Facciamo... lavorare gli altri

Una delle caratteristiche più interessanti di Windows è costituita dalla possibilità di far «collaborare tra di loro» due applicativi. Possono essere percorse numerose strade, in alcuni casi alternative le une rispetto alle altre, in altri casi abbastanza sovrapponibili al punto che un utente potrebbe chiedersi quale dei possibili percorsi sia il più conveniente. Se ci si limita alla possibilità di copiare elementi realizzati con un prodotto in un lavoro fatto con un altro prodotto esistono le tre alternative classiche, ben note a chi conosce anche in modo non approfondito un qualsiasi prodotto Windows: Clipboard, DDE e OLE. Le ricordiamo brevemente, a beneficio dei meno esperti

di Francesco Petroni

Tramite la Clipboard (in italiano Appunti) è possibile copiare qualsiasi oggetto, un brano di testo, un'immagine, un sonoro, da un'applicazione ad un'altra. E poi ognuno per la sua strada.

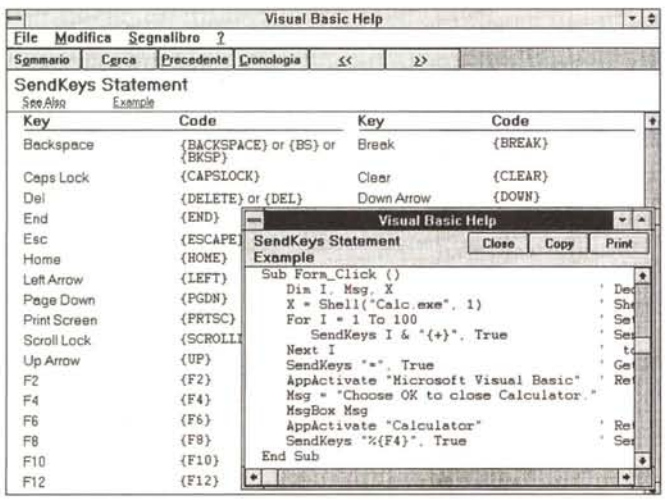
La tecnica DDE è invece più sofisticata, nel senso che si copia un oggetto, che però rimane dinamicamente conforme all'originale. Nel DDE, che è un vero e proprio protocollo di comunicazione, esistono due soggetti: il DDE Server, che ha i dati, e il DDE Client, che li riceve. Tra Server e Client si instaura un collegamento dinamico (Dynamic Data Exchange) monodirezionale (ma, come vedremo, si può rendere bidirezionale) nel senso che nell'applicazione Client non si può modificare l'oggetto, mentre se lo si modifica nell'applicazione Server viene aggiornato (immediatamente o con un processo di ricalcolo) anche nel Client.

La tecnica OLE è una variante della DDE.

Nel collegamento DDE i due prodotti, il Server e il Client, hanno pari dignità, nel senso che ciascuno dei due può essere usato normalmente e può salvare il proprio file. Solo il file realizzato con l'applicazione Client tiene traccia del fatto che contiene un oggetto proveniente dall'esterno. Quindi se si carica il file Client occorre decidere cosa fare del collegamento, in pratica se attivarlo o meno, e nel caso lo si attivi, se desiderare il ricalcolo dinamico o meno.

Nella tecnica OLE, invece, il prodotto Server perde ogni dignità nel senso che viene attivato dal Client come se fosse una sua particolare funzionalità. L'oggetto realizzato con il Server viene salvato nel file realizzato con l'applicazione Client. OLE, Object Linking and Embedding. «Embedded» in italiano traducibile con «incapsulato».

Se si usa un linguaggio di programmazione, ad esempio il Visual Basic, oppure il linguaggio Macro di un applicativo per utente, si possono automatizzare questi processi, con istruzioni che dapprima attivano il collegamento (in una



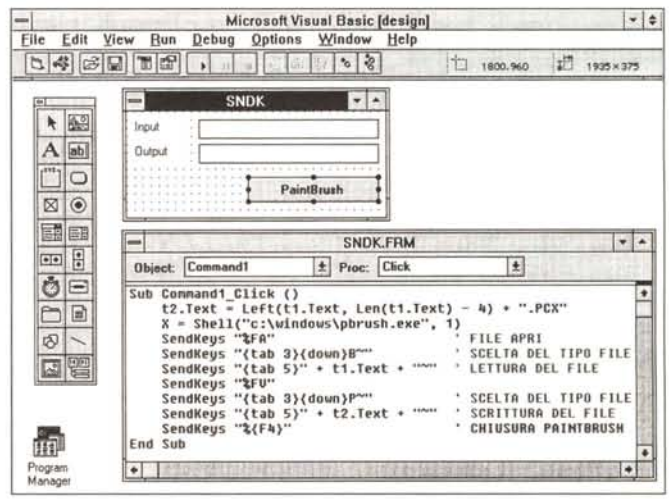
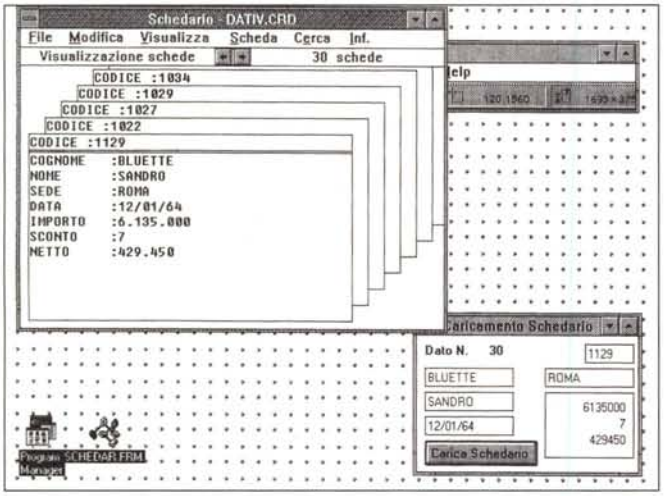


