Estructuras de control

Las estructuras de control nos permiten crear programas con múltiples caminos posibles a seguir, dependiendo de ciertas condiciones a comprobar. Veamos qué estructuras tenemos disponibles en R.

1. Estructuras selectivas

R dispone de las típicas estructuras selectivas de otros lenguajes.

1.1. La estructura *if/else*

Por un lado, tenemos la estructura **if** e **if..else** para poder decidir entre uno o varios caminos, según la condición que se cumpla:

```
if(condicion1)
{
    # Codigo
} else if(condicion2) {
    # Codigo
} else {
    # Codigo
}
```

> **NOTA**: es IMPORTANTE la colocación de las llaves de apertura y cierre en la misma línea que el siguiente *else*, ya que de lo contrario podemos obtener un error de sintaxis.

1.2. La estructura *switch/case*

Alternativamente, también disponemos de la estructura **switch/case**, aunque con una sintaxis diferente a lo habitual. Comenzaremos con la partícula `switch` indicando, entre paréntesis, el valor que queremos evaluar, y cada una de las acciones a realizar en cada posible caso. Veamos algunos ejemplos:

```
# Asignamos a la variable s un texto según el valor
# del primer parámetro
s <- switch(
    4,
    "Uno",
    "Dos",
    "Tres",
    "Cuatro",
    "Cinco"
)
# s valdrá "Cuatro"
```

```
# Hacemos una acción u otra según cuánto vale texto
texto = "p"
switch(
    texto,
    "a" = cat("Vale a"),
    "b" = cat("Vale b"),
    "p" = cat("Vale p"),
    "z" = cat("Vale z")
)
# Se mostrará "Vale p"
```

2. Estructuras repetitivas o bucles

R dispone de tres tipos de bucles que podemos utilizar:

2.1. El bucle *repeat*

El bucle `repeat` permite repetir un conjunto de instrucciones, y terminar cuando se cumpla una determinada condición

```
repeat
{
    # Instrucciones
    if(condicion)
    {
        break
    }
}
```

Ejemplo:

```r
# Contador del 1 al 10
n <- 1
repeat
{
    print(n)
    n <- n+1
    if(n > 10)
    {
        break;
    }
}
```

2.2. El bucle *while*

Similar a otros lenguajes, ejecuta un conjunto de instrucciones mientras se cumpla una determinada condición. El ejemplo anterior podría quedar así:

```r
n <- 1
while(n <= 10)
{
    print(n)
    n <- n+1
}
```

2.3. El bucle *for*

El bucle *for* se emplea para recorrer todas las opciones de una lista de valores:

```r
for(n in 1:10)
{
    print(n)
}

datos <- c("uno", "dos", "tres")
for(x in datos)
{
    print(x)
}
```

> **Ejercicio 1**:
>
> Crea un programa `mediaNotas.r` que le pida al usuario una secuencia de 10 notas numéricas. Deberá indicar al final la nota nominal (SOBRESALIENTE, NOTABLE, APROBADO, SUSPENSO) dependiendo de la

> media de esas 10 notas
>
> **AYUDA:** vídeo con solución del ejercicio

3. Gestión de excepciones

Al igual que ocurre en otros lenguajes como Java o Python, en R pueden producirse errores durante la ejecución de un programa, y las excepciones son un mecanismo que nos permite controlar estos errores, mostrar un mensaje apropiado y continuar con la ejecución.

Para ello disponemos, entre otras opciones, de la función `tryCatch`, que recibe como parámetro el bloque de código que se debe ejecutar con normalidad. Adicionalmente, se le pasan unos parámetros `error`, `warning` y/o `message` con el código a ejecutar dependiendo del tipo de error que se haya producido.

El siguiente ejemplo trata de obtener la raíz cuadrada de lo que diga el usuario. Si no puede se ejecutará el segundo bloque de código para mostrar un error:

```
tryCatch({
  cat("Escribe un número:")
  valor <- sqrt(as.integer(readLines("stdin", n=1)))
  print(valor)
}, error = function(e) {
  cat("Se ha producido un error:", conditionMessage(e), "\n")
}, warning = function(w) {
  cat("Advertencia:", conditionMessage(w), "\n")
}, message = function(w) {
  cat("Mensaje:", conditionMessage(w), "\n")
}, finally = {
  cat("Finalizando tryCatch\n")
})
```

Como decimos, en función del error producido se ejecutará uno de los parámetros (*error*, *warning* o *message*), y con la función `conditionMessage` podemos mostrar el mensaje de error en cuestión. El bloque `finally` podemos emplearlo para lo que queramos que se ejecute al finalizar la ejecución del bloque, haya sido exitosa o no.