di Francesco Petroni

Visual Basic Intermedio
Array, Collezioni ed affini

Continuiamo la nostra serie di articoli dedicati agli utilizzatori del Visual Basic, articoli di "taglio intermedio" pensati per una precisa categoria di utilizzatori, non quelli alle prime armi nè quelli già esperti, ma quelli che stanno in mezzo, che hanno già le conoscenze di base e che vogliono approfondire solo alcuni temi particolari. Gli esperti invece già dovrebbero conoscere gli argomenti trattati e quindi dovrebbero anche essere in grado di svolgere facilmente gli esercizi che corredano l'articolo... ma non è detto che lo siano.

Gli articoli sono prevalentemente pratici, nel senso che propongono una serie di esercizi facilmente rieseguibili da ciascuno di voi. Anche in questo secondo articolo ogni esercizio corrisponde ad una figura che mostra sia il listato sia, in primo piano, il Form (da questo articolo usiamo il maschile) che costituisce l'aspetto esteriore dell'applicazione. Il primo articolo, apparso sul numero scorso di MC, trattava di Liste, Griglie ed affini, ovvero dei componenti adatti a mostrare insiemi organizzati di dati. In questo secondo articolo parliamo di Matrici, di Collezioni ed affini.

E' comunque evidente che anche in questi nuovi esercizi continueremo ad usare Liste e Griglie necessarie per ospitare e per visualizzare i nostri dati, per cui diamo per scontato che abbiate letto l'articolo del numero scorso oppure che sappiate già usare correttamente questi oggetti.

Seconda parte

Le Matrici

Non intendiamo certo spiegare il concetto di matrice ad un utilizzatore "intermedio" di un linguaggio di programmazione, che già le conosce alla perfezione, vogliamo solamente proporre un paio di esempi di loro utilizzazione. Nel primo (figura 1) vediamo un file testuale semplicissimo, letto con la classica istruzione Open e scaricato in una matrice 12 per 3, 12 righe con i dati riferiti a ciascun mese e 3 colonne con il nome del mese, un dato testuale ed un dato numerico. Se scarichiamo anche in una ListBox la prima colonna di valori possiamo facilmente sincronizzare l'indice della lista (proprietà ListIndex, che parte da 0) con l'indice della Matrice (che parte anch'essa da 0) in modo che selezionando il mese vengano visualizzati anche i corrispondenti valori testuale e numerico.

Nel secondo esercizio (figura 2) usiamo una matrice (MT) un pò più grande, in grado di ospitare 500 righe e 2 colonne), nella quale carichiamo un valore di una Matricola e quello di un Cognome. Carichiamo i dati, solo al fine

di una loro visualizzazione, in una ListBox. Per estrarre direttamente (senza dover scorrere tutti i dati) il singolo dato dobbiamo necessariamente utilizzare l'indice numerico della matrice. Anche in questo caso è possibile sincronizzare indice della matrice con indice della lista.

Uno dei "difetti" delle matrici consiste nella difficoltà di caricarle e scaricarle dinamicamente, ad esempio se si debba inserire o eliminare "al volo" un elemento. Occorre usare le Collezioni.

Una Collezione di dati è più maneggevole di una Matrice di dati

Il terzo esercizio è un'evoluzione del precedente. I dati non vengono inseriti in una matrice ma in una Collezione, oggetto che permette sia il caricamento degli elementi (Item) che l'assegnazione di una chiave (Key) all'elemento stesso.

Innanzitutto occorre creare una classe e definire un prototipo dell'oggetto che caricheremo nella collezione. Nel nostro caso abbiamo creato un modulo di Classe che abbiamo chiamato ES5C, all'interno del quale abbiamo semplicemente definito una variabile pubblica **IC**.

```
Public IC
' dichiarazione di variabile pubblica
Public ID
' dichiarazione di eventuale altra variabile
```

Le istruzioni per gestire la collezione sono:

```
Dim MC As New Collection
' dimensionamento della collezione
  Set II = New ES5C
' nuova istanza della classe
II.IC  = "valore da caricare"
' assegnazione di un valore
 II.ID  = "valore da caricare"
' assegnazione di un altro eventuale valore
```

