

FORTRAN NOTES - Prof. Richard B. Goldstein

BRIEF HISTORY

FORTRAN was first developed in the mid-50's by John Backus (who was only in his twenties) and others from IBM. It has progressed through many versions from FORTRAN II to FORTRAN IV to FORTRAN 77 and it is still being upgraded such as Digital/Compaq's Developer Studio 90 and 95. In the early versions traditional punched cards were the input form. They had 80 columns in which the punched holes represented various letters and numbers. Today's line input structure still uses that form, called fixed-form, but the newer versions allow for free-form. These versions also allow for graphics, fonts, interaction with languages such as C and Visual Basic, and output in multiple windows.

FIXED-FORM PROGRAM LINE STRUCTURE (FORTRAN 1977)

Columns 1-5	6	7-72	73-80
statement numbers where necessary	continuation column &, *, or + are typical choices	PROGRAM SAMPLE - title line of main routine REAL ... - declarations of variables, arrays _____ - {Program} _____ END SUBROUTINE/FUNCTION _____ {subprogram(s)} _____ RETURN END	optional line numbers

remarks:

- [1] The traditional column 7 is used for program statements
- [2] Comment statements: a "C" in column 1 followed by anything in columns 2-72
- [3] Continuation of large algebraic formulas for example is accomplished by:

```

column      6      7-72
              X=(A+B*SIN(3*SQRT(C+5*D)...
              & /(3*FX+.....
              & +(Q-P)-....
    
```

- [4] Statement numbers are usually left-justified within columns 1-5 such as 40__ __
- [5] Indenting within columns 7-72 for ease of reading program structure such as for loops or conditional statements may be used.
- [6] **DO NOT CONTINUE A PROGRAM LINE BEYOND COLUMN 72.**

CONSTANTS

TYPE	EXAMPLES	COMMENTS
Integer	23 0 -5128	32-bit range from -2^{31} to $2^{31}-1$ =2,147,483,647 the metacommand \$STORAGE:2 changes it to 2^{15}
Real	3.56 -0.0033367 2.8146E-7	24 bits for mantissa, 8 for exponent 6-7 digit accuracy with smallest= $1.1755E-38=2^{-126}$ largest= $3.4028E+38=2^{128}$
Double Precision	3.141592653589D+2	56 + 8 bits give 13-digit accuracy smallest= $2.2251D-308=2^{-1022}$ largest= $1.7977D+308=2^{1024}$
Complex	(1.56,-2.0)	real, imaginary parts

VARIABLES

legal:	X ROOT Q12AB	(up to 6 characters)
illegal:	36ABC QA.4 EPSILON	starts with a number no periods allowed too long

DECLARATION STATEMENTS

INTEGER I, ABC, J(10)
REAL X, A, FUNCT, Q(10,5)
DOUBLE PRECISION Y, ROOT
COMPLEX Z, W
PARAMETER (MAXN=100, PI=3.141593)
LOGICAL FLAG, BOOL(100)

notes:	[1]	J(10) also declares the array size
	[2]	one definition to a variable
	[3]	unlike variables, parameter do <u>not</u> change value in a program
	[4]	default usage: I, J, K, L, M, and N begin integers A-H and O-Z begin reals

FORMULAS

arithmetic operators + - * / **

$\frac{3.5(a-b)^4}{\sqrt{x+8}}$ becomes $3.5*(A-B)**4/(SQRT(X)+8)$

Note: 4 is OK in exponent, and the sqrt makes the decimal after 8 unnecessary

3^{2^4} is the same as 3^{16} , not 9^4 - that is, $3**2**4$ evaluates correctly, unlike BASIC

priority inner-parentheses, exponents, multiplication & division, addition & subtraction are done from left to right where equal - except exponentiation which is done from right to left

integer arithmetic 3/2 evaluates as 1, 1/2 evaluates as 0
1 + 6.0 evaluates as 7.0 4.0**2 evaluates as 16.0
2**3. is illegal

assignment statement

$X=Y**N - 3.5*A/(B + C)$ spaces are OK
 $U(J)=U(J-1) + \text{COS}(\text{THETA})$ uses COS, an Intrinsic Function (Built-in functions)

SQRT, EXP, LOG, LOG10, SIN, COS, TAN, ASIN, ACOS, ATAN, SINH, COSH, TANH, ABS, MAX, MIN, INT, REAL, MOD are some of the Intrinsic functions

CONTROL STATEMENTS

all logical variables and expressions result in either of two values: .TRUE. or .FALSE.

relation operators .EQ. .LT. .GT. .LE. .GE. .NE.

logical operators .AND. .OR. .NOT. .EQV. .NEQV.

logical expressions .TRUE.
X .LT. 2.53*Y X < 2.53Y
A .GE. -2.4 .AND. Y .LE. 5.4 {A\$-2.4} 1 {Y#5.4}

logical assignments FLAG=.TRUE. FLAG is set to "True"
X=A.LE.B+C X is set to "True" if A#B+C

IF statement forms

1. IF (logical expression) THEN

block of 1 or more statements
that execute if true

ENDIF
2. IF (logical expression) THEN

block of 0 or more statements
that execute if true

ELSE

block of 0 or more statements
that execute if false

ENDIF
3. IF (logical expression #1) THEN

block of 0 or more statements
that execute if #1 is true

ELSE IF (logical expression #2) THEN

block of 0 or more statements
that execute if #2 true, #1 false

 •
 •
 •
ELSE

block of 0 or more statements
that execute if all above exprs. are false

ENDIF

Nested IF

```
IF ( ) THEN
  IF ( ) THEN
    ●●●
  ENDIF
  IF ( ) THEN
    ●●●
    IF ( ) THEN
      ●●●
    ENDIF
  ENDIF
  ●●●
ENDIF
ENDIF
```

