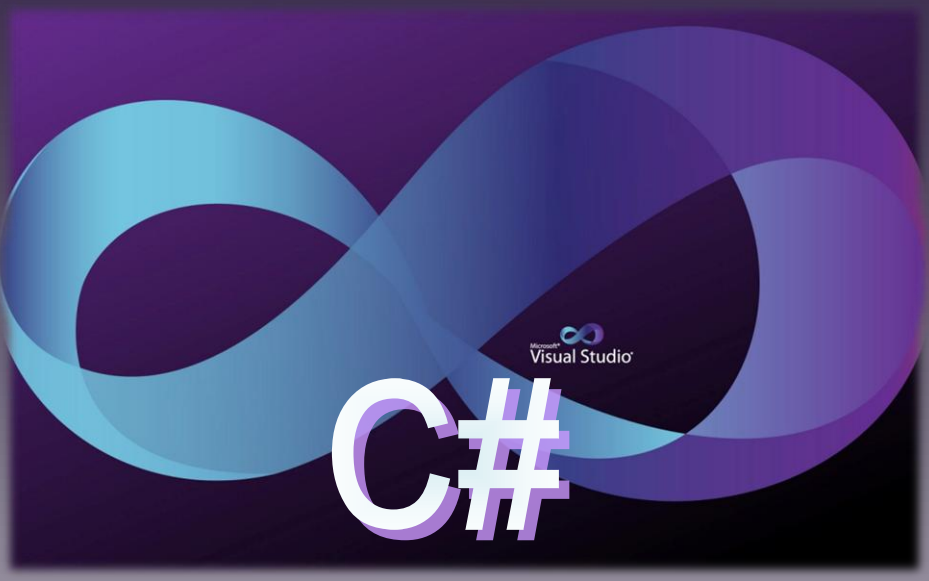
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

INTRODUZIONE AI VETTORI

DISPENSA 05.01

05-01_Vettori_[ver_15]

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

Prof. Andrea Zoccheddu
Dipartimento di Informatica

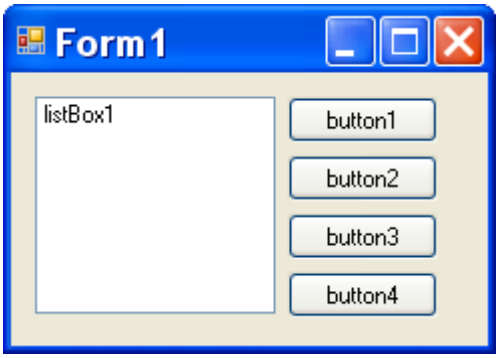
DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

INTRODUZIONE AGLI ARRAY MONODIMENSIONALI

I VETTORI

INTRODUZIONE AI VETTORI

PROGETTO GUIDATO



- Si prepari un Form1 simile al seguente:
- Preparare la variabile globale seguente

```
int[] pippo = new int[20];
```

- Se non ricordi le variabili globali chiedi ad un insegnante essa va dichiarata dopo il comando

```
public partial class Form1 : Form {
```

- Doppio clic su button1 e associare il seguente codice:

```
//pulisce la listBox1 dal contenuto
listBox1.Items.Clear();
```

- Doppio clic su button2 e associare il seguente codice:

```
//inserisce una riga nel listBox1
listBox1.Items.Add("nuova frase");
```

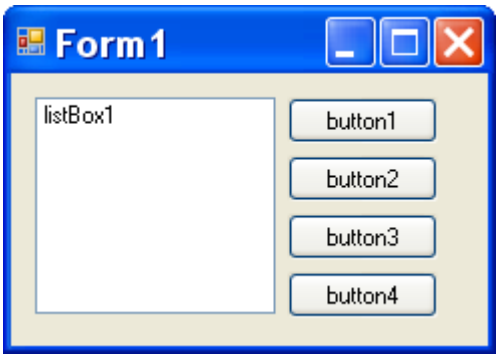
- Doppio clic su button3 e associare il seguente codice:

```
//inserisce 20 righe nel vettore pippo, ma niente cambia nel listBox1
for (int i = 0; i < 20; i++)
 pippo[i] = (i*20);
```

- Doppio clic su button4 e associare il seguente codice:

```
//inserisce i valori di pippo nel listBox1
for (int i = 0; i < 20; i++)
 listBox1.Items.Add(Convert.ToString(pippo[i]));
```

ESERCIZIO GUIDATO



- Si prepari un Form1 simile al seguente:
- Preparare la seguente **variabile globale**:

```
// dichiarazione di un array di 100 celle
int[] pippo = new int[100];
```

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

```
// inserisce i valori in pippo
Random dado = new Random();
for (int i = 0; i < 100; i++)
    pippo[i] = dado.Next(100);
// e poi . . .
```

```
// . . . li copia nel listBox1
listBox1.Items.Clear();
for (int i = 0; i < 100; i++)
    listBox1.Items.Add(Convert.ToString(pippo[i]));
```