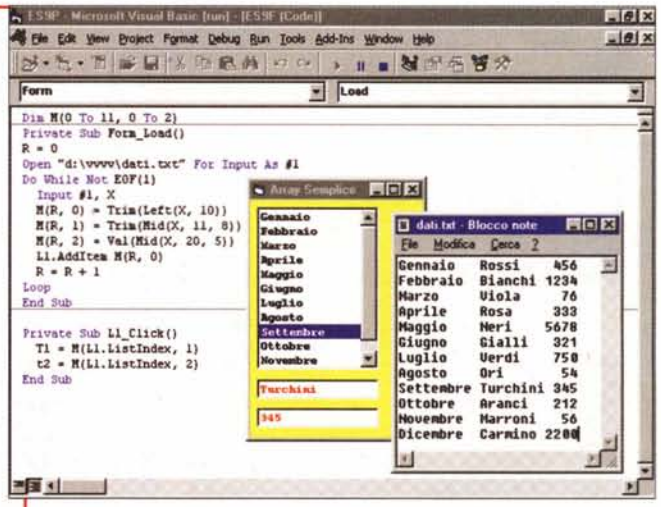


figura 1 - Visual Basic 5.0 - Riempimento di una Matrice con dati letti da un file testuale In questa figura vediamo tre cose: in basso a sinistra il Notepad di Windows con il quale abbiamo realizzato il piccolo file testuale usato per il nostro esercizio; sullo sfondo il Programma con il quale viene letto il file e vengono scaricati i dati in una matrice (12 righe per 3 colonne); in mezzo la nostra applicazione che comprende un Form che mostra la lista dei mesi (letti dal file testuale). Scelto un mese nella ListBox nelle due TextBox appaiono i due dati, un cognome ed un numero, associati a ciascun mese.

```
MC.Add item := II, key := CStr(chiave)
' inserimento del nuovo Item nella collezione MC
```

Importanti sono, oltre al metodo Add, il metodo Remove e la proprietà Count:

```
MC.Count
' restituisce il numero degli elementi
MC.Rmove(key)
' elimina un elemento data la sua chiave
```

Importantissimi sono i due sistemi per selezionare un elemento. Si passa o il suo indice numerico, supposto noto, o la sua key, che invece è una stringa. L'istruzione è:

```
Set XX = MC.Item(indice)
' selezione da indice
```

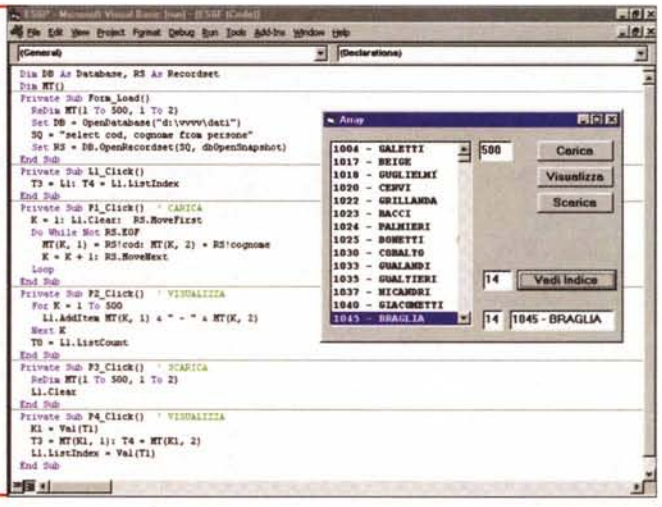


figura 2 - Visual Basic 5.0 - Riempimento di una Matrice con i dati letti da un file MDB attraverso DAO In questo secondo esercizio riempiamo una matrice un po' più grande (500 righe e 2 colonne) con due campi, una Matricola (stringa numerica da 4 caratteri) ed un Cognome, letti da una tabella di un Database in formato Access. Via via che leggiamo i dati li "concateniamo" in un'unica stringa che carichiamo in una ListBox. L'unico modo per "ripescare" i dati da una matrice è attraverso il suo indice numerico.

