Ejercicios Matrices

Descargar estos ejercicios

Índice

Ejercicio 1

Crea un programa que cree un array bi-dimensional de **10 x 10**. Rellena la matriz usando bucles, de forma que las **filas pares se rellenen con unos y las impares con ceros**.
Posteriormente muestra su contenido en pantalla.

Ejercicio 2

Programa que cree un array bi-dimensional de **5 x 5**. Inicializa la matriz, usando bucles, de forma que los componentes pertenecientes a la **diagonal** de la matriz tomen valor **uno**, y el resto valor cero. Una vez inicializada la matriz muestra su contenido en pantalla con un **foreach**.

```
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
```

💡 **Pista:** para visualizar la matriz con foreach, debes de controlar de algún modo la cantidad de elementos impresos, cuando sean igual a los de una fila insertas el salto de línea. Ya que una matríz

es una sola tabla y al usar el foreach se vuelcan todos los elementos.

Ejercicio 3

Dada una matriz de 3x5, diseña un programa que lea dicha matriz y posteriormente cree una nueva matriz a partir de la primera **permutando filas por columnas**. A esta nueva matriz la llamaremos matriz traspuesta.

$$A_{2\times 3} = \begin{bmatrix} -1 & 2 & 1 \\ -3 & 3 & 5 \end{bmatrix} => A^T_{2\times 3} = \begin{bmatrix} -1 & 3 \\ 2 & 3 \\ 1 & 5 \end{bmatrix}$$

☑ Ejercicio 4

Tenemos una **tabla dentada de caracteres** `char[][] paises` , que tiene almacenados los nombres de **20 países**.
Se pide diseñar un programa que realice, tantas veces como sea requerido por el usuario, las siguientes operaciones:

1. Buscar un país.
2. Mostrar países.
3. Ordenar países.
4. Añadir prefijo a un país.

Otras consideraciones...

- El prefijo estará formado por 2 caracteres, habrá un espacio en blanco entre el nombre y el prefijo. Para añadir el prefijo a la fíla correspondiente del país, puedes redimensionar la fila de la dentada usando el método:

```
Array.Resize(ref paises[paisEncontrado], paises.Length+3)
```

- Para ordenar la dentada alfabéticamente, utilizaremos el método de ordenación de la burbuja (hay un ejemplo en los apuntes). Para comparar cadenas alfabéticamente debes usar el método CompareTo de cadena.

💡 **Pista:** para programar este ejercicio podemos pasar de array de char a string o viceversa, cuando lo necesitemos. Recordar que para pasar de string a array de char usaremos `cadena.TocharArray()` y de array de char a string con `new String(array)`

Ejercicio 5

Dada la siguiente tabla dentada:

```
int[][] m = new int[][]
{
        new int [] {1,2,3,4},
        new int [] {5,6,7},
        new int [] {9,10,11,12,5},
        new int [] {9,10}
};
```

Haz un programa que busque en la tabla y posteriormente muestre, el **array con mas número de elementos**, usando solo bucles foreach.

📌 **Nota:** No hace falta que crees ningún método auxiliar.

Ejercicio 6

Crea un programa para **mostrar** los elementos de la siguiente tabla dentada:

```
int[][][] tabla = new int[][][]
{
    new int [][]
    {
        new int [] {1,2,3,4},
        new int [] {5,6,7},
        new int [] {9,10,11,12}
    },
    new int [][]
    {
        new int [] {13,14,15,16},
        new int [] {17,18,19,20},
        new int [] {21,22}
    },
};
```

Cada tabla se debe mostrar de la siguiente manera.
Utiliza las funciones necesarias para que el código quede claro.

```
1,2,3,4                 13,14,15,16
5,6,7                   17,18,19,20
9,10,11,12              21,22
```

✅ Ejercicio 7

Escribe un programa que se encargue de controlar el **aforo de un Multicine**:

- El **cine** tendrá **tres salas** (A, B, C), en las cuales se pasarán diariamente tres sesiones (1ª, 2ª, 3ª).
- El número **máximo de personas** de cada una de las salas es:
  - Sala **A** = **200** personas.
  - Sala **B** = **150** personas.
  - Sala **C** = **125** personas.
- Tendremos un menú con dos opciones:
  1. Venta de entradas.
  2. Estadística de aforo.
- Para salir del programa se tendrá que pulsar la tecla **ESC**.

