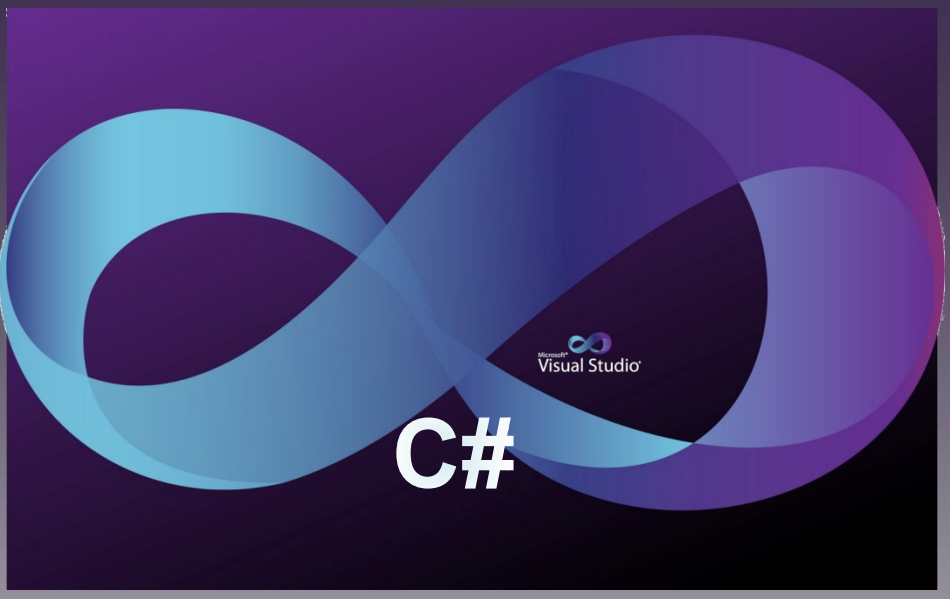
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

ISTRUZIONI ITERATIVE NIDIFICATE

DISPENSA 03.02

03-02_Iterazioni_Nidificate_[ver_17]

Versione del: 28/11/2020
Revisione numero: 17

Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

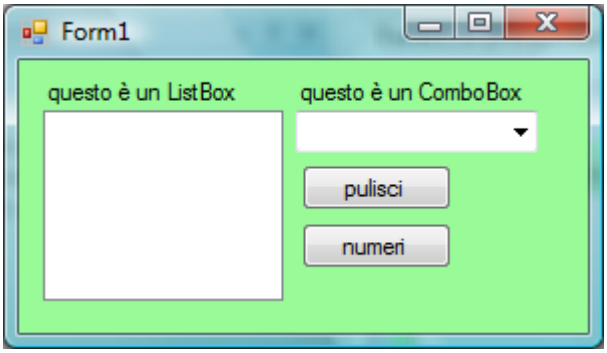
LISTBOX E COMBOBOX

LISTBOX E COMBOBOX

CONTROLLI BASE PER ELENCHI

PROGETTO GUIDATO

- prepara un form1 simile alla figura: oltre alle due etichette e ai due pulsanti, ci sono anche un ListBox ed un ComboBox che trovi nella finestra degli strumenti; puoi lasciare le dimensioni originali;



- associa al pulsante **pulisci** il seguente gestore di evento:

```
listBox1.Items.Clear();
comboBox1.Items.Clear();
```

- associa al pulsante **numeri** il seguente gestore di evento:

```
for (int cont = 0; cont < 20; cont++)
   listBox1.Items.Add(cont);
for (int cont = 0; cont < 20; cont++)
   comboBox1.Items.Add(cont);
```

prova a eseguire il progetto: verifica come si comportano i controlli e prova a usarli.

LISTBOX

Il ListBox è un tipo di controllo visuale, che serve per mostrare un elenco di scelte. Una ListBox somiglia ad un rettangolo che mostra un elenco di frasi, ciascuna delle quali è un elemento. Di solito si usa quando si desidera occupare abbastanza spazio nel Form per consentire all'utente di avere una visione immediata delle opzioni possibili. Nei nostri esercizi sarà usato per mostrare collezioni di numeri o di frasi.

Il ListBox offre molte proprietà utili. Un controllo ListBox può consentire sia selezioni singole (ovvero non oltre un'unica scelta) o multiple (ovvero selezionare più opzioni contemporaneamente) impostando la proprietà **SelectionMode**.

ListBox fornisce anche la proprietà **MultiColumn** per consentire la visualizzazione degli elementi ordinati in colonne piuttosto che in un unico elenco verticale. In tal modo, il controllo può visualizzare più elementi visibili e l'utente non deve scorrere l'elenco per individuare quello desiderato.

