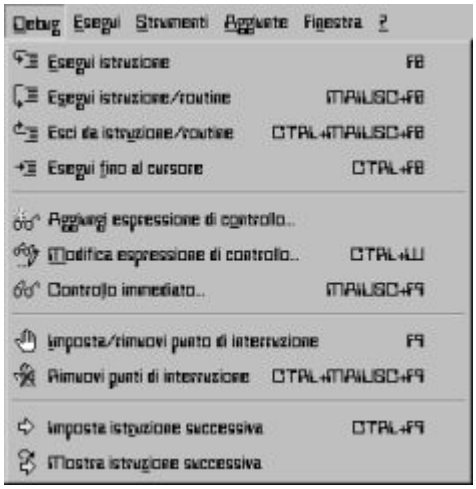
Corso base - Lezione 15

http://www.vbsimple.net/base/bas_15.htm

- Alla scoperta dei menu di Visual Basic (quarta parte).
- Menu Debug.
- Menu Esegui.

Il Corso Base si conclude con la visione di tutti i menu dell'IDE di Visual Basic. Alcune voci sono semplicissime da comprendere e presenti in quasi tutti i programmi per Windows, ma alcune sono realmente complesse e talvolta ostiche da utilizzare.
Nelle lezioni 12-17 vedremo uno per uno tutti i menu ed approfondiremo le loro opzioni.

Menu Debug



Il termine Debug indica la procedura di rimozione dei bug, errori, nel sorgente di un programma.

Questo menu, appunto, contiene alcuni comandi dedicati al controllo dell'esecuzione del progetto sviluppato. Sarà possibile seguire riga per riga il comportamento del codice, forzare dei salti, inserire dei blocchi e fissare delle viste su determinati dati.

Per avviare il progetto in modalità debug basterà utilizzare la prima voce al posto del normale esecuzione. Per passare dalla modalità debug alla normale esecuzione premere il pulsante **Avvia** (F5), mentre per ritornare dalla normale esecuzione alla fase di debug utilizzare il pulsante **Interrompi** (CTRL+INTERR); entrambe le opzioni si trovano sia nel menu Esegui sia nella barra degli strumenti.

1. **Esegui istruzione**
   Avanza il punto di esecuzione di un'istruzione semplice. Se la prossima istruzione contiene la chiamata ad una funzione o Sub, il punto di esecuzione si porterà all'interno della funzione e si fermerà sulla prima istruzione all'interno della funzione.

2. **Esegui istruzione/routine**
   Avanza il punto di esecuzione di un'istruzione all'interno del modulo in cui ci si trova. Se la prossima istruzione contiene la chiamata ad una funzione, il codice all'interno della funzione sarà eseguito ed il cursore sarà posizionato sulla riga immediatamenta successiva a quella appena eseguita. Viene utilizzato in alternativa

all'*Esegui istruzione* per seguire il comportamento del codice senza preoccuparsi del funzionamento delle varie funzioni presenti.

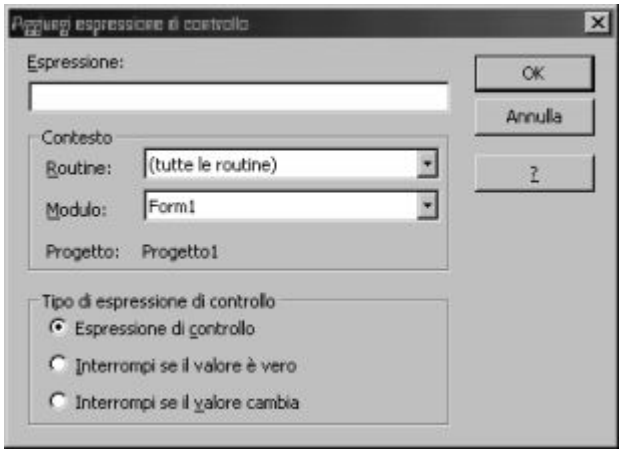
3. **Esci da istruzione/routine**
   Esegue tutto il codice che si trova all'interno della routine corrente, riportando il cursore di esecuzione alla routine che si trova in posizione immediatamente superiore nello stack delle chiamate e che ha chiamato la routine corrente.

   Utile per seguire il funzionamento del codice, senza preoccuparsi di cosa viene svolto all'interno della funzione in cui l'esecuzione si trova al momento.
   Viene utilizzato anche quando, per sbaglio, si è utilizzato il comando *Esegui Istruzione*, anziché *Esegui istruzione/routine*, per riportare l'esecuzione al punto in cui si trovava prima di richiamare la routine.

4. **Esegui fino al cursore**
   Esegue tutto il codice compreso tra il punto di esecuzione attuale fino al punto in cui si trova il cursore. Molto utile per eseguire porzioni di codice (specialmente nei cicli) e proseguire il controllo di debug dal punto in cui si trova il cursore.