Figura 1 - Visual Basic 3.0 - Codici utilizzabili nel SendKeys.
All'istruzione SendKeys, che utilizziamo per eseguire in un'applicazione richiamata delle azioni, si possono passare, come parametri, solo i comandi eseguibili da tastiera nell'applicazione chiamata (che sono in genere tutti), tenendo presente che ci sono una serie di convenzioni per poter utilizzare anche i tasti più particolari. Poiché alcuni comandi dispongono di scorciatoie esiste anche una convenzione per definire i tasti Shift, Ctrl e Alt.

Figura 2 - Visual Basic 3.0 - Il Comando SendKeys.
L'istruzione SendKeys permette di inviare dei comandi di tastiera da un'applicazione ad un'altra. Nel nostro caso dal Visual Basic inviamo al PaintBrush una serie di comandi che servono per caricare un file grafico in formato BMP e per salvarlo, con lo stesso nome, in formato PCX. Usato in questo modo è come se il PaintBrush diventasse un'applicazione programmabile. Un programma del genere diventa utile nel caso in cui si dovessero convertire un certo numero di file.



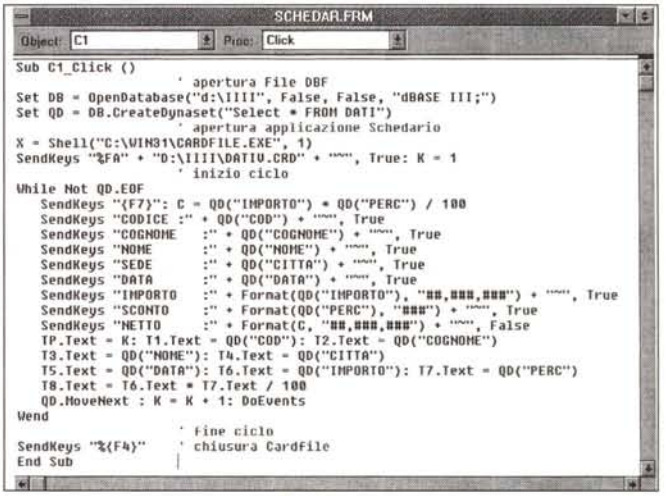


Figura 3 - Visual Basic 3.0 - Caricamento di un file dati DBF nello schedario di Windows - Risultato finale.
Lo Schedario è un applicativo in dotazione in Windows che serve per raccogliere dati un minimo organizzati. Uno dei suoi difetti è che non è in grado di importare né di esportare i dati in altri formati. Il nostro obiettivo è quello di realizzare, con il Visual Basic, un programma che si incarica di leggere i dati da un file DBF e di scaricarli, tramite l'istruzione SendKeys, nelle schede dello Schedario (un record una scheda).

Figura 4 - Visual Basic 3.0 - Caricamento di un file dati DBF nello schedario di Windows - Il listato.
*Abbiamo concentrato tutto il programma sotto l'evento click del pulsante Carica Schedario. Inizialmente viene aperto il Database (che è un volgarissimo file DBF), poi viene aperto lo Schedario di Windows, caricando un suo file *.CRD vuoto, cosa che viene fatta con una serie di SendKeys. Poi viene eseguito lo scorrimento della tabella DBF, i cui dati vengono scaricati nello Schedario di Windows, e contemporaneamente fatti apparire nella Form di Visual Basic. Il programma finisce un po' bruscamente... andrebbe completato con alcune istruzioni che vi suggeriamo nel testo.*

comunicazione DDE le due applicazioni devono essere contemporaneamente aperte), con istruzioni che definiscono il dettaglio del collegamento, con istruzioni che permettono anche di inviare dati dal Client al Server.

Nei vari linguaggi di programmazione sono in genere presenti anche comandi che permettono di «navigare» tra le applicazioni (apertura e passaggio) e di eseguire «azioni» all'interno delle applicazioni attivate.

In altre parole è possibile scrivere dei programmi che agiscono all'interno di altre applicazioni, ad esempio per sfruttare servizi che l'applicazione chiamante non è in grado di svolgere.

Insomma le possibilità sono tante, alcune sono interessanti da un punto di vista teorico, ma non sono adatte ad essere utilizzate in applicazioni complesse, altre possono essere invece prese concretamente in considerazione.

In questo articolo svolgeremo qualche esperimento, qualcuno dei quali potrebbe stimolare il vostro interesse.

Il comando SendKeys

Il comando SendKeys è uno dei più... inquietanti in quanto, essendo presente in tutti i linguaggi e permettendo di programmare l'esecuzione di comandi da tastiera in qualsiasi applicazione Windows, consente di scrivere programmi che eseguono qualsiasi comando in qualsiasi applicazione.

Tutti i linguaggi di programmazione sotto Windows, e quindi anche il Visual Basic e l'Excel che utilizzeremo per le prove, dispongono del comando SendKeys e dispongono di alcuni comandi accessori al SendKeys, come quelli che servono per aprire, per attivare, per chiudere applicazioni Windows, alle quali passare i comandi di tastiera.

