Ancora Visual Basic for Application
Oggetti, Proprietà e Metodi

di Francesco Petroni e Raffaele Valensise

Abbiamo parlato del Visual Basic for Application in precedenti articoli di MC. Ritorniamo, dopo qualche mese, sull'argomento parlando ancora del VBA di Excel, in attesa che, con la prossima uscita di Office per Windows 95, in cui troveremo la nuova versione di Excel e il nuovo VBA, la tecnologia si consolidi e possibilmente si semplifichi

L'idea che sta alla base del Visual Basic for Application è nota. Si vuole, o meglio la Microsoft vuole, realizzare un unico linguaggio di programmazione comune a tutte le applicazioni Windows, con la possibilità, qualsiasi sia l'applicazione da cui si parte, di richiamare ed utilizzare oggetti propri di altre applicazioni (mediante la tecnologia OLE Automation). Questo consente in pratica di costruire applicazioni che «vedono» qualsiasi altra applicazione, che deve essere VBA compatibile, come proprio «servitore». I primi prodotti che hanno adottato VBA sono stati Excel 5.0 e Project 4.0 della Microsoft.

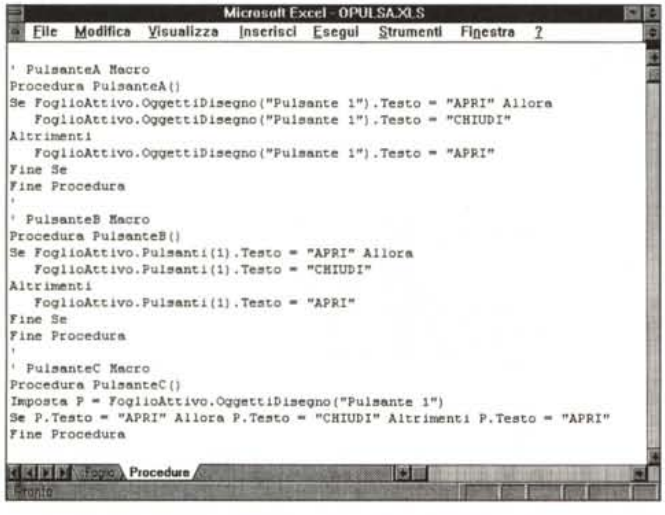
Con Office per Windows 95 entra nel gruppo, e questo è interessantissimo, anche Access, che è il DBMS della casa. Dei prodotti della Suite Office della Microsoft rimane però fuori ancora il Word (che conserva il suo Word Basic, che non è VBA).

Inoltre, fra qualche mese, dovrebbe uscire finalmente anche Visual Basic 4.0 (nella doppia versione 16 e 32 bit). VB4 dovrebbe fare da tessuto connettivo in quei casi in cui l'applicazione da sviluppare non abbia una connotazione precisa e quindi non sarebbe corretto svilupparla con il VBA di Excel, oppure con quello di Access. Di questi argomenti riparleremo a tempo debito.

Visual Basic for Application: vantaggi e svantaggi

Sintetizziamo per punti la filosofia che caratterizza il VBA di Excel e la conseguente programmazione Object Based:

– in un linguaggio visuale, come VBA, si lavora con gli OGGETTI,
– il prodotto, nel nostro caso Excel, mette a disposizione una serie di Oggetti predefiniti o preconfezionati, che l'utente deve ben conoscere,
– in generale esiste un'organizzazione gerarchica degli oggetti, per cui per individuare l'oggetto occorre percorrere tutta la gerarchia cui esso appartiene,
– ogni oggetto è caratterizzato da una serie di PROPRIETÀ,
– le proprietà possono essere impostate «una tantum» oppure essere impostate durante l'esecuzione del programma,