COMBOBOX

Il ComboBox è un tipo di controllo visuale, che serve per mostrare un elenco di scelte. Somiglia ad una casella di testo con una freccia al lato. In effetti si chiama Combo (che significa combinato) poiché combina le capacità sia della casella di testo che della ListBox. A differenza della ListBox non mostra subito gli elementi di scelta, ma occorre cliccare sulla freccetta della discesa. Inoltre è possibile anche usarlo per scrivere una scelta dell'utente che NON è in elenco (come accade per una casella di testo); il controllo consente di impedire questa opzione, comunque.

PROPRIETÀ E METODI COMUNI A LISTBOX E COMBOBOX

Prima di analizzare le caratteristiche dei due controlli osserviamo che, mentre le proprietà si possono pensare come contenitori di valori, i metodi sono dei comandi (analoghi alle istruzioni) che richiedono di eseguire un compito o un calcolo tipico del controllo. Senza entrare in dettagli prematuri, possiamo usare questo criterio: le Proprietà si usano senza parentesi tonde e si possono generalmente usare come delle variabili (per assegnare o leggere dei valori). Invece **i metodi si usano sempre con le parentesi tonde e non si possono assegnare loro dei valori.**

I due controlli dispongono delle consuete proprietà come le dimensioni (**`Height`** e **`Width`**), la posizione (**`Left`** e **`Top`**) o il colore (**`BackColor`** e **`ForeColor`**). Inoltre offrono anche altre proprietà e metodi specifici per gli elenchi a discesa.

La proprietà **`Items`** del ListBox è il contenitore degli elementi visualizzati. Questa proprietà consente di ottenere l'elenco di elementi correnti del controllo. Con questa proprietà è possibile aggiungere e rimuovere elementi, ottenere un conteggio degli elementi contenuti nell'insieme.

Attraverso la proprietà Items, è possibile accedere ad altre proprietà. La proprietà **`Count`** dell'Items restituisce il numero di elementi del controllo. Essa si può usare in lettura come, per es.:

```
int quanti = listBox1.Items.Count; //
```

che serve per copiare nella variabile **`quanti`** il numero degli elementi mostrati nella ListBox1.

Per eliminare tutti gli elementi del controllo si usa il metodo Clear (letteralmente Pulizia), come:

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

Si notino le parentesi tonde che lo contraddistinguono come un metodo e non come una proprietà.

Il metodo **`Add`** consente di aggiungere un elemento alla fine dell'elenco. Per esempio:

```
listBox1.Items.Add("cane");
```

aggiunge la frase cane come ultimo elemento della lista. Il codice seguente:

```
for (int cont = 0; cont < 20; cont++)
   listBox1.Items.Add(cont);
```

inserisce 20 elementi numerici nella lista, ordinati da 0 a 19.

Quando un utente seleziona col mouse uno degli elementi, questo viene evidenziato. Esiste una proprietà che serve per sapere quale elemento è selezionato: **`SelectedIndex`**. Questa proprietà si può usare sia in lettura che in scrittura. Essa vale -1 se non ci sono elementi selezionati; vale zero se è selezionato il primo elemento; e ogni altro elemento è numerato progressivamente (0, 1, 2, 3, … per gli elementi primo, secondo, terzo, quarto,…).

```
int quale = listBox1.SelectedIndex; //
```

serve per copiare nella variabile **`quale`** il numero dell'elemento selezionato nella ListBox1.

```
listBox1.Items.Clear(); //
int quale = listBox1.SelectedIndex; //
```

copierà nella variabile **`quale`** il numero –1 poiché non ci sono elementi selezionati nella ListBox1.

Il metodo **`InsertAt`** consente di inserire un elemento in qualsiasi punto dell'elenco. Per esempio:

```
listBox1.Items.Insert(3 , "cane");
```

AGGIUNGE LA FRASE CANE COME **QUARTO** ELEMENTO DELLA LISTA.

Riepilogo delle Proprietà e dei Metodi fondamentali dei Xbox

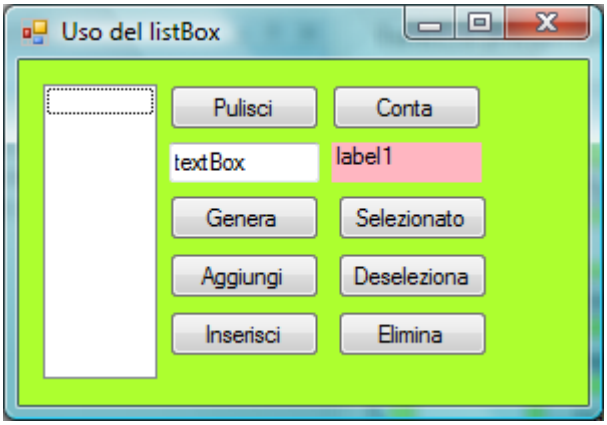
Per semplicità d'uso conviene riportare qui di seguito i principali metodi e proprietà dei controlli di tipo ListBox e ComboBox, che saranno maggiormente usate. Ovviamente i controlli offrono anche altre caratteristiche che puoi scoprire da solo o consultando la guida in linea oppure in seguito quando discuteremo delle interfacce visuali in generale.

