Le collezioni di oggetti

Le collezioni sono strutture dati fondamentali quando si tratta di memorizzare insiemi di oggetti, più o meno grandi, in memoria.

Gli array ordinari hanno delle limitazioni, ad esempio le loro dimensioni sono fissate in fase di istanziazione e non più possibile variarle, inoltre l'unico modo per leggere e scrivere elementi, è attraverso l'indicizzazione. Nel caso in cui si necessiti di maggiore flessibilità, ad esempio per la ricerca di un elemento, per l'ordinamento, per la rimozione di un elemento, o altro, il namespace **System.Collections** contiene classi adatte ad ogni scopo, permettendo ad esempio di creare e manipolare liste ordinate, code, pile, tabelle hash, dizionari. Tali classi sono basate su delle interfacce che standardizzano i metodi per trattare tali insiemi di oggetti, ad esempio ogni collezione di oggetti implementa l'interfaccia ICollection, che espone la proprietà Count per ottenere il numero di elementi di una collezione ed il metodo CopyTo per copiare i suoi elementi in un array. Rimarchiamo inoltre il fatto che l'interfaccia ICollection deriva a sua volta da IEnumerable, che espone il metodo GetEnumerator, ed è questo che permetterà ad esempio la possibilità di iterare, in modo sequenziale, gli elementi di una collezione per mezzo della già vista istruzione foreach.

LISTE

List <structlibri> lista =new List < structlibri >

Vedere metodi e proprietà dalle dispense NetFramework – appendice pag.26

Esempio

```
public struct libro
    {
        public int id;
        public string titolo;
        public string autore;
    }
    List<libro> listaLibri=new List<libro>();
    private void btnListe_Click(object sender, EventArgs e)
    {
        libro l;
        l.id = Convert.ToInt32(textBox1.Text);
        l.titolo = textBox2.Text;
        l.autore = textBox3.Text;

        listaLibri.Add(l);
    }
    private void btnVisLista_Click(object sender, EventArgs e)
    {
        int i = 0;
        foreach (libro l in listaLibri)
        {
            MessageBox.Show(l.titolo);
            i++;
        }
    }
   }
```

Al posto delle struct si possono utilizzare oggetti e costruire così un a List di object

Esempio

```
public class book
        {
            public int id;
            public string title;
            public string autor;
            public book(int a, string b, string c)
            {
                this.id = a;
                this.title = b;
                this.autor = c;
            }
         }

List<book> collectionbook =new List<book>();

private void btnListe_Click(object sender, EventArgs e)
        {
            book b = new book(Convert.ToInt32(textBox1.Text), textBox2.Text, textBox3.Text);

            collectionbook.Add(b);
        }
```

Metodi Find e FindAll

inserisce in un oggetto **ris** di classe book un oggetto **bf** che soddisfa il predicato indicato in parentesi

```
book ris= collectionbook.Find(bf=> bf.title=="VS Programming");
MessageBox.Show(ris.autor);
```

inserisce in una lista di appoggio di oggetti di classe book tutti gli oggetti che soddisfano il predicato indicato in parentesi

```
List<book> collectionbookAppogio = new List<book>();
   collectionbookAppogio     = collectionbook.FindAll(bf => bf.autor == "Rossi");
```

DICTIONARY

Dictionary <int, structlibri> lista =new Dictionary <int, structlibri >
int rappresenta in pratica il codice di accesso al libro

Ha le stesse proprietà e metodi di List, ma definisce una collezione di coppie chiave – valore. Queste strutture si chiamano **tabelle di Hash** - *Vedere metodi e proprietà dalle dispense* -
Le tabelle hash sono strutture dati che conservano delle coppie chiave-valore, organizzate in base al codice hash della chiave. Il concetto è analogo a quello di un dizionario o ad una rubrica, in cui le voci sono organizzati in ordine alfabetico, ed a ognuna di esse corrisponde ad esempio una descrizione o un numero di telefono. Per la loro struttura le Hashtable sono utilizzate quando è necessario memorizzare e ricercare dei dati in maniera veloce ed efficiente, infatti quando si deve memorizzare una coppia chiave-valore, viene ricercata la locazione di memorizzazione valutando il codice hash della chiave. Naturalmente più oggetti possono avere uno stesso codice hash, dunque saranno memorizzati in una stessa locazione. In fase di ricerca viene ancora valutato il codice hash, e ciò restringe il campo di ricerca dell'oggetto agli elementi memorizzati in una particolare locazione, dei quali viene letto e confrontato il valore della chiave con il valore ricercato.

