// MARK: Ejercicio 1

Escribe una función **operacion** que reciba dos enteros y un closure que indique la operación a realizar (+, -, *, /).
El closure debe devolver el resultado.
Prueba la función con diferentes operaciones.

———

// MARK: Ejercicio 2

Crea una función **esMayorQue** que reciba dos enteros y un closure que devuelva true si el primer número es mayor que el segundo.
Usa la función con diferentes pares de números.

———

// MARK: Ejercicio 3

Crea una función **procesarCadenas** que reciba un arreglo de String y un closure que combine dos cadenas en una.
La función debe recorrer el arreglo y combinar todos los elementos usando el closure, devolviendo el resultado final.

Ejemplo de uso:

- Arreglo: ["Swift", "es", "genial"]
- Closure: concatenar con espacio " "
- Resultado: "Swift es genial"

———

// MARK: Ejercicio 4

Escribe una función **compararStrings** que reciba dos cadenas de texto y un closure que defina la comparación (por ejemplo: si son iguales, si una es más larga que la otra, etc.).
El resultado debe imprimirse en consola.

———

// MARK: Ejercicio 5

Crea una función **transformarTexto** que reciba un String y un closure.
El closure debe transformar el texto (ejemplo: pasarlo a mayúsculas, agregarle un prefijo, invertirlo).
Devuelve el resultado.

———

// MARK: Ejercicio 6

Escribe una función **filtrarNumeros** que reciba un arreglo de enteros y un closure que indique si un número debe incluirse o no.
Devuelve un nuevo arreglo con los números filtrados. (Implementa el ciclo for, no uses filter).

———

// MARK: Ejercicio 7

Crea una función **modificarArreglo** que reciba un arreglo de enteros y un closure que transforme cada número (ejemplo: multiplicar ×2, restar 1).
Devuelve el nuevo arreglo. (Investiga como usar map).

———

// MARK: Ejercicio 8

Escribe una función **buscarElemento** que reciba un arreglo de enteros, un número a buscar y un closure que indique la condición de igualdad.
Devuelve true si encuentra el número, false en caso contrario.

———

// MARK: Ejercicio 9

Crea una función **ordenarArreglo** que reciba un arreglo de enteros y un closure que defina el criterio de ordenamiento (ejemplo: ascendente, descendente).
No uses sorted, implementa el algoritmo con ciclos (for / while).