Proprietà	Tipo	Scopo
`HEIGHT, WIDTH`	*int*	Altezza e larghezza del controllo sullo schermo
`Left, Top`	*int*	Posizione del controllo dentro il Form
`Items`	*oggetto*	Elenco degli elementi del controllo
`Items.Count`	*int*	Numero di elementi dell'elenco del controllo
`Items.Clear()`	*metodo*	Elimina tutti gli elementi dal controllo
`Items.Add( … )`	*metodo*	Aggiunge un elemento in coda all'elenco
`SelectedItem`	*int*	Indica il numero dell'elemento selezionato oppure –1
`Items.InsertAt( … )`	*metodo*	Inserisce un elemento in una posizione qualsiasi

Osservazione: il controllo visuale non ha un elemento selezionato, inizialmente.

PROGETTO GUIDATO

1. Apri un nuovo progetto visuale e disponi i controlli come nella figura di lato



2. Dichiara la seguente variabile globale:

```
Random estrai = new Random();
```

3. Associa al pulsante «**Pulisci**» il codice:

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

4. Associa al pulsante «**Conta**» il codice:

```
int num = listBox1.Items.Count;
label1.Text = "" + num;
```

5. Associa al pulsante «**Genera**» il codice:

```
for (int k = 0; k < estrai.Next(100); k++)
     listBox1.Items.Add(k);
```

6. Associa al pulsante «**Selezionato**» il codice:

```
int num = listBox1.SelectedIndex;
label1.Text = num + "; " + listBox1.Items[num];
```

7. Associa al pulsante «**Aggiungi**» il codice:

```
if ( textBox1.Text != "" )
     listBox1.Items.Add(textBox1.Text);
else
     MessageBox.Show("Non c'è un testo nella casella!");
```

8. Associa al pulsante «**Deseleziona**» il codice:

```
listBox1.SelectedIndex = -1;
```

9. Associa al pulsante «**Inserisci**» il codice:

```
int pos = listBox1.SelectedIndex;
if (pos < 0)
     pos++;
if (textBox1.Text != "")
     listBox1.Items.Insert(pos , textBox1.Text);
```

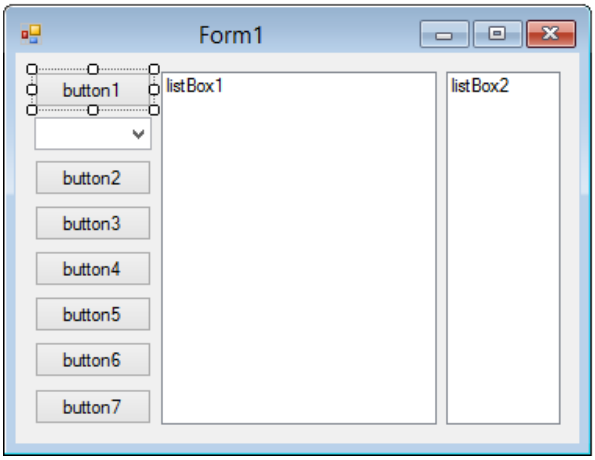
10. Associa al pulsante «**Elimina**» il codice:

```
int pos = listBox1.SelectedIndex;
if (pos > -1)
     listBox1.Items.RemoveAt(pos);
else
     MessageBox.Show("Non c'è un elemento selezionato!");
```

ISTRUZIONI ITERATIVE NIDIFICATE

ANNIDARE ISTRUZIONI ITERATIVE

NIDIFICARE DEI CICLI FOR



PROGETTO GUIDATO CICLI ANNIDATI

- prepara un form1 simile alla figura
- associa al pulsante **button1** il seguente gestore di evento:

VC#

```
private void button1_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear(); //pulizia
    for (int i = 0; i < 3; i++)
        for (int k = 0; k < 3; k++)
                listBox1.Items.Add (i + "\t" + k);
}
```

- associa al pulsante **button2** il seguente gestore di evento:

VC#

```
private void button2_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear(); //pulizia
    for (int i = 0; i < 5; i++)
        for (int k = 0; k < 5; k++)
            listBox1.Items.Add (i * 10 + k );
}
```

