struct in c#

Lezione

Table of contents

- Struct
- List<T>

struct

Definizione

- La struttura è uno dei tipi di dato definiti dall'utente ADT (che si affiancano ai tipi già esistenti, come int, bool, Random ...)
- La sintassi generale della dichiarazione della struct è la seguente

```
public struct NuovoTipo {
  . . . //corpo della struttura
}
```

- Dopo aver dichiarato la struttura è possibile dichiarare delle variabili

```
NuovoTipo miaVariabile;
```

Definizione

- Lo scopo della struttura è quindi di creare un tipo di dato costituito da numerosi elementi di diverso tipo; per esempio è possibile dichiarare un tipo di dato Cane costituito da campi come il nome, la razza, l'età e il sesso; il Cane quindi sarebbe dichiarato simile al seguente esempio:

```
public struct Cane {
   public string nome;
   public string razza;
   public int età;
   public bool maschio;
}
```

Accesso ai campi

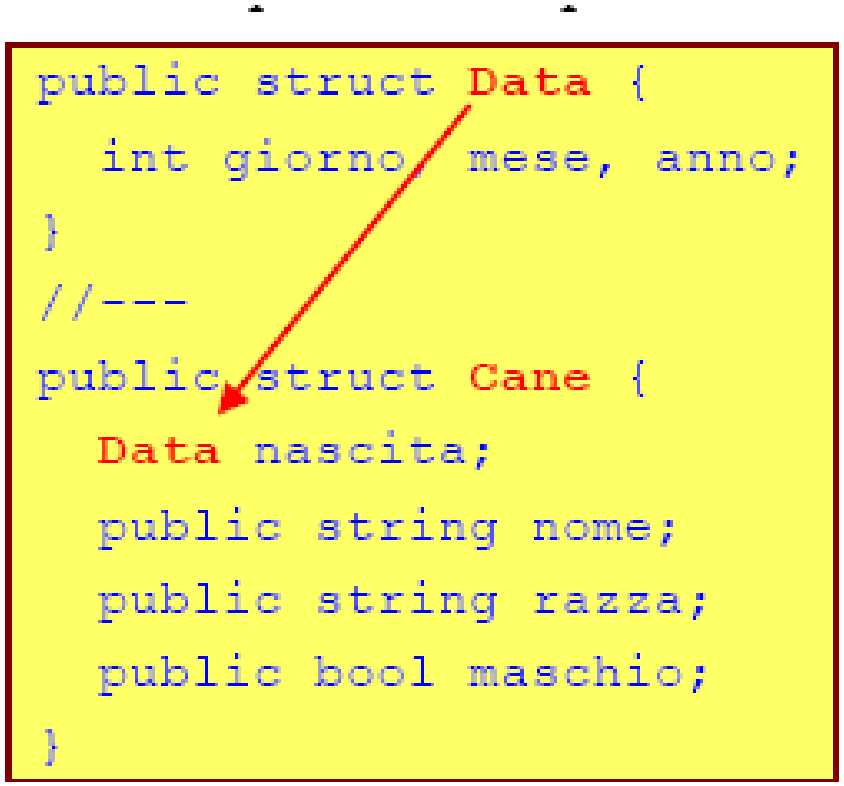
- L'accesso ai campi avviene con la notazione puntata; si scrive cioè il nome della variabile seguito da un punto, poi seguito dal nome del campo. Per esempio si può accedere al campo nome di un cane come segue:

```
Cane cuccioloNato;
```

```
cuccioloNato.nome = "Pluto";
```

Strutture nidificate

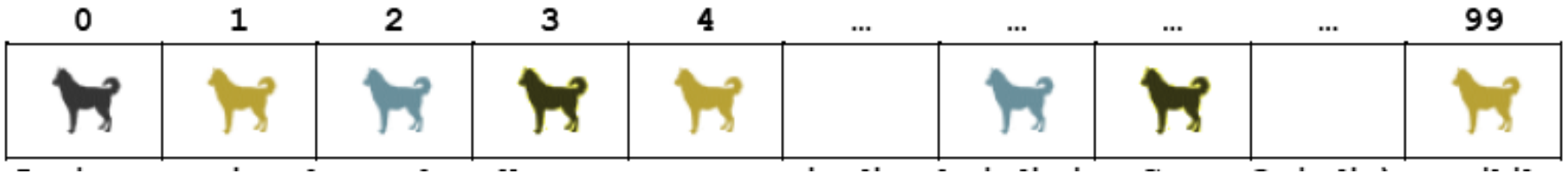
- Se si definisce una struttura è possibile usarla come un qualsiasi tipo di dato; in particolare è possibile usarla come tipo di dato di un campo di un'altra struttura.



