FORTRAN PROGRAMMING, PROGRAMS, AND SCHEMATIC STORAGE MAPS

FORTRAN PROGRAMMING, PROGRAMS, AND SCHEMATIC STORAGE MAPS

Myron G. Mochel
Professor of Mechanical Engineering
Clarkson College of Technology

McGRAW-HILL BOOK COMPANY
New York St. Louis San Francisco Düsseldorf
Johannesburg Kuala Lumpur London Mexico Montreal New Delhi
Panama Rio de Janeiro Singapore Sydney Toronto

PREFACE

This text is an introduction to programming in the FORTRAN language. Its purpose is to give students, as quickly as possible, enough information to write and debug their own programs. Therefore it contains no information that is not directly related to getting a program on and off a computer on a closed-shop batch-processed basis. I have not discussed computer operation, machine language, or assembly language.

Instead, the text material is centered around two methods of presentation that are probably unique in the field of digital computer instruction: *Ilustrative programs* and *schematic storage maps.* These concepts have been used at Clarkson College of Technology since the fall of 1966; Clarkson freshmen have received instruction in the FORTRAN language since the fall of 1961. Schematic storage maps have proved particularly helpful, since they afford the student a schematic understanding of the internal workings of the computer without a knowledge of the complex electronics involved.

The material is presented in a learning sequence. This sequence may not satisfy everyone's desires exactly—it represents one way of proceeding. The material is flexible enough to permit adjustments according to an instructor's wishes.

The initial phases of learning the FORTRAN language should center around evaluating mathematical equations. This should be followed by a study of the logic of data manipulation. Sections 79, 80, and 81 describe methods of number manipulation and Sections 118, 119, and 120 demonstrate how blood types can be matched.

Many programs are based on engineering and scientific applications. They have been chosen in order to illustrate techniques and problems arising from the use of a computer and the FORTRAN language. Sections 51 and 52 illustrate in an unusual manner why it is important to understand the difference between the decimal system and the binary system of numbers.

I wish to thank Dr. Alexander L. Cicchinelli, Assistant Director of Analytical Studies and Management Analysis of the Central Administration of the State University of New York, and Professor William H. Lyman, Acting Director of the Clarkson College Computing Center, for their help and advice.

Myron G. Mochel

CONTENTS

Section 1
THE DIGITAL COMPUTER PROGRAM

A digital computer can do little more than add digits, sense which of two digits is larger, and move digits around in its storage according to a set of instructions. But a digital computer is fast, and these basic capabilities can be combined and repeated fast enough to make a computer respond as if it were capable of division, multiplication, and other higher operations directly. Integration and differentiation by the methods taught in introductory calculus are not performed, but definite integrals can be approximated and derivatives can be evaluated by numerical methods involving only simple arithmetic operations. A digital computer also can be made to respond as if it were looking up certain handbook information such as logarithms, functions of angles, and square roots.

To keep up these appearances, a digital computer must follow a sequence of precise instructions. The sequence of instructions is called a "computer program," and constructing the sequence is called "computer programming."

There are four basic units and a control unit that essentially go into a digital computer. Their arrangement is shown schematically in Figure 1.