Come noto le applicazioni Windows normali dispongono di due o tre modalità di esecuzione dei comandi che è bene ricordare:

– comandi via menu, ad esempio per salvare il comando è menu File Salva;

– comandi di menu eseguiti da tastiera: AltM e poi S;

– scorciatoie di comandi di menu. Al File Salva in genere corrisponde Shift F12;

– toolbar, in cui il comando Salva utilizza una Icona che rappresenta un dischetto.

Volendo utilizzare i comandi SendKeys si possono sfruttare solo le operazioni eseguibili via tastiera, tenendo presente che ci sono una serie di convenzioni che consentono di utilizzare anche i tasti più particolari (fig. 1).

La sintassi del comando SendKeys prevede due parametri. Il primo è la sequenza dei caratteri da inviare, comprendenti i caratteri convenzionali sopra citati. Il secondo parametro è quello che serve per indicare se proseguire con il SendKeys oppure no, e quindi se ritornare il controllo delle operazioni al programma chiamante. È un parametro logico, in cui True significa che continua l'azione del SendKeys nel programma chiamato, e False che il controllo ripassa al programma chiamante, che dovrà eseguire altre azioni.

Il secondo parametro risulta molto importante in quei casi in cui si passi alternativamente da un programma ad un altro.

Sfruttiamo Paintbrush per convertire i nostri file grafici (figura 2)

Paintbrush legge file in formato BMP e PCX e salva in formato BMP e PCX. Se si legge un file BMP e lo si salva in formato PCX si realizza una conversione. Se si volessero convertire parecchi file da BMP a PCX, o viceversa, il lavoro sarebbe lungo e noioso.

Con il nostro Visual Basic realizziamo un programma, molto semplice e molto facilmente implementabile, che contiene due Caselle di Testo. Nella prima scriviamo il nome del file in lettura e nella seconda il nome del file in scrittura. Poi si preme il tasto Paintbrush e viene eseguito il piccolo programma che apre il Paintbrush (con l'istruzione Shell) e che esegue all'interno del Paintbrush una serie di comandi tramite SendKeys:

«%FA»	è il File Apri
«{TAB 3}{DOWN}B~	serve per spostarsi sulla finestra File Apri e per scegliere il tipo di File da leggere
«{TAB 5}»	si sposta nella casella dove si scrive il nome del file
T1.TEXT+«~»	scrive il nome del file, che è posto in T1, e esegue Invio
«%FV»	esegue il comando Salva con Nome
...	
«%{F4}»	Alt F4 per chiudere l'applicazione Windows

Invece di convertire un file per volta è possibile, inserendo il cuore del programma in un ciclo che legga un elenco di figure, una completa automazione del processo. Inoltre se si usa un prodotto più «ricco» del Paintbrush, ad esempio il Paint Shop Pro, che lavora ben 22 tipologie, più alcuni sottotipi, sono possibili numerose ulteriori varianti.

Caricamento automatico dello Schedario di Windows (figure 3 e 4)

Uno degli accessori «classici» di Windows è costituito dalla Schedario che serve per immagazzinare dati molto poco strutturati. Ogni scheda ha un titolo, che appare nella vista Elenco, e un contenuto, che può essere costituito da un testo lungo, o da una serie di righe che possono sembrare dei campi, che si può vedere in vista Scheda.

Ci sono funzioni di Ricerca, funzioni di Stampa, eccetera. Se si crea uno schedario di nomi, che è l'utilizzazione più propria di un prodotto del genere, è anche possibile memorizzare dei numeri di telefono e chiamarli automaticamente.

Il file che si produce ha desinenza CRD (CardFile) ed è utilizzabile solo con l'applicazione Schedario.

Ci proponiamo l'obiettivo di realizzare un'applicazioncina Visual Basic che legga un file DBF, lo scarichi, record per record, in un file CRD, e sempre record per record, lo mostri in un Form Visual Basic. La Form VB ha un solo bottone e una serie di caselle di testo che ricevono i dati del record DBF.

A parte le due istruzioni di dichiarazione delle variabili globali del database:

```
Dim DB As Database, QD As Dynaset
```

tutto il programma risiede sotto il pulsante Carica Schedario.

Le prime istruzioni servono per aprire la Tabella DBF cosa che facciamo sprecando (dato che prendiamo tutti i dati da una sola tabella) un'istruzione SQL e un Dynaset.

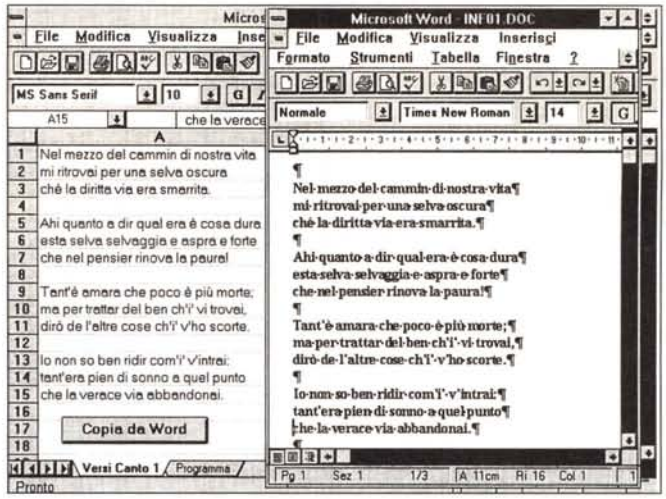


Figura 5 - Excel 5.0 & Word 6.0 - La lettura della Divina Commedia fatta... da Excel. Il comando SendKeys è presente in tutti i prodotti Windows che dispongano di un linguaggio di programmazione, di qualsiasi tipo. Anche una macro di Excel, tramite il comando SendKeys, tradotto in italiano InviaTasti, può far lavorare un... collega. In questo caso vengono lette, una per una, le righe di un testo Word, e poi vengono copiate nelle celle di Excel.

