# Clase: Introducción a Matrices en C#

**¿Qué es una matriz?**

Una matriz es una estructura de datos que permite almacenar múltiples valores del mismo tipo en una sola variable. En C#, existen matrices unidimensionales y matrices multidimensionales.

**1. Matrices unidimensionales (vectores / arrays)**

Las matrices unidimensionales conocidas también como vectores o arreglos (arrays en inglés) son matrices que tienen n cantidad de columnas, pero solamente una fila.

📌 **Ejemplo:**

Declaración de un vector/array:  
**int**[] edades = **new** **int**[5] { 18, 25, 30, 22, 28 };

Como recorrer un array unidimensional:

**for** (**int** i = 0; i < edades.Length; i++)  
{  
    Console.**WriteLine**("Edad: " + edades[i]);  
}

También se puede usar **foreach**:

**foreach** (**int** edad **in** edades)  
{  
      Console.**WriteLine**("Edad: " + edad);  
}

**Importante:** El método Length es una funcion que permite obtener el tamaño del array

**2. Matrices multidimensionales**

Se parece a una tabla con filas y columnas. Estas matrices a diferencia de las unidimensionales pueden tener n dimensiones. Las matrices que comúnmente conocemos son de 2 dimensiones (largo, alto) pero puede haber matrices de más dimensiones, normalmente las de 2 dimensiones son las más utilizadas. Para declarar una matriz de 2 dimensiones se utilizan 2 índices los cuáles indican la cantidad de columnas y filas.

📌 **Ejemplo:**

Declaración de un vector/array de 2 dimensiones:

**int**[,] calificaciones = **new** **int**[2, 3]{   { 7, 8, 9 },   { 6, 5, 10 } };

Para recorrer una matriz bidimensional podemos aplicar la siguiente lógica:

**for** (**int** fila = 0; fila < calificaciones.GetLength(0); fila++)  
{  
    **for** (**int** col = 0; col < calificaciones.GetLength(1); col++)  
    {  
        Console.**WriteLine**("fila: "+fila + " col: " +col +" valor: "+calificaciones[fila, col]);

}  
}

**Importante:** El método GetLength(idx) es una funcion que permite obtener el tamaño de una de las dimensiones del array. El idx es un número entero que indica qué dimensión se desea consultar:

**0 → cantidad de filas**

**1 → cantidad de columnas**

**3. Diferencias clave:**

| **Característica** | Array Unidimensional | Array Multidimensional |
| --- | --- | --- |
| **Forma** | Lista lineal | Tabla (filas y columnas) |
| **Declaración** | int[] | int[,] |
| **Acceso a datos** | edades[i] | matriz[fila, columna] |
| **Uso común** | Listas de valores simples | Tablas, mapas, tableros de juego |

**4. Como almacenar objetos en vectores/matrices**

En lugar de guardar valores simples (como números o textos), también podemos guardar objetos (es decir, instancias de una clase que nosotros mismos definimos) dentro de los arrays.

Esto permite estructurar mejor la información, por ejemplo:

* Un vector de alumnos
* Una matriz de productos
* Un vector de empleados, etc.

📌 **Ejemplo:**

**public** class Alumno  
{  
    **public** string Nombre

**public** **int** Edad

**public** Alumno(string nombre, **int** edad)  
    {  
        Nombre = nombre;  
        Edad = edad;  
    }  
}

//Se crea el array de alumnos  
Alumno[] alumnos = **new** Alumno[3];

//Se crea cada nuevo alumno que deseamos guardar en el array y al mismo tiempo lo almacenamos en el vector  
alumnos[0] = **new** Alumno("Ana", 20);  
alumnos[1] = **new** Alumno("Luis", 22);  
alumnos[2] = **new** Alumno("Marta", 19);

**📝 Ejercicios**

# Crear una aplicación en Windows Forms que simule una tienda con 3 productos (nombre y stock). El usuario ingresará el nombre del producto en una caja de texto, y al hacer clic en un botón, el programa deberá mostrar si hay stock disponible. Condiciones:

# Debe usar una matriz bidimensional de string para guardar los datos de los productos.

# Al menos un producto debe tener stock 0 para validar ese caso