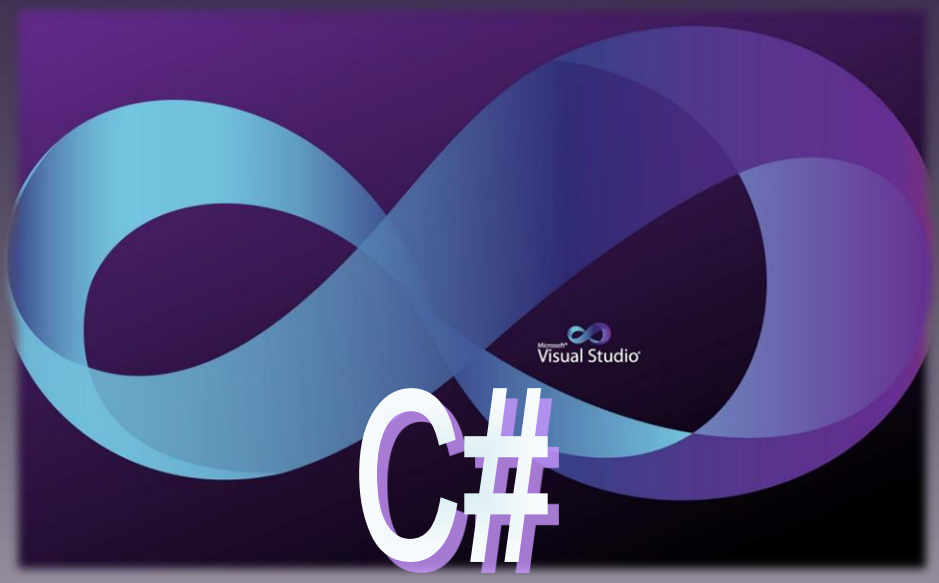
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

ISTRUZIONE DI ASSEGNAZIONE

DISPENSA 01.03

01-03_Assegnazione_[ver_15]

Questa dispensa è rilasciata sotto la licenza Creative Common CC BY-NC-SA. Chiunque può copiare, distribuire, modificare, creare opere derivate dall'originale, ma non a scopi commerciali, a condizione che venga riconosciuta la paternità dell'opera all'autore e che alla nuova opera vengano attribuite le stesse licenze dell'originale.

Versione del: 03/10/2017
Revisione numero: 17

Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

ISTRUZIONI

ISTRUZIONI

COS'È UN'ISTRUZIONE?

ISTRUZIONI, DICHIARAZIONI ED ESPRESSIONI

Le **istruzioni** sono comandi che impongono le azioni che il programma deve eseguire. Le istruzioni comuni sono ad esempio: assegnazione di valori, assegnazioni con operazioni incluse, blocchi di gruppi di istruzioni, invocazioni di metodi (funzioni), esecuzioni di decisioni, iterazioni in cicli.

L'ordine in cui le istruzioni vengono eseguite in un programma, viene chiamato flusso di controllo o flusso di esecuzione. Il flusso di controllo può variare ogni volta che viene eseguito un programma, a seconda di come il programma reagisce agli input ricevuti in fase di esecuzione.

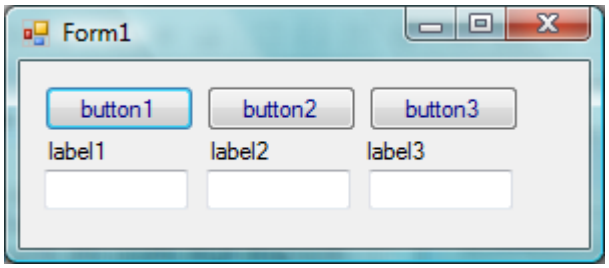
Noi distingueremo le istruzioni dalle dichiarazioni, che invece sono dei modi per costruire qualcosa che il programma utilizzerà in seguito: dichiarazioni di variabili, costruzione di oggetti, definizione di metodi (funzioni).

