LISTE TIPIZZATE e CLASSE List<T>

Un **oggetto di classe List<T>** (**Lista Tipizzata**) consente di *memorizzare e gestire una Sequenza di Dati Omogenei*, ossia tutti dello stesso tipo. Al posto della "**T**" è necessario indicare il *tipo di dato* che si desidera memorizzare.

La Lista Tipizzata è simile a un Vettore: il **Vettore** offre un accesso più veloce ai dati, ma è poco performante in caso di inserimenti/eliminazioni; la **Lista Tipizzata** offre più funzionalità, ma è meno efficiente nell'accesso ai dati.

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
```

Per utilizzare la *classe List<T>*, è necessario aggiungere, all'inizio del codice, la dicitura: **using System.Collections.Generics**

Essa attiva il ***namespace*** (spazio dei nomi … ossia la *libreria di oggetti*) che contiene le *classi Collection* e, quindi, anche la *classe List<T>*.

Per **Dichiarare e Creare un oggetto di classe List<T>** (ad esempio, per dati di tipo **string**), si procede come già visto:

```
List<string> Elenco;  // … dichiarazione          Elenco = new List<string>( );  // … creazione
```

```
List<string> Elenco = new List<string>( );   // … oppure, in un'unica istruzione: dichiarazione + creazione
```

La classe List<T> offre molti **Membri** (*proprietà, metodi, eventi, ecc.*), i più importanti dei quali sono di seguito esposti.

Il **metodo Add( )** consente di *Aggiungere un Nuovo Dato* in coda alla lista gestita dall'oggetto List<T>:

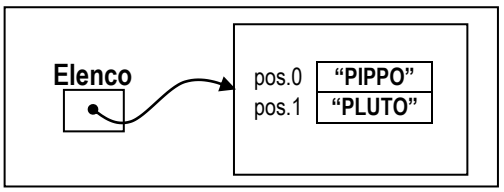
```
<istanza>.Add ( <dato-da-aggiungere> )
```

Il parametro <*dato-da-aggiungere*> è una *variabile* (o una *espressione*) il cui valore sarà il *nuovo dato* da aggiungere all'insieme. Esso *deve essere di tipo* <*T*>, ossia dello stesso tipo di cui è stato dichiarato l'oggetto List<T>.

Per aggiungere al nostro *List<string>* di nome *Elenco*, due nuovi dati **"PIPPO"** e **"PLUTO"**, entrambi *valori di tipo string*, scrivi così:

```
Elenco.Add ( "PIPPO" );
Elenco.Add ( "PLUTO" );
```

Nella figura a destra, puoi vedere l'*effetto* di queste istruzioni.



Gli elementi di un oggetto List<T> sono memorizzati *rispettando l'ordine in cui vengono aggiunti*. Ogni nuovo dato occupa una nuova posizione *a partire dalla posizione 0* (…come i Vettori, anche le Liste Tipizzate sono *in* ***Base Zero***).

La **proprietà Count** restituisce il *Numero di Elementi* presenti nella Lista Tipizzata:

```
<istanza>.Count          … restituisce un valore di tipo int
```

Si tratta di una proprietà **Read-Only** (di *Sola Lettura*): *non è possibile assegnarvi un valore*, si può solo "leggere".

E' utile ricordare che, essendo una proprietà, *<istanza>.Count* va usata *come se fosse una variabile*, quindi si può usare in una espressione, in una assegnazione, ecc.

Se vuoi visualizzare in una Label di nome *lbl*, quanti elementi sono presenti nell'oggetto *List<T>* di nome *Elenco*, scrivi:

```
lbl.text = Elenco.Count;
```

Se eseguiamo questa istruzione nella situazione indicata nella figura di sopra, nella Label appare il *numero* 2.

Il primo elemento di un oggetto List<T> è sempre in **posizione 0** *(prima posizione)*. Di conseguenza, l'ultimo elemento è sempre in posizione <*istanza*>**.Count – 1** *(ultima posizione)*.

Se nell'oggetto *List<T>* di nome *Elenco* hai aggiunto **10 dati**, allora *Elenco.Count restituisce 10*.
Il primo elemento sarà in posizione **0** e l'ultimo elemento sarà il posizione **Elenco.Count-1**, ossia in **posizione 9**.

Per **Accedere a un Singolo Elemento** di una Lista Tipizzata, si utilizza la stessa regola vista per i Vettori, indicando, fra *parentesi quadre*, la posizione dell'elemento desiderato:

<*istanza*> **[** <*posizione*> **]** *… accede all'elemento nella posizione indicata*

Il parametro <*posizione*> è di tipo Intero e deve contenere *la posizione dell'elemento desiderato*.

- Il valore del parametro <*posizione*> **DEVE *essere compreso* fra 0 e <*istanza*>.Count – 1** , perché deve indicare una posizione *realmente esistente* nell'oggetto List<T>. Se *Indice* non è compreso in questo intervallo, *viene generato un* ***errore in fase di esecuzione*** e il programma si arresta.

- Nell'esempio in figura, l'istruzione **MessageBox.Show(Elenco[1])** visualizza ***"PLUTO"***, cioè l'elemento in *posizione 1*.

Il **metodo Clear( )** provoca lo *Svuotamento della Lista Tipizzata* (dopo lo svuotamento la Lista ha ZERO elementi).

<*istanza*>**.Clear( )**

Il **metodo InsertAt( )** consente di *Inserire un Nuovo Dato in una specifica Posizione*.

<*istanza*>**.InsertAt (** <*posizione*>, <*nuovo-dato*>)

Il parametro <*posizione*> è di tipo Intero e deve contenere *una posizione valida in cui inserire il* <*nuovo-dato*>.

Il **metodo RemoveAt( )** consente di *Rimuovere dalla lista il dato che si trova in una specifica Posizione*.

<*istanza*>***.RemoveAt*** **(** <*posizione*> **)**

Il parametro <*posizione*> è di tipo Intero e deve contenere *la posizione dell'elemento da eliminare*.

- Nelle operazioni di Inserimento ed Eliminazione su Liste Tipizzate, **NON E' NECESSARIO LO SPOSTAMENTO** (shifting) visto nelle analoghe operazioni sui Vettori: i metodi **InsertAt** e **RemoveAt** garantiscono automaticamente l'integrità della sequenza dei dati, *senza necessità di spostamenti espliciti*.

- Se una *List*<*string*> di nome *Elenco* contiene inizialmente **0 - "PIPPO" / 1 - "PLUTO"**, l'istruzione:

  Elenco.InsertAt (1, "TOPOLINO");

  inserisce "TOPOLINO" in posizione 1, per cui la lista diventa: **0 - "PIPPO" / 1 - TOPOLINO / 2 - "PLUTO"**. L'istruzione:

  Elenco.RemoveAt (0);

  elimina l'elemento in posizione 0 (cioè "PIPPO"), per cui la lista diventa: **0 - TOPOLINO / 1 - "PLUTO"**.

L'istruzione foreach

L'istruzione iterativa **foreach** (*per ogni …*) è una *speciale istruzione iterativa (ciclo)* appositamente creata per **scandire Liste di Elementi** (come *Liste Tipizzate, Vettori*, ecc.) nel caso in cui non sia strettamente necessario l'uso di un "contatore":

```
foreach ( <tipo> <elemento> in <istanza> )
    {  … istruzioni che usano <elemento> …  }
```

Il ciclo *foreach* effettua un *numero di ripetizioni pari al numero di elementi* che sono presenti nell'oggetto <*istanza*> (Lista Tipizzata, Vettore, ecc.). Ad ogni passo, <*elemento*> indica un diverso elemento della lista, a partire dall'elemento in posizione 0, fino all'ultimo. Ovviamente è possibile terminare forzatamente il ciclo usando l'istruzione "break".

- Ad esempio, il seguente semplicissimo codice …

```
int Somma = 0;
foreach ( int Numero in ElencoDati )
    Somma = Somma + Numero;
```

… calcola la somma di tutti i dati contenuti nella Lista Tipizzata **ElencoDati** di classe **List<int>**.

Ad ogni passo, la variabile *Numero*, rappresenta un *diverso elemento* della Lista Tipizzata, dal primo all'ultimo. Questo ciclo effettua quindi un numero di ripetizioni pari a *ElencoDati.Count*.