FORTRAN 90

Basic Concepts of FORTRAN 90

A Brief History

The particular set of rules for coding the instructions to a computer is called a programming language. There are many such languages, for example Fortran, BASIC, Pascal and C++. Fortran, which stands for FORmula TRANslation, was the first "high level" programming language. It made it possible to use symbolic names to represent mathematical quantities, and to write mathematical formulae in a reasonably comprehensible form. The idea of FORTRAN was proposed in late 1953 by John Backus, in New York, and the first Fortran program was run in April 1957

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.

Data Types, Declaration, and Parameterization

FORTRAN 90 deals with Five types of data.

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 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.

Integer: Integers are represented by a string of numbers not including decimal points.

An integer variable is declared as follows:

INTEGER :: A, B

A and B are variables of integer type.

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 c d e f 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.

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.

Complex: FORTRAN allows variables to be declared as complex numbers.

The following statement show a complex variable declaration :

COMPLEX :: C

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.

Parameter (Constant) Declaration

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

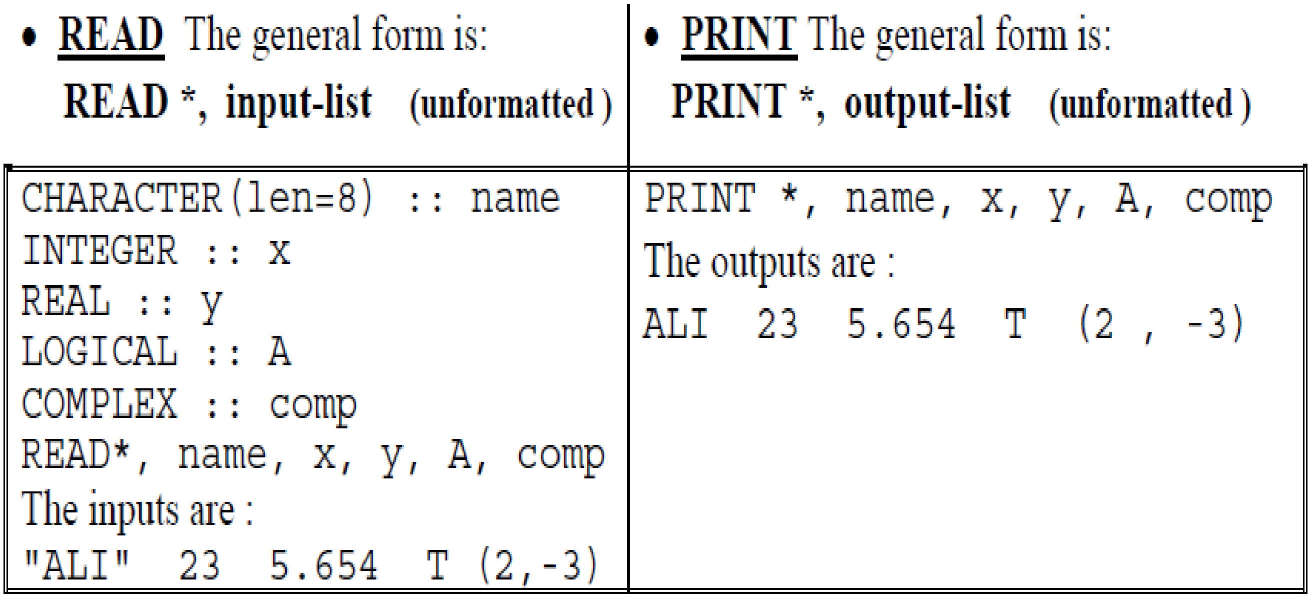
INTEGER, PARAMETER :: N = 10

REAL, PARAMETER :: pi = 3.141593

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

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)

```
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** The general form is:

  PRINT *, output-list (unformatted)

```
PRINT *, name, x, y, A, comp
```

The outputs are :