Distingueremo anche tra istruzioni e espressioni: un'espressione è una sequenza di simboli che può essere valutata e restituisce un valore di un certo tipo. Per esempio `1+1` è un'espressione e NON un'istruzione; `1+1` può essere valutata come 2, di tipo numero intero.

Un'istruzione **può essere** costituita da una sola riga di codice che termina in un punto e virgola o da una serie di istruzioni a riga singola in un blocco. Un **blocco di istruzioni** è racchiuso fra parentesi graffe **{}** e può contenere blocchi nidificati. Nel progetto seguente vengono mostrati due esempi di istruzioni a riga singola e un blocco di istruzioni su più righe.

PROGETTO GUIDATO

- Prepara un **form1** simile alla figura, con tre pulsanti, etichette e caselle di testo



- Adesso fai DOPPIO CLIC sul **button1** e scrivi le seguenti istruzioni nel corpo:

```
textBox1.Text = "Ada";
label1.Text = "Eco";
button2.Top = 0;
button3.Visible = false;
```

- Osserva che ci sono tre righe diverse (tre istruzioni separate) tutte all'interno delle parentesi graffe del gestore di evento
- Seleziona il pulsante **button2** e fai DOPPIO CLIC e scrivi le seguenti istruzioni nel corpo:

```
textBox1.Text = textBox1.Text + "mo" ;
button2.Top = button2.Top + 10;
button3.Visible = ( ! ( button3.Visible ) ) ;
```

- Seleziona il pulsante **button3** e fai DOPPIO CLIC e scrivi le seguenti istruzioni nel corpo:

```
button1.Width = button1.Width + 5;
button1.Height = button1. Height + 5;
button2.Enabled = ( ! ( button2.Enabled ) ) ;
```

ISTRUZIONI

Un programma è un modo per gestire informazioni e situazioni. Quando si scrive un programma occorre spiegare nel dettaglio cosa deve fare.

Immaginiamo che un'azienda assuma un nuovo impiegato. L'impiegato deve svolgere un lavoro, poniamo che debba assumere il ruolo di guardia che deve controllare gli ingressi nel palazzo dove ha sede l'azienda. All'impiegato verrà spiegato cosa deve fare per svolgere bene il suo lavoro: ogni volta che qualcuno vuole entrare nel palazzo, la guardia deve controllare i documenti, verificare che sia un maggiorenne, verificare se è un impiegato dell'azienda o un estraneo, consegnare un distintivo temporaneo ai visitatori e annotare l'accesso nel registro.

Ogni piccolo lavoro che la guardia deve eseguire è spiegata nei dettagli. Per questo la guardia sarà «ISTRUITA» ovvero preparata per sapere cosa fare e farlo bene. Ogni compito è quindi un'istruzione. Un esempio di istruzione potrebbe essere «chiedi il documento alla prima persona della fila all'ingresso», oppure «scrivi il codice del documento nella prima riga libera del registro degli ingressi».

È importante notare che anche se una persona sa cosa deve fare, non è detto che lo stia facendo. Se un vigile sa come dirigere il traffico, può darsi che sia pronto ma che ancora non stia gestendo nulla. Anche per l'elaboratore elettronico è così: quando si predispone un programma lo si «istruisce» ma ancora non lo si manda in esecuzione; quando lo si manda in esecuzione si suppone che abbia tutte le istruzioni necessarie.

Quando si scrive un programma, si è in modalità di Progettazione (**`Editing`**).

Quando si esegue un programma, si è in modalità di Esecuzione (**`Running`**).

Ogni istruzione deve essere scritta bene in modo che il programma sappia attribuirle un corretto e preciso significato e sappia eseguirla quando richiesto.

Un programma è composto di molte istruzioni. Ed esistono molte diverse istruzioni.

SINTASSI

La sintassi è il modo corretto di scrivere un'istruzione. La sintassi segue delle regole che specificano come scrivere bene l'istruzione. Se l'istruzione è scritta male, è scorretta e l programma non può essere eseguito. Ogni istruzione ha le sue regole sintattiche.

