

Ejercicio 1: Sumatoria de números primos en un rango

Escribe un programa que solicite dos números y calcule la sumatoria de los números primos que existen entre esos dos valores. Utiliza un bucle for o while para recorrer los números en el rango y verifica si son primos.

```
12
Run | Debug
13 void main() {
14     print('Ingresa el primer número:');
15     int? inicio = int.parse(stdin.readLineSync());
16
17     print('Ingresa el segundo número:');
18     int? fin = int.parse(stdin.readLineSync());
19
20     if (inicio > fin) {
21         print('El primer número debe ser menor o igual que el
22             segundo número. ');
23         return;
24     }
25     int sumaPrimos = 0;
26
27     for (int i = inicio; i <= fin; i++) {
28         if (isPrime(i)) {
29             sumaPrimos += i;
30         }
31     }
32     print('La sumatoria de los números primos entre $inicio y $fin es: $sumaPrimos');
33 }
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS powershell + v

```
PS D:\emigmatic> dart ejercicio1.dart
Ingresa el primer número:
9
Ingresa el segundo número:
15
La sumatoria de los números primos entre 9 y 15 es: 24
PS D:\emigmatic>
```

Ejercicio 2: Números de Fibonacci hasta N términos

Implementa un programa que genere la secuencia de Fibonacci hasta un número n de términos ingresado por el usuario. Utiliza un bucle while o for para ir generando los números de la secuencia.

```
7  if (n <= 0) {
8      print('Por favor, ingresa un número mayor que 0.');
```

```
9      return;
10 }
11
12 List<int> fibonacci = [];
13
14 for (int i = 0; i < n; i++) {
15     if (i == 0) {
16         fibonacci.add(0);
17     } else if (i == 1) {
18         fibonacci.add(1);
19     } else {
20
21         int siguiente = fibonacci[i - 1] + fibonacci[i - 2];
22         fibonacci.add(siguiente);
23     }
24 }
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS powershell + ▢

```
PS D:\emigmatic> dart ejercicio2.dart
Ingresa el número de términos de la secuencia de Fibonacci que deseas generar:
8
La secuencia de Fibonacci con 8 términos es:
[0, 1, 1, 2, 3, 5, 8, 13]
PS D:\emigmatic>
```

Ejercicio 3: Factorial de números grandes

Escribe un programa que calcule el factorial de un número grande (por ejemplo, 100) utilizando estructuras repetitivas y el tipo de datos BigInteger para manejar grandes números.

```
ejercicio3.dart > factorial
1  import 'dart:io';
2
3  BigInt factorial(int n) {
4      BigInt resultado = BigInt.from(1);
5      for (int i = 2; i <= n; i++) {
6          resultado *= BigInt.from(i);
7      }
8
9      return resultado;
10 }
11
Run | Debug
12 void main() {
13     print('Ingresa un número para calcular su factorial:');
14     int? n = int.parse(stdin.readLineSync());
15
16     if (n < 0) {
17         print('El factorial no está definido para números
negativos.');
```

negativos.');

```
18     return;
19 }
20
21 BigInt resultado = factorial(n);
22 print(resultado);
23 }
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS powershell + v

```
PS D:\emigmatic> dart ejercicio3.dart
Ingresa un número para calcular su factorial:
20
El factorial de 20 es: 2432902008176640000
PS D:\emigmatic>
```

Ejercicio 4: Inversión de un número

Crea un programa que invierta los dígitos de un número entero ingresado por el usuario, utilizando un bucle while para extraer y reordenar los dígitos.

```
3 void main() {
8     print('Por favor, ingresa un número entero');
9     return;
10 }
11
12 bool esNegativo = input.startsWith('-');
13 if (esNegativo) {
14     input = input.substring(1);
15 }
16
17 String numeroInvertido = '';
18 int index = input.length - 1;
19
20 while (index >= 0) {
21     numeroInvertido += input[index];
22     index--;
23 }
24
25
26 if (esNegativo) {
27     numeroInvertido = '-' + numeroInvertido;
28 }
29
30 print('El número original: $input');
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

PS D:\emigmatic> dart ejercicio4.dart
Ingresa un número entero:
80
El número original: 80
El número invertido: 08
PS D:\emigmatic>

Ejercicio 5: Suma de matrices NxN

Escribe un programa que solicite dos matrices de tamaño Nx N (donde N es proporcionado por el usuario) y luego realice la suma de las dos matrices utilizando bucles anidados for.

```
9      List<List<int>> matrizSuma = List.generate(n, (i) => List
      filled(n, 0));
10
11      print('Ingresa los elementos de la matriz A:');
12      for (int i = 0; i < n; i++) {
13          for (int j = 0; j < n; j++) {
14              print('Elemento A[$i][$j]:');
15              matrizA[i][j] = int.parse(stdin.readLineSync());
16          }
17      }
18
19      print('Ingresa los elementos de la matriz B:');
20      for (int i = 0; i < n; i++) {
21          for (int j = 0; j < n; j++) {
22              print('Elemento B[$i][$j]:');
23              matrizB[i][j] = int.parse(stdin.readLineSync());
24          }
25      }
26
27      for (int i = 0; i < n; i++) {
28          for (int j = 0; j < n; j++) {
29              matrizSuma[i][j] = matrizA[i][j] + matrizB[i][j];
30          }
      }
  }
```


PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS powershell +

Elemento A[1][1]:
1
Ingresa los elementos de la matriz B:
Elemento B[0][0]:
6
Elemento B[0][1]:
9
Elemento B[1][0]:
4
Elemento B[1][1]:
5
La suma de las matrices A y B es:
11 15
8 6
PS D:\emigmatic>

Ejercicio 6: Número perfecto

Implementa un programa que encuentre y muestre todos los números perfectos entre 1 y 10,000. Un número perfecto es aquel que es igual a la suma de sus divisores propios. Usa un bucle para iterar y otro para encontrar los divisores de cada número.

```
Run | Debug
1 void main() {
2     print('Números perfectos entre 1 y 10,000:');
3
4     for (int numero = 1; numero <= 10000; numero++) {
5         int sumaDivisores = 0;
6
7         for (int i = 1; i < numero; i++) {
8             if (numero % i == 0) {
9                 sumaDivisores += i;
10            }
11        }
12
13        if (sumaDivisores == numero) {
14            print(numero);
15        }
16    }
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS  powershell

```
PS D:\emigmatic> dart ejercicio6.dart
Números perfectos entre 1 y 10,000:
6
28
496
8128
PS D:\emigmatic>
```

Ejercicio 7: Matriz de espiral

Crea un programa que imprima una matriz cuadrada de tamaño $n \times n$ en forma de espiral. Utiliza bucles anidados para recorrer las posiciones de la matriz en el orden adecuado.

```
ejercicio7.dart > main
3  void main() {
11  int filaFin = n - 1;
12  int colInicio = 0;
13  int colFin = n - 1;
14
15  while (filaInicio <= filaFin && colInicio <= colFin) {
16
17      for (int col = colInicio; col <= colFin; col++) {
18          matriz[filaInicio][col] = valor++;
19      }
20      filaInicio++;
21
22      for (int fila = filaInicio; fila <= filaFin; fila++) {
23          matriz[fila][colFin] = valor++;
24      }
25      colFin--;
26
27      if (filaInicio <= filaFin) {
28          for (int col = colFin; col >= colInicio; col--) {
29              matriz[filaFin][col] = valor++;
30          }
31          filaFin--;
32      }

```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS powershell + -

```
PS D:\emigmatic> dart ejercicio7.dart
Ingresa el tamaño de la matriz (n):
5
Matriz en forma de espiral:
1  2  3  4  5
16 17 18 19 6
15 24 25 20 7
14 23 22 21 8
13 12 11 10 9
PS D:\emigmatic>
```

Ejercicio 8: Verificación de un número Armstrong

Escribe un programa que verifique si un número de n dígitos ingresado por el usuario es un número de Armstrong (o narcisista). Utiliza un bucle for para separar y elevar cada dígito a la potencia correspondiente.

```
ejercicio8.dart > main
1  import 'dart:io';
2  import 'dart:math';
3
4  void main() {
5      print('Ingresa un número entero:');
6      int? numero = int.parse(stdin.readLineSync()!);
7
8      String numeroStr = numero.toString();
9      int cantidadDigitos = numeroStr.length;
10     int suma = 0;
11
12     for (int i = 0; i < cantidadDigitos; i++) {
13         int digito = int.parse(numeroStr[i]);
14         suma += pow(digito, cantidadDigitos).toInt();
15     }
16
17     if (suma == numero) {
18         print('$numero es un número de Armstrong.');
```

Run | Debug

```
19     } else {
20         print('$numero no es un número de Armstrong.');
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
21     }
22 }
```

powerhell

```
PS D:\emigmatic> dart ejercicio8.dart
Ingresa un número entero:
4
4 es un número de Armstrong.
PS D:\emigmatic> dart ejercicio8.dart
Ingresa un número entero:
15
15 no es un número de Armstrong.
PS D:\emigmatic> 
```

Ejercicio 9: Cálculo de potencias usando multiplicación repetida

Crea un programa que calcule la potencia de un número usando multiplicación repetida, es decir, sin utilizar la función `Math.pow()`. El programa debe solicitar una base y un exponente, y luego calcular la potencia utilizando un bucle `while` o `for`.


```
ejercicio9.dart 7 main
1  import 'dart:io';
2
   Run | Debug
3  void main() {
4      print('Ingresa la base:');
5      int base = int.parse(stdin.readLineSync());
6
7      print('Ingresa el exponente:');
8      int exponente = int.parse(stdin.readLineSync());
9
10     int resultado = 1;
11
12     for (int i = 0; i < exponente; i++) {
13         resultado *= base;
14     }
15
16     print('$base elevado a la $exponente es: $resultado');
17 }
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS powershell +

PS D:\emigmatic> dart ejercicio9.dart
Ingresa la base:
6
Ingresa el exponente:
9
6 elevado a la 9 es: 10077696
PS D:\emigmatic>