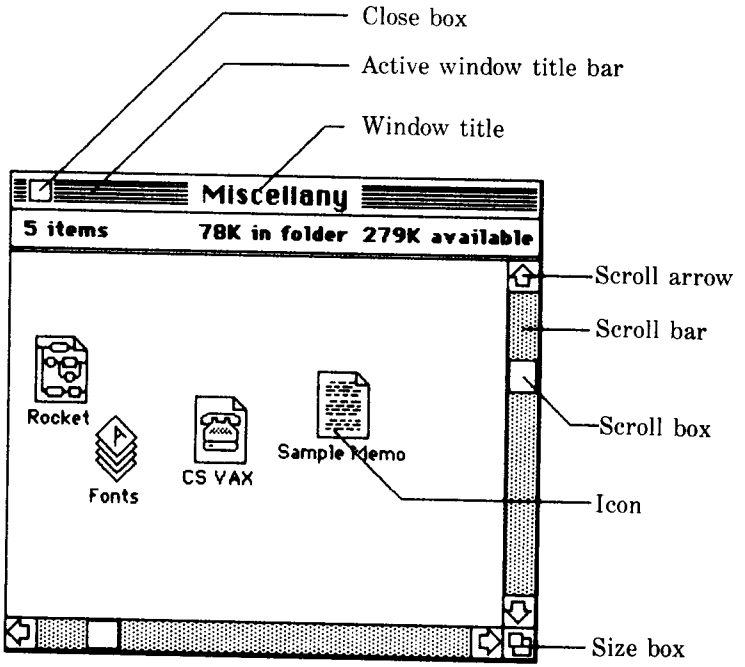


Figure 1-1.

Window anatomy