Índice

Ejercicios Unidad 8

Descargar estos ejercicios

Antes de empezar

Para realizar estos ejercicios, deberás descargar los recursos del enlace de proyecto_arrays. Como puedes ver, la solución está compuesta de varios proyectos. Cada uno de ellos corresponde con un ejercicio, deberás implementar todo el código, tanto de la Main como de los métodos que se piden en cada ejercicio. Cada proyecto contiene el test correspondiente, que deberás pasar para comprobar que has hecho el ejercicio correctamente.

Ejercicio 1. Tabla dentada cuadrada con patrón alternado

Crea un programa en el proyecto ejercicio1 que cree una array de arrays **con 10 tablas de 10 elementos cada una**. Rellena el array usando bucles, de forma que las **filas pares se rellenen con unos y las impares con ceros**.

```
Ejercicio 1: Tabla dentada cuadrada con patrón

Array 10x10 generado:
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0

Presiona cualquier tecla para salir...
```

Requisitos:

- Método **CreaArray** al que le llegan dos enteros filas y columnas con el tamaño. El método creará y devolverá la tabla dentada, asignando la memoria a cada una de las tablas.
- Método **RellenaConPatron** que rellene el array según el patrón especificado.
- Método **MuestraArray** al que le llega la tabla dentada y visualice el contenido del array en pantalla. Usa bucles `for` anidados para el recorrido.

Ejercicio 2. Diagonal en tabla de tablas

Crea un programa en el proyecto ejercicio2 que cree una tabla dentada de 5 tablas con 5 columnas cada una. Inicializa el array, usando bucles, de forma que los componentes pertenecientes a la **diagonal** del array tomen valor **uno**, y el resto valor cero.

```
Ejercicio 2: Diagonal en tabla de tablas

Array identidad 5x5:
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1

Presiona cualquier tecla para salir...
```

Requisitos:

- Método **CreaArrayIdentidad** que cree la tabla dentada del tamaño que le llega como parámetro. Es decir, que inicialice todas las tablas que la componen, y la devuelva.
- Método **InicializaDiagonal** al que le llega la tabla anterior y coloca 1 en la diagonal principal y 0 en el resto.
  - La diagonal principal tiene índices `[i][i]` donde i va de 0 a 4.
- Método **MuestraArrayConForeach** que use `foreach` para mostrar el contenido.
  - Controlar el salto de línea en cada cambio de tabla.

Ejercicio 3. Transposición de arrays

Programa en el proyecto ejercicio3 una tablas dentada que contenga tres tablas con 5 columnas cada una. Posteriormente se deberá crear otra tabla dentada **permutando filas por columnas** (transposición).

```
Ejercicio 3: Transposición de arrays

Introduce los elementos del array 3x5:
Fila 0, Columna 0: -1
Fila 0, Columna 1: 2
Fila 0, Columna 2: 1
Fila 0, Columna 3: 4
Fila 0, Columna 4: 7
Fila 1, Columna 0: -3
Fila 1, Columna 1: 3
Fila 1, Columna 2: 5
Fila 1, Columna 3: 8
Fila 1, Columna 4: 9
Fila 2, Columna 0: 6
Fila 2, Columna 1: 0
Fila 2, Columna 2: -2
Fila 2, Columna 3: 1
Fila 2, Columna 4: 3

Array original (3x5):
-1  2  1  4  7
-3  3  5  8  9
 6  0 -2  1  3

Array transpuesto (5x3):
-1 -3  6
 2  3  0
 1  5 -2
 4  8  1
 7  9  3

Presiona cualquier tecla para salir...
```

Requisitos:

- Método `int[][] LeeArray(int filas, int columnas)` que lea una tabla dentada de 3x5 desde el usuario.
- Método `int[][] CreaTranspuesta(int[][] original)` que genere el array transpuesto de 5x3 y lo devuelva.
  - El array transpuesto convierte filas en columnas y viceversa.
  - El elemento `[i][j]` del original va a la posición `[j][i]` del transpuesto.
- Método `void MuestraArray(int[][] array, string titulo)` al que le llega el array y el texto del título y muestra un array formateado.

> **Pista**

> Aunque en el ejemplo y durante el enunciado se habla de 3 filas y 5 columnas, el código debe ser genérico y funcionar para cualquier tamaño de array dentado inclusive cuadrados. Solo se indicará en el Main los tamaños 3 y 5 al llamar a `LeeArray` y el resto del código debe funcionar para cualquier tamaño.

