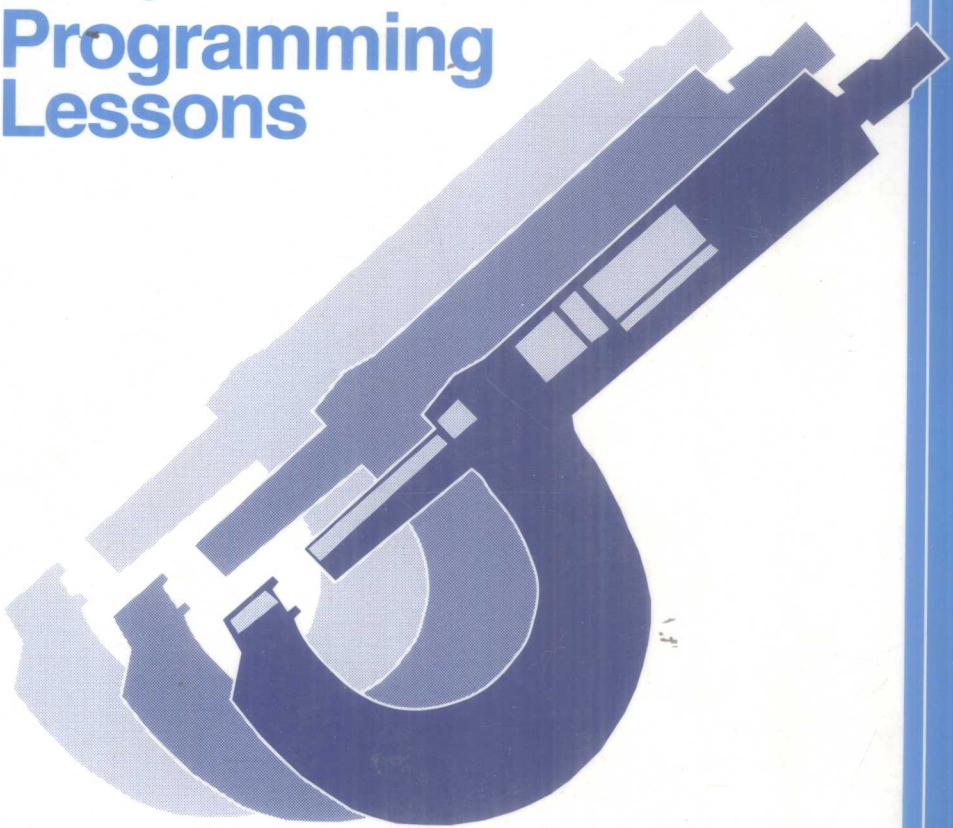
IBM® Microcomputer C

in 10 Programming Lessons



Julio Sanchez / Maria P. Canton

PRENTICE HALL PROGRAMMING SKILLS SERIES

IBM® Microcomputer C in 10 Programming Lessons

Julio Sanchez
Northern Montana College

Maria P. Canton
Skipanon Software Co.

PRENTICE HALL, Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Sanchez, Julio
IBM microcomputer C in 10 programming lessons / Julio Sanchez, Maria P. Canton.
p. cm.
Includes bibliographical references (p.) and index.
ISBN 0-13-726423-2
1. IBM Personal Computer--Programming. 2. C (Computer program language) I. Canton, Maria P. II. Title.
QA76.8.I2594S24 1992
005.265--dc20 92-19617
CIP

Acquisitions editor: Marcia Horton
Production editor: Bayani Mendoza de Leon
Cover designer: Wanda Lubelska
Copy editor: Brenda Melissaratos
Prepress buyer: Linda Behrens
Manufacturing buyer: Dave Dickey
Editorial assistant: Dolores Mars

A Simon & Schuster Company
Englewood Cliffs, New Jersey 07632

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-726423-2

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*
Editoria Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Preface

Twelve or fifteen years ago computer languages, at the college level, were taught almost exclusively to students of computer science and electrical or electronic engineering. As computing machines became an established fact of our culture, as well as a major tool in business and technology, these programming courses started appearing in non-computer science curricula; business programs began requiring Cobol and engineering schools ALGOL or Pascal. At present, many college programs in both science and humanities require theoretical courses in elementary computer science and skill courses in programming languages. Moreover, computer courses are rapidly migrating into the general education component of many curricula, to join, justifiably or not, the traditional freshman courses in English, speech, and mathematics.

