DEPARTAMENTO DE CIENCIAS DE LA ATMÓSFERA Y LOS OCÉANOS

LABORATORIO DE PROCESAMIENTO DE INFORMACION METEOROLÓGICA

Lenguaje de Programación R

2do Cuatrimestre 2023 - Moira E. Doyle

INTRODUCCIÓN A LA LÓGICA DE LA PROGRAMACIÓN

- Expresiones lógicas
- Operadores lógicos
- Operadores relacionales
- Diagramas de Flujo
- **Estructuras de control repetitivas (Funciones iterativas FOR,WHILE,REPEAT)**
- Estructura condicional simple (Función IF – ELSE – IFELSE, SWITCH)
- **Interrupciones de ciclos (BREAK, NEXT,RETURN)**

CICLOS

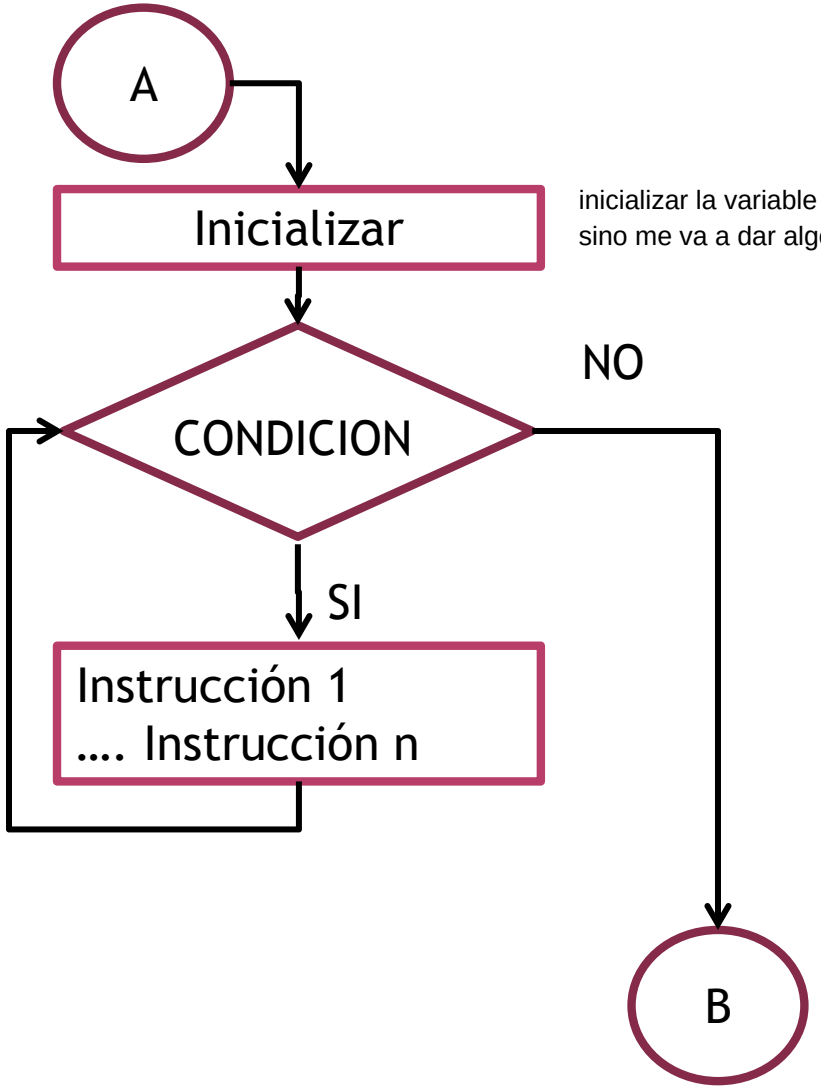
Varios tipos de ciclos o repeticiones: repeticiones por un número determinado de veces, repeticiones mientras se cumple una condición y repeticiones infinitas.

Funciones: ***FOR - WHILE- REPEAT***

condicion repite instrucciones dentro del ciclo
la cantidad de veces que dice el ciclo

la diderencia con if esque if se repite una sola vez

DIAGRAMA FLUJO FOR



inicializar la variable en el valor adecuado xq
sino me va a dar algo erroneo

FUNCIÓN *FOR*

REPETICIONES POR UN NÚMERO DETERMINADO DE VECES

Existe una construcción repetitiva de la forma

for (nombre in expr 1) expr 2

ej: for(i in 1:5) es lo mismo q hacer v=v+1

Si expr 2 es una puede escribirse en un solo renglón

nombre es la variable de control de iteración,
expr 1 es un vector (a menudo de la forma m:n)
expr 2 es una expresión, a menudo agrupada, en cuyas sub-expresiones puede aparecer la variable de control, nombre.

expr 2 se evalúa repetidamente conforme ***nombre*** recorre los valores del vector ***expr 1***.

```
letras <- c("c", "l", "i", "M", "T", "A")
for (i in 1:6) {
+     print(letras[i])
+ }
[1] "c"
[1] "l"
[1] "i"
[1] "M"
[1] "T"
[1] "A"
```