Ejercicio 4. Jardín de flores con inventario colorido

Crea el código necesario en el proyecto ejercicio4 que gestione un jardín representado por una tabla dentada donde cada fila es un arriate con flores de distintos colores. El programa mostrará las flores con colores en consola y generará un inventario completo.

```
Ejercicio 4: Jardín de flores con inventario colorido

1 3 2 1
4 4 2
2 1 3 3 5
3 2

Color 1: 3 flores
Color 2: 4 flores
Color 3: 4 flores
Color 4: 2 flores
Color 5: 1 flores

Arriate más diverso: Arriate 3 con 4 colores distintos.

Presiona cualquier tecla para salir...
```

Requisitos:

- Crear la tabla dentada del jardín usando la nueva sintaxis de arrays:

```
int[][] jardin = [
    [1, 3, 2, 1],
    [4, 4, 2],
    [2, 1, 3, 3, 5],
    [3, 2]
];
```

- Método **Muestra** que imprima el jardín con colores usando `Console.ForegroundColor`:
  - Usar `foreach` anidados para recorrer la tabla dentada.
  - Aplicar color a cada número usando `(ConsoleColor)(colorFlor % 16)`.

    - Resetear el color con `Console.ResetColor()` después de cada número.
- Método **ColorMasAlto** que encuentra y devuelve el valor de color más alto en todo el jardín:
    - Usar `foreach` anidados para recorrer toda la estructura.
    - Comparar cada color encontrado con el máximo actual.
- Método **CuentaFloresPorColor** que devuelva un array con el inventario de flores:
    - Usar el método auxiliar **ColorMasAlto** para determinar el tamaño del array inventario.
    - Recorrer toda la tabla dentada incrementando el contador correspondiente.
- Método **MuestraInventarioColoresFlores** que muestre el inventario con colores:
    - Recorrer el array de inventario con bucle `for`.
    - Solo mostrar colores que tengan cantidad > 0.
    - Aplicar el color correspondiente a cada línea de salida.
- Método **ArriateMasDiverso** que encuentre el arriate con más colores distintos. Usa una tupla para devolver la información necesaria, numero de arriate y cantidad de flores distintas de ese arriate.
- Método **CuentaColores** que cuente colores únicos en un arriate:
    - Crear un array vacío `int[] coloresFlores = []`.
    - Usar `Array.IndexOf` para verificar si el color ya existe.
    - Si no existe el color añádelo al array.

Ejercicio 5. Panadería: pedidos semanales por tipo de harina

En el proyecto se debe crear un programa que registre los pedidos semanales de harina realizados por una panadería.

Tendrá un array con los tipos de harina disponibles y una **tabla dentada** que guardará, para cada día de la semana (lunes a domingo), los pedidos realizados ese día en cantidad por tipo de harina (tupla de string y int). Como cada día no tiene por qué pedirse todas las harinas, la estructura será irregular (jagged): cada fila representa un día y contiene las cantidades pedidas de tipos de harinas que se solicitaron ese día.

Ejemplo de tipos de harina:

```
string[] harinas = new string[] { "Trigo", "Centeno", "Espelta", "Maíz" };
```

```
Ejercicio 5. Panadería: pedidos semanales por tipo de harina

-- Pedidos del Lunes --
Introduce la cantidad pedida en KG, de:
Trigo: 0
Centeno: 10
Espelta: 2
Maíz: 5
-- Pedidos del Martes --
Introduce la cantidad pedida en KG, de:
Trigo: 4
Centeno: 10
Espelta: 0
Maíz: 0
...
Resumen pedidos semana (Lun-Dom):
Lunes:     [Centeno: 10][Espelta: 2][Maíz: 5]
Martes:    [Trigo: 4][Centeno: 10]
Miércoles: [Centeno: 1]
Jueves:    [ ]
Viernes:   [Trigo: 5][Centeno: 12][Espelta: 5][Maíz: 7]
Sábado:    [Trigo: 4]
Domingo:   [Espelta: 1]

Total por harina:
Trigo: 13
Centeno: 33
Espelta: 8
Maíz: 12

Harina más pedida: Centeno (33 unidades)
```

Requisitos:

- Método `CreaPedidosDiarios(string[] harinas, string dia)` que genere y devuelve un `(string, int)[]`. El programa mostrará el tipo de harina y le pedirá al panadero que introduzca la cantidad comprada para ese día, si la cantidad es 0 no se añadirá ese tipo de harina al array
- Método `CreaPedidosSemana(string[] harinas)` que genere y devuelva un `(string, int)[][]` con 7 filas de la semana, usando el método anterior para generarlas.
- Método `MuestraPedidosSemana` al que le llega la tabla dentada y muestra los pedidos por día como en el ejemplo.
- Método `MuestraTotalesPedidos` al que le llegan las dos tablas y calcula y muestra los totales como se ve en la salida, utiliza un array de `int[]` para ir acumulando los totales de cada harina, que coincidirán en posición con el array harinas.