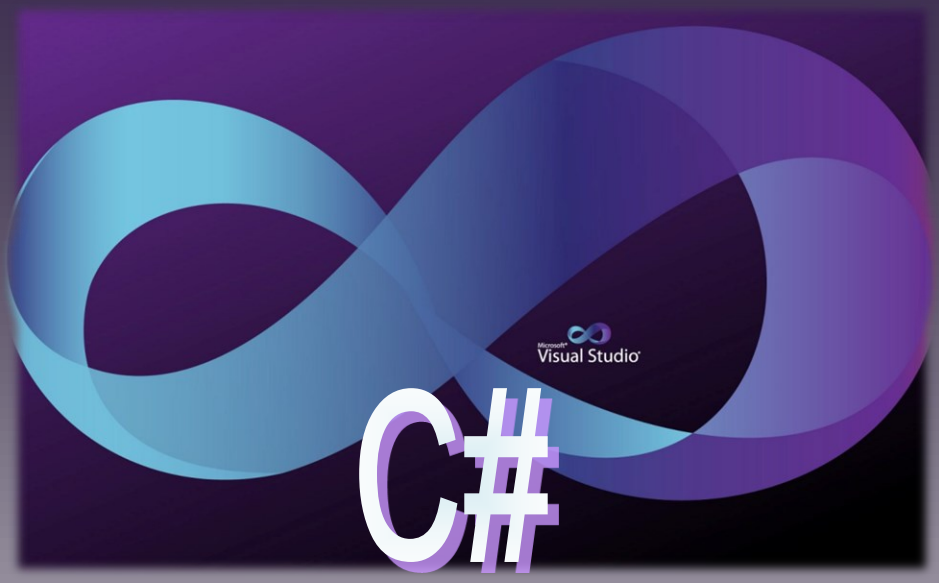
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

SCOPO E USO DELLE VARIABILI ELEMENTARI

DISPENSA 01.04

01-04_Variabili_Base_[ver_22]

Versione del: 04/10/2022
Revisione numero: 22

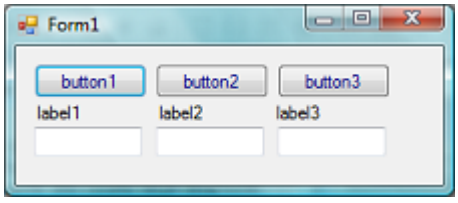
Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

VARIABILI DI TIPI ELEMENTARI

TIPI E CONVERSIONI

PROGETTO GUIDATO 1



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

```
label1.Text = textBox1.Text + textBox2.Text;
```

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

```
label2.Text = textBox1.Text + "1";
```

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

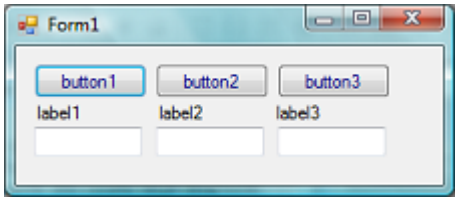
```
label3.Text = "1" + textBox1.Text;
```

- prova a eseguire il progetto e scrivi dei numeri nelle caselle di testo; per esempio scrivi **4** nella prima casella e scrivi **6** nella seconda; quale effetto hanno i pulsanti?

L'esercizio guidato dimostra che il testo nelle caselle è una stringa. Difatti il programma non usa i valori numerici (apparentemente 4 e 6) ma usa le stringhe "4" e "6".

Ma come fare se si desidera confrontare numeri e non testo? Occorre convertire (trasformare) la stringa in numero.

PROGETTO GUIDATO 2



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

```
label1.Text = "" + (Convert.ToInt32(textBox1.Text) + Convert.ToInt32(textBox2.Text));
```

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

```
label2.Text = textBox1.Text + 1;
```

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

```
label3.Text = "" + ( Convert.ToInt32(textBox1.Text) + 1 );
```

- prova a eseguire il progetto e scrivi dei numeri nelle caselle di testo; per esempio scrivi **4** nella prima casella e scrivi **6** nella seconda; quale effetto hanno i pulsanti?

Osservazione: i comandi (i metodi) **Convert.ToInt32** sono un esempio di conversioni di tipo ovvero degli strumenti che permettono di trasformare un valore di un certo tipo in un valore di un altro tipo.

