di Francesco Petroni

MILLE MODI PER LEGGERE FILE ESTERNI DA UN PROGRAMMA VISUAL BASIC

Come al solito il titolo dell'articolo serve per attirare l'attenzione di quei lettori che invece di leggere la rivista con la dovuta attenzione la sfogliano... svogliatamente. Questi lettori, supposti utilizzatori a vario livello di Visual Basic, che distrattamente sfogliano le pagine di MC e leggono la frase «Mille modi...», probabilmente pensano «Accidenti, ma i modi sono proprio 1.000? Quanti ne conosco, e quelli che non conosco potrebbero essere utili per le mie necessità?». A chi si è fatto attrarre dal titolo e, bontà sua, ha cominciato a leggere l'articolo dobbiamo subito delle spiegazioni.

Prima parte

Visual Basic dispone di numerosissime funzionalità (istruzioni, metodi, proprietà, ecc., che hanno a che fare con operazioni su file esterni). È possibile, per queste funzionalità, tentare vari tipi di classificazione, la prima delle quali sicuramente è quella che fa riferimento al tipo di file da caricare. Ad esempio, famiglia dei file Testuali:

file testuali lunghi, non formattati non strutturati (es. formato TXT)

file testuali lunghi, formattati, non strutturati (es. formato RTF)

file testuali lunghi, formattati, letti come oggetti OLE (es. formato DOC)

formati testuali organizzati a «righe»

formati testuali semplici omogenei (es. elenco di voci per una lista)

formati testuali semplici disomogenei (es. elenco di costanti per un file di configurazione)

formati testuali strutturati, di tipo «comma delimited», contenenti Dati

formati testuali strutturati, di tipo «fixed lenght», contenenti Dati

formati ad accesso Casuale (da citare per dovere di cronaca, ma ormai abbandonati) ecc.

– famiglia dei file Grafici:

formati Bitmap, dei quali conservare le dimensioni e le proporzioni (es. riproduzione di quadri)

formati Bitmap, dei quali non conservare le dimensioni e/o le proporzioni

formati Vettoriali, dei quali conservare le proporzioni

altri formati, accessibili con altri Control non standard, ecc.

– famiglia dei file Dati Database (composti da Tabelle, Relazioni, Indici, ecc.):

formati Dati, accesso via DataControl e normali Controlli (TextBox)

formati Dati, accesso via DataControl e Controlli DB (DBGrid, DBcombo, DBList)

formati Dati, accesso via tecniche DAO e programmazione tradizionale.

Quest'ultima famiglia può essere a sua volta suddivisa in sottofamiglie, distinte per tipologia di Database:

formato MS Access, quello standard di VB, raggiunto tramite il motore interno JetEngine

formato dBase, molto diffuso, raggiunto tramite il Driver ISAM a disposizione

formato SQL (o altri), raggiunto tramite il Driver ODBC.

Altra classificazione altrettanto importante è quella relativa alla tipologia di Utilizzo (parliamo sempre di programmi con accesso ai file esterni):

programmi Gestionali, in cui si prevedono molte operazioni di aggiornamento,

programmi di Interrogazione «scemi», in cui le interrogazioni sono sempre le stesse,

programmi di Interrogazione «intelligenti», in cui si impostano in modo

estemporaneo le regole di interrogazione.

Un'altra classificazione, questa volta non «lato dati» ma «lato Visual Basic», è quella che si basa, non sul tipo del file esterno che si deve leggere, ma sul tipo di Controllo che riceve i campi del file dati.

Per i file testuali:

TextBox
Label
ListBox
Grid
controlli aggiuntivi non presenti nella dotazione di base

ecc.

Per i file grafici:

Form
PictureBox
ImageBox
Grid
controlli aggiuntivi non presenti nella dotazione di base.

Per i file dati Database:

DataControl
TextBox
ComboBox, ListBox, Grid, da caricare «a mano»
DBGrid, DBCombo, DBList, da caricare automaticamente.

Insomma, calcolando tutte le combinazioni possibili tra tipo di file dati, tipo di oggetto che riceve i dati, tipo di applicazione, il numero risultante potrebbe arrivare (e se non ci arriva non ci arriva per poco) al migliaio.

Cosa intendiamo fare

Nei numerosi articoli che hanno riguardato, più o meno direttamente, il Visual Basic, abbiamo già trattato qualcuno di questi argomenti, ma mai in maniera così sistematica e definitiva.

Data la vastità dell'argomento, dovuta al fatto che Visual Basic permette svariati metodi di lettura di file e dispone di numerosissime istruzioni dedicate a questo compito, lo dividiamo un due parti. La prima, questa, riguarda la lettura dei file non strutturati, file testuali di qualsiasi tipo (organizzati a campi, organizzati a righe, testi lunghi, ecc.) e di file grafici.

La seconda parte invece sarà dedicata ai file strutturati (ad esempio file in formato DBF, in formato Access, ecc.) e riguarderà l'utilizzo (... anche spregiudicato) del DataControl e lo sfruttamento della tecnologia DAO.