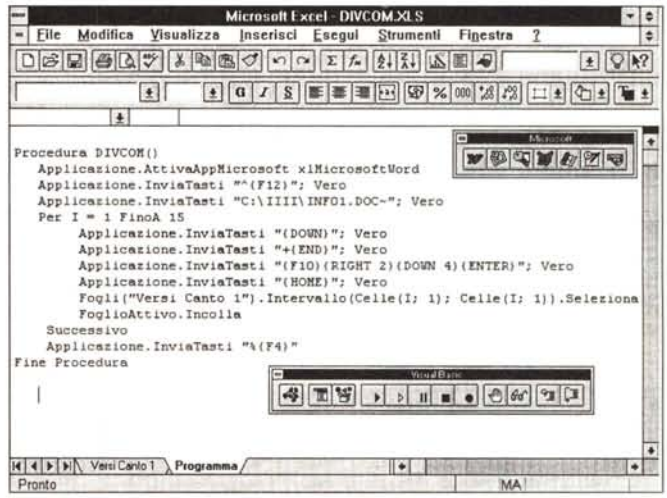


Figura 6 - Excel 5.0 & Word 6.0 - Excel legge i dati di Word. In programmi del genere è molto difficile eseguire il «Debug» per il fatto che quando si passa dall'altra parte si perde, in un certo senso, il controllo della situazione. Il discorso si complica quando, come in questo caso, il programma deve fare avanti ed indietro tra le due applicazioni. Per ridurre i pericoli conviene innanzitutto aprire l'applicazione chiamata, eseguire i comandi da tastiera (trascrivendoli attentamente), poi tornare nel programma chiamante, quello con la macro e riportare con attenzione i comandi.

```
Set DB = OpenDatabase(«D:\IIII», False, False, «dBASE III;»)
Set QD = DB.CreateDynaset(«SELECT * FROM DATI»)
```

L'istruzione AppActivate «Microsoft Visual Basic» non servirebbe dato che stiamo ancora in Visual Basic, la successiva MsgBox serve per avvertire del fatto che si sta partendo.

Le successive istruzioni servono per attivare l'applicazione Schedario e per caricare un file CRD, preparato prima e lasciato vuoto.

```
X = Shell(«C:\WIN31\CARDFILE.EXE», 1)
SendKeys «%FA» + «D:\IIII\DATIV.CRD» + «~», True
```

A questo punto si scorre il file DBF con la classica struttura:

```
While Not QD.EOF
        ....
        QD.MoveNext
Wend
```

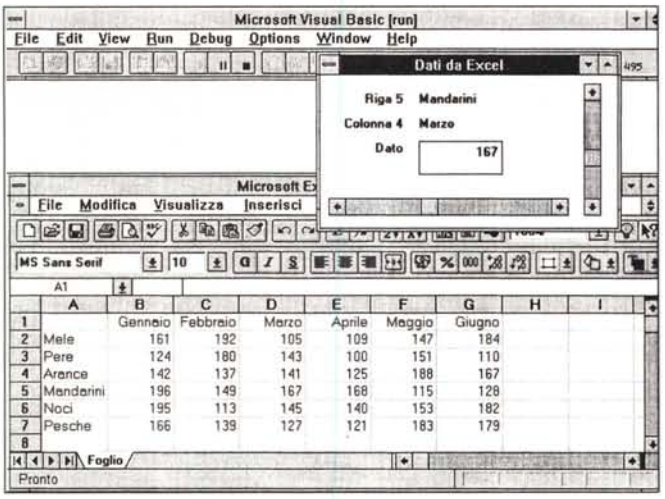
All'interno della quale si individuano i due blocchi, quello che esegue il comando SendKeys per scaricare i vari campi del record corrente nello schedario, e quello che riempie con gli stessi valori la Form. C'è un contatore, si chiama K, che viene mostrato nella Form.

In chiusura l'istruzione «%{F4}» (che corrisponde ad Alt F4) si incarica di chiudere lo schedario. In realtà viene anche chiesto se salvare o meno il file CRD o se annullare l'operazione. A questo pensateci voi.

L'istruzione InviaTasti del VBA di Excel 5.0 (figure 5 e 6)

SendKeys in italiano si può tradurre InviaTasti. Ed è proprio il comando InviaTasti, disponibile nel linguaggio di programmazione Visual Basic for Application di Excel 5.0, che vogliamo sfruttare.

Lo usiamo in un'applicazione, lanciata da un Pulsante, che apre Word, carica un documento che contiene un Canto della Divina Commedia e, riga per riga, lo riporta sul foglio Excel: ogni riga una cella.



	A	B	C	D	E	F	G
1		Gennaio	Febbraio	Marzo	Aprile	Maggio	Giugno
2	Mele	161	192	105	109	147	184
3	Pere	124	180	143	100	151	110
4	Arance	142	137	141	125	188	167
5	Mandarini	196	149	167	168	115	128
6	Noci	195	113	145	140	153	182
7	Pesche	166	139	127	121	183	179

Figura 7 - Visual Basic 3.0 - VB, attraverso un canale DDE, legge i dati di Excel. Usiamo Visual Basic come «client» in un collegamento DDE. L'obiettivo è quello di collegare una casella di testo, in una form Visual Basic, con una zona di celle di Excel e poi, tramite una coppia di ScrollBar, scorrere le righe e le colonne della zona alla ricerca della cella con il dato desiderato.