5. **Aggiungi espressione di controllo**
   Permette di aggiungere un'espressione di controllo, meglio conosciuta come watch. Uno strumento potentissimo per investigare i dati in fase di debug. I watch saranno visti nella finestra Controlla in struttura gerarchica. La loro efficacia si rivela nell'investigazione di variabili oggetto; infatti saranno mostrate tutte i nomi delle proprietà ed i loro valori al momento del controllo.

   Cliccando sulla voce ci apparirà la finestra di dialogo che richiede i parametri per il watch da aggiungere.



   Nella casella **Espressione** andrà inserito il nome della variabile o dell'oggetto da controllare. Sarà possibile anche inserire operazioni più complesse.

   Alla voce **Contesto** troviamo le voci Routine, Modulo e Progetto (in sola lettura). Sarà possibile infatti aggiungere un watch di una variabile relativa ad una certa funzione di un certo modulo. Se la variabile è dichiarata nell'area dichiarazioni del form potrà essere utilizzata la voce *(tutte le routine)*.
   Altresì se la variabile è dichiarata pubblica all'interno di un modulo potrà essere utilizzata la voce *(tutti i moduli)*. Se la variabile non sarà accessibile nella routine e nel modulo scelto, il suo valore non sarà mostrato.

   In fondo alla finestra di dialogo abbiamo il tipo di espressione di controllo:
   - ***Espressione di controllo***
     Definisce un watch normale che sarà utilizzabile esclusivamente durante la fase di debug.

- ***Interrompi se il valore è vero***
  Definisce un watch che interromperà automaticamente l'esecuzione del programma nel momento in cui l'espressione indicata restituisce il valore Vero.
- ***Interrompi se il valore cambia***
  Definisce un watch che interromperà automaticamente l'esecuzione del programma nel momento in cui il valore dell'espressione cambia.

Il tipo di watch più comune è il primo, ma ogni tanto può essere utile utilizzare gli altri due, per verificare il punto in cui certi dati vengono modificati. In tal caso basterà riprendere l'esecuzione del programma mediante il pulsante Avvia e lasciar fare al watch che effettuerà l'interruzione nella situazione richiesta.

Riprendendo l'esempio visto l'ultima volta nella Lezione 10, aggiungiamo un watch normale relativo alla variabile complessa MioClick. Essa era stata dichiarata in cima al codice del form e pertanto sarà visibile in tutte le funzioni del form. Sceglieremo per cui ***(tutte le routine)*** alla voce Routine e ***Form1*** alla voce Modulo.
Aggiunto il watch eseguiamo il progetto, clicchiamo un paio di volte il pulsante Aggiorna e poi interrompiamo l'esecuzione del programma.

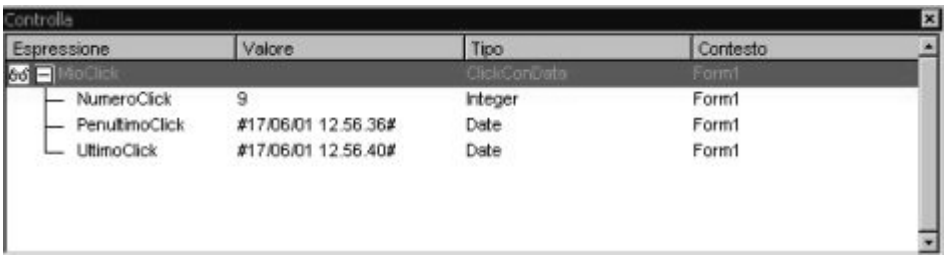


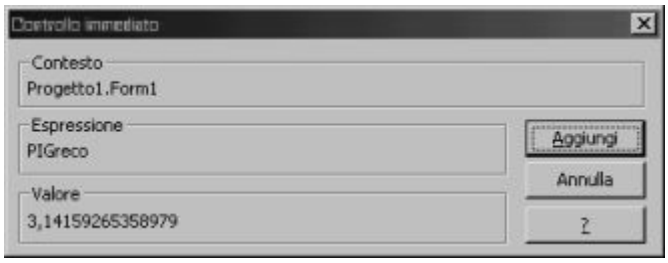
Figura 3

La variabile MioClick è di tipo di dati complesso, definito da noi con il nome di **ClickConData**. Aprendo il tipo di dati viene mostrata tutta la sua gerarchia, con i suoi membri. Appaiono al suo interno i membri **NumeroClick** con il valore di 9, **PenultimoClick** con la sua data ed **UltimoClick** con la sua data. Se il membro non è di sola lettura, sarà possibile anche alterarne i valori semplicemente cliccando sul valore del membro.

Le espressioni di controllo dell'IDE di Visual Basic sono uno strumento molto potente e spesso chiariscono dubbi sulle strutture dati complesse. Ho conosciuto decine di programmatori che avevano problemi per loro irrisolvibili perché non sapevano usare i watch. Dove il sistema Intellisense si ferma, il watch può andare oltre.

6. **Modifica espressione di controllo**
   Consente di modificare i criteri con i quali si è aggiunta un'espressione di controllo. Permette quindi di modificare l'espressione, la routine, il modulo oppure il tipo di un watch.