Per esempio, **Convert.ToInt32**(textBox1.Text) prende il testo di textBox1 e lo trasforma in numero intero. Invece, **Convert.ToString**(13) prende il numero 13 e lo trasforma in una frase «13» di tipo stringa. Per esempio:

```
label1.Text = Convert.ToInt32( 1 + Convert.ToInt32(textBox1.Text) );
```

prende il testo di textBox1, poi lo trasforma in numero intero, poi ci somma +1, poi lo trasforma in stringa e infine lo scrive nel testo di label1.

IN INTERO (LETTURA DA TEXTBOX)

Per convertire una stringa (che rappresenta un numero) in un numero intero si usa il metodo:

```
Convert.ToInt32(stringaDaConvertire)
```

Per esempio:

```
button1.Left = Convert.ToInt32(textBox1.Text);
```

converte la frase scritta nella casella textBox1 in un numero e lo assegna alla Proprietà Left di button1; l'assegnazione è corretta poiché il risultato della conversione è un intero, così come Left.

Attenzione: se il testo scritto nella casella non fosse un numero (per esempio è una stringa vuota oppure è una frase che non rappresenta un numero, come «poldo» solleva un errore!

IN STRINGA (SCRITTURA SU TEXTBOX)

Per convertire un numero in una stringa si usa il metodo:

```
Convert.ToString(numeroDaConvertire)
```

Per esempio:

```
textBox1.Text = Convert.ToString(button1.Left);
```

converte il valore (intero) della Proprietà Left di button1 in stringa e lo assegna come frase della scritta nella casella textBox1. Per esempio:

```
textBox1.Text = Convert.ToString (3.14);
```

converte il valore (decimale) costante 3.14 **in stringa** e lo assegna come frase della scritta nella casella textBox1.

IN DECIMALE (DOUBLE)

Per convertire una stringa (che rappresenta un numero) in un numero decimale si usa il metodo:

```
Convert.ToDouble(stringaDaConvertire)
```

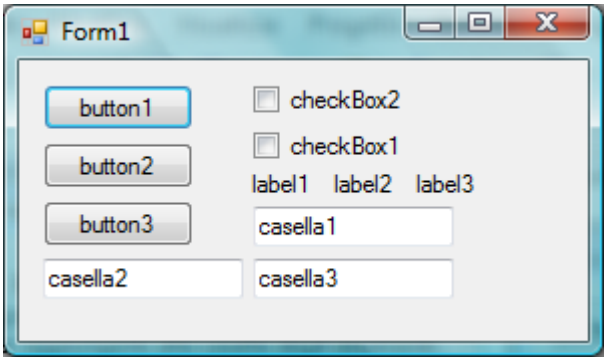
Per esempio:

```
button3.Left = Convert.ToInt32(textBox1.Text);
```

converte la frase scritta nella casella textBox1 in un numero e lo assegna alla Proprietà Left di button1; l'assegnazione è corretta poiché il risultato della conversione è un intero, così come Left.

VARIABILI LOCALI

PROGETTO GUIDATO 3



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

```
string frase = textBox1.Text;
textBox1.Text = button1.Text;
button1.Text = frase;
```

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

```
int numero = label1.Left;
label1.Left = label2.Left;
label2.Left = numero;
```

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

```
bool logico = checkBox1.Checked;
checkBox1.Checked = ! checkBox1.Checked;
checkBox2.Checked = logico;
```

- prova a eseguire il progetto

PROPRIETÀ

Abbiamo già visto che le Proprietà sono caratteristiche di un oggetto. Quando metti un componente in Form1 puoi definirne l'aspetto usando le sue proprietà. Hai anche visto che oggetti diversi, per esempio button1 e label2, hanno proprietà simili, come Text. Per riconoscere le proprietà di oggetti diversi è spesso necessario specificare il nome dell'oggetto seguito da un punto e poi dall'identificatore di proprietà. Per esempio `button1.Text` indica la proprietà Text del button1, ma `button2.Text` indica la proprietà Text del button2.

LOCAZIONI, VARIABILI E PROPRIETÀ

Oltre alle Proprietà esistono numerose altre locazioni. Le locazioni più comuni e utilizzate sono le variabili. Una variabile è un contenitore di dati, connotato da un nome e un tipo.

Tutti i linguaggi hanno bisogno di conservare i dati su cui lavorano. Se per esempio il programma chiede di inserire un numero, è necessario che sia possibile conservare questo numero da qualche parte, per poterlo usare in seguito. Solitamente nei linguaggi di programmazione si usa il termine variabile per indicare il luogo dove conservare tale valore. In realtà il contenitore di valori dovrebbe essere chiamato locazione. Le variabili sono particolari locazioni.