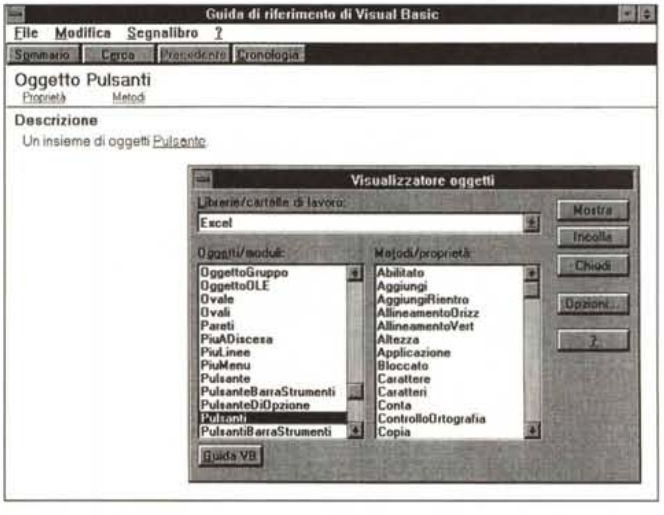
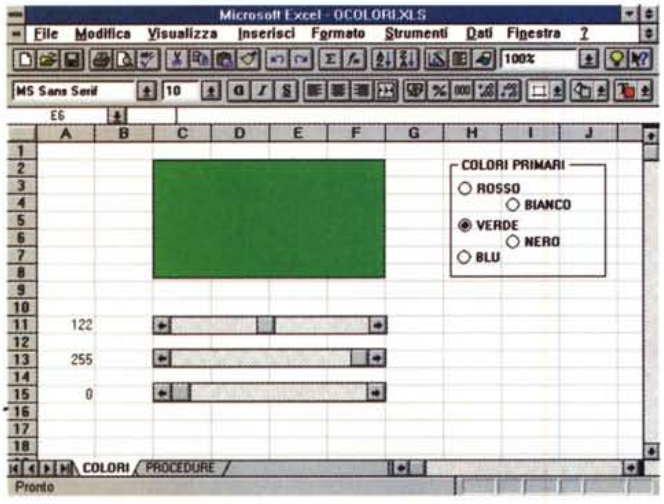


Figura 1 - Excel 5.0 - Visual Basic for Application - Tre modi per gestire un Interruttore.
Queste sono tre «varianti» di uno stesso programma, cambia solo il modo di definire l'oggetto coinvolto. L'oggetto coinvolto è un pulsante la cui scritta è «APRI». Le tre macro, alternative l'una alle altre, collegate all'evento click sul pulsante stesso, provocano la variazione della scritta da APRI a CHIUDI e viceversa. Nel primo caso viene definito il Pulsante 1, nel secondo caso viene individuato il primo elemento della Collezione Pulsanti e nell'ultimo viene definita una Variabile oggetto, che corrisponde all'Oggetto Pulsante 1. In questa maniera si semplificano tutti i successivi comandi per gestire le altre caratteristiche dell'oggetto.

Figura 2 - Excel 5.0 - Visual Basic for Application - Il Visualizzatore degli Oggetti.
Esiste l'Oggetto Pulsante ed esiste l'Oggetto Pulsanti, che è, come spiega, in un modo eccessivamente sintetico, l'Help del VBA, un insieme, una collezione, di oggetti Pulsante. Come usare l'uno o l'altro oggetto lo abbiamo verificato nell'esercizio precedente.



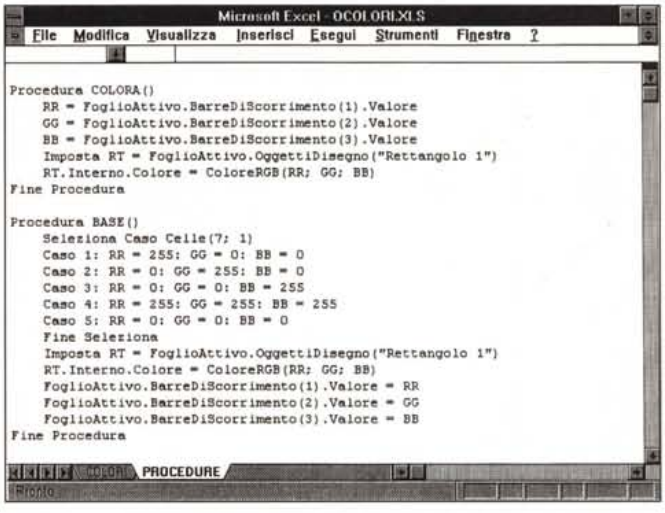


Figura 3 - Excel 5.0 - Visual Basic for Application - Il metodo RGB - In azione.
Abbiamo un rettangolo del quale vogliamo definire il colore sfruttando la funzione ColoreRGB, alla quale vanno passati tre parametri corrispondenti alle tre componenti dei tre colori fondamentali: il Rosso, il Verde e il Blu. Vogliamo poter impostare cinque colori fondamentali, i tre di prima più il Bianco e il Nero, con i cinque pulsanti di opzione. Poi vogliamo impostare tutti i colori intermedi agendo su tre Scroll Bar. Il programma, molto spettacolare ma semplice da realizzare, può essere scritto con qualsiasi linguaggio di programmazione in Windows.

