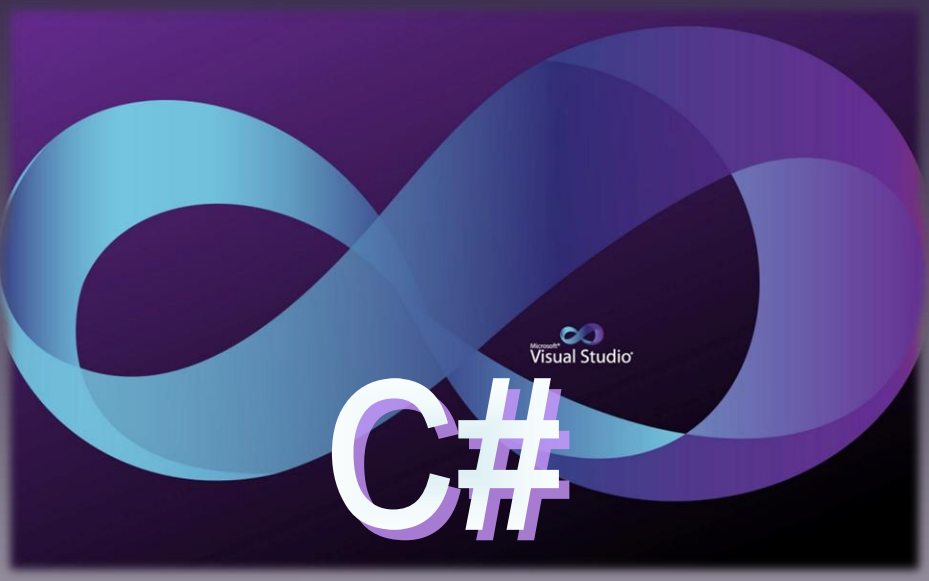
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

INTRODUZIONE ALLE STRUTTURE

DISPENSA 06.01

06-01_Strutture_[ver_15]

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

Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

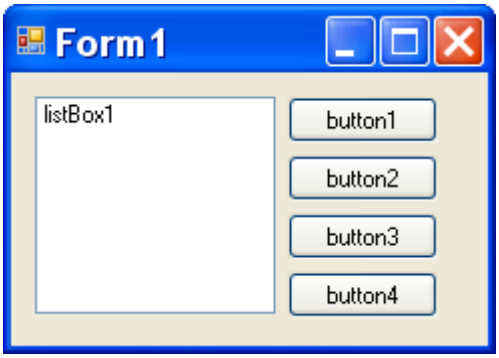
INTRODUZIONE ALLE STRUTTURE

STRUTTURE

COME USARE LE STRUTTURE

PROGETTO GUIDATO

- Si prepari un Form1 simile al seguente:



- Preparare la dichiarazione globale seguente

```
//dichiarazione di un nuovo tipo (struttura)
public struct Dog
{     //campi della struttura
      public string name;
      public string breed;
      public int age;
      string tryTo;
}
//dichiarazione di variabile
Dog d , h;
```

- Doppio clic su **button1** e associare il seguente codice:

```
//istanziazione di variabile
d = new Dog();
//utilizzo di variabile
d.name = "Rex" ;
d.breed = "Wolfhound" ;
d.age = 5 ;
```

- Doppio clic su **button2** e associare il seguente codice:

```
//visualizzazione di variabile
listBox1.Items.Clear();
listBox1.Items.Add(d.name + ", " + d.breed + ", " + d.age);
```

- Doppio clic su **button3** e associare il seguente codice:

```
//istanziazione di variabile
h = d;
//visualizzazione di variabile
listBox1.Items.Clear();
listBox1.Items.Add(d.name + ", " + d.breed + ", " + d.age);
```

- Doppio clic su **button4** e associare il seguente codice:

```
//istanziazione di variabile
h = d;
d.age = 7;
d.name = "Lassie";
//visualizzazione di variabile
listBox1.Items.Clear();
listBox1.Items.Add(h.name + ", " + h.breed + ", " + h.age);
listBox1.Items.Add(d.name + ", " + d.breed + ", " + d.age);
```

- Provare il progetto

SCOPO DI UNA STRUTTURA

Una struttura è un tipo di dato costruito dal programmatore.