È però doveroso specificare questa distinzione perché, per programmare in Visual C#, sarà necessario usare locazioni che non sono variabili. Un esempio di locazioni che abbiamo già visto ed usato, è costituito dalle proprietà dei componenti, le quali, tuttavia, non sono variabili.

La locazione è un contenitore di dati. Puoi pensare una locazione come un recipiente di cucina. I recipienti servono per contenere gli ingredienti utili per cucinare e per mangiare. Ogni recipiente ha una forma e una dimensione diversa, perché serve per uno scopo diverso dagli altri. Se per bere è preferibile evitare una padella, così per scaldare una bistecca è meglio non usare un bicchiere. Se usi il recipiente errato puoi ottenere effetti indesiderati e pericolosi.

Anche Visual C# usa diverse locazioni. Ogni locazione ha una forma particolare; la forma della locazione è detto tipo. Ogni locazione può contenere un valore ovvero un contenuto. Il contenuto deve essere adatto al

tipo del contenitore (prova a pensare: puoi mettere dell'acqua nello scolapasta per cucinare?). Per dirlo correttamente il valore contenuto nella locazione deve essere compatibile col tipo di dato della locazione. Per poter riconoscere le locazioni Visual C# deve dare loro un nome. In pratica è come se per riconoscere i bicchieri di tutta la cucina ogni bicchiere avesse un proprio nome. Così quando mi serve il bicchiere "Andrea" non lo confondo col bicchiere "Giulia". Per esempio per risolvere un certo problema potrei usare i seguenti contenitori:

Nome	Tipo	Valore
CANE	stringa	`"Bruto"`
PESO	decimale	`12.8`
ANNI	intero	`4`
ADDESTRATO	logico	`True`

Per poter riconoscere le locazioni Visual C# deve dare loro un nome; questo nome è detto identificatore. Gli identificatori devono essere tutti diversi. Quindi le locazioni hanno un identificatore, un tipo ed un valore.

- Una **locazione** ha un nome detto **identificatore**
- Una locazione ha una forma detto **tipo**
- Una locazione può avere un contenuto detto **valore**
- Una **variabile** è una locazione
- Una **proprietà** è una locazione

DICHIARAZIONI DI VARIABILI LOCALI

Quando si apre un gestore di metodo come button1_Click abbiamo visto che si predispone un corpo, racchiuso tra parentesi graffe. Dentro le parentesi graffe è possibile scrivere istruzioni, ma non solo. Dentro il corpo è possibile scrivere anche locazioni temporanee, utili al gestore, per memorizzare temporaneamente dei dati. La dichiarazione di una variabile avviene con una delle forme seguenti:

```
tipo nome ;
```

```
tipo nome = valore ;
```

per esempio:

```
string proverbio ;
int altezza ;
double prezzoMedio ;
bool acceso ;
```

```
string frase = textBox1.Text;
int numero = 13;
double pigreco = 3.14;
bool promosso = True;
```

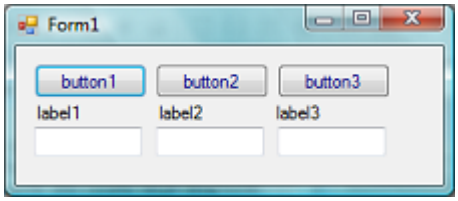
Come si vede le variabili ammettono diversi nomi. I tipi delle variabili sono molteplici, ma noi analizzeremo in particolare questi quattro tipi fondamentali:

Tipo	Descrizione	Esempio
BOOL	Logico (true oppure false)	`True`
DOUBLE	Decimale con virgola	`12.8`
INT	Intero senza virgola	`4`
STRING	Frase	`"Due Mezzi Minuti!"`

Quando si dichiara una variabile locale essa ha una vita breve. La variabile è creata quando serve (per esempio quando si esegue il gestore di evento) usata per il blocco in cui è dichiarata (per esempio nel corpo del metodo gestore di evento) e poi è distrutta ed il suo valore perduto.

Nello stesso blocco non è possibile avere variabili con lo stesso nome. Ma è possibile avere variabili dello stesso tipo, purché con nomi diversi.

