

Programación básica

FORTRAN (PRIMERA PARTE)

Introducción

FORTRAN es uno de los primeros lenguajes de alto nivel. Fue desarrollado por un equipo de programadores de IBM liderado por John Backus, y fue publicado por primera vez en 1957.

60 años después, FORTRAN continúa siendo el lenguaje de programación de preferencia para hacer cálculos numéricos de gran escala, en ciencias e ingeniería.



FORTRAN

FORTRAN es un lenguaje compilado, es decir, existe un programa (llamado compilador) que, a partir de un código en lenguaje FORTRAN, genera un código objeto (ejecutable).

Para crear un programa en FORTRAN se siguen tres etapas principales: edición, compilación y ejecución.

- Edición: Consiste en escribir el código fuente en lenguaje FORTRAN desde algún editor de textos.
- Compilación: A partir del código fuente (lenguaje FORTRAN) se genera el lenguaje maquina (se crea el código objeto o ejecutable).
- Ejecución: El archivo en lenguaje maquina se puede ejecutar en la arquitectura correspondiente (en la maquina donde se compiló el código fuente).

COMPILAR UN CÓDIGO



```
vi holaMundo.f  
gfortran holaMundo.f -o holaMundo  
./holaMundo
```

Versiones FORTRAN

Han surgido varias versiones de FORTRAN desde sus inicios, estas versiones son:

FORTRAN 66

FORTRAN 77

FORTRAN 90 (95)

FORTRAN 2003

FORTRAN 2008

FORTRAN 2010

Tutoriales

En línea

Fortran 77 y 91

<https://www.obliquity.com/computer/fortran/>

Fortran 77 y 91

<http://www.famaf.unc.edu.ar/~vmarconi/numerico1/FortranTutorial.pdf>

Fortran 77 matemático

<http://numat.net/fortran/FT77.pdf>

ALGUNAS REGLAS PARA ESCRIBIR CÓDIGO

Reglas de posición de columnas

FORTRAN no es un lenguaje libre de formato, existen un conjunto de reglas estrictas sobre el formato que debe tener el código fuente, las reglas más importantes son las de posición de columnas.

COLUMNA	USO
Col 1:	Comentarios
Col 1-5:	Declaración de etiquetas (opcional)
Col 6:	Continuación de una línea previa (opcional)
Col 7- 72:	Declaraciones
Col 73-80:	Secuencia de números (opcional raramente utilizado)

ALGUNAS REGLAS PARA ESCRIBIR CÓDIGO

Comentarios

Es una buena practica en cualquier lenguaje de programación realizar comentarios para documentar el programa. En FORTRAN las líneas que comienzan con una 'c' o un asterisco (*) en la primera columna son considerados comentarios, algunos compiladores también aceptan el uso del signo '!' para comentarios.

! Este es un comentario

* Este es otro comentario

c Este es otro comentario

INICIO Y FIN DE UN PROGRAMA

Como lo hemos manejado todo algoritmo, pseudocódigo y diagrama de flujo, deberá tener un inicio y un fin. En el caso de Fortran también

```
1  Algoritmo prueba
2    Escribir "Favor de
3    Leer x;
4    Escribir "El valor
5    x<-x+20
6    Escribir "El valor
7    Si x>40 Entonces
8      Escribir "El v
9    SiNo
10     Escribir "El
11    Fin Si
12  FinAlgoritmo
```

program nombre_programa
lineas de codigo del programa

....

....

....

stop
end

Escribir. Mandar un mensaje a pantalla “Hola Mundo”

```
1  Algoritmo prueba  
2      Escribir "Hola Mundo";  
3  FinAlgoritmo
```

```
program prueba  
write(*,*) 'Hola mundo'  
stop  
end
```

Identificadores y asignación

Memoria RAM

Identificador 1

- valor 1

Identificador 2

- Valor 2

Identificador 3

```
1  Algoritmo prueba
2      Escribir "Favor de ingresar un número";
3      Leer x;
4      Escribir "El valor ingresado fue: ",x
5      x<-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
```

Fortran necesita saber el tipo de dato

Tipos de datos

En este lenguaje hay que establecer el tipo de dato a utilizar al momento de asignar el identificador del dato.

- Caracteres (character): codificación definida por la máquina (caracteres simples).
- Enteros (integer): números sin punto decimal.
- Flotantes (real): números reales de precisión normal.
- Dobles (real*8): números reales de doble precisión.
- Complejos (complex): números complejos.
- Lógicos (logical): Permite representar los valores lógicos verdadero o falso.