The result of this shift is that part of the student population enrolled in computer programming courses either have no plans for using these skills professionally, have little natural inclination for the subject matter, or have had little previous contact with computers and programming. For them an encounter with a programming language can be a frightening experience. Those of us who, through daily association, are familiar with the intricacies of computer terminology and programming must make efforts to recall the intimidation and confusion that can result from a first confrontation with a computer language.

This series is an attempt at reducing some of the inherent difficulty and the psychological consternation usually associated with a first course in a computer programming language. The task is not trivial, especially if it is to be achieved without patronizing the reader, overdoing illustrations, or sacrificing correctness. Nevertheless, the simplification of a naturally complex subject must be achieved at some expense; namely, less rigor in the material presented, less elegance in the code samples, and the exclusion of the more intricate topics. Thus, we have bypassed certain subjects that we feel are not strictly required at the elementary level. Since performance of any skill must be acquired gradually, we have felt free to limit our present aim at providing a first and unpretentious level of programming achievement.

On the other hand we have stressed certain subjects and programming habits that we feel should be emphasized from the start. Students often approach a programming project by turning on the computer and executing the editor program, unaware that programming is not merely typing the code, but also designing, planning, thinking, and organizing. We have tried to stress the preparatory phase by underlining the usefulness of flowcharts, which are introduced in the first chapter and used in illustrations and

examples throughout the text, and by relating programming with the fundamentals of computer organization and architecture, introducing such topics as number systems, microprocessors, memory, operating system services, and video display hardware.

The titles in the Prentice Hall Programming Skills Series are designed for use in a computer lab setting, in which each student has access to a machine or terminal. The ideal teaching environment includes a computer screen or an overhead projector. Each programming lesson contains one or more sample programs that illustrate the concepts, keywords, and language elements covered in the chapter. Except for some introductory material presented in the first session, each class meeting can be approached as a discussion of the sample programs. Additionally, the code samples projected on a screen at class time can serve the instructor as unobtrusive lecture notes. A vocabulary list, questions, and programming exercises are included in each chapter. The exercises have been designed at several levels of complexity, allowing some individualization of the course contents, within its elementary nature. The text is compatible with all popular C language compilers for IBM microcomputers, including the various Microsoft, IBM, and Borland C compilers.

The authors would like express their appreciation to students, friends and associates who have provided help and support in this project. The course was classroom tested at the Great Falls campus in the Fall of 1991. At this time several students helped in revising and corroborating the material. The authors are particularly thankful to Darwin Stull, Frank Peirce, Maria Elena Lazaga, and Robert Keaster for locating typos, errors, and imprecisions in the original. We would also like to thank Marcia Horton, series editor at Prentice Hall, and her assistant Diana Penha, as well as Dr. Martha Ann Dow, Kevin Carlson, Sharon Lowman, and Marna Singleton at Northern Montana College.

Great Falls, Montana

Julio Sanchez
Maria P. Canton

Contents

1

The C Language, MS-DOS, Flowcharts, and Tools

1.0 WHAT IS C LANGUAGE ?

C is a computer programming language originally designed and implemented by Dennis Ritchie in 1972 while working at the Bell Laboratories. The first version of C language ran under the UNIX operating system on a DEC PDP-11 machine. The predecessor of C is a language called BCPL (Basic Combined Programming Language) developed in 1969 by Martin Richards of Cambridge University. The name C language originated in the fact that Bell Laboratories' version of BCPL, which was developed by Ken Thompson, is named B.

C is often described as a relatively small, compact, and simple high-level programming language suitable for general use. During the 1970s the C language was generally associated with the UNIX operating system. In fact, the newer versions of UNIX are written in C. C language became more popular in the 1980s, mainly due to its use in microcomputers. During this period several major software and hardware companies adopted C language as their preferred programming medium. Today C language is available in most microcomputers and in many mainframe and mini computers.

It is generally accepted that C is a member of the ALGOL family of programming languages. Therefore, it is closely related to the algebraic languages, such as ALGOL and Pascal, and not as similar to BASIC or FORTRAN.

Advantages of C Language

The following are the most often-cited advantages of C language:

1. C is not a specialized programming language; therefore, it is suitable for developing a wide range of applications, from major system programs to minor utilities.
2. Although C is a small language, it contains all the necessary operators, data types, and control structures to make it generally useful.
3. C includes an abundant collection of library functions for dealing with input/output, data and storage manipulations, system interface, and other functions not directly implemented in the language.
4. C data types and operators closely match the characteristics of the computer hardware. This makes C programs efficient as well as easy to interface with assembly language programs.
5. The C language is not tied to any particular environment or operating system. The language is available on machines that range from microcomputers to mainframes. For this reason, C programs are portable; that is, they are relatively easy to adapt to other computer systems.