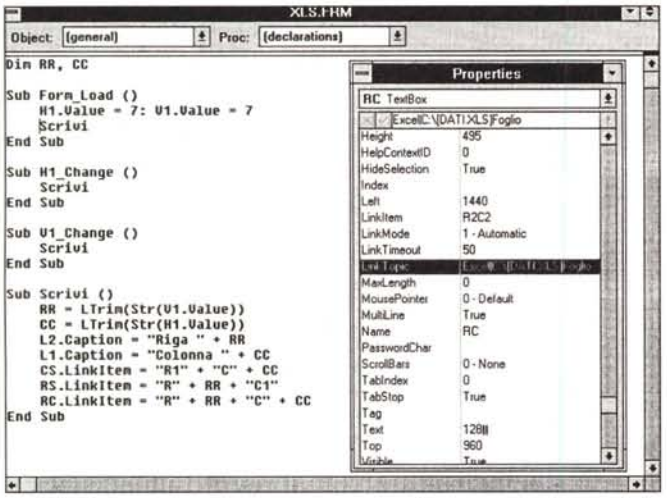


Figura 8 - Visual Basic 3.0 - VB, attraverso un canale DDE, legge i dati di Excel. I passi operativi per impostare un collegamento tra una cella di un foglio Excel e una casella di testo in una Form Visual Basic sono semplici. Si crea nella Form la casella ricevente, poi si passa ad Excel, si punta la cella desiderata e si esegue il comando Copia. L' Incolla si fa poi nella casella di testo dell'applicazione Visual Basic. In tale maniera vengono definite automaticamente le caratteristiche iniziali del collegamento, che verranno successivamente gestite dal programma. L'evento Change delle due Scroll Bar provocherà la modifica della caratteristica LinkItem della casella di testo.

La sintassi del VBA di Excel risente del fatto che si tratta di un linguaggio di programmazione Object Oriented.

L'applicazione richiamata, cioè Word, è un Oggetto cui si applica il Metodo InviaTasti. Altro Oggetto è l'Intervallo del Foglio, cui si applica il Metodo Seleziona. Altro Oggetto è il FoglioAttivo.

Ricordiamo ai nostri lettori che Excel 5.0 rappresenta un punto di passaggio tra il vecchio linguaggio di programmazione Macro usato fino ad Excel 4.0 e il nuovo che ha visto la luce proprio con Excel 5.0.

I termini usati in VBA, tra oggetti, proprietà, metodi, sono quasi 1.800, le funzioni del linguaggio Macro Excel 4.0 sono (erano?) 425. VBA è quindi più potente, oltre ad essere proiettato verso il futuro, in quanto VBA sarà il linguaggio comune a tutte le applicazioni Windows della Microsoft. La sua filosofia è assolutamente differente, ci vorrà del tempo prima di impadronirsene completamente.

VB legge i dati di Excel attraverso un canale DDE (figure 7 e 8)

Vi ricordate il DDE, il Dynamic Data Exchange di Windows? È un po' passato di moda da quando si è imposto l'Object Linking and Embedding. Con il DDE, che è anche programmabile, si gestisce uno scambio di dati tra due applicazioni Windows. Dunque cosa aspettiamo a fare qualche programmino di scambio dati tra due prodotti?

Realizziamo una piccola tabellina con Excel. Poi costruiamo una Form Visual Basic con due Scroll Bar e una Casella di Testo, più qualche Label. Vogliamo agire sulle Scroll Bar e vogliamo che conseguentemente vari la cella puntata e quindi il contenuto della casella di testo che riporta in VB il contenuto della cella del foglio.

Quattro proprietà della casella di testo sono dedicate all'impostazione delle caratteristiche del collegamento DDE. LinkTopic è l'oggetto collegato (il foglio di Excel), LinkItem è il singolo elemento (la cella del foglio), LinkMode serve ad impostare se l'aggiornamento deve essere automatico oppure manuale, LinkTimeOut indica il tempo oltre il quale il tentativo di collegamento va dichiarato fallito.

Questi quattro parametri possono essere digitati o, più semplicemente, prodotti automaticamente con un Copia ed Incolla dalla cella di Excel alla casella di testo di Visual Basic.

I valori Min e Max delle due ScrollBar dipendono dalla dimensione della tabella cui si sta accedendo. Il movimento delle due ScrollBar richiama la routine Scrivi, che, con una serie di manipolazioni di stringa, ridefinisce la proprietà LinkItem, e quindi la cella del foglio puntata, della casella di testo e delle varie Label che riportano intestazioni di riga e colonna e progressivi di riga e colonna.

VB si fa fare i calcoli da Excel (figura 9)

Nell'esercizio appena svolto c'è un collegamento DDE tra il foglio Excel e la casella di testo nella Form Visual Basic. Excel è il DDE Server, Visual Basic il DDE Client ed il collegamento è monodirezionale.

Se si crea un collegamento con le funzionalità di copia ed incolla si ottiene sempre un collegamento monodirezionale. Se invece si fruga tra le istruzioni di programmazioni se ne trova una, la LinkPoke, che permette di percorrere il collegamento al contrario.

La sperimentiamo subito creando una miniapplicazione Excel che esegue il calcolo della rata di un mutuo, cosa che Excel fa molto facilmente disponendo di una specifica Funzione (che vedete nell'immagine). Creiamo una analoga applicazione in Visual Basic con quattro caselle di testo, ognuna collegata alla corrispondente cella di Excel.