Cada vez que se realice una venta de entradas se pedirá:

- El número de entradas que se van a comprar.
- La sala
- La sesión a la que se quiere asistir.

Las entradas vendidas quedarán registradas en la matriz bi-dimensional.
Si el número de entradas sobrepasa el aforo máximo de la sala, se indicará mediante un mensaje por pantalla.

En la opción de estadística de aforo, se mostrará una tabla de la siguiente manera:

```
            Sesión1         Sesión2         Sesión3
SalaA       178             100             99
SalaB       12              50              100
SalaC       32              101             55
```

💡 **Pista:** puedes plantearte la solución de dos formas distinta. Inicializando todos los elementos de la matriz de 3X3 a 0 y en cada venta añadir, al elemento correspondiente a los indices, las entradas vendidas sin pasarse del afora de la sala. La otra opción sería inicializar la matriz una fila a 200, la otra a 150 y la última a 125, y en cada venta decrementar la cantidad vendida, controlando de no vender si se ha llegado a 0.

✅ Ejercicio 8

Realiza un programa para **crear un array de array de arrays de enteros**.

- Para ello deberás crear una array de dos elementos, donde cada elemento será una tabla dentada de enteros.
- El programa pedirá cuantas filas tiene y el número de elementos de cada fila, para cada una de las tablas dentadas.
- Reservará la memoria para cada una y rellenará la fila con los elementos generados de forma aleatoria.
  Posteriormente se mostrará el array de dos matrices dentadas de la siguiente forma:

```
1,2,3,4                 13,14,15,16
5,6,7                   17,18,19,20
9,10,11,12              21,22
```

💡 **Pista:** La tabla se creará de la siguiente manera int[][][] m = new int[2][][]. Recuerda que las tablas se dimensionan con la palabra reservada new, y deberás redimensionar tanto, el número de filas de cada tabla como el número de columnas que tiene cada fila.

Ejercicio 9

Dado el siguiente código:

```
static void Main()
{
    int[,] m1 = new int[,] { { 1, 3, 5 }, { 2, 4, 6 } };
    int[,] m2 = new int[,] { { 6, 5 }, { 4, 3 }, { 2, 1 } };

    int[,] m3 = MultiplicaMatricesEnteras(m1, m2);
}
```

Implementa el código del método estático:

```
static int[,] MultiplicaMatricesEnteras(int[,] m1, int[,] m2);
```

Para multiplicar dos matrices debes tener en cuenta lo siguiente:

$$\left[\ A\ \right]_{m,n} \cdot \left[\ B\ \right]_{n,p} = \left[\ C\ \right]_{m,p}$$

*Dada una matriz **A de m filas y n columnas** y una matriz **B de n filas y p columnas**, el número de columnas de la primera debe coincidir con el número de filas de la segunda*.

Nota: Si las matrices de entrada no cumplen la condición para poder ser multiplicadas, se mostrará un mensaje de error informativo.

El resultado será una matriz C de m filas y p columnas (las filas de A y las columnas de B) de tal manera que Cij (fila i-ésima y columna j-ésima de C) se obtiene de sumar los productos de los elementos correspondientes a la fila i-ésima de la primera matriz (A), por los de la columna j-ésima de la segunda matriz (B).

Ejemplo resultado de multiplicar las matrices m1 y m2 de código del main:

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} \cdot \begin{bmatrix} 6 & 5 \\ 4 & 3 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1*6+3*4+5*2 & 1*5+3*3+5*1 \\ 2*6+4*4+6*2 & 2*5+4*3+6*1 \end{bmatrix} = \begin{bmatrix} 28 & 19 \\ 40 & 28 \end{bmatrix}$$

📌 **Nota:** Para solucionar el ejercicio debes tener en cuenta que el método `m1.GetLength(dimension)` devuelve el número de elementos de `m1` en la dimensión especificada, de tal manera que: `m1.Getlength(0)` me devuelve el número de filas de `m1` y `m1.GetLength(1)` me devuelve el número de columnas.