Realizzeremo una serie di programmi, dei veri e propri esercizi, che avranno le seguenti caratteristiche:

potranno essere sviluppati con Visual Basic, qualsiasi versione (salvo uno)

usano file dati facilmente realizzabili con il Blocco Note o facilmente reperibili in giro

necessitano di un codice molto corto (nel caso peggiore riempie un videata)

sono completamente documentati nelle figure a corredo dell'articolo

ogni figura è un collage che mostra codice, file dati e l'applicazione in fase di esecuzione.

Insomma chiunque di voi (a meno che non sia colpevolmente pigro) potrà ricostruire l'applicazione in pochi minuti. Personalmente ritengo questo sistema didatticamente molto valido, specie con il Visual Basic, nel quale ci sono strumenti di Debug, sistemi di prova passo passo, ecc. che possono servire per capire ogni singolo elemento che costituisce il programma.

Molto utile è il sistema di realizzare l'applicazione un pezzetto per volta, per provare il funzionamento del singolo pezzetto prima di passare al secondo pezzetto.

Due chiacchiere sul codice VB

È il caso di spendere due parole sull'interesse che sta assumendo il Visual Basic.

Come noto, in attuazione della strategia della Microsoft, il Visual Basic sta diventando il linguaggio comune di programmazione dei vari applicativi. Questo è già vero per Excel, Access e Project.

In termini pratici questo significa varie cose, la più evidente delle quali è la trasportabilità del codice da un'applicazione ad un'altra. È chiaro però che se usiamo, ad esempio, il codice dell'esercizio numero 2 nel Visual Basic di Excel dovremo cambiare tutte le istruzioni che riguardano la visualizzazione dei dati, che non sarà più delegata alle TextBox di una Form VB, ma alle celle di Excel. Altro discorso, più tecnico, riguarda le tecniche OLE di cui parleremo prossimamente.

Vai con gli esercizi

Passiamo alla parte pratica dell'articolo cominciando con una serie di esercizi dedicati alla lettura dei file testuali.

Lettura di dati sparsi

Supponiamo di aver realizzato un'applicazione da distribuire a molti utilizzatori per ciascuno dei quali occorra predisporre una specifica configurazione, che riguardi sia aspetti esteriori dell'applicazione, come le famose «preferences», sia aspetti più interni, come le impostazioni di configurazione.

Ci sono migliaia di modi per risolvere questo problema, uno di questi è quello di creare un filetto testuale con i vari parametri, da leggere, uno per uno, alla partenza dell'applicazione.

Si può trattare di parametri di tipo testuale, oppure di tipo numerico. Nell'esercizio, documentato con listato e Form in figura 1, ve ne proponiamo un esempio. Vediamo, in basso nella figura, il file testuale, a sinistra il programma di lettura, a destra il risultato ottenuto.

Le istruzioni fondamentali sono:

`Open <nomefile> For Input As #1` per aprire il file

`Input #1, A` per leggere il valore che viene assunto dalla variabile A

`Close #1` per chiudere il file aperto.

L'istruzione Input provoca l'automatico posizionamento sul dato successivo, cosicché non occorre scrivere una istruzione di spostamento. Quindi

`Input #1,A, B` che legge due valori ed è identica alla

```
Input #1,A
Input #1,B
```

Lettura di un File «comma delimited»

Quando si parla di lettura di file in genere ci si riferisce alla lettura di un file dati.

Il problema diventa quello di scrivere le istruzioni in modo che siano adatte ai file in lettura.

Uno dei tipi più utilizzati di formato file è quello cosiddetto «comma delimited», nel quale i vari campi sono separati da virgole e nel quale le stringhe sono racchiuse tra virgolette (in basso a destra nella figura 2). In questi casi occorre preparare una istruzione Input con una sequenza di variabili adatte alla sequenza dei campi.

Altro problema è quello di leggere via via tutte le righe e di fermarsi quando le righe sono finite.

Quando le righe sono finite può essere rilevato dalla funzione logica Eof(1), che restituisce il valore True

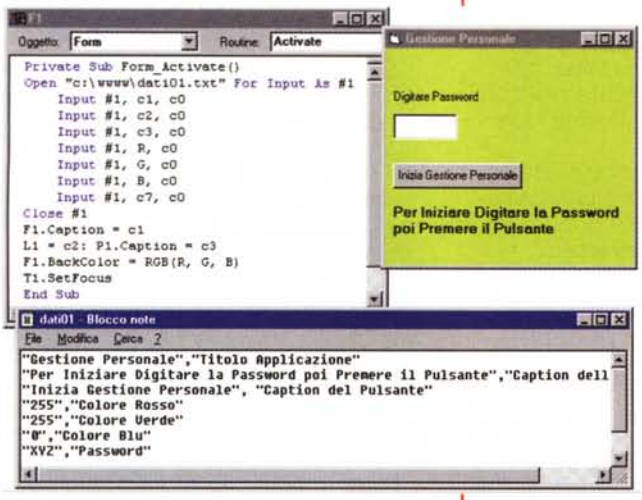


Figura 1 - VB4 - Un facile sistema per creare un File di Inizializzazione.
Supponete di dover realizzare un'applicazione da distribuire a decine di utilizzatori e che ogni copia di questa applicazione debba essere personalizzata. Si può prevedere la memorizzazione dei vari elementi che contraddistinguono la singola personalizzazione in un file testuale, facile da confezionare, anche con un normale editor, facile da leggere alla partenza dell'applicazione. Si possono memorizzare elementi testuali, caratteristiche fisiche ed estetiche della personalizzazione, ed altro ancora.