PROGETTO GUIDATO 4



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

```
string alberto = label1.Text;
label1.Text = textBox1.Text;
textBox1.Text = alberto;
```

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

```
int bianca = label1.Left;
label1.Left = label2.Left;
label2.Left = bianca;
```

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

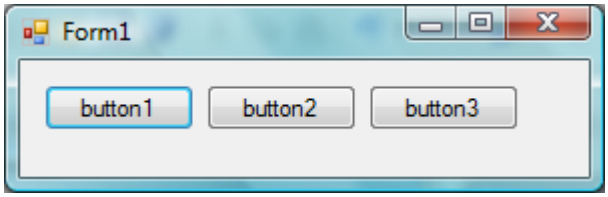
```
bool claudio = label1.Visible;
label1.Visible = ! label1.Visible;
label2.Visible = claudio;
```

- prova a eseguire il progetto e scrivi «davide» nella casella di testo textBox1; quale effetto hanno i pulsanti?

DICHIARAZIONI DI VARIABILI GLOBALI

- Apri un nuovo progetto e vai a vedere il codice scritto associato. Esso si presenta simile al seguente:

```
using System;
. . .
namespace WindowsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```



- La parte di programma che ci interessa è contrassegnata dalla intestazione:

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

questo è il programma principale, chiamato da Visual C# classe del programma.

Tralasciando cosa significhi, osserviamo che se poniamo dei controlli (es. pulsanti e associamo gestori di evento, essi vengono scritti dentro questa classe.

Per esempio:

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        //gestore dell'evento clic del pulsante button1
    }

    private void button2_Click(object sender, EventArgs e)
    {
        //gestore dell'evento clic del pulsante button2
    }

    private void button3_Click(object sender, EventArgs e)
    {
        //gestore dell'evento clic del pulsante button3
    }
}
```

Abbiamo visto che se scriviamo delle variabili dentro il corpo dei metodi gestori di evento, esse sono locali, ovvero temporanee e verranno distrutte. Tuttavia è possibile dichiarare variabili non locali ma utilizzabili da tutti i metodi.

Per esempio:

```
public partial class Form1 : Form
{
    //variabili globali
    int perTutti = 2;

    public Form1()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        //variabili locali a button1
        int prima = 13;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        //variabili locali a button2
        prima = 17;
        perTutti = 3;
    }
}
```

Questa variabile è utilizzabile da tutti i metodi

Questa variabile è utilizzabile solo in questo corpo del button1_Click

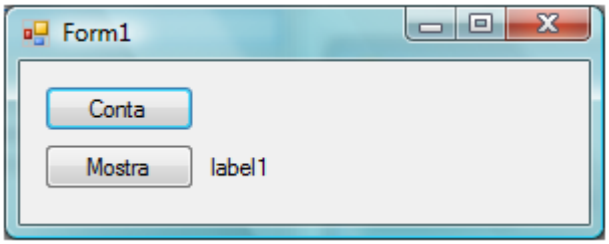
Errore: button2_Click non può usarla

OK: è utilizzabile anche qui

CONTATORE

PROGETTO GUIDATO 5

- Prepara un form1 simile alla figura
- associa al pulsante **Conta** il seguente gestore:

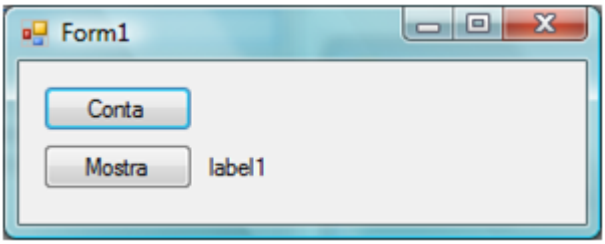



```
int contatore = 0;
contatore = contatore + 1 ;
```

- associa al pulsante **Mostra** il seguente gestore:

```
int contatore;
label1.Text = Convert.ToString (contatore) ;
```

- prova a eseguire il progetto
- questo progetto NON funziona: hai compreso il motivo?



PROGETTO GUIDATO 6

- Prepara un form1 simile alla figura
- associa al pulsante **Conta** il seguente gestore:

```
contatore = contatore + 1 ;
```

- associa al pulsante **Mostra** il seguente gestore:

```
label1.Text = Convert.ToString (contatore) ;
```

- adesso dichiara una variabile globale contatore di tipo intero, ovvero:

```
int contatore = 0;
```

- prova a eseguire il progetto

Il progetto deve assomigliare al seguente:

```
int contatore = 0;

private void button1_Click(object sender, Event
{
    contatore = contatore + 1;
}

private void button2_Click(object sender, Event
{
    label1.Text = Convert.ToString(contatore);
}
```

La dichiarazione di contatore è globale rispetto ai gestori di evento dei pulsanti.