Per i meno esperti riportiamo la sintassi della funzione, di categoria finanziaria, che serve a calcolare la rata di un mutuo:

=RATA(tasso;periodi;valore attuale), in cui:

tasso è il tasso di interesse del mutuo (es. 11,75%),

periodi è il numero di rate, es. 15 anni, oppure 30 semestri,

valore attuale è il valore ad oggi dell'operazione e quindi il capitale.

Va detto che Excel dà un risultato negativo (in quanto la rata è una passività) e assegna al risultato un orrendo formato valuta. Mettiamo un segno meno davanti alla formula (=–RATA(...)) e assegniamo alla cella un formato «punto migliaia».

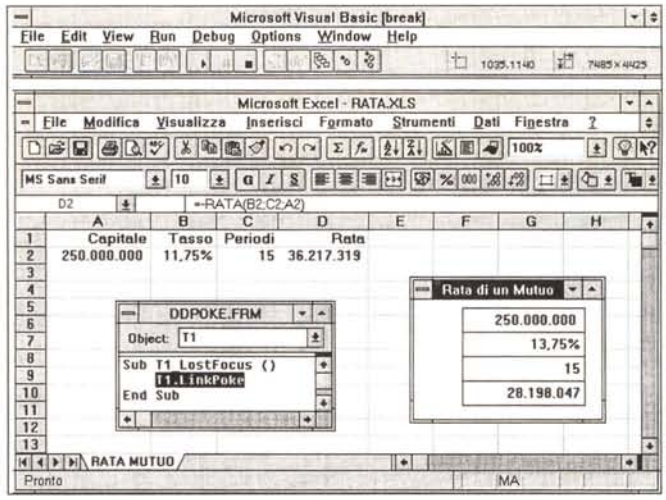


Figura 9 - Visual Basic 3.0 - VB si fa fare i calcoli da Excel. Un canale DDE è bidirezionale. Nel nostro caso ipotizziamo di dover eseguire, in una Form Visual Basic, un calcolo che... non sappiamo fare. Sfruttiamo quindi una Funzione di Excel che abbiamo posto in un foglio di lavoro al quale passiamo dei valori e dal quale riceviamo il risultato.

Va anche precisato che il tasso è riferito al tipo di periodo: tasso semestrale: periodo semestre; tasso annuale: periodo anno. Leghiamo all'evento LostFocus di ciascuna delle tre caselle con i dati il metodo LinkPoke che è quello che rimanda il valore al Server, l'Excel, il quale riesegue il calcolo, e restituisce il nuovo valore della rata.

È chiaro che in questo caso si sta asservendo l'Excel, nel senso che lo si carica solo per fargli eseguire un microcalcolo. Altra soluzione sarebbe quella di sfruttare Excel come «libreria» di funzioni, cosa che si potrebbe fare sfruttando altri comandi del Visual Basic e conoscendo la sintassi delle librerie, che però non sono documentate, per lo meno nei manuali tradizionali.

Scheda e sottoscheda (figure 10 e 11)

Usciamo un po' dal tema dell'articolo per proporvi un esercizio di argomento database. Usiamo il file DBF contenente l'elenco degli articoli apparsi su MC dal numero 1 al numero 146. Tale file contiene pochi campi: il numero di MC in cui è apparso l'articolo, la pagina, la sigla dell'autore, il titolo e la rubrica. Il campo con la rubrica si chiama Tipologia. Precisiamo inoltre che i campi numerici li abbiamo, direttamente nel file DBF, trasformati in testuali con caratteri «0» di riempimento. Il file DBF lo abbiamo chiamato ARTICOLI.

Guardando la figura 10 risulta ben chiaro lo scopo dell'esercizio: c'è una Form con pochi oggetti:

– una lista con i nomi delle Rubriche, dalla quale si sceglie la rubrica desiderata,

– una casella di testo in cui appare, per un opportuno controllo, l'istruzione SQL che estrae gli articoli di tale rubrica,

– una griglia di cinque colonne (compresa la prima in cui inseriremo un progressivo) che riceverà l'elenco degli articoli. Il numero di righe sarà variabile in funzione dei record in arrivo,

– due pulsanti, perché vogliamo testare due possibili modalità di ricerca e selezione dei record, che eseguono la selezione e riempiono la griglia.

Passiamo ad osservare i listati. Quelli della figura 11 riguardano le fasi iniziali dell'operazione e quindi definizione delle variabili, attivazione della Form con impostazione delle caratteristiche iniziali della griglia (valide a griglia vuota) e con la routine di riempimento della lista (che in realtà è una combo box).

La combo è riempita automaticamente grazie ad una selezione, senza ripetizione (DISTINCT) del solo campo TIPOLOGIA, dalla tabella ARTICOLI. In altre parole la lista di Tipologie viene caricata leggendo le tipologie direttamente nella tabella ARTICOLI in modo che se ad un certo punto nasce un'altra rubrica di MC la lista la comprende immediatamente.

Quando si sceglie una Rubrica dalla lista (evento Sub C1_Click()) viene generata la conseguente istruzione SQL che appare nella text box, ma, si badi bene, non viene eseguita.