quando i dati sono finiti. La struttura del programma diventa quindi.

```
Open <nomefile> For Input As #1
  Do While Not Eof(1)
  Input #1, A, B, C, ..
  Loop
  Close #1
```

Questo programma legge, in modo sequenziale, tutti i dati, dal primo all'ultimo. L'errore più frequente si verifica quando si cerca di leggere i dati quando questi sono finiti. L'errore è Input oltre la fine del File (nella versione italiana di Visual Basic).

Il problema immediatamente successivo è quello di decidere cosa fare dei dati letti.

Se si decide, come nel nostro caso, di scaricarli in una griglia il programma si complica per il semplice fatto che le istruzioni per gestire l'indirizzamento delle celle della lista sono molte di più di quelle che servono per aprire il file e leggere i dati.

L'istruzione AddItem, che vedete nel listato, serve per aggiungere una riga alla griglia senza... dar fastidio a nessun'altra riga.

Leggere un elenco di voci da un file

Altro ottimo esempio di utilizzazione delle istruzioni di lettura delle righe di un file sequenziale è quando occorra alimentare una ListBox, con il metodo AddItem.

Nell'esercizio, che vi proponiamo in figura 3, vediamo anche come gestire l'inserimento e la cancellazione di Item. Alla fine dobbiamo anche scrivere, nello stesso file che abbiamo letto all'inzio, il nuovo elenco.

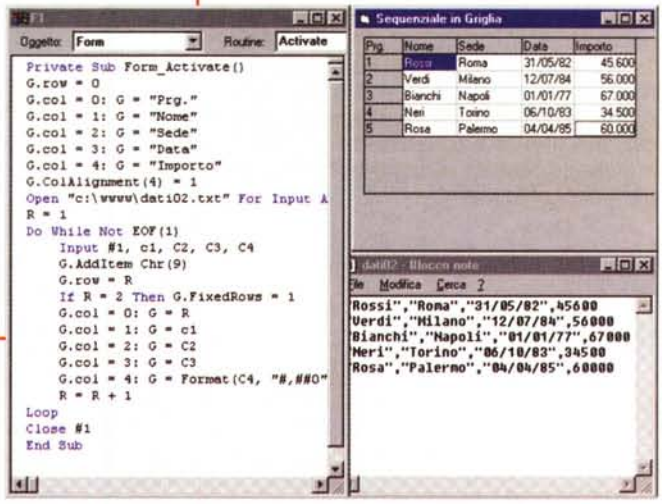
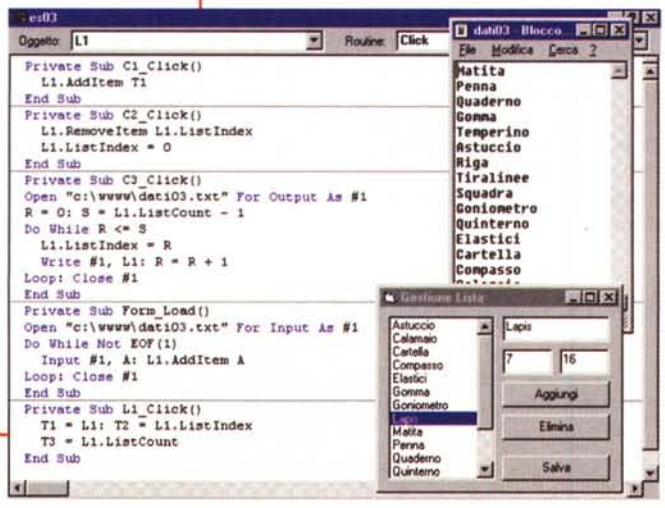


Figura 2 - VB4 - Lettura di un file testuale di tipo «Comma Delimited».
Quello che vedete in basso a sinistra è un file «comma delimited». Contiene record (la riga) e campi. I campi sono separati da virgole, i campi alfanumerici inoltre sono delimitati da virgolette, i campi numerici invece no. La routine di lettura, basata sull'istruzione Open, è in grado di leggere i vari campi separandoli gli uni dagli altri.

Figura 3 - VB4 - Caricamento e gestione di una Lista per una ListBox.
Supponiamo che in un'applicazione occorra scegliere un elemento da una lista non molto lunga (max. qualche decina di componenti). In questo caso può essere conveniente memorizzare gli elementi della Lista in un file Testuale, semplice da costruire, semplice da memorizzare e semplice da caricare all'occorrenza. Anche nel caso in cui occorra prevedere di aggiungere e/o togliere elementi lo si può fare facilmente, realizzando una semplice routine di scrittura, simmetrica rispetto a quella di lettura.



