Corso Intermedio - Lezione 5

http://www.vbsimple.net/intermed/int_05.htm

- Possiamo cominciare.
- L'istruzione Option Explicit.
- La gestione dei files binari.

Nella fase iniziale di questo progetto ci occuperemo soltanto dell'inserimento dei dati nel file di archivio e soltanto in seguito vedremo come leggerne i dati scritti in precedenza. Il file verrà aperto all'avvio dell'applicazione e chiuso all'uscita del programma. Questi due momenti sono infatti segnati da due eventi specifici dell'oggetto Form:

- ***Load*** indicherà l'operazione di caricamento del form ovvero il momento prima che il form relativo venga caricato.

- ***Unload*** indica invece l'operazione di scaricamento del form ovvero il momento prima della sua chiusura, subito dopo il tentativo di scaricamento (***QueryUnload***).

In genere questi sono gli eventi più idonei per regolare il comportamento del codice all'avvio e alla chiusura del form. Lo ricordiamo, in questo nostro progetto i due eventi sono utilizzati rispettivamente per aprire e per chiudere il file di dati che si troverà nella stessa cartella della nostra applicazione. Sarà quindi ovvio che il nostro file di dati resterà aperto fino a quando l'applicazione sarà aperta e non dovremo preoccuparci di aprirlo e chiuderlo ad ogni suo utilizzo.

Nel terzo capitolo di questo corso abbiamo visto come definire la nostra base dati. Inseriamo all'interno di un nuovo modulo standard il codice visto in precedenza:

```
 1. Option Explicit
 2.
 3. Public Type uDisco
 4.     intIndice As Integer
 5.     blnEliminato As Boolean
 6.     strTitoloBrano As String * 30
 7.     strAutore As String * 30
 8.     strTitoloCD As String * 20
 9.     intTraccia As Integer
10.     intAnnoPubblicazione As Integer
11.     strNote As String * 255
12. End Type
```

Al fine di evitare un problema comunissimo, si raccomanda fortissimamente di **usare sempre** l'istruzione *Option Explicit*. Essa consente infatti di evitare involontari errori nella scrittura del codice, obbligando il programmatore a dichiarare ogni variabile prima di utilizzarla. Dove sta il vantaggio? Sembra piuttosto un limite fastidioso!

Supponiamo di scrivere un certo programma che effettua dei semplici calcoli geometrici

dove Base = 5 e Altezza = 4:

```
Area = Base * Altezza / 2
Perimetro = Base * 2 + Alteza * 2
```

Mentre la prima delle due righe di codice produce il risultato corretto: Area = 10, la seconda delle due righe produce un risultato errato: Perimetro = 10.

Chi ha l'occhio più attento si sarà accorto che la seconda riga presenta un errore: la variabile Altezza è scritta in maniera errata; manca infatti una Z ma il compilatore non se ne accorge neanche? Non si presenta alcun errore? Il compilatore se ne accorge, eccome! La prova è data dal fatto che il risultato presentato per il perimetro è 10.

Il calcolo che viene infatti eseguito dal programma sarà `Perimetro = 5 * 2 + 0 * 2` perché alla variabile **Alteza** non è mai stato assegnato un valore e pertanto assume il valore predefinito 0.

L'utilizzo di Option Explicit obbliga invece il programmatore a dichiarare ogni variabile e quindi elimina a priori la possibilità di errori di scrittura involontari. Nel momento in cui avessimo provato a compilare il precedente codice si sarebbe generato un errore di *"Variabile non definita"*.

Questa deve essere la prima regola, la Golden Rule (regola d'oro) da applicare in ogni progetto, la prima riga da scrivere in ogni modulo ma c'è una soluzione semplice per evitare di doversene ricordare ogni volta. Tramite il menu ***Strumenti*** dell'IDE si accede al pannello delle ***Opzioni*** raffigurato nella sotto.