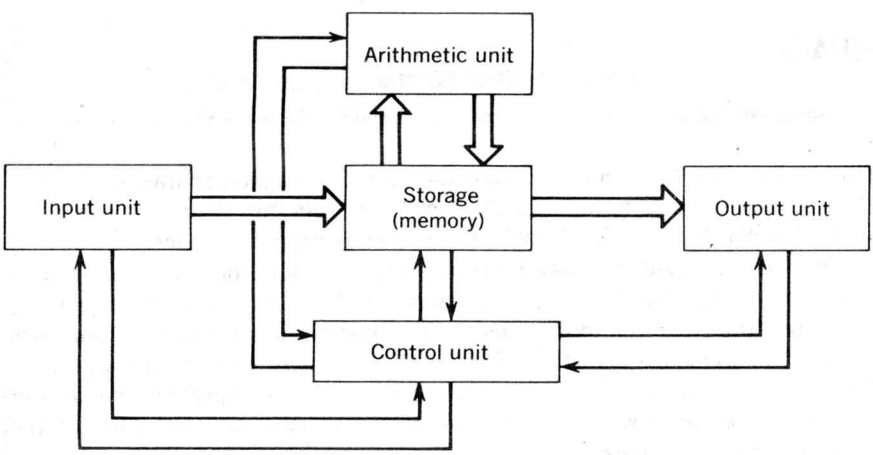

FIGURE 1

Section 2
MACHINE LANGUAGE

The most primitive language for programming a digital computer is the machine language. It is an entirely numeric language; no alphabetic characters are used to make up instructions, and it requires the programmer to have a detailed knowledge of the construction of the computer. Machine language is the only language understood

or interpreted by a computer without translation, but it is too detailed for programmers to work with effectively.

3000300030003000300030003000300030003000300030003000300030003000300030003000300030003000300030003000300030003000300030

Section 3
ASSEMBLY LANGUAGE

3000300030003000300030003000300030003000300030003000300030003000300030003000300030003000300030003000300030003000300030

A digital computer usually can be programmed in assembly language as well as in machine language. Assembly language uses alphabetic and numeric characters to make up the instructions. The alphabetic characters make the assembly language easier for a programmer to learn and retain, but it is still quite detailed. When an assembly language program is written, the computer must translate it, or assemble it, into the machine language. Both machine language and assembly language are complex and time-consuming to use; they are used primarily by the "professional programmer" for tasks which cannot be accomplished by programs written in a compiler language.

4000400040004000400040004000400040004000400040004000400040004000400040004000400040004000400040004000400040004000400040

Section 4
COMPILER LANGUAGE

4000400040004000400040004000400040004000400040004000400040004000400040004000400040004000400040004000400040004000400040

A compiler uses a few English words combined with punctuation characters and the familiar algebraic symbols to instruct or program a computer. Compiler languages, such as FORTRAN, ALGOL, and COBOL, have been developed and designed so that they can be easily learned and used by people who are not particularly interested in the details of the logic and design of the computer. When a compiler language program is written, the computer must translate it, or compile it, into machine language. The engineer, scientist, economist, businessman, or someone with only an occasional problem to solve on a computer will usually program the computer in a compiler language. This text deals with the FORTRAN compiler language and refers to the programmer's compiler language as the "source program."

5000500050005000500050005000500050005000500050005000500050005000500050005000500050005000500050005000500050005000500050

Section 5
COMMUNICATION WITH A DIGITAL COMPUTER

5000500050005000500050005000500050005000500050005000500050005000500050005000500050005000500050005000500050005000500050

The source program written on a sheet of paper by a programmer is not in a form which can be communicated directly into a computer. Similarly, a solution of a computational problem in a computer's storage is not in a form easily comprehended by a programmer.

In this text, our communication with a digital computer essentially follows the sequence below.

FORTRAN IV LANGUAGE ⇄ TRANSLATOR ⇄ MACHINE LANGUAGE

Peripheral devices of some sort are needed to take care of input and output. Devices commonly used for these purposes are (1) a typewriter, (2) punched paper tape, (3) punched cards, (4) magnetic tape, and (5) magnetic discs. Output from a computer also can be put directly onto a sheet of paper by a printer.

Whichever type of input/output device is used, a computer is said to "read" when it takes information into its storage and is said to "write" when it relays the contents of some section of its storage to an external device.

This text considers a computer that will read a source program and data into its storage by interpreting the holes in punched cards and will write by printing a line on a sheet of paper to relay information from its storage to the user.

Section 6
THE PUNCHED CARD CODE

A card is organized with 80 vertical columns across the card and 12 horizontal rows down the card. Figure 2 shows a blank (i.e., unpunched) card on which the columns are numbered 1 to 80 and 10 of the 12 rows are numbered 0 to 9. The position of the two unnumbered rows at the top of the card is apparent in Figure 3. Each column can contain the punched hole representation for any one of the FORTRAN characters, i.e., letters, digits, arithmetic operators, and punctuation symbols.

