




Corso base - Lezione 6

http://www.vbsimple.net/base/bas_06.htm

- [Estendiamo il nostro progetto.](#)
- [Definizione di nuovi tipi di dati.](#)
- [Le costanti.](#)
- [L'indentatura.](#)


Abbiamo visto nella lezione precedente come dichiarare una variabile semplice e sfruttarla per effettuare i nostri calcoli.

Riapriamo il nostro Progetto1 e portiamoci nella finestra del codice  e rivediamo il codice che abbiamo visto la volta precedente.

```
1. Dim Numerol As Integer
2.
3. Private Sub Form_Click()
4.     Numerol = Numerol + 1
5.     Label1.Caption = Numerol
6. End Sub
```

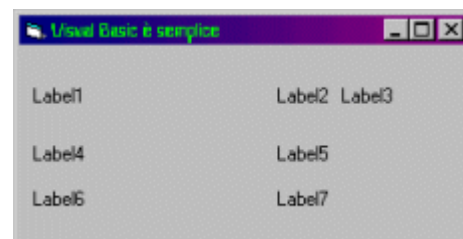
Abbiamo nel nostro codice due aree separate: l'area delle dichiarazioni e l'area della routine legata all'evento Click dell'oggetto Form.

Continuando l'esempio nostro, facciamo in modo che ad ogni click sul form, venga mostrato il numero dei click complessivi e l'orario dell'ultimo e del penultimo click.

Torniamo per un attimo al nostro form. Apriamo il menu Visualizza, diamo un click sulla voce Oggetto  e torneremo alla finestra di progettazione.

Abbiamo già una Label, di nome Label1. Clicchiamo una volta sopra d'essa e verrà evidenziata; una volta fatto, apriamo il menu Modifica e clicchiamo sulla voce Elimina. Così facendo la elimineremo definitivamente ed avremo il form completamente libero.

Ora inseriamo un paio di Labels nel modo visto nelle lezioni precedenti e disponiamole nella maniera visibile qui a lato.



Per ogni Label impostiamo la proprietà **Autosize** a True, in modo che venga calcolata automaticamente la dimensione necessaria per il testo che andremo a scrivere dentro. È necessario specificare il valore della proprietà Autosize per tutte le label, una per una; non basta farne una sola.

Ora impostiamo il testo contenuto in ogni Label.

Ricordiamo che per impostare il testo di una Label bisogna innanzitutto scegliere la Label da modificare cliccandoci sopra una volta e poi modificare il contenuto della proprietà

Caption dalla finestra delle proprietà. Se questi concetti non sono ben chiari, date una ripassata alla [lezione 4](#) e poi ritornate qui.

La Label1 conterrà il testo **L'utente ha cliccato sul form**

La Label2 conterrà il testo **X**

La Label3 conterrà la parola **volte**

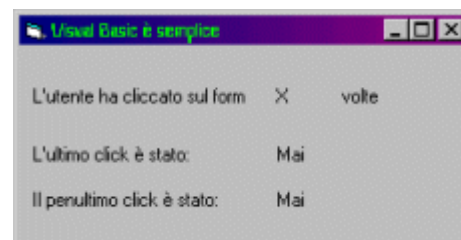
La Label4 conterrà: **L'ultimo click è stato:**

La Label5 conterrà: **Mai**

La Label6 conterrà: **Il penultimo click è stato:**

Ed infine la Label7 conterrà il testo: **Mai**

Se avrete fatto correttamente, il form dovrebbe essere uguale a quello mostrato nella figura qui a fianco.



Com'è facilmente intuibile nella Label2 verrà inserito il numero di click effettuati sul form, nella Label5 verrà inserito l'orario dell'ultimo click e la Label7 conterrà l'orario del penultimo click.

Ora possiamo ritornare al codice del nostro form.

Immaginiamo che le variabili di tipo elementare siano atomi e proviamo a collegarle tra loro per formare una molecola. Infatti in Visual Basic è possibile creare nuovi tipi di dati complessi unendo tra loro più variabili elementari di diverso genere.

