Visual Basic

Introduzione al linguaggio e panoramica sulle basi della programmazione

1. Scopri quanto è facile programmare

Iniziamo a conoscere Visual Basic

1. I tipi di dati
2. le strutture iterative
3. Casella degli strumenti e la finestra delle proprietà
4. Gli eventi fondamentali del mouse
5. Gli eventi fondamentali della tastiera
6. Il form, l'elemento centrale di un'applicazione
7. Le procedure e le funzioni

I controlli di Visual Basic

8. CommandButton, TextBox e Label
9. I controlli Frame, CheckBox e OptionButton
10. I controlli ComboBox e ListBox
11. I controlli ImageBox e PictureBox

Visual Basic

Le caratteristiche che fanno di Visual Basic un linguaggio di programmazione estremamente versatile e facile da usare sono due:

1. **le funzioni di progettazione dell'interfaccia completamente visuali;**
   All'avvio di VB, nell'area centrale, si può osservare una finestra, il **form**, che rappresenta la nostra applicazione. Per inserire elementi all'interno del form (i controlli), quali pulsanti, caselle di testo, etichette, è sufficiente selezionarli all'interno della Casella degli strumenti e trascinarli sul form stesso: il controllo selezionato verrà posizionato nel punto esatto che si è deciso. Altrettanto facilmente è possibile modificare la posizione e la dimensione di un controllo semplicemente utilizzando il mouse.
1. **il linguaggio di tipo event-driven. L'ambiente di sviluppo visuale consente di essere produttivi fin da subito.**
   L'altra caratteristica di Visual Basic è quella di essere un linguaggio **event-driven**. Con questo termine si intende che l'elemento che sta alla base del linguaggio è l'evento, cioè, più in generale, l'azione: un evento è il clic dell'utente su un pulsante, la digitazione in una casella di testo, la selezione di un comando di menu. Gli oggetti inseriti in un form Visual Basic sono in grado di riconoscere in automatico gli eventi più comuni, senza bisogno che il programmatore si preoccupi, ad esempio, di stabilire quando l'utente fa clic su un pulsante, seleziona un elemento da una lista, ecc. Grazie a queste peculiarità, Visual Basic è un linguaggio di programmazione facile da usare ma, nello stesso tempo, potente e flessibile.

Visual Basic

Nella trattazione che seguirà ci riferiamo a Visual Basic 2010. Esistono versioni più recenti che vanno acquistate mentre, quella proposta e ottenibile senza costi.
Lo scopo che ci poniamo è indipendente dalla versione utilizzata in quanto le caratteristiche del linguaggio non cambiano (mi riferisco principalmente alla sintassi, agli eventi, agli oggetti, ecc.

Il pacchetto **Visual Basic 2010 Exspress Edition** si può ottenere digitando dalla barra degli indirizzi del vostro browser l'indirizzo di seguito riportato:

microsoft-visual-basic.forumer.it

Visual Basic

dichiarazione delle costanti

Const *Nome* [As *Tipo*] = valore

Const è una parola chiave riservata di VB che si usa per definire una costante. ***Nome*** è il nome che si attribuire alla costante. Nella scelta dei nomi (sia delle costanti, delle variabili, ma anche delle procedure, delle funzioni e dei controlli, che vedremo più avanti), è necessario seguire alcune regole. I nomi non devono essere più lunghi di 40 caratteri, non posso iniziare con un numero né contenere spazi e caratteri come ?, !, :, ;, . e ,.

As è un parametro opzionale che indica il ***Tipo*** di dato contenuto nella costante; se non viene specificato, il compilatore lo determinerà sulla base del valore assegnato alla costante stessa.

valore è il valore vero e proprio della costante.

Visual Basic, al contrario di altri linguaggi come il C o Java, **non fa differenza tra maiuscole e minuscole** (non è case sensitive).

Esempi:

```
Const PI = 3.14
Const nome As String = "Filippo"
```

Visual Basic

dichiarazione delle variabili

Dim *Nome* [As *Tipo*]

Nella dichiarazione delle variabili si usa la parola chiave ***Dim*** per indicare al compilatore che quella che si sta per definire è una variabile. Anche per le variabili il parametro ***As*** è opzionale: se non viene specificato, la variabile verrà dichiarata di tipo ***Variant***, un particolare tipo che può contenere dati di tutti i tipi. È sconsigliabile definire variabili di tipo ***Variant***, se non espressamente necessario. Questo tipo di dato occupa molta memoria.