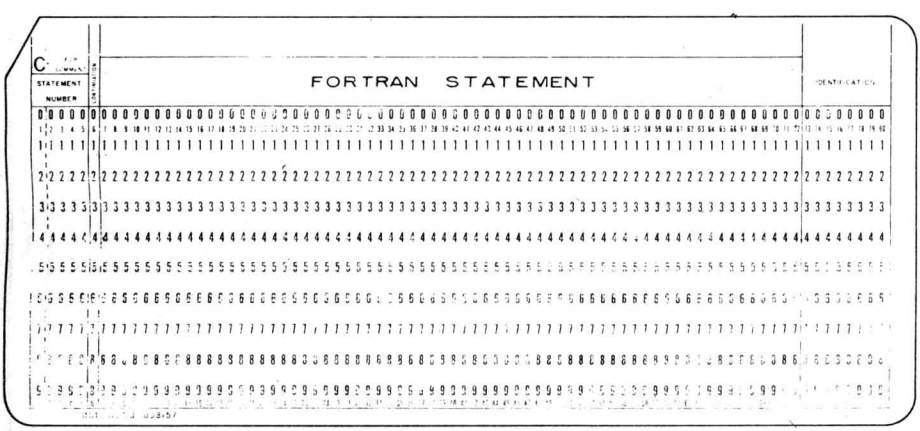

FIGURE 2 A BLANK CARD

Figure 3 is a card that has been punched by a keypunch machine. A keypunch machine punches a one-, two-, or three-hole code into a column of the card and simultaneously prints the character corresponding to the code on the top edge of the card.

