4.1.- EXPRESIONES LOGICAS

Una expresión lógica únicamente puede tomar dos valores que son :

.true. o .false.

Este tipo de expresiones permiten crear *estructuras de control* dentro del programa FORTRAN que dependiendo de los valores obtenidos por la expresión se bifurcan y recorren caminos alternativos.

A la hora de construir una expresión lógica FORTRAN cuenta con una serie de **operadores lógicos** que se aplican <u>siempre entre</u> dos elementos del mismo tipo, bien numéricos o bien tipo carácter. Una vez construida la expresión esta únicamente puede tomar dos valores cierto o falso.

Los operadores en FORTRAN son los siguientes:

| OPERADOR | SIGNIFICADO |
|----------|-------------------------------|
| .EQ. | Igual que (=) |
| .NE. | No igual que (¹) |
| .LT. | Menor que (<) |
| .LE. | Menor o igual que (${f f}$) |
| .GT. | Mayor que (>) |
| .GE. | Mayor o igual que (³) |

Ejemplo

Supongamos que una variable real **A** posee un valor de 5.6 y otra variable real **B** posee un valor de 7.8, la siguiente tabla muestra los valores de lógicos para una serie de expresiones lógicas:

| EXPRESION | VALOR |
|---------------|---------|
| (A .gt. 50.0) | .false. |
| (A .ge. B) | .false. |
| (A.ne.B) | .true. |

Por otro lado, hay otro tipo de **operadores lógicos** que son utilizados únicamente entre expresiones lógicas, en concreto .**NOT.** opera sobre una sola expresión lógica y el resto sobre dos. La siguiente tabla muestra este tipo de operadores y sus significado.

| OPERADOR | SIGNIFICADO |
|----------|---|
| .NOT. | Cambia el valor de la expresión lógica a su opuesto |
| .AND. | Cierto únicamente si ambas expresiones lógicas son ciertas |
| .OR. | Cierto si una de las expresiones es cierta |
| .XOR. | Cierto únicamente si una de las expresiones lógicas es cierto |
| .EQV. | Cierto si ambas expresiones tiene el mismo valor |
| .NEQV. | Cierto si ambas expresiones no tienen el mismo valor |

Ejemplo

Supongamos dos variables con valores A=2.0 y B=10.0, entonces:

.not. (a .lt. b)
(a .ne. b) .and. (a .lt. b)
(a .lt. b) .or. (a .gt. b)
(a .ne. b) .xor. (a .lt. b)

(a .lt. b) .eqv. (a .gt. b)
(a .lt. b) .neqv. (a .gt. b)

.true. (una de las expresiones lógicas es cierta)

.true. (una de las expresiones son ciertas)

.true. (ambas expresiones son ciertas)

.false. (los resultados de las expresiones no son iguales)

.true. (los resultados de las expresiones no son iguales)

Cuando en una misma expresión lógica compuesta existen varios operadores la prioridad de mayor a menor es:

- 1. Paréntesis
- 2. Operadores aritméticos (**, *, /, +, -)
- 3. Operadores lógicos (.lt., .gt., .eq., .ne., .le., .ge.)
- 4. .not.
- 5. .and.
- 6. .or.
- 7. .eqv. and .neqv.

Cuando en una misma expresión existen operadores de la misma prioridad las operaciones se realizaran recorriendo la expresión *de izquierda a derecha*.

4.2.- ESTRUCTURAS DE CONTROL

Una vez vistas las posibilidades que ofrecen la conjunción de expresiones lógicas, así como su funcionamiento podemos introducir las *estructuras de control*. Este tipo de estructuras son fundamentales en el desarrollo de un programa FORTRAN y permiten implementar bifurcaciones operacionales que son indispensables en la mayoría de los programas.

La sentencia FORTRAN que permite este tipo de estructuras es la sentencia **IF** (**lógico**) la cual tiene una estructura como la que se muestra a continuación:

IF (expresión lógica) THEN

Sentencias nº 1 ejecutadas cuando expresión logica=.true.

