**C# - Array**

http://msdn2.microsoft.com/it-it/library/9b9dty7d(VS.80).aspx

Una matrice è una struttura di dati contenente una serie di variabili dello stesso tipo. Le matrici vengono dichiarate con un tipo:

type[] arrayName;

Negli esempi riportati di seguito vengono create matrici unidimensionali, multidimensionali e irregolari:

class TestArraysClass

{

static void Main()

{

// Declare a single-dimensional array

int[] array1 = new int[5];

// Declare and set array element values

int[] array2 = new int[] { 1, 3, 5, 7, 9 };

// Alternative syntax

int[] array3 = { 1, 2, 3, 4, 5, 6 };

// Declare a two dimensional array

int[,] multiDimensionalArray1 = new int[2, 3];

// Declare and set array element values

int[,] multiDimensionalArray2 = { { 1, 2, 3 }, { 4, 5, 6 } };

// Declare a jagged array

int[][] jaggedArray = new int[6][];

// Set the values of the first array in the jagged array structure

jaggedArray[0] = new int[4] { 1, 2, 3, 4 };

}

}

# Cenni preliminari sulle matrici

Di seguito sono riportate le caratteristiche delle matrici:

* Una matrice può essere unidimensionale, multidimensionale o irregolare.
* Il valore predefinito degli elementi numerici della matrice è impostato su zero, mentre gli elementi di riferimento sono impostati su null.
* Gli elementi di una matrice irregolare sono tipi di riferimento inizializzati su **null**.
* Le matrici sono a indice zero. Una matrice con n elementi viene indicizzata da 0 a n-1.
* Gli elementi di una matrice possono essere di qualsiasi tipo, anche di tipo matrice.

**Matrici come oggetti**

In C# le matrici sono in realtà oggetti e non solo aree indirizzabili di memoria contigua come in C e C++. *Array* rappresenta il tipo di base astratto di tutti i tipi di matrice. È quindi possibile utilizzare le proprietà e gli altri membri di classe previsti da *Arra***y**. È ad esempio possibile utilizzare la proprietà *Length* per ottenere la lunghezza di una matrice. Nel codice riportato di seguito viene assegnata la lunghezza della matrice numbers, 5, a una variabile denominata lengthOfNumbers:

int[] numbers = { 1, 2, 3, 4, 5 };

int lengthOfNumbers = numbers.Length;

La classe **System.Array** offre numerosi altri utili metodi e proprietà, quali metodi per ordinare, copiare le matrici ed effettuare ricerche all'interno di esse.

Esempio

In questo esempio viene utilizzata la proprietà *Rank* per visualizzare il numero di dimensioni di una matrice.

class TestArraysClass

{

static void Main()

{

// Declare and initialize an array:

int[,] theArray = new int[5, 10];

System.Console.WriteLine("The array has {0} dimensions.", theArray.Rank);

}

}

In questo esempio l’output prodotto sarà il seguente

The array has 2 dimensions.

**Array unidimensionali**

È possibile dichiarare una matrice composta da cinque valori integer, come nell'esempio che segue:

int[] array = new int[5];

Questa matrice contiene gli elementi da array[0] a array[4] compresi. L'operatore *new* viene utilizzato per creare la matrice e inizializzarne gli elementi con i corrispondenti valori predefiniti. In questo esempio tutti gli elementi della matrice sono inizializzati a zero.

Allo stesso modo è possibile dichiarare una matrice contenente elementi stringa. Esempio:

string[] stringArray = new string[6];

Inizializzazione di una matrice

È possibile inizializzare una matrice al momento della dichiarazione, nel qual caso non è necessario specificarne il numero di dimensioni, in quanto è già fornito dal numero di elementi presente nell'elenco di inizializzazione. Esempio:

int[] array1 = new int[5] { 1, 3, 5, 7, 9 };

Allo stesso modo, è possibile inizializzare una matrice di stringhe. Di seguito è illustrata la dichiarazione di una matrice di stringhe in cui ciascun elemento della matrice è inizializzato con il nome di un giorno della settimana:

string[] weekDays = new string[] { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };

Quando si inizializza una matrice al momento della dichiarazione, è possibile utilizzare le seguenti abbreviazioni:

int[] array2 = { 1, 3, 5, 7, 9 };

string[] weekDays2 = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };

È possibile dichiarare una variabile di matrice senza inizializzazione, ma sarà necessario utilizzare l'operatore *new* quando si assegna una matrice a questa variabile. Esempio:

int[] array3;

array3 = new int[] { 1, 3, 5, 7, 9 }; // OK

//array3 = {1, 3, 5, 7, 9}; // Error

**Array multidimensionali**

Le matrici possono avere più di una dimensione. La dichiarazione che segue, ad esempio, crea una matrice bidimensionale di quattro righe e due colonne:

int[,] array = new int[4, 2];

La dichiarazione che segue crea inoltre una matrice a tre dimensioni, 4, 2 e 3:

int[, ,] array1 = new int[4, 2, 3];

Inizializzazione di una matrice

È possibile inizializzare la matrice al momento della dichiarazione come illustrato nel seguente esempio:

int[,] array2D = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };

int[, ,] array3D = new int[,,] { { { 1, 2, 3 } }, { { 4, 5, 6 } } };

È inoltre possibile inizializzare la matrice senza specificarne il numero di dimensioni:

int[,] array4 = { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };

Se si sceglie di dichiarare una variabile di matrice senza inizializzazione, sarà necessario utilizzare l'operatore **new** per assegnare una matrice alla variabile. Esempio:

int[,] array5;

array5 = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } }; // OK

//array5 = {{1,2}, {3,4}, {5,6}, {7,8}}; // Error

È inoltre possibile assegnare un valore a un elemento di matrice, ad esempio:

array5[2, 1] = 25;

Nell'esempio di codice riportato di seguito le variabili di matrice vengono inizializzate in base ai valori predefiniti, ad eccezione delle matrici irregolari:

int[,] array6 = new int[10, 10];

**Utilizzo di foreach con Array**

In C# è disponibile anche l'istruzione foreach. Questa istruzione offre un metodo semplice e diretto per scorrere gli elementi di una matrice. Il codice che segue, ad esempio, consente la creazione di una matrice denominata numbers e consente di scorrere tale matrice con l'istruzione **foreach**:

int[] numbers = { 4, 5, 6, 1, 2, 3, -2, -1, 0 };

foreach (int i in numbers)

{ System.Console.WriteLine(i); }

Con le matrici multidimensionali è possibile utilizzare lo stesso metodo per scorrere gli elementi, ad esempio:

int[,] numbers2D = new int[3, 2] { { 9, 99 }, { 3, 33 }, { 5, 55 } };

foreach (int i in numbers2D)

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

L'output dell'esempio è il seguente:

9 99 3 33 5 55

Con le matrici multidimensionali, tuttavia, l'utilizzo di un ciclo for nidificato fornisce maggior controllo sugli elementi della matrice.

**Passaggio di Array come parametri**

È possibile passare matrici ai metodi come parametri. Le matrici sono infatti tipi di parametri, pertanto il metodo può modificare il valore degli elementi.

Passaggio di matrici unidimensionali come parametri

È possibile passare una matrice unidimensionale inizializzata a un metodo. Esempio:

PrintArray(theArray);

Il metodo chiamato nella riga sopra riportata può essere definito nel modo seguente:

void PrintArray(int[] arr)

{

// method code

}

È inoltre possibile inizializzare e passare una nuova matrice in un unico passaggio. Esempio:

PrintArray(new int[] { 1, 3, 5, 7, 9 });

Esempio 1

Nell'esempio che segue una matrice di stringhe viene inizializzata e passata come parametro al metodo PrintArray, in cui ne vengono visualizzati gli elementi:

class ArrayClass

{

static void PrintArray(string[] arr)

{

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

{

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

}

System.Console.WriteLine();

}

static void Main()

{

// Declare and initialize an array:

string[] weekDays = new string[] { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };

// Pass the array as a parameter:

PrintArray(weekDays);

}

}

Output 1

Sun Mon Tue Wed Thu Fri Sat

Passaggio di matrici multidimensionali come parametri

È possibile passare una matrice multidimensionale inizializzata a un metodo. Se ad esempio theArray è una matrice bidimensionale:

PrintArray(theArray);

Il metodo chiamato nella riga sopra riportata può essere definito nel modo seguente:

void PrintArray(int[,] arr)

{

// method code

}

È inoltre possibile inizializzare e passare una nuova matrice in un unico passaggio. Esempio:

PrintArray(new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } });

Esempio 2

In questo esempio una matrice bidimensionale viene inizializzata e passata al metodo PrintArray, in cui ne vengono visualizzati gli elementi.

class ArrayClass2D

{

static void PrintArray(int[,] arr)

{

// Display the array elements:

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 2; j++)

{

System.Console.WriteLine("Element({0},{1})={2}", i, j, arr[i, j]);

}

}

}

static void Main()

{

// Pass the array as a parameter:

PrintArray(new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } });

}

}

Output 2

Element(0,0)=1

Element(0,1)=2

Element(1,0)=3

Element(1,1)=4

Element(2,0)=5

Element(2,1)=6

Element(3,0)=7

Element(3,1)=8

**Passaggio di Array mediante ref e out**

Analogamente a tutti i parametri out, anche il parametro **out** di un tipo matrice deve essere assegnato prima dell'uso, ovvero assegnato dal chiamato. Esempio:

static void TestMethod1(out int[] arr)

{

arr = new int[10]; // definite assignment of arr

}

Come tutti i parametri [ref](http://msdn2.microsoft.com/it-it/library/14akc2c7%28VS.80%29.aspx), anche il parametro **ref** di un tipo matrice deve essere assolutamente assegnato dal chiamante. Non è pertanto necessario che venga assegnato dalla procedura che viene chiamata. Il parametro **ref** di un tipo matrice può risultare modificato in conseguenza della chiamata. È ad esempio possibile che alla matrice venga assegnato il valore [null](http://msdn2.microsoft.com/it-it/library/edakx9da%28VS.80%29.aspx) o che essa venga inizializzata con una matrice diversa. Esempio:

static void TestMethod2(ref int[] arr)

{

arr = new int[10]; // arr initialized to a different array

}

Nei due esempi che seguono viene illustrata la differenza tra **out** e **ref** quando vengono utilizzati per passare matrici a metodi.

Esempio 1

In questo esempio la matrice theArray viene dichiarata nel chiamante (il metodo Main) e inizializzata nel metodo FillArray. Gli elementi vengono quindi restituiti al chiamante e visualizzati.

class TestOut

{

static void FillArray(out int[] arr)

{

// Initialize the array:

arr = new int[5] { 1, 2, 3, 4, 5 };

}

static void Main()

{

int[] theArray; // Initialization is not required

// Pass the array to the callee using out:

FillArray(out theArray);

// Display the array elements:

System.Console.WriteLine("Array elements are:");

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

{

System.Console.Write(theArray[i] + " ");

}

}

}

Output 1

Array elements are:

1 2 3 4 5

Esempio 2

In questo esempio la matrice theArray viene inizializzata nel chiamante (il metodo Main) e passata al metodo FillArray utilizzando il parametro **ref**. Alcuni degli elementi della matrice vengono aggiornati nel metodo FillArray. Gli elementi vengono quindi restituiti al chiamante e visualizzati.

class TestRef

{

static void FillArray(ref int[] arr)

{

// Create the array on demand:

if (arr == null)

{

arr = new int[10];

}

// Fill the array:

arr[0] = 1111;

arr[4] = 5555;

}

static void Main()

{

// Initialize the array:

int[] theArray = { 1, 2, 3, 4, 5 };

// Pass the array using ref:

FillArray(ref theArray);

// Display the updated array:

System.Console.WriteLine("Array elements are:");

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

{

System.Console.Write(theArray[i] + " ");

}

}

}

Output 2

Array elements are:

1111 2 3 4 5555