CONTATORI

Alcune applicazioni richiedono di effettuare dei **conteggi** su determinati eventi. Per esempio potrebbe essere importante sapere quante volte è stato premuto un pulsante. Per ricordare un valore ed incrementarlo si usa una variabile (di solito intera) che viene aumentata di 1 ogni volta.

Definizione: le variabili di conteggio si dicono contatori.

L'istruzione di assegnazione è della forma:

```
variabile = variabile + 1;
```

Si deve notare che il linguaggio lavora coi seguenti passi:

1. Valuta l'espressione di destra nel seguente modo: valore della variabile e calcolo della somma con 1
2. Controllo del tipo
3. Assegnazione del valore alla variabile e conseguente perdita del vecchio valore

Se per esempio variabile vale 3 allora dopo avere eseguito l'istruzione contiene 4, mentre il 3 è andato perduto.

Nota: una variabile elementare conserva UN SOLO VALORE

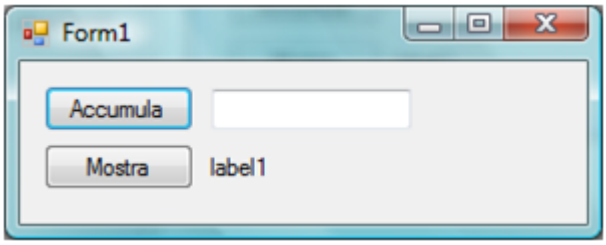
ACCUMULATORE

In alcuni casi è possibile incrementare di un valore costante qualsiasi anziché 1. In questo caso il contatore assume la forma:

```
variabile = variabile + numero;
```

PROGETTO GUIDATO 7

- Prepara un form1 simile alla figura



- associa al pulsante **Accumula** il seguente gestore:

```
int x = Convert.ToInt32 (textBox1.Text);
accumulatore = accumulatore + x ;
```

- associa al pulsante **Mostra** il seguente gestore:

```
label1.Text = Convert.ToString (accumulatore) ;
```

- adesso dichiara una variabile globale contatore di tipo intero, ovvero:

```
int accumulatore = 0;
```

- prova a eseguire il progetto

ACCUMULATORI

In altre situazioni occorre una variabile per ricordare la somma di valore diversi; per esempio la somma di una sequenza di numeri inseriti da tastiera. In tal caso occorre una variabile da incrementare con valori incerti.

Definizione: le variabili di sommatoria si dicono accumulatori.

L'istruzione di assegnazione è della forma:

```
variabile = variabile + valore;
```

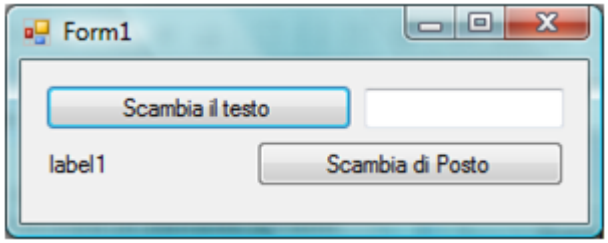
Per comprendere la differenza tra contatore e accumulatore si pensi a questo esempio:

Per organizzare una festa, un ragazzo è incaricato di raccogliere i soldi degli amici e comprare cibo e bevande. Tuttavia ciascun amico può contribuire diversamente, in funzione della sua disponibilità economica attuale. Per questo il ragazzo si presenta con una busta dove riporre il denaro. Ogni volta che un amico versa del denaro, la busta contiene il denaro che c'era in precedenza a cui si somma il denaro appena versato: la busta è un accumulatore. Il ragazzo invece scrive in un foglio una crocetta per ogni amico che ha contribuito: il foglio è un contatore.

SCAMBIO

PROGETTO GUIDATO 8

- Prepara un form1 simile alla figura
- associa al pulsante **Scambia il testo** il seguente gestore:




```
string x = textBox1.Text;
textBox1.Text = label1.Text;
label1.Text = x ;
```

- associa al pulsante **Scambia di posto** il seguente gestore:

```
int x = textBox1.Top;
textBox1.Top = label1.Top;
label1.Top = x ;
x = textBox1.Left;
textBox1.Left = label1.Left;
label1.Left = x ;
```

- prova a eseguire il progetto

SCAMBIO

Talvolta occorre scambiare i valori contenuti in due variabili. Consideriamo ad esempio due variabili x ed y. Un tentativo provvisorio per scambiare il valore di x e di y potrebbe essere il seguente che però produce un risultato scorretto:

```
x = y ;
y = x ;
```

Perché non funziona?