Le cose da vedere sono, oltre alla solita routine di lettura, la simmetrica routine di scrittura:

```
Open <nomefile> For Output As #1
  ...
  Write #1, A
  ...
  Close #1
```

E le varie proprietà che consentono di scorrere e di leggere tutte le righe della Lista:

`L1.ListCount` che fornisce il numero degli elementi della lista.

`L1.ListIndex` che legge o imposta il progressivo della lista.

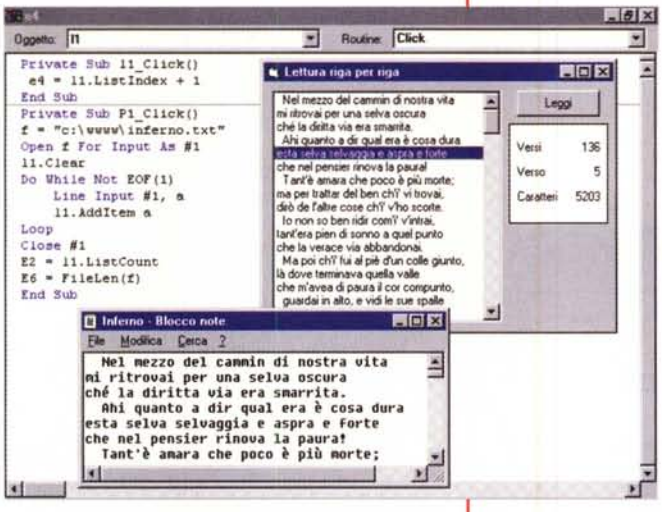
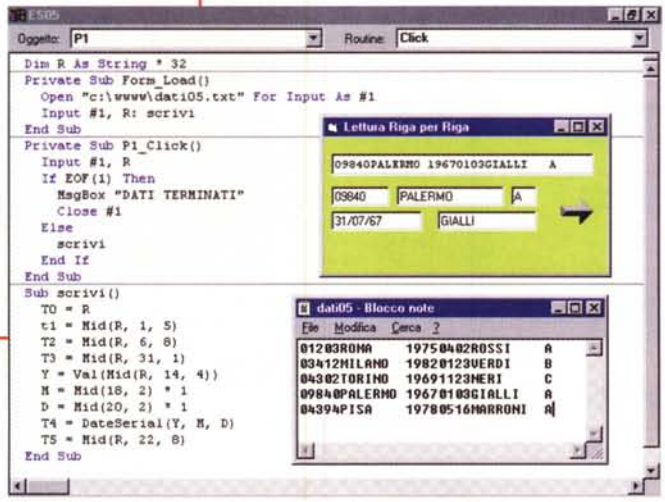


Figura 4 - VB4 - Lettura delle righe intere.
In altre situazione occorre leggere le righe intere. In queste righe le virgole non debbono essere interpretate come separatori ma come componenti del testo a pieno titolo, così come gli altri caratteri... strani. Nel caso di una composizione in versi la riga assume una sua individualità al punto che può essere caricata, con le altre, in una ListBox che dispone di una sua proprietà per contare le righe e per sapere quale è la riga selezionata. Non approfittarne, per sapere quale sia il verso del Primo Canto dell'Inferno su cui ci si è posizionati, sarebbe un delitto.

Figura 5 - VB4 - Lettura del formato Fixed Lenght.
Questo formato deve la sua (immeritata) fama al fatto che è un formato facilmente prodotto dai Mainframe e da tutti i programmi di stampa che permettono di stampare su file anziché su carta. Il programma ricevente deve affrontare il problema (peraltro facile da risolvere) di scomporre la riga nei suoi pezzetti, ovvero nei vari campi. Ovviamente occorre conoscere l'organizzazione del file (in termini tecnici si chiama struttura del record).



Lettura di righe intere

Abbiamo visto come sia possibile leggere un file «comma delimited». In altri casi la virgola non è un separatore di campi ma va letta come un qualsiasi altro carattere.

Vi proponiamo la lettura della Divina Commedia, che è piena di virgole e di altri caratteri strani, come accenti, apostrofi, ecc. In questi casi l'istruzione più indicata è la:

```
Line Input #1, A
```

che legge la riga tutta intera, fino alla fine della riga stessa, identificata da un carattere CR (Carriage Return, o più banalmente Andata a Capo).

Se le varie righe si caricano in una ListBox è possibile, in un certo senso, conservarne il numero progressivo, in quanto il numero della riga corrisponde al valore della proprietà ListIndex (+1, s'intende, in quanto l'indice parte da 0) della Lista.

Lettura di un File Fixed Lenght

Altro formato molto diffuso, per vari motivi citati nella didascalia della figura 5, è il FixedLenght. Per leggerlo si usano i sistemi visti fino ad ora, con la semplificazione che si legge una sola variabile e con la complicazione che questa riga va scomposta nei vari campi (l'organizzazione della riga va conosciuta). Per farlo basta usare la funzione Mid.

Nell'esercizio, di cui vi mostriamo la soluzione, leggiamo una sola riga, e la lettura della riga successiva la eseguiamo al Click sul pulsante con la freccia.

