# Argonne Aational Laboratory AN INTRODUCTION TO 704 FORTRAN

by

G. S. Pawlicki

RETURN TO REFERENCE FILE TECHNICAL PUBLICATIONS DEPARTMENT

#### LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

- A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or
- B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

ANL-6542 (Corrected)
Mathematics and
Computers
(TID-4500, 17th Ed.)
AEC Research and
Development Report

ARGONNE NATIONAL LABORATORY 9700 South Cass Avenue Argonne, Illinois 60440

#### AN INTRODUCTION TO 704 FORTRAN

by

G. S. Pawlicki

Institute of Nuclear Science and Engineering

March 1962

(Corrected November 1963)

Operated by The University of Chicago under
Contract W-31-109-eng-38
with the
U. S. Atomic Energy Commission

#### AN INTRODUCTION TO 704 FORTRAN

by

#### G. S. Pawlicki

#### ABSTRACT

It is hoped that this paper will permit the reader to begin to program computations on the 704 computer without referring to the Fortran Manual. Only a fraction of the possibilities of Fortran are described, but at all times an effort has been made to indicate where only a partial expose has been made. Many specific examples are included to illustrate the word descriptions of Fortran programming. No information is supplied concerning operation of the 704 control console and accessories. These functions are performed by personnel of the Applied Mathematics Division. The main body of the paper is devoted to Fortran programming, and an appendix, which gives specific information concerning procedures of the Applied Mathematics Division relative to the use of the 704, is included.

FORTRAN is an abbreviation of the name "Formula Translator." It is called a formula translator, because formulas written almost like ordinary algebra are read from a hand-punched set of IBM cards by the 704 acting as a translator, and a set of cards is punched which serve to operate the 704 as a computer. The set of hand-punched cards is called the "Source Deck." The source deck is translated by the computer, which punches a translated deck of cards called the "Object Deck." The process of preparing an object deck from the source deck is called "Compiling." At "Object Time," that is, when a problem is being solved, the object deck is placed in the card reader, possibly followed by hand-punched "Data Cards" which supply numerical data required by the program. Data can also be read from magnetic tape, but this will not be considered.

The IBM cards used in the source deck and for data have a row of printed numbers from 1 to 80 which identify the 80 columns into which rectangular holes can be punched. The holes, which are punched in a single column, are coded to represent a unique character. The characters are the symbols - +, () \*./ as well as numerals and capital letters. At the top of each column is a space in which the card-punching machine also prints the character corresponding to what has been punched. Columns 73 to 80 inclusive are not read by the computer and can therefore be punched with identification such as numerical sequence of source cards and identification of cards as data and their sequence.

AG

## G. S. Pawiicki

## ABSTRACT

It is noped that this paper will pormit the reader to begin to program combutations on the 70% computer without releving to the Portram Manual. Only a fraction of the passibilities of Bortram and described but at all times an effort has been made to indicate where only a partial expose has been made than appoint on a partial expose has been made than appoint on Fortram programming. No trate the word descriptions of Fortram programming. No mior mation is supplied concerning operation of the 70% control console and screeces us. These functions are performed by personnel of the Applied Malhematics Daylors. The main body of the penetria devoted to Fortram programming, and an appendix, which gives specific information concerning an appendix, which gives specific information relative to the 70% is included.

Ever called a formula translator, because for multip written abrost the problem of the called a formula translator, because for multip written abrost the problem of greatery algebra are read from a pand-pulment set of 18M ared, by the 704 acting as a consister, and a set of pand-superhad which serve to person the 704 acting as a commune. The set of hand-superhad or desire is called in "Suger Dack." The approve decid is translated by the computer, which computer are sailed in "Object Dack." The process of preparate and acting the consister of the source deck is called. "Object Dack." The process of preparate and the object deck is placed to that is, when a graphen as length solved the object deck is placed to the consister. On a can also be which supply minner tend nata required by the program. On a can also be read from magnetic tage, but this will not be considered.

The IBM opril, used in the source deck and for data have a row of princed numbers (from 1 to 80 volume in a 80 golumns into which feet tangular holes can be bunched. The roles value are punched in a single column, are coded to represent a window character. The characters are the symbols - . ( ) \*... as well a monorals and repiral latters. At the tup of each column is a space in a light are card punching machide size principle the character corresponding to what has been punched. Columns 73 to 80 includive are not read by the non puter and can increase be punched with identification such as quagarical sequence of source cards and identification such as quagarical sequence of source cards and identification such as quagarical sequence of source cards and identifications.

To describe the use of columns 1 through 72, probably it is best first to discuss the form of numbers which can be entered as data. Numbers can be entered as floating-point decimal numbers in normal or exponential representation, integers, and octal numbers. Only decimal and integer representation will now be considered. The normal decimal representation of a number will henceforth be referred to as "Floating Point." An integer represented without a decimal point will be called "Fixed Point." Any number is represented by punched holes in a certain series of columns on the IBM cards. The number of columns assigned to a specific number is called the Field Width.

The 72 columns available for numerical input of data can be divided into any desired number of fields of any width desired. The source deck must contain a <u>FORMAT</u> statement which instructs the computer in which way it should interpret the data card. The specification in the FORMAT statement for the floating-point format is

#### n Fw · d

where n is the number of identical fields to be read (the first field begins with column 1!); F is the code letter for floating-point representation; w is the field width; and d is the number of places to the right of the decimal point.

The specification of a fixed-point number in a FORMAT statement is of the form

#### nIw

where n and w are as previously defined and I is the code letter for fixed-point representation. In punching a data card with a number, the sign precedes the number. A floating-point number can be punched anywhere within the field. With a fixed-point number the number must be at the right side of the field; otherwise, the blank spaces at the right end of the field, read by the computer as zeros, would alter the number by powers of ten. In both representations, if the numerical sign is missing, the computer assumes that the number is positive.

Only single-line format for numerical quantities will be described. The source card for a <u>FORMAT statement</u> might be punched as follows:

Columns 1-5

A Statement Number located anywhere in this field, an integral number without sign.

Column 6

Blank

Column 7 onward to 72

might be punched as follows:

FORMAT (3 F 6.2, F 5.4, I2, 3I4).

This FORMAT statement <u>could</u> be used to specify reading of a data card. The data card would have 3 fields of 6 columns, a field of 5 columns, a field of 2 columns, and 3 fields of 4 columns. The spelling, commas, and parentheses are essential parts of the statement and in all other types of statements to be described.