SEMANTICA

La semantica è il significato attribuito all'istruzione. La semantica include il suo funzionamento o scopo o uso. Quando si scrive un'istruzione occorre riflettere a cosa essa serva e come si comporti. Ogni istruzione ha la sua semantica.

ASSEGNAZIONE

ISTRUZIONE DI ASSEGNAZIONE

L'assegnazione è una particolare istruzione.

Sintassi

```
Locazione = Espressione ;
```

Semantica

L'istruzione di assegnazione consente di copiare un valore dentro una locazione.

La Locazione è un contenitore di valori. Per il momento pensiamo alle Locazioni come a delle Proprietà. Per esempio **button1.Left** è una locazione. L'espressione è qualcosa che ha un valore. Per es. 324 è una espressione.

Valore Locazione locazione valore

L'assegnazione è l'istruzione che pone il valore di una espressione dentro una locazione. La locazione può contenere un solo valore alla volta; se vi si scrive sopra, essa perde il valore precedente. Per esempio:

```
button1.Left = 13 ;
```

è una istruzione che copia il valore 13 dentro la locazione **button1.Left** . e cancella il dato preesistente.

L'istruzione lavoro nel seguente modo:

- controlla se la locazione è compatibile con il tipo dell'espressione da copiare;
- controlla l'espressione e ne determina il valore;
- copia il valore nella locazione.

Nota: ogni locazione deve avere un nome unico. Per esempio la locazione **button1.Left** è unica e corrisponde alla distanza tra il bordo sinistro della finestra e il bordo sinistro del button1.

ESPRESSIONI

Una Espressione è una sequenza di simboli che può essere valutata e rende un valore di un determinato tipo.

Quindi un'espressione è qualcosa che può essere valutato: per esempio:

```
13 +  17
```

è un'espressione che vale 30 ed è di tipo intero (un numero senza virgola).

Le espressioni possono essere di diverso tipo; in questa introduzione esamineremo tre tipi possibili di espressione (ovviamente ne restano altre che non esamineremo per il momento).

Espressioni numeriche	Rendono valori numerici, come **23** oppure **-17** oppure **+3.14**
Espressioni testuali	Rendono frasi, come **"cane"** oppure **"ciao Mondo"** oppure **"Spa Zio"**
Espressioni logiche	Rendono valori di verità ovvero **true** (vero) oppure **false** (falso)

Una espressione banale è una costante; per esempio **17** è un valore immediato.

Ma altre espressioni si calcolano svolgendo delle **operazioni**. Per esempio 2+1 è una espressione che va calcolata, svolgendo la somma (il computer riesce a calcolare bene!) ed ottenendo 3. Tuttavia esistono molte operazioni non così intuitive che è bene approfondire.

OPERATORI

Un'operazione è contrassegnata da un simbolo che è chiamato **operatore**. Per esempio l'operatore + può essere usato per indicare la somma (ma non solo). Gli operatori lavorano su elementi, detti **operandi**. Per esempio: l'operatore + lavora su due operandi, posti uno a destra ed uno a sinistra. Quindi 3 + 2 è una somma dove l'operatore + agisce su due operandi, il 3 ed il 2.

Alcuni operatori lavorano su due elementi; sono detti **operatori binari**.

Alcuni operatori lavorano su un solo elemento; sono detti **operatori unari**.

OPERAZIONI NUMERICHE

OPERAZIONI ALGEBRICHE

Una operazione algebrica è un calcolo tra numeri che restituisce un numero. Un'operazione intera è un'operazione tra numeri interi che rende un numero intero. Un'operazione decimale è un'operazione tra numeri con la virgola che rende un numero con la virgola. Le operazioni algebriche vanno suddivise tra intere e decimali.