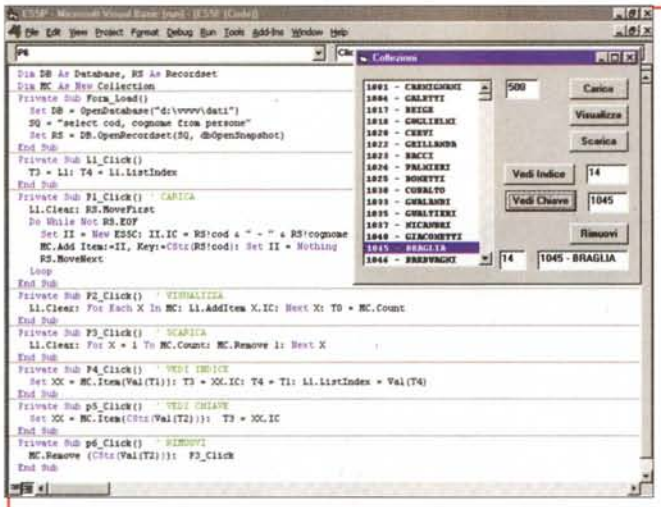


figura 3 - Visual Basic 5.0 - Riempimento di una Collection con i dati letti da un file MDB attraverso DAO
Quello di Collection è un concetto fondamentale che trova numerose applicazioni in ambito Visual Basic. Esistono vari tipi di Collection che fortunatamente si manipolano tutti allo stesso modo. In questo esercizio, meglio descritto nel testo, creiamo una Collection che si alimenta come una Matrice, solo che è un pò più sofisticata in quanto permette di inserire dinamicamente elementi (Item), cosa non possibile con una matrice tradizionale, e di creare chiavi (Key) riutilizzabili per le operazioni di ricerca di un dato, cosa anche questa non possibile con una matrice.

Gli Array di controlli: statici e dinamici

Torniamo agli Array per parlare dei **Controls Array**, ovvero degli insiemi di oggetti, di uguali caratteristiche, inseribili in un Form.

Gli insiemi di controlli si possono realizzare partendo da un controllo iniziale e poi copiandolo più volte. Durante la copia il Visual Basic chiede se si vuole creare un array oppure semplicemente duplicare i controlli lasciandoli indipendenti l'uno dagli altri.

Supponiamo che il controllo iniziale sia una TextBox che si chiama **XX**, nel primo caso alla fine dell'operazione avremo due controlli, uniti in un Array:

XX(0)
XX(1)

nel secondo caso invece i due controlli saranno separati e si chiameranno:

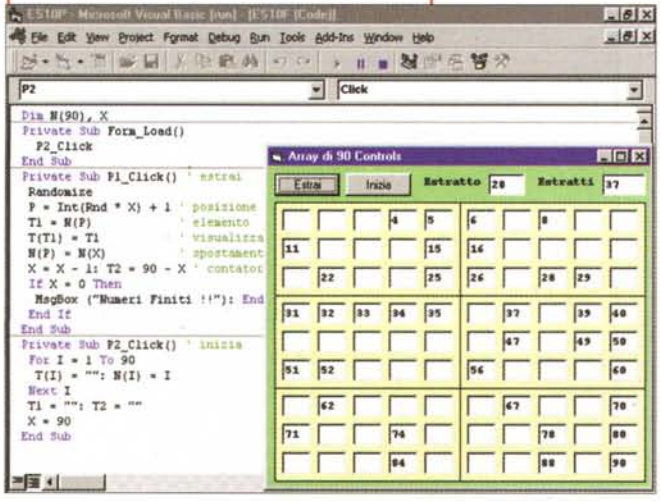
XX

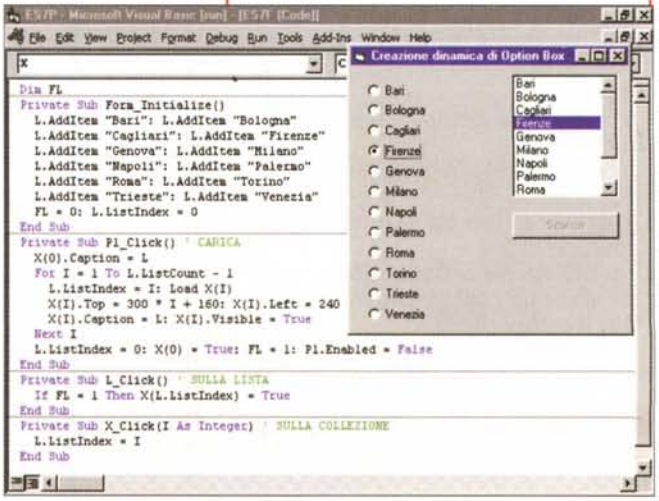
figura 4 - Visual Basic 5.0 - Array di Controlli - Tipo Fisso
Un secondo tipo di matrice è quello realizzabile con i controlli presenti in un Form. In pratica se nel Form si inseriscono tanti controlli (nel nostro esempio si tratta di 90 TextBox) che si debbono comportare tutti allo stesso modo, piuttosto che inserire tanti controlli differenti tra di loro conviene generare una "famiglia" di controlli. Operativamente basta inserire nel Form il primo oggetto, prepararlo a puntino e poi copiarlo, rispondendo SI alla domanda che ci viene posta e cioè se desideriamo realizzare un Array di controlli. Questi hanno tutti lo stesso nome e sono differenziati da un Index, che li identifica individualmente (es. T(0), T(1)..T(n)). Nel nostro esercizio, non inedito, creiamo un tabellone della tombola con estrazione dei numeri, senza possibilità di ripetizione di numeri già estratti.