If a Fortran statement requires more space than the columns from 7 to 72, a statement can be continued on a second card by punching column 6. Up to 9 continuation cards can be used. With the exception of zero, what is punched in column 6 is immaterial and can even be the same on all continuation cards in the entire program, since the computer merely reads the cards in the order in which they are stacked in the source deck. Only one statement can be punched on a single card. The statement itself need not start in column 7, and any number of blanks can be inserted anywhere in a statement. The statement number is an integral number without a sign and can be located anywhere in the columns from 2 to 5. Statement numbers uniquely identify statements and need not be in any numerical sequence.

Numerical quantities in FORTRAN can be numerals or identified with a letter name. A floating-point constant has a decimal point whereas a fixed-point constant must not. The literal expression for a number can consist of a series of not more than 6 characters, which can be both letters and numerals with certain restrictions. The first character must be a letter. A fixed-point number name must begin with letters I, J, K, L, M, or N, and floating-point number names must not begin with these letters. A good rule is to avoid the use of F as the last character to prevent possible confusion with names of functions which are discussed on page 6.

Both floating- and fixed-point variables can be subscripted with up to 3 subscripts. The subscripts must necessarily be fixed-point numbers without sign. Subscripts cannot take on zero or negative values. The following examples completely cover limitations of naming and arithmetic operations for subscripts:

algebraic	FORTRAN					
aj,k,l	A(J,K,L)					
xa	X(IA)					
c <sub>i+2,m</sub>	C (I + 2, M)					
ij,m	I(J,M)					
Pzi-1	P(2*I-1)					

As previously mentioned, an equation can be punched in a form which looks very much like ordinary algebra. The algebraic symbols used are +, -, \*, \*\*, /, (), and =. These symbols represent addition, subtraction, multiplication, exponentiation, division, parentheses, and equality. As an example, the algebraic expression

Tata FORLAGE TARGET statement round in seed to specify reading a fair of a fatures a field of a fatures a field of a fatures a field of a fatures as a field of a fatures of the statement and in all other areas are essential part of the statement and in all other areas are essential part of the statement and in all other areas are the bed described.

If a Tostess election regulars, more a pace than the entirement of to 12, a startement and to continue the continued on a second care by processing entirement of the formal can be used. With the exception of send what is supplied in column to the immediated and can be used to be the tame on all continued and in the column to the immediated and can be the tame of all continued at the continued in the order in while letters and the continued to the source doubt Locky one statement controls marked on a studyle card. The statement controls and the column to the instancent. The statement of the integral number with any that, is attached. The statement of the columns to the integral number while a sign and statement, the statement is the columns toom 2 to 5. Statement and the description of the columns from 2 to 5. Statement and the columns and assigned the columns and the columns to the cast of the columns and the columns and the cast of the cast of the columns and the cast of the

Tail linking and fived-paint variables the adapting white we was supposed white up to a color of the adapting a suppose the adapting the adapting values. The first without at it, defectly sometime of action of carrier values and mailting the first of carrier without or adapting the adapting the complete completely forest invitations of carrier and available the completely forest invitations of carrier and carrier

ALLE CONTRANT OF THE STATE OF T

As previously magilement an equation can be purched in a form work last a week much lake ordinary algebra. The elgebraic symbols used as to the second much last a consequence of the co

$$y = (ax^2 + bx + c)^3$$

could be written as a FORTRAN Arithmetic Statement as follows

$$Y = (A*X**2 + B*X + C)**3$$

The <u>mode</u> of all terms (i.e., fixed or floating point) on the right side of the equal sign must be the same! An exception is that an exponent of a floating-point number can and should be fixed point if it is an integral number. The mode of the left side need not be the same as the right side. When the right-side mode is different from the left-side mode, the statement implies a conversion of the right-side quantity into the mode specified by the left side. In the example above, if the left side were I, I would be equal to Y truncated, to an integer.

In writing a FORTRAN arithmetic statement, the variables, constants, and equal symbol can frequently be thought to have the same meaning they have in algebra. Actually, the variables in FORTRAN are specifications of memory locations rather than the numbers themselves. The quantity computed by using the contents of memory locations specified on the right of the equal sign are to be stored in the location specified on the left of the equal sign. Thus the equal sign really has the meaning "store." This is best illustrated with the statement

$$A = A + B$$

Algebraically this only holds true for B = 0; however, in FORTRAN the results of the summing of the contents of memory locations A and B are to be stored in memory location A.

The use of parentheses in FORTRAN is very much like their use in algebra; that is, after encountering a left parenthesis, the computer will perform the indicated algebraic operations up to the right parenthesis, independently of any further algebraic operations outside of the parentheses. Thus, in the previous statement for Y, the quantity in the parentheses is computed and then raised to the third power.

In the absence of parentheses or within parentheses the order in which the algebraic operations are performed is exponentiation, multiplication or division, and finally addition or subtraction. Thus BB = A\*B\*\*C + X algebraically means the single variable called BB equals  $(a \cdot b^c) + x$ .

Parentheses are frequently necessary to write an arithmetic statement properly, and when in doubt it is probably wise to use them even if they are not absolutely necessary. To illustrate some cases in which parentheses are necessary, consider the following algebraic expression and their FORTRAN representation: could be written as a FORTSAN Artifement Statement as follows

# 14 (5 + 50 g + 50 % A) = 7

The freely of all terms (de. fixed or Heating point) on his right side of the equal sign number the same. An exception is that an exponence of a flatting-point number can a si should be fixed point in an integral number. The mode of the last side that side need dot be the same as the right alist. When therefore side under a different from the latterness that the latterness is not be mode aparticled by the left side. The terms of the right edge quantity into the mode aparticled by the left side. In the example above, if the last side were it fixed as a great of the side were it the side were it to be side of the side.

In writing a structure and anticometric statement, the variables, constants and equal symbol can frequently be thought to have the same metalog they have the same angebra. Sometimes to algorial sortially, the variables in FORTHAN are spiriting order or marriary loc from a struct than the unminers harmselves. The quantity compared on the contents of the spiriting on the fight of the putter of the structure of the contents of the spiriting of the spiriting of the spiriting spiriting of the spiriting structure of the spiriting spiriting of the spiriting spiriting of the spiriting spiriting the spiriting of the spiriting of the spiriting spir

# S S A S A

Atabraically this only holds true for B = 0; however, in FORTHAM the rain of the summing of the contents of memory locations A and B are to be stored in memory over locations.

