5. Basic Concepts of FORTRAN 90

5.1 A Brief History

FORTRAN from its development was intended for translating scientific equations into computer code. It's name is derived from FORmula TRANslation. FORTRAN was developed between 1954 and 1957 by IBM, it was the first programming language that converted an algorithm into machine language that the computer could read and execute.

FORTRAN went through many revisions over the years. FORTRAN II came out in 1958 and FORTRAN IV in 1962. FORTRAN IV was renamed FORTRAN 66 when it became an ANSI standard in 1966. The next major update came in 1977 with FORTRAN 77. This is the version that has been used for the last 20 years. FORTRAN 90 was developed in 1991.

5.2 Structure of a FORTRAN Program

A FORTRAN program can be divided into three sections:

Program Name – All programs and subprograms have names. The name can consist of up to 31 characters (letters, digits, or underscore), starting with a letter. The name of the program is optional.

Comments – Comments are non-execution statements used to describe each part of the program. Each comment starts with exclamation mark !.

Declarations – This section consists of a group of statements at the start of the program which are used to declare the variables of the program.

Execution – This section consists of one or more statements describing the actions to be performed by the program.

Termination – This section consists of a statement (or statements) telling the computer to stop/end running the program.

5.3 Data Types, Declaration, and Parameterization

FORTRAN 90 deals with Five types of data.

1. Real: There are two representations,

- Decimal Representation: Real data must contain the decimal point like 23.45, 0.123, 123.0, -0.12, -0.12.
- Exponential Representation: It consists of an integer or a real number in decimal representation followed by the letter E followed by an integer (the exponent) like 12.3456E2 , -1.2E-3 , 12E3.

Real numbers are also stored in 4 bytes and have only about 7 significant digits. Real numbers can lie typically between approximately $\pm 10^{38}$. Doubl precision numbers are stored in 8 bytes. They have about 15 significant digits and can lie in the approximate range $10^{-307} - 10^{308}$.

Real variables are declared as follows:

```
REAL :: A, B, C

REAL (KIND = double) :: D OR REAL*8 :: D
```

A, B and C are variables of real type. D is a double precision real variable.

2. Integer: Integers are represented by a string of numbers not including decimal points. Integers are normally stored in 4 bytes (that is, 32 bits, i.e. 32 binary 0s and 1s). This allows integers from -2147483648 (-2^{31}) to +2147483647 (2^{31}- 1) to be represented. For examples 12, -2745 .

An integer variable is declared as follows:

```
INTEGER :: A, B
```

A and B are variables of integer type.

3. Character: Fortran only uses the following characters:

- Letters: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
  a b cd ef g h i j k l m n o p q r s t u v w x y z
- Digits: 0 1 2 3 4 5 6 7 8 9
- Special Characters: Space ' " () * + - / : = _ ! & $; < > % ? , .

A character variable is declared as follows:

CHARACTER (LEN = 9) :: name = "age is 25"

CHARACTER :: C

The above will declare a variable called *name* that can be up to 9 characters long. If the (len =) is omitted, the length is assumed to be 1.

CHARACTER(LEN=*) :: Title, Position

Here, the actual lengths of variables **Title** and **Position** are unknown and will be determined elsewhere.

4. Logical: Logical variables have one of two values**: .TRUE.** or **.FALSE.** They take storage space of 1 byte. A logical variable is declared as follows:

LOGICAL :: A, B

Here A and B have been declared as logical variables.

```
logical :: a, b
a = .TRUE.
b = a .AND. 3 .LT. 5/2
```

5. Complex: FORTRAN allows variables to be declared as complex numbers. The complex number is stored in 8 bytes (4 bytes for the real part and 4 bytes for the imaginary part).

The following statement show a complex variable declaration:

COMPLEX :: C

5.4 Rules have to be followed in forming a variable name:

1. The first character must be a letter, either lowercase or uppercase;

2. Case is insignificant, uppercase and lowercase letters are same;

3. Variable names are composed of letters, numbers, and the underscore character without spaces.

Examples: sum , sum2 , my_age , AVE12rage

The following variable names are not allowed:

2X (starts with a number)	Z2+ (contains an operator)	my age (contains a space)

5.5 Parameter (Constant) Declaration

We declare a constant in FORTRAN 90 using the word PARAMETER after the type as follows :

INTEGER, PARAMETER :: N = 10

REAL, PARAMETER :: pi = 3.141593

CHARACTER(len = 7), PARAMETER :: ERROR= "error 1"

5.6 Input and Output Statements

One of the important features of programming is being able to read input (from the keyboard, a data file, etc.) and output results (to the screen, a data file, etc.). The general commands available for these actions are:

READ The general form is: **READ *, input-list (unformatted)**	**PRINT** The general form is: **PRINT *, output-list (unformatted)**

`CHARACTER(len=8) :: name`
`INTEGER :: x`
`REAL :: y`
`LOGICAL :: A`
`COMPLEX :: comp`
`READ*, name, x, y, A, comp`
The inputs are :
`"ALI" 23 5.654 T(2,-3)`
	`PRINT *, name, x, y, A, comp`
	`The outputs are :`
	`ALI  23  5.654  T  (2 , -3)`