Set XX = MC.Item(key)
' selezione da chiave
Text1 = XX.IC
utilizzo del valore dell'elemento

La collezione è un pò più difficile da usare di una matrice ma i vantaggi sono numerosi. Oltre alla già citata possibilità di caricare e scaricare dinamicamente i membri della collezione, oltre alla possibilità di richiamare un elemento partendo dal suo indice oppure dalla sua chiave, citiamo il vantaggio di poter caricare nella collezione dati di vario tipo, citiamo la possibilità di usare lo statement For ... Each per scorrere tra i vali elementi.

figura 5 - Visual Basic 5.0 - Array di Controlli creato dinamicamente
Nell'esercizio precedente abbiamo visto un esempio di Array di controlli creati in fase di disegno del Form. In questo esercizio vediamo come sia possibile, in caso di necessità, creare dinamicamente dei controlli di tipo OptionBox. Abbiamo un elenco di nomi di città con i quali alimentiamo una unica ListBox e con il quale creiamo un Array di Controlli (X(i)) di tipo OptionBox (utilizzando l'istruzione Load). Per ogni OptionBox creata occorre definire posizione (Top, Left), scritta (Caption) e il fatto che sia visibile (in quanto la Load la lascia invisibile). Una volta creato l'Array facciamo in modo che selezionando una città nella lista venga selezionata la corrispondente OptionBox e viceversa.





Text1

La proprietà che indica quale sia l'indice dell'oggetto è Index ed è un numero che parte da 0.

Ogni volta che si userà un elemento dell'Array ne andrà specificato l'Index, es.

XX(1) = "valore"

Così pure al verificarsi di un evento l'array permetterà di individuare il singolo elemento che lo ha generato grazie al suo Index.

Private Sub XX_LostFocus(Index As Integer)

..

End Sub

L'esercizio che vi proponiamo non è inedito, si tratta della realizzazione di un Tabellone della Tombola, con le sue novanta TextBox, che nel nostro caso sono un array di controlli.

T(I)

Parallelamente all'array di controlli dobbiamo usare un array di numeri (**N(i)**) nel quale, in fase di inizializzazione, riversiamo tutti i numeri da estrarre, che vanno, notoriamente, da 1 a 90. Siamo anche stati attenti ad allineare l'indice dell'array dei controlli con quello dell'array dei numeri per facilitare le "manovre". In pratica abbiamo eliminato l'elemento con indice 0.

Quando estraiamo un numero usando la funzione RND, ad esempio il numero X, in realtà estraiamo l'X-mo valore della matrice che, una volta estratto, provvediamo a sostituire con il valore dell'ultimo elemento della matrice (che contiene un numero non ancora estratto). Via via che generiamo il numero casuale lo generiamo in un intervallo via via più piccolo in quanto non si tratta del numero estratto ma di una posizione nella matrice, che parte da 90 e diventa via via più piccola.