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

```
//somma tutti i valori di pippo e poi mostra la somma
int somma = 0; //accumulatore
for (int i = 0; i < 100; i++)
  somma += pippo[i];
button2.Text = Convert.ToString(somma);
```

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

```
//trova e mostra il valore MINIMO tra tutti i valori di pippo
int minimo = pippo[0];
for (int i = 1; i < 100; i++)
   if (minimo > pippo[i]) //segno di maggiore
      minimo = pippo[i];
button3.Text = Convert.ToString(minimo);
```

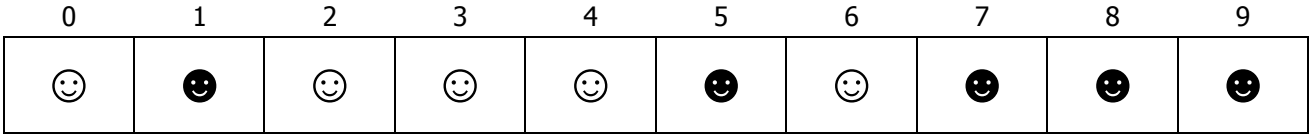
- Doppio clic su **button4** e associare il seguente codice:

```
// trova e mostra il valore MASSIMO tra tutti i valori di pippo
int max = pippo[0];
for (int i = 1; i < 100; i++)
   if (max < pippo[i]) //segno di minore
      max = pippo[i];
button4.Text = Convert.ToString(max);
```

INTRODUZIONE AGLI ARRAY

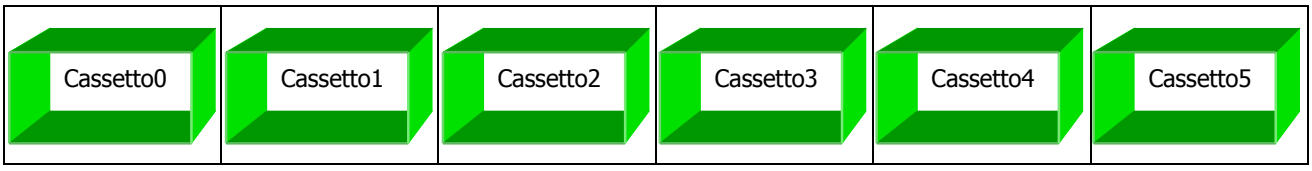
INTRODUZIONE AI VETTORI

Nei linguaggi di programmazione si è avvertita da subito l'esigenza di aggregare dati insieme e di costruire una struttura di dati da gestire talvolta come un dato unico (una collezione di dati) talvolta come dati separati. La struttura che vogliamo costruire permette di aggregare insieme molti dati dello stesso tipo (es. molti interi) e vogliamo poterli rintracciare attraverso la loro posizione, ovvero un numero. Vorremmo cioè una struttura che aggreghi insieme molti dati simili, ciascuno reperibile dalla sua posizione nella collezione. Una figura che descriva quanto detto potrebbe essere la seguente:



Le faccine dentro la struttura sono un esempio di valori contenuti delle varie posizioni.

Proviamo a tornare alla metafora dei contenitori. Se ricordi, avevamo pensato le variabili come dei contenitori che possedevano un nome (l'identificatore), che avevano una forma ben definita (il tipo di dato della variabile) e un contenuto (il valore della variabile). Immaginiamo un problema, dove sia necessario mettere insieme più elementi simili tra loro, per esempio si può pensare ad una cassettiera formata da tanti cassetti messi insieme. Il contenitore ha una forma aggregata, costituita da più contenitori saldati tra loro. Cassettiera =



Ogni cassetto della cassettiera deve avere una forma stabilita; C# impone che il primo cassetto sia sempre il numero zero; richiede che i cassetti possiedano esattamente la stessa forma (cioè siano dello stesso tipo) ma ovviamente permette ad ogni cassetto di contenere valori diversi. La cassettiera prevista dal nostro linguaggio si chiama vettore o **array**. Pur avendo compreso il concetto di array restano tuttavia alcuni interrogativi fondamentali: quanti cassetti possiede la cassettiera? Di che tipo sono i cassetti? Come posso accedere (scrivere o leggere) i contenuti di ciascun singolo cassetto? Vediamolo.

DICHIARAZIONE DI VETTORE

Un array è un tipo di dato che aggrega valori di tipo omogeneo. Gli elementi di un array sono tutti dello stesso tipo. Per dichiarare un array generico la sintassi è la seguente:

```
Tipo[] NomeVariabile ;
```

Il tipo indica la forma comune a tutte le celle; i nomi sono arbitrari. Per esempio:

```
int[] pippo;
bool[] poldo;
double[] pluto;
string[] frasi;
```

nel precedente esempio **pippo** è un array di interi (ogni sua cella contiene un intero), **poldo** è un array di booleani (ogni sua cella contiene un valore logico), **pluto** è un array di numeri decimali (ogni sua cella contiene un numero decimale), infine **frasi** è un array di stringhe (ogni sua cella contiene una stringa).