5.7 Operators in FORTRAN 90

5.7.1 Assignment Operator:

The basic assignment operator is (=) which is often called equal to. The assignment has the form:

variable_name* = *expression

Consider the following assignments:

```
INTEGER:: x=5 , y=10
Logical:: a, b
a = .TRUE.
b = a .AND. (3 .LT. 5/2)
z = x**2-3*x+7
w = x+y
x = x + 0.5
```

```
REAL, PARAMETER :: PI = 3.14
REAL :: Area
INTEGER :: Radius
Radius = 5
Area = (Radius ** 2) * PI
```

5.7.2 Arithmetic Operators

Arithmetic operators in FORTRAN 90 are Explained in the table below:

Operator	Usage	Examples
+	Used for addition	Sum = a + b
-	Used for subtraction	Difference = a – b
*	Used for multiplication	Product = a * b
/	Used for division	Quotient = a / b
**	Exponentiation	z = a ** b

1.7.3 Relational Operators:

The relational operators are explained in the following table:

Operator	Usage	Examples
.LT. or <	Less than	The following expressions are TRUE 5 < 7 , 3 >= 2 , 'A' < 'B' "HASAN" < "HASSAN" The following expressions are FALSE 5 > 7 , 2 /= 2 , 'a' > 'b' "AAA">"AAB"
.GT. or >	Greater than	
.LE. or <=	Less than or equal	
.GE. or >=	Greater than or equal	
.EQ. or ==	Equality	
.NE. or /=	Not equal to	

5.7.4 Logical Operators

The logical operators are used to combine multiple conditions (logical statements). The following table describes the logical operators

Operator	Usage	Examples
.AND.	The compound condition is true, if both conditions are true.	(a>b .AND. a>c) (3>2 .AND. 'A'>'B')
.OR.	The compound statement is true, if any or both conditions are true.	(a>b .OR. a>c) (3>5 .OR. 'A'>'B')
.NOT.	It negates the condition.	.NOT.(a>b) .NOT. (FALSE)

5.8 Precedence of Operators

Precedence is an important aspect of operators. A list of operators and their precedence are given in the following table:

Type	Operator		Associativity
Arithmetic	**		right to left
	*	/	left to right
	+	-	left to right
Logical	.NOT.		right to left
	.AND.		left to right
	.OR.		left to right

Note: Parentheses ( ) are given the highest precedence over all the arithmetic and logical operators . Inner parentheses is evaluated first.

1.9 Expressions in FORTRAN 90

FORTRAN 90 expressions can be illustrated through the following examples:

$$(1)\ \frac{ax+b}{c} \quad , \quad (2)\ \frac{1-e^{a\sqrt{x}}}{1+xe^{-|x|}} \quad , \quad (3)\ \frac{a}{\sqrt{a^2+b^2}}\cos(bt+c)$$

(1) `(a*x+b)/c`

(2) `(1.0 - EXP(a*SQRT(x)))/(1.0 + x*EXP(-ABS(x)))`

(3) `a*COS(b*t+c)/SQRT(a**2 + b**2)`

Ex: Evaluate the following expressions:

$$(i)\frac{4*6^2}{2+3*4-12} \ , \qquad (ii)\frac{a(a+2.5)}{2a} \ , a=2.5\ , b=2.5$$

$$(iii)\ (3>2).AND.(1+2)<3\ .OR.(4<=3)$$

Solution

```
(i) 4*6**2/(2+12-12)= 4*6**2/(14-12)= 4*6**2/2=4*36/2
    = 144/2= 72

(ii) a*(a+2.5)/(2*b)=2.5*(2.5+2.5)/(2*2.5)
```

```
     =2.5*(5)/(2*3.5)= 2.5*(5)/5.0 =(12.5)/5.0=2.5

(iii) (3>2) .AND. (1+2)<3 .OR. (4<=3)
       =.TRUE. .AND. .FALSE. .OR. .FALSE.
       = .FALSE. .OR. .FALSE=.FALSE.
```

Ex: Write a program to convert a distance in feet to meters

```
PROGRAM Feet_to_Meters
! Program to convert a length given in feet
! to the corresponding length in meters.
    IMPLICIT NONE
    REAL, PARAMETER :: FtoM = 0.3048
    REAL :: feet, meters
    PRINT *, "Enter the length in feet"
    READ *, feet
    meters = feet * FtoM
    PRINT *, "Meters= ", meters
END PROGRAM Feet_to_Meters
```

Ex: Write a program to compute the area and circumference of a circle.

```
PROGRAM Area_Circumference
   IMPLICIT NONE
   REAL,PARAMETER :: PI=3.141593
   REAL :: radius, area, circum
   PRINT *,"Enter the radius:"
   READ *, radius
   area=radius**2*PI
   circum = 2*PI*radius
   PRINT *,"Area=", area, "  Circumference=", circum
END Area_Circumference
```