Resolución de Ejercicios en Dart

Ejercicio 1: Sumatoria de números primos en un rango

Enunciado

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.

Requerimientos

1. El programa debe solicitar dos números enteros positivos al usuario.
2. Calcular la suma de todos los números primos dentro del rango definido.
3. Utilizar un bucle para iterar dentro del rango y verificar si cada número es primo.
4. Mostrar la suma de los números primos al usuario.

Código en Dart

```
void main() {
  print("Ingrese el primer número:");
  int num1 = int.parse(stdin.readLineSync()!);

  print("Ingrese el segundo número:");
  int num2 = int.parse(stdin.readLineSync()!);

  int sumaPrimos = 0;

  for (int i = num1; i <= num2; i++) {
    if (esPrimo(i)) {
      sumaPrimos += i;
    }
  }

  print("La suma de los números primos entre $num1 y $num2 es: $sumaPrimos");
}

bool esPrimo(int num) {
  if (num <= 1) return false;
  for (int i = 2; i <= num ~/ 2; i++) {
    if (num % i == 0) return false;
  }
  return true;
}
```

Ejercicio 2: Números de Fibonacci hasta N términos

Enunciado

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.

Requerimientos

1. Solicitar al usuario un número entero positivo n para la cantidad de términos.
2. Generar la secuencia de Fibonacci hasta n términos.
3. Utilizar un bucle while o for para ir generando cada número de la secuencia.
4. Mostrar la secuencia de Fibonacci completa al usuario.

Código en Dart

```
void main() {
  print("Ingrese el número de términos de la secuencia de Fibonacci:");
  int n = int.parse(stdin.readLineSync()!);

  int a = 0, b = 1, temp;

  print("Secuencia de Fibonacci hasta $n términos:");
  for (int i = 1; i <= n; i++) {
    print(a);
    temp = a + b;
    a = b;
    b = temp;
  }
}
```

Ejercicio 3: Factorial de números grandes

Enunciado

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

Requerimientos

1. Solicitar un número entero positivo al usuario.
2. Calcular el factorial del número utilizando el tipo de datos BigInt para grandes valores.
3. Utilizar un bucle while o for para calcular el factorial.
4. Mostrar el resultado del factorial al usuario.

Código en Dart

```
import 'dart:math';

void main() {
  print("Ingrese un número para calcular su factorial:");
  int num = int.parse(stdin.readLineSync()!);

  BigInt factorial = BigInt.from(1);
  for (int i = 1; i <= num; i++) {
    factorial *= BigInt.from(i);
  }

  print("El factorial de $num es: $factorial");
}
```

Ejercicio 4: Inversión de un número

Enunciado

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.

Requerimientos

1. Solicitar un número entero positivo al usuario.
2. Extraer los dígitos y reordenarlos utilizando un bucle.
3. Mostrar el número invertido al usuario.

Código en Dart

```
void main() {
  print("Ingrese un número para invertir sus dígitos:");
  int num = int.parse(stdin.readLineSync()!);

  int invertido = 0;
  while (num != 0) {
    int digito = num % 10;
    invertido = invertido * 10 + digito;
    num ~/= 10;
  }

  print("El número invertido es: $invertido");
```

Ejercicio 5: Suma de matrices NxN

Enunciado:

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.

Requerimientos:

1. Solicitar el tamaño N de las matrices al usuario.
2. Solicitar los elementos de ambas matrices.
3. Realizar la suma de las matrices utilizando bucles anidados.
4. Mostrar la matriz resultante al usuario.

