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** que cree y devuelva 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 indicado.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 dentada de tres tablas con 5 columnas cada tabla. 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 **LeeArray** que lea una tabla dentada de 3x5 desde el usuario.
- Método **CreaTranspuesta** 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 **MuestraArray** que muestre un array formateado.

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. Sistema completo de gestión de multicine

Programa en el proyecto ejercicio5 que gestione un multicine completo combinando el control de aforo (array bidimensional) con la información de películas (tabla dentada).

```
Ejercicio 5. Sistema completo de gestión de multicine

CARTELERA ACTUAL:
Sala A: ["Avengers", "Matrix", "Inception"]
Sala B: ["Titanic", "Avatar"]
Sala C: ["Interstellar", "Dune", "Blade Runner", "Star Wars"]


=== MENÚ PRINCIPAL ===
1. Venta de entradas
2. Estadística de aforo
3. Mostrar cartelera completa
4. Película más popular
ESC. Salir

Opción: 1


Número de entradas: 4
Indique Sala (A,B,C): B
Indique Sesion (1,2,3): 2
Película (0-Titanic, 1-Avatar): 1

Entradas vendidas para "Avatar" - Sesión 2

=== MENÚ PRINCIPAL ===
1. Venta de entradas
2. Estadística de aforo
3. Mostrar cartelera completa
4. Película más popular
ESC. Salir

Opción: 1


Número de entradas: 4
Indique Sala (A,B,C): C
Indique Sesion (1,2,3): 1
Película (0-Interstellar, 1-Dune, 2-Blade Runner, 3-Star Wars): 9
ERROR: Película inexistente
Película (0-Interstellar, 1-Dune, 2-Blade Runner, 3-Star Wars): 1

Entradas vendidas para "Dune" - Sesión 1

=== MENÚ PRINCIPAL ===
1. Venta de entradas
2. Estadística de aforo
3. Mostrar cartelera completa
4. Película más popular
ESC. Salir
```

```
Opción: 2

=== ESTADÍSTICA DE AFORO ===
                Sesión1    Sesión2    Sesión3    Total
Sala A (3 films)        0        0        0       0
Sala B (2 films)        0        4        0       4
Sala C (4 films)        4        0        0       4

=== DETALLE POR PELÍCULA ===
Sala A - Avengers: 0 entradas
Sala A - Matrix: 0 entradas
Sala A - Inception: 0 entradas
Sala B - Titanic: 0 entradas
Sala B - Avatar: 4 entradas
Sala C - Interstellar: 0 entradas
Sala C - Dune: 4 entradas
Sala C - Blade Runner: 0 entradas
Sala C - Star Wars: 0 entradas

=== MENÚ PRINCIPAL ===
1. Venta de entradas
2. Estadística de aforo
3. Mostrar cartelera completa
4. Película más popular
ESC. Salir

Opción: 1


Número de entradas: 2
Indique Sala (A,B,C): B
Indique Sesion (1,2,3): 2
Película (0-Titanic, 1-Avatar): 1

Entradas vendidas para "Avatar" - Sesión 2

=== MENÚ PRINCIPAL ===
1. Venta de entradas
2. Estadística de aforo
3. Mostrar cartelera completa
4. Película más popular
ESC. Salir

Opción: 2

=== ESTADÍSTICA DE AFORO ===
                Sesión1    Sesión2    Sesión3    Total
Sala A (3 films)        0        0        0       0
Sala B (2 films)        0        6        0       6
Sala C (4 films)        4        0        0       4

=== DETALLE POR PELÍCULA ===
```

```
Sala A - Avengers: 0 entradas
Sala A - Matrix: 0 entradas
Sala A - Inception: 0 entradas
Sala B - Titanic: 0 entradas
Sala B - Avatar: 6 entradas
Sala C - Interstellar: 0 entradas
Sala C - Dune: 4 entradas
Sala C - Blade Runner: 0 entradas
Sala C - Star Wars: 0 entradas

=== MENÚ PRINCIPAL ===
1. Venta de entradas
2. Estadística de aforo
3. Mostrar cartelera completa
4. Película más popular
ESC. Salir

Opción: 3
CARTELERA ACTUAL:
Sala A: ["Avengers", "Matrix", "Inception"]
Sala B: ["Titanic", "Avatar"]
Sala C: ["Interstellar", "Dune", "Blade Runner", "Star Wars"]


=== MENÚ PRINCIPAL ===
1. Venta de entradas
2. Estadística de aforo
3. Mostrar cartelera completa
4. Película más popular
ESC. Salir

Opción: 4

=== PELÍCULA MÁS POPULAR ===
Película: "Avatar" (Sala B)
Total entradas vendidas: 6
Sesiones: 3

=== MENÚ PRINCIPAL ===
1. Venta de entradas
2. Estadística de aforo
3. Mostrar cartelera completa
4. Película más popular
ESC. Salir

Opción:

Presiona cualquier tecla para salir...
```

Requisitos:

- Array bidimensional `int[3,3]` para controlar entradas vendidas por sala y sesión.
- Tabla dentada `string[][]` para almacenar las películas de cada sala:

```
static string[][] cartelera = new string[][]
{
    new string[] {"Avengers", "Matrix", "Inception"},           // Sala A
    new string[] {"Titanic", "Avatar"},                         // Sala B
    new string[] {"Interstellar", "Dune", "Blade Runner", "Star Wars"} // Sala C
};
```

- Método **MuestraCartelera** que muestre todas las películas por sala, formateadas como se muestra en la salida.
- Método **PideSala** que pide al usuario que introduzca una sala y devuelve información como entero, encargándose el método de gestionar el cambio de carácter a entero. (corresponderá con el inidce del array), indicando los errores si no se ponen los datos corrector.
- Método **PideSesion** que devolverá el entero coorespondiente a la entrada del usuario, pero controlando que la entrada es correcta.
- Método **PidePelicula** que reciba como parámetro el número de sala y pida al usuario seleccionar una película:
  - Mostrar las opciones disponibles en formato: "0-Avengers, 1-Matrix, 2-Inception".
  - Usar `do-while` para repetir la entrada hasta que sea válida.
  - Validar la selección comprobando que la película está en la sala.
  - Mostrar mensaje de error si la película no existe.
  - Devolver el índice de la película seleccionada.
- Método **VenderEntradas** que gestione venta incluyendo selección de película. Le llega como entrada el array de aforo. Pasará a pedir los datos mediante los métodos anteriores, comprobando que la venta es posible.
- Método **EstadisticasCompletas** que combine datos de aforo y películas mostrando la tabla formateada como en la salida:
  - Mostrar encabezados de sesiones alineados (Sesión1, Sesión2, Sesión3, Total).
  - Para cada sala mostrar el nombre, número de películas y entradas por sesión.
  - Calcular y mostrar el total de entradas por sala sumando las tres sesiones.
  - Usar bucles `for` para recorrer el array bidimensional de aforo.
  - Acceder a la tabla dentada de cartelera para obtener el número de películas por sala.
  - Formatear la salida con espaciado adecuado para alineación de columnas.
- Método **PeliculaMasPopular** que encuentre la película con más entradas vendidas y muestre la información calculada como se ve en la salida.
- Main con el menú correspondiente y las llamadas a los métodos.