Per eseguire la selezione e conseguentemente riempire la griglia con l'elenco degli articoli della tipologia desiderata, si possono percorrere due strade ben differenti tra di loro. La prima è

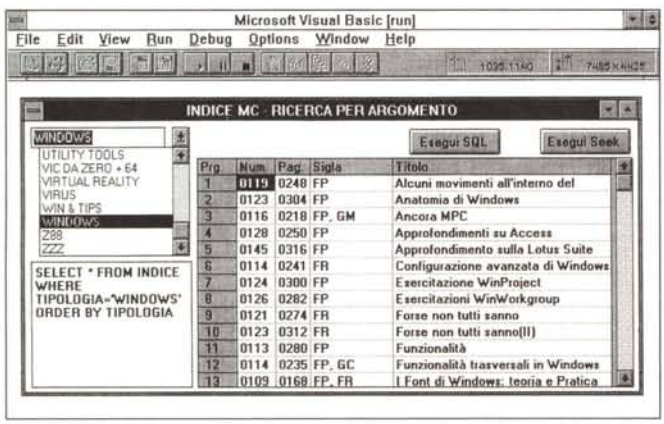


Figura 10 - Visual Basic 3.0 - Una griglia utilizzata come SubForm. Usiamo ancora una volta il «famoso» file DBF contenente i dati relativi a tutti gli articoli di tutti i primi 146 numeri di MC, per un totale di oltre 4.000 articoli. Nella struttura del file dBase è presente il campo Tipologia, che indica la Rubrica cui l'articolo appartiene. Il nostro obiettivo è quello di realizzare una Form in cui si possa scegliere la Rubrica. Sulla base di questa scelta vogliamo che venga dimensionata ed alimentata una griglia che contenga i dati degli articoli della rubrica selezionata.

Figura 11 - Visual Basic 3.0 - Griglia Subform - Fasi iniziali. Queste routine riguardano le fasi iniziali del programma, contengono le dichiarazioni delle variabili e le istruzioni eseguite all'attivazione della Form, che servono a due cose. Alla predisposizione delle caratteristiche «estetiche», titoli, larghezza, ecc. della Griglia ed al caricamento della Combo Box con l'elenco delle Tipologie degli articoli. Se ne occupa un'elegante istruzione SQL che estrae l'elenco evitando i duplicati.

```
ARTICOLI.FRM
Object: (general)   Proc: (declarations)

Dim DB As Database, TB As Table, SN As Snapshot, T$


Sub C1_Click ()
    T$ = "SELECT * FROM ARTICOLI WHERE TIPOLOGIA='"
    T$ = T$ + C1.Text + "' ORDER BY TIPOLOGIA": T1.Text = T$
End Sub


Sub Form_Activate ()
    Set DB = OpenDatabase("D:\IIII", False, False, "dBASE III;")
    G1.Row = 0: G1.Col = 0: G1.Text = "Prg.": G1.ColWidth(2) = 400
    G1.Col = 1: G1.Text = "Num.": G1.ColWidth(2) = 500
    G1.Col = 2: G1.Text = "Pag.": G1.ColWidth(2) = 500
    G1.Col = 3: G1.Text = "Sigla": G1.ColWidth(3) = 1200
    G1.Col = 4: G1.Text = "Titolo": G1.ColWidth(4) = 3200
    T$ = "SELECT DISTINCT TIPOLOGIA FROM ARTICOLI"
    Set SN = DB.CreateSnapshot(T$)
    SN.MoveFirst
    While Not SN.EOF
        On Error Resume Next
        C1.AddItem SN("TIPOLOGIA"): SN.MoveNext
    Wend: C1.ListIndex = 0
End Sub
```

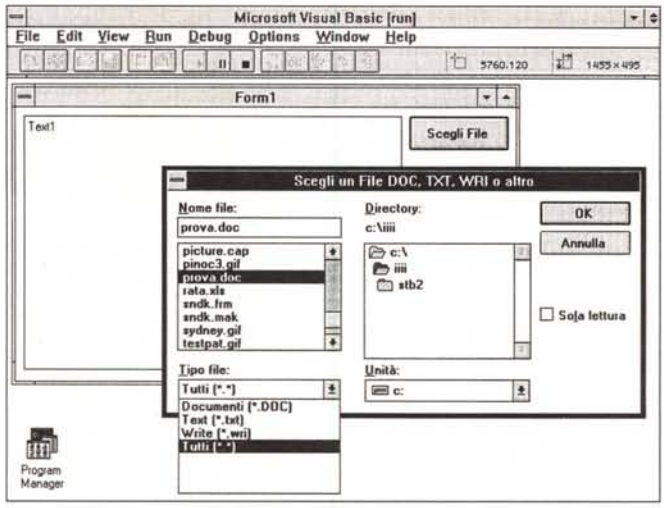


Figura 12 - Visual Basic 3.0 - Una casella di testo ben piena. Ancora SendKeys usato per pescare un testo lungo, un intero documento Word, e per poi piazzarlo in una casella di testo di Visual Basic. Sembra che Windows '95 disporrà di numerosi altri «controls», uno dei quali sarà una casella di testo in grado di accettare un testo formattato. Ne riparleremo tra qualche mese. La Form Visual Basic contiene la casella di testo e un pulsante. La prima cosa che fa il programma è l'attivazione della Common Dialog Box «File Apri», tramite la quale l'utente può scegliere il file desiderato.