La dichiarazione può anche specificare quale debba essere la dimensione dell'array ovvero quanti elementi può contenere. La sintassi diventa allora:

```
Tipo[] NomeVariabile = new Tipo[N];
```

definisce un tipo di dato array di N elementi, dove N deve essere un intero costante. Per esempio è possibile scrivere le seguenti dichiarazioni:

```
int[] pippo = new int[10];
bool[] poldo = new bool[25];
double[] pluto = new double [100];
string[] frasi = new string[250];
```

nel precedente esempio pippo è un array di 10 interi (ci sono 10 celle e ciascuna cella contiene un intero), poldo è un array di 25 booleani (ci sono 25 celle e ciascuna cella contiene un valore logico), pluto è un array di 100 numeri decimali (ci sono 100 celle e ciascuna cella contiene un numero decimale), è un array di 250 stringhe (ci sono 250 celle e ciascuna cella contiene una stringa).

UTILIZZO DEI VETTORI

Dopo aver dichiarato un vettore è possibile usarlo. L'uso di un vettore di solito è la scrittura dentro le sue celle oppure la lettura dalle sue celle. Quando vuoi usare una cella devi fare riferimento al numero che la distingue nel vettore. Questo numero è detto indice. L'indice si scrive tra parentesi quadre. Per esempio:

```
pippo[0] = 13;   //mette 13 nella cella zero, cioè la prima del vettore pippo
poldo[1] = True; //mette True nella cella 1 di poldo, cioè la seconda cella
pluto[99] = 3.14; //mette 3.14 nella cella 99 di pluto, cioè l'ultima cella
frasi[249] = "jo"; //mette "jo" nella ultima cella del vettore frasi
```

è anche possibile usare le celle come locazioni qualsiasi, in lettura e in scrittura,

per esempio:

```
pippo[1] = pippo[0]; //copia 13 nella cella 1
poldo[1] = !poldo[1]; //inverte il valore presente nella cella 1 di poldo
pluto[99]++; //incrementa il valore della cella 99 di pluto
frasi[0] = "mary " + frasi[249]; //mette "mary jo" nella cella 0 del vettore
frasi
```

Un errore comune è quello di utilizzare un valore di indice che eccede i limiti possibili, per es:

```
int [] pippo = new int [40]; //il vettore va da 0 a 39
pippo [40] = 17; //Errore!! 40 è un valore inaccettabile
pippo [-1] = 17; //Errore!! -1 è un valore inaccettabile
pippo [99] = 17; //Errore!! 99 è un valore inaccettabile
```

INIZIALIZZAZIONE DI UN VETTORE

Gli elementi del vettore possono essere inizializzati con valori costanti (valutabili a tempo di compilazione) contestualmente alla dichiarazione del vettore. Esempio:

```
int[] vi = new int[4] { 11, 22, 33, 44 };
double[] vd = new double[4] { 1.1, 2.2, 3.3, 4.4 };
string[] weekDays = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };
string[] mesi =  {"gen","feb","mar","apr","mag","giu",
                                        "lug","ago","set","ott","nov","dic" };
string[] semi = {"cuori","quadri","fiori","picche" };
string[] voti = { "insufficiente","sufficiente","distinto","buono","ottimo" };
```

L'inizializzazione deve essere contestuale alla dichiarazione:

```
int uffa = new int [4];  //Dichiarazione del vettore uffa di 4 elementi
uffa = {11, 22, 33, 44}; //Errore! Andava scritto insieme alla dichiarazione!!
```

Se ci sono meno inizializzatori di elementi, si solleva un errore:

```
int [] n = new int [10] {3}; // Errore! Occorrono esattamente 10 elementi
double[] af = new double[3] { 3.14 }; // Errore! Occorrono 3 elementi
string[] frasi = new string[100] { "ciao" }; // Errore! Occorrono 100 elementi
```

Se ci sono più inizializzatori di elementi, si ottiene un errore di sintassi

```
int [] v = new int [2] {1, 2, 3}; //Errore! Il vettore ha solo due celle!!
```

Se si mette una lista di inizializzatori, si può evitare di specificare la lunghezza (viene presa la lunghezza della lista)

```
int n[] = {1, 2, 3}; //equivale a int n[3] = {1, 2, 3};
```

In **Visual C#** l'unica operazione possibile sugli array è l'accesso agli elementi.