Disadvantages of C Language

The following are the most often noted disadvantages of C:

1. Because C is not a language of very high level, it is not the easiest to learn. Beginners find that some constructions in C language are complicated and difficult to grasp.
2. The rules of C language are not very strict, and the compiler often permits considerable variations in the coding. This allows some laxity in style, which often leads to incorrect or inelegant programming habits.
3. Although the language itself is small and portable, most of the C library functions are devised to operate on a specific machine. This sometimes complicates the conversion of C language software to other implementations or systems.

C Language Authorities

The traditional authority on C languages is a book titled *The C Programming Language*, by Brian W. Kernighan and Dennis M. Ritchie (Prentice-Hall, 1978). This book, sometimes known as the "white book" due to its cover color, contained the original definition of C and became a de facto standard for the language which is referred to as "K & R C."

In 1982 a technical committee of the American National Standards Institute (ANSI) proposed a standard for C language. This standard, referred to as ANSI C, was approved in 1989. In this manner ANSI C became the industry-level authority for the language and its implementations.

In some respects ANSI C maintained the original design and essence of the C language, but in certain fields the standard tried to overcome limitations of K & R C. One of the consequences of these efforts on the part of the ANSI C developers has been a considerable expansion of the language, sometimes estimated at twice its original size, and also more complex. In the course of this book we will point out any differences between the C language as originally described by Kernighan and Ritchie and by the ANSI C standard.

Implementations of C in IBM Microcomputers

Several software companies have developed C compilers for the IBM microcomputers. Some of these products have gained and lost popular favor as other new versions or implementations of C were introduced to the market. Some of the better-known implementations of C for the IBM microcomputers are the Microsoft C and Quick C compilers, IBM C-2 compiler (which is a version of Microsoft C licenced to IBM), Intel iC-86 and iC-286 compilers, Borland Turbo C, Turbo C Professional, and C + +, Lattice C, Aztec C, Zortech C, and Metaware High C. The Microsoft and Borland C language development systems are discussed in some detail in Section 1.4.

1.1 THE IBM MICROCOMPUTER

In 1981 IBM introduced the first member of its microcomputer family, named the Personal Computer, or PC. Since then, many models of this machine have been developed and marketed. The PC XT, which included a fixed disk drive, was released in 1983. In 1985, IBM unveiled a more powerful personal computer, named the PC AT, and a home-market model named the PCjr. In 1987 the Personal Computer line was replaced with a new generation of machines called the Personal System /2 line. This new line includes the low-end models 25 and 30, the more advanced models 50, 55, and 60, and the high-end machines model 70 and 80. In 1990 IBM revived the idea of a home computer with a line of inexpensive desktops called the Personal System /1. In 1991 it introduced the high-end model 90 and model 95 which use a more powerful microprocessor and are equipped with a new graphics system.

In addition to IBM, many other companies manufacture computers that use similar hardware component and software. These machines are often called IBM-compatible microcomputers. Companies that manufacture IBM-compatible microcomputers include Tandy Corporation (Radio Shack), Compaq, and Hewlett-Packard.

In spite of variations in the different models of the IBM microcomputers and compatible machines, there are certain features that have remained unchanged:

1. All IBM microcomputers use a central processing unit (CPU) of the Intel iAPX 80x86 family. These chips include the 8088 CPU, used in the original PC; the 8086, used in some versions of the model 25 and 30, the 80286, used in the PC AT and some machines

of the PS/2 line, the 80386, used in the model 70 and 80. The newest member of the Intel iAPX 80x86 family, the 486, is used in the IBM model 90 and model 95 as well as in some non-IBM machines. There is also a 486 upgrade option for some versions of the PS/2 model 70.

2. All IBM and the IBM-compatible microcomputers are furnished with a fundamental program named Basic Input/Output System (*BIOS*). The BIOS program, which gains control automatically when the power switch is turned on, checks memory, tests and initializes the hardware components, and hands control over to the disk operating system program (MS-DOS). BIOS provides many programmer's services that are accessible to assembly language programs. We will use many of these services in the programming lessons. Although the BIOS program has been modified and updated in practically every new IBM machine, the fundamental services and storage locations have remained unchanged.

3. All IBM microcomputers are furnished with the MS-DOS disk operating system, developed by Microsoft Corporation. Five major versions and over a dozen sub-versions of MS-DOS have been introduced since 1981. However, Microsoft has maintained the operating system's fundamental structure and services.

