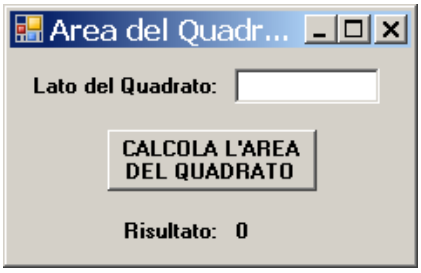
VARIABILI, ASSEGNAZIONE, INPUT / OUTPUT – Linguaggio C#

Le Variabili

☞ Nell'esporre questo argomento, useremo come esempio la realizzazione di un Programma C# che risolve il seguente problema: *dato il Lato di un Quadrato, ne calcoli l'Area.*



☞ Anzitutto *definisci la Form*, che potrebbe essere come quella nella figura accanto. La *TextBox* si chiama ***txtLato***, il *Button* si chiama ***plsCalcola***. Il risultato viene visualizzato in una *Label* (inizialmente contenente 0) di nome ***lblArea***.

☞ Dalla *Finestra Proprieta* del pulsante *plsCalcola*, accedi alla *Lista degli Eventi* e genera il *Sottoprogramma-Evento* "*plsCalcola_click*", in modo da far apparire il blocco di codice vuoto, pronto per inserirvi le istruzioni che saranno eseguite quando l'utente "farà click sul pulsante".

Le **Variabili** sono strumenti del linguaggio che consentono di memorizzare dati nella memoria RAM ed elaborarli.

Prima di utilizzare una Variabile, è necessario **Dichiarare la Variabile** specificandone il **Nome** e il **Tipo.**

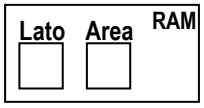
Regola per scrivere la **Dichiarazione di Variabile**: *<tipo> < nome>*

✋ Il *Nome* identifica la Variabile *(identificatore)*; il *Tipo* specifica la *natura del dato* che la Variabile dovrà memorizzare.

Il **Tipo int** specifica che una Variabile può contenere solo **Numeri Interi** (positivi e negativi). Il **Tipo double**, invece, specifica che una Variabile può contenere **Numeri Reali** (cioè sia numeri interi che numeri con cifre decimali).

☞ Nel programma utilizzi due *Variabili*, che chiameremo **Lato** e **Area** entrambe destinate a contenere numeri reali. Le dichiarazioni delle due variabili nel programma devono quindi essere scritte così, nel codice dell'evento:

```
private void plsCalcola_Click(object sender, EventArgs e)
{
        double Lato;
        double Area;
        ...
```



Una Variabile, dichiarata nel codice, viene creata quando si Esegue il Programma: per gestirla, C# **riserva una piccola porzione di memoria RAM (Allocazione)** entro la quale "memorizzerà" il dato.

Durante l'esecuzione, se nel codice è riportato il nome di una variabile, C# ne considera *il valore in essa memorizzato*.

L'Istruzione di Assegnazione e le Espressioni di Calcolo

L'**Istruzione di Assegnazione** consente di memorizzare un dato in una Variabile.

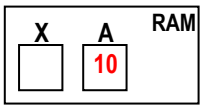
Regola per scrivere l'**Istruzione di Assegnazione**: *<nome-variabile>* **=** *<espressione>*

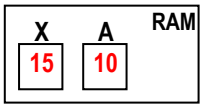
Una **Espressione** è un "calcolo" che produce come risultato un dato. Un'*Espressione* può contenere valori, variabili, operatori, funzioni, parentesi, proprio come nelle normali espressioni matematiche.

☞ Esempi di *Istruzione di Assegnazione* potrebbero essere queste:

Delta = B*B – 4*A*C **Area = B*H** **Media = (X1 + X2) / 2**

L'**Esecuzione di una Istruzione di Assegnazione** provoca il **Calcolo dell'Espressione** (*a destra dell'uguale*) e la **Memorizzazione del Risultato nella Variabile** (indicata *a sinistra dell'uguale*).