Figura 4 - Excel 5.0 - Visual Basic for Application - Il metodo RGB - Listati.
Su un foglio normale abbiamo inserito le tre Scroll Bar, i cui valori massimo e minimo sono 255 e 0, e i cinque Pulsanti di Opzione. Le tre Scroll Bar sono legate alla procedura COLORA, che legge i tre valori e li usa per colorare il rettangolo. I cinque pulsanti di opzione lanciano la procedura BASE, nella quale l'istruzione Seleziona Caso, indirizza i cinque sottocasi. La procedura Base provvede anche a reimpostare i valori delle Scroll Bar.

– le proprietà possono anche essere lette e/o variate durante l'esecuzione del programma,
– le operazioni che, compiute sugli oggetti, ne modificano le proprietà si chiamano METODI,
– le varie operazioni compiute sugli oggetti vengono scatenate al verificarsi di EVENTI (programmazione Event Driven).

In altre parole la programmazione in un prodotto Object Based ed Event Driven consiste nel programmare gli Eventi che provocano, in generale, la modifica di alcune Proprietà degli Oggetti in gioco.

Il Linguaggio Visuale VBA di Excel 5.0, per quanto innovativo e stimolante, risulta essere un po' farraginoso, in quanto la filosofia che è alla base della programmazione Object Based si sposa con una certa difficoltà con un prodotto in cui ci sono oggetti così particolari e così differenti gli uni dagli altri. Inoltre la versione 5.0 di Excel è caratterizzata da una gerarchia molto spinta, in cui ci sono comunque almeno quattro livelli: Applicazione, Cartella, Foglio ed Intervallo. Questo rende l'individuazione del singolo elemento più lunga e quindi più complicata.

In questo articolo affronteremo di nuovo VBA, cercando però di essere più sistematici nella trattazione degli argomenti. Abbiamo già detto dei concetti fondamentali della programmazione Object Based, proporremo ora una serie di semplici esercizi esplicativi di questi concetti. Lo scopo è quello di migliorare le nostre conoscenze del nuovo linguaggio, anche allo scopo di cercare di alleggerire il più possibile il Codice che dovremo scrivere.

Alla fine proporremo un esempio un po' più complesso e completo, in cui il VBA, e più in generale le potenzialità in termini di elaborazione dati di Excel, vengono ben sfruttate.

Nella trattazione diamo per scontati alcuni concetti già sviluppati nei due articoli introduttivi, apparsi nei numeri 141 e 142 di MC.

Gli Oggetti, le Proprietà e le Procedure

A chi avesse perso le due puntate precedenti, diremo che il VBA considera quali oggetti manipolabili: la stessa Applicazione «Excel», la Cartella (come insieme di fogli), il singolo Foglio di lavoro, la colonna, la riga, la cella, la barra degli strumenti, un qualsiasi oggetto, grafico e non grafico, posto sul foglio, nonché gli «intervalli» che l'utente può definire a piacimento fra le celle.

Ad esempio, volendo modificare da VBA l'estetica del carattere della cella F3 del foglio «Archivio», dobbiamo specificare «chi appartiene a chi» e quale valore deve assumere la proprietà che ci interessa dell'oggetto specifico:

```
Procedura Estetica()
  Fogli(«Archivio»).Celle(3;6).Ca-
rattere.Nome = «Times New Roman»
  Fogli(«Archivio»).Celle(3;6).Ca-
rattere.Grassetto = vero
Fine Procedura
```

Ricordiamo anche che il VBA di Excel 5.0 consente anche di utilizzare alcuni oggetti tipici dell'interfaccia Windows trattandoli sia come semplici pulsanti «cliccabili», posti direttamente su un foglio, e collegati alle celle dello stesso o di un altro foglio di lavoro, sia inserendoli come parte integrante di una finestra di dialogo di un foglio di tipo Dialogo.

Un pulsante, una casella di testo, una barra di scorrimento, un menu a discesa, ecc., oltre a possedere proprietà intrinseche sono anche in grado di pilotare procedure (Excel intercetta gli Eventi legati a tali oggetti). Il gioco diventa complesso quando si tratta di sfruttare un oggetto sia come erogatore di dati che come acquisitore che come scatenatore di Eventi. E si tratta di casi che si verificano non raramente.

Un po' di sperimentazione

Come primo esperimento vi proponiamo di piazzare su un foglio normale un pulsante la cui scritta sia APRI. Vogliamo che al click sul pulsante la scritta cambi in CHIUDI e viceversa.

