Corso base - Lezione 5

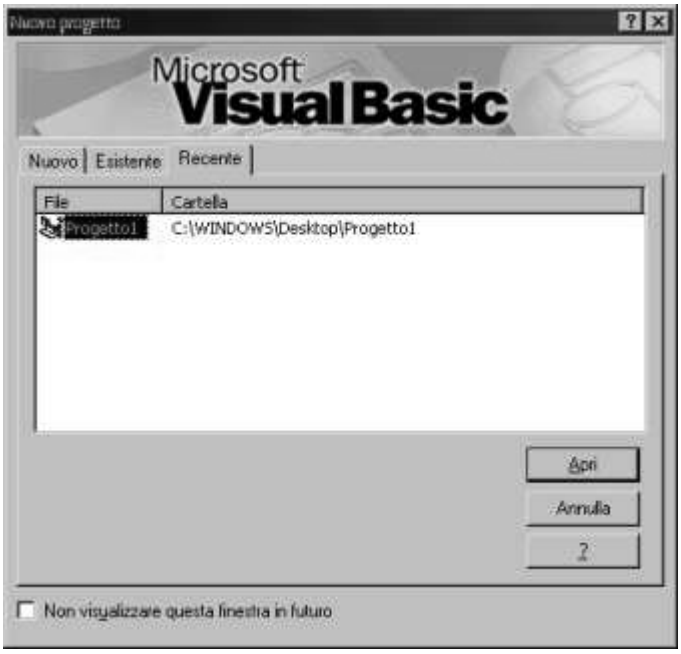
http://www.vbsimple.net/base/bas_05.htm

- Richiamo di un progetto attraverso la voce Recente.
- Le variabili.
- I tipi di dati elementari.
- Vita di una variabile.

Da questa lezione in poi lavoremo sempre sul nostro progetto iniziato nella lezione 1.

Per cui, anche se non viene espressamente detto, ogni operazione da eseguire su Visual Basic andrà fatta su questo progetto, richiamato nel modo già visto nella lezione 2, oppure attraverso la voce **Recente** presente sulla finestra di avvio di Visual Basic.

Basta cliccare sulla voce Recente, che apparirà una finestra del tutto simile a quella mostrata qui a fianco. Selezionare il progetto da aprire, nel nostro caso il Progetto1 e premere il pulsante Apri.



Abbandoniamo per un attimo la finestra di progettazione dei form e dedichiamoci alla vera programmazione Visual Basic, concentrandoci sulla finestra del codice del form.

Ogni programma per funzionare correttamente deve appoggiarsi sulle variabili, delle aree di memoria dedicate alla memorizzazione di dati temporanei.

Perché?
Perché il programmatore in fase di progettazione non conosce completamente le decisioni che prenderà l'utente nell'utilizzo del programma. Supponiamo di avere una casella dentro la quale è possibile inserire numeri. Il programmatore, a priori, non sa il valore preciso che l'utente immetterà in essa. Egli sa solo che immetterà un valore.
Le variabili servono per immagazzinare i dati immessi dall'utente e poi svolgere i calcoli con il valore contenuto nella variabile, il tutto in fase di esecuzione.

In Visual Basic, come in ogni altro linguaggio di programmazione, è necessario definire comunque, prima del loro utilizzo, il tipo di dati che la variabile utilizzerà, per permettere

al computer di allocare la memoria necessaria per contenere i dati che vi verranno immessi.

Se decidiamo che una variabile dovrà contenere numeri interi, essa non potrà contenere numeri decimali o parole. Quest'operazione durante la quale si decide il tipo di dati da trattare con la variabile è detta **dichiarazione**.