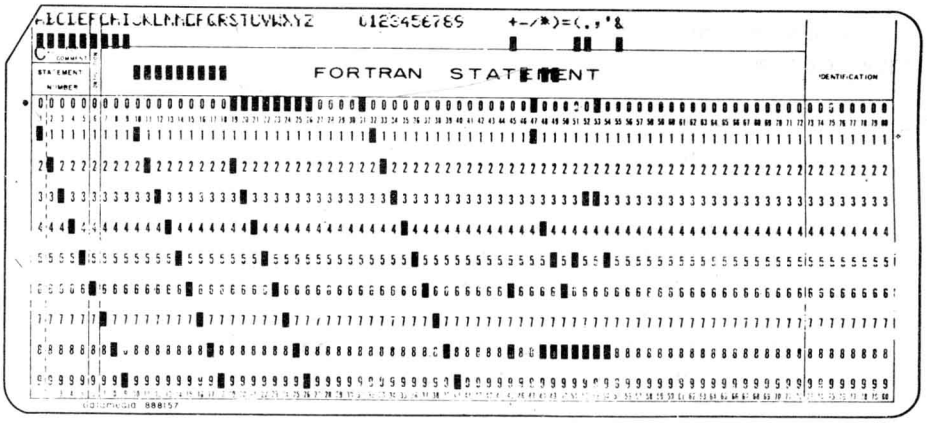

FIGURE 3 A KEYPUNCHED CARD

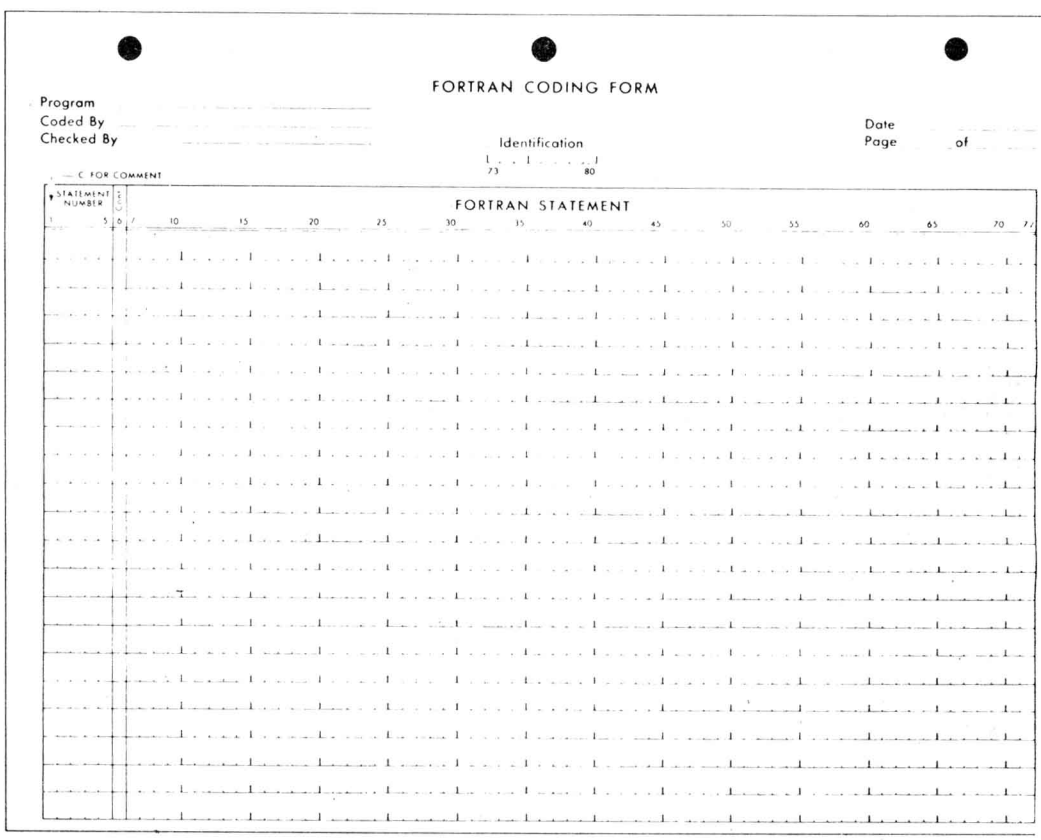

FIGURE 4 FORTRAN CODING FORM

Each letter of the alphabet has a unique representation made up of two holes; each of the numbers 0 to 9 is represented by one hole; and mathematical operators and punctuation symbols are coded by one, two, or three holes.

Section 7
FORTRAN CODING FORM

After a source program has been written on a sheet of paper, it must be transferred character by character, line for line, onto keypunched cards before it can be read and interpreted by the computer. The computer receives information according to the keypunched holes. The printing on the card is not readable by the computer.

Figure 4 is a standard FORTRAN coding form used for convenience in writing a program on a sheet of paper. The column numbers printed on the coding form in Figure 4 correspond to column numbers printed on the keypunched card in Figure 3.

Each line of the coding form represents one instruction or statement of the source program. Each line or statement on the coding form is keypunched, column for column, onto a separate card.

Section 8
METHOD OF INTRODUCING NEW TOPICS OF THE FORTRAN LANGUAGE

To thoroughly understand a language it is necessary to be familiar with the rules of grammar of the language. As each new rule of grammar or punctuation is introduced, there will be an explanation and an illustration.

All illustrations center around a series of illustrative programs. Each illustrative program is headed by a C card stating the "Illustrative Program Number (IPN)."

Section 9
THE C STATEMENT (ILLUSTRATIVE PROGRAM NUMBER 1)

The first statement of a source program has C in column 1. The C stands for "comment" and instructs the computer that the rest of the card, columns 2 through 80, contains a comment to the programmer or reader. Often, many C cards are included in a program to provide more introductory information. It is a good practice to include comment cards at the beginning of programs for identification purposes. When a program is lengthy, comment cards in the middle of the program will help the programmer and reader follow the program easily.

Figure 5 shows seven cards that represent a very simple program. The first six cards demonstrate a variety of ways to convey a C message as long as the C is in column 1.