Una struttura serve per aggregare insieme dati di tipo diverso tra loro per formare un nuovo tipo di dato che tiene insieme questi dati.

Supponiamo per esempio di voler definire un nuovo tipo di dato Autovettura che possa conservare dati quali Targa, Marca e Modello, Cilindrata, Colore, Posti e Esente dal bollo. Le informazioni sono diverse tra loro; per esempio Targa, Marca e Modello saranno di tipo stringa; Cilindrata e Posti possono essere numeri interi; Colore potrebbe essere stringa (ma anche di tipo Color); infine Esente dal bollo potrebbe essere un bool (vero o falso).

Una singola autovettura sarà una variabile di tipo Autovettura e conterrà tutte queste informazioni insieme, come un pacchetto. Per esempio:

Targa	**XY 987 AB**
Marca	**Feat**
Modello	**Pinto**
Cilindrata	**1200**
Colore	**Carminio**
Posti	**6**
Esente dal bollo	**false**

Poiché questo tipo è formato da tanti dati aggregati si dice anche dato aggregato o strutturato. Anche gli array sono dati aggregati (o strutturati).

SINTASSI BASE IN C#

Per costruire una struttura in Visual C# conviene analizzare due passi distinti: il primo passo definisce un nuovo tipo di dato, l'autovettura in generale. Il secondo passo è quello di dichiarare una (o molte) variabile di tipo Autovettura.

Primo Passo: costruire (dichiarare) un nuovo tipo di dato.

Nel seguente esempio è mostrata una semplice struttura che descrive una Autovettura:

```
public struct Autovettura
{
      public string Targa;
      public string Marca;
      public string Modello;
      public int Cilindrata;
      public string Colore;
      public int Posti;
      public bool Esente;
}
```

Come si nota ci sono una grande quantità di parole chiave **`public`** che servono per indicare che sono accessibili da qualsiasi punto del programma.

Poi c'è la parola chiave **`struct`** che indica che voglio costruire una struttura, seguita dal nome che avrà il mio nuovo tipo di dato.

Tra parentesi graffe si elencano i campi, ovvero le caratteristiche della mia struttura. Anche questa è preferibile siano public. Per ogni campo è necessario indicare il tipo.

Secondo Passo: dichiarare variabili del nuovo tipo.

Nel seguente esempio è mostrato come dichiarare delle variabili di tipo Autovettura:

```
Autovettura miaAuto;
Autovettura autoMamma;
Autovettura rottame = new Autovettura();
miaAuto = new Autovettura();
```

le variabili miaAuto e autoMamma sono solo dichiarate; la variabile rottame è anche istanziata (ovvero si alloca memoria per essa). È possibile anche allocare memoria (istanziare la variabile anche separatamente dalla dichiarazione, come nel caso si miaAuto. Anche in altri punti del programma così:

```
autoMamma = new Autovettura();
```

OSSERVAZIONI SULLA DICHIARAZIONE

La struttura è uno dei tipi di dato definiti dall'utente. L'utente dichiara un nuovo tipo di dato (che si affianca ai tipi già esistenti, come int, bool, Random, Button, ecc.) con un nuovo nome.

Dopo aver dichiarato il nuovo tipo è possibile utilizzarlo per dichiarare locazioni, come variabili e parametri.

La sintassi generale della dichiarazione del tipo è simile alla seguente:

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

Dopo aver dichiarato la struttura è possibile dichiarare delle variabili come nel seguente esempio:

```
NuovoTipo miaVariabile;
```

La sintassi della struttura prevede la possibilità di dichiarare dei campi interni alla struttura; un campo è una locazione racchiusa dentro una struttura; la dichiarazione diventa simile al seguente esempio:

```
public struct NuovoTipo {
public bool campoLogico;
public double campoReale;
public int campoIntero;
public string campoFrase;
. . .
}
```

Lo scopo della struttura è quindi di creare un tipo di dato costituito da numerosi «pezzi» 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;
}
```

È poi possibile dichiarare variabili di tipo Cane come nel seguente esempio:

