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.

```dart
/*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.*/
import 'dart:io';
bool esPrimo(int n) {
  if (n < 2) return false;
  for (int i = 2; i <= n ~/ 2; i++) {
    if (n % i == 0) return false;
  }
  return true;
}
Run | Debug
void main() {
  print('Ingrese el primer número:');
  int inicio = int.parse(stdin.readLineSync()!);
  print('Ingrese el segundo número:');
  int fin = int.parse(stdin.readLineSync()!);
  int sumaPrimos = 0;
  for (int i = inicio; i <= fin; i++) {
    if (esPrimo(i)) {
      sumaPrimos += i;
    }
  }
  print('La sumatoria de números primos entre $inicio y $fin es: $sumaPrimos');
}
```

Requerimientos Funcionales:

- Requerimiento 1: El programa debe solicitar al usuario dos números enteros que definan un rango.
- Requerimiento 2: El programa debe identificar todos los números primos dentro del rango (incluyendo los límites).
- Requerimiento 3: El programa debe calcular y mostrar la sumatoria de los números primos encontrados.

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.

```dart
/*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.*/
import 'dart:io';
Run | Debug
void main() {
  print('Ingrese la cantidad de términos de Fibonacci:');
  int n = int.parse(stdin.readLineSync()!);

  int a = 0, b = 1;
  print('Secuencia de Fibonacci:');

  for (int i = 0; i < n; i++) {
    print(a);
    int temp = a;
    a = b;
    b = temp + b;
  }
}
```

Requerimientos Funcionales:

- Requerimiento 1: El programa debe solicitar al usuario un número entero que representa la cantidad de términos de la secuencia de Fibonacci.
- Requerimiento 2: El programa debe generar e imprimir la secuencia de Fibonacci hasta el número de términos especificado.

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.

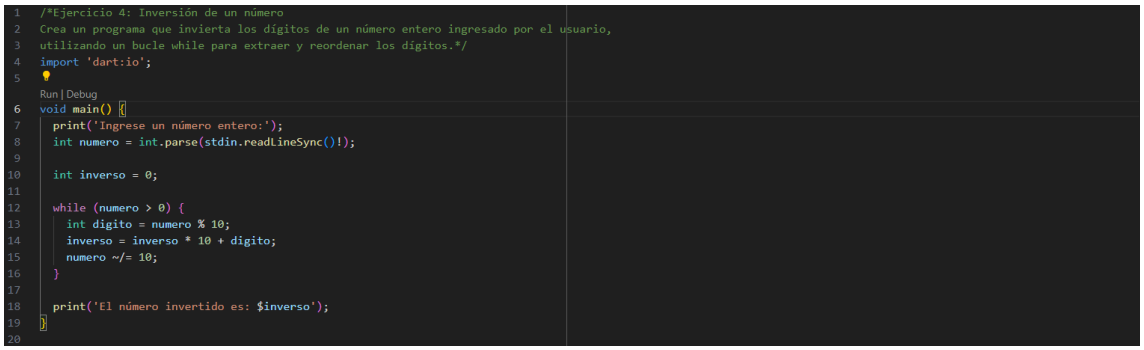
```dart
/*Ejercicio 3: Factorial de números grandes
Escribe un programa que calcule el factorial de un número grande (por ejemplo, 100)
utilizando estructuras repetitivas y el tipo de datos BigInteger para manejar grandes
números.*/
import 'dart:io';
import 'dart:math';

BigInt factorial(int n) {
  BigInt result = BigInt.from(1);
  for (int i = 2; i <= n; i++) {
    result *= BigInt.from(i);
  }
  return result;
}
Run | Debug
void main() {
  print('Ingrese un número entero para calcular su factorial:');
  int numero = int.parse(stdin.readLineSync()!);

  if (numero < 0) {
    print('El factorial no está definido para números negativos.');
  } else {
    BigInt resultado = factorial(numero);
    print('El factorial de $numero es: $resultado');
  }
}
```

Requerimientos Funcionales:

- Requerimiento 1: El programa debe calcular el factorial de un número grande (por ejemplo, 100).
- Requerimiento 2: El programa debe utilizar el tipo de datos BigInt para manejar grandes números.
- Requerimiento 3: El programa debe mostrar el resultado del factorial calculado.

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.



```dart
/*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.*/
import 'dart:io';

Run | Debug
void main() {
  print('Ingrese un número entero:');
  int numero = int.parse(stdin.readLineSync()!);

  int inverso = 0;

  while (numero > 0) {
    int digito = numero % 10;
    inverso = inverso * 10 + digito;
    numero ~/= 10;
  }

  print('El número invertido es: $inverso');
}
```

Requerimientos Funcionales:

- Requerimiento 1: El programa debe solicitar al usuario un número entero.
- Requerimiento 2: El programa debe invertir los dígitos del número ingresado.
- Requerimiento 3: El programa debe mostrar el número resultante después de la inversión.