Non si possono effettuare direttamente delle assegnazioni tra vettori.

```
int a[3] = {11, 22, 33};
int b[3];
b = a; //Errore!
```

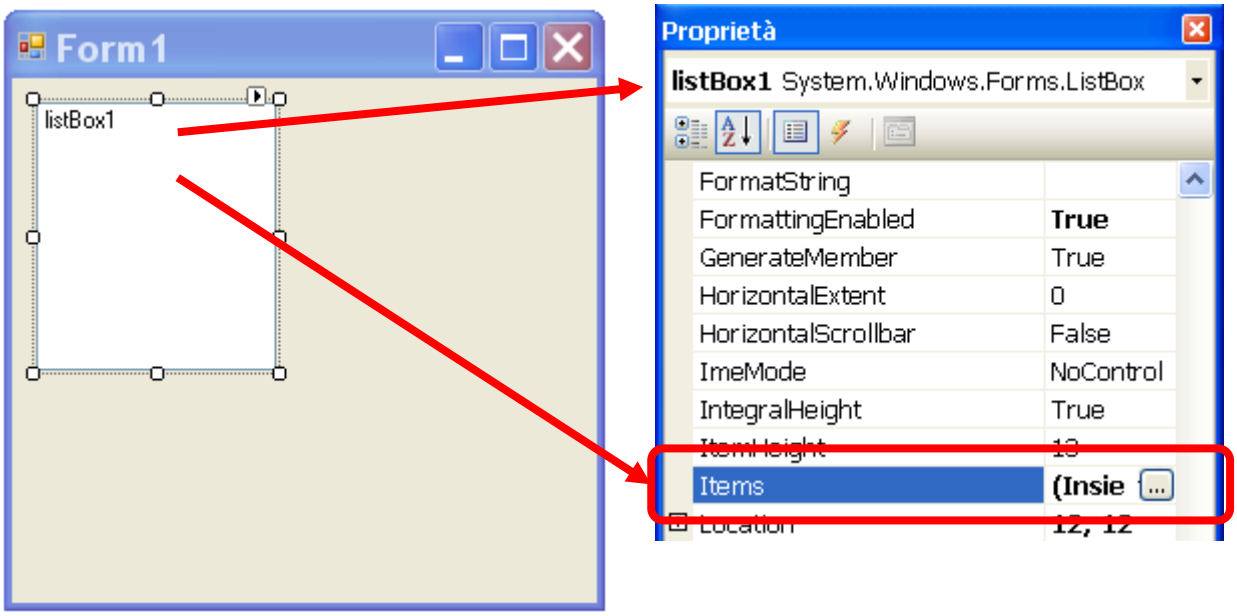
PROPRIETÀ DEI VETTORI

In Visual C#, i vettori sono oggetti e non semplici tipi di dato. Per questo gli array offrono diversi metodi e proprietà. Ne vediamo alcune rilevanti:

Length	Ottiene un valore intero (a 32 bit) che rappresenta il numero totale di elementi in tutte le dimensioni di Array (cioè quante sono le celle del vettore)
LongLength	Ottiene un valore intero a 64 bit che rappresenta il numero totale di elementi in tutte le dimensioni dell'oggetto Array (lo stesso)
Rank	Ottiene il rango (numero di dimensioni) dell'oggetto Array. Per il momento questa proprietà rende sempre 1, ma sarà utile quando si affronteranno le matrici o array multidimensionali.

LISTBOX: UN BREVE STUDIO

PROGETTARE CON LA PROPRIETÀ ITEMS



Il ListBox è un controllo per elencare tante scelte possibili. L'elenco delle scelte è contenuto nella proprietà Items che elenca generalmente stringhe. La proprietà può essere modificata a tempo di progettazione (ovvero prima di mandare in esecuzione il programma) e posso, per esempio, mostrare in un ListBox l'elenco dei nomi delle regioni dello stato italiano. Si cerca nell'elenco la proprietà Items.

La proprietà Items (elementi) del controllo serve per gestire gli elementi scritti. Nella finestra delle proprietà compare un piccolo pulsante con puntini. Fai clic sul piccolo pulsante . . . :

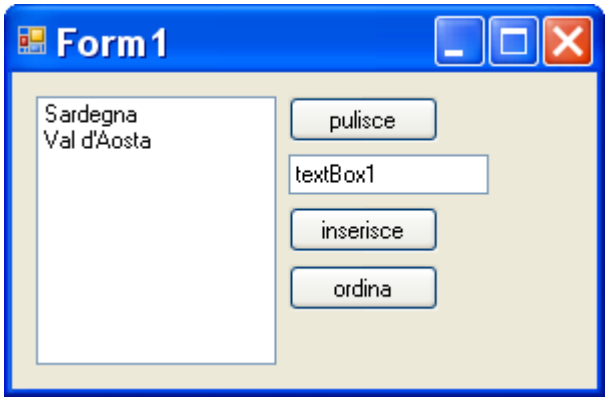
Compare una finestrella come nella figura a lato e qui puoi scrivere le regioni che ricordi. Dopo fai OK.

Gli elementi scritti nella casella a discesa non sono un vettore (array) ma qualcosa di molto simile e la proprietà Items offre possibilità analoghe ai vettori.

In seguito posso modificare a tempo di esecuzione il controllo, richiedendo l'esecuzione di algoritmi che agiscono sulle sue proprietà.

Per esempio è possibile aggiungere al Form1 un listBx1 e dei controlli che agiscono su di esso per modificarlo opportunamente. Vediamo un esempio.