```
Cane cercaTartufi;
Cane accompagnaCieco;
Cane cuccioloNato;
```

ACCESSO AI CAMPI

Prima di proseguire è indispensabile osservare che in generale il tipo di dato non è una locazione. Il tipo può essere pensato come un modello, un progetto (analogo al progetto di una casa) che pur descrivendo nei dettagli l'elemento NON è un elemento. Quindi il progetto di una casa NON è una casa, anche se descrive la casa con precisione.

Sempre restando nell'analogia, il progetto serve per poter costruire l'elemento; quindi il progetto della casa serve per costruire una casa e il tipo di dato Cane serve per costruire cani.

Quando si affronta la questione dell'accesso ai campi, lo studente deve porre attenzione che (n generale) non è possibile accedere ai campi del tipo, ma si accede ai campi delle locazioni di quel tipo. Nel nostro esempio, quindi, non si accede ai campi del Cane inteso come tipo, ma si accede ai campi di una variabile di tipo Cane.

L'accesso ai campi avviene (in modo simile a quello delle Proprietà dei componenti) 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:

```
cuccioloNato.nome = "Pluto";
int numero = accompagnaCieco.eta;
```

ovviamente l'accesso è in lettura e in scrittura.

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.

Per esempio sarebbe possibile dichiarare:

```
public struct Data
{
     int giorno, mese, anno;
}
//---
public struct Cane
{
     Data nascita;
     public string nome;
     public string razza;
     public bool maschio;
}
```

DIFFERENZE TRA TIPI E DATI

Consideriamo di nuovo l'esempio della struttura che descrive una Autovettura:

```
public struct Autovettura
{
      public string Targa;
      public string Marca;
      public string Modello;
      public int Cilindrata;
      public string Colore;
      public int Posti;
      public bool Esente;
}
```

Il tipo Autovettura non è un contenitore di dati ma solo un modello per costruire autovetture. È come fare il progetto di una casa: poiché è solo disegnato su un foglio non è possibile abitarvi o metterci gli elettrodomestici; tuttavia descrive bene come la casa sia fatta (dove sono muri, porte, finestre, ecc..).

Il vero contenitore di dati è la variabile di tipo Autovettura. Nella nostra metafora corrisponde alla casa costruita secondo quel progetto. Essa ha muri, porte e finestre esattamente come previste dal progetto, ma è realmente una casa dove metterci mobili, elettrodomestici e persone.

La scrittura

```
. . .  = new Autovettura();
```

indica che voglio proprio costruire una variabile nuova (**new**) secondo il progetto previsto; poiché c'è un piccolo programma che fa questo è necessario indicare anche le parentesi tonde.

ACCESSO AI CAMPI

Abbiamo detto che una struttura ha dei campi, ovvero le informazioni contenute in essa. Per esempio una Autovettura ha un campo Targa. In questi campi normalmente ci si può scrivere e leggere (come accadeva per le celle di un array).

SCRITTURA NEI CAMPI

Nel seguente esempio è mostrato come accedere in scrittura ai campi delle variabili di tipo Autovettura (si noti che la variabile deve essere stata precedentemente dichiarata e istanziata):

```
miaAuto.Targa = "XX987AB";
miaAuto.Marca = "Feat";
miaAuto.Modello = "Brutta";
miaAuto.Cilindrata = 1200;
miaAuto.Colore = "Carminio";
miaAuto.Posti = 6;
miaAuto.Esente = false;
```

come si nota i valori assegnati devono essere compatibili coi tipi dei campi.

Si osserva anche che (ma non è una novità, vero?) occorre prima indicare il nome della variabile e dopo il nome del campo.

LETTURA DEI CAMPI

Nel seguente esempio è mostrato come accedere in lettura ai campi delle variabili di tipo Autovettura (si noti che la variabile deve essere stata precedentemente dichiarata e istanziata):

