Liste concatenate

La **lista concatenata** (in inglese *linked list*) è una struttura di dati in cui gli elementi sono ordinati linearmente. Ogni elemento conosce qual è il suo successore e in questo modo, a partire dal primo elemento è possibile ricostruire tutti gli elementi presenti nella lista.

Oltre a gestire strutture ordinate, la lista ha il vantaggio di inserire e cancellare i dati in modo più veloce rispetto ad altre soluzioni. Con le liste concatenate, dopo che è stata trovata la posizione, l'inserimento è immediato. La stessa operazione, eseguita con un array, risulta molto più costosa in quanto, per fare spazio al nuovo elemento, si devono spostare tutti gli elementi di un posto.

Una lista può essere vista come una successione di nodi, in cui ogni nodo è costituito dai dati relativi all'elemento più un riferimento al nodo successivo. Per tenere traccia di questa successione è necessario conoscere il riferimento al primo elemento della lista (*testa*) e all'ultimo elemento (*coda*). Il riferimento alla *testa* è utile perché, partendo da questo, è possibile scorrere l'intera lista, eseguendo un'operazione chiamata di **attraversamento**. Il riferimento alla *coda* è utile per aggiungere rapidamente un elemento in fondo alla lista senza doverla scorrere per recuperare l'ultimo nodo.

In Visual Basic la lista concatenata è rappresentata dalla classe **LinkedList**.
Più precisamente, la classe è utilizzata nella forma **LinkedList(Of T)**, indicando tra parentesi dopo *Of* il tipo degli elementi della lista.
I nodi della lista concatenata sono oggetti di tipo **LinkedListNode(Of T)**.

La differenza tra la classe *List* e la classe *LinkedList* è la seguente.
List è un array di elementi e l'accesso avviene tramite un indice; è una struttura lenta nelle operazioni di inserimento e rimozione degli elementi.
Nella *LinkedList* ogni nodo conosce quello che lo precede e quello che lo segue. Le operazioni di inserimento sono più veloci, prima o dopo un determinato nodo, oppure direttamente in testa o in coda.
Di contro una *LinkedList* richiede più memoria rispetto alla *List*, perché deve ricordare anche i riferimenti ai nodi precedente e successivo.

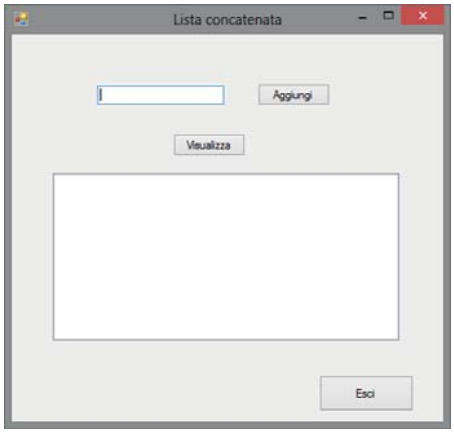
Il seguente diagramma descrive la classe, con alcune tra le proprietà e i metodi della classe.

LinkedList (Of T)
Count First Last
AddFirst AddLast AddAfter AddBefore Remove RemoveFirst RemoveLast ClearContains

Le proprietà restituiscono il numero di elementi contenuti attualmente nella lista, il primo nodo e l'ultimo nodo.

I metodi gestiscono le operazioni di inserimento e rimozioni dei nodi, la rimozione di tutti gli elementi e il controllo per determinare se un nodo è presente nella lista.

Per testare l'uso di una lista concatenata di nomi, predisponiamo la seguente interfaccia, che contiene la casella di testo (*txtNodo*) per inserire un nome e una ListBox (*lstNomi*) per visualizzare il contenuto della lista.



Codice Visual Basic

```
Imports System.Collections.Generic
Public Class Form1
   Dim Nomi As New LinkedList(Of String)

   Private Sub btnAggiungi_Click(sender As System.Object, e As
System.EventArgs) Handles btnAggiungi.Click
      If Nomi.Count = 0 Then
         Nomi.AddFirst(txtNodo.Text)
      Else
         Nomi.AddLast(txtNodo.Text)
      End If
      txtNodo.Clear()
      txtNodo.Focus()
   End Sub

   Private Sub btnVisualizza_Click(sender As System.Object, e As
System.EventArgs) Handles btnVisualizza.Click
      lstNomi.Items.Clear()
      For Each nodo As String In Nomi
         lstNomi.Items.Add(nodo)
      Next
   End Sub

   Private Sub btnEsci_Click(sender As System.Object, e As
System.EventArgs) Handles btnEsci.Click
      End
   End Sub
End Class
```