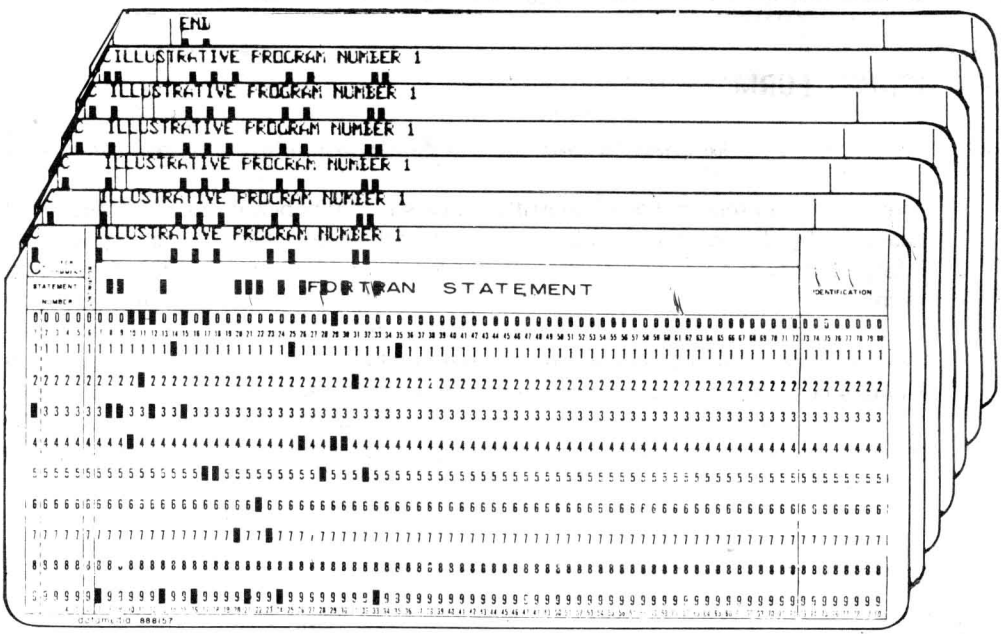

FIGURE 5 KEYPUNCHED CARDS THAT REPRESENT A VERY SIMPLE PROGRAM

The END card is explained in the next section.

The usual form is to start the message in column 7, as shown in the first line of the output of IPN 1 shown below.

```
C      ILLUSTRATIVE PROGRAM NUMBER 1
C     ILLUSTRATIVE PROGRAM NUMBER 1
C    ILLUSTRATIVE PROGRAM NUMBER 1
C   ILLUSTRATIVE PROGRAM NUMBER 1
C  ILLUSTRATIVE PROGRAM NUMBER 1
CILLUSTRATIVE PROGRAM NUMBER 1
0001        END
```

The four-digit number to the left of a statement (C statements excluded) is automatically printed on the output sheet by the computer. This is a convenient and automatic way for a programmer to refer to lines of printout.

1000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010

Section 10
END STATEMENTS (IPN 1)

1000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010

An END statement must appear at the end of a program. It is the only way that the computer knows that the compiling of the source program is complete. See Illustrative Program Number 1.

Section 11
REAL AND INTEGER VARIABLE NAMES AND REAL AND INTEGER NUMBERS (IPN 2 and IPN 3)

Programs contain variables, and these variables must be assigned names. The name of a variable consists of from one to six characters, where the characters can be either letters or digits. The first character of a name, however, must be a letter.

When the first letter of a variable's name is either I, J, K, L, M, or N, the variable is called an "integer variable." Any numerical value assigned to an integer variable name will be an integer: it will not contain a decimal point. Integer variables and integer numbers are sometimes referred to as "fixed-point variables" and "fixed-point numbers." When the first letter of the name of a variable is any letter other than I, J, K, L, M, or N, the variable is called a "real variable." Any numerical value assigned to a real variable will contain a decimal point. Real variables and real numbers are sometimes referred to as "floating-point variables" and "floating-point numbers."

These two rules can be overridden by the programmer's including specific declarations to the contrary; these declarations will be considered later (Section 38).

Examples of real variable names are

A	HIGH
XYZ	RESLT1
R235	RESLT2
H3X7B9	RESLT3

Examples of real numbers are

−6.	860.086
35.0	−74.3695
108.3	26385.4
2869.25	5016.2

Examples of integer variable names are

J	N5X6Z
KALL	LRESL1
I34568	LRESL2
MAN	LRESL3

Examples of integer numbers are

6	860
−35	74
108	−26385
2869	5016

In Illustrative Program Number 2 the variables A, BATE, SYSTEM, XA394, Z5Y3AA, RESULT, and DEF23 are real variables. Any numbers assigned to these variables must have decimal points.