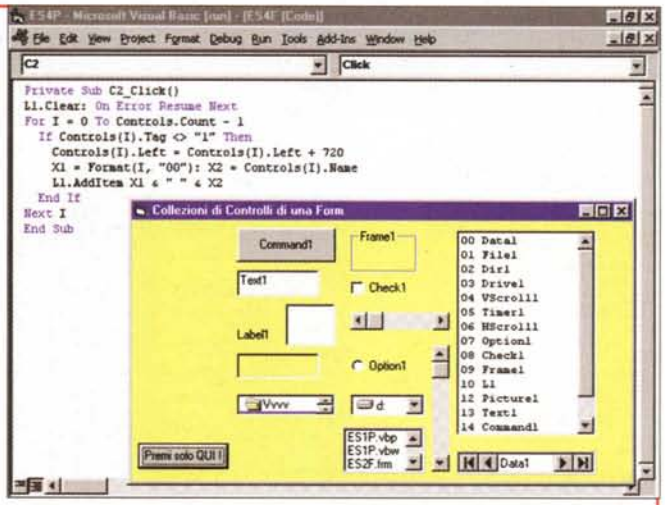


figura 6 - Visual Basic 5.0 - Collection di Controlli in un Form
In un Form esiste una Collection naturale costituita dall'insieme dei controlli in esso presenti. E' possibile contarli (Controls.Count), è possibile manipolarli tutti insieme o per gruppi. Un modo per differenziarli in gruppi consiste nell'utilizzare la proprietà TAG, che è di tipo alfanumerico, e che si può assegnare a piacere a ciascun oggetto per poi leggerla quando occorre. La nostra applicazione crea, nella ListBox, l'elenco degli oggetti in scena e al click sul pulsante "Premi solo qui!!" sposta a destra tutti i controlli esclusi quelli a cui abbiamo assegnato come TAG la stringa "1".

figura 7 - Visual Basic 5.0 - Collection di Form in una Applicazione
Così come in un Form c'è una Collection di Controlli, in un'Applicazione c'è una Collection di Form. In questo esercizio creiamo dinamicamente una Collection di sei Form, partendo da un Form "prototipo" che abbiamo chiamato ESF2. I sei Form "neonati" vengono ospitati in un Form MDI che li visualizza. I Form hanno una Caption che li identifica e contengono una TextBox che mostra un numero progressivo. Inoltre, tramite un Timer piazzato sul Form MDI, viene reso attivo ciascuno dei sei Form ogni secondo.

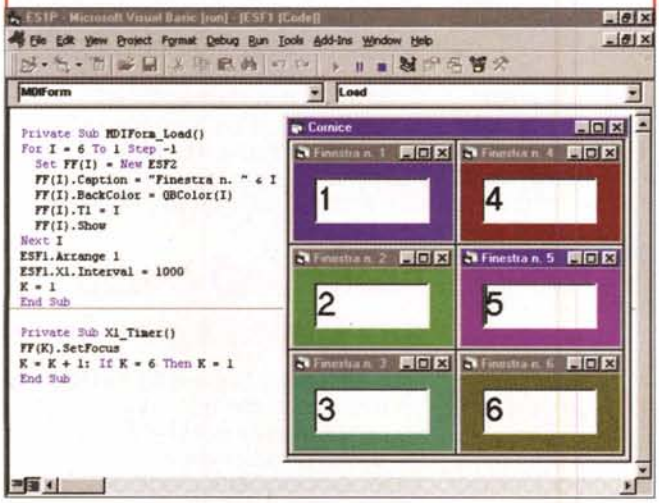
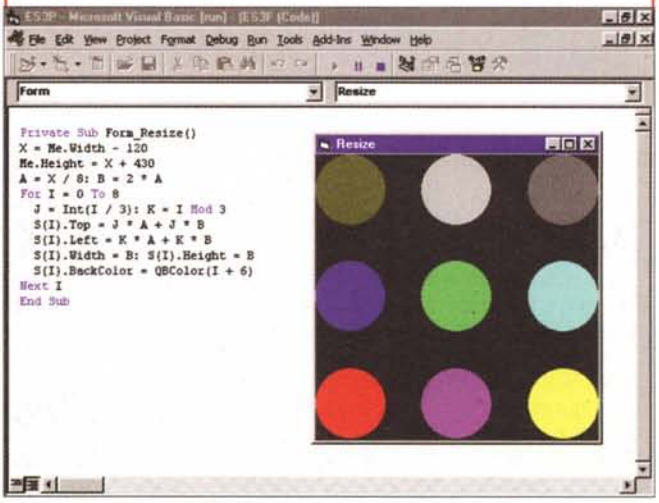


figura 8 - Visual Basic 5.0 - Array di Oggetti "unbound"
Possono essere create Array anche con gli oggetti più ... scemi in assoluto, ad esempio con degli oggetti di tipo Shape. Li definiamo "scemi" in quanto si tratta di oggetti che non sono in grado di intercettare eventi. Nel nostro esercizio i nove cerchietti colorati vengono ridistribuiti nel Form ogni volta che questo subisce un evento Resize, viene cioè ridimensionato con il mouse.