Esempi:

Dim Utenti As Integer

Dim Nome As String, Cognome As String

Riepilogo dei principali tipi di dato

Integer	da -32.768 a 32.767
Single	da -3.402823E38 a -1.401298E45 (4 byte)
Double	da -4.94065645841247E-324 a 4.94065645841247E-324
Boolean	True o False
String	Da 0 a, circa, 2 miliardi

Visual Basic

casella degli strumenti

Label (etichetta)

Consente di visualizzare sul form un breve testo. Viene usato per inserire un'intestazione generale (descrizione dell'applicazione) oppure la descrizione delle funzioni di un altro controllo. Il testo visualizzato da un'etichetta non può essere modificato in fase di runtime.

TextBox (casella di testo)

Viene utilizzata per inserire o visualizzare dati numerici o testuali. La TextBox utilizza solo dati testuali e per questo motivo che ci avvaliamo di alcune funzioni, di seguito descritte, per la trasformazioni di dati testuali in valori numerici e viceversa.

CommandButton (pulsante di comando)

Viene utilizzato per avviare, interrompere o concludere un'elaborazione. Solitamente attivato con un clic del mouse.

Visual Basic

Oggetto TextBox

Applicazione delle funzioni Val e CStr all'oggetto TextBox

Esempio TextBox (casella di testo)

nome = Txt_nome.Text
La variabile **nome** (di tipo stringa) assume il valore digitato nella TextBox di nome **Txt_nome.Text**

n = Val (Txt_n.Text)
La variabile **n** (di tipo numerico) assume il valore digitato nella TextBox di nome **Txt_n.Text** trasformato (dalla funzione **Val**) in numero.

Txt_n.Text = CStr (n)
La variabile **n** (di tipo numerico) è trasformata in una stringa (dalla funzione **CStr**) per poter essere visualizzata nella casella nella TextBox di nome **Txt_n.Text.**

Visual Basic

Proprietà TextBox

Proprietà Enable e Focus

Enable (True/False) permette di attivare/disattivare la scrittura in una casella TextBox

Txt_Nome.Enabled = False
Questa istruzione inibisce la possibilità di scrivere nella casella di testoTxt_Nome.Text.
Quando viene utilizzata una casella di testo questa proprietà, per default, è settata con il valore True.

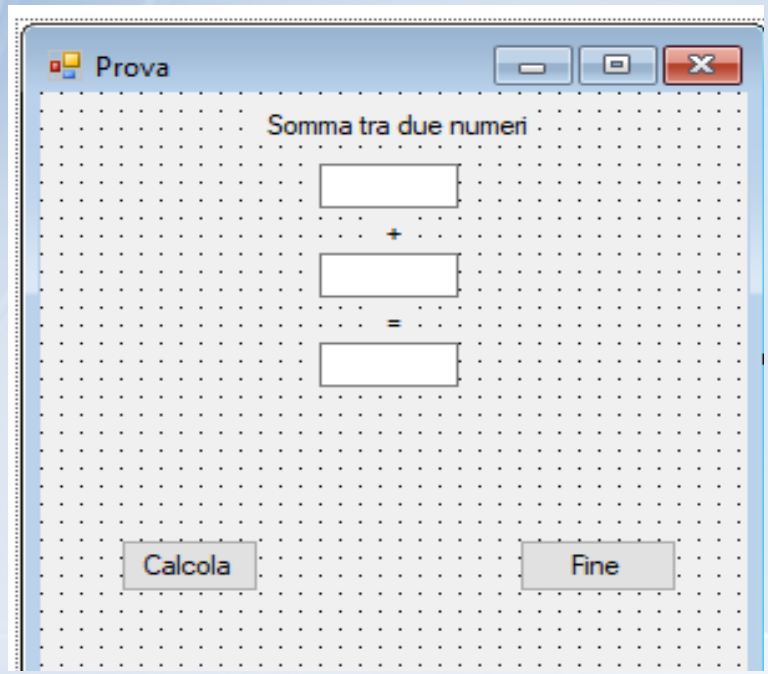
Focus posiziona il cursore in una casella TextBox specificata

Txt_Nome.Focus( )
Questa istruzione posizione il cursore nella casella di testo Txt_Nome.Text.