Vale la pena di ricordare che nei file sequenziali... non si torna indietro.

La caratteristica principale del file sequenziale è che si legge dal primo record fino all'ultimo senza possibilità di tornare indietro.

Al massimo si può cadenzare, come in questo caso, la lettura, oppure, raggiunto il record desiderato, si può interrompere la lettura.

Lettura di un File lungo...

Ecco ora la lettura di un file testuale eseguita... tutta di un colpo. L'intero file viene letto tutto insieme e trasferito su una variabile di tipo stringa, opportunamente dimensionata. L'istruzione di lettura è la:

```
Input <nomefile> For Binary #1
Get #1, , A
Close #1
```

Il parametro mancante nell'istruzione Get è quello che permette di leggere il file a partire da un certo carattere. La variabile A va dimensionata opportunamente, es.

```
Dim A As String*20000
```

significa una stringa lunga di 20.000 caratteri.

Esiste anche una funzione LenFine(<nomefile>) che riporta la lunghezza del file, in byte, e che può essere usata per dimensionare esattamente la variabile.

Il limite teorico di tale procedimento è di 64kbyte, mentre quello pratico, che dipende anche da altri fattori è di circa la metà.

Lettura di un File formattato

In Windows 95 il formato RTF, Rich Text Format, per i file testuali diventa molto importante.

Detto in parole povere è un formato in cui oltre al testo viene memorizzata

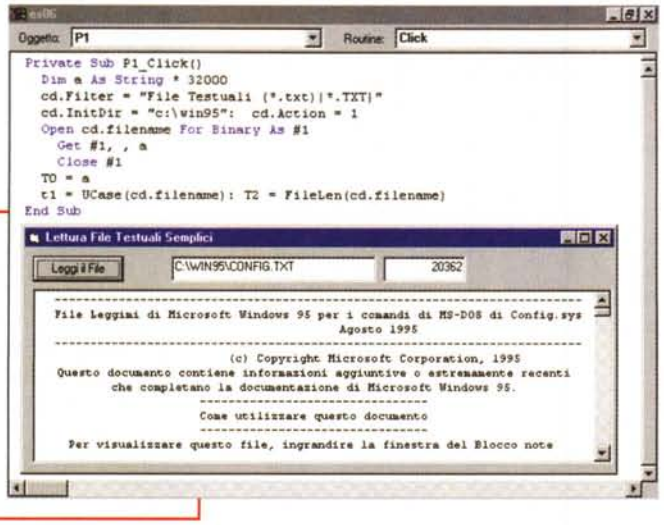


Figura 6 - VB4 - Lettura di un file lungo... ma non troppo. Ulteriore problematica è quella che riguarda la lettura di un file testuale testuale, da leggere tutto in blocco, senza separatori né di campo né di riga. Esiste un'apposita variante del comando Open (Open For Binary) che trova il suo limite nella dimensione della variabile alla quale passare il file testuale.

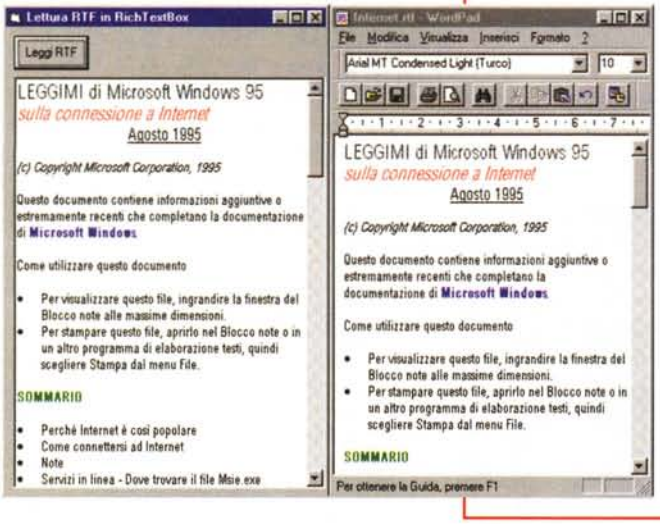


Figura 7 - VB4 - Lettura di un file RTF. Uffa, che noia.. occorre imparare un nuovo formato! Usando Visual Basic 4 in modalità 32 bit (la versione per Windows 95 quindi) è possibile utilizzare un nuovo controllo, la RichTextBox, che permette di caricare un file in formato RTF, quindi un testo formattato. È chiaro che per poter sfruttare bene questo nuovo oggetto occorre conoscere le possibilità offerte da questo controllo e occorre conoscere quali siano le caratteristiche «internal» del formato RTF. Ad esempio pare che il formato RTF non contenga le specifiche relative alla formattazione della pagina, ma solo quelle relative alle parole e ai paragrafi. Questo significa che occorre formattare localmente la RichTextBox.

anche la formattazione (attenzione del testo, non della pagina). WordPad, ad esempio, salva in formato RTF e Visual Basic legge (e visualizza) tale formato in un apposito Controllo, che si chiama RichTextBox, che dispone, tra le altre cose, di un apposito metodo:

```
RTBox.LoadFile(<nomefile>),
rtfRTF
```

Lettura di un File ad Accesso Casuale

Ai tempi dei primi Basic esistevano solo due modi per accedere ai Dati, il primo dei quali è quello visto fino ad ora, con le sue numerose varianti, che si chiama accesso Sequenziale perché comporta la lettura di tutti i dati, nel senso che non si può leggere il secondo se non si è letto anche il primo.