ELSE

Sentencias nº 2 ejecutadas cuando expresión logica=.false.

ENDIF

En caso de no existir o no ser necesarias las sentencias nº 2, la sentencia **ELSE** no es de uso obligado. En ese caso el Bloque IF queda del siguiente modo:

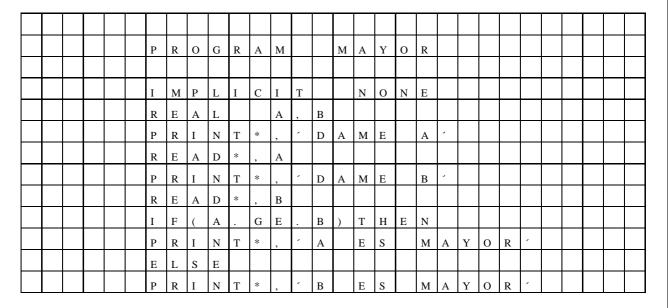
IF (expresión lógica) THEN

Sentencias nº 1 ejecutadas cuando expresión logica=.true.

ENDIF

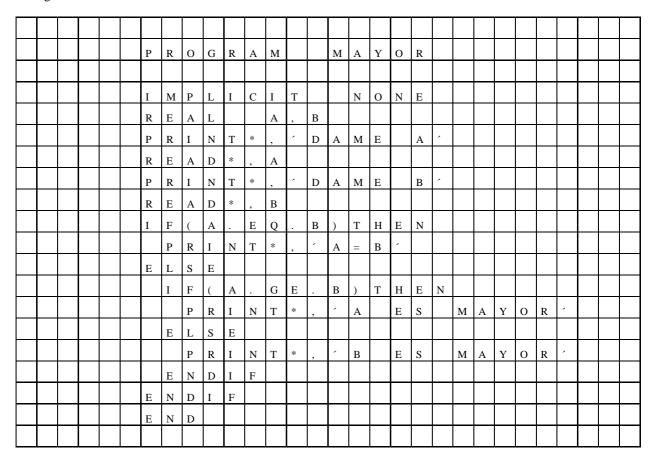
Ejemplo:

Programa que lee dos números y nos dice por pantalla cual de ellos es el mayor.



| | | | Е | N | D | I | F | | | | | | | | | | |
|--|--|--|---|---|---|---|---|--|--|--|--|--|--|--|--|--|--|
| | | | Е | N | D | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |

Obsérvese que el programa responderá que A es mayor, cuando A=B, lo que no es muy correcto. Analizando el problema observamos que deberíamos haber tenido en cuenta el caso. Para ello podemos incluir otra sentencia IF del siguiente modo:



De la modificación anterior es preciso observar detenidamente dos cuestiones:

a) Al existir dos sentencia **IF**, su anidamiento debe respetarse. Esto significa que las sentencias **ELSE** y **ENDIF** del segundo **IF** nunca deberán se escritas con posterioridad a las sentencia **ELSE** y **ENDIF** de del primer **IF**.

| ANIDAMIENTO CORRECTO | ANIDAMIENTO INCORRECTO | | | | | | | | | |
|----------------------------|----------------------------|--|--|--|--|--|--|--|--|--|
| IF (expresión lógica) THEN | IF (expresión lógica) THEN | | | | | | | | | |
| IF (expresión lógica) THEN | IF (expresión lógica) THEN | | | | | | | | | |
| Sentencias FORTRAN | Sentencias FORTRAN | | | | | | | | | |
| ELSE | ELSE | | | | | | | | | |
| Sentencias FORTRAN | Sentencias FORTRAN | | | | | | | | | |
| ENDIF | ELSE | | | | | | | | | |
| ELSE | ENDIF | | | | | | | | | |
| Sentencias FORTRAN | Sentencias FORTRAN | | | | | | | | | |
| ENDIF | ENDIF | | | | | | | | | |

b) Al aumentar el numero de sentencia anidadas es muy conveniente para una correcta comprensión y lectura del programa aplicar tabulaciones hacia la derecha para cada bloque **IF**, con el fin de que cada bloque quede alineado verticalmente y separado de los demás bloques.