The use distributed of the control o

In the absence of perentiases or within parentiases the order in which the affection or the affection of the affection of the affection of the affection of substantian (and B) - A\*B\*C - X algorithm monor the virgin was table (allow DB equals (a bb) + x

Fa cultience are Inequally necessary to write an arithmetic state-

ment properly, and when in donor it is prohibly steer to use them even if they are not absolutely negetian. To illustrate some cases in which perentificates are necessary, consider the inflowing alcubrate expression and their FORLIEAN representations.

$$a(b + c)$$
  $A ** (B + C)$   
 $a(b \cdot c)$   $A ** (B * C)$   
 $(a \cdot b)^{c}$   $(A * B) ** C$ 

An expression like

could have been written with extraneous parentheses as  $(A^{**B})^*C$ . Whenever two of the characters +, -, /, \*, \*\*, appear consecutively, it is necessary to separate them with parentheses. Consider the following relationship which would not be ambiguous in algebra:

$$y = x \cdot -2. - + 4.6$$
;

this would have to be written in FORTRAN as

An example of an expression containing a subscripted variable might be

$$y = a_i^2 + b_j^m$$
  $Y = A(I)**2 + B(J)**M$ 

where the parentheses arise because of the FORTRAN subscripting notation.

FORTRAN has 5 types of functions, of which only 3 will be considered. These 3 are open (or built in) functions, closed (or library) functions, and arithmetic statement functions. These 3 functions have the same rules for their naming and for referring to them (calling).

The names of these 3 types of functions consist of 4 to 7 alphabetic or numeric characters of which the first must be alphabetic and the last is F. A function can have one or more arguments, but the function is single-valued. The mode of the arguments may be fixed or floating, and the mode of the function value can be either fixed or floating. If the value of the function is fixed point, the function name must begin with X. It is desirable to name a function with a sequence of letters which suggest its meaning.

To illustrate the naming of functions, we can consider the algebraic statement for y previously used:

$$y = (ax^2 + bx + c)^3$$

and define an Arithmetic Statement Function. The statement to be punched could be

$$YFUNF(X) = (A*X**2 + B*X + C)**3$$

An expression like

could have been written with extraneous parentheses as (A\*\*B)\*C. Whenever two of the characters +, -, /, \* \*\*, appear conscentively, it is necessary to separate them with parentheses. Consider the following relationable which would not be ambiguous in algebra:

this would have to be written in FORTRAM as

An example of an expression containing a subscripted variable might be

where the parentheses arise because of the FORTRAN subscripting petation.

FORTRAN has 5 types of functions, of which only 3 will be considered These 3 are open (or built in) functions, closed (or library) functions, and carifformile statement functions. These 3 functions have the same rules for their naming and for referring to them (calling)

The names of these 3 types of functions consist of 4 to , algorithm or numeric characters of which the livet must be alphabetic and the last as F. A function can have one or more arguments, but the function is single valued. The mode of the arguments may be fixed or floating and the runde of the function value can be either fixed or floating. It the value of the function is fixed point, the function active manual larger with X. It is desirable to name a function with x sequence of letters which suggest its meaning.

To illustrate the naming of functions, we can consider the algebraic statement for y previously used;

and define an Arithmetic Statement Function. The statement to be purched

The name YFUNF is followed by its argument X in parentheses. YFUNF has a value which is floating point (no X at start of name). Its argument on the left side clearly shows that the argument is floating point and, we note that to the right side of the equal sign the statement is correctly written in the same mode. As defined above, YFUNF has the argument X and has A, B, and C as parameters. If the left side of the statement had been written YFUNF(X,A,B,C), the function would have 4 arguments.

To have the argument floating and the function value fixed point, the statement would be

$$XYFUNF(X) = (A*X**2 + B*X + C)**3$$

If we wished to have a fixed-point argument and floating-point function, we might write

YFUN2F (I) = 
$$(JA*I**2 + JB*I + JC)**3$$

Note that the absence of X at the beginning of the name means that the value of the function is floating point. The argument in parentheses is fixed point, and to the right of the equal sign the arithmetic manipulations have the fixed-point variable I and the 3 fixed-point coefficient JA, JB, and JC to give a fixed-point-mode arithmetic expression consistent with the mode of the argument of the function in parentheses on the left side of the = sign.

The fourth possibility is to have fixed-point argument and fixed-point value for the function:

$$XYFUN2F(I) = (JA*I**2 + JB*I + JC)**3$$

In FORTRAN as in ordinary algebra we can put in any letters or numbers for the argument of a function, and so if the 4 foregoing functions are defined in a program, they may later be called by arithmetic statements as follows:

ALPHA = 2.5 + YFUNF(7.2)

BETA = 6.3 + YFUNF(Y(K))

GAMMA = 5.75 + YFUN2F(J)

DELTA = 20.5 + YFUN2F(31)

IOTA1 = 20 + XYFUNF(2.6)

IOTA2 = 25 + XYFUNF(Z)

LAMDA1 = 31 + XYFUN2F(IT)

LAMDA2 = 436 + XYFUN2F(39)

The name YFUNE is initiowed by its argument X in paraordesian. YFUNE has a value which is flusting point (no X at start of dame). Its organisation the left side clearly shows that the argument inclosuring point and, we had that to the right side of the equal sign his statement is correctly written in the same mode. As defined allow, YFUNE are the argument X and has A. B. and C as parameters. If his left side of the statement had been written YFUNE X.A.B. Ch. the Digitles would have 4 arguments.

To have the argument lighting and the function value fixed point, the

# XY FUNT (X) = (A\*X\*\*Z\* & 8\*X + C) TENTY X

If we wished to have a fixed-point argument and floading-point Micro

# TYUNEF (0) = (LA\*1\*\*2 \* 38\*1 \* 30)\*\*\*

Note that the absence of X at the beginning of the dama means that the follow of the function is floating point. The arregiment is parenthesen to fixed some and to the eligible of the end of the end

The fourth passibility is to have fixed-point argument and fixed-point argument and fixed point

# ET OUT THE HIS THAT ALL IN THE WAY

of n FOR TRAIL as in ordinary algebra we can put in any lotters of countries of interest for the argument of a lunction, and so if it is a loregoing interpretation are defined in a program; they may large be called by a ripment atatement as a tollows:

ALPHA = 2.5 + YFUNF (7.2)

BETA = 0.1 + YFUNF (Y(X))

GAMAKA = 5.25 + YFUNSE(J)

DELIA = 20.5 + YFUNSE(J)

GOTAL = 20.5 \* XYFUNE(Z, 6)

LANDAL = 5.1 + XYFUNZE(LT)

LANDAL = 5.1 + XYFUNZE(LT)

The <u>built-in</u> and <u>library</u> functions are defined within the 704 and, just as the foregoing functions were utilized in arithmetic statements, so library functions can be used by appropriately indicating the function name and its arguments. Seven library functions with floating-point values and arguments are

LOGF	natural log of the magnitude of the argument
SINF	sine (angle in radians)
COSF	cosine (angle in radians)
EXPF	exponential function
SQRTF	Plus square root of the magnitude of the argument
ATANF	$-\pi/2$ < arctangent in radians < $\pi/2$
TANHF	hyperbolic tangent

Table I, taken from the FORTRAN manual, lists the built-in functions in the 704 FORTRAN, and their use is probably self-explanatory.