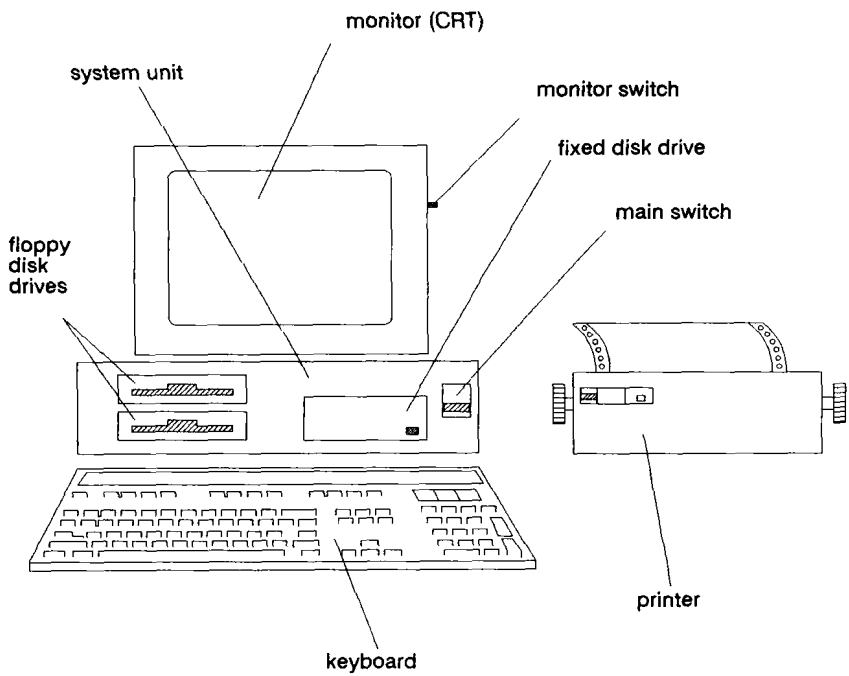


Figure 1.1 Principal Components of an IBM Microcomputer

Machine Description

Figure 1.1 shows the essential components of a typical IBM microcomputer system.

System unit. The system unit is the enclosure that holds the fundamental elements of a computer system. In an IBM microcomputer the system unit contains a board of electronic components, sometimes called the *motherboard*. The motherboard contains the central processing unit, memory, the BIOS program chips, and the auxiliary controllers. Also in the system unit are connectors for expansion cards, a socket for an optional mathematical coprocessor chip, storage devices, such as floppy and fixed disk drives, and electronic ports for communicating with other devices.

Floppy disk drives. Floppy disks or diskettes are removable electromagnetic devices used for storing computer data and programs. The diskette drive is often described as being similar to a record player turntable. The diskette holds data magnetically recorded on its surface, which is made of a special plastic called mylar. The floppy disk has a central hole that engages in a drive mechanism. This mechanism makes the diskette spin on its axis while a read-write arm moves radially along the diskette's recorded surface.

All IBM microcomputers are equipped with at least one diskette drive. The older machines use a 5 1/4 inch diameter floppy disk. The newer machines use a 3 1/2 inch version of the floppy disk which is encased in a plastic shell. The 3 1/2 inch model is usually called a microdisk. The capacity of a diskette drive is measured in the number of units of information (bytes) that can be stored in each diskette. The capacity of the floppy disk on IBM microcomputers has gone from 160 thousand bytes in the original personal computer to over 1.4 million bytes in some of the newer models.

Fixed disk drive. The fixed disk drive is an electromagnetic device for storing data and programs; it is quite similar to the diskette drive previously described. Fixed disks are also called hard disks or Winchester drives. The main difference between the diskette and the fixed disk system is that in the fixed disk the storage media are not removable from the drive. Hard disk drives usually contain several metal disks coated with a magnetic substance similar to the one used in the floppies. These metals disks are called *platters*.

Another difference between the floppy and the fixed disk drives is that the floppy drives spin the diskette only when data is being accessed, while the platters of a fixed drive start spinning when the computer is turned on. This determines that data can be accessed much more rapidly on a fixed disk since the fixed disk platters do not have to be brought up to speed before each read or write operation. In addition, fixed disk drives are built to a higher precision than floppy drives and are sealed from airborne particles that can damage the recording surface.

The capacity of a fixed disk drive is usually measured in megabytes of data that can be stored on its surface. One megabyte (1 Mb) is approximately equal to 1 million bytes. Popular fixed disk drive sizes in IBM microcomputers are 20, 40, 60, 80, and 120 megabytes.