ESERCIZIO GUIDATO (USARE IL LISTBOX)



- Aggiungere al Form1 i controlli come nella figura qui sotto
- Associare ai pulsanti i seguenti algoritmi:
- Doppio clic sul pulsante **pulisce** e associa il seguente codice:

```
listBox1.Items.Clear();
```

- Doppio clic sul pulsante **inserisce** e associa il seguente codice:

```
listBox1.Items.Add(textBox1.Text);
```

- Doppio clic sul pulsante **ordina** e associa il seguente codice:

```
listBox1.Sorted = ! listBox1.Sorted ;
```

ESERCIZIO GUIDATO (USARE ITEMS COME UN VETTORE)

- Si prepari un Form1 simile al seguente:
- Pulsante Genera:

```
Random dado = new Random();
listBox1.Items.Clear();
for (int a = 0; a < 100; a++)
 listBox1.Items.Add(dado.Next(10,100));
```

- Pulsante Pulisci:

```
listBox1.Items.Clear();
listBox2.Items.Clear();
```

- Pulsante Posizione:

```
int pos = listBox1.SelectedIndex;
if (pos == -1)
     MessageBox.Show("Seleziona un elemento nel primo elenco");
else
     MessageBox.Show("Hai selezionato il: " + pos);
```

- Pulsante Elemento:

```
int pos = listBox1.SelectedIndex;
if (pos == -1)
     MessageBox.Show("Seleziona un elemento nel primo elenco");
else
     MessageBox.Show("Hai selezionato: " + listBox1.Items[pos]);
```

- Pulsante Copia:

```
int pos = listBox1.SelectedIndex;
if (pos == -1)
     MessageBox.Show("Seleziona un elemento nel primo elenco");
else
{
     listBox2.Items.Add(listBox1.Items[pos]);
     MessageBox.Show("Aggiunto un elemento nel secondo elenco");
}
```

- Pulsante Minimo:

```
if (listBox1.Items.Count == 0)
     MessageBox.Show("Ma non c'è niente in elenco!");
else
{
     int minimo = Convert.ToInt16(listBox1.Items[0]);
     for (int pos = 0; pos < listBox1.Items.Count; pos++)
          if (minimo > Convert.ToInt16(listBox1.Items[pos]))
               minimo = Convert.ToInt16(listBox1.Items[pos]);
     MessageBox.Show("Il valore minimo è: " + minimo);
}
```

- Pulsante Sposta:

```
int pos = listBox1.SelectedIndex;
if (pos == -1)
     MessageBox.Show("Seleziona un elemento nel primo elenco");
else
{
     listBox2.Items.Add(listBox1.Items[pos]);
     listBox1.Items.RemoveAt(pos);
     MessageBox.Show("Spostato un elemento nel secondo elenco");
}
```

- Pulsante Modifica:

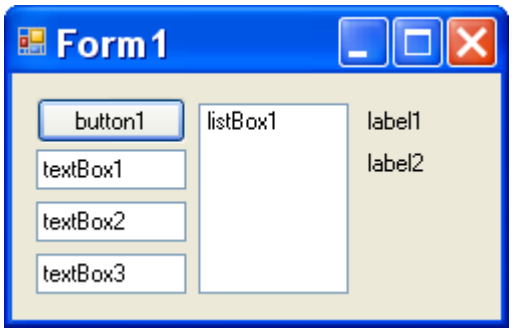
```
int pos = listBox1.SelectedIndex;
if (pos == -1)
     MessageBox.Show("Seleziona un elemento nel primo elenco");
else
{
     int tmp = Convert.ToInt16(listBox1.Items[pos]);
     tmp++;
     listBox1.Items[pos] = tmp;
}
```

ESERCIZI

ESERCIZI DI BASE SUI VETTORI

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. DICHIARAZIONI

Dichiarare 4 vettori come variabili globali ciascuno di 100 celle. Ciascun vettore ha un nome e il tipo delle celle, come segue:

- Il vettore *numeri* è un vettore di interi
- Il vettore *veritas* è un vettore di bool
- Il vettore *frasi* è un vettore di stringhe
- Il vettore *decim* è un vettore di numeri con la virgola

ESERCIZIO 2. INIZIALIZZAZIONI

Riempire il vettore *numeri* ponendo in ciascuna cella il quadrato del rispettivo indice. Poi visualizzare il vettore nel listBox1.

Esempio: nella cella 7 ci metto 49

ESERCIZIO 3. RIEMPIMENTI

Riempire il vettore *decim* ponendo in ciascuna cella un terzo del rispettivo indice. Poi visualizzare il vettore nel listBox1.

Esempio: nella cella 7 ci metto 2.3333

ESERCIZIO 4. VETTORI DI STRINGHE

Riempire il vettore *frasi* ponendo in ciascuna cella una tra le parole "cuori", "quadri", "fiori", "picche" scelta casualmente. Poi visualizzare il vettore nel listBox1.