Table I

				Mode of				
Type of Function	Definition	Number of Args	Name	Argument	Function			
Absolute value	Arg	1	ABSF XABSF	Floating Fixed	Floating Fixed			
Truncation	Sign of Arg times largest integer≤ Arg	1	INTE	Floating Floating	Floating Fixed			
Remaindering (see note below)	Arg <sub>1</sub> (mod Arg <sub>2</sub> )	2	MODF 20MODF	Floating Fixed	Floating Fixed			
Choosing largest value	Max (Arg <sub>1</sub> , Arg <sub>2</sub> ,)	≥2	MAXOF MAXOF XMAXOF XMAXLF	Fixed Floating Fixed Floating	Floating Floating Fixed Fixed			
Choosing smallest value	Min (Arg <sub>1</sub> , Arg <sub>2</sub> ,)	≥ 2	MINOF MINOF XMINOF XMINOF	Fixed Floating Fixed Floating	Floating Floating Fixed Fixed			
Float	Floating a fixed number	1	FLOATF	Fixed	Floating			
Fix	Same as XINTF	1	XFIXF	Floating	Fixed			
Transfer of sign	Sign of Arg <sub>2</sub> times   Arg <sub>1</sub>	2	SIGNF XSIGNF	Floating Fixed	Floating Fixed			
Positive difference	Arg <sub>1</sub> , -Min (Arg <sub>1</sub> , Arg <sub>2</sub> )	2	DIMF	Floating Fixed	Floating Fixed			

NOTE: The function MODF (Arg<sub>1</sub>, Arg<sub>2</sub>) is defined as Arg<sub>1</sub> - [Arg<sub>1</sub>/Arg<sub>2</sub>] Arg<sub>2</sub>, where [x] = integral part of x.

Reproduced by permission from IBM Reference Manual - 704 FORTRAN Programming System, @ 1958, 1959, 1961 by International Business Machines Corp.

The built-in and library tractions are coloosed within the and real as the library functions were unit in sufficient evaluations were unit real in sufficient evaluation when and its arguments. So the library functions with the time-point relates and new mones are

LOGE natural log of the magnified of the argument
live and lande to radiate;
COSE cosine Write in radiate)
EXPE expendential function

thomogra out to obvinged out to forward and a THOS.

TANE | - m / d / arctangord in radiang / m / L

Table 1, taken from the FORTKAN manual, that the builten functions in the

Mark the thought and Arch and Mark the transfer of the control of

Reproductive to purel view from 19th delevel, whereat 704 FOR. A.A. Programming System, Quickly 1955; 1961 by Journal Systems Maother forces In summary with regard to the use of the 3 classes of functions described, the part of a statement which calls the function consists of the function name followed by a <u>list</u> of the <u>arguments</u> separated by <u>commas</u> and in parentheses, as for instance,

# SOMEF (X, Y, Z(I))

Whenever a variable is subscripted, it is necessary to assign a sufficient number of memory locations for the array. The assignment of memory locations is done with a DIMENSION statement.

For instance, if in a problem we have the subscripted variables A(I), B(I,J), and C(I,J,K), which would be 1-, 2-, and 3-dimensional arrays, respectively, where I takes on values from 1 to 5, J values from 1 to 4, and K values from 1 to 3, then 5 memory locations are required for A(I), 20 locations for B(I,J), and 60 locations for C(I,J,K). The DIMENSION statement for this specific example would be

# DIMENSION A(5), B(5,4), C(5,4,3)

In the source deck, the DIMENSION statement must appear before the first statement which refers to the subscripted variable.

A simple problem will now illustrate the use of the statements which have been described and to introduce a few other statements. The problem will be to have the program read 5 data cards, each of which has a single value of an angle in radians, and to have the computer evaluate the sine, cosine, and tangent of these angles, and form a four-column tabulation consisting of the angle, its sine, cosine, and tangent.

Columns 1 to 5	Columns 7 to 72
1	DIMENSION A(5)
2	FORMAT (F10.5)
3	FORMAT (4F10.5)
4	READ 2, $(A(I), I = 1,5,1)$
5	DO 9 I = 1,5,1
6	SI = SINF (A(I))
7	CO = COSF (A(I))
8	TA = SI/CO
9	PRINT 3, A(I),SI,CO,TA
10	STOP
11	END (0,1,0,0,1)

In surprisery with regard to inc use of the Valence of functions deviserviced, the part of a statement which calls the function consists of the function mane followed by a 1.21 of the arguments segmented by commiss and un personal tests that the test of the arguments segmented by

# SOMEF (X, Y, Z(I))

Whenever a variable is subscripted, it is necessary to assign a sufficient number of momenty increase for the array. The assignment of membry location is done with a DIMENSION statement.

For instance: If he problem we have the subscripted variables Adh.

2(13), and CHIEL which would be 1-v2-vaid 1-dimensional sericy, vesses appeared. To head Received values from 1 to head Research to head Received from 1 to head Received from 1 to head Received from 1 to head Received for high Strange and to head to

# DIMENSION AGE BER ALL CONTROL

in the source fack, the DIMENSION statement must appear before the first statement which refers to the subscripted variable.

A sample problem will now illustrate the view of the statements which have been described and to introduce a few office Statements. The problem will be to have interpropriet read 5 data cards each of which has a single waite of an angle or radians and to have inc computer evaluate the rises of coordinate the rises and form a formation tabulation tapes and the single of the single that contact, and form a formation tapes.

Columns 1 to 5 Columns 11

FORMAT [FLD 1]

READ & (AG) 1 = 1,5,1)

Late 1000

SI - SIKE (A(I))

CO 1 COST (ALL)

PRINT S. A(I) M.CO.T.

