**Array irregolari (Jagged Array)**

http://msdn.microsoft.com/it-it/library/2s05feca.aspx

Una matrice irregolare è una matrice i cui elementi sono costituiti da matrici. Gli elementi di una matrice irregolare possono presentare dimensioni e misure differenti. Una matrice irregolare è anche detta "matrice di matrici". Negli esempi di codice riportati di seguito viene illustrato come dichiarare, inizializzare e accedere alle matrici irregolari.

Di seguito è riportata la dichiarazione di una matrice unidimensionale di tre elementi, ciascuno dei quali è una matrice unidimensionale di valori integer:

int[][] jaggedArray = new int[3][];

Per poter utilizzare jaggedArray, è necessario che i corrispondenti elementi siano stati inizializzati. È possibile inizializzare gli elementi nel modo seguente:

jaggedArray[0] = new int[5];

jaggedArray[1] = new int[4];

jaggedArray[2] = new int[2];

Ciascuno degli elementi è costituito da una matrice unidimensionale di valori integer. Il primo elemento è una matrice di 5 valori integer, il secondo una matrice di 4 valori integer e il terzo una matrice di 2 valori integer.

È anche possibile utilizzare inizializzatori per immettere i valori negli elementi delle matrici. In questo caso, non occorre conoscere la dimensione delle matrici. Ad esempio:

jaggedArray[0] = new int[] { 1, 3, 5, 7, 9 };

jaggedArray[1] = new int[] { 0, 2, 4, 6 };

jaggedArray[2] = new int[] { 11, 22 };

È inoltre possibile inizializzare la matrice al momento della dichiarazione, come nell'esempio che segue:

int[][] jaggedArray2 = new int[][]

{

new int[] {1,3,5,7,9},

new int[] {0,2,4,6},

new int[] {11,22}

};

È possibile utilizzare la seguente forma abbreviata. Si noti che è possibile omettere l'operatore **new** poiché non è prevista alcuna inizializzazione predefinita per gli elementi:

int[][] jaggedArray3 =

{

new int[] {1,3,5,7,9},

new int[] {0,2,4,6},

new int[] {11,22}

};

Gli elementi di una matrice irregolare sono tipi di riferimento inizializzati su **null**.

**Singoli elementi di una matrice**

È possibile accedere a singoli elementi di una matrice, come negli esempi che seguono:

// Assign 77 to the second element ([1]) of the first array ([0]):

jaggedArray3[0][1] = 77;

// Assign 88 to the second element ([1]) of the third array ([2]):

jaggedArray3[2][1] = 88;

**Array irregolari e matrici multidimensionali**

È possibile combinare matrici irregolari e matrici multidimensionali. Di seguito sono riportate la dichiarazione e l'inizializzazione di una matrice irregolare unidimensionale che contiene elementi matrice bidimensionali con dimensioni diverse:

int[][,] jaggedArray4 = new int[3][,]

{

new int[,] { {1,3}, {5,7} },

new int[,] { {0,2}, {4,6}, {8,10} },

new int[,] { {11,22}, {99,88}, {0,9} }

};

È possibile accedere a singoli elementi, come illustrato in questo esempio, in cui viene visualizzato il valore dell'elemento [1,0] della prima matrice (valore 5):

System.Console.Write("{0}", jaggedArray4[0][1, 0]);

**Proprietà Length**

La proprietà Length restituisce il numero di matrici contenute nella matrice irregolare. Si supponga, ad esempio, che sia stata dichiarata la matrice jaggedArray precedente. In questo caso, la riga

Console.WriteLine(jaggedArray.Length); //Valore restituito: 3

Per ottenere il numero di elementi delle matrici contenute:

Console.WriteLine(jaggedArray[1].Length); //Valore restituito: 4

**Esempio: Visualizzazione degli elementi di un jagged array con ciclo for**

In questo esempio viene creata una matrice i cui elementi sono costituiti da matrici. Ciascun elemento della matrice ha una dimensione differente.

static void Main()

{

// Declare the array of two elements:

int[][] arr = new int[2][];

// Initialize the elements:

arr[0] = new int[5] { 1, 3, 5, 7, 9 };

arr[1] = new int[4] { 2, 4, 6, 8 };

// Display the array elements:

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

{

System.Console.Write("Element({0}): ", i);

for (int j = 0; j < arr[i].Length; j++)

{

System.Console.Write("{0}{1}", arr[i][j], j == (arr[i].Length - 1) ? "" : " ");

}

System.Console.WriteLine();

}

}

In questo esempio l’output prodotto sarà il seguente

Element(0): 1 3 5 7 9

Element(1): 2 4 6 8

**Esempio: foreach e jagged array**

static void Main(string[] args)

{

int[][] jaggedArray = new int[3][];

jaggedArray[0] = new int[] { 1, 3, 5, 7, 9 };

jaggedArray[1] = new int[] { 0, 2, 4, 6 };

jaggedArray[2] = new int[] { 11, 22 };

foreach (int[] array in jaggedArray)

{

Console.WriteLine("Starting new array");

foreach (int item in array)

{

Console.WriteLine(" Item: {0}", item);

}

}

}

In questo esempio l’output prodotto sarà il seguente

Starting new array

Item: 1

Item: 3

Item: 5

Item: 7

Item: 9

Starting new array

Item: 0

Item: 2

Item: 4

Item: 6

Starting new array

Item: 11

Item: 22

**Esempio: Passaggio di un jagged array come parametro**

static void Main()

{

// Declare the array of two elements:

int[][] arr = new int[2][];

// Initialize the elements:

arr[0] = new int[5] { 1, 3, 5, 7, 9 };

arr[1] = new int[4] { 2, 4, 6, 8 };

MyMethod(arr);

}

static void MyMethod(int[][] jarr)

{

// Display the array elements:

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

{

Console.Write("Element({0}): ", i);

for (int j = 0; j < jarr[i].Length; j++)

{

Console.Write("{0}{1}", jarr[i][j], j == (jarr[i].Length - 1) ? "" : " ");

}

Console.WriteLine();

}

}