Visual Basic

Applicazione delle funzioni Val e CStr all'oggetto TextBox, un primo esempio: somma tra due numeri interi

IL FORM



Visual Basic

Applicazione delle funzioni Val e CStr all'oggetto TextBox, un primo esempio

IL CODICE

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    End Sub
    Private Sub Btn_End_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btn_End.Click
        End
    End Sub
    Private Sub Btn_Calcola_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btn_Calcola.Click
        Dim a As Integer
        Dim b As Integer
        Dim totale As Integer
        a = Val(Text_a.Text)
        b = Val(Text_b.Text)
        totale = a + b
        ' visualizza risultato con TextBox
        Text_totale.Text = CStr(totale)
    End Sub
End Class
```

Visual Basic

Operatori di confronto e connettivi logici

Di seguito sono riportati gli operatori di confronto ed i connettivi logici utilizzati da Visual Basic

OPERATORI DI CONFRONTO

= < > <= >= <>

CONNETTIVI LOGICI

And Or Not

Visual Basic

Struttura di selezione (unaria)

La **pseudocodifica** di una struttura di selezione unaria ha la seguente forma:

```
...
SE <condizione>
   ALLORA
      <istruzioni>
FINE SE
...
```

Nel **linguaggio Visual Basic** la struttura di selezione sarà tradotta come segue:

```
...
If <condizione> Then
   <istruzioni>
End If
...
```

Visual Basic

Struttura di selezione (binaria)

La **pseudocodifica** di una struttura di selezione completa (o binaria) ha la seguente forma:

```
...
SE <condizione>
   ALLORA
      <istruzioni>
   ALTRIMENTI
      <istruzioni>
FINE SE
...
```

Nel **linguaggio Visual Basic** la struttura di selezione sarà tradotta come segue:

```
...
If <condizione> Then
   <istruzioni>
Else
   <istruzioni>
End If
...
```

Visual Basic

Selezione multipla Select Case

La **pseudocodifica** di una struttura di selezione multipla ha la seguente forma:

```
...
NEL CASO CHE <var>|<espressione> SIA
   < valore_1> : <istruzioni>
   < valore_2> : <istruzioni>
    .....
    < valore_n> : <istruzioni>
   [ALTRIMENTI : <istruzioni>]
FINE CASO
...
```

Nel **linguaggio Visual Basic** la struttura di selezione multipla sarà tradotta come segue:

```
...
Select Case <var>|<espressione>
   Case <valore_1>
      <istruzioni>
   .....
   Case <valore_n>
      <istruzioni>
   [Case Else
      <istruzioni>]
End Select
...
```

Visual Basic

Iterazioni PRE-condizionali Do - While

La **pseudocodifica** di una struttura di iterazione precondizionale MENTRE ha la seguente forma:

```
...
MENTRE <condizione> ESEGUI
    <istruzioni>
FINE MENTRE
...
```

Nel **linguaggio Visual Basic** la struttura di iterazione precondizionale sarà tradotta come segue:

```
...
Do While <condizione>
    <istruzioni>
Loop
...
```

Il processo di iterazione:
- avviene per condizione **VERA**
- si arresta quando la condizione iniziale diventa **FALSA**.

Visual Basic

Iterazioni POST-condizionali Do – Loop Until

La **pseudocodifica** di una struttura di iterazione precondizionale **FINCHE'** ha la seguente forma:

```
...
RIPETI
    <istruzioni>
FINCHE' <condizione>
...
```

Nel **linguaggio Visual Basic** la struttura di iterazione precondizionale sarà tradotta come segue:

```
...
Do
    <istruzioni>
Loop Until <condizione>
...
```

Il processo di iterazione:
- avviene per condizione **FALSA**
- si arresta quando la condizione iniziale diventa **VERA**.

.

Visual Basic

Iterazioni PRE-condizionali Do Until - Loop

La **pseudocodifica** di una **struttura di iterazione precondizionale FINCHE'** che cicli per **FALSO** ed esca per **VERO** non è presente nella nostra pseudocodifica

Nel **linguaggio Visual Basic** invece esiste **ANCHE** la seguente struttura di iterazione

```
...
Do Until <condizione>
   <istruzioni>
Loop
...
```

Il processo di iterazione:
- avviene per condizione **FALSA**
- si arresta quando la condizione iniziale diventa **VERA**.

.

Visual Basic

Iterazioni enumerativa For

La **pseudocodifica** di una struttura di iterazione enumerativa **PER** ha la seguente forma:

```
...
PER <indice> ← <inizio>  [INDIETRO] A <fine> ESEGUI
    <istruzioni>
    <indice> ← <indice> + 1  [ <indice> ← <indice> -1]
FINE PER
...
```

Nel **linguaggio Visual Basic** la struttura di iterazione enumerativa sarà tradotta come segue:

```
...
For <indice> = <inizio> To <fine> [Step incremento]
   <istruzioni>
Next  <indice>
...
```

La costante numerica **incremento** può contenere **valori INTERI sia positivi che negativi**.

Visual Basic

Funzioni di carattere generale

IsNumeric <espressione>

Restituisce un valore Boolean che indica se un'espressione può essere valutata come un numero.

Esempio di utilizzo della funzione IsNumeric( )

```
Dim n As Integer
 ...
 If (IsNumeric (Txt_n.Text)) Then
    n = Val (Txt_n.Text)
 Else
    MsgBox ("n - inserire un valore numerico")
 End If
...
```

Visual Basic

Message Box

IsNumeric <espressione>

La funzione **MsgBox()** ci permette di produrre una finestra di dialogo che dà la possibilità all'utente di interagire con la nostra applicazione come in quest'esempio:

Nella finestra "Esempio MsgBox()" è visualizzato un messaggio che chiede all'utente se vuole continuare con la stampa del documento mettendo a disposizione due pulsanti attivi OK o Annulla che danno la possibilità all'utente di continuare o interrompere il processo in corso.
Il codice che ci ha permesso di ottenere questa finestra è il seguente:

```
Dim risp As Integer
risp = MsgBox("Vuoi stampare il documento?", 1, "Esempio MsgBox")
```

dove **risp** è una variabile integer che coglie il valore della risposta dell'utente alla finestra.
La variabile **risp** può essere omessa nel caso non necessita rilevare il valore della risposta.

Visual Basic

Message Box

Come abbiamo visto nell'esempio MsgBox() ha all'interno delle parentesi diversi argomenti che servono per dare la forma alla nostra finestra di dialogo.

MsgBox (prompt [,button] [,title] [helpfile, context])

prompt : indica il messaggio che sarà visualizzato nelle finestra di dialogo. E' l'unico argomento obbligatorio per la funzione e va scritto fra virgolette

buttons : indica il valore numerico dei pulsanti da visualizzare nel nella finestra di dialogo. La tabella seguente indica i valori da attribuire a buttons affinché vengano visualizzati i pulsanti:

Valore numerico	Pulsante visualizzato
0	Pulsante OK
1	Pulsanti OK e Annulla
2	Pulsanti Termina, Riprova e Ignora
3	Pulsanti Sì, No, Annulla
4	Pulsanti Sì e No
5	Pulsanti Riprova e Annulla

Visual Basic

Message Box

Ognuno di questi pulsanti ha un valore numerico che ci permette di richiamarli in eventuali routine che controllano il programma. Questi valori sono elencati nella seguente tabella:

Valore	Pulsante cliccato dall'utente
1	OK
2	Annulla
3	Termina
4	Riprova
5	Ignora
6	Sì
7	No

Visual Basic

Message Box

title : Indica il titolo della finestra di dialogo e va scritto fra virgolette

helpfile e context : sono relativi alla guida dell'applicazione quindi ancora non li tratteremo per evitare confusione anche perché non sono indispensabili per la funzionalità delle nostra finestra di dialogo.

Visual Basic

Message Box

Sommando anche il valore di visualizzazione delle icone a quello dei buttons può essere inserita anche un'icona nella finestra di dialogo così:

I valori da inserire sono elencati nella seguente tabella:

Valore	Icona
16	
32	
48	
64	

Per sommare i valori da attribuire a **buttons** basta scriverli separati da una **+**, in modo da poterli visualizzare sullo schermo insieme come nell'immagine dell'esempio precedente, nella quale vi erano presenti sia i pulsanti di comando che un'icona.

Visual Basic

Funzioni di carattere generale

Sqr <numero>

Restituisce un valore Double che specifica la radice quadrata di un numero.

Esempio di utilizzo funzione Sqr( )

```
Dim n As Integer
Dim r As Double
 ...
 r = Sqrt (numero)
...
```

Visual Basic

I vettori (array monodimensionale)

L'istruzione di DICHIARAZIONE per definire un vettore ha il seguente formato:

Dim nomevettore (dimensione) As Integer

Dove nomevettore è il nome collettivo dei componenti del vettore, dimensione indica il massimo numero di valori che il vettore può contenere.

Esempio di dichiarazione di vettori