In Visual Basic esistono varie tipologie di dati; queste sono:

- **Integer**
  Capace di memorizzare soltanto numeri interi, positivi o negativi.
- **Byte**
  Che può contenere un numero compreso tra 0 e 255 ovvero la dimensione di un byte, quindi 8 bit (2 ^ 8 = 255).
- **Long**
  Rappresenta un numero intero lungo, con una capacità di molto maggiore rispetto all'Integer.
- **Single**
  Rappresenta un numero decimale con virgola mobile a precisione singola.
- **Double**
  Analogo al single, ma con precisione doppia e quindi ampiezza molto maggiore.
- **Currency**
  Può contenere numeri molto ampi con un massimo di 4 cifre decimali.
- **Date**
  Tipo usato per rappresentare date complete di giorno, mese, anno a 4 cifre ed orari comprensivi di ora, minuti e secondi.
- **Boolean**
  Rappresenta una condizione. Il suo valore può essere soltanto **Vero** o **Falso**.
- **String**
  Contenitore per stringhe, testi. Il suo valore non è un numero, ma una sequenza di simboli. Per essere utilizzato in formule matematiche deve essere convertito.
  Il valore di una variabile di tipo String deve sempre racchiuso tra virgolette.
- **Object**
  Puntatore ad un oggetto in memoria. Il suo valore non è una copia di un altro. Qualunque modifica fatta all'oggetto puntato da una variabile di questo tipo si riflette sull'oggetto puntato da essa.
- **Variant**
  Tipologia di dati variante. Può cioè contenere un valore qualunque, poichè si adata a tutti i tipi precedentemente visti.
- **Decimal**
  Particolare tipologia di dati, non assegnabile direttamente ad una variabile, ma è un sottotipo della tipologia Variant. Può contenere dati numerici molto ampi.

Tutti i tipi di dati fondamentali sono approfonditi nella sezione Informazioni aggiuntive.

Allora perché non utilizzare le tipologie di dati più ampie, in modo da evitare un overflow, un superamento del limite massimo memorizzabile da una variabile?
Perché tutte le tipologie di dati hanno un'occupazione diversa. Il tipo **Byte** occuperà sempre 8 bit, ma il tipo **Long**, occuperà sempre 32 bit, cioè 4 volte lo spazio occupato dal

tipo precedente. Questo può provocare un notevole rallentamento in caso di decine di dichiarazioni di quel genere.

Se siamo sicuri che un utente immetterà un numero compreso tra 0 e 999 senza virgola decimale, il tipo byte non sarà sufficiente. Se il numero fosse minore di 256 potrebbe andar bene, ma se l'utente immettesse il numero 789 e noi avessimo definito la variabile di tipo byte, si verificherebbe un errore ed il programma si chiuderebbe con un avviso.

Per cui è bene pensar prima alle scelte dell'utente. Il tipo Integer, può andar bene, perché è il tipo di dati più piccolo dopo il byte e può contenere numeri fino a 32767.

Per dichiarare una variabile in Visual Basic, si scrive:
Dim NomeDellaVariabile **As** TipoDeiDati

dove NomeDellaVariabile è un nome contenente lettere o numeri, ma la cui prima, tuttavia, deve essere una lettera e TipoDeiDati è uno dei tipi di dati visti prima. Un esempio di dichiarazione valida può essere:

Dim **Numero1** As Integer

mentre non è valida una dichiarazione del genere:

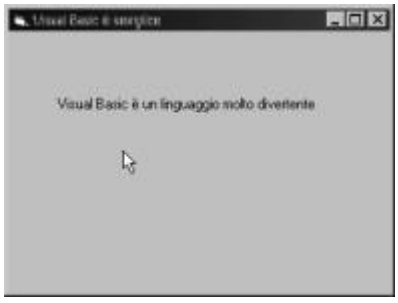
Dim **1Numero** As Integer

poiché la prima posizione del nome non è una lettera.

Vediamo ora dove scrivere queste cose. Questo problema in programmazione si chiama *Vita di una variabile* (gli inglesi invece utilizzano la dizione **Scope**, area di visibilità di una variabile). Infatti se una variabile è dichiarata all'interno di una funzione collegata ad un evento (vedi a tal proposito la lezione 4), la sua vita sarà legata a tale funzione. Noi non potremo utilizzare quella variabile all'esterno della funzione legata all'evento. Inoltre ogni volta che si verificherà tale evento il contenuto della variabile sarà azzerato, perdendo il valore memorizzato prima. Vediamo in pratica quanto detto finora. Apriamo la finestra del codice, selezionate dalla zona sinistra in alto l'oggetto Form e dalla destra l'evento Click e scriviamo questo codice:

```
1. Private Sub Form_Click()
2.     Dim Numero1 As Integer
3.     Numero1 = Numero1 + 1
4.     Label1.Caption = Numero1
5. End Sub
```

Fatto questo eseguiamo il programma attraverso la voce Avvia dal menu Esegui e proviamolo.
Verrà visualizzata la finestra con la voce *Visual Basic è un linguaggio molto divertente*.