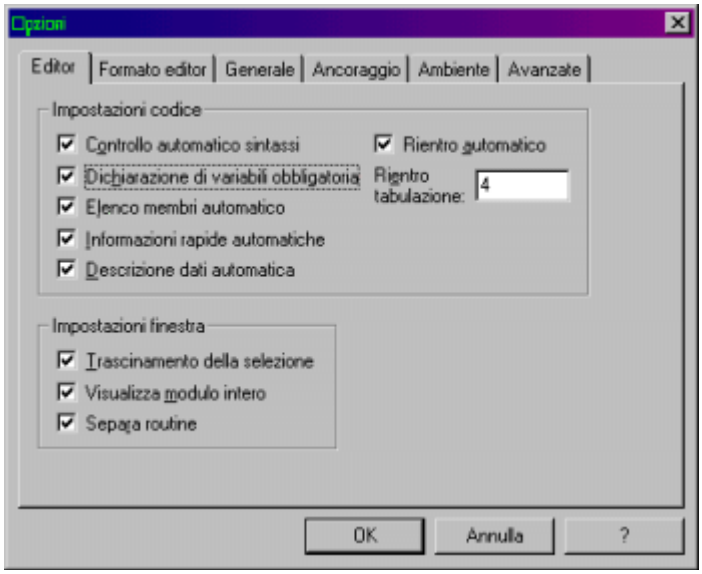

Figura 1

L'opzione da selezionare è **"Dichiarazione di variabili obbligatoria"**. Essa farà in modo che ogni nuovo modulo aggiunto al progetto contenga alla prima riga l'istruzione Option Explicit.

Torniamo al nostro progetto e al codice visto in precedenza: l'istruzione Type definisce il nuovo tipo di dati di nome **uDisco**. Ricordiamo che non è possibile definire tipi di dati pubblici all'interno di un form ma soltanto all'interno di un modulo standard.

Gestire dei semplici files binari con record a lunghezza fissa è molto semplice. Visual Basic stesso infatti provvede alcune semplici istruzioni per l'apertura e successiva gestione di files binari. Avendo definito il record da utilizzare possiamo passare al resto del codice, da inserire **all'interno del form**:

```
1. Option Explicit
2.
3. Private intFileDati As Integer
4. Private udtDiscoCorrente As uDisco
5.
```

Della riga 1 ne abbiamo già discusso, vero? 🙂
Chi ha seguito il corso base dovrebbe aver notato la differenza di dichiarazione delle variabili. Non utilizzeremo più l'istruzione *Dim* ma uno dei modificatori di accesso visti nella lezione precedente: *Private*.

Sebbene Dim e Private si equivalgano come significato, è preferibile dichiarare i dati a livello di modulo tramite Private o Public e dichiarare i dati a livello di routine tramite Dim.

Alla riga 3 è dichiarata una variabile a livello di modulo, cioè accessibile a tutte le routine del form, di nome **intFileDati**, che conterrà il numero del file di dati aperto all'avvio e chiuso alla fine del programma.

Alla riga 4 è dichiarata un'altra variabile, sempre a livello di modulo, di nome **udtDisco** e di tipo ***uDisco*** che servirà a contenere le informazioni su un singolo record. Verrà utilizzata per recuperare le informazioni dal file di dati e per scrivere nuove informazioni sul file.

```
6. Private Sub Form_Load()
7.     intFileDati = FreeFile
8.     Open App.Path & "\BRANI.DAT" For Random As intFileDati Len = Len
   (udtDiscoCorrente)
9. End Sub
10.
```

Alla luce di quanto detto in precedenza il file di dati verrà aperto all'avvio del progetto (evento ***Load*** del form principale) e verrà chiuso in corrispondenza alla chiusura (evento ***Unload*** dello stesso form).

Prima di richiedere l'apertura del file di dati è necessario recuperare un ***numero di file*** relativo all'applicazione. Esso servirà da riferimento al file aperto durante le operazioni di lettura e scrittura. Il numero di file sarà un numero compreso tra 1 e 255 se il file non dovrà essere accessibile ad altre applicazioni oppure compreso tra 256 e 511 se il file dovrà essere utilizzato anche da altre applicazioni.