Si deve notare che il linguaggio lavora coi seguenti passi:

1. Valuta l'espressione di destra nel seguente modo: valore della variabile e calcolo della somma col valore
2. Controllo del tipo
3. Assegnazione del risultato alla variabile e conseguente perdita del vecchio contenuto

Perciò la prima assegnazione cancella il valore della x che va perduto. La seconda assegnazione assegna ad y il suo stesso valore. Alla fine entrambe hanno il valore della y. Per risolvere il problema occorre una variabile di appoggio, temporanea.

```
Temp = x ;
x = y ;
y = Temp ;
```

La variabile Temp conserva il valore di x per poterlo assegnare ad y nella ultima istruzione.

Il tipo della variabile di appoggio deve essere compatibile col tipo delle variabili da scambiare; per esempio se occorre scambiare locazioni intere, allora la variabile deve essere di tipo intero:

```
int x = textBox1.Top;
textBox1.Top = label1.Top;
label1.Top = x ;
```

se occorre scambiare locazioni di tipo logico, anche la variabile deve essere di tipo logico:

```
bool b = textBox1.Visible;
textBox1.Visible = label1.Visible;
label1.Visible = b ;
```

se occorre scambiare locazioni di tipo stringa, anche la variabile deve essere di tipo stringa:

```
stringa s = textBox1.Text;
textBox1.Text = label1.Text;
label1.Text = s ;
```

se occorre scambiare locazioni di tipo colore, anche la variabile deve essere di tipo Color:

```
Color c = textBox1.BackColor;
textBox1.BackColor = label1.BackColor;
label1.BackColor = c ;
```

COMMENTI

Spesso è opportuno annotare dei commenti nel programma per ricordarsi lo scopo o il funzionamento dell'algoritmo. I commenti si possono inserire in qualsiasi punto del programma. tra i commenti che dovresti inserire ci sono i seguenti:

```
// Autore: Verde Giada
// Classe: IV A Informatica
// Descrizione: Gioco degli scacchi in rete
// Data inizio: 01/04/2039
// Data ultima modifica: 21/05/2041
```

COMMENTI SU SINGOLA RIGA

Spesso è opportuno annotare dei commenti nel programma per ricordarsi lo scopo o il funzionamento dell'algoritmo. I commenti si possono inserire in qualsiasi punto del programma. I commenti su una riga si scrivono iniziandoli con il doppio slash ovvero così:

```
//commento
```

COMMENTI SU PIÙ RIGHE

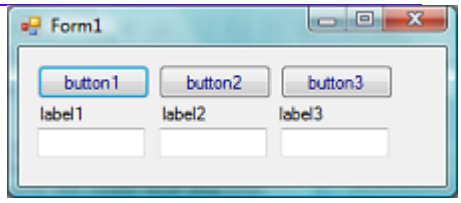
I commenti su più righe si scrivono iniziandoli con /* e infine chiudendoli con */

```
/*   Autore: Carta Rosa
     Classe: IV A Informatica
     Descrizione: Gioco degli scacchi in rete
     Data inizio: 01/11/2017
*/   Data ultima modifica: 21/05/2018
```

questa tecnica serve anche per escludere pezzi di programma dall'esecuzione.

ESERCIZI

ESERCIZIO 0. 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:

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

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

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

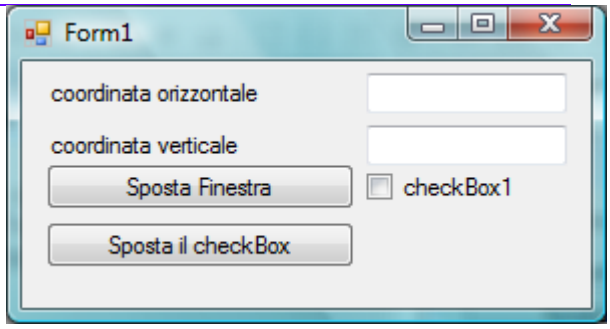
- associa al pulsante **button3** il seguente gestore di evento:

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

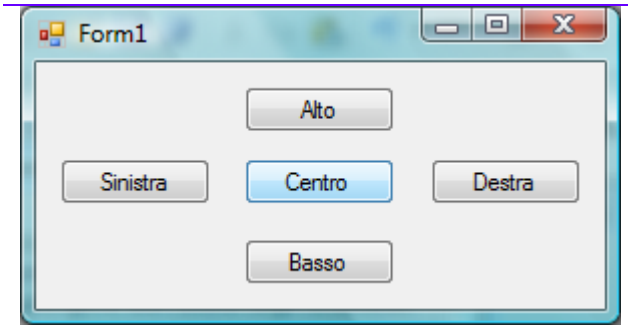
prova a eseguire il progetto e scrivi dei numeri nelle caselle di testo; per esempio scrivi **Anna** nella prima casella e scrivi **Otto** nella seconda; quale effetto hanno i pulsanti? Prova anche ad usare **Anna** in entrambe.