Esempio: nella cella 7 ci metto casualmente "fiori"

ESERCIZIO 5. VETTORI DI BOOLEANI

Riempire il vettore *numeri* ponendo in ciascuna cella un numero casuale tra 0 e 999. Poi riempire il vettore *veritas* con true se nella corrispondente cella di numeri c'è un valore pari; false altrimenti.

Poi visualizzare il vettore *veritas* nel listBox1.

Esempio: se nella cella 7 di *numeri* c'è 811 allora nella cella 7 di *veritas* ci metto false

ESERCIZIO 6. VETTORI DI BOOLEANI II

Riempire il vettore *veritas* ponendo in ciascuna cella un valore bool casuale. Poi invertire ogni cella del vettore *veritas* .

Infine visualizzare il vettore *veritas* nel listBox1.

Esempio: nella cella 7 di *veritas* ci metto true; poi dopo lo cambio in false.

ESERCIZIO 7. VETTORE DI DECIMALI

Riempire il vettore *decim* ponendo in ciascuna cella un valore decimale casuale compreso tra -1 e +1. Estremi esclusi.

Infine visualizzare il vettore *decim* nel listBox1.

ESERCIZIO 8. VETTORE DI STRINGHE

Riempire il vettore *frasi* ponendo in ciascuna cella una parola casuale scelta tra "cane" e "gatto".

Modificare il vettore *frasi* concatenando in ciascuna cella la parola " bianco".

Infine visualizzare il vettore *frasi* nel listBox1.

ESERCIZIO 9. VETTORI DI NUMERI CASUALI

Riempire il vettore *numeri* ponendo in ciascuna cella un numero casuale tra 0 e 999.

Riempire il vettore *veritas* con true se nella corrispondente cella di *numeri* c'è un valore maggiore di 500; false altrimenti.

Riempire il vettore *decim* ponendo in ciascuna cella un quinto del valore contenuto nella rispettiva cella di *numeri*.

Riempire il vettore *frasi* ponendo in ciascuna cella la parola "oca" se nella rispettiva cella di *veritas* c'è true. Altrimenti porre la parola "ape".

Modificare il vettore *frasi* concatenando in ciascuna cella la parola " bianca" solo se nella rispettiva cella di *numeri* c'è un valore maggiore di 800. Altrimenti porre la parola "gialla"..

Infine visualizzare il vettore *frasi* nel listBox1.

ESERCIZIO 10. VETTORI DI NUMERI CASUALI II

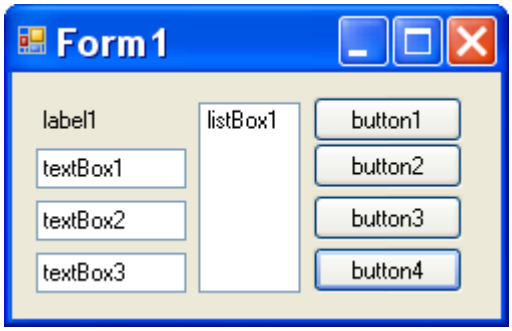
Riempire il vettore *numeri* ponendo in ciascuna cella un numero casuale tra 0 e 999.

Trovare il valore maggiore di numeri. Visualizzarlo nella textBox1.

Trovare il valore minore di numeri. Visualizzarlo nella textBox2.

ESERCIZIO 11. VETTORI DI NUMERI CASUALI III

Preparare il Form1 come in figura :



pulsante 1

Riempie il vettore *numeri* ponendo in ciascuna cella un numero casuale tra -999 e 999.

pulsante 2

Riempie il vettore *veritas* ponendo in ciascuna cella true se il valore rispettivo in *numeri* è positivo, false altrimenti.

pulsante 3

conta quanti numeri positivi ci sono in *numeri* e mostra il risultato in textBox1.

pulsante 4

conta quanti zeri ci sono in *numeri* e mostra il risultato in textBox2.

SOLUZIONI AGLI ESERCIZI

SOLUZIONI AGLI ESERCIZI

SOLUZIONE

```
int[] numeri = new int[100];
bool[] veritas = new bool[100];
string[] frasi = new string[100];
double[] decim = new double[100];
```

SOLUZIONE

```
for (int i = 0; i < 100; i++)
       numeri[i] = i * i;
listBox1.Items.Clear();
for (int i = 0; i < 100; i++)
       listBox1.Items.Add(Convert.ToString(numeri[i]));
```

SOLUZIONE

```
for (int i = 0; i < 100; i++)
       decim[i] = i /3.0;
listBox1.Items.Clear();
for (int i = 0; i < 100; i++)
       listBox1.Items.Add(Convert.ToString(decim[i]));
```

SOLUZIONE

```
Random R = new Random();
for (int i = 0; i < 100; i++)
{
       int x = R.Next(4);
       switch (x)
       {
              case 0:
              frasi[i] = "cuori";
              break;
              case 1:
              frasi[i] = "quadri";
              break;
              case 2:
              frasi[i] = "fiori";
              break;
              case 3:
              frasi[i] = "picche";
              break;
       }

}
listBox1.Items.Clear();
for (int i = 0; i < 100; i++)
       listBox1.Items.Add(Convert.ToString(frasi[i]));
```