In Illustrative Program Number 3 the variables IQ, JATE, KSYSTE, LA394, M5Y3AA, NET, and JDEF23 are integer variables. Any numbers assigned to these variables must not have decimal points.

In Illustrative Programs Numbers 2 and 3 (as well as all programs) four-digit numbers are shown to the left of each statement. The computer automatically prints these numbers on the output sheet. The information following the END statement represents the result of the WRITE statements of the program.

```
      C     ILLUSTRATIVE PROGRAM NUMBER 2
      C     THIS PROGRAM PRINTS OUT CERTAIN PROGRAMMED INFORMATION.
      C     ANY SENTENCES, SUCH AS THIS ONE, THAT START WITH A C IN
      C     COLUMN ONE ARE FOR INFORMATION ONLY.
      C     ONLY REAL VARIABLES AND REAL NUMBERS ARE USED
      C     IN THIS PROGRAM.
      C     THE SEVEN STATEMENTS BELOW DEFINE NEW VARIABLES.
      C     THE VARIABLES A, BATE, SYSTEM, XA394, Z5Y3AA,
      C     RESULT AND DEF23 ARE REAL VARIABLES.
      C     THE NUMBERS TO THE LEFT, 0001, 0002, 0003 ETC. ARE
      C     ASSIGNED BY THE COMPUTER.
      C     THE NUMBERS 10., -20.6378, 3.5, 378.287, -0.583
      C     AND 125.39 ARE REAL NUMBERS.
0001        A = 10.
0002        BATE = -20.6378
0003        SYSTEM = 3.5
0004        XA394 = 378.287
0005        Z5Y3AA = -0.583
0006        RESULT = A + BATE
0007        DEF23 = XA394 - 125.39
      C     THE FIVE WRITE STATEMENTS BELOW ASK THE COMPUTER TO PRINT OUT THE
      C     NUMERICAL VALUES THAT CORRESPOND TO THE VARIABLES' NAMES.
      C     IN THE FIVE WRITE STATEMENTS, THE FIRST NUMBER WITHIN THE
      C     PARENTHESES IS A DEVICE NUMBER.
      C     THE SECOND NUMBER IS A FORMAT STATEMENT NUMBER.
0008        WRITE(3, 1) A
0009        WRITE(3, 13) BATE, SYSTEM
0010        WRITE(3, 407) XA394, Z5Y3AA, RESULT
0011        WRITE(3, 9638) DEF23, A, SYSTEM, Z5Y3AA
0012        WRITE (3, 36275) RESULT, A, DEF23, SYSTEM, BATE
      C     THE STATEMENTS BELOW ARE FORMAT STATEMENTS AND ARE LABELED WITH
      C     A NUMBER.
      C     FOLLOWING THE LEFT PARENTHESIS OF EACH FORMAT STATEMENT
      C     IS THE CARRIAGE CONTROL. THIS, IN TURN, IS
      C     FOLLOWED BY FORMAT SPECIFICATIONS.
      C     FOLLOWING THE CARRIAGE CONTROL IN THE FIRST FOUR FORMAT STATEMENTS
      C     BELOW ARE FORMAT SPECIFICATIONS. THESE ARE F SPECIFICATIONS.
      C     IN THE FORMAT STATEMENT 36275, A BLANK COLUMN SYMBOL FOLLOWS THE
      C     CARRIAGE CONTROL.
      C     FOLLOWING EACH F THERE IS A NUMBER CONTAINING A DECIMAL POINT.
      C     THE DIGITS TO THE LEFT OF THE DECIMAL POINT ESTABLISH THE TOTAL
      C     NUMBER OF COLUMNS RESERVED FOR THE NUMERICAL VALUE.
      C     THE NUMBER TO THE RIGHT OF THE DECIMAL POINT ESTABLISHES HOW MANY
      C     PLACES THERE ARE AFTER THE DECIMAL POINT.
0013      1 FORMAT (1X, F8.0)
0014     13 FORMAT('0', F10.4, F6.1)
0015    407 FORMAT ('-', 2F12.3, F13.4)
0016   9638 FORMAT ('1', 4F9.3)
0017  36275 FORMAT ('+', 45X, 5F11.4)
      C     SEE SECTION 18 FOR AN EXPLANATION OF 45X
      C     THE FOLLOWING STATEMENT MUST APPEAR AT THE END OF ALL FORTRAN
      C     PROGRAMS.
      C     FOR A DISCUSSION OF THE PRINTOUT SEE SECTION 19.
0018        END
```

```
   10.

-20.6378    3.5

   378.287       -0.583      -10.6378


252.897   10.000    3.500   -0.583          -10.6378    10.0000   252.8969     3.5000   -20.6378
```

```
                C     ILLUSTRATIVE PROGRAM NUMBER 3
                C     THIS PROGRAM PRINTS OUT CERTAIN PROGRAMMED INFORMATION.
                C     THE STATEMENTS BELOW INTRODUCE INTEGER VARIABLES.
                C     THE VARIABLES I, JATE, KSYSTE, LA394, M5Y3AA, NET AND JDEF23 ARE
                C     INTEGER VARIABLES.
                C     THE NUMBERS 10, -21, 3, -378, 2 AND 125 ARE INTEGER NUMBERS.
 0001                 IQ = 10
 0002                 JATE = -21
 0003                 KSYSTE = 3
 0004                 LA394 = -378
 0005                 M5Y3AA = 2
 0006                 NET = I + JATE
 0007                 JDEF23 = LA394 - 125
                C     THE STATEMENTS BELOW ASK THE COMPUTER TO PRINT OUT THE NUMERICAL
                C     VALUES THAT CORRESPOND TO THE VARIABLE NAMES.
 0008                 WRITE (3, 5) IQ
 0009                 WRITE (3, 43) JATE, KSYSTE
 0010                 WRITE (3, 705) LA394, M5Y3AA, JATE
 0011                 WRITE (3, 8326) NET, M5Y3AA, JDEF23, IQ
 0012                 WRITE (3, 54983) JDEF23, JATE, LA394, NET, KSYSTE
                C     THE FORMAT SPECIFICATIONS BELOW ARE I SPECIFICATIONS.
 0013              5 FORMAT (1X, I5)
 0014             43 FORMAT ('0', I5, I8)
 0015            705 FORMAT ('-', I6, 2I10)
 0016           8326 FORMAT ('1', 2I7, 2I9)
 0017          54983 FORMAT ('+', 38X, 5I9)
 0018                 END

 10

-21       3


-378          2        -21

-21       2      -503        10             -503       -21      -378       -21          3
```