Modifichiamo il codice fino ad ottenere il seguente:

```
1. Private Type ClickConData
2.     NumeroClick As Integer
3.     UltimoClick As Date
4.     PenultimoClick As Date
5. End Type
6.
7. Dim MioClick As ClickConData
8.
9. Private Sub Form_Click()
10.    MioClick.NumeroClick = MioClick.NumeroClick + 1
11.    MioClick.PenultimoClick = MioClick.UltimoClick
12.    MioClick.UltimoClick = Now
13.    Label2.Caption = MioClick.NumeroClick
14.    Label5.Caption = MioClick.UltimoClick
15.    Label7.Caption = MioClick.PenultimoClick
16. End Sub
```

Analizziamo con calma i tre segmenti del codice:

```
1. Private Type ClickConData
2.     NumeroClick As Integer
3.     UltimoClick As Date
4.     PenultimoClick As Date
5. End Type
```

La prima riga definisce un nuovo tipo di nome **ClickConData**.


La seconda riga definisce il membro NumeroClick, del tipo elementare Integer.

La terza riga definisce il membro UltimoClick, del tipo elementare Date.

La quarta riga definisce il membro PenultimoClick, del tipo elementare Date.

La quinta riga termina la definizione del tipo ClickConData.

È fondamentale ricordare che ogni nuovo tipo deve avere un nome univoco, unico, in modo da evitare confusioni.

Quello appena fatto è un modello per variabili, un nuovo tipo di dati, un *tipo di dati definito dall'utente* . È un macrotipo, formato da 3 sottotipi elementari.

Continuiamo la nostra analisi del codice:

```
7. Dim MioClick As ClickConData
```

Quest'unica riga dichiara una variabile di nome **MioClick** del nuovo tipo ClickConData. Questa variabile conterrà i dati del nostro programma, cioè il numero di click totali, la data dell'ultimo click e la data del penultimo click.

Vediamo ora le ultime righe del nostro codice:

```
9. Private Sub Form_Click()  
10.     MioClick.NumeroClick = MioClick.NumeroClick + 1  
11.     MioClick.PenultimoClick = MioClick.UltimoClick  
12.     MioClick.UltimoClick = Now  
13.     Label2.Caption = MioClick.NumeroClick  
14.     Label5.Caption = MioClick.UltimoClick  
15.     Label7.Caption = MioClick.PenultimoClick  
16. End Sub
```

La prima e l'ottava riga non necessitano di spiegazione; le abbiamo viste in una lezione precedente.

La seconda riga introduce qualche novità: MioClick.NumeroClick identifica il membro NumeroClick della variabile MioClick. Per accedere ad un membro di una variabile di tipo utente è necessario scrivere il nome della variabile, farlo seguire da un punto ed accodargli il nome del membro a cui ci si riferisce. Infatti in Visual Basic le variabili di tipo complesso sono trattate come fossero degli oggetti.

Per cui la seconda riga dice al programma di prendere il valore di MioClick.NumeroClick, inizialmente 0, aggiungere 1 e memorizzare il risultato in MioClick.NumeroClick.

La terza riga è analoga alla precedente. Essa dice di prendere il valore del membro UltimoClick della variabile MioClick e scriverlo nel membro PenultimoClick della stessa.

La quarta riga dice semplicemente di leggere la data e l'ora attuale (istruzione **Now**) e memorizzarla nel membro UltimoClick della variabile MioClick.

Infatti **prima** di eseguire la quarta riga, il valore di MioClick.UltimoClick è la data del penultimo click, poiché essa ancora non è stata aggiornata. Sarà la quarta riga che l'aggiognerà. Per questo motivo prendiamo il valore di MioClick.UltimoClick e lo mettiamo in MioClick.PenultimoClick **prima** dell'esecuzione della quarta riga.

La quinta, la sesta e la settima riga scrivono dentro le Label i valori dei singoli membri della variabile MioClick.

Eseguiamo il nostro programma e vediamo che succede. Non appena clicchiamo una prima volta sul form vedremo che sarà mostrato l'orario completo del click sul form.

Invece alla voce del penultimo click risulta 0.00.00 perché non c'è stato il penultimo click. Come già detto tutte le variabili al loro primo utilizzo hanno valore 0.



Proviamo ora a cliccare una seconda volta sul form e vediamo il risultato.

Innanzitutto viene incrementato il numero di click progressivi da 1 a 2.

Inoltre l'orario dell'ultimo click va a finire dentro lo spazio dedicato all'orario del penultimo click e viene aggiornata l'area dell'ultimo click con l'ora corrente.