```
Const N 5
Dim vettore (N) As Integer

Dim vettore (5) As Double
```

Gli indici, del vettore in esempio, assumeranno i valori da "0" a "4".

Visual Basic

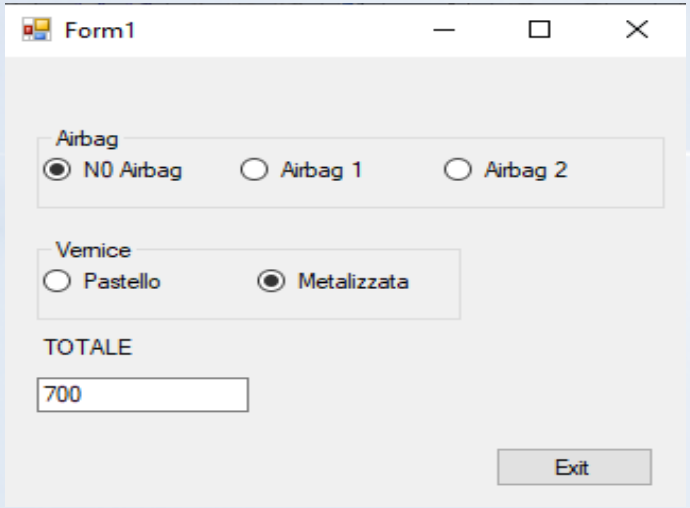
RadioButton

I controlli RadioButton sono progettati per fornire agli utenti una scelta tra due o più impostazioni, di cui solo una può essere assegnata a una procedura o a un oggetto. È pertanto possibile selezionare un solo **RadioButton** alla volta, anche se fa parte di un gruppo funzionale.

Per raggruppare i pulsanti di opzione, è possibile disegnarli all'interno di un contenitore, ad esempio un controllo **GroupBox.** Tutti i pulsanti di opzione aggiunti direttamente a un modulo diventano un gruppo. Per aggiungere gruppi distinti, è necessario inserirli nei pannelli o nelle caselle di gruppo.

Nell'esempio, di seguito riportato, sono stati inseriti due gruppi di opzioni (**GroupBox**), **Group 1** e **Group 2,** contenenti ognuno una possibile scelta su tre opzioni.

RadioButton restituisce il valore "True o False" in funzione del suo stato.



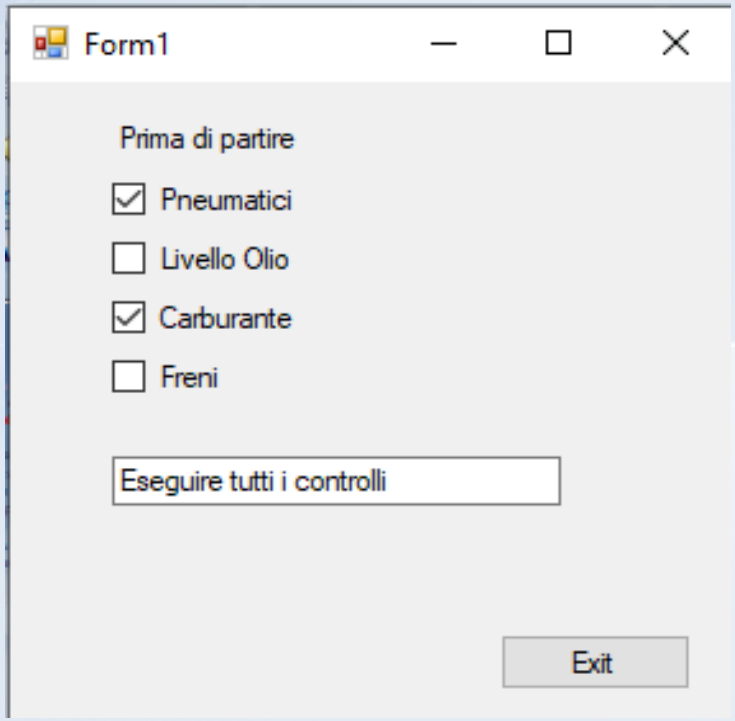
Visual Basic

CheckBox

I controlli CheckBox e RadioButton hanno una funzione simile: consentono all'utente di scegliere da un elenco di opzioni. **CheckBox** consente all'utente di scegliere una combinazione di opzioni. Al contrario, i controlli RadioButton consentono a un utente di scegliere tra opzioni che si escludono a vicenda.

Nell'esempio, di seguito riportato, la possibilità di scelta (una o più) tra quattro opzioni.

CheckBox restituisce il valore "True o False" in funzione del suo stato.



Visual Basic

I SOTTOPROGRAMMI: **Funzioni e procedure (Function e Sub)**

I **SOTTOPROGRAMMI** in informatica sono dei blocchi di codice che permettono di eseguire una serie di istruzioni per risolvere una parte del programma (sottoalgoritmo) quando vengono attraverso un'apposita istruzione.

Anche nel **linguaggio Visual Basic** abbiamo:

- le **procedure** (**Sub**) **sono dei sottoprogrammi** che eseguono le istruzioni al loro interno interfacciandosi con il programma chiamante tramite i **parametri;**

- le **funzioni (Function): sono dei sottoprogrammi** che. oltre ad eseguire le istruzioni al loro interno ed interfacciarsi con il programma chiamante tramite i **parametri,** hanno la capacità di restituire un valore di ritorno attraverso il proprio nome.

Visual Basic

Sub

Vediamo come si dichiara e si richiama un **sottoprogramma** di tipo **procedura** o **Sub**:

```
Private Sub NomeProcedura (Param1 As Tipo, Param2 As Tipo)
        Dichiarazioni locali ed istruzioni
End Sub
```

L'istruzione che attiva (chiama) la procedura nel programma chiamante ha la seguente sintassi:

```
NomeProcedura (Valore1, Valore2)
```

Per l'insieme dei dati *Valore1*, *Valore2* possono essere specificate sia variabili che costanti, e che vi deve essere esatta corrispondenza rispetto all'insieme *Param1*, *Param2* sia per quanto riguarda il tipo di dati, sia per quanto riguarda l'ordine in cui sono specificati.

Visual Basic

Function

Vediamo come si dichiara e si richiama un **sottoprogramma** di tipo **funzione** o **Function**:

Function <NomeFunction> (Param1 As Tipo, Param2 As Tipo) As Tipo
Dichiarazioni locali ed istruzioni
NomeFunction = espressione
End Function

L'istruzione che attiva (chiama) la funzione nel programma chiamante ha la seguente sintassi:

Var = NomeFunction (Valore1, Valore2)

OVVIAMENTE la variabile ***Var*** deve essere dello stesso tipo di **<NomeFunction>.**

Per l'insieme dei dati *Valore1*, *Valore2* possono essere specificate sia variabili che costanti, e che vi deve essere esatta corrispondenza rispetto all'insieme *Param1*, *Param2* sia per quanto riguarda il tipo di dati, sia per quanto riguarda l'ordine in cui sono specificati.

Visual Basic

Passaggio dei parametri

Esistono due modi per passare una variabile a una procedura (funzione):

Per **valore**
Per **reference (o riferimento o indirizzo)**

Quando si effettua il passaggio per **valore**, il parametro **ATTUALE** presente nell'istruzione che richiama il sottoprogramma conserva il suo valore anche se, nel sottoprogramma chiamato (Function o Sub) , il valore del parametro **FORMALE** corrispondente si modifica

Quando si effettua il passaggio per **reference** (o **riferimento** o **indirizzo**), il parametro **ATTUALE** presente nell'istruzione che richiama il sottoprogramma, NON conserva il suo valore essendo stato passato direttamente (attraverso il suo indirizzo) al sottoprogramma chiamato nel corrispondente parametro FORMALE.

Le due modalità di passaggio vengono così indicate:

- **ByVal**
- **ByRef**

Esempio:

```
Sub Calcola (ByVal x As Single, ByRef  y As Double)
```

Visual Basic

Operazioni sui FORM

Un progetto può contenere più **FORM** con la possibilità di passare il controllo su uno di essi.

<NomeForm1>.Show() passa il controllo al form <**NomeForm1>**

<NomeForm1>.Hide() Nasconde il form **<NomeForm1>,** che resta in esecuzione. Non utilizzando questa funzione il Form richiamato viene sovrapposto al **<NomeForm1>**

<NomeForm1>.Close() Termina l'esecuzione del form **<NomeForm1>**

N.B. il nome del Form può essere sostituito con **Me,** in questo caso l'azione richiesta viene effettuata sul Form stesso.

Esempio: supponiamo di voler passare da un Form **PRIMO** ad un Form **SECONDO** e viceversa, utilizzando un **CommandButton (Btn_Secondo):**

FORM PRIMO

```
Private Sub Btn_Secondo_Click(sender As System.Object, e As System.EventArgs) Handles Btn_Secondo.Click
    Secondo.Show()
    Me.Hide()
End Sub
```