Insomma anche se si desidera raggiungere il milionesimo, occorrere leggere i primi novecentonovantanovemila e rotti.

L'altro sistema è l'Accesso Casuale. Si può leggere un record indipendentemente da tutti gli altri a patto che se ne conosca l'indirizzo (costituito da un numero identificativo), si conosca la lunghezza, che in questo caso deve essere fissa, del record e che si dichiari la struttura del Record.

Questa dichiarazione va fatta in un Modulo BAS e si basa sulla definzione di una variabile di tipo RecordType che consente, sostanzialmente, di riferirsi ad una sola variabile anziché al record completo.

Ad esempio RV è la variabile «record», RV.Cognome è la variabile «campo».

Nell'esercizio, che vi proponiamo per «toccare con mano» l'argomento, facciamo tre cose:

leggiamo il nostro file sequenziale e lo trasferiamo in un Array

lo trasferiamo dall'Array al file Random

ne permettiamo lo scorrimento sfruttando una ScrollBar i cui limiti sono 1 (quello minimo) e il numero di record scritti (quello massimo).

Le istruzioni le trovate riportate integralmente nella figura, mentre il file dati è lo stesso dell'esercizio di figura 2. Rieseguendo l'esercizio ne capirete tutte le implicazioni.

Lettura del contenuto di una Directory e visualizzazione dei file Grafici

Un esercizio facilissimo da realizzare ma molto spettacolare è quello che mette insieme il controllo File, quello che mostra una lista di file di una directory (ma che è una Lista a tutti gli effetti), e un controllo Picture che, con la sua funzione LoadPicture, è sempre pronto a visualizzare un'immagine.

Nell'esercizio, propostovi in figura 9, vediamo tre cose:

ciò che serve a fare in modo che la lista File punti una Directory con dei file grafici (proprietà Path e proprietà Pattern), come fare a visualizzare nella Picture il file il cui nome è puntato nella lista File,

una routine, eseguita dal pulsante Scorri Tutti, che esegue un ciclo che punta via via tutti gli Item della lista e che li fa vedere nella PictureBox.

Va precisato che l'evento Click su una lista si ottiene sia, ovviamente, facendoci click sopra, sia spostandosi su e giù nella lista con i tasti freccia, sia impostando dal fuori il numero dell'Item da puntare (ad esempio l'istruzione L1.ListIndex = 3 produce un evento L1.Click).

Lettura di un File Grafico e gestione dinamica delle sue dimensioni

La funzione LoadPicture può essere usata per alimentare sia una PictureBox, sia una Image, sia una cella di una Grid (lo sapevate?), sia una Form.

Vi proponiamo un esercizio che mostra come caricare un'immagine in un controllo Image (che si chiama L1), scegliendola con una finestra Apri File, attivata da un Controllo Common Dialog Box (che si chiama CD).

Altra sperimentazione proposta nell'esercizio è quella sull'evento Resize.

Lo usiamo per ridimensionare l'Immagine dinamicamente al ridimensionamento della Form. Occorre in pratica legare la proprietà Width della Image alla proprietà ScaleWidth della Form. Lo stesso dicasi per L1.Height e Form.ScaleHeight. Il ridimensionamento deve comunque lasciare libera una zona, in alto nella Form, per i pulsanti.

Figura 8 - VB4 - Lettura di File ad accesso casuale.
Ogni volta che si parla di accesso ai file non si sa se parlare anche di questo metodo di accesso, oggi assolutamente improponibile (le tecniche basate sul DataControl e su DAO sono ben più efficenti). Poiché era presente già nei primi Basic è stato sicuramente lasciato per garantire la compatibilità con precedenti programmi. Per poter accedere direttamente ad un record, senza dover leggere necessariamente tutti i record precedenti, occorre che il file abbia una struttura fissa, che questa sia dichiarata in un file di tipo Modulo (BAS) e che ne sia dichiarata, nelle istruzioni di apertura, la lunghezza. In questo esercizio, che legge il Sequenziale e scrive il Casuale, usiamo lo stesso file dati usato nell'esercizio 2.