ESERCIZIO 1. COORDINATE

- Scrivere un programma che sposta la finestra a seconda delle coordinate inserite nelle due caselle di testo.
- L'altro pulsante sposta la casella di spunta in base alle coordinate inserite nelle due caselle di testo.



ESERCIZIO 2. LOCAZIONI



- Scrivere un programma con 4 pulsanti.
- Il button1 sposta la finestra verso destra di 10 pixel
- Il button2 sposta la finestra verso sinistra di 10 pixel
- Il button3 sposta la finestra verso l'alto di 10 pixel
- Il button4 sposta la finestra verso il basso di 10 pixel
- Il button5 sposta la finestra al centro dello schermo (usa i valori 600 e 400 se non ne trovi di migliori).

ESERCIZIO 3. RUOTA POSIZIONI

- Scrivere un programma con cinque pulsanti.
- Il pulsante al centro fa ruotare le posizioni degli altri quattro, in senso orario.
- Usare una variabile di tipo int.
- Per avere una visibilità migliore cambia i colori dei 4 pulsanti impostando le rispettive Proprietà BackColor.

ESERCIZIO 4. RUOTA COLORI

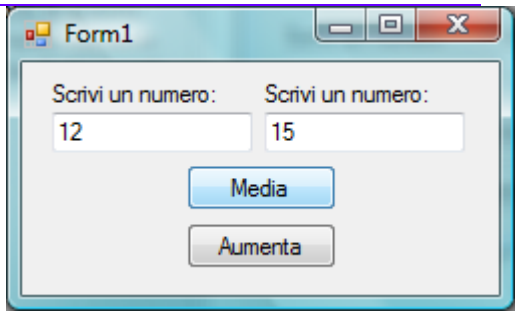
- Scrivere un programma con cinque pulsanti. Il pulsante al centro fa ruotare i colori (non le posizioni) degli altri quattro, in senso orario. La variabile di appoggio è di tipo Color, come nell'esempio:

```
Color appoggio = button1.BackColor;
button1.BackColor = button2.BackColor;
button2.BackColor = appoggio;
```

Nota: il codice qui sopra non risolve l'esercizio!

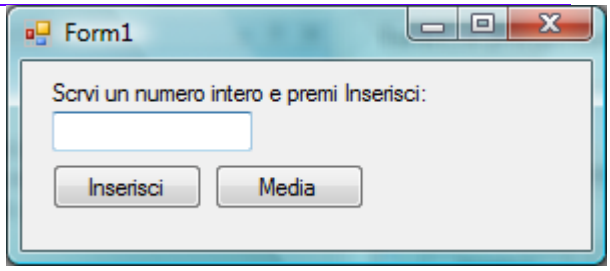
ESERCIZIO 5. CALCOLI

- Scrivere un programma che calcola la media dei due valori numerici scritti nelle caselle di testo. La media sarà scritta nel titolo della finestra.
- Il pulsante `Aumenta` invece incrementa di +2 il valore scritto nella prima casella e diminuisce di –3 il valore scritto nella seconda casella.
- Decidere se il programma lavora su numeri interi o su numeri decimali.



ESERCIZIO 6. CONTATORE E ACCUMULATORE

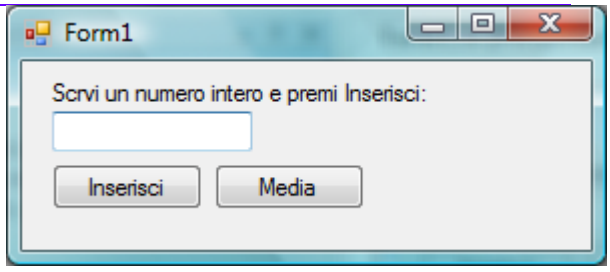
- Scrivere un programma per calcolare la media dei numeri interi in una casella di testo.
- Il pulsante `Inserisci` somma i valori inseriti nella variabile accumulatore e incrementa un conteggio di quanti valori sono stati inseriti.
- Il pulsante `Media` mostra nel titolo della finestra la somma dei valori inseriti.