☞ L'esecuzione dell'*Istruzione di Assegnazione* **X = A + 5**, provoca anzitutto il *calcolo dell'Espressione presente a destra dell'uguale:* nel nostro caso, se nella variabile A è memorizzato il numero 10, allora il risultato dell'Espressione è 10+5 ossia 15. Quindi, il risultato ottenuto viene *memorizzato nella Variabile indicata a sinistra dell'uguale*, ossia in X. In figura vedi la situazione della RAM *prima* e *dopo* l'esecuzione dell'Istruzione di Assegnazione.

In particolare, una *Espressione*, può essere anche un **singolo valore** o una **singola variabile**.

☞ Esempi di Assegnazioni con Espressioni così semplici, sono: **A = 10** **B = C**

Il linguaggio C# è molto rigido nelle assegnazioni: **esige che il dato da assegnare sia dello stesso tipo della variabile a cui viene assegnato**.

☞ Se K è una variabile di *tipo int*, l'assegnazione *K = 3.56* genera un errore perché il valore 3.56 è di tipo double. Analogamente, l'assegnazione *K = "ciao"* genera un errore perché "ciao" è un testo, non un numero intero.

Le funzioni **Convert.ToDouble**, **Convert.ToInt32**, **Convert.ToString** permettono di effettuare delle "**conversioni di tipo**" in modo da "forzare" le assegnazioni anche quando i tipi sono diversi.

✋ Ad esempio, se N è una variabile di *tipo int* e X una di *tipo double*, è possibile la seguente assegnazione:

N = Convert.ToInt32 (X) … prende il valore di X, lo converte in intero e lo assegna ad N

Analogamente *Convert.ToDouble* converte in numero reale e *Convert.ToString* converte in testo.

Nota bene: le *funzioni Convert* "forzano" le conversioni. La conversione viene tentata da C#, ma *non è detto che sia possibile*: dipende dal dato da convertire. La conversione potrebbe riuscire oppure, se impossibile, generare un errore.

Uso delle Proprietà degli Oggetti all'interno del Codice C#

La notazione ***oggetto.proprietà*** consente di fare riferimento alla *Proprietà di un Oggetto* all'interno del Codice. Le Proprietà, se usate nel Codice con *oggetto.proprietà*, sono considerate da C# **come se fossero delle Variabili**.

☞ Quindi è possibile **"*assegnare*" qualcosa a una Proprietà**, oppure **usare una proprietà in una espressione**:

lblSaluto.Text = "ciao" … visualizza il testo "ciao" nella Label di nome lblSaluto

✋ E' anche possibile "*assegnare un colore*" alle proprietà *ForeColor* o *BackColor* usando come espressione la particolare notazione **Color**.*colore*, come segue:

txtLato.BackColor = Color.Red … la TextBox di nome txtLato diventa con sfondo Rosso
plsCalcola.ForeColor = Color.Green … la scritta sul Button di nome plsCalcola diventa Verde

Nota importante: la *notazione Color* viene riconosciuta dal linguaggio C# solo se, all'inizio del codice, è presente la clausola "**using System.Drawing;**"

La **clausola *using***, specificata all'inizio del codice C#, attiva un **namespace** (*spazio dei nomi*), cioè attiva una delle *Librerie di Oggetti* del linguaggio. Il **namespace System.Drawing** è quello che contiene l'oggetto *Color*.

✋ Tutto ciò che è stato presentato finora, viene riconosciuto dal linguaggio C# con l'attivazione di due sole "librerie": il namespace **System** e il namespace **System.Windows.Forms**

Come effettuare degli Input da TextBox e degli Output su Label

L'Utente, in Fase di Esecuzione, *inserisce il Dato in una TextBox* e, così facendo, effettua un *Input*. E' quindi necessario **prelevare il Dato dalla TextBox e memorizzarlo in una Variabile** per le successive elaborazioni.