```
int cil = miaAuto.Cilindrata;
string marc = miaAuto. Marca;
autoMamma.Marca = miaAuto.Marca;
autoMamma.Modello = miaAuto.Marca;
rottame.Esente = ! miaAuto.Esente;
miaAuto.Cilindrata += 100;
miaAuto.Posti++;
```

anche in lettura occorre indicare prima il nome della variabile e dopo il nome del campo.

È possibile copiare dati da un campo all'altro, purché siano rispettate le regole di compatibilità.

È possibile eseguire operazioni sui dati.

È possibile eseguire assegnazioni speciali (con operazione) che leggono e scrivono insieme; per esempio `miaAuto.Cilindrata += 100;` che porta la cilindrata a 1300.

ERRORI COMUNI

Abbiamo detto che il tipo non è un contenitore di dati ma solo un modello per costruire variabili. Per questo no è possibile assegnare o usare valori di un tipo.

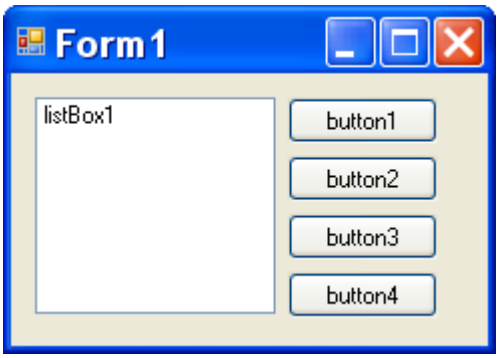
Nel seguente esempio sono mostrati alcuni errori:

```
Autovettura = miaAuto;       //errore Autovettura NON è una variabile
miaAuto = Autovettura; //idem
Autovettura.Marca = "Feat";  //errore Autovettura NON ha dati
int x = Autovettura.Posti;   //errore Autovettura NON ha dati
```

COSTRUTTORI DI STRUTTURE

PROGETTO GUIDATO

- Si prepari un Form1 simile al seguente:



- Preparare la dichiarazione globale seguente

```
public struct Dog
{     //campi della struttura
      public string name;
      public string breed;
      public int age;
      string tryTo;
      public Dog(string n, string b, int a, string tt)
      {
            name = n;
            breed = b;
            age = a;
            tryTo = tt;
      }
}
Dog d;
Dog myPet = new Dog("Jolly", "Cocker", 1, "New Ark");
```

- Doppio clic su **button1** e associare il seguente codice:

```
//istanziazione di variabile
d = new Dog("Rex", "Wolfhound", 5, "London");
```

- Doppio clic su **button2** e associare il seguente codice:

```
//visualizzazione di variabile
MessageBox.Show (myPet.name + "\n" + myPet.breed + "\n " + myPet.age);
MessageBox.Show (d.name + "\n" + d.breed + "\n " + d.age);
```

- Doppio clic su **button3** e associare il seguente codice:

```
//istanziazione di variabile locale
Dog h = new Dog("Lassie", "Collie", 7, "Cork");
//visualizzazione di variabile locale
MessageBox.Show (h.name + "\n" + h.breed + "\n " + h.age);
```

- Doppio clic su **button4** e associare il seguente codice:

```
//istanziazione di variabile locale
Dog h ;
//visualizzazione di variabile locale
MessageBox.Show (h.name + "\n" + h.breed + "\n " + h.age);
```

- Provare il progetto, correggere l'errore e provare il progetto di nuovo …

COSTRUTTORI DI STRUTTURE

Un costruttore di strutture è un particolare metodo che costruisce (istanzia e inizializza) una variabile di tipo struttura.

Prima di usare una struttura il linguaggio richiede l'uso di un costruttore.

Per esempio l'invocazione seguente:

```
//istanziazione di variabile locale
Dog h = new Dog("Lassie", "Collie", 7, "Cork");
```

esprime il fatto che la variabile h (appena dichiarata) venga allocata in memoria e sia predisposta con i valori iniziali indicati come parametri.

Il metodo è definito internamente alla struttura e deve essere pubblico per poterlo usare dall'esterno. Ad esempio:

```
public Dog(string n, string b, int a, string tt)
{
        name = n;
        breed = b;
        age = a;
        tryTo = tt;
}
```

deve essere scritto dentro la struttura Dog, senza tipo restituito (nemmeno **void**) e prevede abbastanza parametri per definire l'intera struttura.

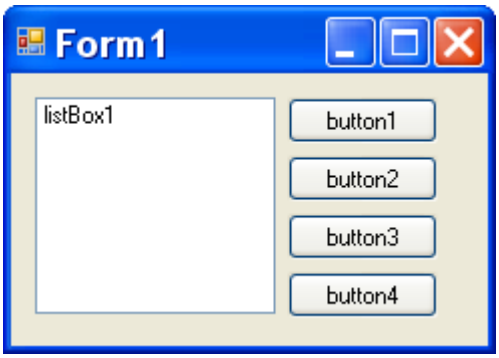
Ovviamente è possibile usare meno parametri oppure nessuno, se si riesce a inizializzare opportunamente i campi. Ad esempio:

```
public Dog(string n, string b)
{
        name = n;
        breed = b;
        age = 1;
        tryTo = "Rome";
}
```

METODI DI STRUTTURE

PROGETTO GUIDATO

- Si prepari un Form1 simile al seguente:



- Preparare la dichiarazione globale seguente

```
public struct Dog
{     //campi della struttura
      public string name;
      public string breed;
      public int age;
      string tryTo;
```

```
      public string GetDog()
      {
            return (name + "\n" + breed + "\n " + age);
      }
```

```
      public void SetDog (string n, string b, int a, string tt)
      {
            d.name = n ;
            d.breed = b;
            d.age = a;
            d.tryTo = tt;
      }
}
//dichiarazione di variabile
Dog d;
```

- Doppio clic su **button1** e associare il seguente codice:

```
//istanziazione di variabile
d = new Dog();
//utilizzo di metodo
d.SetDog ("Rex" , "Wolfhound" , 5 , "Lei" );
//utilizzo di metodo
string s = d.GetDog ();
//visualizzazione di variabile
MessageBox.Show ( s );
```

- Provare il progetto

METODI DI STRUTTURE

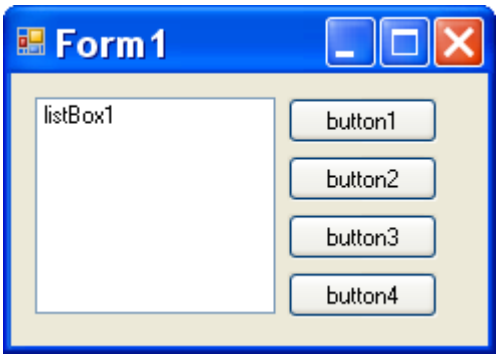
Un metodo di strutture è un particolare metodo interno alla struttura che permette di comunicare con la struttura richiedendo informazioni, imponendo modifiche o alterazioni ai dati interni della struttura.

Per poter invocare i metodi dall'esterno occorre specificare il nome della variabile, seguito dal punto e seguito dal nome del metodo; l'invocazione richiede obbligatoriamente le parentesi tonde.

VETTORI DI STRUTTURE

PROGETTO GUIDATO

- Si prepari un Form1 simile al seguente:



- Preparare la dichiarazione globale seguente

```
//dichiarazione di un nuovo tipo (struttura)
public struct Dog
{     //campi della struttura
      public string name;
      public string breed;
      public int age;
      string tryTo;
}
//dichiarazione di variabile
Dog d , h;
//dichiarazione di vettore
Dog [] canile;
```

- Doppio clic su **button1** e associare il seguente codice:

```
//istanziazione del vettore
canile = new Dog[15];
//istanziazione delle celle del vettore
for (int i=0; i<15; i++)
{
  canile[i] = new Dog();
}
```

- Doppio clic su **button2** e associare il seguente codice:

```
//visualizzazione di variabile
listBox1.Items.Clear();
for (int i=0; i<15; i++)
{
  Dog tmp = canile[i];
  listBox1.Items.Add(tmp.name + ", " + tmp.breed + ", " + tmp.age);
}
```

- Doppio clic su **button3** e associare il seguente codice:

```
//inizializzazione delle celle del vettore
for (int i=0; i<15; i++)
{
  Dog tmp;
  tmp.name = "Rex_" + i;
  tmp.breed = "Wolfhound_" + i/3;
  tmp.age = (i+4)/3;
  canile[i] = tmp;
}
```

- Provare il progetto

VETTORI DI STRUTTURE

Poiché, come appena visto, 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 {
      public string nome;
      public string razza;
      public bool maschio;
}
//---
Cane[] canile = new Cane[100];
```

nell'esempio appena proposto, la variabile canile è formata da 100 celle ciascuna delle quali può contenere un singolo cane:

Le icone canine dentro le celle sono un esempio di valori di tipo Cane. Quindi è possibile eseguire assegnazioni come le seguenti:

```
canile[0] = cuccioloNato;
accompagnaCieco = canile[99];
```

Poiché ogni cella è una locazione di tipo cane è possibile accedere ai campi con la solita notazione puntata. Per esempio è possibile fare:

```
canile[0].nome = "Rex";
canile[17].nome = canile[13].nome;
string attestato = canile[99].nome;
```

accedendo prima alla cella e dopo al campo.

STRUTTURE CON CAMPI DI TIPO VETTORE

Come appena visto, una struttura ammette campi di qualsiasi tipo di dato, per cui è possibile dichiarare campi di tipo vettore. Per esempio sarebbe possibile dichiarare:

```
public struct Studente {
      public int[] voti;
      public string nome;
      public bool maschio;
}
//---
Studente rappresentante;
```

Nel precedente esempio, la variabile rappresentante è uno Studente il cui campo voti è un vettore di interi.

Quindi è possibile accedere ai campi come nel seguente esempio:

```
rappresentante.nome = "Dante";
rappresentante.maschio = true;
rappresentante.voti = new int[12];
rappresentante.voti[0] = 8;
```

nel precedente esempio si assegna un nome e un sesso al rappresentante, poi si alloca memoria per 12 voti (indicizzati da zero a 11) e infine nella materia di indice zero viene assegnato il valore 8.

Sarebbe possibile anche eseguire algoritmi come il seguente:

```
for (int i=0; i< rappresentante.voti.Lenght; i++)
rappresentante.voti[i] = 6;
```

per assegnare il 6 a tutti i voti del rappresentante.

ANNIDAMENTI DI VARIO LIVELLO

Per concludere questa brevissima trattazione sulle strutture vediamo il caso di un vettore di

strutture con campi di tipo vettore. Se si dichiara:

```
public struct Studente {
public int[] voti;
public string nome;
public bool maschio;
}
//---
Studente rappresentante;
Studente [] classe3A = new Studente [27];
```

Nel precedente esempio, la variabile classe è un vettore di celle con uno Studente il cui campo voti è un vettore di interi.

Quindi è possibile eseguire algoritmi come il seguente:

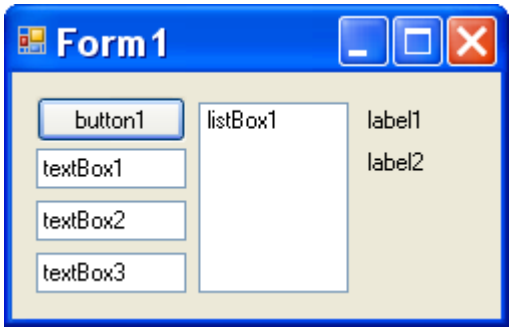
```
for (int k=0; k< classe3A.Lenght; k++)
for (int h=0; h< rappresentante.voti.Lenght; h++)
classe3A[k].voti[h] = 6;
```

per assegnare il 6 a tutti i voti di tutti gli studenti della classe3A.

ESERCIZI

Prima leggi gli esercizi elencati di seguito. Prova a svolgerli da solo uno alla volta. Dopo verifica la tua soluzione con quella mostrata alla fine del documento.

Per risolvere gli esercizi è possibile creare una applicazione simile alla seguente figura e associare gli esercizi al pulsante



ESERCIZIO 1. ANIMALE

Dichiarare una struttura Animale con Nome, Specie, Razza, Sesso, Età e Peso.

Programmare il pulsante in modo che crei un gatto soriano maschio, il cui nome, età e peso siano caricati dalle textBox.

ESERCIZIO 2. COMPACT DISC

Dichiarare una struttura CD con Titolo, Durata, Anno, Artista.

Programmare il pulsante in modo che crei un disco di quest'anno con i dati caricati dalle textBox.

Modificare la struttura con un costruttore e modificare il pulsante in modo da utilizzarlo per caricare i campi richiesti. Modificare infine la struttura rendendo i campi non public.