Infatti quello è il testo che abbiamo impostato alla Label1 nella lezione precedente.

Proviamo a cliccare su un punto qualunque della finesta, eccetto la barra del titolo in alto, i

pulsanti su essa e sulla nostra Label1.

Sparirà il testo visto prima ed apparirà al suo posto il numero 1.
Questo accade perché nella routine legata all'evento Click sul form abbiamo deciso che la Label1 avrà il valore della variabile Numero1.
Dentro il nostro codice c'è scritto infatti:

```
2. Dim Numero1 As Integer
3. Numero1 = Numero1 + 1
4. Label1.Caption = Numero1
```

La prima riga è la definizione della variabile Numero1, di tipo Integer (numero intero).

La seconda riga dice al programma di aumentare il valore della variabile Numero1 di 1 e memorizzare il risultato sulla variabile Numero1.
All'inizio tutte le variabili Integer sono impostate a zero. Quest'operazione allora farà questo calcolo: Numero1 = 0 + 1
per cui alla fine dell'operazione la variabile Numero1 avrà il valore 1.

La terza riga imposta il contenuto della Label1 (ovvero il contenuto della proprietà Caption dell'oggetto Label1) uguale al contenuto della variabile Numero1. Per cui all'interno del Form apparirà 1.

Ma che succede se l'utente clicca nuovamente sul form?
Ci si potrebbe aspettare che il secondo click producesse il numero 2 e così via per i click successivi.
Ma non è così. Infatti la vita della variabile Numero1 inizia dal punto della sua dichiarazione e finisce dove termina la routine dentro la quale la variabile è dichiarata; quindi la variabile Numero1 *muore* all'uscita della routine legata all'evento Click. Per cui alla sua successiva dichiarazione la variabile non *ricorda* il valore che ha avuto *nelle vite precedenti* e le verrà sempre assegnato il valore iniziale 0.

Se volessimo dichiarare una variabile il cui valore non viene azzerato ogni volta, dovremmo dichiarare la variabile all'interno di un modulo  oppure nell'area dedicata alle dichiarazioni del form. La creazione dei moduli la vedremo più avanti. La seconda è facilmente accessibile, basta cliccare sulla lista dedicata agli oggetti nella finestra del codice e scegliere la voce **(generale)** e sulla lista di destra scegliere la voce **(dichiarazioni)**.

Proviamo ora a copiare la riga dichiarazione della variabile Numero1 nell'area delle dichiarazioni, ricordandoci di cancellare quella dichiarata precedentemente, perché non possono esistere due variabili con lo stesso nome nella stessa area.
Otteniamo quindi il seguente codice:

```
1. Dim Numero1 As Integer
2.
3. Private Sub Form_Click()
4.     Numero1 = Numero1 + 1
5.     Label1.Caption = Numero1
6. End Sub
```

Eseguiamo il programma e vediamo che succede!

Ad ogni click del mouse sul form il numero (la Label1) aumenterà di 1 perché la vita della variabile Numero1 non cessa di esistere all'uscita della routine **Form_Click**, ed al successivo richiamo il suo valore sarà ricordato. L'utente potrà cliccare fino a 32767 volte e vedere il numero aumentare. Se cliccasse un'altra volta in più accadrebbe che la variabile Numero1 andrebbe in overflow e si genererebbe un errore, perché la tipologia di dati Integer può contenere numeri fino a 32767.

È bene limitare l'uso delle variabili dichiarate nella zona dichiarazioni, poiché la loro vita durerà fintanto che vivrà l'oggetto (il form o la classe) nel quale esse sono dichiarate.

Se ci dovesse servire una variabile per fare la somma di 5 numeri e poi il valore di questa non ci servisse più, sarebbe del tutto inutile e rischioso dichiarare la variabile nella zona dichiarazioni. Dichiareremo la variabile all'interno della routine dentro la quale ci serve.

Così facendo semplificheremo il lavoro al computer, ed allontaneremo la possibilità di commettere errori involontari.

Fibia FBI
28 Ottobre 2000

 Torna alla quarta lezione

Vai alla sesta lezione