- associa al pulsante **button3** il seguente gestore di evento:

VC#

```
private void button2_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear(); //pulizia
    for (int i = 0; i < 5; i++)
        for (int k = i; k < 5; k++)
            listBox1.Items.Add (i * 10 + k );
}
```

- associa al pulsante **button4** il seguente gestore di evento:

VC#

```
private void button4_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear(); //pulizia
    for (int i = 0; i < 5; i++)
        for (int k = (i+1); k < 5; k++)
            listBox1.Items.Add (i + "\t" + k);
}
```

- associa al pulsante **button5** il seguente gestore di evento:

VC#

```
private void button5_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear(); //pulizia
    for (int i = 0; i < 5; i++)
        for (int k = 0; k < i; k++)
            listBox1.Items.Add(i + "\t" + k);
}
```

- associa al pulsante **button6** il seguente gestore di evento:

VC#

```
private void button6_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear(); //pulizia
    for (int i = 0; i < 5; i++)
        for (int k = 0; k < i + 1; k++)
            listBox1.Items.Add(i + "\t" + k);
}
```

- associa al pulsante **button7** il seguente gestore di evento:

VC#

```
private void button7_Click(object sender, EventArgs e)
{
    comboBox1.Items.Clear();//pulizia
    listBox1.Items.Clear(); //pulizia
    listBox2.Items.Clear(); //pulizia
    for (int i = 0; i < 10; i++)
        for (int k = 0; k < 10; k++)
            for (int h = i; h < k; h++)
            {
                listBox1.Items.Add(i + "\t" + k + "\t" + h);
                listBox2.Items.Add(h);
                comboBox1.Items.Add(i * 100 + k * 10 + h);
            }
}
```

- Avvia il progetto e prova tutti i pulsanti e cerca di comprendere il motivo dei risultati ottenuti

OSSERVAZIONI SUL PROGETTO CON ISTRUZIONI ITERATIVE NIDIFICATE

Come abbiamo già studiato per le istruzioni decisionali può accadere di voler inserire una istruzione (iterativa) nel corpo di un'istruzione iterativa; si parla allora di annidare istruzioni iterative o istruzioni iterative nidificate.

Il meccanismo fondamentale da comprendere è che l'istruzione interna viene eseguita mentre l'istruzione iterativa esterna deve attendere il suo completamento. Per esempio nel caso dell'algoritmo seguente:

VC#

```
private void button1_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear(); //pulizia
    for (int i = 0; i < 3; i++)
        for (int k = 0; k < 3; k++)
                listBox1.Items.Add (i + "\t" + k + "\t");
}
```

accade che

Esempio

Traccia dell'esecuzione dell'algoritmo:

i	vale 0	k	vale 0	Add ("0" + "0") → "00"
		k	vale 1	Add ("0" + "1") → "01"
		k	vale 2	Add ("0" + "2") → "02"
		fine ciclo interno		
i	vale 1	k	vale 0	Add ("1" + "0") → "10"
		k	vale 1	Add ("1" + "1") → "11"
		k	vale 2	Add ("1" + "2") → "12"
		fine ciclo interno		
i	vale 2	k	vale 0	Add ("2" + "0") → "20"
		k	vale 1	Add ("2" + "1") → "21"
		k	vale 2	Add ("2" + "2") → "22"
		fine ciclo interno		

È importante osservare che l'iterazione interna deve essere conclusa prima che l'istruzione iterativa esterna possa proseguire.

NIDIFICARE WHILE

È possibile utilizzare istruzioni iterative nidificate in ogni modo. Per esempio:

```
while (condizione)
      istruzione ;
```

```
while (condizione)
      for (regole_indice)
            istruzione ;
```

```
while (condizione_1)
      while (condizione_2)
            istruzione ;
```

```
while (condizione_1)
      do
            istruzione ;
      while (condizione_2)
```

```
do
      istruzione;
while      (condizione);
```

```
do
      for (regole_indice)
            istruzione ;
while (condizione);
```

```
do
      while (condizione)
            istruzione ;
while (condizione);
```

```
do
      do
            istruzione;
      while (condizione);
while (condizione);
```

```
for (inizio; condizione; passo)
      istruzione ;
```

```
for (inizio; condizione; passo)
      while (condizione)
            istruzione ;
```

```
for (inizio; condizione; passo)
      do
            istruzione;
      while (condizione);
```

```
for (inizio; condizione; passo)
      for (inizio; condizione; passo)
            istruzione ;
```