STOP

O.O. I. b. chra

The statements have all been numbered for ease of description. though only statements 2, 3, and 9 require statement numbers for execution of this program. Statement I says that a "subscripted variable A, i.e., Angle, will require 5 memory locations." Statement 2 says; "a single floating-point number occupying a field of width 10 with 5 places to the right of the decimal point." This statement will be used by statement 4 to instruct the computer how to read data cards. Statement 3 says: "4 floatingpoint numbers, each occupying a field of width 10 with 5 places to the right of the decimal point." This statement is used by statement 9 to instruct the computer how to type the value of angle, sine, cosine, and tangent, Statement 4 has not been previously explained; it can be read "Read data cards according to the specification of statement 2 (i.e., FORMAT statement); what is being read defines the value of subscripted variable A(I) where the index I begins with 1, and runs to 5 in steps of 1." This read statement, at object time, will cause 5 data cards to be read because of the way the index I and FORMAT were specified. The numbers read are stored in the computer memory.

Statement 5 has not previously been explained. It can be read "do the instructions specified by the statements following up to and including statement 9, and then repeat this series of instructions as governed by the index I which is to take on the values 1 to 5 in unit steps." Thus in our case the DO statement specifies that statements 6 to 9 are to be executed 5 times; each time it repeats the index I is increased by unity. We can say the DO statement has defined a loop which is to be circled 5 times. The last number (1) which defines how I is incremented need not appear in statements 5 and 4 because its absence implies that it is unity. Within the DO loop the index number I is available for use in fixed-point arithmetic expressions. After the DO loop is satisfied, the index number I is no longer defined.

Statements 6 and 7 are simple arithmetic expressions which define floating-point variables in terms of the library functions SINF and COSF. Note that in naming the variables SI and CO, it is forbidden to name them SIN or COS, since a variable must never have a name which is that of a defined function without the terminal F.

Statement 8 is an Arithmetic Statement that TA the tangent is the sine divided by the cosine.

Statement 9 has not been previously explained; it can be read "Print on the output printer (typewriter) according to the specifications of FORMAT statement 3, the 4 quantities A(I), SI, CO, and TA."

If it is desirable to print out the 5A(I) values before entering the DO loop, a PRINT Statement could be placed between statements 4 and 5:

The statements have all been sambered for case of description, though only statements 2, 3, and 9 require statement mumbers for execution of this program. Statement I save that a "subscripted variable A. i.e.; Angle, will require 5 memory locations." Statement 2 says, "a single floating-point number occupying a field of width 10 with 5 places to the right of the decimal point." This statement will be used by statement 2 to instruct the computer how to read dita cards. Statement 3 says; "4 floating-point numbers, each occupying a field of yidils 10 with 5 places to the right of the decimal point." This statement is used by statement 9 to instruct the computer how to type the value of angle, sine, craine, and tangent Statement 4 has not been previously explained: it can be read "Read data cards according to the specification of statement 6 in . FORMAT statement) what is being read defined the value of subscripted variable A(1) means) what is being read defined the value of subscripted variable A(1) statement, at object time, with 1, and runs to 5 in steps of 1." This read are statement, at object time, will cause 5 data cards to be read because of the way the index I and FORMAT were specified. The numbers read are stored in the computer memory.

Statement 5 has not previously been explained. It can be read "do the instructions specified by the statements following up to and including statement 9, and then repeat this series of instructions as governed by the index I which is to take on the values i to 5 in unit steps. This in our case the DO statement specifies that statements 0 to 9 are up be executed 5 times: each time it repeats the index I is increased by unit. We can say the DO statement has defined a loop which is to be circled 5 times. The last number (1) which defines how I is incremented need not appear in statements 5 and 4 because its absonce unaited that it is unity. Within the DO toop the infex number I is available for use in fixed-point archanotic expressions. After the DO loop is satisfied, the index number I is no longer defined.

Statements b and 7 are simple artitude the expressions which define floating noist variables in terms of the library-functions SIMF and COSF. Note that in manifug the variables SI and CO, is a orbidden to name them SIM or COS, since a variable must never have a pame which is that of a defined function without the terminal F.

Statement 8 is an Arithmetic Statement that TA the tangent is the sine divided by the cosine.

Statement 9 has not been providually explained; it can be read "Print on the output printer (typewriter) according to the specifications of FORMAT statement 1, the 4 quantities A(1), SI, CO. and TA."

H it is desirable to print out the bA(I) values before entering the DO loop, a PRINT Statement could be placed between statements 4 and 5;

This would print the 5 values of A(I) on 5 successive lines. Note that the PRINT statement above and the READ statement 4 have the nature of a DO loop. Note that the list of variables in PRINT and READ statements requires parentheses only if the statements define an index and require looping.

Statement 10 will be executed after the DO loop is satisfied. Its effect at object time is to stop the computer.

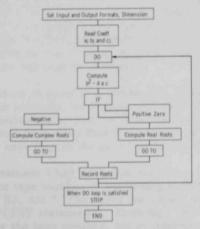
The <u>END Statement</u> is the physically last card in the source deck. It essentially separates independent programs whose source decks are put into the card reader at the same time during compiling. At the Argonne National Laboratory, the numbers 01001 are the normal END statement specifications. Note that the computer performs the program in the sequence of the source deck cards except for the change caused by statements such as DO.

FORTRAN has a number of different conditional branching statements, of which only one will be considered. The statement

can be read "If the variable or expression A is negative, skip in the program to statement 5; if it is zero, skip to statement 20; and if it is positive, skip to statement 30." Care must be exercised in testing floating-point expressions for zero because of truncation and round-off errors which are normally introduced into a calculation. Fixed-point arithmetic, on the other hand, is exact, and the skip on zero presents no difficulties.

The unconditional branch statement is GO TO.

The following problem will serve to illustrate the use of IF and GO TO statements. The problem is to factor 5 quadratic equations by means of the quadratic formula. The roots of the equations can be either real or complex. The block diagram of the program is as follows.



This would print the S values of AC on S arccessive lines. Note part the PRINT statement above and the RE ID statement staye the patter of a DO loop. Note that the list of variables in PRINT and READ statements requires parentheses only if the alatements define an index and require toogless.

Statement 10 will be executed after the DO loop is satisfied. Its.

The END Statement is the physically last dard in the source docks.

