# Kotlin tabla de ejercicios

**Rubén Agra Casal**

**Ejercicio 1**

*Ejercicio 1:*

*Crea una función llamada* `serieFibonacci` *que genere la serie de Fibonacci hasta un número límite proporcionado por el usuario. Utiliza un bucle* `while` *para construir la serie.*

**Pista:** La serie de Fibonacci se genera sumando los dos números anteriores.

Comienza con 0 y 1.

***Respuesta Alumno***

```
// Ejercicio 1. (Sucesión de Fibonacci)
  fun sucesionFibonacci() {
      val scanner = Scanner(System.`in`)
      println("Introduzca el número límite: ")
      val numeroLimite = scanner.nextInt()

      var numeroAnterior = 0
      var numero = 1
      var suma = 1

      while (suma <= numeroLimite) {
          println(suma)
          suma = numero + numeroAnterior
          numeroAnterior = numero
          numero = suma
      }
```

}

## *Ejercicio 2:*

Escribe una función llamada `calcularFactorial` que calcule el factorial de un número ingresado por el usuario utilizando un bucle `for`. El factorial de un número `n` se calcula como `n * (n - 1) * (n - 2) * ... * 1`.

**Pista:** El factorial de 0 es 1.

***Respuesta Alumno***

```
// Ejercicio 2. (Calcular factorial)
 fun calcularFactorial() {
    val scanner = Scanner(System.`in`)
    println("Introduzca un número: ")
    val numero = scanner.nextInt()
    var operacion = 1
    for (i in 1..numero) {
       operacion *= i
    }

    println("El factorial de $numero es: $operacion")
 }
```

## *Ejercicio 3:*

Crea una función llamada `sumarDigitos` que sume los dígitos de un número entero ingresado por el usuario. Utiliza un bucle `do-while` para realizar la suma.

**Pista:** Puedes usar el operador de módulo (%) para obtener los dígitos.

***Respuesta Alumno***

```
// Ejercicio 3. (Sumar dígitos)
  fun sumarDigitos() {
      val scanner = Scanner(System.`in`)
      println("Introduce un número entero:")
      var numero = scanner.nextInt()
      var suma = 0

      do {
         suma += numero % 10  // Suma el último dígito
         numero /= 10         // Elimina el último dígito
      } while (numero > 0)      // Continúa mientras queden dígitos

      println("La suma de los dígitos es: $suma")

  }
```

## *Ejercicio 4:*

Escribe una función llamada `numerosPrimosEnRango` que encuentre y muestre todos los números primos en un rango dado (por ejemplo, del 1 al 100). Utiliza bucles `for` anidados.

**Pista:** Un número es primo si no tiene divisores más que 1 y él mismo.

***Respuesta Alumno***

```
// Ejercicio 4. (Lista de números primos)
  fun numerosPrimosEnRango(inicio: Int, fin: Int) {
    println("Números primos entre $inicio y $fin:")

    for (numero in inicio..fin) {
      if (numero < 2) continue
      var esPrimo = true
      for (i in 2 until numero) {
        if (numero % i == 0) {
          esPrimo = false
          break
        }
      }
      if (esPrimo) {
        println(numero)
      }
    }
  }
```

## *Ejercicio 5:*

Crea una función llamada `tablasMultiplicarConFormato` que imprima las tablas de multiplicar del 1 al 10 en un formato organizado. Utiliza bucles `for` anidados.

**Pista:** Usa `String.format()` o interpolación de cadenas para dar formato a la salida.

***Respuesta Alumno***

```
// Ejercicio 5. (Tablas de multiplicar del 1 al 10)
 fun tablasMultiplicarConFormato() {
    for (i in 1..10) {
        println("Tabla de multiplicar del $i:")
        for (j in 1..10) {
            val resultado = i * j

            println(String.format("%d x %d = %d", i, j, resultado))
        }
        println()
    }
 }
```

## *Ejercicio 6:*

Desarrolla un juego sencillo de adivinanza donde el usuario tiene que adivinar un número generado aleatoriamente entre 1 y 100. La función `jugarAdivinanza` utilizará un bucle `while` para permitir que el usuario siga adivinando hasta que lo haga correctamente.