Come detto la proprietà di un ogget-

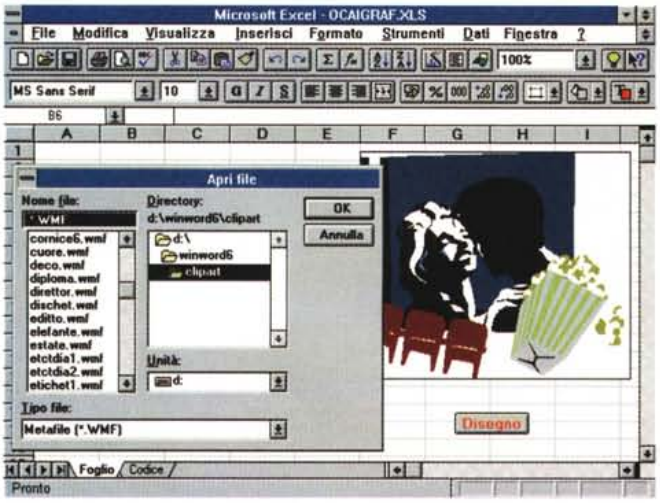


Figura 5 - Excel 5.0 - Visual Basic for Application - Finestra di Dialogo File Apri - In azione.
Con VBA è possibile accedere in vari modi alle risorse di Windows. Qui vediamo in azione il Metodo AttivaApriNomefile che si applica all'Oggetto Applicazione. In pratica manda in esecuzione la finestra di Dialogo «FileApri» di Windows e restituisce il nome del file scelto. Lo usiamo per impostare il nome di un disegno che poi leggiamo e piazziamo sul foglio di Excel.

Figura 6 - Excel 5.0 - Visual Basic for Application - Finestra di Dialogo File Apri - Listato.
Il Metodo AttivaApriNomefile accetta una serie di parametri (noi ne abbiamo usato uno solo) che potete controllare nell'Help. Se nella finestra File Apri si preme Annulla viene restituito il valore Falso con il quale si fa, da programma, interrompere la procedura sottostante. Se viene invece scelto un file corretto la procedura lo visualizza sul foglio come Oggetto Immagine. La procedura provvede anche a cancellare l'Oggetto Immagine precedentemente posizionato.

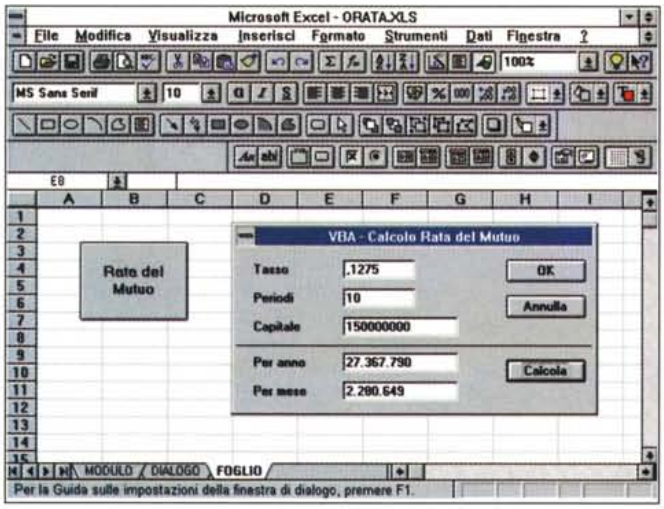


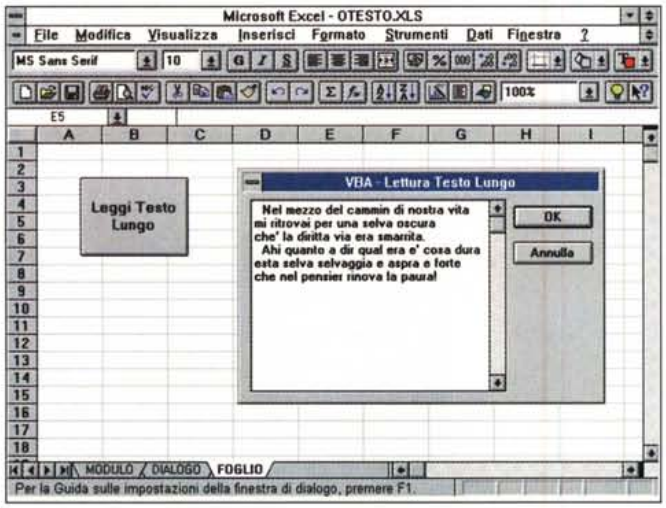
Figura 7 - Excel 5.0 - Visual Basic for Application - Una Dialog Box del tutto autonoma - In azione.
Le Funzioni disponibili nel linguaggio VBA non sono le stesse presenti sul Foglio. Fortunatamente è possibile, passando attraverso l'Oggetto Applicazione, usare in VBA le funzioni del Foglio Excel. Questo consente di realizzare applicazioni VBA che sfruttano le potenzialità enormi dei comandi e delle funzioni del foglio. In questo caso abbiamo realizzato una Finestra di Dialogo in cui si inseriscono dei dati (il problema trattato è il calcolo della rata di un Mutuo) e nella quale, al click sul Pulsante Calcola, viene fatto apparire il risultato del calcolo.