SOLUZIONE

```
Random R = new Random();
for (int i = 0; i < 100; i++)
       numeri[i] = R.Next(1000);
for (int i = 0; i < 100; i++)
       if ((numeri[i] % 2) == 0)
              veritas[i] = true;
else
      veritas[i] = false;
listBox1.Items.Clear();
for (int i = 0; i < 100; i++)
       listBox1.Items.Add(Convert.ToString(veritas[i]));
```

SOLUZIONE

```
Random R = new Random();
for (int i = 0; i < 100; i++)
       if (R.Next(2) == 0)
             veritas[i] = true;
else
      veritas[i] = false;
for (int i = 0; i < 100; i++)
      veritas[i] = !veritas[i];
listBox1.Items.Clear();
for (int i = 0; i < 100; i++)
      listBox1.Items.Add(Convert.ToString(veritas[i]));
```

SOLUZIONE

```
Random R = new Random();
for (int i = 0; i < 100; i++)
      if (R.Next(2) == 0)
              decim[i] = R.NextDouble();
else
      decim[i] = -R.NextDouble();
listBox1.Items.Clear();
for (int i = 0; i < 100; i++)
      listBox1.Items.Add(Convert.ToString(decim[i]));
```

SOLUZIONE

```
Random R = new Random();
for (int i = 0; i < 100; i++)
      if (R.Next(2) == 0)
             frasi[i] = "cane";
      else
             frasi[i] = "gatto";
for (int i = 0; i < 100; i++)
      frasi[i] += " bianco";
listBox1.Items.Clear();
for (int i = 0; i < 100; i++)
      listBox1.Items.Add(Convert.ToString(frasi[i]));
```

SOLUZIONE

```
Random R = new Random();
for (int i = 0; i < 100; i++)
      numeri[i] = R.Next(1000);
for (int i = 0; i < 100; i++)
      if (numeri[i]>500)
            veritas[i] = true;
      else
            veritas[i] = false;
for (int i = 0; i < 100; i++)
      decim[i] = numeri[i] / 5.0;
for (int i = 0; i < 100; i++)
      if (veritas[i])
            frasi[i] = "oca";
      else
            frasi[i] = "ape";
for (int i = 0; i < 100; i++)
      if (numeri[i] > 800)
            frasi[i] += " bianca";
      else
            frasi[i] += " gialla";
listBox1.Items.Clear();
for (int i = 0; i < 100; i++)
      listBox1.Items.Add(Convert.ToString(frasi[i]));
```

SOLUZIONE

```
Random R = new Random();
for (int i = 0; i < 100; i++)
      numeri[i] = R.Next(1000);
int max = numeri[0];
for (int i = 0; i < 100; i++)
      if (numeri[i]>max)
            max = numeri[i];
textBox1.Text = Convert.ToString(max);
```

SOLUZIONE

Pulsante 1

```
Random R = new Random();
for (int i = 0; i < 100; i++)
      numeri[i] = R.Next(1000) - R.Next(1000);
listBox1.Items.Clear();
for (int i = 0; i < 100; i++)
      listBox1.Items.Add(numeri[i]);
```

Pulsante 2

```
for (int i = 0; i < 100; i++)
      veritas[i] = (numeri[i] > 0);
listBox1.Items.Clear();
for (int i = 0; i < 100; i++)
      listBox1.Items.Add(veritas[i]);
```

Pulsante 3

```
int cont = 0;
for (int i = 0; i < 100; i++)
      if ( numeri[i] > 0)
            cont++;
textBox1.Text = Convert.ToString(cont);
```

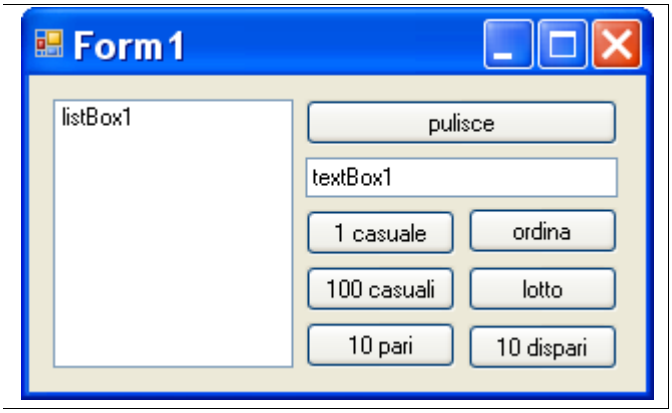
Pulsante 4

```
int cont = 0;
for (int i = 0; i < 100; i++)
      if ( numeri[i] == 0)
            cont++;
textBox1.Text = Convert.ToString(cont);
```

ESERCIZI

ESERCIZIO 12.

