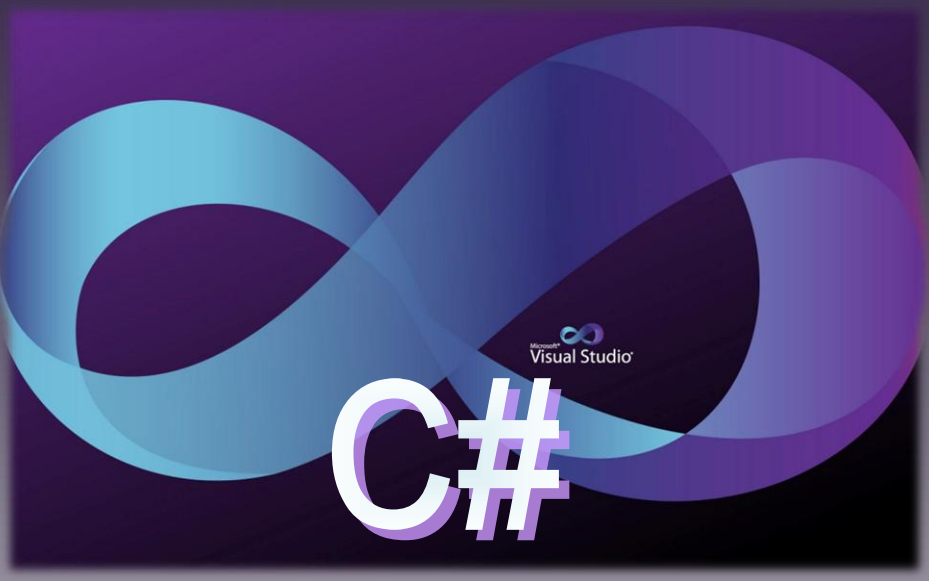
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

ALTRE COLLEZIONI DI DATI

DISPENSA 11.03

11-03_Altre_Liste_[ver_15]

Versione del: 07/11/2015
Revisione numero: 15

Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

ALTRE COLLEZIONI DI VISUAL STUDIO

LE LISTE HASH

SCOPO E FUNZIONAMENTO DELLE LISTE HASH

Mentre in C# le Liste permettono di associare un intero (indice) ad un elemento, esistono delle collezioni che associano una informazione non int (es. string, double per esempio) ad un'altra informazione. Puoi pensare queste collezioni come un vettore, che però al posto degli indici usa altre informazioni. Qui ne vedremo qualcuno.

LA LIBRERIA OBBLIGATORIA

Per usare una collezione occorre aggiungere una direttiva **using** di inclusione dello spazio **System.Collections**. Prima di usare perciò dovrai scrivere:

```
using System.Collections;
```

dopo le altre direttive **using** del tuo programma.

HASHTABLE (VETTORE ASSOCIATIVO)

È una classe con cui collezionare elementi. La dichiarazione di un oggetto HashTable è:

```
HashTable cambio;
```

che dichiara il nome ma non lo istanzia. Se vuoi anche istanziarlo devi fare:

```
HashTable cambio = new HashTable();
```

ovviamente "cambio" è un nome qualsiasi; potevo usare pippo.

OPERAZIONI SULLA TABELLA ASSOCIATIVA

Per cancellare qualsiasi elemento dalla tabella usa il metodo Clear():

```
cambio.Clear();
```

Per inserire un elemento nell'elenco associativo puoi semplicemente assegnarlo:

```
cambio ["dollari"] = 1.3135;
cambio ["yen"] = 112.1437;
cambio ["sterline"] = 0.8447;
```

adesso nell'elenco esistono tre elementi, che hanno per indice nomi di monete straniere, e come valore il loro cambio (approssimativo a settembre 2010).