Invece di assegnare arbitrariamente un numero ad ogni file da aprire si raccomanda di utilizzare la funzione **FreeFile** che recupera il primo numero di file disponibile come è stato fatto alla riga 7 del nostro codice. Il numero di file da aprire sarà indicato quindi dalla variabile **intFileDati** e tutte le operazioni di apertura, lettura, scrittura e chiusura verranno effettuate su tale numero di file.

L'istruzione *Open* consente di aprire un file di dati in diverse modalità, accennate

nell'HowTo dedicato alla lettura del contenuto di un file.
Queste modalità indicano il tipo di accesso ai dati: in lettura (*Input*), in scrittura (*Output*), in aggiunta (*Append*), modalità binaria (*Binary*) oppure ad accesso casuale (*Random*).

Il file che verrà aperto si chiama ***BRANI.DAT*** ed è contenuto nella stessa cartella in cui si trova l'applicazione. Nella scrittura di un programma è fondamentale mantenere una certa generalizzazione: sono da metter al bando i riferimenti assoluti ai file quali "C:\Windows\Desktop\Programma\Nomefile.txt" quando il programma è contenuto nella cartella Programma del Desktop. Se il file si trova nella stessa cartella del programma oppure in una sua cartella è possibile utilizzare la proprietà **Path** dell'oggetto globale **App** per recuperare il percorso della cartella del programma.

Nel nostro esempio abbiamo indicato il file tramite ***App.Path & "\BRANI.DAT"*** e questa è la soluzione ideale nella maggioranza dei casi. Si può generare un errore nel momento in cui il programma si trova sulla radice del disco (fisico o logico). Vedremo solo in seguito come evitare problemi di questo genere.

Le modalità di apertura del file determinano il comportamento delle istruzioni di lettura e scrittura. Rivediamo per un attimo l'istruzione di apertura:

```
8.      Open App.Path & "\BRANI.DAT" For Random As intFileDati Len = Len
   (udtDiscoCorrente)
```

La modalità di apertura è quella ad accesso casuale (*Random*), che consente operazione sia di lettura che di scrittura e spostamenti nel contenuto del file a record interi. Sarà infatti possibile leggere il record 12000 direttamente, senza doverli scorrere uno per uno o calcolare la posizione fisica del suo inizio.

Per effettuare operazioni di questo genere, però, è necessario definire l'ampiezza del record. durante l'apertura del file. In seguito al numero di file da aprire (**intFileDati**) verrà specificata l'ampiezza del record a dimensione fissa. L'ampiezza è calcolata tramite la funzione **Len** applicata al record dichiarato in precedenza. Per specificare la dimensione del record da utilizzare è necessario indicare la dimensione in bytes nella forma **`Len = Ampiezza`**.

L'istruzione di apertura in questa forma assicura che gli spostamenti avvengano sul singolo record, concedendo un accesso casuale al file cui verrà fatto riferimento tramite il numero di file indicato dalla variabile **intFileDati**.

```
11. Private Sub Form_Unload(Cancel As Integer)
12.     Close intFileDati
13. End Sub
14.
```

Aperto un file è fondamentale ricordarsi di chiuderlo al termine del suo utilizzo. Nel nostro caso il termine dell'utilizzo corrisponde alla chiusura del form principale. Utilizzeremo quindi l'evento ***Unload*** per chiudere il file aperto in precedenza.

L'istruzione **Close** seguita dal numero di file da chiudere effettua la chiusura del singolo file specificato, ma può anche omettere il numero del file da chiudere. Nel caso di omissione del numero di file saranno chiusi tutti i files aperti tramite l'istruzione **Open**. È

altresì possibile specificare più numeri di file separandoli tra loro con una virgola.

Il progetto così come si presenta al momento definisce soltanto il record a lunghezza fissa e la modalità di apertura del file. L'esecuzione del progetto non eseguirà nulla tranne l'apertura e la chiusura del file di dati.

Vedremo nella prossima lezione come effettuare i primi accessi al file.

Fibia FBI
4 Aprile 2002

 Torna alla quarta lezione

Vai alla sesta lezione