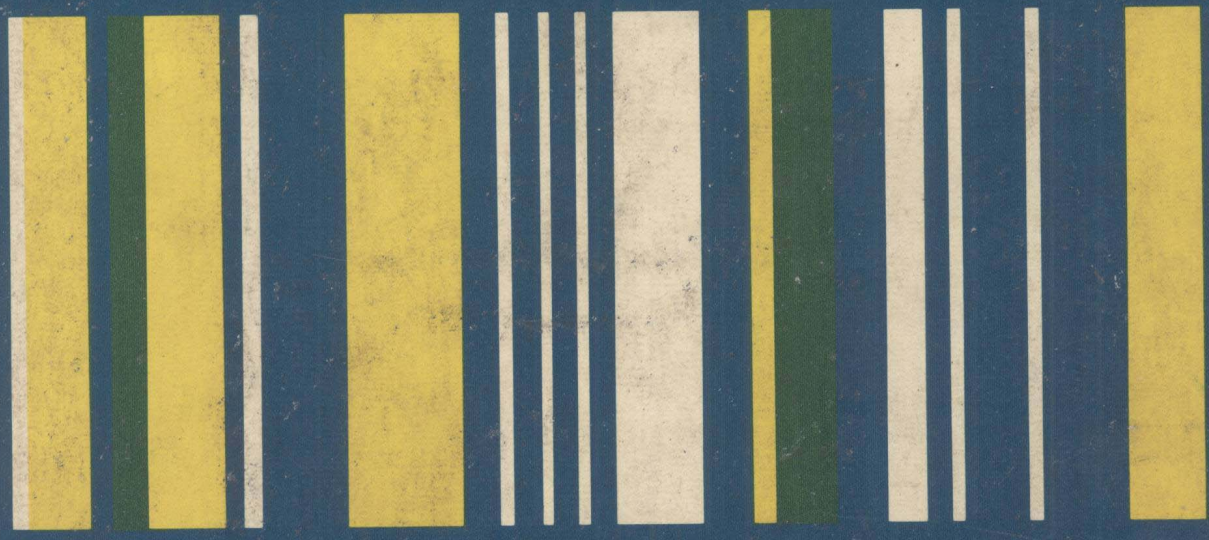
SECOND EDITION

ELEMENTARY COMPUTER PROGRAMMING IN FORTRAN IV



boris w. boguslavsky

elementary computer programming in fortran iv

2nd edition

boris w. boguslavsky
louisiana state university

reston publishing company, inc.
a prentice-hall company
reston, virginia

elementary computer programming in fortran iv

2nd edition

preface

to the second edition

Computers have become an ineradicable part of our lives, as if anyone needs to be told. Witness the spate of computer jokes that have replaced the Polish jokes since the election of Pope John Paul II. Some of them have even acquired an ethnic tint in reverse. Sample: Why does it take five Americans to multiply six by nine? One to punch the keys, and four to fix the equipment.

Ubiquitous as they are, computers seldom get good press. Just as rare incidents of crime get instant play in the media while predominantly good and uneventful behavior receives scant attention, computer failures and computer "mistakes" get the publicity while the enormous benefits that flow from the use of the computers seem to be known only to their users.

Few people realize that *computers do not make mistakes;* mistakes are being made by the people who use them. To paraphrase an old joke: Doctors bury their mistakes; mistakes bury computers. Manufacturers of faulty computers go bankrupt.

To any professional an ability to communicate with a computer has become, aside from his field of competency, a skill next in importance to reading. Whether his work has to do with men, maps, machines, or money, it requires him to seek information, to provide information, to make calculations, to set up records. All these tasks are simplified and speeded up by computers. Computers reach him even at home, in the form of tape-recorded messages, bills, and junk mail.

Gibbon said, way back in 1776: "All that is human must retrograde if it do not advance." To stay abreast of the developments in today's world, we must gain familiarity with computers. Better yet, we must learn to use computers. Fortunately, any one who can read can learn to program a computer, that is, to make the computer answer questions. All that is needed is a textbook that presents the art of writing instructions to the computer in short, easy-to-understand steps. Of course, access to a computer will make the reader's progress more rapid and tangible.