Código en Dart:

```
void main() {

  print("Ingrese el tamaño de las matrices (N):");
```

```
int n = int.parse(stdin.readLineSync()!);
List<List<int>> matriz1 = List.generate(n, (i) => List.filled(n, 0));
List<List<int>> matriz2 = List.generate(n, (i) => List.filled(n, 0));
List<List<int>> sumaMatriz = List.generate(n, (i) => 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++) {
    print("Elemento [$i][$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++) {
    print("Elemento [$i][$j]:");
    matriz2[i][j] = int.parse(stdin.readLineSync()!);
  }
}

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

print("La matriz resultante de la suma es:");
for (int i = 0; i < n; i++) {
```

```
  print(sumaMatriz[i]);
 }
}
```

Ejercicio 6: Número perfecto

Enunciado:
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.

Requerimientos:

1. Iterar sobre los números desde 1 hasta 10,000.
2. Para cada número, encontrar sus divisores propios (excluyendo el número en sí).
3. Sumar los divisores y verificar si es igual al número original.
4. Mostrar los números perfectos encontrados.

Código en Dart:

```
void main() {
 print("Números perfectos entre 1 y 10,000:");
 for (int num = 1; num <= 10000; num++) {
  int sumaDivisores = 0;
  for (int i = 1; i <= num ~/ 2; i++) {
   if (num % i == 0) {
    sumaDivisores += i;
   }
  }

  if (sumaDivisores == num) {
   print(num);
  }
```

```
 }
}
```

Ejercicio 7: Matriz de espiral

Enunciado:
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.

Requerimientos:

1. Solicitar al usuario el tamaño n de la matriz.
2. Utilizar bucles para recorrer la matriz en orden de espiral.
3. Mostrar la matriz en espiral al usuario.

Código en Dart:

```
void main() {
  print("Ingrese el tamaño de la matriz en espiral:");
  int n = int.parse(stdin.readLineSync()!);

  List<List<int>> matriz = List.generate(n, (i) => List.filled(n, 0));
  int valor = 1, inicioFila = 0, finFila = n - 1, inicioCol = 0, finCol = n - 1;

  while (inicioFila <= finFila && inicioCol <= finCol) {
    for (int i = inicioCol; i <= finCol; i++) matriz[inicioFila][i] = valor++;
    inicioFila++;
    for (int i = inicioFila; i <= finFila; i++) matriz[i][finCol] = valor++;
    finCol--;
    if (inicioFila <= finFila) {
      for (int i = finCol; i >= inicioCol; i--) matriz[finFila][i] = valor++;
      finFila--;
```

```
    }
    if (inicioCol <= finCol) {
      for (int i = finFila; i >= inicioFila; i--) matriz[i][inicioCol] = valor++;
      inicioCol++;
    }
  }
  print("Matriz en espiral:");
  for (var fila in matriz) {
    print(fila);
  }
}
```

Ejercicio 8: Verificación de un número Armstrong

Enunciado:
Escribe un programa que verifique si un número es un número Armstrong (o narcisista). Utiliza un bucle para separar y elevar cada dígito a la potencia correspondiente.

Requerimientos:

1. Solicitar un número entero positivo al usuario.
2. Separar cada dígito y elevarlo a la potencia correspondiente.
3. Sumar los dígitos elevados y comparar con el número original.
4. Mostrar si el número es Armstrong o no.

Código en Dart:

```
void main() {
```

```
  print("Ingrese un número para verificar si es Armstrong:");
  int num = int.parse(stdin.readLineSync()!);
  int originalNum = num, suma = 0, n = num.toString().length;

  while (num > 0) {
    int digito = num % 10;
    suma += pow(digito, n).toInt();
    num ~/= 10;
  }
  if (suma == originalNum) {
    print("$originalNum es un número Armstrong.");
  } else {
    print("$originalNum no es un número Armstrong.");
  }
}
```

Ejercicio 9: Cálculo de potencias usando multiplicación repetida

Enunciado:
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.

Requerimientos:

1. Solicitar la base y el exponente al usuario.
2. Calcular la potencia utilizando multiplicación repetida.
3. Mostrar el resultado al usuario.

Código en Dart:

```
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 = 1; i <= exponente; i++) {
    resultado *= base;
  }

  print("El resultado de $base^$exponente es: $resultado");
}
```