Le **operazioni intere** sono:

+	Addizione	Rende la somma di due interi; esempio 3+2 → 5
-	Sottrazione	Rende la differenza di due interi; esempio 3-11 → 8
*	Moltiplicazione	Rende il prodotto di due interi; esempio 3*2 → 6
/	Divisione	Rende il quoziente intero della divisione senza virgola di due interi; esempio 3/2 → 1
%	Modulo	Rende il resto intero della divisione senza virgola di due interi; esempio 33%7 → 5

Le **operazioni decimali** sono:

+	Addizione	Rende la somma di due decimali; esempio 3.1+2.07 → 5.17
-	Sottrazione	Rende la differenza di due decimali; esempio 3.1-2.07 → 1.03
*	Moltiplicazione	Rende il prodotto di due decimali; esempio 3.1*2.0700 → 6.417
/	Divisione	Rende il prodotto di due decimali; esempio 1.0/4.0 → 0.25

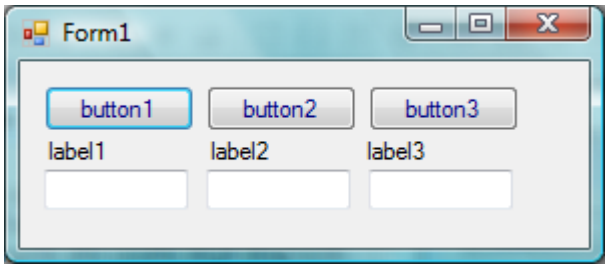
OPERATORI SOVRACCARICATI

Come si può notare alcuni operatori possono adattarsi al tipo degli operandi proposti. Se **entrambi** gli operandi sono interi, il programma usa l'operatore intero; se **almeno uno** degli operandi è decimale, il programma usa l'operatore decimale. Questo può portare a risultati inaspettati;

per esempio:

`3 / 2`	rende	`1`
`3.0 / 2`	rende	`1.5`
`20 / 3`	rende	`6`
`20 / 3.0`	rende	`6.666 666 666`

PROGETTO GUIDATO



- Prepara un form1 simile alla figura, con tre pulsanti, etichette e caselle di testo
- Adesso fai DOPPIO CLIC sul **button1** e scrivi le seguenti istruzioni nel corpo:

```
button2.Left = button2.Left / 2;
```

prova a eseguire il progetto e usare più volte il pulsante **`button1`**

RIFLESSIONE SUL PROGETTO GUIDATO

Esaminiamo il programma precedente. Se la locazione **`button2.Left`** conteneva inizialmente il valore 90, nel momento in cui si preme il pulsante button1, l'assegnazione compie le seguenti operazioni:

1) valuta il contenuto di **`button2.Left`** che vale 90 (intero)
2) esegue la divisione intera (tra interi) **`button2.Left = button2.Left / 2`** che rende 45
3) copia il valore 45 nella locazione **`button2.Left`** che adesso varrà 45 (intero)
4) quindi la locazione, usata nella parte destra della assegnazione si usa come un'espressione e si usa il suo contenuto (valore). L'effetto visivo è di spostare verso sinistra il pulsante button2.

Se si preme una volta ancora il pulsante **`button1`**, l'assegnazione compie le seguenti operazioni:

1) valuta il contenuto di **`button2.Left`** che vale 45 (intero)
2) esegue la divisione intera (tra interi) **`button2.Left = button2.Left / 2`** che rende 22
3) copia il valore 45 nella locazione **`button2.Left`** che adesso varrà 22 (intero)

La locazione può contenere un solo valore alla volta; se vi si scrive sopra, essa perde il valore precedente.

OPERAZIONI TESTUALI

CONCATENAZIONE

Una operazione testuale è un calcolo tra stringhe (frasi) che restituisce una stringa. Esamineremo la concatenazione, che usa il simbolo **+** che rende la stringa ottenuta giustapponendo le due frasi operando.