Si expr 2 es una pueden omitirse los { }

```
for (J in letras) {
    print(J)
}
[1] "c"
[1] "l"
[1] "i"
[1] "M"
[1] "T"
[1] "A"
```

```
for (i in seq_along(letras)) {
    print(letras[i])
}
[1] "c"
[1] "l"
[1] "i"
[1] "M"
[1] "T"
[1] "A"
```

genera una secuencia de enteros de acuerdo con el número de elementos que tenga el objeto que se le de como argumento

IGUAL RESULTADO EN LOS 3

3 formas distintas de hacer lo mismo

si yo quiero hacerlo general tengo que poner length(variable)
for(i in 1:length(a)) suma[i]=a[i]+b[i]

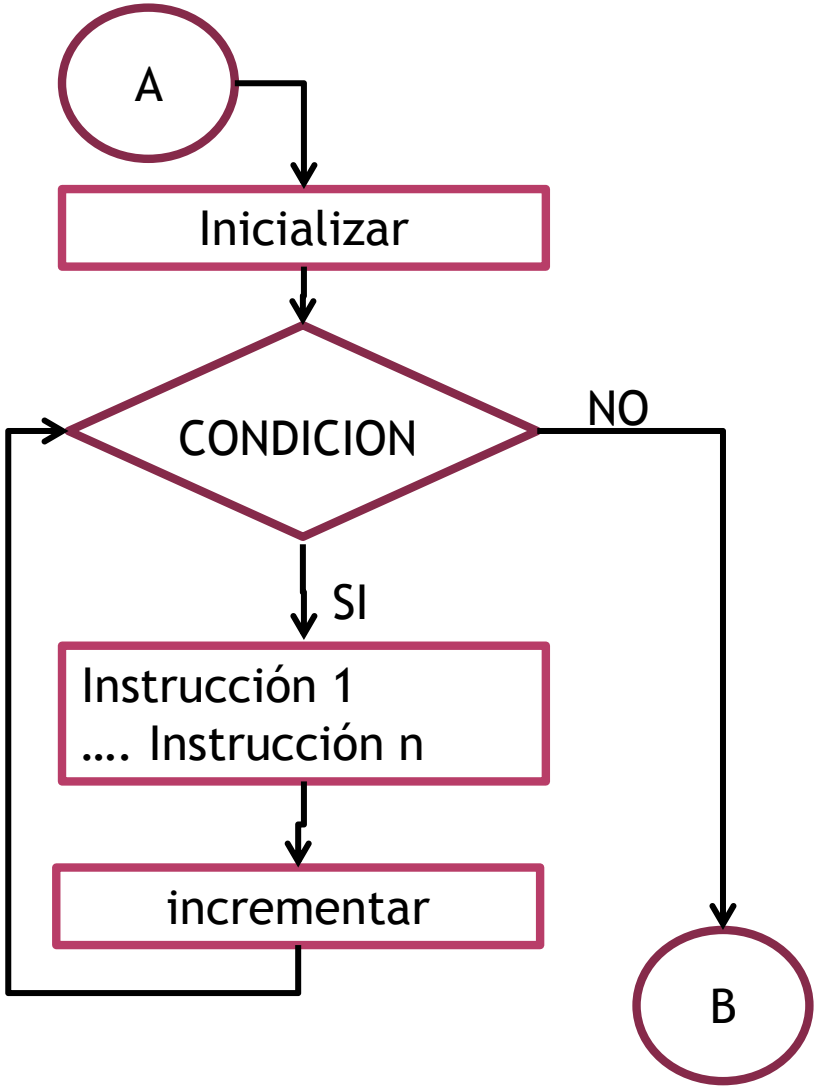
EJERCICIO

Sea:
a la secuencia de valores del 1 al 10
De que formas puedo crear la variable ***a***?
b los primeros 10 números pares
De que formas puedo crear la variable ***b***? Que pasa con el 0?

Calcular usando el comando ***for***

- La suma de **a** y **b** (guardar en un nuevo vector de 10 elementos)
- El producto entre los elementos 1,3 ,5 y 7 de **a** y **b** (guardar en un nuevo vector de 4 elementos)
- El cuadrado de cada elemento de ***a*** (guardar en un nuevo vector de 10 elementos)
- La raíz cuadrada de cada elemento de ***b*** (guardar en un nuevo vector de 10 elementos)
- Mostrar por pantalla los resultados, indicando lo calculado en cada caso

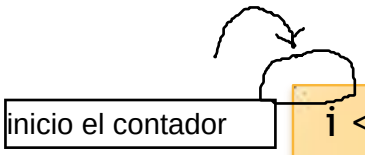
DIAGRAMA FLUJO WHILE



FUNCIÓN *WHILE*

REPETICIONES MIENTRAS SE CUMPLE UNA CONDICIÓN

while (condición) expr




```
i <- 1
while (i <= 6) {
    print(letras[i])
    i <- i + 1
}
[1] "c"
[1] "l"
[1] "i"
[1] "M"
[1] "T"
[1] "A“
```

La salida de este ejemplo es la misma que la de los ejemplos anteriores.
En este caso, si no se tiene cuidado en el manejo del índice i, involucrado en la condición, se puede dar lugar a un ***ciclo sin salida***.