- Prepara un form come nella figura e i seguenti gestori di evento:



pulisce cancella gli elementi del listBox1

1 casuale inserisce 1 numero casuale tra gli items del listBox1

ordina ordina gli Items del listBox1

100 casuali inserisce 100 numeri casuali tra gli items del listBox1

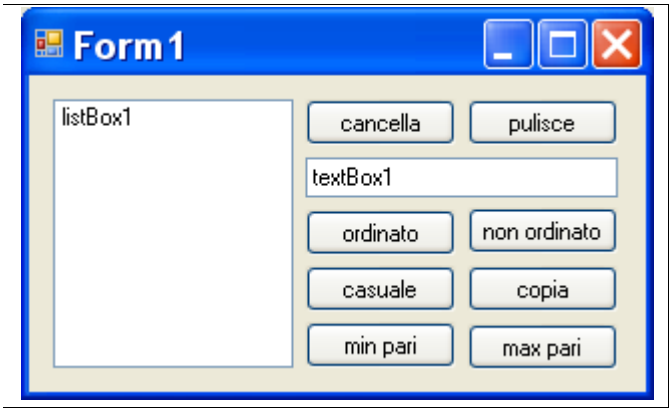
lotto inserisce 1 numero del lotto negli Items del listBox1

10 pari inserisce 10 numeri pari negli Items del listBox1

10 dispari inserisce 10 numeri dispari negli Items del listBox1

ESERCIZIO 13.

- Si dichiari una variabile globale di tipo vettore con 100 celle di interi
- Prepara un form come nella figura e i seguenti gestori di evento:



pulisce cancella gli elementi del listBox1

cancella mette zero in tutte le celle del vettore

ordinato ordina gli Items del listBox1

non ordinato imposta Sorted a False

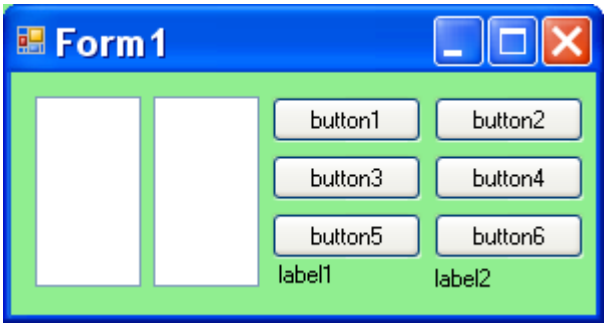
copia copia il vettore nel listBox1

casuale genera 100 numeri casuali nel vettore e senza modificare il listBox1

min pari trova il minore numero pari del vettore e lo scrive nella textBox1

max pari trova il maggiore numero pari del vettore e lo scrive nella textBox1

ESERCIZIO 14.



Crea un progetto visuale e predisponi i controlli come nella figura

Dichiara due vettori di interi

Pulsante 1: genera valori casuali nel primo vettore di interi e lo visualizza nella listBox1

Pulsante 2: cerca nel primo vettore i valori pari e li copia nel secondo vettore; quindi visualizza il vettore nella listBox2

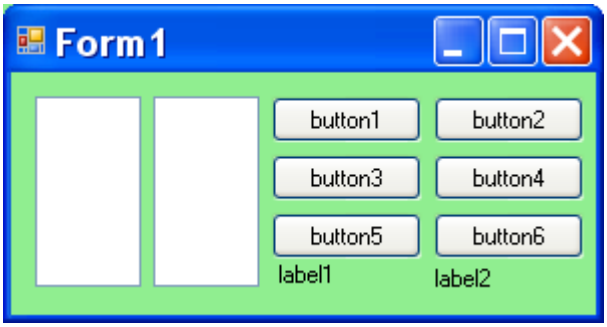
Pulsante 3: cerca nel primo vettore i valori pari e li dimezza; poi visualizza il vettore nella listBox2

Pulsante 4: controlla se nei due vettori ci sono valori comuni; se non ce n'è mostra un messaggio «Non hanno valori comuni» altrimenti mostra il valore del primo trovato

Pulsante 5: scambia di posto nel primo vettore il minimo col massimo; poi visualizza il vettore nella listBox2

Pulsante 6: inverte di segno nel primo vettore i valori superiori alla media; poi lo mostra nella listBox2

ESERCIZIO 15.



Crea un progetto visuale e predisponi i controlli come nella figura

Dichiara due vettori di decimali

Pulsante 1: genera valori casuali compresi tra -1 e zero nel primo vettore e lo visualizza nella listBox1

Pulsante 2: cerca nel primo vettore i valori superiori alla media e li copia nel secondo vettore; quindi visualizza il vettore nella listBox2

Pulsante 3: cerca nel primo vettore i valori pari e li dimezza; poi visualizza il vettore nella listBox2

Pulsante 4: controlla se il primo vettore è ordinato; se non lo è mostra un messaggio «Non ordinato» altrimenti mostra un messaggio «Ordinato»

Pulsante 5: ordina il primo vettore

Pulsante 6: scambia i valori tra i due vettori

SOMMARIO