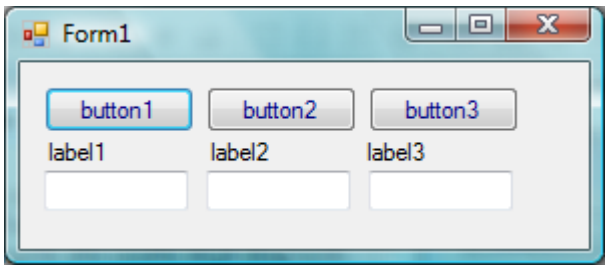
+	Concatenazione	Rende la stringa ottenuta giustapponendo le due stringhe operando; esempio "pomo" + "doro" → "pomodoro"

Per esempio:

```
"accade" + "mia"          rende    "accademia"
"capo" + "collo"          rende    "ciaoMondo"
"ciao" + " " + "Mondo"    rende    "ciao Mondo"
```

Osserva che le stringhe costanti (letterali) sono delimitate da doppie virgolette (dette apici) che non fanno parte della frase, ma ne indicano inizio e fine.

PROGETTO GUIDATO



- Modifica il programma precedente
- Adesso fai DOPPIO CLIC sul **button2** e scrivi le seguenti istruzioni nel corpo:

```
textBox1.Text = textBox1.Text +
"a";
```

prova a eseguire il progetto e usare più volte il pulsante **button2**

RIFLESSIONE SUL PROGETTO GUIDATO

Esaminiamo il programma precedente. Se la locazione **textBox1.Text** inizialmente non aveva alcun valore (si dice che conteneva il valore **stringa vuota**), nel momento in cui si preme il pulsante **button2**, l'assegnazione compie le seguenti operazioni:

1) valuta il contenuto di **textBox1.Text** che vale **""** (stringa vuota)
2) esegue la concatenazione (tra stringhe) **textBox1.Text + "a"** che rende **"a"**
3) copia il valore **"a"** nella locazione **textBox1.Text** che adesso varrà **"a"** (stringa)

quindi la locazione, usata nella parte destra della assegnazione si usa come un'espressione e si usa il suo contenuto (valore). L'effetto visivo è di aggiungere lettere al testo della casella di testo.

Se si preme una volta ancora il pulsante button2, l'assegnazione compie le seguenti operazioni:

1) valuta il contenuto di **textBox1.Text** che vale **"a"** (stringa)
2) esegue la concatenazione (tra stringhe) **textBox1.Text + "a"** che rende **"aa"**
3) copia il valore **"aa"** nella locazione **textBox1.Text** che adesso varrà **"aa"** (stringa)

OPERAZIONI LOGICHE

VALORI DI VERITÀ

Analizzeremo ora la logica intesa come algebra booleana. L'algebra booleana è una particolare algebra (insieme di valori e operazioni) inventata dal matematico George Boole e che usa due soli valori elementari:

`true` (il valore vero)

`false` (il valore falso)

su questi valori elementari è possibile eseguire alcune operazioni, tra cui analizzeremo:

`!` (not ovvero la negazione logica, che inverte il valore originario)

`&&` (and ovvero la congiunzione logica, che rende vero se e solo se gli operandi sono veri)

`||` (or ovvero la disgiunzione logica, che rende falso se e solo se gli operandi sono falsi)

NOT	
True	False
False	True

AND	True	False
True	True	False
False	False	False

OR	True	False
True	True	True
False	True	False

OPERATORE NOT

È un operatore unario, che prende un solo valore logico e rende l'altro valore possibile.

Per esempio:

```
textBox1.Visible = !true ;
```

assegna alla Proprietà **`Visible`** della textBox1 il valore false.

Per esempio:

```
textBox1.Visible = ! textBox1.Visible ;
```

assegna alla Proprietà **`Visible`** l'inverso del suo valore iniziale.

OPERATORE AND

È un operatore binario, che prende due valori logici e rende il valore stabilito dalla tabella già esposta. Per esempio:

```
textBox1.Visible = (true && false);
```

assegna alla Proprietà **`Visible`** della textBox1 il valore false (la nasconde).Per esempio:

```
textBox1.Visible = ( textBox2.Visible && textBox3.Visible );
```

rende visibile la **`textBox1`** solo se sono visibili le caselle **`textBox2`** **e anche** **`textBox3`**.