It essentially separates independent programs alone source docks are put this excit reader at 12s same time daring comming. At the Argenne National Laboratory, the gumbers 1900s, are the administ END statement specifications. Note that the computer performs the program is the sequence of the source deck cards except for the change caused by statements afth as 100.

FORTRAN has a number of different conditional branching statements of winds only one will be considered. The statement

## 05.08.8 (A) TI

can be read "If the variable or expression A is negative, ship in the programs to statement up it it is vero, who to statement 30,4 and it is positive; ship to statement 30.1" Care must be exercised in testing flusting-point expressions for zero because of truncation and coined-off errors which are sornally up treadment into a calculation. Pixed-point arithmetic, on the other hand, is exact, and the skip on zero presents to difficulties.

# The unconditional branch statement is CO TO.

The following problem will serve to illustrate the use of IF and CO.

TO enterments. The problem as to factor a conferrite quations by means of the quatrons can be sides read or court the quatrons can be sides read or court plan. The block diagram of the program is as belows.



```
DIMENSION A(5), B(5), C(5)
       FORMAT (3 F 10.5)
       FORMAT (4 F 10.5)
       READ 5, (A(I), B(I), C(I), I = 1.5)
           DO 9 I = 1.5
                 D = B(I)**2 - 4.*A(I)*C(I)
                     IF(D)10.4.4
                R1 = -B(I)/(2.*A(I))
10
                R2 = R1
               AI1 = SQRTF(D)/(2.*A(I))
               AI2 = -AI1
                        GO TO 9
               R1 = (-B(I) + SQRTF(D))/(2.*A(I))
               R2 = (-B(I) - SQRTF(D))/(2.*A(I))
               AI1 = 0.
               AI2 = 0.
               WRITE OUTPUT TAPE 2, 6,
                                            R1, AI1, R2, AI2
                    END FILE 2
                    STOP
                    END (0.1.0.0.1)
```

In the foregoing program, the IF statement tests if the discriminant  $b^2$  - 4ac is negative, zero, or positive. If it is negative, the program beginning with statement 10 is followed, since the roots are complex. After the complex roots are found, the GO TO statement tells the computer to execute statement #9, which records the roots on magnetic tapes, and then the program continues until the DO loop is satisfied.

If the discriminant is zero or positive, the roots are real, and the IF statement says that the program for finding the real roots beings with statement #4. If the discriminant was zero or positive, statement 9 follows immediately after the roots have been determined, and no GO TO statement is necessary to have the next step in the program be the recording of the roots.

The output statement 9 has not previously been explained. It can be read: "Write on output tape number 2, according to FORMAT statement 6, the numbers R1, AI1, R2, AI2." The rules for listing the output quantities are the same as for PRINT statements. Each tape unit has a ten-position switch which identifies the tape number, but at Argonne, tape number 1 is

DIMENSION A(3), B(3), C(4)

FORMAT (3 F 10.5)

BORMAT (4 F 10.6

HBAD S. (A(1), B(1), C(1), I = 1,5)

DO 9 1 = 1, 5

D = B(I)\*\*2 - 4.\*A(I)\*G(I)

4.01(0)TL

((1)A\*,S)\((1)E\) = "III

131 = 52

ATT = SORTE (D)/(2,\*A(I))

TA+ = SIA

coros.

RI = (-B(I) + SQRTF(D)) (2.\*A(I))

R2 = (=B(I) - SQRUF(D))/(2.\*A(I))

I ILA

O - SIA

WRITE CUTFUT TAPE 2, 5, BL, ALL, R2, ALZ

END FILE

TO THE

1,0,0,1,0) CME

In the inregions program, the IF statement tests if the descriminant be + fac to negative, serio, or positive. If it is negative, the program beginning with statement 10 is followed, since the route are complex. After the complex roots are found, the GO TO statement of the computer to be computed attenuent of, which records the roots on magnetic tapes, and then the program continues until the DO toop is satisfied:

If the discriminant is zero or positive, the roots are real, and the IF statement says that the program for finding the real roots beings with slaterment Ms. If the discriminant was zero or positive, statement 9 follows time mediately after the roots have been determined, and no QO TO statement is necessary to have the next step in the program be the recording or the roots.

The output statement 9 has not proviously been appliched. It may be read: "Write on output tape muraber 2. Activities the PORMAT statement of the numbers R1, All 32, Al2." The rules for listing the output quantities are the same as for PRINT statements, Each tape unit has a temposition switch which identifies the tape number, but at Argones, tape number 1 is

not to be used. Since magnetic tape recording is much faster than the "On line" printing (on the Printer directly associated with the 704), the time charges on the 704 are reduced. The recorded output tape is later read and printed by an "Off line" printer for which there is no additional charge at the Argonne computer center.

The <u>END FILE</u> statement marks the output tape number 2 after all the output has been recorded. The END FILE mark on the tape is used to stop the off-line printer. At the Argonne computer, the convention is not to include a REWIND statement for the output tape, so that the computer operator can record the output from several problems on the same reel of tape.

Thus far, a major part of the description has been concerned with introducing various types of statements and describing the form in which they are written; some further mention must be made of restrictions.

Of the statements described, the FORMAT and DIMENSION statements are called non-executable statements, that is, they do not cause an operation to be performed. The first statement after a DO statement must not be a non-executable statement. The last statement of a DO loop must not be a statement which transfers control, as does GO TO. When a control transfer statement is the last executable statement in a DO loop, the non-executable dummy statement CONTINUE is made the last statement of the DO loop.

The successive statements between the DO statement and the last statement of the DO loop define the range of the loop. Frequently, other DO statements occur within a DO loop. When this occurs, the range of the earlier DO statement encountered must extend up to or beyond the range of any DO loops defined within the range of the first DO loop. This says that DO statements must be <u>nested</u>; there cannot be just a partial overlap of ranges of DO loops.

Transfer into a DO loop from statements outside of the range of a DO loop can only be made to the DO statement itself, and not to any other statement in the range of the DO loop. With a single DO loop, transfers are permitted to statements within the range of the loop. If there are nested DO statements in a program, any transfers to the inner DO statements can only be to the DO statements itself so as not to violate the rule in regard to transferring stated above. The calling of any function or subroutine involves a transfer out and then back into a program; this sort of transfer is permitted within the range of a DO loop provided the subroutine returns to the same part of the loop and that the subroutine does not alter the value of indices. It should also be emphasized that no statement within a DO loop must redefine the indices. It should be noted that all restrictions apply to entering DO loops and not to exiting.

uch to be used. Since magnetic tape recording is much faster than the sing time" printing (on the Printing directly associated with the 104), the time charges on the 104 are reduced. The recorded output tage is later read and printed by an "QQI lime" printer the which there is no middisonal charge as the frequency communic center.