EL TIPO DE DATO NO PUEDE CAMBIAR DURANTE LA EJECUCIÓN DEL PROGRAMA

Tipos de datos (uso)

```
integer numeroEntero  
real numeroFlotante  
real*8 numeroFlotanteDoble  
character caracterA  
logical variableLogica .TRUE. .FALSE.
```

Identificadores y asignación

Memoria RAM

TIPO e
Identificador 1

- valor 1

TIPO e
Identificador 2

- Valor 2

TIPO
Identificador 3

```
1  Algoritmo prueba
2  Escribir "Favor de ingresar
3  Leer x;
4  Escribir "El valor ingresado
5  x<-x+20
6  Escribir "El valor asignado
7  Si x>40 Entonces
8  ..... Escribir "El valor asign
9  SiNo
10 ..... Escribir "El valor asign
11 Fin Si
12 FinAlgoritmo|
```

```
program nombre_programa
integer x
x = 20
stop
end
```

¿Constante o variable?

Tipos de datos

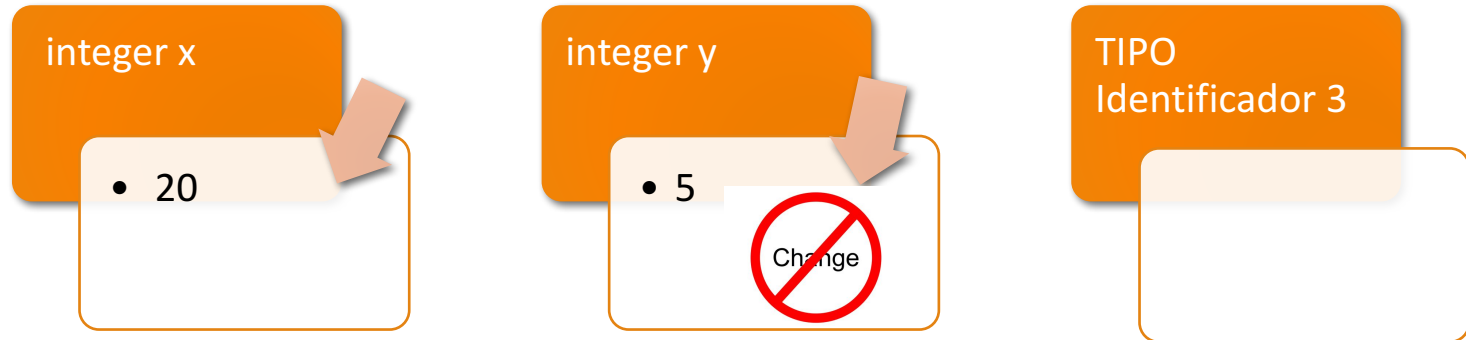
Además de identificar el tipo de dato, es importante saber si el dato es constante o variable. La característica de constante o variable se le da al momento de la primera asignación de dato en memoria.

- Si el dato nunca cambia durante la vida del programa y se asigna al mismo espacio de memoria el dato es constante.
- Si el dato cambia durante la vida del programa y se asigna al mismo espacio de memoria, el dato es variable.

UN DATO NO PUEDE CAMBIAR DE CONSTANTE A VARIABLE EN TIEMPO DE EJECUCIÓN DEL PROGRAMA

¿Constante o variable?

Memoria RAM



```
program nombre_programa
integer x,y
parameter (y=5)
x = 20
stop
end
```

Se pueden convertir por un momento los datos

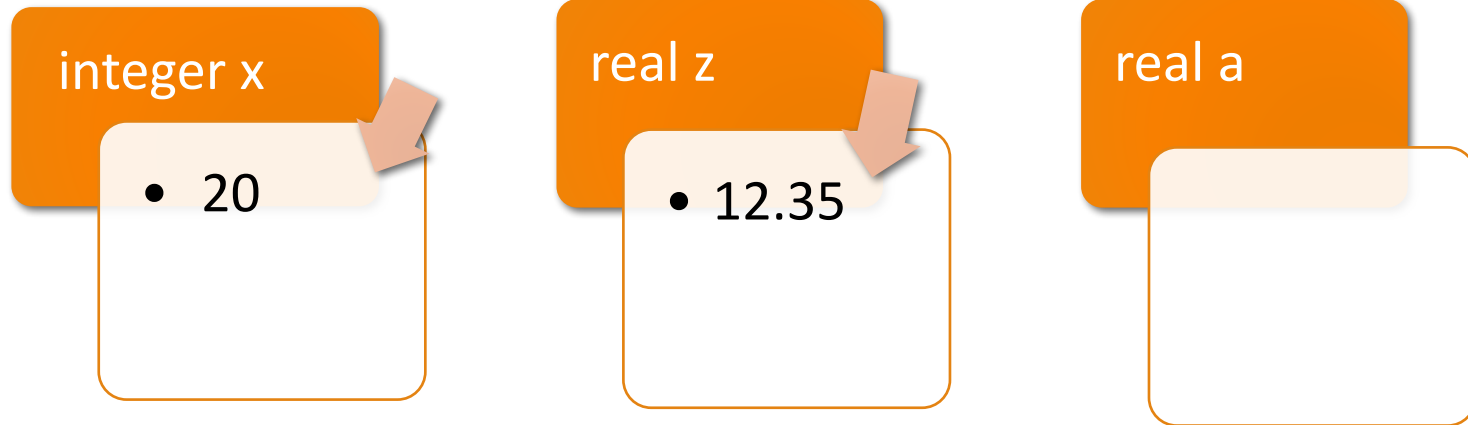
Conversión de datos (cast)

Cuando en una expresión se presentan diferentes tipos de datos, es necesario hacer una conversión entre éstos.