To provide such a textbook, I have revised the first edition of my ELEMENTARY COMPUTER PROGRAMMING IN FORTRAN IV to make the material

clearer, simpler and easier for a beginner to assimilate. At the same time I have added material to make the book interesting to a programmer with some experience. To keep the book from becoming too long and unwieldy, I have deleted the chapters on graphics and magnetic tape and disc. The material on graphics, in greatly expanded form, will be published in a separate book. To make it easier for a student to check his or her progress, I have provided answers to every odd question and problem in the textbook.

I close with words of appreciation to two members of the System Network Computer Center at the Louisiana State University. They are Robert L. Jenkins, manager of computer services, and Malcom D. McNaylor, manager of computer systems. These two gentlemen are walking encyclopedias of computer science. Time and again they pulled me out of the depths of my dilemmas, and they did it with speed, courtesy, and, last but not least, with a clear explanation. These two men are not only capable scientists: they are excellent teachers.

Boris W. Boguslavsky

contents

chapter 1

digital computers - FORTRAN

1-1. introduction

A digital electronic computer is an automatic device that (1) accepts (a) data and (b) instructions on how to manipulate these data, and stores both in its magnetic memory in the form of digits; (2) performs step-by-step operations on the data in accordance with the instructions; and (3) issues the results of these operations. The *data* consist of the related items of information pertaining to a field of activity or study, such as the payroll of an industrial plant, the dimensions of a structure, or the grades received by students in an examination. The sequence of instructions that guide the computer in the processing of the data is called a *program.* A program is written by a person, called a *programmer.* It is the purpose of this book to teach you to become a computer programmer.

A competent programmer can usually write a short program directly. Longer or involved programs often require a carefully laid plan. A plan may be in the form of either an *algorithm** or a *flow chart.* An algorithm may be defined as a finite sequence of unambiguous steps that lead to a desired result. A flow chart is a graphical representation of the operations that bring forth a result. An algorithm and a flow chart to find the sum of any two numbers, and a program that parallels them may be written as follows:

* From the name of the Arab mathematician, al-Khwarizmi, 780-850 A.D.

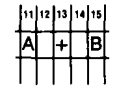
Algorithm	*Flow Chart*	*Program*
1. Given any two numbers, record or store them under the names J and K	Read values of J and K on a data card	READ J, K
2. Add the numbers stored at J and K, and store the sum under N	Find the sum N = J + K	N = J + K
3. Write the number stored at N	Write the sum N	WRITE N

If the numbers 7 and −2 accompany the program as the data, the machine will take the path described in the algorithm, or the flow chart, or the program, and will print the result, 5.

A program is prepared for every different problem that is submitted to the computer for a solution. The computer is only a machine: presenting to it, as the data, two numbers, such as 7 and −2 above, does not cause the computer to calculate their sum, 5, unless it receives an appropriate instruction, such as N = J + K. Conversely, the computer executes the instruction N = J + K only if the values of J and K are *defined*, that is, if it receives the values of J and K. Finally, the computer does not print the result unless it receives a command to do so, such as WRITE N.

A program is usually written on a coding sheet, such as is shown in Fig. 1-1, employing the notation of numbers, letters of the alphabet, punctuation marks, mathematical and other symbols, and blanks. The ten numerical digits (0 to 9) and the 26 alphabetical characters are usually referred to as the *alphameric* (or *alphanumeric*) characters; all the other marks and symbols are called *special* characters, and most commonly include:

$$+ \quad - \quad * \quad / \quad = \quad , \quad . \quad (\quad) \quad ' \quad \textit{blank}$$



The special character *blank* is simply a blank space on the coding sheet; in the illustration shown above, right,blanks occupy spaces 12 and 14; a blank is a character, as much as A, +, and B. If the information containing blanks is printed on plain paper, as will happen in many places in this textbook, the blanks may be indicated by the symbol ␣; thus the illustration above may be printed A␣+␣B.

Three additional special characters, $ (dollar), # (pound), and @ (at), are also known as *national* characters.

For their interpretation by the computer, the alphameric and special characters are transformed into magnetic spots on tape or disks coated with iron oxide or into holes punched on paper tape or cards. Figure 1-2 shows a typical card punched with holes that are coded to represent X = 2.4.