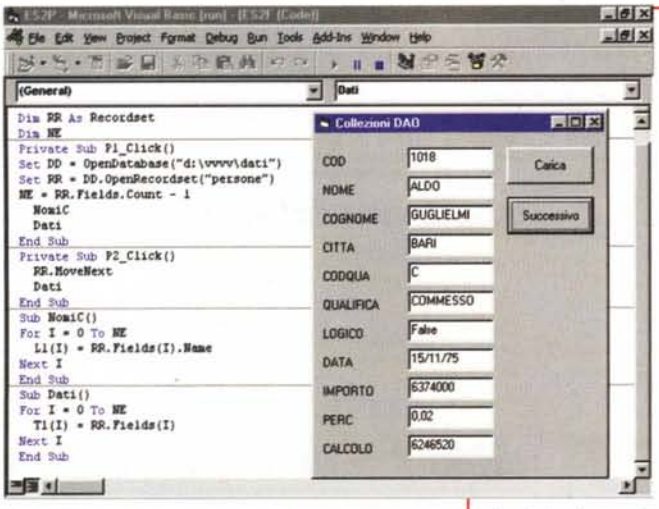


figura 9 - Visual Basic 5.0 - Collezioni di Fields tra gli oggetti DAO
La filosofia delle Collection è stata felicemente utilizzata anche nell'organizzazione degli oggetti DAO (Data Access Object). Nell'esempio in figura utilizziamo un Array di Label (L(n)), un Array di TextBox (T(n)), e due Collection, la prima di FieldsName, che riporta il nome dei campi di un recordset DAO, e la seconda di Fields, che visualizza il contenuto della collection dei campi. Il programma provvede anche a sincronizzare le due Collection con le due Array.

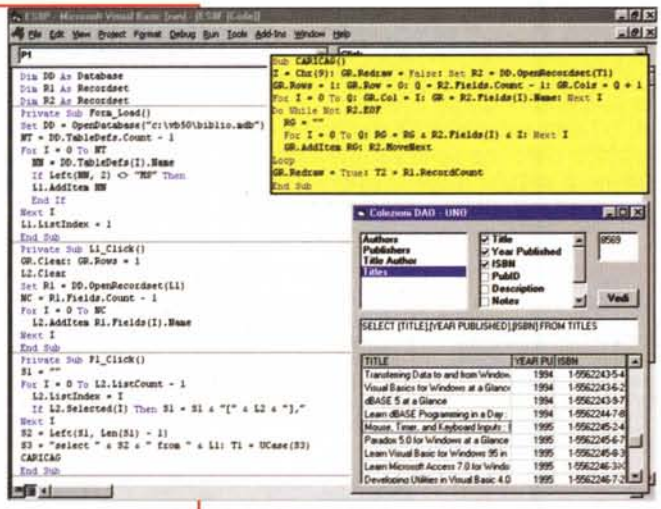


figura 10 - Visual Basic 5.0 - Collezioni di TableDefs e di Fields
Nel modello ad oggetti DAO un Database contiene una Collection di TableDefs (in pratica tutte le tabelle, comprese quelle di sistema) e di QueryDefs (tutte le query salvate). Ogni recordset, e conseguentemente ogni tabella ed ogni query, contengono una Collection di Fields (i campi). Come tutte le collezioni anche quella di TableDefs e quella di Fields hanno la proprietà Count che indica rispettivamente quante tabelle ci sono nel Database e quanti campi nel RecordSet. Abbiamo detto che la collezione delle tabelle comprende anche le tabelle di sistema, quelle che iniziano con i caratteri MSys.

Vediamo il programma, Form e Listato, in figura 4.

Nel successivo esercizio (figura 5) vediamo come in certi casi sia possibile creare dinamicamente un'array di controlli. Nel nostro caso creiamo un Array di OptionBox. Occorre preparare una prima copia del pulsante di opzione da cui partire e, per inizializzare l'indice, impostarne a 0 il valore della proprietà Index.