ESERCIZI

ESERCIZI CON LISTBOX E COMBOBOX

ESERCIZIO 1. NUMERI

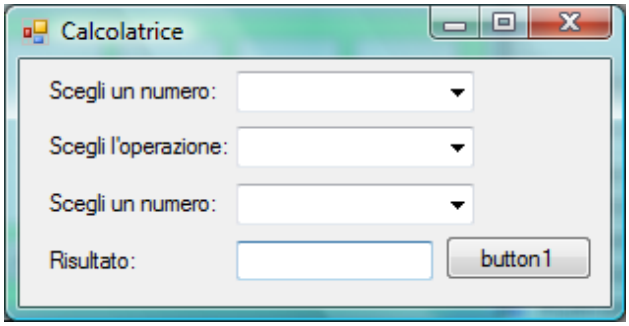
- Prepara un form1 simile alla figura
- Il pulsante «Pari» genera in tutte le listBox 50 numeri pari.
- Il pulsante «Negativi» genera in tutte le listBox 50 numeri negativi.
- Il pulsante «Pulizia» pulisce tutte le listBox.
- Il pulsante «Altro» elimina gli elementi selezionati in tutti gli elenchi.

ESERCIZIO 2. CREARE FRASI

- Prepara un form1 simile alla figura
- Il pulsante «Crea la frase» concatena le frasi selezionate e compone la frase finale nell'etichetta.

ESERCIZIO 3. CALCOLATRICE

- prepara un form1 simile alla figura
- realizza una calcolatrice che esegue tra due numeri le operazioni di base (somma, sottrazione, moltiplicazione, divisione intera, resto della divisione)
- I numeri e le operazioni devono essere scelte con dei ComboBox.

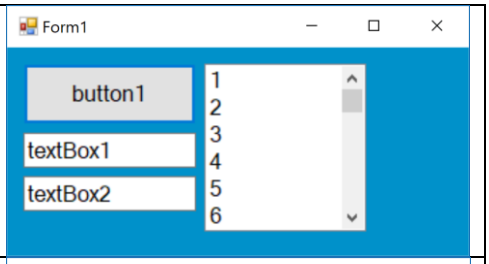


ESERCIZI SUI CICLI DI BASE

ESERCIZIO 1. SEQUENZA DI NUMERI

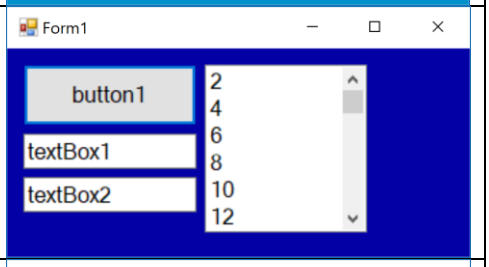
ESERCIZIO 1. SEQUENZA DI NUMERI

- Costruire un programma che scrive in una ListBox la sequenza dei **primi 100 numeri** naturali. Ogni volta che si generano, pulire la ListBox.



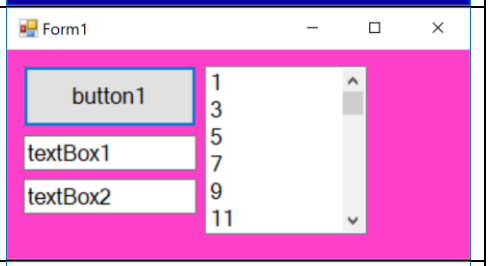
ESERCIZIO 2. SEQUENZA DI NUMERI PARI

- Costruire un programma che scrive in una ListBox la sequenza dei primi 100 numeri naturali **pari**.
- Ogni volta che si generano, pulire la ListBox.



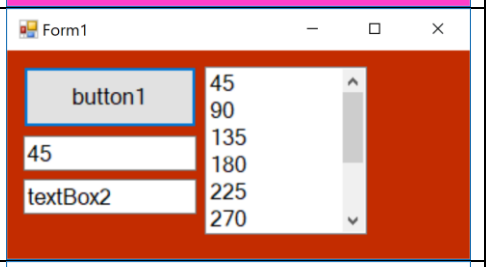
ESERCIZIO 3. SEQUENZA DI NUMERI DISPARI

- Costruire un programma che scrive in una ListBox la sequenza dei primi 100 numeri naturali dispari.
- Ogni volta che si generano, pulire la ListBox.