The END FILE statement district the countries on the tage is used to the the this best recorded. The Elvis pile mark on the tage is used to stop the off-line printer. At the Argonic computers the convention as to inclode a file the computer the computer of the computer operator, can record the owner from several problems on the simulated tage.

Then far a major part of the description has seen concerned with introducing various types of statements and describing the torm in what they are written; some further investion must be made of to statellons.

Of the statements described the EORMAT and DIMENSION states ments are called for executable statements, that is, they no not cause an operation to be performed. The flare statement after a DO semicent must not be a non-executable statement. The last statement of a DO loop must not be a statement which transfers, control, as loop GO TO. When a control transfer statement is the last processed at statement of the last processed at made like last attempts of the DO loop. The most executable dummy statement OOTTINUE is made the last statement of the DO loop.

The successive emissions between the hope of the loop, Trequently office, statement of the DO loop of the two rance at the loop, Trequently office, DO statements occur within a DO loop. When the occurs, the chape of the carrier of a server of the chape of the carrier of any DO statement encountered must extend the the the box beyond the range of the three the post of the server of the DO statement must be made to the carrier be just a partial outline of ranges of DO loops.

Transfer into a DO loss from statements contained of the range of the statement in the range of the DO loop. With a single DO loop, and not to any other statement in the range of the DO loop. With a single DO loop, franklers are permitted to attainments within the range of the loop. If there are occided not permitted to attainments within the range of the loop. If there are occided to only the loop of the loop in the range of a loop transfer the substantian and the range of a loop loop and the range of a loop loop and the substantial of the range of a loop loop and the color of the loop and the loop and the color of the loop and in the should be account within a DO loop must pedent the should be account within a DO loop must pedent the should be account at the value of the loop must pedent the should be account at the restrictions. It should be account at the restrictions apply to entering the should be accounted that all restrictions apply to entering

Some mention should be made of size limitations of numbers. A fixed-point number must be less than or equal to 2<sup>17</sup>. A floating-point number can be between 10<sup>+38</sup> and 10<sup>-38</sup> or zero; however, the computer carries only 8 significant digits! A greater number of significant digits on a data card are not read.

The maximum length of a FORTRAN statement is 660 characters or blanks, and requires 9 continue cards. The maximum number of characters and blanks which can be printed on a single line is 120. The length of a single tape record on output tape is also 120 characters and blanks. A tape record is printed off-line as a single line.

Whenever an Arithmetic Statement Function is defined, it must precede the first executable statement in the program.

It is frequently convenient to include comment card in the source deck. Comment cards are punched with a C in column #1, and any comments such as program name, date, and program author are punched in columns 2 through 80 on any number of cards in succession. Comment cards at the beginning are a convenient way for the operator to identify the program being compiled, since the comment cards and all the statement cards are printed during compiling. Comment cards produce no effect upon the object deck.

In writing FORTRAN, care must be taken not to confuse the letter O with the numeral zero, which are punched differently on a card. The letter O should be written as  $\emptyset$ . The letter I and the number 1 can also cause trouble although this is due more to careless writing. The printing on a punched card is different for I and 1, so no confusion results.

# Acknowledgments

The author appreciates the suggestions and comments of numerous people in the Applied Mathematics and International Institute Divisions. The author is particularly grateful to Mr. Burton Garbow of the Applied Mathematics Division for his detailed review of the original manuscript. Suggested rephrasing of certain parts of the text which might be more satisfying to an experienced 704 user have not been incorporated in the final form of the paper in the hope that its form is more comprehensible to a novice.

Some monther areal to reade of size limitations of numbers. At fixed point miniber ment to less than or equal to 21. A thermorepoint number can be between 10 to and 10 to a sero; however, the computer darries only 2 significant digital. A greater number of significant digital.

The maximum length of a FURTRAN statement is 550 coaracters or blanks, and requires 7 contint cards. The maximum number of characters and blanks which can be printed on a single line is 120. The tength of a single time is 120 than the card of a single time is 120 than a single time. A capa record is printed off line as a single time.

Whenever as Arithmetic Salement Function is defined, it must proceed the first executable statement in the program.

It is frequently convergent to include comment card in the saured deck. Comment cards are published with a C in column 11, and any comment ments are program dense, dile, and program author are punched in columns 3 through 80 on any mumber al cards in succession. Comment cards at the be finding and a convenient way for the operator to in-stury the program being compiled, since the comment cards and all the stateshort cards are printed during compiling. Comment cards produce no suffect.

upon the object dock.

in writing FORTRAM, care must be taken not to confuse the letter Q with the admeral here, which are principed differently on a card. The letter Cashoold be written as Q. The letter I did the comber I can also cause the remaining on a request subough infairs due proper to carefuse writing. The printing on a punched card is different for I and I, so no confusion results.

# Athemybalwenisa

The author appreciates the suggestions and conditions of were cross papers in the Applie 1 Markements and International Institute Divisions. The author is particularly grateful to Mr. Purion Dathow of the Applied Markement matics Dreision for the lettiled review of the original manuscript. Suggestive replicating of meriain parts of the rows with a might be more satisfying to an experienced 704 uses have not been incorporated in the final form with the papers in the final form with the papers in the hope that its form is more comprehensible to a novices.

#### APPENDIX

Some additional information on procedures of the Argonne computer center will now be given. This will be helpful in carrying through a program to the final solution of the problem.

The program is most conveniently written on the form shown in Figure 1. Data are written on the form shown in Figure 2. The forms are available in the stationery section of the Building 203 stockroom, Room R129 on the main floor.

After preparing the written program, the source cards can be punched in Room C055 in the basement of Building 203. After an initial keypunch practice period, source cards should be punched by the girls in Room C055. Arrangements for card punching should be made with Lead Data Transcription Operator, Room C055, Extension 2876. Other card punches are located in various other divisions of the Laboratory, including one in Building 25A, Extension 2294. Generally, at each card-punching machine a supply of blank IBM cards is available. Figure 3 shows a sample source card and a data card.