**Pista:** Utiliza `Random` para generar el número y `Scanner` para leer la entrada del usuario.

```
import kotlin.random.Random
```

***Respuesta Alumno***

```
fun adivinarNumero() {
    val scanner = Scanner(System.`in`)
    val randomGenerator = Random()
    val numeroAleatorio = randomGenerator.nextInt(1001)
    var intentos = 0
    var numeroSeleccionado: Int

    println("Adivina el número")
    println("------------------")
    while (true) {
        println("Escribe un número: ")
        numeroSeleccionado = scanner.nextInt()
        intentos++

        if (numeroSeleccionado < numeroAleatorio) {
            println("Fallo, el número es mayor.")
        } else if (numeroSeleccionado > numeroAleatorio) {
            println("Fallo, el número es menor.")
        } else {
            println("Número encontrado: $numeroAleatorio")
```

```
            println("Intentos: $intentos")
            break
        }
    }
}
```

### *Ejercicio 7:*

Crea una función llamada `trianguloAsteriscos` que imprima un triángulo de asteriscos de una altura dada. Utiliza bucles `for` para construir el triángulo.

**Pista:** En cada fila, imprime espacios en blanco y luego los asteriscos.

```
// Ejercicio 7. (Dibujar un triángulo con asteriscos)
fun trianguloAsteriscos() {
    println("Introduce la altura del triángulo: ")
    val scanner = Scanner(System.`in`)

    val altura = scanner.nextInt()

    for (i in 1..altura) {
        for (j in 1..(altura - i)) {
            print(" ")
        }
        for (k in 1..(2 * i - 1)) {
            print("*")
        }
        println()
    }
}
```

## *Ejercicio 8:*

Crea una función llamada `contarFrecuenciaCaracteres` que reciba una cadena y cuente la frecuencia de cada carácter en ella.

***Respuesta Alumno***

```
// Ejercicio 8. (Contar frecuencia de caracteres)
    fun contarFrecuenciaCaracteres(cadena : String) : Map<Char, Int> {
        val frecuencia = mutableMapOf<Char, Int>()
        for (caracter in cadena) {
            if (frecuencia.containsKey(caracter)) {
                frecuencia[caracter] = frecuencia[caracter]!! + 1
            } else {

                frecuencia[caracter] = 1
            }
        }
        return frecuencia
    }
```

## *Ejercicio 9:*

Escribe una función llamada `tablaPotencias` que imprima las potencias de 2 desde $2^0$ hasta $2^{10}$. Utiliza un bucle `for` para realizar los cálculos.

***Respuesta Alumno***

```
// Ejercicio 9. (Tabla de potencias)
    fun tablaPotencias() {
        println("Tabla de potencias")
        println("==================")
        for(i in 0..10) {
            var operacion = Math.pow(2.0, i.toDouble())
```

```
        println("2 ^ $i = $operacion")
    }
}
```

## *Ejercicio 10:*

Desarrolla una función llamada `maximoYMinimo` que reciba una lista de números enteros y determine el valor máximo y mínimo. Utiliza un bucle `for` para recorrer la lista.

***Respuesta Alumno***

```
// Ejercicio 10. (Número máximo y número mínimo)
fun maximoYMinimo(numeros : Array<Int>) {
    var minimo = numeros[0]
    var maximo = numeros[0]

    for (i in 1 until numeros.size) {
        if (numeros[i] < minimo) {
            minimo = numeros[i]
        }
        if (numeros[i] > maximo) {
            maximo = numeros[i]
        }
    }
    println("El número máximo de la lista es $maximo")
    println("El número mínimo de la lista es $minimo")
}
```

## *Ejercicio 11:*

Crea una función llamada `cuadradoAsteriscos` que imprima un cuadrado de asteriscos de tamaño `n`, donde `n` es un número ingresado por el usuario. Utiliza bucles `for` anidados.

***Respuesta Alumno***

```
// Ejercicio 11. (Cuadrado astericos)
    fun cuadradoAsteriscos() {
        val scanner = Scanner(System.`in`)
        println("Ingrese el tamaño del cuadrado (n): ")
        val n = scanner.nextInt()

        for (i in 1..n) {
            for (j in 1..n) {
                print("* ")
            }
            println()
        }
    }
```