7. **Controllo immediato**
   Il watch rapido. Consente di valutare immediatamente un'espressione e decidere se aggiungere o meno un watch per essa.



Premendo il pulsante aggiungi, sarà inserito un watch con l'espressione selezionata.

Se viene utilizzato durante la fase di progettazione riporta il valore *<fuori contesto>*. Trova migliore efficacia se utilizzato durante la fase di esecuzione interrotta e provvede una via più rapida dell'inserimento normale di watch. Aggiunto il watch rapido, se i criteri forniti automaticamente non fossero soddisfacenti, sarà possibile modificarli tramite la voce **Modifica espressione di controllo** vista subito prima.

8. **Imposta/rimuovi punto di interruzione**
   I punti di interruzione sono un'ottima soluzione per verificare la normale esecuzione di un codice ed interromperlo soltanto quando viene eseguita una determinata riga di codice. Quando il codice tenta di eseguire quella riga, l'esecuzione del programma sarà interrotta per permettere il debug da quel punto. Sarà possibile riprendere l'esecuzione mediante la voce Avvia. Se la riga include già un'interruzione, questo comanda rimuoverà l'interruzione precedente.

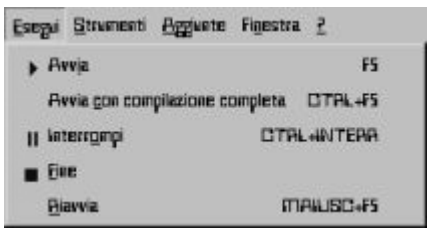
9. **Rimuovi punti di interruzione**
   Elimina tutti i punti di interruzione nel codice.

10. **Imposta istruzione successiva**
    Un comando un po' anomalo che consente di spostare il punto di esecuzione da una riga ad un'altra. Utile soltanto in determinate situazioni, per **evitare temporaneamente** di eseguire una sezione di codice. Sposta il punto di esecuzione attuale alla riga di codice sulla quale si trova il cursore. Lo spostamento può essere effettuato esclusivamente all'interno della stessa routine.

11. **Mostra istruzione successiva**
    Un comando solitamente inutile, che consente di riportare il cursore alla riga in cui il codice è interrotto. Se il progetto contiene molti moduli può essere una via rapida per ritrovare il punto di esecuzione corrente, senza doverlo cercare in tutti i moduli.

È importante ricordare che tutti i punti di interruzione ed i watch non saranno salvati all'interno del progetto. Pertanto saranno cancellati alla chiusura dell'IDE.

Menu Esegui



Il menu Esegui contiene pochi comandi per avviare, interrompere o terminare l'esecuzione di un progetto.

1. **Avvia**
   Avvia l'esecuzione del programma in maniera normale.

2. **Avvia con compilazione completa**

Prima di avviare il programma effettua una compilazione di controllo. Se esistono errori di compilazione il progetto non sarà eseguito. Se il codice è **formalmente** corretto il progetto sarà eseguito.
Si consiglia di utilizzare questo comando invece del precedente per rivelare gli errori che esistono prima di eseguire il programma.

3. **Interrompi**
   Interrompe l'esecuzione del progetto in esecuzione, passando dalla modalità di esecuzione alla modalità di debug, permettendo l'utilizzo dei comandi visti nel menu precedente.

4. **Fine**
   Una soluzione rapida per terminare l'esecuzione di un progetto. Corrisponde all'utilizzo dell'istruzione di codice **End**. Se ne **sconsiglia fortemente** l'uso, poiché il programma non sarà chiuso in maniera corretta, ma sarà forzatamente bloccato, senza eseguire i codici di uscita di tutti i moduli. Saranno lasciate risorse allocate bloccate e files aperti, generando una situazione potenzialmente instabile.
   Utilizzare questo comando soltanto in casi di blocco effettivo del progetto, in cui risulta impossibile chiudere il programma, in modo da poter rimettere mani al codice e fissare tali problemi. È importante ricordare che un programma, una volta compilato, non viene eseguito all'interno dell'IDE. Pertanto, se esiste una situazione di blocco del genere, l'utente non avrà a disposizione il comando Fine per terminare il programma.

5. **Riavvia**
   Altro comando da evitare. Consente di eseguire in una sola operazione il comando Fine ed Avvia, terminando il progetto in esecuzione e riavviandolo dall'inizio.

Tra i due menu analizzati indubbiamente il primo è più difficoltoso, ma contiene tanti comandi utili per rintracciare gli errori presenti nel progetto.
"Una volta che si è ottenuta la prima versione funzionante di un programma, occorrerà ancora moltissimo tempo perché questa versione sia resa affidabile."
(tratto da Cantù e Tendon - Borland C++ versioni 4 e 4.5 - 1994 Apogeo)

Fibia FBI
17 Giugno 2001

Torna alla quattordicesima lezione Vai alla sedicesima lezione