1200120012001200120012001200120012001200120012001200120012001200120012001200120012001200120012001200120012001200120012

Section 12
DEFINING A NEW VARIABLE (IPN 2 and IPN 3)

1200120012001200120012001200120012001200120012001200120012001200120012001200120012001200120012001200120012001200120012

A common way to define a new variable is by means of an equation, where the new variable is to the left of the equation. In the equation

ANGLE 3 = X + 3.

the new variable is ANGLE3, and the variable X had to be previously defined or introduced. Note that all three terms are real.

In the equation

INDEX = J + M45 − KX7

the new variable is INDEX, and the variables J, M45, and KX7 had to be previously defined or introduced. Note that all four terms are integers.

Illustrative Program Number 2 contains the statement A = 10. Since this is the first time that the variable A is presented, and since A is to the left of the equals sign, a new variable is defined. This programming statement specifies that the real variable A is equal to the real number 10.

Likewise, BATE is defined as −20.6378, SYSTEM is defined as 3.5, XA394 is defined as 378.287, and Z5Y3AA is defined as −0.583. RESULT, however, is defined

in terms of the variables A and BATE, which have previously been defined. The variable DEF23 is defined in terms of the variable XA394 (previously defined) and the real number 125.39.

Illustrative Program Number 3 defines the integer variables IQ, JATE, KSYSTE, LA394, M5Y3AA, NET, and JDEF23.