- Per es. se si inserisce 4 6 e 11 dovrebbe mostrare il valore 21.

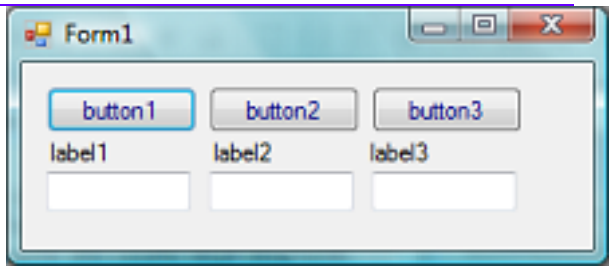
ESERCIZIO 7. ACCUMULATORE

- Modifica il programma precedente in modo che lavori sui numeri con la virgola.
- Il pulsante `Media` deve mostrare nel titolo della finestra la media dei valori inseriti.
- Per es. se si inserisce 10 11 14 e 16 dovrebbe mostrare il valore 12.75



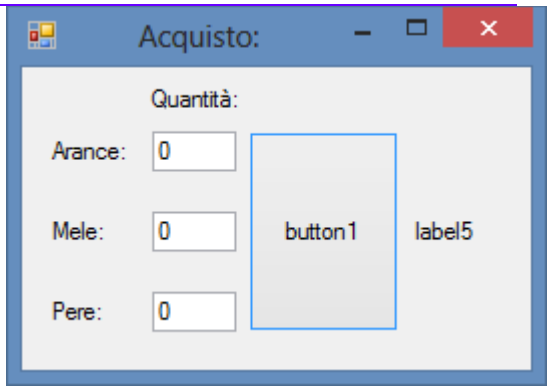
ESERCIZIO 8. RIEPILOGO

- Preparare un nuovo progetto
- Il pulsante button1 accumula nella label1 i valori scritti nella textBox1
- Il pulsante button2 scrive nella label2 la media dei valori scritti nella textBox2
- Il pulsante button3 scambia il valore scritto nella textBox3 con la proprietà Width della finestra



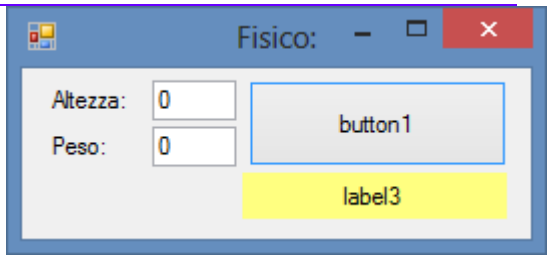
ESERCIZIO 9. PROGRAMMA COMMERCIALE

- Preparare un nuovo progetto per un programma di acquisto di frutta
- Le arance sono vendute a 0,55 euro / Kg.
  Le mele sono vendute a 0,45 euro / Kg.
  Le pere sono vendute a 0,85 euro / Kg.
- Il pulsante acquisisce le quantità acquistate dalle rispettive textBox e poi calcola l'importo totale da pagare e lo mostra nella etichetta label5.



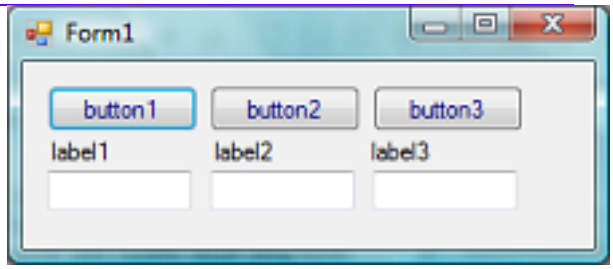
ESERCIZIO 10. PESO CORPOREO

- Preparare un nuovo progetto per un programma di controllo del peso corporeo
- Il pulsante acquisisce altezza e peso di una persona e calcola il valore della seguente formula `PC = Peso / ( Altezza * Altezza);`
- Il peso va espresso in Kg, mentre l'altezza in cm.
- Il risultato va mostrato nella etichetta label3.



ESERCIZIO 11. CONTROLLI

- Preparare un nuovo progetto per un programma di controllo di dati
- Il pulsante button1 acquisisce tre valori dalle tre caselle di testo e rende visibile label1 solo se sono tutte uguali
- Il pulsante button2 acquisisce tre valori dalle tre caselle di testo e rende visibile label2 se almeno due sono uguali
- Il pulsante button3 acquisisce tre valori dalle tre caselle di testo e rende visibile label1 solo se sono tutti diversi



SOMMARIO