Figura 8 - Excel 5.0 - Visual Basic for Application - Una Dialog Box del tutto autonoma - Listato.
In questo esercizio le procedure sono due. Quella che provoca l'attivazione della Finestra di Dialogo e quella, collegata al pulsante Calcola, da pigiare dopo aver inserito i dati necessari al calcolo, che utilizza la Funzione del Foglio RATA(a,b,c) passandole correttamente i parametri e ricevendone indietro il risultato desiderato. La possibilità di sfruttare il «tesoro» in termini di «functions» del Foglio di Excel e la possibilità di eseguire dei calcoli senza abbandonare la Finestra di Dialogo rendono possibili una grande quantità di applicazioni (ne vedremo un'altra alla fine dell'articolo).

to può essere impostata, ma può anche essere letta. Il nostro oggetto è il pulsante, ne leggiamo la proprietà Testo e, se questa è uguale ad APRI, la cambiamo in CHIUDI e viceversa. La figura 1 mostra tre varianti del programma che esegue questi cambiamenti.

Nella prima variante facciamo riferimento all'oggetto Pulsante, proprio a quel pulsante con quel nome. Nella seconda facciamo riferimento all'oggetto Pulsanti, ed in particolare al primo dei pulsanti presenti nel foglio, di cui non è più necessario conoscere il nome.

Nella terza variante vediamo come sia possibile, dovendo riferirci in molte istruzioni allo stesso oggetto pulsante, creare una variabile che identifica in una



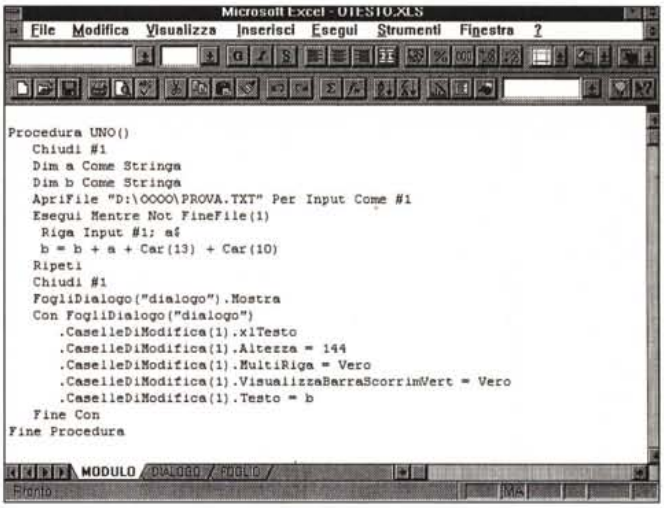
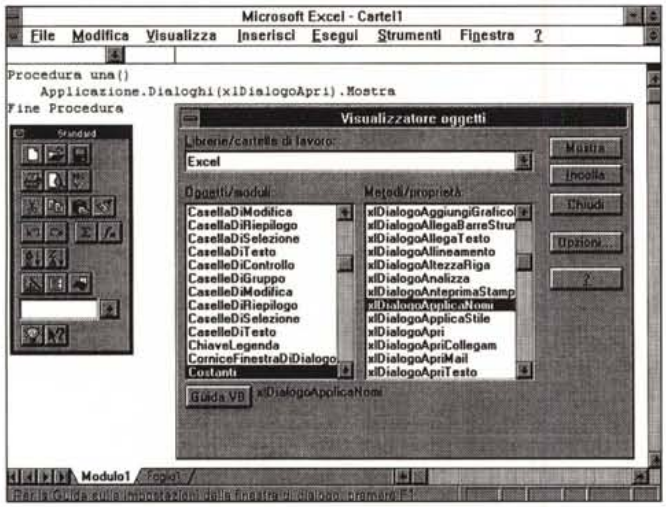