OPERATORE OR

È un operatore binario, che prende due valori logici e rende il valore stabilito dalla tabella già esposta. Per esempio:

```
textBox1.Visible = (true || true);
```

assegna alla Proprietà Visible della textBox1 il valore true (la mostra). Per esempio:

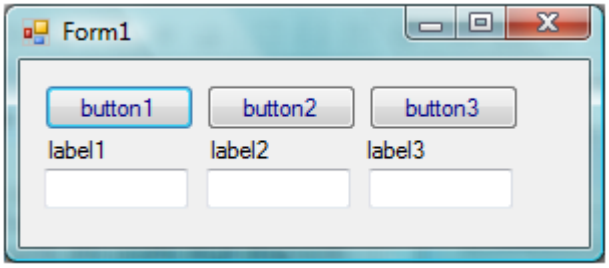
```
textBox1.Visible = ( textBox2.Visible || textBox3.Visible );
```

rende visibile la textBox1 solo se è visibile almeno una delle caselle textBox2 **oppure** textBox3.

PROGETTO GUIDATO

- Prepara un form1 simile alla figura, con tre pulsanti, etichette e caselle di testo
- associa al pulsante **button1** il seguente gestore di evento:

```
textBox1.Visible = ! textBox1.Visible ;
textBox2.Visible = ! textBox1.Visible ;
```



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

```
label1.Visible = ( textBox1.Visible || textBox2.Visible );
```

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

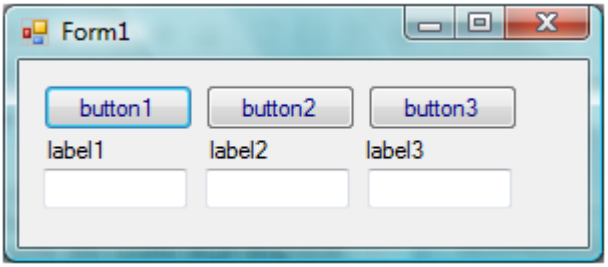
```
label2.Visible = ( textBox1.Visible && textBox2.Visible );
```

prova a eseguire il progetto e cerca di capire come funzionano i pulsanti

OPERATORI DI CONFRONTO

PROGETTO GUIDATO

- Prepara un nuovo progetto
- Prepara il form1 simile alla figura, con tre pulsanti, tre etichette e tre caselle di testo



- associa al pulsante **button1** il seguente gestore di evento:

```
label1.Visible = (textBox1.Text == textBox2.Text);
```

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

```
label1.Visible = !(textBox1.Text == textBox2.Text);
```

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

```
label1.Visible = (textBox1.Text != textBox2.Text);
```

- prova a eseguire il progetto e cerca di capire come funzionano i pulsanti

RIFLESSIONI SUL PROGETTO GUIDATO

Nel progetto precedente, il pulsante **button1** compara il testo delle due caselle; il risultato è un valore logico: vale true se sono uguali, altrimenti vale false. il pulsante **button2** controlla se il testo della prima casella è successivo (in ordine alfabetico) a quello della seconda; il risultato è un valore logico: vale true se il primo segue il secondo, altrimenti vale false. il pulsante **button3** verifica se il testo delle due caselle è diverso; il risultato è un valore logico: vale true se sono uguali, altrimenti vale false.

OPERATORI DI CONFRONTO

Gli operatori come == (uguale), != (diverso), > (maggiore), >= (maggiore o uguale) e altri sono detti operatori di confronto. Tutti gli operatori di confronto restituiscono un valore logico booleano, ovvero rendono true oppure false.