The punched card is the most commonly used medium for recording in-

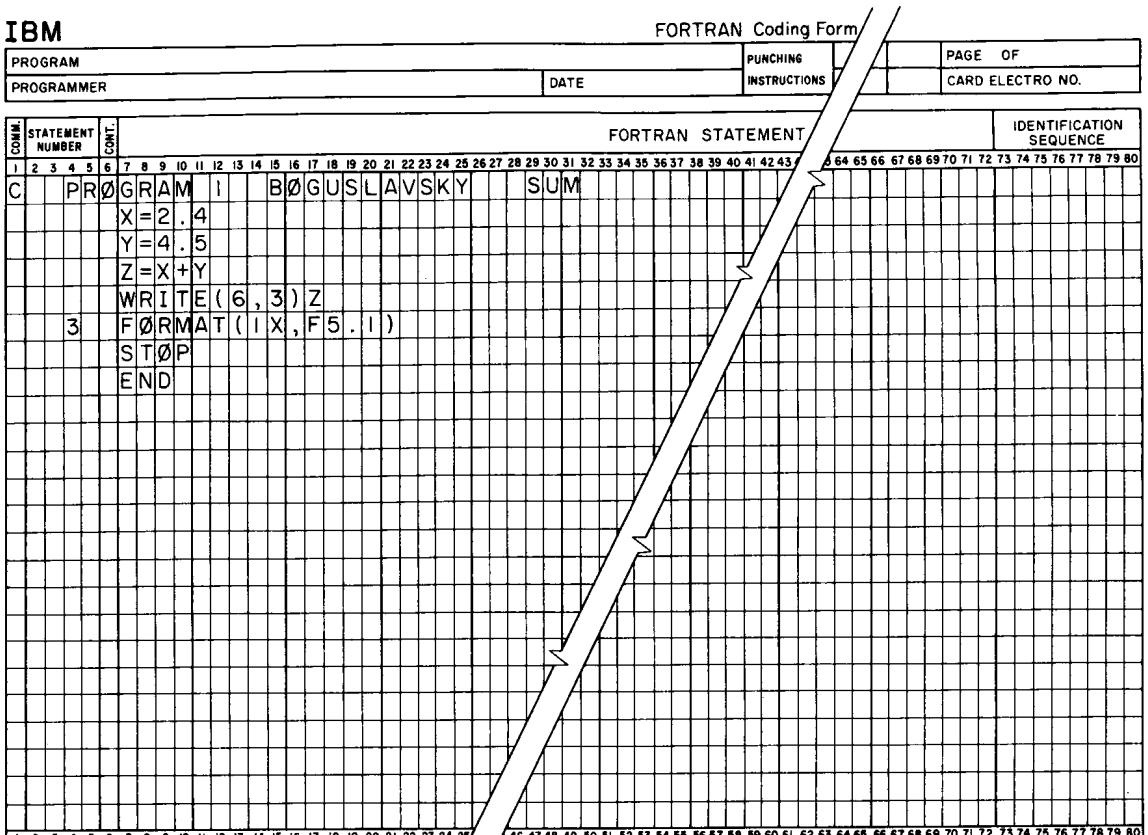

IBM
FORTRAN Coding Form
PROGRAM
PROGRAMMER
DATE
PUNCHING INSTRUCTIONS
PAGE OF
CARD ELECTRO NO.
COMM.
STATEMENT NUMBER
CONT.
FORTRAN STATEMENT
IDENTIFICATION SEQUENCE
C PRØGRAM 1 BØGUSLAVSKY SUM
X=2.4
Y=4.5
Z=X+Y
WRITE(6,3)Z
3 FØRMAT(1X,F5.1)
STØP
END