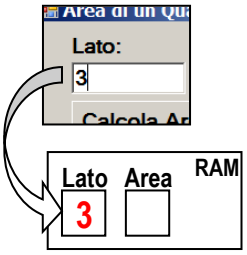
✋ Ricordando che la *proprietà Text* di una TextBox ha, come valore, il testo digitato in essa, sembrerebbe facile recuperare il dato e porlo in una variabile Lato, con la semplice assegnazione **Lato = txtDato.Text.** Purtroppo, questa istruzione genera una errore perché la proprietà Text è un testo, mentre Lato è una variabile di tipo float. E' necessaria una conversione…

Per **Recuperare un Dato Numerico digitato in una TextBox e memorizzarlo in una Variabile (INPUT)** è necessario: (a) utilizzare la *proprietà Text* della TextBox, il cui valore è il testo digitato dall'utente; (b) convertire tale testo in numero con la *funzione Convert* più adatta; (c) *assegnare tutto alla variabile* in cui si desidera memorizzare il dato.

Tutto ciò si realizza con la seguente istruzione:

<nome-variabile> **= Convert.ToDouble (** *<nome-TextBox>***.Text)** **… se il dato è un numero reale**
<nome-variabile> **= Convert.ToInt32 (** *<nome-TextBox>***.Text)** **… se il dato è un numero intero**

☞ Quando l'utente usa il nostro programma, anzitutto inserisce la misura del Lato nella TextBox di nome *txtLato*, quindi "fa click" sul pulsante *plsCalcola* e provoca così l'esecuzione del sottoprogramma-evento "*plsCalcola_click*".

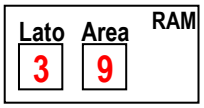


☞ Di conseguenza, nel codice del sottoprogramma-evento "*plsCalcola_click*" (subito dopo le dichiarazioni), devi prelevare il dato dalla TextBox e memorizzarlo nella variabile Lato, come spiegato:

```
double Lato;
double Area;
Lato = Convert.ToDouble ( txtLato.Text )
```

☞ Poi usi un'altra Istruzione di Assegnazione per calcolare l'Area:

```
Area = Lato * Lato
```



Quando il Risultato viene calcolato e memorizzato in una Variabile, si pone il problema di "renderlo visibile" all'Utente, ossia di **effettuare l'*Output***. E' quindi necessario **prelevare il Dato dalla Variabile e porlo nella Label** per visualizzarlo sulla Form.

☞ Anche qui, come per l'input, non basta la semplice assegnazione **lblArea.Text = Area**: anch'essa genera una errore perché la proprietà Text è un testo, mentre Area è una variabile di tipo float. E' di nuovo necessaria una conversione…

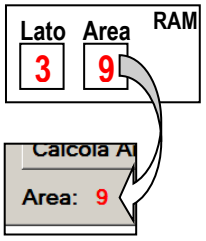
Per **Visualizzare in una Label il Risultato memorizzato in una Variabile (OUTPUT)** è necessario: (a) individuare la variabile in cui è stato memorizzato il risultato; (b) convertire il valore di questa variabile in testo con la *funzione Convert.toString*; (c) *assegnare tutto alla proprietà Text* della Label.

Quindi, l'istruzione da usare per l'Output è:

*<nome-Label>***.Text = Convert.ToString (** *<nome-variabile>* **)**

☞ Di conseguenza, per rendere visibile il risultato nel nostro programma, che si trova nella *variabile Area*, scrivi:

```
private void plsCalcola_Click(object sender, EventArgs e)
{
        double Lato;
        double Area;
        Lato = Convert.ToDouble ( txtLato.Text )
        Area = Lato * Lato
        lblArea.Text = Convert.ToString ( Area )
}
```



Ora il nostro *sottoprogramma-evento* "*plsCalcola_click*" è completo ed è possibile eseguire l'intero progetto per verificare se funziona correttamente. Non dimenticare che il codice, *completo della parte generata automaticamente da C#,* è:

```
using System;
using System.Windows.Forms;
namespace AreaDiUnQuadrato
{
        public partial class Form1 : Form
        {
                public Form1()
                {
                        InitializeComponent();
                }
                private void plsCalcola_Click(object sender, EventArgs e)
                {
                        … vedi sopra …
                }
        }
}
```