The IBM 704 computer is currently installed in Room D001 in the basement of Building 203. In the same room is a 1401 computer which is programmed to check cards of the source deck. The checking of a source deck is referred to as "Pre-processing." An applied mathematics memo is available which includes a list of errors checked by the preprocessor and the code name by which they are identified. There is no additional time charge for preprocessing and, since it greatly increases the probability that the program will work on the 704, all source decks should be preprocessed. The 1401 is scheduled for preprocessing from 1030 to 1115 and from 1500 to 1545; if requested, the 1401 operator will check the results and, if it appears that the program can be compiled, will place the deck in the next compiling batch. Compiling is accomplished from 1200 to 1300 (if requested) during the day; also, compiling is done at night for pickup the next morning.

An application for authorization to obtain use of the 704 must be filed with the Applied Mathematics Division in the division office, Room B153, Building 203, by means of form AMD 6. AMD then establishes a job number and job time cards are sent to the computer room. Argonne National Laboratory employees should become acquainted with the policy of their Division in relation to authorization of computer use. Use of the 704 by persons not employed by Argonne is authorized by the Division sponsoring the person.

The specific time at which the computer is to be used is scheduled at the computer room or by calling the scheduler at Extension 2877. Requests for time of 15 min or less from 0830 to 1200 are made after 1200 of

Vermings SA

Some state around the ball the semble and great the control of the semble of the sembl

Physical Data are section on the form answer of English 200 and a form and a control of the form and the first and

After property of written program, the sauved carrie on the purchased in Rocar Curs. Author as artist to providing the Rocar Curs. Author as artist to providing providing providing position, assume cards about he sauch method for the grain that are common contract, and providing absolute as madely after the approviding to the card punchased and the approvidence of the card punchased and the card in various also the artist of the card of the card and the card the card and a state that are applied a state of the card and and a state of the card and a sta

The 18M 704 community of a vice pasts required the master of the community of the first pasts of the community of the state of the community o

As application for authorized alexion of policy to allow of the VD must be a sold on the value of the sold of the

The specific time at which the computer is to be part is a resolute. Set if the computer round of the saling the achequent at Falm stan 2879. Set give to for time of 15 min or lives them to \$2.00 are made after acts of

the preceding day; requests for time from 1300 to 1600 are made prior to 1200 of that day. The schedule is written on the blackboard in the computer room. Time on the 704 can be scheduled from 1700-2400 for periods of time greater than 15 min.

Whenever time is scheduled, a time card and a set of instructions for the computer operator must be provided. Form AMD 5 is used for the instructions. The necessary information on the form is the following:

- (A) The nature of the run, that is, whether it is compiling, computation, or both.
- (B) Does the program have on-line printing or punching?
- (C) Does it have output on magnetic tape and, if so, which tape switch number setting should be made? It should be specified if a single or double copy should be produced by the off-line printer. The single and double copy are referred to as 1 or 2 part paper, respectively. If the tape is to be saved, what identification should it have on its label? (Label information should include the name of the job, the name of the person who requested that the tape be saved, and the date after which the tape can be reused.) If no labeling information is provided, it is assumed that the tape need not be saved.
- (D) Does the program use any data cards?
- (E) What are the sense switch settings? (This introduction has not covered sense switches, so one merely says they are all up.)
- (F) What is estimated duration of the run? The operator should be informed when to stop a run, particularly if the program is new, to avoid wasting time on a program which may malfunction. For programs which have been previously run successfully, the operator can be told to allow the program to run to completion and the time required can be predicted with reasonable accuracy.

After compiling, the source and object decks and the printed sheets produced by the compiler are on distribution shelves in D-001 where they can be picked up.

The off-line printing from magnetic tapes is routinely done during the day within an hour and a half of the completion of the computer run. The printed output can be picked up from the shelves adjacent to the 1401 computer. the preceding day; requests for time from here the are made order to the of that day. The achievance is writing on the place heard in the computer room. Time on the 704 can be scheduled from 1706-1400 for periods of time greater than 15 min.

Whenever time in scheduled, a time cord and a set of mercetlans.

for the computer appender must be provided. Form Mill b is used for the instructions. The accessary information on in form is the following:

- (A) The nature of the run, that is whether it is complicated during
  - rgalifornia ra gaintina anchina aved atempera nal acou.
- (C) Bone it have dupid on magnetic contraint in an which taked

  switch number detting should be maded it should be consided

  if a single of double dopy should be preduced by the cit-line

  printer. The single and double core are edismission as long

  2 part popul, respectively. It the fair is to be saved, what

  identification should it have on its label? (Label Lifteriation

  should believe the time name of the lot, the care of the period whe

  requested that the tipe to caved, and the date after which ins

  tage can be recased.) If no labeling interfraction is provided.
  - (6) Does the program day any data carde
  - What are the sense switch actings (The introduction of the fall (The control of the control of t
  - (#) What is eliminated thration or the run; The operator should be informed when to stop a run, particularly if the program of a few to avoid wasting lime on a program whiching ideas described. For programs which have been proceeding with the law the appropriate was be told to blick the programs. The programs of the time requires can be predicted.

(Miles normalling, the fource and disjoint decide and the printed special greed and by the compiler are on distribution shylver in D-001 where they can be picked up.

The off-line printing printing from magnetic trees is routinely done during the day within an four and a half of the long letter of the computer rout.

The printed outpit com he picked up from the similar adjacent to the languager.

PROBLEM	DATE PAGE OF	PROGRAMMER
		IDENTIFICATION
	7.	2 73 8
	The state of the s	
La production of the second	The state of the s	
		The South Print
	7	FORTRAN STATEMENT

Figure 1

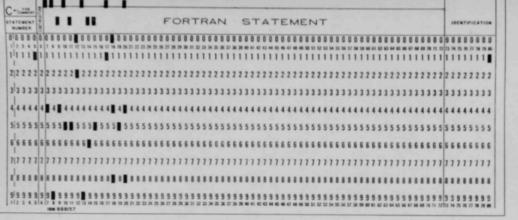
IBM Fortran Programming Form (reproduced by permission of International Business Machines Corporation.)

704 INPUT DATA

COST CODE

PROGRAM PROBLEM ORIGIN AT OR DATE PAGE OF 1 2 3 4 5 6 7 8 9 0 1 2 3 4 111111111 111111111111111111111111111111 'IIIIIII IIIIIII ...... LILLI · IIIIIIIII IIIIII . 111 1 2 3 4 5 6 7 8 9 0 1 2 3 4

Figure 2



Sample FORTRAN Statements card. Note difference in printing of capital letter I and the numeral one. The standard letter O and zero can only distinguished by the punchanged holes.

Sample data card. Note that a different type of card is normally used for the source cards and data cards.

Figure 3