Come elementi dell'array utilizziamo una lista di nomi di città caricata preventivamente in una ListBox. Il pulsante serve per creare le varie OptionBox "sincronizzandole" con l'elenco delle città. Poi, giocando sull'allineamento della proprietà Index dell'Array con quella ListIndex della ListBox, facendo click su un elemento della lista selezioniamo anche la OptionBox corrispondente e viceversa.

Le Collection gratuite

Per Collection gratuite intendiamo quelle collezioni di elementi che Visual Basic genera comunque, indipendentemente dal nostro intervento, e che in molti casi possiamo sfruttare a nostro vantaggio.

Ad esempio se inseriamo un certo numero di controlli in un Form questi costituiscono comunque una Collection. Se ne possono contare gli elementi, usando la proprietà Count della collection Controls, se ne possono modificare le proprietà, referenziandole come Controls(I).nomeproprietà.

Per categorizzarli, in modo da differenziare il loro comportamento in base alle nostre necessità, si può utilizzare la proprietà Tag (che accetta solo valori testuali) associando ciascun controllo ad una sua categoria di appartenenza. L'esercizio di figura 6 esplora questa collezione.

Così come i Controlli costituiscono una collezione all'interno di un Form, anche i Form costituiscono una collezione all'interno di un'applicazione (es. Forms.Count). Anche i Form si possono creare "al volo" partendo da un Form prototipo.

L'esercizio (figura 7), prevede che realizziate due Form, quello prototipo iniziale e quello MDI, che ospita tutti i form figli creati dinamicamente, e che creiate un modulo (BAS) che contenga le dichiarazioni delle variabili usate "a cavallo" tra i due form:

Global FF(6)
Global K

Il form figlio è del tutto passivo, nel senso che tutto il programma risiede ed è eseguito dal form MDI. Questo, al verificarsi del suo evento Load, genera i sei form figli (**FF(n)**), ne imposta la Caption, ne imposta il Colore di sfondo, ne riempie la TextBox con un numero progressivo e poi con un Timer rende attivo un Form dopo l'altro.

L'esercizio di figura 8 mostra come sia gestibile anche un array di oggetti "passivi", quelli che non subiscono eventi come, ad esempio, le shapes.

Anche DAO è tutto una Collection

La collection può essere considerata un'evoluzione del concetto di matrice sia per il fatto che si estende dalle variabili a tutti gli altri elementi dell'applicazione, sia per il fatto che (stiamo parlan-

do della collection) mette a disposizione dell'utilizzatore moltissimi strumenti per la sua manipolazione, molti di più di quanti ne propongono le matrici.

La bontà della filosofia della Collection è dimostrata anche da come questa si sia benissimo sposata con la tecnologia DAO, quella che propone una "visione ad oggetti" anche per i Database. Un Database è un oggetto, che contiene una collezione di Tabelle, oppure una collezione di Query. Un RecordSet, concetto che si riferisce sia a tabelle che a query, è a sua volta un oggetto che contiene una collezione di Fields. Qualsiasi oggetto, qualsiasi collezione nel suo insieme, qualsiasi membro della collezione sono caratterizzati da una serie di proprietà che ne permettono la loro manipolazione.

Vediamo i tre esercizi che abbiamo preparato.

Il primo (figura 9) coniuga un'array di Label con una collezione di nomi di Fields, ed un'array di TextBox con una collezione di Fields, di contenuti di campi. Gli oggetti e le proprietà più interessanti in gioco sono:

Set DD = OpenDatabase("d:\vvvv\dati")
' il database
Set RR = DD.OpenRecordset("persone")
' il recordset
NE = RR.Fields.Count - 1
' il numero di campi
T1 = RR.Fields(I).Name
' il nome del campo
T2 = RR.Fields(I)
'il contenuto del campo

Spingendo un pò sull'acceleratore si possono realizzare applicazioni più articolate.

Ad esempio in figura 10 vediamo un'applicazione con due ListBox in serie, nelle quali scegliere il nome di una tabella ed i nomi dei campi della tabella. Con l'insieme dei campi selezionati si crea una query che mostra i relativi dati in una Grid.

Da notare come, con le collection, si possa raggiungere una assoluta generalizzazione dell'applicazione che "funziona" con qualsiasi tabella, qualsiasi recordset, qualsiasi campo.

