

# Programación básica

---

FORTRAN (SEGUNDA PARTE)




# Acciones en un código

---

Operaciones matemáticas  
+ - \* /

Operaciones lógicas  
If else - select case



Muchas veces  
Mientras , repetir

# Expresiones lógicas

---

Los operadores de relación permiten comparar elementos numéricos, alfanuméricos, constantes o variables.

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>	<i>Resultado</i>
<b>.EQ.</b>	Igual que	'h' .EQ. 'H'	Falso
<b>.NE.</b>	Diferente a	'a' .NE. 'b'	Verdadero
<b>.LT.</b>	Menor que	7 .LT. 15	Verdadero
<b>.GT.</b>	Mayor que	11 .GT. 22	Falso
<b>.LE.</b>	Menor o igual	15 .LE. 22	Verdadero
<b>.GE.</b>	Mayor o igual	20 .GE. 35	Falso

# Expresiones lógicas

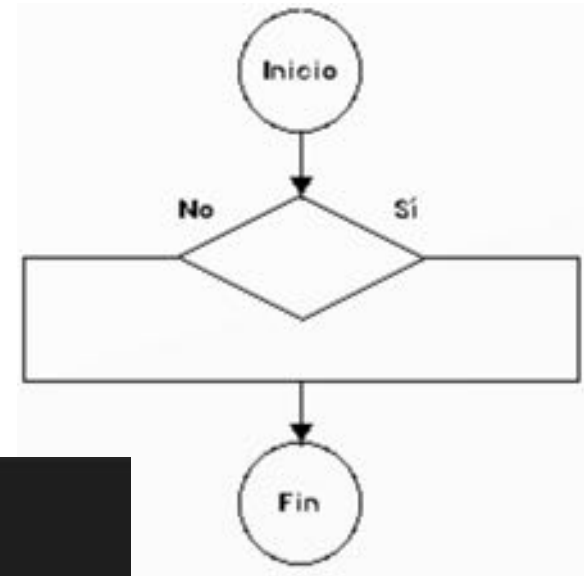
---

Los operadores de lógicos permiten formular condiciones complejas a partir de condiciones simples.

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>
<b>.NOT.</b>	No	.NOT. p
<b>.AND.</b>	Y	a > 0 .AND. a < 11
<b>.OR.</b>	O	opc == 1 .OR. salir != 0

## Si-Entonces. Operaciones lógicas de control

```
1  Algoritmo prueba
2  Escribir "Favor de ingresar un número";
3  Leer x;
4  Escribir "El valor ingresado fue: ",x
5   $x \leftarrow x + 20$ 
6  Escribir "El valor asignado es: ",x
7  Si  $x > 40$  Entonces
8  ..... Escribir "El valor asignado es mayor a 40: "
9  SiNo
10 ..... Escribir "El valor asignado es menor o igual a 40"
11 Fin Si
12 FinAlgoritmo
```



### ! Operaciones logicas de control

```
program prueba
integer x
write(*,*) 'Favor de ingresar un numero:'
read(*,*) x
x = x + 20
write(*,*) 'El valor asignado es:', x
if (x .GT. 40) then
    write(*,*) 'El valor asignado es mayor a 40'
else
    write(*,*) 'El valor asignado es menor o igual a 40'
endif
stop
end
```



## Según o casos. Operaciones lógicas de control

```
Segun variable_numerica Hacer
  opcion_1:
    secuencia_de_acciones_1
  opcion_2:
    secuencia_de_acciones_2
  opcion_3:
    secuencia_de_acciones_3
  De Otro Modo:
    secuencia_de_acciones_dom
Fin Segun
```



```
! Operaciones logicas de control
program prueba
integer op
write(*,*) '¿Como se siente el dia de hoy?'
write(*,*) '1) Estoy contento'
write(*,*) '2) Estoy'
write(*,*) '3) Estoy triste'
read (*,*) op
select case (op)
  case (1)
    write(*,*) 'Siga contento por favor'
  case (2)
    write(*,*) 'Queremos que este contento'
  case (3)
    write(*,*) 'No queremos que este triste'
  case default
    write(*,*) 'No selecciono alguna opcion valida'
end select
stop
end
```

## Mientras. Operaciones lógicas de control

Mientras expresion\_logica Hacer  
secuencia\_de\_acciones  
Fin Mientras

```
program tablaMultiplicar
```

```
c Este programa genera la tabla de multiplicar de un número dado.  
c El número se lee desde la entrada estándar (teclado).
```

```
integer num, cont
```

```
cont = 0
```

```
write (*,*) '----- Tabla de multiplicar -----'
```

```
write (*,*) 'Ingrese un número:'
```

```
read (*,*) num
```

```
write (*,*) 'La tabla de multiplicar del',num,'es'
```

```
do while (cont .LT. 10)
```

```
    cont = cont + 1
```

```
    write (*,*) num,'x',cont,'=',num*cont
```

```
enddo
```

```
stop
```

```
end
```





## Estructura de control repetitiva DO WHILE

La estructura repetitiva (o iterativa)) *do - while* primero valida la expresión lógica y si ésta se cumple (es verdadera)) procede a ejecutar el bloque de instrucciones de la estructura.. Si la condición no se cumple se continúa el flujo normal del programa sin ejecutar el bloque de la estructura, es decir, el bloque se puede ejecutar de cero a ene veces.



# Estructura de control de repetición DO

Lenguaje FORTRAN posee la estructura de repetición **do** la cual permite realizar repeticiones cuando se conoce el número de elementos que se quiere recorrer

```
do var = inicio, fin[, incremento]
```

```
!Bloque de código  
!a ejecutar
```

```
enddo
```



```
program doConIncremento
```

c Este programa debe obtener el promedio de 5 calificaciones  
c ingresadas por el usuario. Las calificaciones se leen  
c desde la entrada estándar (teclado).

```
real*8 sum, calif  
integer veces, cont
```

```
sum = 0  
cont = 1
```

```
do veces = 1, 10, 2  
  write (*,*) 'Suma de calificaciones'  
  write (*,*) 'Ingrese la calificación', cont  
  read (*,*) calif  
  sum = sum + calif  
  cont = cont + 1  
end do
```

```
write (*,*) 'El promedio de las calificaciones es:'  
write (*,*) sum/(cont-1)
```

```
stop  
end
```

## Escapar de todo el ciclo EXIT

EXIT produce la salida inmediata del ciclo

```
do  
    !Bloque de código  
    !a ejecutar  
  
    IF (expresión_lógica) EXIT  
  
    !Bloque de código  
    !a ejecutar  
end do
```



## Escapar de una vuelta (iteración) del ciclo CYCLE

CYCLE detiene la ejecución de la iteración actual y devuelve el control al inicio del ciclo, continuando la ejecución de la iteración siguiente del ciclo.

```
PROGRAM sumaPares

c Programa que suma los primeros 5 números pares a partir de un número dado.

INTEGER numero, suma, contador
suma = 0
contador = 1

WRITE (*,*) 'Ingrese el número inicial'
READ (*,*) numero

DO WHILE (contador .LE. 5)
    numero = numero + 1
c si el número es par se devuelve el control al inicio del ciclo,
c es decir, no se ejecutan las líneas que están debajo de la estructura IF
    IF (MOD(numero, 2) .EQ. 1) CYCLE
    contador = contador + 1
    suma = suma + numero
END DO

WRITE (*,*) 'La suma de los números pares es: ', suma

STOP
END
```