Relación C

Estructuras de control: condicionales y bucles

Nota

Esta relación practica el uso de estructuras selectivas (`if-else`, `switch`) y repetitivas (`for`, `while`, `do-while`, `for-of`, `for-in`, `forEach`) en JavaScript, con especial énfasis en contextos estadísticos y matemáticos familiares para el alumnado.

Ejercicio 1. Escriba una función `clasificarNota(nota)` que reciba un número y devuelva la calificación en texto según el sistema universitario español: Suspenso (0–4,9), Aprobado (5–6,9), Notable (7–8,9), Sobresaliente (9–9,9) y Matrícula de Honor (10). Implemente al menos dos versiones: una con `if-else` anidado y otra con `switch`. Pruebe la función con varios valores, incluyendo valores límite y valores fuera de rango (negativos o mayores que 10).

Ejercicio 2. Dado un array de 10 números enteros (algunos positivos, negativos y ceros), escriba un bucle que los recorra y, para cada elemento, muestre por consola:

- Si es positivo, negativo o cero.
- Si es par o impar (solo para los distintos de cero).
- Si es divisible por 3.

Implemente el mismo comportamiento usando `for` clásico, `for-of` y `forEach` con función flecha.

Ejercicio 3. Implemente con JavaScript los siguientes cálculos, mostrando los resultados con `document.write()`:

- Tabla de multiplicar del 7, como lista HTML (`<ol>`).
- Suma de los primeros n números naturales (n pedido al usuario con `prompt`), comprobando también con la fórmula $\frac{n(n+1)}{2}$.
- Los primeros 10 términos de la sucesión de Fibonacci.
- Factorial de un número introducido por el usuario.

Ejercicio 4. Cree una función `esPrimo(n)` que devuelva `true` si `n` es primo y `false` en caso contrario. Use un bucle `for` con `break` para detener la búsqueda en cuanto encuentre un divisor. A continuación, genere con `document.write()` una lista de todos los números primos entre 2 y 200.

Ejercicio 5. Implemente la criba de Eratóstenes en JavaScript para encontrar todos los primos hasta un límite N introducido por el usuario (`prompt`). Use un array de booleanos y bucles anidados. Muestre los primos encontrados en el documento y también su cantidad total.

Ejercicio 6. Escriba un programa que genere una tabla HTML de multiplicar completa (del 1 al 10 por el 1 al 10) usando bucles anidados. Aplique estilos básicos con `document.write()` de modo que:

- La primera fila y la primera columna sean cabeceras (`<th>`).
- Las celdas de la diagonal estén resaltadas con algún color de fondo.

Ejercicio 7. Dado el array de objetos estudiante de la Relación B (ejercicio 6), recórralo con distintos tipos de bucle y realice las siguientes operaciones:

- Con `for` clásico: imprimir nombre y nota de cada estudiante.
- Con `for-of`: construir una cadena con todos los nombres separados por comas.
- Con `for-in` sobre el primer objeto: listar todos los campos del objeto.
- Con `forEach` y función flecha: mostrar en el documento una lista HTML de estudiantes aprobados.

Ejercicio 8. Implemente el algoritmo de búsqueda binaria en un array ordenado. La función `busquedaBinaria(arr, objetivo)` debe devolver el índice del elemento buscado, o -1 si no se encuentra. Use un bucle `while`. Pruébela con un array de 20 números ordenados y varios valores objetivo. Compare el número de iteraciones con una búsqueda lineal.

Ejercicio 9. Use `do-while` para implementar un juego de adivinanza: el programa genera un número aleatorio entre 1 y 50 con `Math.random()` y el usuario lo intenta adivinar usando `prompt()` repetidamente hasta acertar. El programa debe indicar si el número a adivinar es mayor o menor que la respuesta del usuario. Al final, muestre cuántos intentos necesitó.

Ejercicio 10. Escriba una función `contarPalabras(texto)` que reciba una cadena y devuelva un objeto donde las claves sean las palabras únicas (en minúscula) y los valores sean el número de veces que aparece cada palabra. Pruébela con un párrafo de texto. Use el método `split()` y un bucle para construir el objeto. Muestre el resultado ordenado por frecuencia (de mayor a menor) con `document.write()`.

Ejercicio 11. Implemente el método de ordenación de burbuja (*bubble sort*) en una función `burbuja(arr)` que ordene un array numérico de menor a mayor. Use bucles anidados y la sentencia `break` para optimizar el algoritmo cuando el array ya esté ordenado. Pruébela con un array de 15 números aleatorios y compare el resultado con `Array.prototype.sort()`.

Ejercicio 12. Cree una función `calcularEstadisticos(datos)` que reciba un array de números y, usando estructuras de control, calcule: media, mediana (ordene el array y busque el elemento central), moda (el valor más frecuente) y desviación típica. Devuelva un objeto con los cuatro valores. Pruébela con el array `[4, 7, 13, 2, 7, 7, 5, 10, 8, 3, 7, 12]` y muestre los resultados en el documento.

Ejercicio 13. Cree un programa que simule el lanzamiento de un dado de seis caras n veces (con n introducido por el usuario, entre 100 y 10 000). Cuente la frecuencia de cada cara usando un objeto, calcule la frecuencia relativa de cada una y muestre los resultados en una tabla HTML. Además, indique si la diferencia entre la frecuencia relativa observada y la teórica (1/6) es menor de 0,01 para cada cara.

Ejercicio 14. Implemente tres funciones para trabajar con la serie de Taylor del coseno:

$$\cos(x) \approx \sum_{k=0}^{n} \frac{(-1)^k \cdot x^{2k}}{(2k)!}$$

La primera calcula el factorial de n. La segunda calcula la aproximación para un x y un número de términos n dados. La tercera genera en el documento una tabla HTML que muestre, para $x = \pi/4$, la aproximación con 1, 2, 3, ..., 10 términos junto al error respecto a `Math.cos(x)`.