Vettori (Array) in C#

Cos'è un Vettore

Un **Vettore** (o **Array Monodimensionale**) è una **Struttura Dati** che consente di memorizzare in RAM una **Sequenza di Dati Omogenei** (cioè dello *stesso tipo*).

Esempio:

55	27	63	34	88	21	15	73

Dichiarazione e Creazione di un Vettore

Un Vettore è un "*oggetto*" (tipo *riferimento*), quindi deve essere *dichiarato* e *creato*.

Per **Dichiarare un Vettore**, si usa la seguente forma:

<tipo> **[ ]** = *<nome-vettore>*

Le Parentesi Quadre **[ ]** poste dopo <*tipo*>, indicano che si sta dichiarando un Vettore e non una semplice variabile.

Esempio: per dichiarare un Vettore di nome **Vet**, destinato a memorizzare una sequenza di **numeri interi**, si scrive:

```
int [ ] = Vet ;
```

In genere, un Vettore viene **Dichiarato a Livello Globale**, ossia al di fuori di tutti i Sottoprogrammi-Evento, all'inizio del codice della Form:

Esempio:

```
namespace EsempioVettore
{
   public partial class Form1 : Form
   {
      public Form1()
      {  InitializeComponent();  }

      // area delle DICHIARAZIONI GLOBALI:
      int [ ]  Vet ;
```

Per **Creare un Vettore**, si usa la seguente forma:

<nome-vettore> = **new** *<tipo>* **[** *<numero-elementi>* **]**

<*numero-elementi*> indica il Numero di Elementi (cioè il Numero di "Caselle") con il quale strutturare e dimensionare il Vettore, in memoria RAM.

Esempio: per creare il vettore **Vet** precedentemente dichiarato, e dimensionarlo con 8 elementi, si scrive:

```
Vet = new int [ 8 ] ;
```

N.B.: questa istruzione, essendo una *istruzione di assegnazione* (e non una *dichiarazione*) deve essere usata *all'interno di un Sottoprogamma-Evento* (ad esempio, nell'evento Load della form) e non può essere inserita nell'area delle dichiarazioni globali.

E' possibile **Dichiarare e Creare un Vettore**, in un'unica istruzione:

<tipo> **[ ]** *<nome-vettore>* = **new** *<tipo>* **[** *<numero-elementi>* **]**

Esempio: per dichiarare e creare, in un colpo solo, il vettore **Vet**, dimensionandolo a 8 elementi, si può scrivere:

```
int [ ] Vet = new int [ 8 ] ;
```

Accesso ai Singoli Elementi di un Vettore

Gli elementi di un Vettore sono individuati tramite una **numerazione progressiva**, che indica la **Posizione** dell'elemento nella struttura:

Esempio:

Vet

55	27	63	34	88	21	15	73
0	1	2	3	4	5	6	7

La posizione del **primo elemento** è sempre **0** (struttura in "**Base Zero**").
La posizione dell'**ultimo elemento** è data da **Numero di Elementi – 1**.

L'**Accesso al Singolo Elemento** avviene scrivendo il *nome del vettore* seguito dalla *posizione* (**indice**) dell'elemento desiderato, chiusa fra *parentesi quadre*:

<nome-vettore> **[** *<indice>* **]**

La dicitura precedente è, a tutti gli effetti, **come una variabile** (del tipo del Vettore) e può, quindi, essere usata per *leggere il suo valore* o *assegnarvi un nuovo valore*.

Esempio:

Per accedere al contenuto dell'elemento di **Vet** in **posizione 5** e visualizzarlo in una Label, basta scrivere:

```
lblVis.Text = Vet [ 5 ] .ToString ( ) ;
```

Per assegnare il **valore 63** all'elemento di **Vet** in **posizione 3**, basta scrivere:

```
Vet [ 3 ] = 63 ;
```

Uso Parziale di un Vettore

Un vettore, generalmente, si **dimensiona al massimo** numero di elementi necessari, ma poi se ne utilizza **solo una parte**, in base a **quanti dati è necessario realmente gestire**:

Esempio - in una applicazione che gestisca l'Elenco dei Nominativi di una classe, si dimensiona il Vettore al **massimo numero possibile di studenti** (ad esempio, 30) ma poi, dovendo gestire una classe specifica, solo **una parte del vettore sarà usata.**

Quindi, la dichiarazione/creazione completa di un vettore, prevede anche l'uso di una **costante intera Max** (che indica il **massimo numero di elementi gestibili**) e di una **variabile intera N** (che indica il **numero di dati reali presenti nel vettore**):

Vet

dati reali	non utilizz.