Per esplorare il vettore associativo poso usare foreach; però ogni elemento del vettore è di tipo DictionaryEntry (un tipo speciale apposta per la lista associativa).

```
foreach (DictionaryEntry money in cambio)
{
  double valore = (double)money.Value; //prende il valore
  string nome = (string)money.Key; //chiave
  Console.WriteLine("10 euro = " + (valore*10) + " " + nome);
}
```

Poiché la lista accetta oggetti puoi archiviare elementi diversi tra loro.

```
cambio ["sterline"] = 0.8447;
cambio ["uffa"] = "pippo";
```

Tuttavia di solito conviene usare una lista di tipi omogenei.

Per accedere ad un singolo elemento è possibile usare la notazione vettoriale con un indice tra parentesi quadre; però il valore reso è un oggetto e non un valore elementare. Per esempio:

```
double valore = cambio["yen"]; //rende un errore di tipo
```

solleva un errore poiché `cambio["yen"]` non rende un numero ma un oggetto. Tuttavia:

```
double valore = (double)cambio["yen"];
Console.WriteLine(valore);
Console.WriteLine(valore);
```

```
double valore = (double)cambio["yen"];
potrebbe funzionare.
```

Per memorizzare dati puoi anche usare il metodo **Add** che richiede di specificare prima la chiave e poi il valore; per esempio:

```
cambio.Add("dinari", 100.00); // equivale a: cambio["dinaro"]=100.00;
```

Per eliminare dati puoi usare il metodo **Remove** che richiede di specificare la chiave; per esempio:

```
cambio.Remove("dinari");
```

Per sapere quanti elementi ci sono nella lista puoi usare la proprietà **Count**:

```
int quanti = cambio.Count;
```

LA COLLEZIONE SORTEDLIST

VETTORE ASSOCIATIVO ORDINATO

È una classe con cui collezionare elementi mantenendoli ordinati. La dichiarazione di un oggetto **SortedList** è:

```
SortedList prezzi;
```

che dichiara il nome ma non lo istanzia. Se vuoi anche istanziarlo devi fare:

```
SortedList prezzi = new SortedList();
```

ovviamente **prezzi** è un nome qualsiasi; potevo usare **pippo**.

OPERAZIONI SUL VETTORE ASSOCIATIVO ORDINATO

Le operazioni sulla tabella sono identiche a **HashTable**. Quindi puoi usare:

```
prezzi.Clear();
prezzi["aranciata"] = 1.35;
prezzi["the freddo"] = 0.65;
prezzi["mela"] = 0.35;
prezzi["latte"] = 0.65;
prezzi["pizza cacao"] = 2.65;
prezzi.Add("pizza aglio", 2.80);
prezzi.Remove("latte");
prezzi["panino bianco"] = 1.45;
prezzi["panino nero"] = 3.65;
int quanti = prezzi.Count;
```

tuttavia quando farai:

```
foreach (DictionaryEntry prodotto in prezzi)
{
  double valore = (double)prodotto.Value; // prezzo del prodotto
  string nome = (string)prodotto.Key; // nome del prodotto
  MessageBox.Show(nome + ": " + valore);
}
```

L'elenco sarà restituito in ordine alfabetico.

ESERCIZI

ESERCIZI SULLE COLLEZIONI

ESERCIZI SULLE COLLEZIONI HASH

Esercizio 1) CANI DI RAZZA

Immagine di voler associare ad ogni nome di cane la sua razza; prepara un programma visuale che permette di leggere tramite due caselle di testo il nome e la razza di un cane e poi di memorizzarla in una tabella associativa; prepara poi in un listBox l'elenco dei nomi di cane e ogni volta che fai clic su uno di essi compare una finestra di messaggio che ci informa sulla sua razza. Prova anche a realizzare un sistema per eliminare dalla lista un cane.

Esercizio 2) STUDENTI

Immagina di voler preparare un programma per uno studente (solo uno); il programma memorizza per ciascuna materia il voto finale (da 1 a 10). L'interfaccia permette di memorizzare per ciascuna materia il voto; ogni volta che lo si memorizza, l'interfaccia ci informa sulla media complessiva e su quante materie sono insufficienti e quante invece non hanno ancora una valutazione.