```
ALI  23  5.654  T  (2 , -3)
```

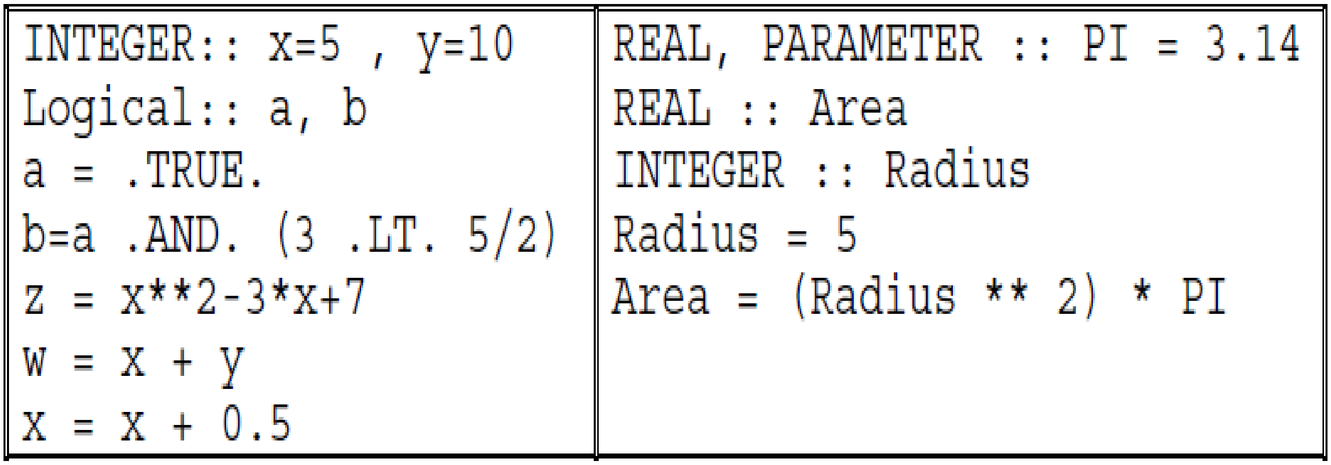
Operators in FORTRAN 90

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
```

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`

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	

Logical Operators

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

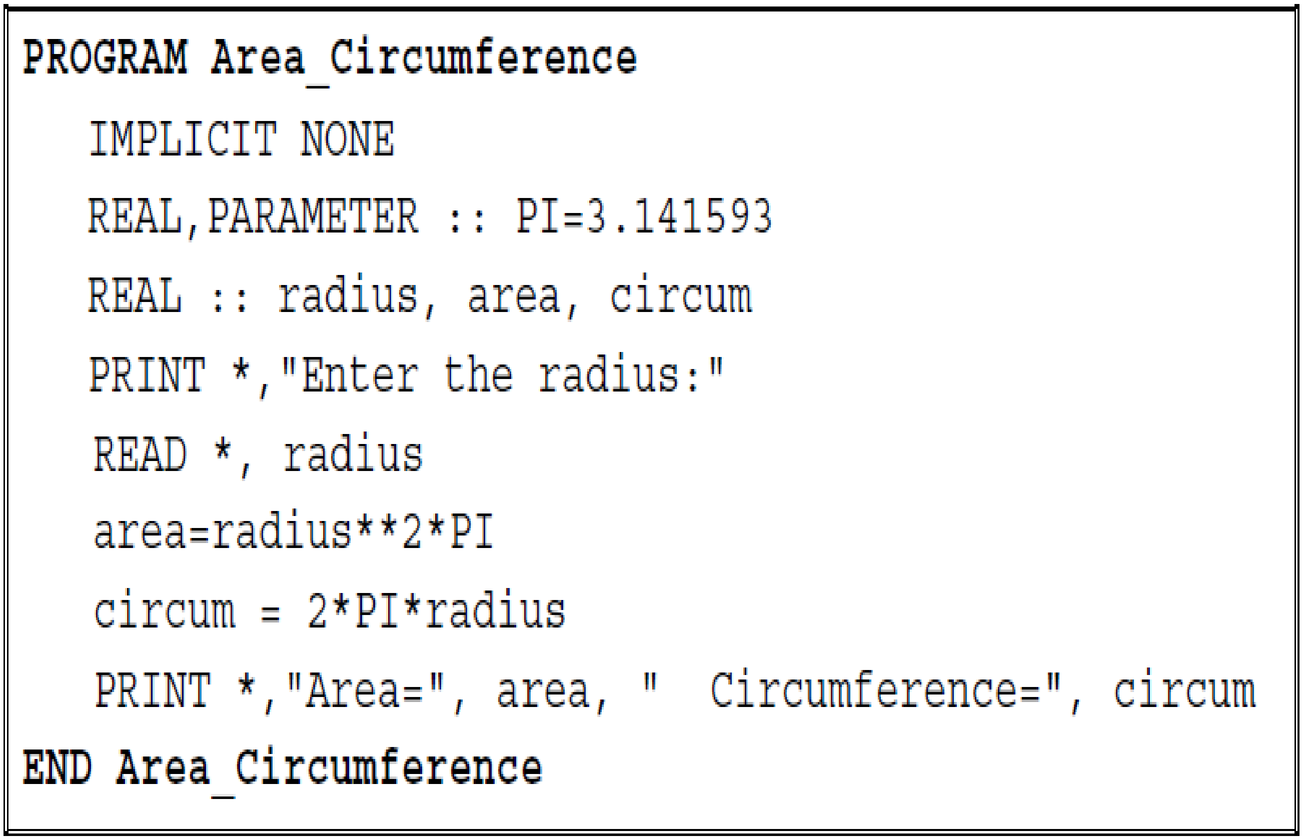
Operator	Usage	Example

.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)`

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
```