Ejercicio 5: Suma de matrices NxN

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

```dart
/*Ejercicio 5: Suma de matrices NxN
Escribe un programa que solicite dos matrices de tamaño N x N (donde N es
proporcionado por el usuario) y luego realice la suma de las dos matrices utilizando
bucles anidados for.*/
import 'dart:io';

Run | Debug
void main() {
  print('Ingrese el tamaño de la matriz (N):');
  int n = int.parse(stdin.readLineSync()!);
  List<List<int>> matriz1 = List.generate(n, (_) => List.filled(n, 0));
  List<List<int>> matriz2 = List.generate(n, (_) => List.filled(n, 0));
  List<List<int>> suma = List.generate(n, (_) => List.filled(n, 0));
  print('Ingrese los elementos de la primera matriz:');
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      matriz1[i][j] = int.parse(stdin.readLineSync()!);
    }
  }

  print('Ingrese los elementos de la segunda matriz:');
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      matriz2[i][j] = int.parse(stdin.readLineSync()!);
    }
  }

  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      suma[i][j] = matriz1[i][j] + matriz2[i][j];
    }
  }

  print('La suma de las matrices es:');
  for (var fila in suma) {
    print(fila);
  }
}

```

Requerimientos Funcionales:

- Requerimiento 1: El programa debe solicitar al usuario un número entero que defina el tamaño de las matrices (N).
- Requerimiento 2: El programa debe permitir al usuario ingresar los elementos de dos matrices de tamaño N x N.
- Requerimiento 3: El programa debe calcular y mostrar la suma de las dos matrices.

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.

```dart
/*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.*/
bool esNumeroPerfecto(int n) {
  int sumaDivisores = 0;
  for (int i = 1; i < n; i++) {
    if (n % i == 0) {
      sumaDivisores += i;
    }
  }
  return sumaDivisores == n;
}

Run | Debug
void main() {
  print('Números perfectos entre 1 y 10,000:');
  for (int i = 1; i <= 10000; i++) {
    if (esNumeroPerfecto(i)) {
      print(i);
    }
  }
}
```

Requerimientos Funcionales:

- Requerimiento 1: El programa debe encontrar todos los números perfectos entre 1 y 10,000.
- Requerimiento 2: Un número perfecto es aquel que es igual a la suma de sus divisores propios.
- Requerimiento 3: El programa debe mostrar todos los números perfectos encontrados en el rango especificado.

Ejercicio 7: Matriz de espiral

Crea un programa que imprima una matriz cuadrada de tamaño n x n en forma de espiral.

Utiliza bucles anidados para recorrer las posiciones de la matriz en el orden adecuado.

```dart
/*Ejercicio 7: Matriz de espiral
Crea un programa que imprima una matriz cuadrada de tamaño n x n en forma de espiral.
Utiliza bucles anidados para recorrer las posiciones de la matriz en el orden adecuado.*/
import 'dart:io';

Run | Debug
void main() {
  print('Ingrese el tamaño de la matriz (N):');
  int n = int.parse(stdin.readLineSync()!);

  List<List<int>> matriz = List.generate(n, (_) => List.filled(n, 0));

  int valor = 1;
  int inicioFila = 0, finFila = n - 1;
  int inicioColumna = 0, finColumna = n - 1;

  while (inicioFila <= finFila && inicioColumna <= finColumna) {
    for (int i = inicioColumna; i <= finColumna; i++) {
      matriz[inicioFila][i] = valor++;
    }
    inicioFila++;

    for (int i = inicioFila; i <= finFila; i++) {
      matriz[i][finColumna] = valor++;
    }
    finColumna--;

    if (inicioFila <= finFila) {
      for (int i = finColumna; i >= inicioColumna; i--) {
        matriz[finFila][i] = valor++;
      }
      finFila--;
    }

    if (inicioColumna <= finColumna) {
      for (int i = finFila; i >= inicioFila; i--) {
        matriz[i][inicioColumna] = valor++;
      }
      inicioColumna++;
    }
  }

  print('Matriz en espiral:');
  for (var fila in matriz) {
    print(fila);
  }
}
```

Requerimientos Funcionales:

- Requerimiento 1: El programa debe solicitar al usuario un número entero que defina el tamaño de la matriz (N).
- Requerimiento 2: El programa debe crear una matriz cuadrada de tamaño N x N.
- Requerimiento 3: La matriz debe ser llenada en forma de espiral y luego impresa.

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.

```dart
/*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.*/
import 'dart:io';
import 'dart:math';

bool esNumeroArmstrong(int numero) {
  String strNumero = numero.toString();
  int n = strNumero.length;
  int suma = 0;
  for (int i = 0; i < n; i++) {
    suma += pow(int.parse(strNumero[i]), n).toInt();
  }
  return suma == numero;
}
Run | Debug
void main() {
  print('Ingrese un número:');
  int numero = int.parse(stdin.readLineSync()!);

  if (esNumeroArmstrong(numero)) {
    print('$numero es un número Armstrong.');
  } else {
    print('$numero no es un número Armstrong.');
  }
}
```

Requerimientos Funcionales:

- Requerimiento 1: El programa debe solicitar al usuario un número entero.
- Requerimiento 2: El programa debe verificar si el número ingresado es un número de Armstrong (o narcisista).
- Requerimiento 3: El programa debe mostrar un mensaje indicando si el número es o no un número de Armstrong.

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.

```dart
/*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.*/
import 'dart:io';

Run | Debug
void main() {
  print('Ingrese la base:');
  int base = int.parse(stdin.readLineSync()!);

  print('Ingrese el exponente:');
  int exponente = int.parse(stdin.readLineSync()!);

  int resultado = 1;

  for (int i = 0; i < exponente; i++) {
    resultado *= base;
  }

  print('$base elevado a la $exponente es: $resultado');
}
```

Requerimientos Funcionales:

- Requerimiento 1: El programa debe solicitar al usuario una base y un exponente (ambos enteros).
- Requerimiento 2: El programa debe calcular la potencia de la base elevada al exponente utilizando multiplicación repetida.
- Requerimiento 3: El programa debe mostrar el resultado del cálculo de la potencia