Figure 1-1

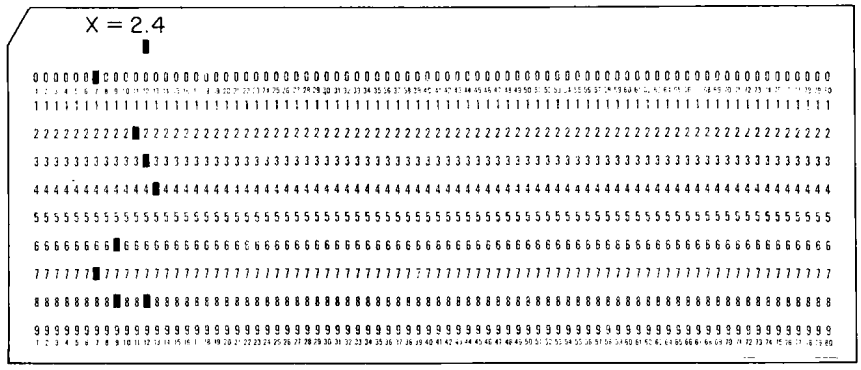


Figure 1-2

formation for a computer because, when additions or corrections have to be made, individual cards can be easily inserted into or removed from a deck of cards. For instructing beginners in the science of computer programming, exclusive reliance is made on punched cards, because mistakes abound in the beginners' work.

A simplified example of a program and the accompanying data are shown on four lines of a coding form in Fig. 1-3.

For submission to the computer, these four lines will be punched on four cards, one line per card (a card may contain only one line of information). The program shown will be encoded on the first three cards, and the data on the fourth.*

When the cards are presented to the computer, the machine will read the program on the first three cards, will scan the data card and interpret 6.0 as P, 4.4 as Q, and 4.0 as R, will add P and Q (6.0 + 4.4 = 10.4), will subtract R from P (6.0 − 4.0 = 2.0), will divide the sum by the difference (10.4/2.0 = 5.2), and will print the result (5.2) on a sheet of paper.

COMM.	STATEMENT NUMBER				CONT.	FORTRAN STATEMENT																										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
						R	E	A	D		P	,	Q	,	R																	
						Y	=	(	P	+	Q	)	/	(	P	−	R	)							P	R	O	G	R	A	M	
						W	R	I	T	E		Y																				
						6	.	0					4	.	4					4	.	0			D	A	T	A				

Figure 1-3

* Additional cards required to establish unfailing communication with the computer will be described later in the book.

More complex problems are handled by the computer in a similar fashion; the only difference is that there are more cards, both for the data and for the instructions that guide the computer in processing the data.

1-2. digital and analog computers

Article 1-1 began with a definition of a *digital* electronic computer. Modern electronic computers are divided into two basic types: *digital* computers and *analog* computers.

A digital computer is a device that receives information in the form of discrete electrical pulses transmitted in coded combinations, stores these pulses in its magnetic memory as strings of only two digits, 0 and 1, and, when these strings represent numbers, operates on them; because the computer preserves the effects of all the pulses, the values stored in the computer and the results obtained by it are *exact* (within the range of the machine).

An analog computer operates on numbers that are represented in a continuous form by measurable amounts of current or voltage: the larger the current or voltage, the larger the number. Since measurements of the currents or voltages, and therefore of the numbers, are made by ammeters or voltmeters, which can be read only approximately, the results supplied by an analog computer are never precise; however, their accuracy is generally sufficient for most purposes.

The difference between a digital computer and an analog computer is comparable to the difference between a beam balance with pans and a spring balance. In the beam balance the weight of an object placed on one pan is found by adding together the discrete known weights placed on the other pan; in the spring balance, such as a bathroom scale, the weight is found by reading a number corresponding to a measured deformation of the spring.

This text will be devoted exclusively to digital computers.

1-3. programming

Although the program and the associated data are always submitted to the computer together, they seldom have the same origin, and their respective developments may be widely separated in time and place. A problem and its data may originate with a scientist, a businessman, or even a schoolboy who has learned the ages of his classmates and wants to determine the average age of the class. The program, on the other hand, which will guide the computer through the solution of the problem, is usually prepared by a programmer, the person who can devise the requisite algorithm or flow chart and who knows the details of communication with the computer in a language that it accepts. For instance,

a competent programmer will not permit the second statement of the program shown in Fig. 1-3 to be written

$$Y = \frac{P + Q}{P - Q}$$

that is, in two lines and without parentheses: such a statement cannot be punched in a single line on a card. Again, the programmer will not allow symbols such as Δ or π to appear in a program, because these symbols are not to be found among the alphameric and special characters that can be punched on a card; he would replace Δ by DELTA and π by PI (or PIE).