1300130013001300130013001300130013001300130013001300130013001300130013001300130013001300130013001300130013

Section 13
WRITE STATEMENTS (IPN 2)

1300130013001300130013001300130013001300130013001300130013001300130013001300130013001300130013001300130013

A WRITE statement is for output purposes. It consists of the word "WRITE" followed by a pair of parentheses containing two numbers separated by a comma. The first of the two numbers identifies the device or machine to be used for the output. The second number identifies the FORMAT statement to be followed for the output arrangement. This will be discussed later (Section 15).

The following statement is an example:

```
WRITE(3,47)X,SUM1,J,JSUM1
```

This statement tells the computer to write numerical values for the real variables X and SUM1 and numerical values for the integer variables J and JSUM1. The above values are to be forthcoming on device 3 (in this text device 3 will be a printer), according to the FORMAT statement number 47.

Illustrative Program Number 2 has five WRITE statements. In all instances, device number 3 identifies a printer as the output device. In the five WRITE statements the FORMAT statements to be used are, in order, 1, 13, 407, 9638, and 36275.

Following the right-hand parenthesis, in each instance, is the name of one or more variables. The computer is instructed to print the numerical value that corresponds to each variable name.

1400140014001400140014001400140014001400140014001400140014001400140014001400140014001400140014001400140014

Section 14
STATEMENT NUMBERS (IPN 2)

1400140014001400140014001400140014001400140014001400140014001400140014001400140014001400140014001400140014

A statement is numbered when it is referred to elsewhere in the program. This number is arbitrary and can be in any of the columns 1 through 5.

In Illustrative Program Number 2, five statements have been numbered as follows: 1, 13, 407, 9638, and 36275. All these statements have been previously referred to in the WRITE statements.

To the left of column 1 appears a series of four-digit numbers, such as 0001 and 0002. These numbers identify selected lines of the printout and are listed by the computer. These numbers are automatically assigned by the computer.

1500150015001500150015001500150015001500150015001500150015001500150015001500150015001500150015001500150015001500150015

Section 15
FORMAT STATEMENTS (IPN 2)

1500150015001500150015001500150015001500150015001500150015001500150015001500150015001500150015001500150015001500150015

A FORMAT statement is used to establish the detailed arrangement of input or output information or data. Such a statement consists of a statement number and the word "FORMAT." This is followed by a left parenthesis, carriage control, FORMAT specifications, and a right parenthesis, as shown below.

```
21 FORMAT('-',F8.2,F12.3)
```

The carriage control '–' is discussed in Section 16, and the two FORMAT specifications, F8.2 and F12.3, are discussed in Section 17.

1600160016001600160016001600160016001600160016001600160016001600160016001600160016001600160016001600160016001600160016

Section 16
CARRIAGE CONTROL (IPN 2)

1600160016001600160016001600160016001600160016001600160016001600160016001600160016001600160016001600160016001600160016

In a WRITE statement, the first column of the output is for carriage control, i.e., the amount of advance of the output paper. The printer (device number 3 in this text) is basically a carriage roll (platen) that feeds the printout paper during the printing process. The amount of advance of the paper is determined by the carriage control.

One line of output information contains 133 columns internally and 132 columns externally (columns 2 through 133). Column 1 is internal only and is for the carriage control only.

The designation for carriage control appears in FORMAT statements as follows:

a blank = advance one line and print
0 (zero) = advance two lines and print
– (minus) = advance three lines and print
1 (one) = advance to the top of the next page of output and print
+ (plus) = advance zero lines and print

Let us analyze the carriage control of the five FORMAT statements of Illustrative Program Number 2.

The carriage control of statement 1 is 1X followed by a comma. The 1X tells the computer to leave the first column of the output blank. This, in turn, instructs the carriage control to advance the printout paper one line and then print.

The carriage control of statement 13 is '0' followed by a comma. This tells the computer to put a zero, *internally*, in the first column of the output. This, in turn, instructs the carriage control to advance the printout paper two lines and print.

The carriage control of statement 407 is '–' followed by a comma. This tells the computer to put a minus sign, *internally*, in the first column of the output. This, in turn, instructs the carriage control to advance the printout paper three lines and print.