Informática

Clase 2: Funciones Intrínsecas y Sentencias de Decisión

Mario Merino Martínez
mario.merino@upm.es

Escuela de Ingeniería Aeronáutica y del Espacio
Universidad Politécnica de Madrid

20 de septiembre de 2011

Índice

1 Funciones intrínsecas

2 Sentencias de Decisión

Funciones Intrínsecas

- Fortran posee **funciones matemáticas** "de serie" (**intrínsecas**), al igual que una calculadora científica

Ejemplo

```
sin(x), abs(x), log(x)
```

Cada función toma **uno o más argumentos de entrada** y produce **un resultado**. El **tipo** (`integer`, `real*4`, etc) de argumento de entrada es **importante**. Fortran devolverá un error si no son del tipo adecuado

Funciones Trigonométricas

- Solamente funcionan en radianes.
- Toman **valores reales** (`real*4`, `real*8`) y devuelven un resultado del mismo tipo

`sin(x)`	`asin(x)`
`cos(x)`	`acos(x)`
`tan(x)`	`atan(x)`

- **Convertir** grados en radianes y viceversa multiplicando o dividiendo por $\pi/180$
- El número π puede **obtenerse fácilmente** con `acos(-1d0)` y similares

Transformación de tipos

A veces es necesario pasar un valor numérico de un tipo a otro.
Las **funciones de transformación de tipo** permiten hacerlo:

`int(x)`	Devuelve la parte entera de `x`
`float(x)`	Convierte `x` en `real*4`
`dble(x)`	Convierte `x` en `real*8`

Ejemplo

¿Qué sucede si `alpha` es una variable entera y queremos calcular `sin(alpha)`?
*Primero **tenemos que convertirla** en real*

Otras funciones útiles

Se aplican sobre **números reales**:

`exp(x)` e^x	`sqrt(x)` $\sqrt{x}$
`log(x)` $\ln x$	`log10(x)` $\log_{10} x$

Se aplican sobre **enteros y reales:**

`min(x,y)`	`max(x,y)`
`floor(x)` (redondea hacia abajo)	`ceiling(x)` (hacia arriba)
`int(x)` (parte entera)	`abs(x)`

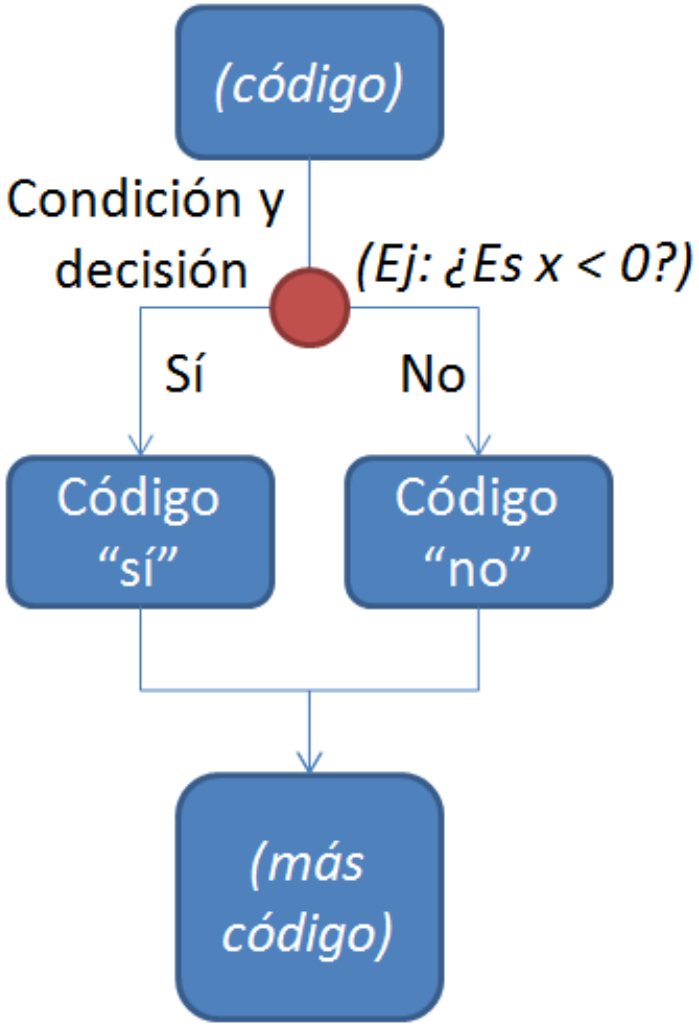
*Imprescindibles

- Recordar que **cada función** admite argumentos de ciertos tipos solamente
- La **notación exponencial** $(1.234 \cdot 10^5)$ para `real*4` es 1.234e5; para `real*8` es 1.234d5
- Recordar que...