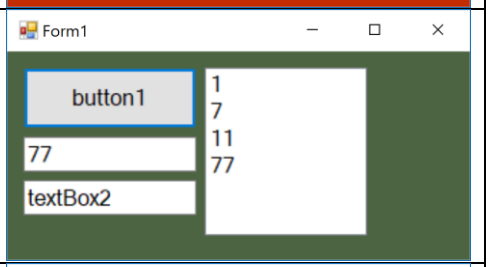
ESERCIZIO 4. TABELLINE

- Costruire un programma che legge da textBox1 un numero intero positivo
- Poi scrive in una ListBox la sua tabellina da 1 a 10
- Pulire la ListBox, prima della scrittura



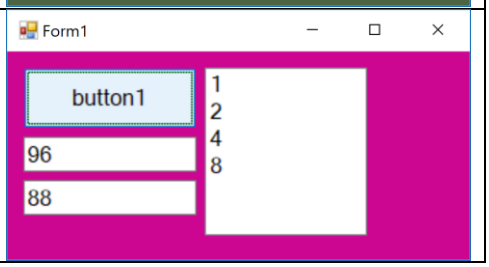
ESERCIZIO 5. DIVISORI

- Costruire un programma che legge da textBox1 un numero intero positivo
- Poi scrive in una ListBox tutti i suoi divisori da 1 a sé stesso
- Pulire la ListBox, prima della scrittura



ESERCIZIO 6. MASSIMO COMUN DIVISORE

- Costruire un programma che legge da due TextBox due differenti numeri interi positivi
- Poi visualizza nella listBox i divisori comuni
-



ESERCIZI SUI CICLI NIDIFICATI

ESERCIZIO 2. ESERCIZIO GUIDATO

- Costruire un programma con un pulsante e una ListBox e associare al pulsante il seguente codice VC#:

```
listBox1.Items.Clear();
for (int d=0; d<10; d++)
  for (int u=0; u<10; u++)
    listBox1.Items.Add(d*10 + u);
```

- Provare a fare un'ipotesi sull'effetto del pulsante prima di provarlo e verificare la correttezza della propria idea.

ESERCIZIO 3. ESERCIZIO GUIDATO

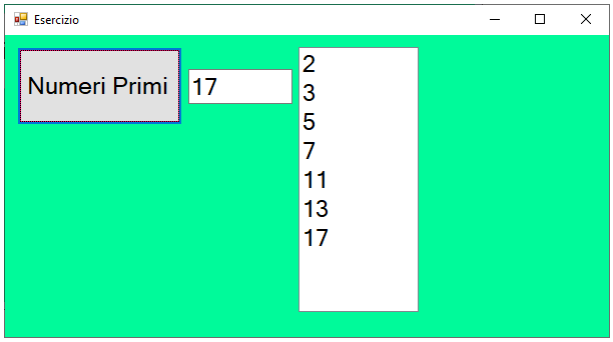
- Costruire un programma con un pulsante e una ListBox e associare al pulsante il seguente codice VC#:

```
listBox1.Items.Clear();
for (int d=0; d<10; d++)
  for (int u=0; u<10; u++)
    if (d==u)
      listBox1.Items.Add("OK " + d)
    else
      listBox1.Items.Add("DIVERSI " + d + " " + u );
```

- Provare a fare un'ipotesi sull'effetto del pulsante prima di provarlo e verificare la correttezza della propria idea.

ESERCIZIO 4. NUMERI PRIMI

- Costruire un programma che legge da una TextBox un numero intero positivo e visualizza in una ListBox quali sono i numeri primi minori o uguali ad esso.
- Ogni volta che si legge un nuovo numero, pulire la ListBox.



ESERCIZIO 5. TABELLINE

- Costruire un programma che scrive in una ListBox Scrivere un programma che chiede all'utente di inserire un numero intero positivo n, e quindi stampa la tabellina moltiplicativa fra tutti i numeri compresi tra 1 e n (inclusi).

ESERCIZIO 6. CONGETTURA DI COLLATZ

- Si consideri la seguente funzione applicabile ad un numero n intero positivo:
  - se n è pari f(n)=n/2
  - se n è dispari f(n)=3*n+1

La congettura di Collatz sostiene che applicando ripetutamente la funzione f() al numero ottenuto dall'iterazione precedente, la sequenza di numeri che si ottiene termina sempre con il valore 1, indipendentemente dal valore n di partenza.

- Scrivere un programma che chiede all'utente un numero k intero positivo, e controlla la veridicità della congettura per tutti i valori interi compresi tra k e 2. Per ogni valore n compreso in tale intervallo il programma dovrà stampare la relativa sequenza numerica.

Esempio: se k=5 il programma stamperà a video

5 -> f(5)=16 -> f(16)=8 -> f(8)=4 -> f(4)=2 -> f(2)=1

4 -> f(4)=2 -> f(2)=1

3 -> f(3)=10 -> f(10)=5 -> f(5)=16 -> f(16)=8 -> f(8)=4 -> f(4)=2 -> f(2)=1

2 -> f(2)=1

Fonte: http://www.dis.uniroma1.it/~fi/wikka.php?wakka=Eserc05091102

ESERCIZIO 7. SEQUENZA DI NUMERI

- Scrivere un programma che visualizza il seguente quadrato usando due cicli annidati:

```
11111111
22222222
33333333
44444444
55555555
66666666
77777777
88888888
```

- Scrivete tre versioni del programma: una usando due for, una usando due while ed una usando due do/while.