Collezioni di strutture - array

- Poiché una struttura si può usare come un qualsiasi tipo di dato, è possibile dichiarare un vettore di strutture. Per esempio sarebbe possibile dichiarare:

```
public struct Cane {
   Data nascita;
   public string nome;
   public string razza;
   public bool maschio;
}
//---
Cane[] canile = new Cane[100];
```



List<t>

Collezioni tipizzate

Collezioni di strutture - List

- Oltre ad array è possibile utilizzare delle collezioni tipizzate, offerte direttamente dal framework .NET, ad esempio List<T> (using System.Collections.Generic;)

```
List<Cane> listC = new List<Cane>();
```

- O collezioni non tipizzate,dette generiche ad esempio ArrayList (using System.Collections;)

- Definizione: Una collezione è definita come un contenitore che memorizza e gestisce un gruppo di oggetti, chiamati elementi

List<T>

- La lista tipizzata richiede, per il suo uso di esplicitare il tipo usato.
- La lista tipizzata può essere definita per qualsiasi tipo, ma una volta scelto tutti gli elementi devono essere dello stesso tipo.
- La lista tipizzata ammette l'accesso indicizzato agli elementi, come per i vettori.
- Dopo aver dichiarato la lista è necessario invocarne il costruttore

```
List<string> numeri = List< string >();
```

List<T> proprietà

- **Proprietà Count** : restituisce il numero degli elementi presenti nella lista.

```
int quanti = mialista.Count;
```

```
for (int indice = 0; indice < mialista.Count; indice++)
     //istruzioni sulla lista
```

List<T> Metodi

Metodo	Prototipo
Add - Aggiunge un elemento alla lista	**void Add(Tipo valore)**
Clear - Rimuove tutti gli elementi dalla lista	**void Clear()**
Contains - Ritorna true se il valore esiste nella lista, false altrimenti	**bool Contains (Tipo valore)**
IndexOf - Ritorna l'indice della prima occorrenza del valore specificato; se il valore non esiste ritorna -1	**int IndexOf (Tipo valore)**
Insert - Inserisce il valore nella posizione indice.	**void Insert (int indice, Tipo valore)**
Remove - Elimina il valore dalla lista, se presente. Altrimenti nulla.	**void Remove (Tipo valore)** -
RemoveAt - Elimina l'elemento che si trova nella posizione indice.	**void RemoveAt (int indice)**

Uso del foreach

- Per le liste tipizzate è possibile usare l'istruzione foreach nel seguente modo:

```
foreach (Tipo variabile in mialista)
        Istruzione;
```

```
foreach (int k in numeri)
      if ( k = 0)
            Text += " § " + Convert.ToString(k);
```

questions





Esercizio Agenzia Viaggi 1

- Creare la struct Viaggio
  - Destinazione
  - Durata in gg
  - Prezzo base
- Creare la struct Agenzia
  - Nome
  - Stagione (2=alta,1=media,0=bassa)
  - Lista di Viaggi

Esercizio Agenzia Viaggi 2

- ✓ **Void CaricaDaFile(…)** carica i viaggi da file
- ✓ **Void ScriviSuFIle(…)** salva i dati su file
- ✓ **Lista<Viaggi> ViaggiBrevi(int durata,…)** ritorna i viaggi in listino di durata minore o uguale a quella passata come parametro

Esercizio Agenzia Viaggi 2

- ✓ **Double PrezzoViaggio(string destinazione,…)** Calcola e restituisce il prezzo del viaggio la cui destinazione è passata come parametro. Se la stagione è alta viene applicato il supplemento che corrisponde ad un aumento del 30% del prezzo base per viaggi che durano fino a 7 giorni, e del 25% per viaggi che durano oltre 7 giorni. Se la destinazione richiesta non è in listino, stampare un messaggio appropriato e ritornare 0. NB: si ipotizza sia presente un solo viaggio per destinazione
- ✓ **Viaggio LowCost(…)** Ritorna il viaggio corrispondente al viaggio più economico tra tutti quelli in listino.

Esercizio Agenzia Viaggi 3

- Nel consumer:
- creare un oggetto di tipo Agenzia
- popolarlo con il file listino.txt
- chiamare il metodo ViaggiBrevi con durata 4
- chiamare il metodo PrezzoViaggio con parametri opportuni
- chiamare il metodo LowCost