Gli **operatori di confronto** in Visual C# sono:

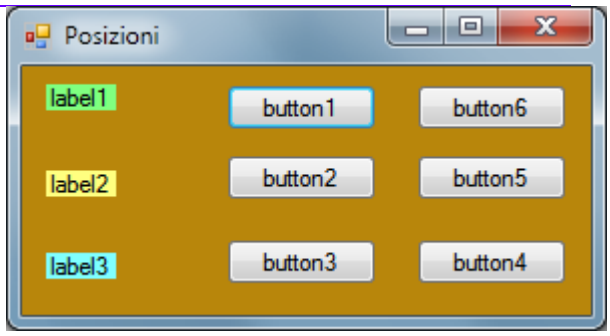
OP	NOME	DESCRIZIONE
==	Uguale	Rende true se i due operandi sono uguali
!=	Diverso	Rende true se i due operandi sono diversi
>	Maggiore	Rende true se il primo operando è maggiore (o successivo) del secondo
>=	Maggiore o uguale	Rende true se il primo operando è maggiore oppure uguale (o successivo) al secondo
<	Minore	Rende true se il primo operando è minore (o precedente) del secondo
<=	Minore o uguale	Rende true se il primo operando è minore oppure uguale (o successivo) al secondo

Se si confrontano numeri, maggiore significa più grande; se si confrontano stringhe, maggiore significa successivo in ordine alfabetico; se si confrontano booleani, allora true **>** false.

ESERCIZI

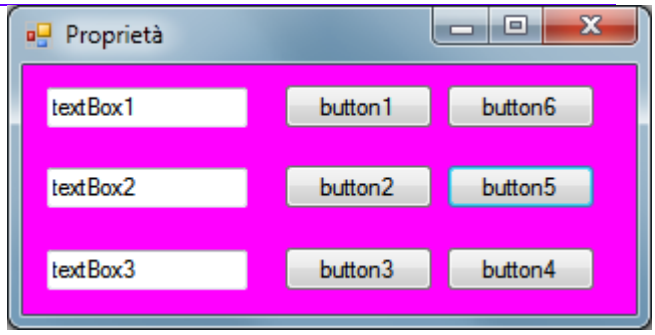
ASSEGNAZIONE E PROPRIETÀ

PROPRIETÀ DEI CONTROLLI



- Prepara un form1 simile alla figura, con sei pulsanti, e tre etichette; porre la proprietà **Autosize** delle etichette a **false**
- Associa al pulsante **button1** un gestore di evento che sposta la prima etichetta di 10 pixel verso destra (suggerimento incrementa la proprietà Left di 10)
- Associa al pulsante **button2** un gestore di evento che sposta la seconda etichetta sotto la prima, ovunque essa sia (usare le proprietà Left e Top);
- Associa al pulsante **button3** un gestore di evento che sposta la prima etichetta di 10 pixel verso l'alto (suggerimento modifica la proprietà Top)
- Associa al pulsante **button4** un gestore di evento che scambia di posizione la prima etichetta con la seconda
- Associa al pulsante **button5** un gestore di evento che raddoppia la larghezza della prima etichetta (porre prima **Autosize** a false)
- Associa al pulsante **button6** un gestore di evento che dimezza la larghezza della prima etichetta (porre prima **Autosize** a false)

PROPRIETÀ LOGICHE



- Prepara un form1 simile alla figura, con sei pulsanti, e tre caselle di testo.
- Associa al pulsante **button1** un gestore di evento che nasconde la prima etichetta e rende visibili le altre due.
- Associa al pulsante **button2** un gestore di evento che nasconde le etichette visibili ma mostra quelle invisibili.
- Associa al pulsante **button3** un gestore di evento che mostra la terza etichetta solo se le altre due sono invisibili; altrimenti la nasconde.
- Associa al pulsante **button4** un gestore di evento che disabilita i primi tre pulsanti.
- Associa al pulsante **button5** un gestore di evento che disabilita sé stesso.
- Associa al pulsante **button6** un gestore di evento che abilita tutti i controlli.

FINESTRE TRASPARENTI

- Scrivere un nuovo programma con un pulsante che nasconde la finestra

CONCATENAZIONI

- Scrivere un nuovo programma con tre caselle di testo ed un pulsante che scrive nella terza casella la concatenazione del testo delle prime due

SOMMARIO