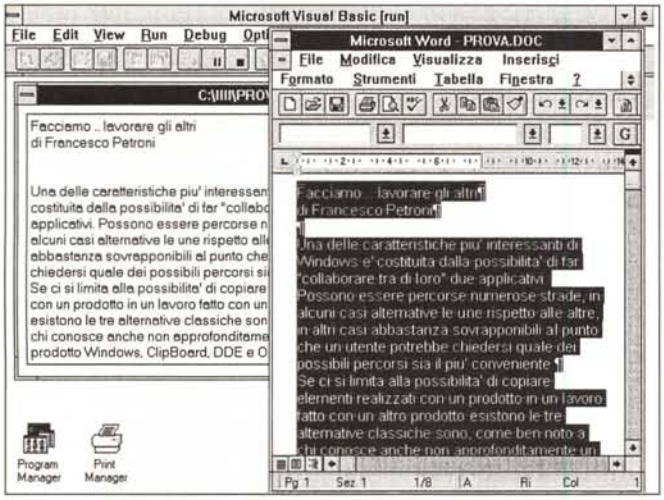


Figura 13 - Visual Basic 3.0 - La copia da Word. In questo caso la parte da padrone la fa la funzionalità di Copia ed Incolla. Con una serie di comandi SendKeys ben assestati si passa in Word, si apre un file (il cui nome è scelto dall'utente) e si seleziona tutto il testo. Si esegue il comando Copia dopodiché il programma ripassa in Visual Basic in cui c'è la comoda funzione Clipboard.GetText() che serve per attribuire ad una casella di testo il valore del Clipboard. Ben più elegante del volgare Incolla.

SQL, la seconda invece utilizza strumenti alla dBase III.

La prima esegue il comando SQL già creato in precedenza, poi scorre due volte la tabella «snapshot» risultante, la prima per contare quanti articoli sono stati trovati, la seconda per riportare l'elenco nella griglia, uno per riga e cinque campi per riga. La struttura di scorrimento è sempre la stessa ricordata prima.

La seconda maniera simula un uso della sintassi dBase, per cui si lavora direttamente con la tabella ARTICOLI, che viene attivata con l'istruzione:

```
Set TB=DB.OpenTable(«ARTICOLI»)
```

La routine di ricerca e scorrimento in dBase sarebbe:

```
USE ARTICOLI INDEX ARTICOLI
SEEK K (se K è la tipologia desiderata)
DO WHILE TIPOLOGIA = K
        ....
        SKIP
ENDDO
```

Il problema principale consiste nel fatto che se si utilizza un file indice NDX questo va dichiarato in un file di tipo INF. Nel nostro caso poiché la tabella si chiama ARTICOLI.DBF e l'indice si chiama ARTICOLI.NDX abbiamo costruito un file ARTICOLI.INF in cui l'indice è stato dichiarato:

```
[dBASE]
NDX1=ARTICOLI.NDX
```

Tutti e tre i file, DBF, NDX e INF vanno posti nella stessa directory. Le prestazioni dei due sistemi utilizzati sono state soddisfacenti (uno o due secondi per cercare ed elencare un centinaio di articoli da una tabella di 4.000).

Una casella di testo ben piena (figure 12 e 13)

Ritorniamo sul SendKeys per proporvi un nuovo e divertente esercizio. Lo scopo è quello di inzeppare di testo una casella di testo. Procediamo con ordine.

Il controllo di Visual Basic più usato è la casella di testo. In una casella di testo si possono inserire fino a 32.000 caratteri. In tal caso è opportuno scegliere, come caratteristiche della casella di testo, un Font piccolo e l'attivazione di una ScrollBar verticale, è invece obbligatorio scegliere la caratteristica Multiline Yes.

La casella di testo ha dei limiti, che pare saranno in parte superati con Windows '95, che, come detto in altre occasioni, si porta appresso dei nuovi tipi di Controls.

Il primo limite è che non è possibile inserire testi formattati, nel senso che è possibile definire, come appena detto, le caratteristiche del testo solo a livello di casella (ad esempio il font, il grassetto, il corsivo, ecc.) ma non è possibile definirle all'interno del testo nella casella stessa.

Il secondo consiste nella gestione delle «andate a capo». Se la casella non ha una barra di scorrimento orizzontale il testo va a capo quando finisce lo spazio orizzontale, se invece ci si mette la barra il testo va a capo quando si preme il tasto Invio, in altre parole quando c'è il carattere CR.

Pur con queste limitazioni può essere conveniente in alcuni casi sfruttare la capacità, in termini di «volume» di caratteri contenuti, della casella di testo. Ad esempio si può ipotizzare di costruire un'applicazione che legge un file strutturato che contiene anche un campo testuale con il nome di un file realizzato con un Word Processor... qualsiasi, ma sotto Windows. Potremo quindi scrivere un'applicazione che lancia il WP, carica il file, seleziona tutto il testo, lo copia e poi torna sulla Form Visual Basic nella casella di testo e ci «incolla» dentro il testo selezionato.

Il programma non presenta grosse novità rispetto a quanto visto prima se non per l'uso della CommonDialog Box File Apri che viene attivata per far scegliere all'utente il file che vuole. Lo commentiamo solo nelle didascalie.

Conclusioni

Abbiamo visto fondamentalmente l'istruzione SendKeys, e abbiamo constatato come sia molto comoda per risolvere quei piccoli problemini a cavallo di più applicazioni. Va un po' presa con le «molle» nel senso che un programma che usa SendKeys per lavorare su un'altra applicazione è «debuggabile» con difficoltà, per il fatto, ovvio, che, nell'applicazione richiamata si lavora solo simulando la tastiera.

Inoltre ci sono prodotti un po' «riottosi» ad essere maltrattati da un SendKeys qualsiasi per cui ad esempio accettano certi comandi e non altri. In ogni caso, prima di azzardarsi a costruire un'applicazione importante, è bene fare una piccola sperimentazione con dei programmini di poche righe, come i nostri.

MC