ESERCIZIO 8. ASTERISCHI

- Scrivere un programma che legge un intero positivo k e visualizza in una ListBox un quadrato di asterischi di lato k; ad esempio se k=4 allora visualizza:

```
****
****
****
****
```

- Scrivere un programma che legge un intero positivo k e visualizza in una ListBox un triangolo rettangolo di asterischi di lato k; ad esempio se k=4 allora visualizza:

```
****
***
**
*
```

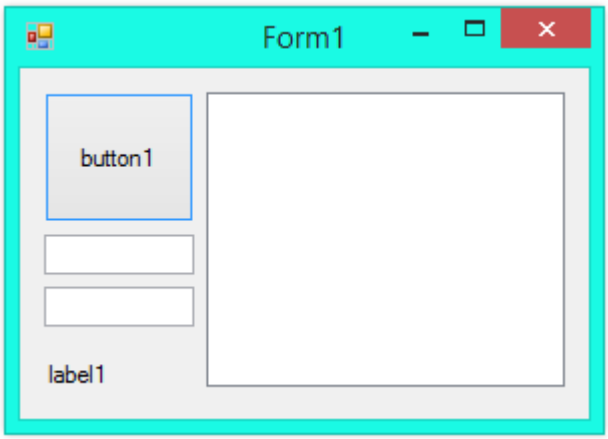
- Scrivere un programma che legge un intero positivo k e visualizza in una ListBox un triangolo rettangolo di asterischi di lato k; ad esempio se k=4 allora visualizza:

```
*
**
***
****
```

ESERCIZI CON ISTRUZIONI ITERATIVE NIDIFICATE

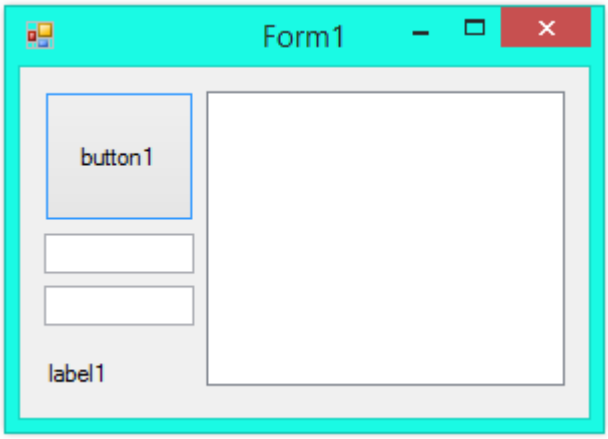
ESERCIZIO 1. TABELLINE

- Prepara un form1 simile alla figura
- Si desidera realizzare un programma che scriva in una listbox la tavola pitagorica di tutte le possibili moltiplicazioni tra i numeri da 1 a 10



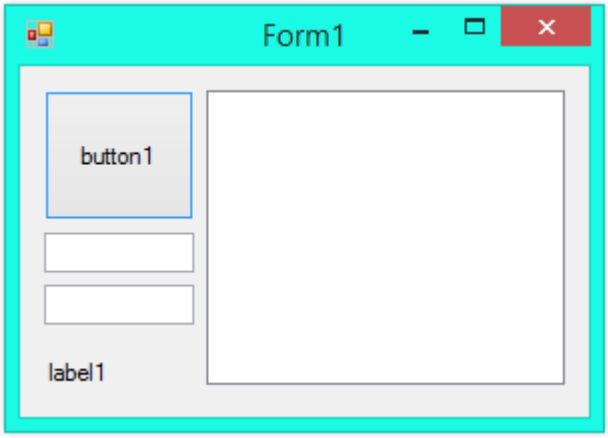
ESERCIZIO 2. DIVISORI

- Prepara un form1 simile alla figura
- Si desidera realizzare un programma che, letto un valore da una casella di testo, deve mostrare in una listbox tutti i divisori di tutti i numeri interi positivi inferiori al numero dato