En fortran 1≠1.≠1d0

Ejecución condicional

A menudo queremos que el **programa haga algo** solo en determinadas circunstancias, y que **realice** otra tarea en caso contrario. Es decir, llegado cierto punto el ordenador **decidirá** si ejecutar un trozo de código o no **en función de** cómo estén las cosas en ese instante.



Estructura IF

Es la más sencilla:

- Una **condición lógica** es comprobada; **si es cierta**, entonces se ejecuta un bloque de código; **si no**, otro (**else**)
- Si no se pone bloque **else**, entonces no se ejecuta nada si la condición no es cierta

```
! ...
if (CONDICIÓN) then  ! "Si CONDICIÓN es cierta..."
  x = 0d0  ! Bloque a ejecutar
  print*, 'hola!'
else  ! "Si no..." ( parte no necesaria)
  x = 1d0  ! Bloque a ejecutar
end if  ! Fin del "IF"
! ...
```

Condiciones

- Una **condición lógica** es una expresión que al evaluarla resulta en un valor lógico (verdadero `.true.` o falso `.false.`)
- Por ejemplo, una condición lógica es la comparación entre dos variables con signos de relación:

`.EQ.`	`==`	`.NE.`	`/=`
`.GT.`	`>`	`.LT.`	`<`
`.GE.`	`>=`	`.LE.`	`<=`

- Las **condiciones** van siempre entre **paréntesis**: `(x>=2d0)`, `(alpha == 5)`

Condiciones II

- Condiciones **más complejas** pueden construirse uniendo otras con los operadores lógicos `.AND.`, `.OR.`, `.NOT.`

Ejemplo

```
(t < 0d0 .OR. t >= 5.2d0)
 (x > 1. .AND. x < 7.6)
(.NOT. x < 1.)
```

Una condición no es una ecuación ni una inecuación: es una **expresión a evaluar**.
Ej.: `(2 < 1)` es una expresión válida en Fortran, que se evalúa **siempre** como `.false.`

Estructuras avanzadas

Existen **otras estructuras condicionales** en Fortran:

- IF "inline"
- Una lista de ELSE IF
- SELECT CASE

Son más cómodas de utilizar en ciertos casos

IF "inline"

Es una **variante reducida** del **IF ... THEN ... ELSE ... END IF**, útil cuando el bloque a ejecutar es una sola orden y no hay **ELSE**. Por ejemplo,

```
! ...
if (x /= t) then
  x = 0d0
end if
! ...
```

...equivale a...

```
! ...
if (x /= t) x = 0d0
! ...
```

ELSE IF

Es una variante ampliada del **IF ... THEN ... ELSE ... END IF**. Se usa con varias condiciones concatenadas. Por ejemplo,

```
! ...
if (x < 0.) then
  print*, 'hola!'
else if (x <= 2.) then
  t = 1d0
else if (x /= 3.) then
  t = 4d0
else
  print*, 'resto de casos'
end if   ! Fin del "IF"
! ...
```

SELECT CASE

Finalmente, **SELECT CASE** es una versión más ordenada del caso anterior, útil para comparaciones de igualdad. Ojo con los paréntesis.

```
! ...
select case (m)
  case (1)  ! bloque a ejecutar si m == 1
    print*, 'm vale 1'
  case (-2)
    print*, 'm vale -2'
  case default  ! no es necesario
    print*, 'resto de casos'
end select  ! se cierra así
! ...
```

*Imprescindibles II

- Las **estructuras condicionales** permiten establecer que un bloque de código se ejecute sólamente si se verifica una condición
- Las **condiciones** no son ecuaciones ni inecuaciones: son **expresiones a evaluar por fortran** (entre paréntesis): `(a == 5), (x <= 0)`
- Se pueden construir condiciones complejas con **operadores lógicos**
- Fortran proporciona flexibilidad para usar varios tipos de estructura condicional: **IF, IF inline, ELSE IF, SELECT CASE.**
- Os aconsejo **dibujar la recta real** para establecer correctamente las condiciones, sobre todo al principio