The computer—strictly speaking, the computing system—is designed to accept a fixed number of symbols and of combinations of these symbols; therefore, it can function only when only these symbols and these combinations are presented to it. The programmer is the person who can digest the problem presented to him by a scientist or a schoolboy and write logical instructions to the computer, using the symbols or combinations of symbols and procedures that are acceptable to the computer.

1-4. FORTRAN IV (four) language; compiler

Although the program of Fig. 1-3 appears to be written in the ordinary English language in combination with a mathematical expression, actually it is written in a specialized and rigidly formulated language—a *programming language*—called FORTRAN IV (there were earlier versions of FORTRAN). This language was invented to simplify writing computer programs.

There are two languages involved in the solution of a problem by a computer: (1) the language of the programmer, a human being, who prepares the program, that is, the sequence of instructions that guide the computer; and (2) the language of the computer, a machine, which must interpret and execute the instructions of the program.

The language of a computer is called a *machine language.* A machine language consists of a limited number of one-step instructions that a computer is designed to interpret and execute. Generally a machine-language instruction consists of two parts: (1) an order to perform an operation and (2) the location of the item of information that is to be operated on; both the operation and the location are identified in a numerical code. A programming-language instruction, such as X = A + B, is broken down, in the machine language, into a sequence of three steps, GET A, ADD B, STORE AT X, all expressed as two-part numbers.

These numbers enter the computer expressed in terms of only two digits, 0 and 1. These two digits can be represented in a machine by the electromagnetic effects of an electric current flowing in the one or opposite directions along a wire. Since only two directions are possible, only 0 and 1 can be represented

by their effects. Therefore the computer can read instructions and operate on data only if they are presented to it in the form of numbers such as 0101 or 1000101. (Numbers expressed in terms of 0 and 1 are called *binary* numbers; a description of the binary system of numbers is given in Appendix C.)

The programmer, being only human, cannot write a program using only 0's and 1's without making mistakes or, worse, losing his mind. Therefore he uses a programming language, such as FORTRAN. FORTRAN is a problem-oriented programming language, that is, a language that, instead of reflecting the machine-language idiosyncrasies of a particular computer, enables the programmer to write the solution of a problem utilizing familiar English words and mathematical expressions. FORTRAN is a most widely used problem-oriented programming language.* FORTRAN is a coined word, formed by joining the first syllables of FOR*mula* TRAN*slation.*

The program written by the programmer in a programming language is called the *source program,* and the language itself (FORTRAN) is called the *source language;* the program written in the machine language is called the *object program* and the machine language is called the *object language.*

A source program written in a programming language, such as FORTRAN, must be translatable into the machine language. This must be so, because a gap exists between the source program that the programmer writes and the object program that the computer reads and executes, just as a gap exists between programs written in English and French. The gap in the latter case is filled by a translator or a dictionary; the gap between the source program and the object program is filled by a *compiler.*

A compiler is a special *program,* written in the machine language, which is loaded into the computer ahead of the source program, written in FORTRAN (or any other programming language). Under instructions from the compiler, the computer reads the source program and translates it into the machine language. The compiler does not just convert one source program statement into one machine-language statement: several minutely detailed machine-language statements are generally generated to create the effect of a single source statement, and the compiler is capable of producing this one-to-many correspondence between the source and the machine languages. In fact, if an elaborately designed compiler is stored in a computer, the source program can be made to look almost like the actual mathematical expressions in the original problem.

A computer may have a library of several different compilers varying in their complexity and efficiency. There are four compilers generally associated with FORTRAN: the E compiler, the G compiler, the H compiler, and the WATFIV compiler. The WATFIV compiler, developed at the University of Waterloo (Ontario), is a truncated compiler, that is, it covers only the elementary

* More than 100 programming languages have been developed for communication with computers. Many of these are completely dead; many others have only special applications. FORTRAN's nearest competitors are ALGOL, COBOL, and PL/1 languages. FORTRAN was the first programming language to be standardized through the U.S.A. Standards Institute procedures.