La relativa lunghezza del listato dipende anche dal fatto che, grazie alla collection di Fields.Name, abbiamo inserito anche i nomi di campi in cima alle varie colonne della griglia.

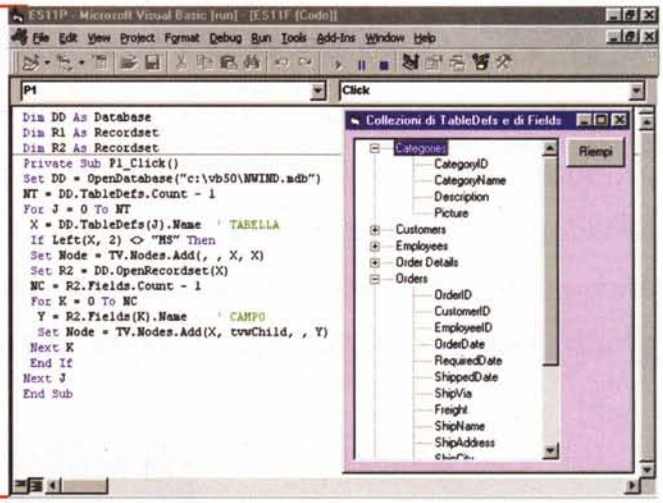


figura 11 - Visual Basic 5.0 - Collezioni di TableDef e di Field in una TreeView
Nell'esercizio precedente abbiamo usato due ListBox per mostrare Tabelle e Campi. In questo usiamo un'unica TreeView, che mostra tabelle e campi secondo una modalità gerarchica ben comprensibile a tutti. L'alimentazione della TreeView prevede lo scorrimento di tutte le tabelle dell'oggetto Database e di tutti i campi dell'oggetto RecordSet.

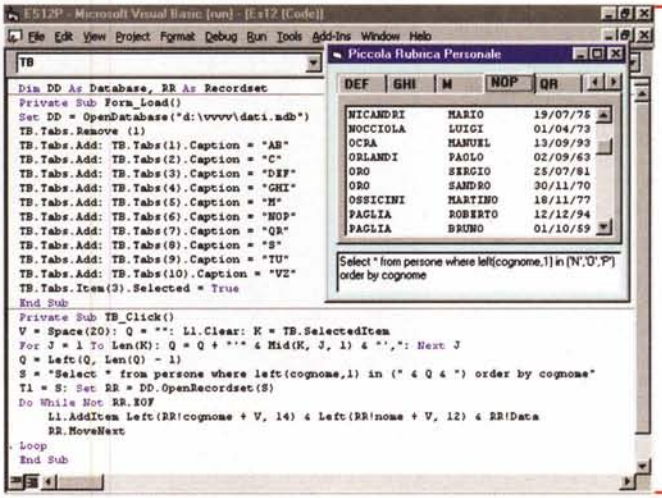


figura 12 - Visual Basic 5.0 - Altri controlli altre collezioni
Sono tanti i controlli di Visual Basic che contengono collezioni di elementi. Ad esempio la TabStrip presenta una collezione di Tabs (linguette) che si possono gestire sia in fase di riempimento che in fase di utilizzo. L'applicazione mostra una piccola rubrichetta personale. Al clic sulla linguetta con le iniziali la ListBox (che è la stessa per tutte le pages) si riempie di dati.

Invece di due liste, una per le tabelle e una per i campi, possiamo usare un'unica lista "gerarchica": una TreeView, che ben si adatta a mostrare gli stessi oggetti: chiaramente i nomi delle tabelle sono al primo livello e i nomi dei campi al secondo livello (figura 11).

Altri oggetti, altre Collection

La Collection, che in definitiva costituiscono un modo standardizzato di trattare insiemi di oggetti, si adattano ad altri oggetti. Ad esempio ai controlli a linguette, in cui ogni linguetta (page è il termine esatto) è un membro della collezione, che vediamo nell'esempio mostrato in figura 12.

Il metodo per aggiungere linguette è Add, la proprietà che indica quale linguetta sia stata selezionata è SelectedItem. Se invece si vuole selezionare "da programma" una delle linguette, basta porre uguale a True la proprietà Selected dell'Item desiderato. Tutto nel rispetto della filosofia "collection".

Nel prossimo numero parleremo di applicazioni SDI, applicazioni MDI e di applicazioni Explorer Style.

MC