Other CONTROL statements

	GOTO statement number	ex.	GOTO 200
statement no.	CONTINUE	ex.	50 CONTINUE
opt. state. no.	STOP	ex.	STOP

LOOPS

DO statement
body of loop
CONTINUE

ex. DO 100 I=1,10
body of loop
100 CONTINUE

ex. DO 200 X=A, B+C*D, E (start, end, step - may be expressions)
●●●
200 CONTINUE

ex. DO ●●● (nested)
DO ●●●
●●●
CONTINUE
CONTINUE

PASCAL approach

FORTRAN approach

[1] WHILE (logical expression) DO
●●●
ENDWHILE

n CONTINUE
IF (logical expr.) THEN
●●●
GOTO n
ENDIF

[2] REPEAT
●●●
UNTIL (logical expression)

n CONTINUE
IF(.NOT.log. expr.) GOTO n

FORMAT STATEMENTS

XXX FORMAT()

FORMATTING TYPES:

A - alphanumeric, general size
A3 - alphanumeric truncated to 3 characters
I7 - 7 digit integer (more than 7 gives *****)
F10.2 - 7 digits (including sign), a decimal point, and 2 after the decimal
examples: 4127821.92, -287818.11 (up to 10 characters in all)
E12.5 - 2 digits (including possible sign), decimal point, 5 digits, E, ±, and 2 digits
examples: 2.18532E-21, -1.91782E+18
'STRING' - string constant (literal)
' - line return
3I5 - 3 integers of length 5 characters each
10X - 10 spaces

others D is used for double precision and G is used for general

DATA STATEMENT:

DATA X,Y,A/2.5,3.24,PI/ where PI=parameter previously defined
DATA C/(2.4,-3.2)/ where C=complex number 2.4-3.2i
DATA (X(I),I=1,100)/100*0.0/ sets the X array's 100 values to 0.0

FUNCTIONS & SUBROUTINES

[1] FUNCTIONS

[A] May be defined on top with other declaration statements (such as REAL, etc.)

Examples: $F(X)=3*X^{**4}-2.5*X^{**2}+17.2$
 $G(X,Y)=EXP(2*X)*SIN(Y)$

[B] As more complicated routines:

PROGRAM MAIN

●●●

Y=FUNCT(one or more variables)

A=2.*A/FUNCT(one or more variables)

●●●

END

REAL FUNCTION FUNCT(an equal sized & type list of variables)

●●● (several preparatory steps)

FUNCT=an arithmetic statement that defines the value of the function

●●●

RETURN (optional to RETURN earlier - before END)

●●●

END

for example P=FUNCT(X,A+3.2*ABS(T),21,.TRUE.) with

REAL FUNCTION FUNCT(A,B,N,L) and the A, B, N, and L are dummy local variables

[2] SUBROUTINES

PROGRAM MAIN

●●●

CALL ASUB(one or more variables)

●●●

END

SUBROUTINE ASUB(an equal sized & type list of variables)

●●●

RETURN (not an option here)

END

- notes:
- [A] if one of the variables in the list is a function name, then EXTERNAL is used (ex. EXTERNAL FUNCT), but if it is a built-in function use INTRINSIC (ex. INTRINSIC COS)
 - [B] variables are local in both Functions & Subroutines
 - [C] Functions & Subroutines may be nested but not recursive
 - [D] a COMMON statement can be used to share a variable list
 - [E] do not redimension arrays used in the calling variable list

ARRAYS

DIMENSION X(10), Y(-5:50), A(10,3,0:18), T(LOW:HI)

default origin is 1, but negative subscripts or 0 can be used for one or more of the array's n-tuples (ex a 3-tuple array)

CHARACTER DATA

CHARACTER*10, NAME, CITY

NAME='WILLIAM'

CITY='PROVIDENCE' (maximum size used is 10 characters)

It is O.K. to compare: NAME.GE.'SMITH'

NAME='WILLIAM'// '//'JONES' where // is concatenation (string addition)

some functions:

LEN('STAR')	is 4
ICHAR('A')	is 65
CHAR(66)	is B
INDEX('UL','CALCULUS')	is 5

SAMPLE PROGRAM

```
PROGRAM BISECT
F(X)=X**3+4*X**2-10
PRINT *, ' INPUT A, B, TOL: '
READ *, A, B, TOL
P=(A+B)/2
FA=F(A)
FB=F(B)
IF (FA*FB.GT.0) THEN
    PRINT *, ' INVALID CHOICE OF A AND B '
    STOP
ELSE
10    FP=F(P)
    IF (FA*FP.GT.0) THEN
        A=P
    ELSE
        B=P
    ENDIF
    WRITE (6,100) P, FP
100   FORMAT(' ROOT =', F12.5, 10X, ' F(ROOT) =', E15.7)
    IF (ABS(FP) .GT. 1.0E-20 .AND. (B-A)/2 .GT. TOL) THEN
        P=(A+B)/2
        GOTO 10
    ENDIF
ENDIF
END
```

SCREEN OUTPUT FROM RUN:

```
INPUT A, B, TOL
-1 3 .001
ROOT =      1.36523    ← final
F(ROOT)= .7202476E-04 ← line
```