Dentro de la estructura **IF** existe una variante permitida en la que no es necesario poner las sentencia **ELSE** y **ENDIF**, su estructura es:

IF (expresión lógica) Sentencia ejecutable que se aplicará cuando expresión logica=.true.

Ejemplo

Programa que lea un carácter y si es la letra S escribe la palabra YES, sino que no escriba nada

| | | | P | R | О | G | R | A | M | | | С | U | A | D | R | A | D | 0 | | | | | | | |
|--|--|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | I | M | P | L | I | С | I | T | | | N | О | N | Е | | | | | | | | | | |
| | | | С | Н | A | R | A | С | T | Е | R | * | 1 | | A | | | | | | | | | | | |
| | | | P | R | I | N | Т | * | , | , | D | A | M | Е | | U | N | A | | L | Е | T | R | A | , | |
| | | | R | Е | A | D | * | , | A | | | | | | | | | | | | | | | | | |
| | | | I | F | (| A | | Е | Q | | ` | S | ` |) | P | R | I | N | T | * | , | ` | Y | Е | S | , |
| | | | Е | N | D | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

Al introducir por teclado un carácter que será leído por la sentencia READ*, A este deberá ser introducido entrecomillado, es decir: 'S'.

Por ultimo, dentro de la construcción de varios bloques IF entrelazados se establece una nueva forma que responde a la siguiente estructura:

IF (expresión lógica 1) THEN

Sentencias n^o 1 ejecutadas cuando **expresión lógica 1 = .true.**

ELSE IF (expresión lógica 2) THEN

Sentencias n° 2 ejecutadas cuando **expresión lógica 2 = .true.**

ELSE IF (expresión lógica 3) THEN

Sentencias nº 3 ejecutadas cuando **expresión lógica 2 = .true.**

.....

ELSE IF (expresión lógica N) THEN

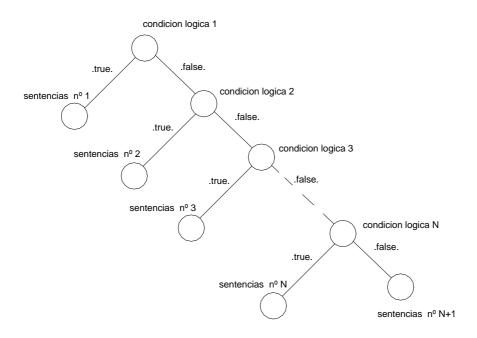
Sentencias nº N ejecutadas cuando **expresión lógica** N = .true.

ELSE

Sentencias $n^o N+1$ ejecutadas cuando **expresión lógica 1 = .false.**

ENDIF

De manera gráfica puede explicarse fácilmente la estructura anterior como se puede ver a continuación:



EJERCICIO 4.1

 Realizar un programa que lea un numero por pantalla y comunique al usuario si es par o impar.

EJERCICIO 4.2

• Realizar un programa que lea un valor entero comprendido entre 1 y 5 y presente por pantalla la vocal a,e,i,o,u en función del numero leído. El programa deberá advertir al usuario en caso de que el valor introducido no este en el intervalo [1,5].

EJERCICIO 4.3

 Realizar un programa que lea una letra del abecedario y diga al usuario si se trata de una vocal o no.

EJERCICIO 4.4

 Programar la ecuación de 2º grado Ax² + B x + C = 0, con la capacidad de dar como resultado las raíces complejas.

EJERCICIO 4.5

 Realizar un programa que calcule el modulo y el argumento de un numero complejo (a+bi), poniendo especial cuidado en la ubicación del argumento resultante en el cuadrante correcto.