Esempio:

```
public struct libro
{
    public string titolo;
    public string autore;
}
public int i=0;
Dictionary<int, libro> dizionarioLibri = new Dictionary<int, libro>();

private void btnDizionario_Click(object sender, EventArgs e)
{
    libro l;
    l.titolo = textBox2.Text;
    l.autore = textBox3.Text;

    dizionarioLibri.Add(i,l);
    i++;


}

private void btnVisDizionario_Click(object sender, EventArgs e)
{
    foreach (int key in dizionarioLibri.Keys)
        MessageBox.Show(Convert.ToString(key));

    foreach (libro l in dizionarioLibri.Values)
        MessageBox.Show(l.titolo+" "+l.autore);

   }
```

Per trovare un oggetto in base alla chiave è necessario inserire il valore della chiave tra le parentesi [] come nell'esempio

```
libro lf = dizionarioLibri[0];
MessageBox.Show(lf.autore);
```

CODE e PILE

Sono strutture che rispettano le politiche
FIFO (First In First Out) le CODE
e
LIFO (Last In First Out) le PILE

CODE

```
Queue<string> coda = new Queue<string>();
```

`.Enqueue(`*elemento*`);` Accoda un nuovo elemento nella coda, il parametro è un OBJ ma accetta anche string e ovviamente struct

`.Dequeue();` Restituisce il primo elemento della coda eliminandolo dalla coda stessa

`.Peek();` Consente di vedere qual è il prossimo elemento della coda senza eliminarlo dalla coda

ESEMPIO

```
public struct libro
    {
        public string titolo;
        public string autore;
    }
    public int i = 0;
    Queue<libro> codaLibri = new Queue<libro>();
    private void btnCoda_Click(object sender, EventArgs e)
    {
        libro l;
        l.titolo = textBox2.Text;
        l.autore = textBox3.Text;

        codaLibri.Enqueue(l);
       }
```

PILE

```
Stack<string> pila = new Stack<string>();
```

`.Push(`*elemento*`);` Accoda un nuovo elemento nella coda, il parametro è un OBJ ma accetta anche string e ovviamente struct

`.Pop();` Restituisce il primo elemento della coda eliminandolo dalla coda stessa

`.Peek();` Consente di vedere qual è il prossimo elemento della coda senza eliminarlo dalla coda

ESERCIZI

- COSTRUIRE UN DIZIONARIO che memorizzi i dati degli articoli di un negozio (chiave e nome articolo). Successivamente trovare il nome dell'articolo in base ad una chiave chiesta in input.
  Per la realizzazione utilizzare due TextBox per il nome e Key. Un tasto inserisci, un tasto visualizza il numero di inserimenti in una label e un tasto ricerca che preleva la chiave e visualizza il nome dell'articolo in una label

- UTILIZZANDO UNA CODA realizzare la gestione di un pronto soccorso:
  al Pronto Soccorso i pazienti sono classificati con un colore: rosso (urgentissimo), giallo (grave), verde (moderato), bianco (lieve). Funziona così: quando una persona arriva al pronto soccorso l'infermiere lo registra (nome e età) e decide anche il colore di priorità; appena un medico è libero deve gestire prima quelli con codice rosso, poi giallo, poi verde e poi bianco. Dichiara una struttura Paziente con i campi Nome, Età, Colore; il programma deve gestire 4 code diverse e inserire il paziente nella lista corretta; quando un medico è libero preme il pulsante e il programma fornisce subito il paziente da servire (quindi sceglie il primo dei pazienti codice verde solo se non ci sono codici rosso e giallo). Le informazioni vanno visualizzate in una etichetta; i dati prelevati da caselle di testo Un altro pulsante determina i valori massimo e minimo giornalieri delle temperature rilevate.

- UTILIZZANDO UNA PILA realizzare la gestione dei Container alla banchina di un porto:
  i Container sono dei grandi contenitori di merci. Una gru solleva i Container e li mette in pila uno sull'altro. Poi dalla banchina li carica nelle stive delle navi. Dichiara una struttura Container coi campi Codice, Peso e Tara; poi prepara un programma che carica in una pila i dati dei Container prelevati da caselle di testo; un altro pulsante invece toglie dalla pila i Container e mostra a video i dati