Comando	Función
<i>int</i>	Convierte una variable numérica a entero
<i>real</i>	Convierte una entero a real
<i>dbl</i>	Convierte un entero a real de doble precisión
<i>ichar</i>	Convierte un carácter a entero
<i>char</i>	Convierte un entero al carácter equivalente en ASCII

Cambio de formato (cast)

Memoria RAM



```
program nombre_programa
  integer x
  real z,a
  x = 20
  z = 12.35
  a = real(x) + z
  stop
end
```

Leer . Ingresar información por el usuario

```
1  Algoritmo prueba
2      Escribir "Favor de ingresar un número";
3      Leer x;
4  FinAlgoritmo|
```

```
program prueba
integer x
read(*,*) x
stop
end
```

Escribir. Mandar a pantalla el valor de un identificador concatenado

```
1 Algoritmo prueba
2   Escribir "Favor de ingresar un número";
3   Leer x;
4   Escribir "El valor ingresado fue: ",x
5 FinAlgoritmo
```

```
program prueba
integer x
read(*,*) x
write(*,*) 'El valor fue', x
stop
end
```

Asignar. Asignar un valor a un identificador

```
1 Algoritmo prueba
2   Escribir "Favor de ingresar un número";
3   Leer x;
4   Escribir "El valor ingresado fue: ",x
5   x<-x+20
6   Escribir "El valor asignado es: ",x
7 FinAlgoritmo
```

```
program prueba
integer x
read(*,*) x
x = x + 20
stop
end
```

Poner en un formato las entradas y salidas de datos

Descriptores de formato

Los descriptores de formato sirven para dar un formato especial a un tipo de dato específico, ya sea para lectura o escritura

1.2945 → 1.3

12 → 00012

Descriptores de formato

I	Descriptor para datos de tipo entero
F	Descriptor para datos de tipo real
E	Descriptor de formato exponencial para tipos de dato reales
ES	Descriptor de formato científico para datos de tipo real
L	Descriptor para datos de tipo lógico
A	Descriptor para datos de tipo carácter
X	Descriptor especial de desplazamiento horizontal (por columnas)
T	
/	Descriptor especial de desplazamiento vertical (por líneas)

Descriptores de formato

A continuación, se listan los símbolos usados con los descriptores de formato más comunes:

c	Número de columna.
d	Número de dígitos a la derecha del punto decimal para entrada/salida de datos reales.
m	Número mínimo de dígitos.
n	Número de espacios saltados
r	Factor de repetición: número de veces que se usa un descriptor o grupo de descriptores.
w	Anchura del campo: Número de caracteres de entrada/salida

La sintaxis para especificar un formato es la siguiente:


```
write (*, et)  
et format (descriptores)
```

En donde et es una etiqueta entera única, que puede tomar cualquier valor entre 1 y 99999.

Acciones en un código

Operaciones matemáticas
+ - * /

Operaciones lógicas
Si, sino / Segun



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

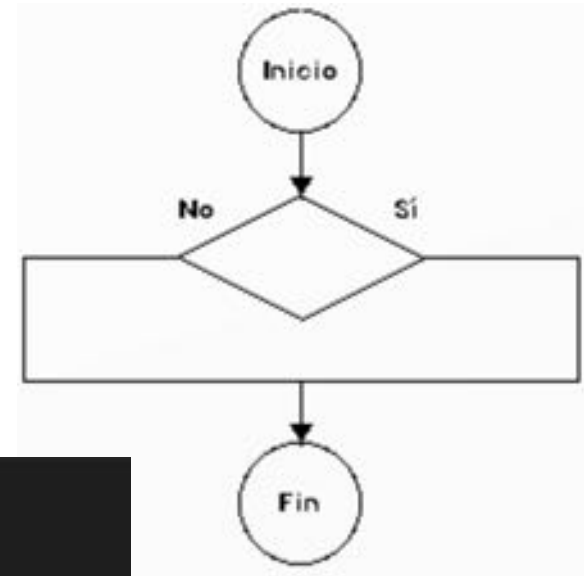
Expresiones lógicas

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

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>
.NOT.	No	.NOT. p
.AND.	Y	a > 0 .AND. a < 11
.OR.	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
```