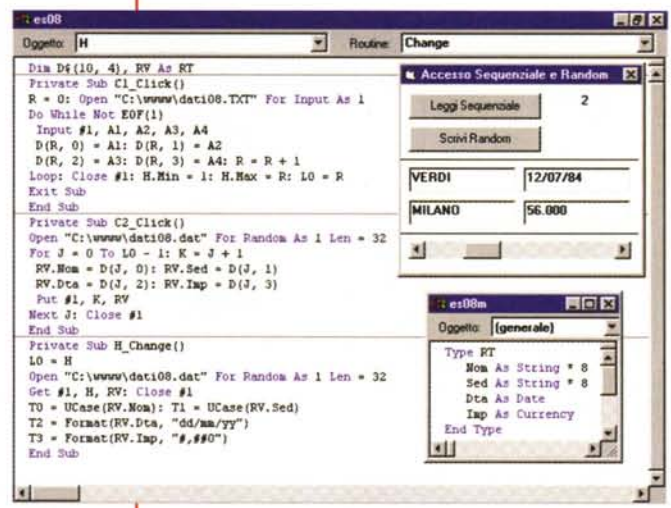


Figura 9 - VB4 - Combinazione tra una Lista di tipo File e una Picture Box.
Una Lista di tipo File è una lista a tutti gli effetti, solo che gli Item mostrati sono File di una Directory. Dispone di tutte le proprietà e di tutti i metodi di una Lista normale. Se la facciamo puntare ad una Directory che contiene file grafici possiamo fare in modo di vederli (i file grafici) in una PictureBox, scegliendoli nella lista oppure facendo scorrere vorticosamente la lista stessa con un ciclo.

nel successivo esercizio. Si legge il file, che contiene, in un certo senso in maniera codificata, i dati necessari per realizzare un Diagramma.

Nel file dati di esempio (sulla sinistra in figura 11) abbiamo inserito alcuni Testi fissi, in pratica i vari titoli, e poi i due valori (4 e 4 nel nostro esempio) che indicano il numero delle Serie ed il numero dei Valori per Serie.

La routine di lettura, dopo ciò che abbiamo visto fino ad ora, non presenta nessuna difficoltà. Passiamo, ma solo per poter controllare meglio i dati, per una Griglia, e poi realizziamo il Diagramma finale.

La scelta del tipo di grafico invece la facciamo non leggendo un dato dal file, ma dalla nostra applicazione, proponendo la lista dei tipi permessi dal Controllo Chart.

In pratica proponiamo un'integrazio-

Passaggio di Dati tra File, Griglia e Chart

Un utilizzo nobile delle istruzioni di lettura di un file sequenziale è proposto

Figura 10 - VB4 - Caricamento di file grafico in un controllo Image e suo ridimensionamento.
Vediamo come caricare un'immagine in un controllo di tipo Image, utilizzando la CommonDialogBox File Apri, e come variare le dimensioni della Form e contemporaneamente quelle dell'Image. Occorre districarsi tra dimensioni della Immagine (come file) dimensione della Form, dimensione dell'oggetto Image, sistema di riferimento, ecc. L'evento da gestire è la Resize della Form.

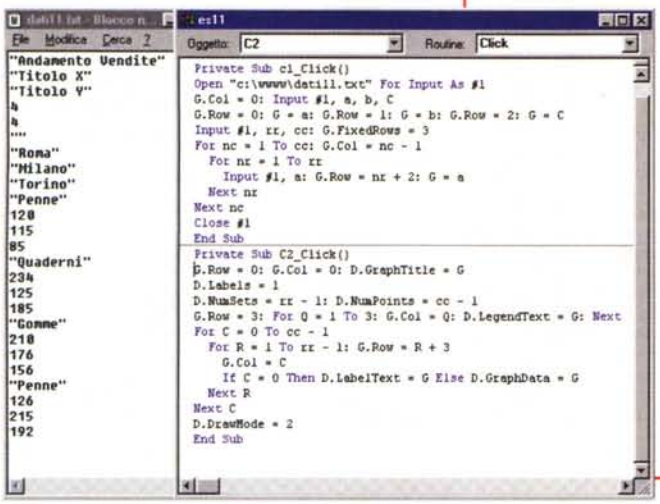


Figura 11 - VB4 - File testuale - Lettura in Griglia - Generazione del Grafico.
Ecco un altro esempio di applicazione delle istruzioni di lettura di un file testuale. Questa volta il file è misto, contiene titoli, legende, etichette e dati numerici necessari per la realizzazione di un grafico a barre. Alcuni di questi dati hanno, nel file testuale, una posizione fissa (ad esempio i titoli), ci sono poi i due numeri indicanti numero di righe e numero di colonne (Serie del grafico e Valori per Serie).

E per finire... un pizzico di OLE

Vi proponiamo, per finire in bellezza, un esercizio nel quale da Visual Basic creiamo un oggetto OLE di Excel, al quale passiamo dei dati, e al quale passiamo dei comandi. Il problema è se tutto questo è un argomento fuori tema oppure no.

OLE che c'azzecca con la lettura dei File?

Il terzo pulsante, il C3, che non ha figure che ne visualizzano l'effetto, apre come oggetto OLE un file di Excel, esistente, in cui c'è una Macro (già scritta) che viene lanciata direttamente dal codice VB.

ne tra dati letti da file e opzioni scelte localmente e dinamicamente.

Excel VBA come VB

Il penultimo esercizio consiste nel verificare la trasportabilità del Codice VB in una Macro scritta con il VBA di Excel.

Le istruzioni, che abbiamo riportate in una cornice sul foglio che riceve i dati letti, sono le stesse dell'esercizio 2, del quale abbiamo anche usato il file dati.