ESERCIZIO 3. FILM

Dichiarare una struttura Film con Titolo, Durata, Anno, Regista.

Programmare il pulsante in modo che crei un film di quest'anno con i dati caricati dalle textBox.

Modificare la struttura rendendo i campi non public.

Modificare la struttura con un costruttore e modificare il pulsante in modo da utilizzarlo per caricare i campi richiesti.

Modificare la struttura con un metodo che renda una stringa con i dati della struttura e modificare il pulsante in modo da utilizzarlo per scrivere il film nella listbox.

ESERCIZIO 4. CARTA DA GIOCO

Dichiarare una struttura Carta con campi Valore e Seme, non public.

Modificare la struttura con un costruttore che prenda valore e seme casualmente; i valori devono essere compresi tra 1 e 7, mentre il seme deve essere uno tra ♥ ♦ ♣ ♠ .

Modificare la struttura con un metodo che attribuisca valore e seme da parametro.

Modificare la struttura con un metodo che attribuisca valore e seme casualmente; i valori devono essere compresi tra 1 e 7, mentre il seme deve essere uno tra ♥ ♦ ♣ ♠ .

Modificare la struttura con un metodo che renda una stringa con i dati concatenati.

Programmare il pulsante in modo che crei una carta casuale e che poi visualizzi la carta nella listbox.

ESERCIZIO 5. TRENO

Dichiara una struttura Vagone con i seguenti campi: classe, posti, fumatore, peso;

Dichiara un vettore Treno formato da celle di tipo Vagone;

Predisponi un pulsante per allocare memoria per un treno formato da X celle, dove X è un numero scelto con un ComboBox;

Al pulsante aggiungi la assegnazione di valori casuali ai campi; il campo classe può essere 1 o 2 o 3; il campo posti deve essere un multiplo di 10 compreso tra 40 e 70; il campo fumatore è booleano; il campo peso è un numero intero compreso tra 5000 e 6500.

ESERCIZIO 6. PINACOTECA

Dichiara una struttura Quadro con i seguenti campi: autore, genere, valore, altezza, larghezza;

Dichiara un vettore Pinacoteca formato da celle di tipo Quadro;

Predisponi un pulsante per allocare memoria per una pinacoteca formata da X celle, dove X è un numero scelto con un ComboBox;

Al pulsante aggiungi la assegnazione di valori casuali ai campi; il campo autore può assumere uno dei seguenti valori «Giotto», «Leonardo», «Raffaello»; il campo genere deve essere uno dei seguenti valori «Madonna», «Venere», «Cenacolo»; il campo valore è un intero compreso tra 5 e 10; le dimensioni di un Quadro sono entrambe un numero intero compreso tra 50 e 2000.

ESERCIZIO 7. LABORATORIO

Dichiara una struttura Laboratorio con i seguenti campi: nome e posti.

Il campo posti è un vettore di booleani, che indica se una postazione è accesa o spenta.

Predisponi un pulsante per assegnare un nome al laboratorio (preso da una casella di testo) e accendere o spegnere casualmente 100 postazioni del laboratorio.

Al pulsante aggiungi la visualizzazione delle postazioni accese e spente con un ListBox.

ESERCIZIO 8. PIANO CARTESIANO

Dichiara una struttura Punto con i seguenti campi: x e y.

Dichiara un vettore di 100 Punti;

Predisponi un pulsante per assegnare casualmente valori ai punti del vettore;

Trova il punto più lontano dall'Origine del Piano Cartesiano e mostra le coordinate a video.

ESERCIZIO 9. VERIFICHE ORALI

Dichiara una struttura Data con i seguenti campi: giorno, mese e anno;

Dichiara una struttura Verifica con i seguenti campi: data (di tipo Data), disciplina (stringa) e voto (intero);

Dichiara una struttura Studente con il campo interrogazioni come vettore di verifiche;

Predisponi un pulsante per assegnare casualmente valori per 100 verifiche di varie discipline (scelte tra «informatica», «sistemi», «inglese», «italiano», «storia»);

Predisponi un altro pulsante che calcola la media aritmetica di tutte le verifiche di una materia scelta da una ComboBox; se non ci sono verifiche deve visualizzare zero.

SOMMARIO