Esercizio 3) HASH DI STRUTTURE

Di sicuro ti sei chiesto se sia possibile costruire una tabella associativa di strutture; prova a dichiarare una struttura (es. studente); prova poi a dichiarare una HashTable e infine prova a memorizzare in essa una struttura funziona?

Considera il seguente esempio:

```
public struct Scatola
{
    public int altezza, larghezza, profondità;
    public Scatola(int a, int l, int p)
    {
          altezza = a; larghezza = l; profondità = p;
     }
}
//---
    Hashtable magazzino = new Hashtable();
    Scatola s = new Scatola(10, 20, 30);
    magazzino["mia"] = s;
```

ESERCIZI SULLE COLLEZIONI SORTEDLIST

Esercizio 4) DISTRIBUTORE DI ALIMENTI

Un distributore di alimenti viene programmato per mostrare il prezzo di ciascun prodotto. L'interfaccia permette di inserire un prodotto e il suo prezzo in euro (2 decimali). I prodotti inseriti sono immediatamente visualizzati in un listBox che mostra solo i nomi dei prodotti. Quando si sceglie un prodotto nell'elenco, subito il programma mostra il prezzo in una casella di testo. Prova anche a realizzare un modo per eliminare un prodotto dall'elenco.

Esercizio 5) RADIO DIGITALE

Una radio digitale deve associare a ciascun nome di radio la sua frequenza (2 decimali) per sintonizzare correttamente la ricevente. L'interfaccia permette di inserire un nome di radio (es. radio Angioy) e la sua frequenza (es. 111.11). le radio inserite sono immediatamente visualizzate in un listBox che mostra solo i nomi delle radio. Quando si sceglie una radio nell'elenco, subito il programma mostra la sua frequenza in una casella di testo. Prova anche a realizzare un modo per eliminare una radio dall'elenco.

Esercizio 6) SORTEDLIST DI STRUTTURE

Sarà possibile costruire una tabella associativa di strutture? Prova a dichiarare una struttura interrogazione composta da nome, data, materia e voto; prova poi a dichiarare una SortedList e infine prova a memorizzare in essa una struttura: funziona? Quale potrebbe essere la chiave per le interrogazioni?

ESERCIZI SU ARRAYLIST, FUNZIONI E STRUTTURE

Esercizio 7) DI RIEPILOGO

Definire una struttura Scatola composta di 3 dimensioni (intere), della tara (peso del contenitore) e del peso massimo trasportabile;

Definire un ArrayList di 100 scatole.

Esercizio 8) DI RIEPILOGO

Definire una funzione che genera una nuova scatola, con dimensioni e pesi generate casualmente.

Esercizio 9) DI RIEPILOGO

Definire una funzione che preso un ArrayList di scatole alloca memoria per 100 scatole e per ciascuna cella genera una scatola richiamando la funzione apposita.

Esercizio 10) DI RIEPILOGO

Definire una funzione che presa una scatola, calcola e restituisce il suo volume (prodotto delle tre dimensioni).

Esercizio 11) DI RIEPILOGO

Definire una funzione che preso un ArrayList di scatole cerca la scatola col massimo volume e lo restituisce.

Esercizio 12) DI RIEPILOGO

Definire una funzione che preso un ArrayList di scatole cerca la scatola con massima tara e restituisce il suo volume.

Esercizio 13) DI RIEPILOGO

Definire una funzione che presa una scatola, incrementa di +1 ciascuna delle tre dimensioni.

Esercizio 14) DI RIEPILOGO

Definire una funzione che preso un ArrayList di scatole incrementa di +1 ciascuna delle tre dimensioni di ciascuna cella (richiamare la funzione precedente).

Esercizio 15) DI RIEPILOGO

Definire una funzione che preso un ArrayList di scatole ed un peso, costruisce e restituisce un nuovo ArrayList con le scatole di peso inferiore a quello indicato.

Esercizio 16) DI RIEPILOGO

Definire una funzione che preso un ArrayList di scatole ed un numero intero, restituisce true se è presente una scatola di volume identico al valore numerico indicato.

SOMMARIO