Variano solo le istruzioni di visualizzazione che con il VB finalizziamo ad una serie di TextBox o ad una Griglia e che con il VBA di Excel indirizziamo verso le celle del foglio.

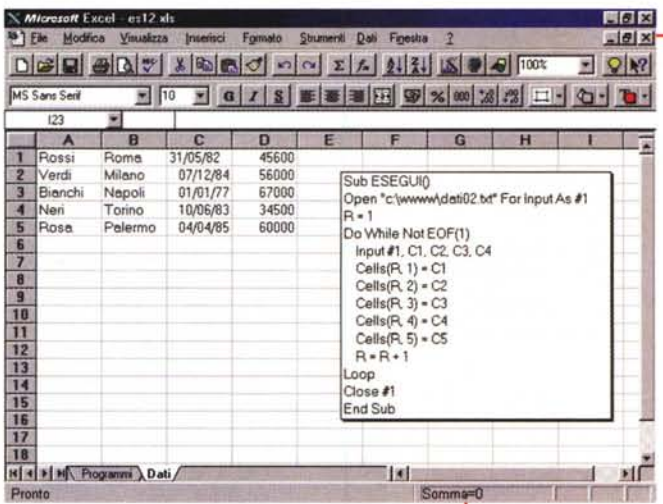


Figura 12 - Excel VBA - Riutilizzo della routine dell'esercizio 2 in una Macro Excel VBA.
Visual Basic sta diventando il «linguaggio comune» di tutti gli applicativi «intermedi» della Microsoft. Vanno esclusi i prodotti Soho (Small Office e Home) e i prodotti evoluti di sviluppo (da Visual C++ in su). Vediamo come lo stesso identico codice utilizzato nell'esercizio 2 sia riutilizzabile in una Macro VBA di Excel. Vanno cambiate solo le istruzioni che servono per trasferire i dati letti dal file nelle celle del foglio.

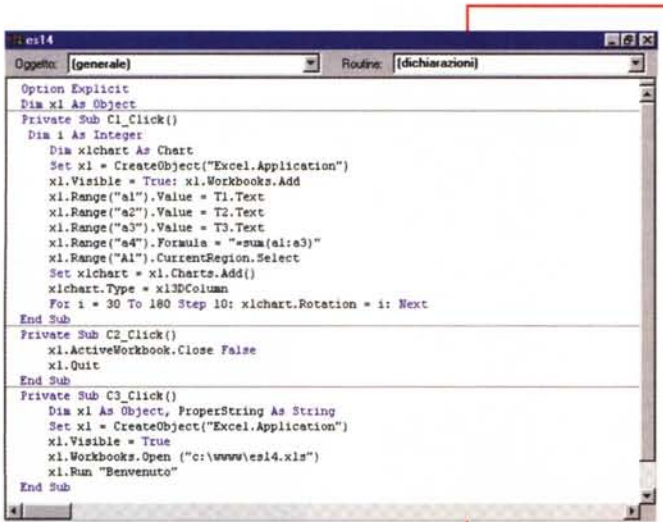


Figura 13 - VB4 - Manipolazione di un oggetto OLE da Visual Basic. Altra frontiera aperta è quella costituita da OLE, ovvero, in Visual Basic, dalla possibilità di incorporare nella propria applicazione «oggetti» realizzati con un'altra applicazione e di gestirli direttamente come fossero propri oggetti. Vediamo un programma Visual Basic che fa tre cose. Lancia Excel, gli passa dei valori numerici e gli fa realizzare un grafico tridimensionale (che si mette a girare). Chiude Excel. Infine crea un oggetto XL nel quale apre un'applicazione che a sua volta contiene una Macro che viene lanciata dalla procedura VB.

Conclusioni

Scusate, appena si comincia a parlare di OLE... mi comincia a far male la testa.

Scherzi a parte... OLE è un altro film, del quale ogni tanto vediamo degli «spezzoni» impressionanti, che però non ci consentono mai di farci un'idea completa ed esauriente, in altre parole soddisfacente, dell'insieme.

Di OLE dovremo prima o poi parlare a fondo, magari proponendovi anche in quel caso 1.000 esercizi, 1.000 spezzoni da montare per avere un film completo.

MC

STA NASCENDO QUALCOSA DI UNICO!

PER SAPERNE DI PIÙ ASPETTA IL PROSSIMO NUMERO

HOBBYTRONICA

HOBBYTRONICA: IL PRIMO SALONE DELLA MULTIMEDIALITÀ PER LA FAMIGLIA.

DOVE PER TUTTI IL FUTURO E' IL PRESENTE.

20 - 24 NOVEMBRE 1996
TORINO, LINGOTTO FIERE

LIN GOT TO

Lingotto Fiere

EXPO 2000 S.P.A. - VIA NIZZA, 294 - 10126 TORINO
TEL. 011/664.4111 - FAX 011/664.6642
E-mail : dev@lingottofiere.it

RISERVATO AGLI ESPOSITORI:

Azienda_____

Indirizzo_____

Settore Merceologico_____

Tel._____

Fax_____