È consigliabile far ricorso a definizione di nuovi tipi quando è necessario utilizzare dei dati costantemente collegati tra loro, piuttosto che utilizzare più variabili semplici separate. Se dovesse esserci la necessità di definire un'altra copia di quel genere non è necessario definire tante variabili separate. Sarà necessaria soltanto una nuova variabile del tipo complesso.

Passiamo ora al concetto delle costanti.

Detto in breve le [costanti](#) sono variabili il cui valore viene definito in [fase di progettazione](#) ed alle quali non si può cambiare valore in [fase di esecuzione](#).

Supponiamo di avere dentro un grande programma 100 forms diversi e sulla barra del titolo di ognuno vogliamo che appaia il nome del programma. Dobbiamo aprire i form uno per uno ed andare a scrivere dentro la proprietà Caption di ogni form, la riga "*Il mio programma - Prima edizione*".

Fin qui è ok. È necessario farlo.

Però pensiamo al futuro... Dopo un certo periodo di tempo il nostro programma, che miglioreremo di giorno in giorno, passerà da *Prima Edizione* a *Seconda Edizione*.

Sarà necessario aprire nuovamente tutti i forms e cambiare tutte le caption di prima... e se uscisse in seguito la terza edizione? Tutto da capo....

Pensiamo pure ad un'altra cosa... Dentro il nostro programma vogliamo calcolare più volte l'area di un cerchio e vogliamo farlo nel modo più preciso possibile. Dobbiamo quindi memorizzare il valore del PI greco. In ogni punto del codice dove si rende necessario utilizzare questo valore dobbiamo scrivere il numero 3.141592.....

Ci sono molti svantaggi in questo: il principale è che possiamo facilmente sbagliare una cifra e far commettere al computer errori nel calcolo e rischiamo di perdere ore intere prima di rintracciare l'errore.

Se invece definiamo una costante e le forniamo il valore del PI greco sarà molto più difficile sbagliare, ci basterà scriverlo una volta sola corretto, poiché tutte le volte che avremo bisogno d'esso scriveremo il nome della costante al posto del valore.

Ecco in pratica quanto detto. Torniamo alla finestra del codice e scriviamo questa riga:

```
Private Const PIGreco As Double = 3.14159265358979
```

Questa infatti dichiara una costante di nome **PIGreco** di tipo Double e le assegna il valore. Così ogni volta che dovremo scrivere 3.141592..... sarà molto più agevole e più comprensibile scrivere PIGreco al suo posto.

Dentro la routine Form_Click aggiungiamo la riga:

```
Me.Caption = PIGreco
```

Questa, infatti scrive il valore del PI greco sulla barra del titolo del form. L'utilizzo delle costanti non velocizza né rallenta in alcun modo l'esecuzione di un programma, ma lo rende sicuramente più comprensibile ad una seconda rilettura o modifica.

Vi sarete sicuramente chiesti perché in queste lezioni alcune righe del codice iniziano un po' più avanti delle altre righe.

Beh, è soltanto una manovra chiamata **Indentatura**, un'italianizzazione del termine inglese *Indent*, cioè rientranza del margine. Il suo unico scopo è rendere il codice più leggibile. Per esempio:

```
1. Private Sub Form_Click()  
2.     Label1.Caption = "Qualcosa"  
3. End Sub
```

È più leggibile di:

```
1. Private Sub Form_Click()  
2. Label1.Caption = "Qualcosa"  
3. End Sub
```

Infatti, se l'indentatura è fatta correttamente si capisce facilmente dove inizia e dove finisce la routine Form_Click. Tutto quello che è indentato è una parte di quella routine o di quel ciclo. Se il testo non fosse indentato, chi rivede il codice sarà costretto a leggersi le righe una per una alla ricerca dell'istruzione End Sub.

Non esiste un numero di caratteri preciso per il rientro di indentatura; alcuni utilizzano due spazi, altri ne utilizzano quattro, mentre altri ancora utilizzano rientrare il testo con le tabulazioni. Noi nei nostri corsi utilizzeremo un'indentatura di 4 caratteri.

L'indentatura si utilizza in tutti i linguaggi; per il computer non cambia nulla, ma per il programmatore o per un revisore del codice cambia molto. È fondamentale imparare ad utilizzarla quanto prima possibile.

[Fibia FBI](#)

28 Ottobre 2000



[Torna alla quinta lezione](#)

[Vai alla settima lezione](#)