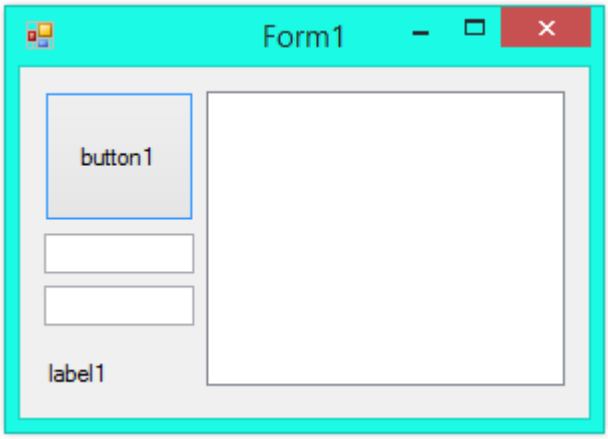
ESERCIZIO 3. NUMERI PRIMI

- Prepara un form1 simile alla figura
- Si desidera realizzare un programma che, letto un valore da una casella di testo, mostri in una listbox tutti i numeri primi inferiori al numero dato



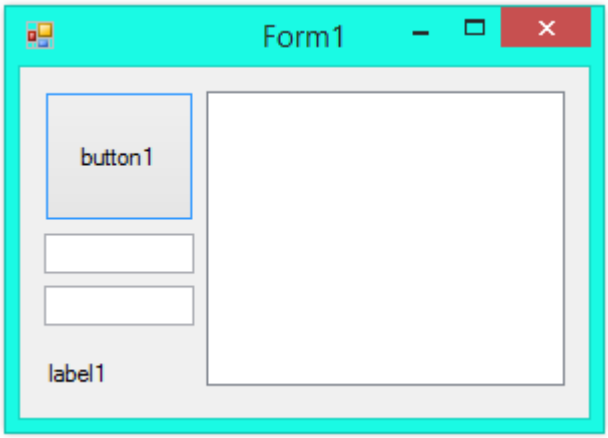
ESERCIZIO 4. DIVISORI BIS

- Prepara un form1 simile alla figura
- Si desidera realizzare un programma che, letti due interi positivi da caselle di testo, deve mostrare in una listbox tutti i divisori di tutti i numeri interi positivi inferiori compresi tra i numeri dati



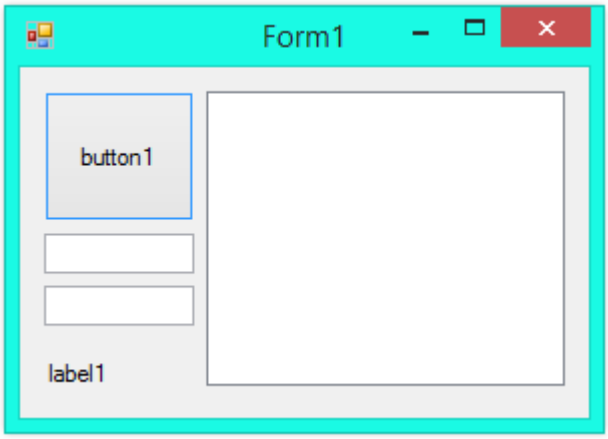
ESERCIZIO 5. PIÙ DIVISORI

- Prepara un form1 simile alla figura
- Si desidera realizzare un programma che, letti due interi positivi da caselle di testo, mostri quale numero ha più divisori (per esempio tra 12 e 13 vince il 12 che ha 6 divisori, mentre il 13 ne ha solo 2)



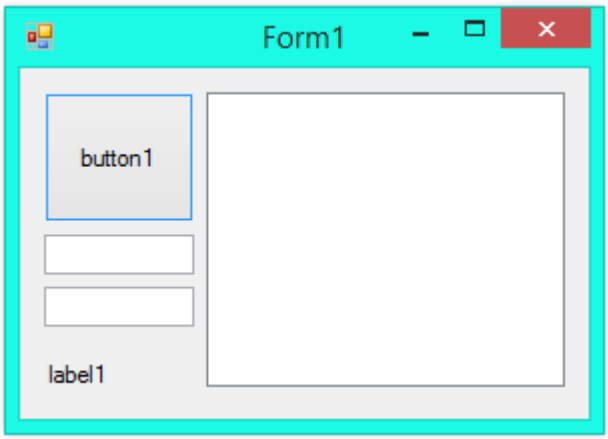
ESERCIZIO 6. MAGGIOR NUMERO PRIMO

- Prepara un form1 simile alla figura
- Si desidera realizzare un programma che, letto un valore da una casella di testo, mostri il più grande numero primo inferiore al numero dato



ESERCIZIO 7. MAGGIOR DIVISORE

- Prepara un form1 simile alla figura
- Si desidera realizzare un programma che, letti due interi positivi da caselle di testo, mostri quale numero ha il maggiore divisore salvo se stessi (per esempio tra 8 e 9 vince l'8 che ha il 4 mentre il 9 ha solo il 3)



SOMMARIO