0 *N-1* *N* *Max-1*

Esempio - il Vettore per l'Elenco degli Studenti potrebbe essere dichiarato/creato così:

```
int const Max = 30;
string[ ] ElencoClasse = new string[ Max ];
int N = 0;
```

Oltre ad indicare sempre il numero di dati reali presenti nel vettore, Il valore della variabile **N indica anche la prima posizione libera nel vettore**. Il valore **N-1 indica la posizione dell'ultimo dato** reale memorizzato nel vettore.

Inizializzazione di un Vettore

E' possibile **dichiarare vettori di qualsiasi tipo**. E' possibile anche **definire vettori i cui elementi sono degli "oggetti"** (anziché semplici valori di un certo tipo).

Esempi:

```
string[ ]  Elenco ;
double[ ]  Altezze ;
bool[ ]  Promossi ;
DateTime[ ]  DateDiNascita ; // … vettore i cui elementi sono Date!
```

Al momento della dichiarazione o creazione, è possibile, **Inizializzare il Vettore**, ossia attribuirvi, già dall'inizio, un elenco di valori (indicandoli fra parentesi graffe).

Esempi – E' possibile "inizializzare" al momento della creazione (quando si usa **new**)…

```
Vet  =  new  int[ ]  { 25,  78,  36,  84,  14 } ;
Giorni = new string[ ]  { "Lun","Mar","Mer","Gio","Ven","Sab", "Dom" } ;
```

… e anche al momento della semplice "dichiarazione" …

```
int[ ]  Vet  =  { 25,  78,  36,  84,  14 } ;
string[ ]  Giorni = { "Lun","Mar","Mer","Gio","Ven","Sab", "Dom" } ;
```

… ma non con una assegnazione diretta …

```
Vet  =  { 25,  78,  36,  84,  14 };     // … NO!  … ERRORE !
```

Scansione di un Vettore

E' ricorrente la necessità di analizzare, uno per uno, tutti gli elementi di un Vettore.

Chiameremo **Scansione di un Vettore** l'operazione che, *utilizzando un ciclo (iterazione)*, *accede a tutti gli elementi* del Vettore: più precisamente, **ogni passo del ciclo accede a un diverso elemento, dal primo all'ultimo**.

La **Scansione Completa di un Vettore** si realizza tramite un **Ciclo a Conteggio (for)**, il cui **contatore K** assume i valori di tutte le posizioni:

```
for ( int K = 0;   K <= N - 1;   K++)
```

Usando la notazione **Vet [K]** all'interno del ciclo, è possibile accedere al 1° elemento del vettore, poi al 2°, poi al 3°, ecc.

Ad ogni passo del ciclo, il valore di K cambia, quindi, la notazione **Vet [K] accederà, ad ogni passo, ad un elemento diverso**: al 1° passo **K=0** e la notazione *Vet[K]* accede a **Vet[0]**, cioè accede al 1° elemento; al 2° passo **K=1** e *Vet[K]* accede a **Vet[1]** , cioè al 2° elemento; … e così via fino all'ultimo elemento, in posizione N-1).

Esempio - Con il seguente codice, conti quanti 5 sono presenti in un vettore Vet di interi:

```
int Conta = 0;                          … dichiari e azzeri la variabile per contare i 5
for ( int K = 0;  K <= N-1;  K++)       … il ciclo ripete con K che varia da 0 a N-1
{
   if  ( Vet[K] == 5 )                  … verifichi se il numero in posizione K è un 5
   {
      Conta++ ;                         … se si, incrementi il contatore che conta i 5
   }
}
```

Al termine del ciclo for, nella variabile Conta, hai quanti 5 sono presenti in Vet.