EJERCICIO

Sea:
a la secuencia de valores del 1 al 10
b los primeros 10 números pares
Crear ***a*** y ***b*** con métodos diferentes al caso ***for***

Calcular usando el comando ***while***

- La suma de ***a*** y ***b*** (guardar en un nuevo vector de 10 elementos)
- El producto entre los elementos 1,3 ,5 y 7 de ***a*** y ***b*** (guardar en un nuevo vector de 4 elementos)
- El cuadrado de cada elemento de ***a*** (guardar en un nuevo vector de 10 elementos)
- La raíz cuadrada de cada elemento de ***b*** (guardar en un nuevo vector de 10 elementos)
- Mostrar por pantalla los resultados, indicando lo calculado en cada caso

INTERRUPCIONES DEL FLUJO NORMAL DE LOS CICLOS

El flujo normal de los ciclos se puede interrumpir básicamente por medio de tres instrucciones diferentes:

- ✓ **Break** se puede utilizar en el interior de cualquier ciclo para forzar su interrupción
- ✓ Next
- ✓ Return.

set.seed hace que el numero al azar sea siempre el mismo (CREO)

```
set.seed(140)              # el argumento puede ser cualquier número
aprox <- 0.003             # Valor determinante para la salida del ciclo
Y_ini <- 2.7               # Supuesto valor inicial de Y
for (iter in 1:1000) {     # aseguro no más de 1000 iteraciones
# Procedimiento para calcular la siguiente Y, que simularemos mediante generador aleatorio:
Y <- Y_ini + 0.008*rnorm(1)
# La condición de salida:
if (abs(Y - Y_ini) <= aprox)
break                      # Uso del break para salir del ciclo
# Preparamos para la siguiente iteración
Y_ini <- Y
}
paste0("Y_ini: ", Y_ini, ", Y: ", Y, ", Num.iter: ", iter)
 [1] "Y_ini: 2.76443400590741, Y: 2.76582777768031, Num.iter: 8"
```

rnorm me genera numeros al azar que provienen de una dist normal

Concatena Vectores

FUNCIÓN *REPEAT*
REPETICIONES INFINITAS

La instrucción no tiene condición de salida o interrupción, el resultado que la instrucción produciría en sí misma sería una repetición interminable

Existen facilidades para que desde el interior del bloque de expresiones que se repiten, se obligue la interrupción del ciclo. Ejemplo: ***break***

repeat expr

```
x <- 1
repeat {
    print(x)
    x = x+1
    if (x == 6){
        break
    }
}
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

para cortar la secuencia porque el repeat no tiene forma de cortar

si quisiera incluir el 6, el x=x+1 iria despues del if

INTERRUPCIONES DEL FLUJO NORMAL DE LOS CICLOS

El flujo normal de los ciclos se puede interrumpir básicamente por medio de tres instrucciones diferentes:

- ✓ Break
- ✓ **Next** en vez de salir de un ciclo, solamente impide la ejecución de las instrucciones siguientes y regresa al principio del ciclo para ejecutar la siguiente iteración
- ✓ Return.

```
for (i in 1:7) {
if (3 <= i && i <= 5)
next
print(i)
}
[1] 1
[1] 2
[1] 6
[1] 7
```

INTERRUPCIONES DEL FLUJO NORMAL DE LOS CICLOS

- ✓ Break
- ✓ Next
- ✓ **Return** Está asociada funciones y su propósito es interrumpir u obligar la salida de la función, entregando, opcionalmente, como resultado de la función un valor si se da como argumento del **return**.

```
fibbonacci <- function(y) {if (y==0 | y==1)     # identifica si un cierto elemento está dentro
                                                #los valores del vector.
      return (1)
   F0 <- 1; F1 <- 1; i <- 2
  repeat {
        s <- F0 + F1              # Suma de los fib anteriores
        if (i == y)               # Ya es el que se busca
            return (s)            # Sale hasta afuera de la función
                                  # recorremos los últimos dos próximos números
        F0 <- F1
        F1 <- s
        i <- i+1                        # incrementamos el índice
    } }

fibbonacci(8)              #calculo para el valor 8
[1] 34
```

EJEMPLOS PRÁCTICOS

Ejemplos 4 a 6 de la Teórica

1. Calcular la suma de los N primeros términos de la sucesión 1, $2x$, $3x^2$, $4x^3$, … *(uso de FOR)*

2. Cuál es el mayor valor de N tal que la suma $1^2 + 2^2 + 3^2 + \ldots + N^2$ sea menor que 10 *(uso de WHILE)*

3. Cuál es el mayor valor de N tal que la suma $1^2 + 2^2 + 3^2 + \ldots + N^2$ sea menor que 100 y N sea menor que 5 (*uso de WHILE Y BREAK)*