Crear una matriz

Las matrices se utilizan para almacenar múltiples valores en una sola variable, en lugar de declarar variables separadas para cada valor.

Para declarar una matriz, defina el tipo de variable con **corchetes** :

```
string[] cars;
```

Ahora hemos declarado una variable que contiene una matriz de cadenas.

Para insertarle valores, podemos usar una matriz literal: coloque los valores en una lista separada por comas, dentro de llaves:

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

Para crear una matriz de enteros, podría escribir:

```
int[] myNum = {10, 20, 30, 40};
```

Acceder a los Elementos de un Array

Accede a un elemento de matriz haciendo referencia al número de índice.

Esta sentencia accede al valor del primer elemento en **coches** :

Ejemplo

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

Console.WriteLine(cars[0]);

// Outputs Volvo
```

Nota: los índices de matriz comienzan con 0: [0] es el primer elemento. [1] es el segundo elemento, etc.

C# Bucle a través de matrices

Puede recorrer los elementos de la matriz con el for bucle y usar la Length propiedad para especificar cuántas veces debe ejecutarse el bucle.

El siguiente ejemplo genera todos los elementos en la matriz **de automóviles**:

Ejemplo

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

for (int i = 0; i < cars.Length; i++)

{

  Console.WriteLine(cars[i]);
```

El bucle foreach

También hay un `foreach` bucle, que se usa exclusivamente para recorrer elementos en una **matriz** :

Sintaxis

```
foreach (type variableName in arrayName)

{

  // code block to be executed

}
```

El siguiente ejemplo genera todos los elementos en la matriz **de autos** , usando un `foreach`bucle:

Ejemplo

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

foreach (string i in cars)
```

```
{
  Console.WriteLine(i);
}
```

El ejemplo anterior se puede leer así: **para cada** `string` elemento (llamado **i** - como en **i** ndex) en **cars** , imprime el valor de **i** .

Si compara el `for` ciclo y `foreach` el ciclo, verá que el `foreach` método es más fácil de escribir, no requiere un contador (usando la `Length` propiedad) y es más legible.

Ordenar una matriz

Hay muchos métodos de matriz disponibles, por ejemplo `Sort()`, que ordena una matriz alfabéticamente o en orden ascendente:

Ejemplo

```
// Sort a string
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
Array.Sort(cars);
foreach (string i in cars)
{
  Console.WriteLine(i);
}


// Sort an int
int[] myNumbers = {5, 1, 8, 9};
Array.Sort(myNumbers);
foreach (int i in myNumbers)
{
  Console.WriteLine(i);
```

```
}
```

Espacio de nombres System.Linq namespace

Otros métodos de matriz útiles, como `Min`, `Max`y `Sum`, se pueden encontrar en el `System.Linq`espacio de nombres:

Ejemplo

```
using System;
using System.Linq;


namespace MyApplication
{
  class Program
  {
    static void Main(string[] args)
    {
      int[] myNumbers = {5, 1, 8, 9};
      Console.WriteLine(myNumbers.Max());  // returns the largest
value
      Console.WriteLine(myNumbers.Min());  // returns the smallest
value
      Console.WriteLine(myNumbers.Sum());  // returns the sum of
elements
    }
  }
}
```

Matrices multidimensionales

En el capítulo anterior, aprendió acerca de las matrices , que también se conocen como **matrices de una sola dimensión** . Estos son geniales, y algo que usará mucho mientras programa en C#. Sin embargo, si desea almacenar datos en formato tabular, como una tabla con filas y columnas, debe familiarizarse con las **matrices multidimensionales** .

Una matriz multidimensional es básicamente una matriz de matrices.

Las matrices pueden tener cualquier número de dimensiones. Los más comunes son los arreglos bidimensionales (2D).

Matrices bidimensionales

Para crear una matriz 2D, agregue cada matriz dentro de su propio conjunto de llaves e inserte una coma (`,`) dentro de los corchetes:

Ejemplo

```
int[,] numbers = { {1, 4, 2}, {3, 6, 8} };
```

Es bueno saberlo: la coma simple `[,]`especifica que la matriz es bidimensional. Una matriz tridimensional tendría dos comas: `int[,,]`.

`numbers`ahora es una matriz con dos matrices como sus elementos. El primer elemento del arreglo contiene tres elementos: 1, 4 y 2, mientras que el segundo elemento del arreglo contiene 3, 6 y 8. Para visualizarlo, piense en el arreglo como una tabla con filas y columnas:

	COLUMN 0	COLUMN 1	COLUMN 2
ROW 0	1	4	2
ROW 1	3	6	8

Elementos de acceso de una matriz 2D

Para acceder a un elemento de una matriz bidimensional, debe especificar dos índices: uno para la matriz y otro para el elemento dentro de esa matriz. O mejor aún, con la visualización de la tabla en mente; uno para la fila y otro para la columna (vea el ejemplo a continuación).

Esta declaración accede al valor del elemento en la **primera fila (0)** y **la tercera columna (2)** de la `numbers`matriz:

Ejemplo

```
int[,] numbers = { {1, 4, 2}, {3, 6, 8} };

Console.WriteLine(numbers[0, 2]);  // Outputs
```