Informática

Clase 3: Bucles

Mario Merino Martínez
mario.merino@upm.es

Escuela de Ingeniería Aeronáutica y del Espacio
Universidad Politécnica de Madrid

4 de octubre de 2011

Índice

La necesidad de hacer bucles

- Frecuentemente necesitamos **hacer una misma tarea varias veces** en nuestro programa (Ej.: dados $n = 10$ puntos del plano, decir para cada uno si está dentro o fuera de una circunferencia). No queremos escribir lo mismo 1000 veces.
- En ocasiones, **ni siquiera sabemos cuántas veces se ha de ejecutar una tarea a priori** (Ej.: el usuario elige cuánto vale n). ¿Cómo programaríamos eso si desconocemos cuántas veces repetir el código?

La necesidad de hacer bucles II

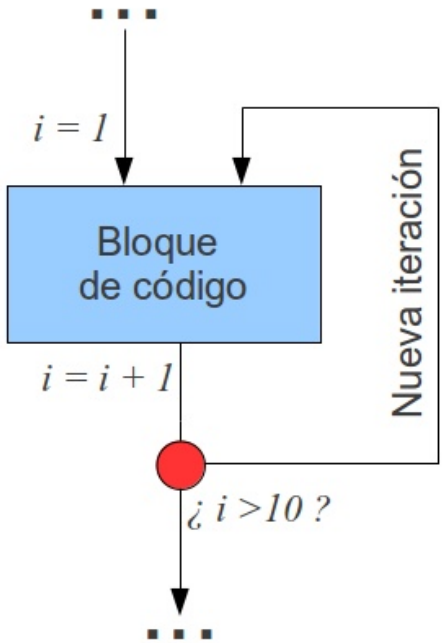
- **Muchos cálculos son repetitivos**. Por ejemplo, un sumatorio $S = \sum_{i=1}^{20} (i^2)$, un producto $P = \prod_{i=1}^{8} (i+1)$, o los factoriales $(i)!$ requieren **evaluar la expresión entre paréntesis varias veces**, variando el valor de i, y **acumular los resultados** en una variable.

En todos estos casos, es conveniente o imprescindible hacer uso de una nueva estructura de Fortran: **el bucle de repetición**

El bucle DO

El **bucle básico** repite un bloque de código n veces. Cuenta con **una variable contador** `integer`, que recorre todos los valores de un rango dado de uno en uno.

```
! ...
do i = 1, 10 ! hacer, desde i = 1
             ! hasta i = 10...
  print *, 'iteracion = ', i ! El
                             ! bloque a repetir
end do
! ...
```



El bucle DO especificando el incremento

La variable contador puede usarse en nuestros cálculos dentro del bucle.
También podemos especificar **cómo incrementa la variable contador**. Incluso podemos usar incrementos negativos

```
! ...
do p = 25, 5, -2 ! Desde p = 25 hasta 5, de -2 en -2...
  print *, 'iteracion = ', p ! El bloque a repetir
end do
! ...
```

El **valor del contador al salir del bucle** es el primero que supera el límite (En el ejemplo, el límite es 5, y el valor será 3)

¿Qué está pasando realmente?

El funcionamiento del bucle "`do i=1,10 ... end do`" se puede resumir como sigue:

1. El programa entra en el bucle. Toma `i=1`, y comprueba que `i`$\leq$10
2. Primera iteración del código, con `i=1`
3. Al llegar a `end do`, el programa **incrementa `i` en 1** y vuelve arriba
4. **Se comprueba que `i`$\leq$10**. Si se cumple, entra en la segunda iteración (`i=2`)
5. Así siempre, **hasta que `i>10`**, que termina el bucle, y continua el resto del programa

Tras el bucle, `i=11`, pues es **el primer valor del contador que incumple la condición**

La sentencia EXIT

Se puede **forzar salir del bucle** antes de su conclusión con **exit**. Se usa dentro de un **if**, de manera que se interrumpa el bucle si se verifica una condición. Al salir del bucle de esta manera, el contador vale lo que valiese al ejecutar **exit**:

```
! ...
do k = 1,10000
  print *, 'Introduce x par'
  read(*,*) x
  if (mod(x,2)==0) then ! mod(x,2) calcula el resto de la
                        división entera x/2
    exit ! Sale del bucle cuando x es par
  else
    print*, 'El numero no es par!'
  end if
end do
! ...
```

*Imprescindibles

- Un **bucle** `do i=i_min, i_max, i_inc ... end do` permite repetir una tarea un número de veces
- El **contador** del bucle es un entero que va desde `i_min` hasta `i_max`, **opcionalmente** en pasos `i_inc` (puede ser negativo)
- Al inicio de cada iteración, **se comprueba que `i` esté dentro del rango**. Si no, finaliza el bucle
- El **contador se incrementa** al llegar al `end do`.
- La sentencia `exit` permite **salir inmediatamente del bucle y continuar** con el resto del programa.
- También es posible hacer **bucles sin contador** (solo `do ... end do`). ¡Pero el bucle será infinito si no ponemos un `exit` adecuado!

Realizar un sumatorio

Los **sumatorios y los productos** requieren una variable de acumulación que hay que **inicializar** antes del bucle. En los sumatorios ha de ponerse a cero. Por ejemplo,

$$S = \sum_{i=a}^{b} i\,(i+1) = a\,(a+1) + (a+1)\,(a+2) + \ldots + b\,(b+1)$$

```
! ...
s=0 ! inicialización de s, variable de acumulación
do i = a,b ! Límites inferior y superior del sumatorio
  s = s + i*(i+1) ! vamos añadiendo cada i(i+1) a s
end do
! ...
```

Realizar un producto

En un **producto**, la variable de acumulación ha de inicializarse a uno, para poder multiplicar por ella. Por ejemplo,

$$P = \prod_{i=1}^{n} i^2 = 1^2 \cdot 2^2 \cdot 3^2 \cdot \ldots \cdot n^2$$

```
! ...
p=1d0 ! inicialización de p, variable de acumulación
do i = 1,n ! Límites inferior y superior del producto
  p = p * i**2 ! añadimos cada i^2 multiplicando a p
end do
! ...
```

*Imprescindibles II

- En sumatorios y productos, **SIEMPRE** se usa una **variable de acumulación**
- La **variable de acumulación** hay que inicializarla ANTES de compenzar el bucle del sumatorio/producto
- En **sumatorios** inicializamos con 0, en **productos** con 1
- Es posible **anidar bucles uno dentro de otro** para realizar operaciones más complicadas, e.g.:

$$\prod_{i=1}^{n} \sum_{j=1}^{i} (i+j)$$

En tal caso, se usan **varias variables contador** y **varias variables de acumulación**, que hay que inicializar justo antes del bucle correspondiente (ver ejercicios)