Figura 9 - Excel 5.0 - Visual Basic for Application - Il testo lungo in una Box - In azione.
Secondo noi la più grande difficoltà che si incontra nella realizzazione di una procedura VBA sta nella mancanza di una documentazione esauriente. Non esiste più il manuale Reference mentre l'Help di Windows è in molti casi scarno (in quello della versione italiana mancano addirittura «pagine» presenti invece in quella americana!). Non siamo riusciti, ma non è detto che non lo si possa fare, ad inserire un testo lungo in una Casella di Testo di una finestra di Dialogo. In VB si può, con il limite di 65.000 byte, in VBA il limite sembrerebbe di 256 byte, lo stesso del contenuto di una cella.

Figura 10 - Excel 5.0 - Visual Basic for Application - Il testo lungo in una Box - Listato.
Il programma accede ad un file testuale, organizzato per righe, lo legge riga per riga, somma le righe in un'unica variabile che piazza, alla fine, nella Casella di Testo. Come detto, in questo caso il limite riscontrato sperimentalmente è di 256 caratteri. Verifichiamo anche la possibilità di definire da programma l'altezza della casella di testo e la possibilità di inserirvi le barre di scorrimento.



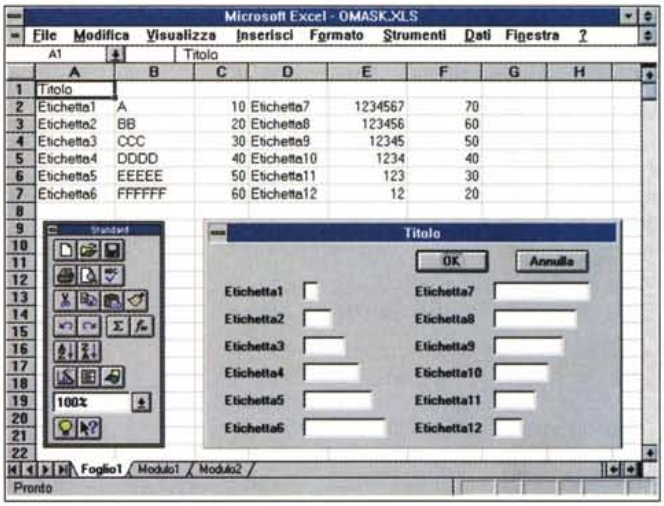


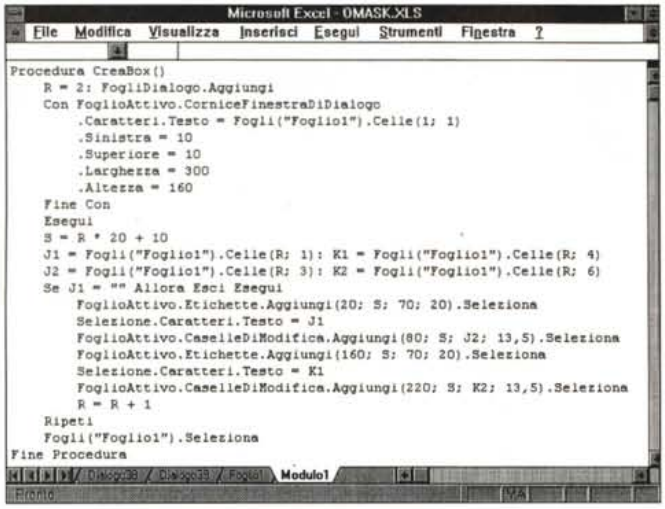
Figura 11 - Excel 5.0 - Visual Basic for Application - Le costanti «xl».
In Microsoft Excel esistono delle costanti incorporate che è possibile manipolare con Metodi e Proprietà. Queste costanti iniziano tutte con le due lettere xl e vengono documentate nell'Help con il metodo o la proprietà a cui sono applicate. È anche possibile utilizzare il Visualizzatore degli Oggetti per sfogliare l'elenco di costanti incorporate. Lo vediamo nella figura, mentre in secondo piano vediamo come usare una costante xl per lanciare la Box File Apri (è un sistema differente da quello visto prima e serve ad altri scopi).

Figura 12 - Excel 5.0 - Visual Basic for Application - Automatizziamo la costruzione di una Dialog Box - Effetto.
Per realizzare una Finestra di Dialogo personalizzata con Excel occorre lavorare su un tipo speciale di Finestra, richiamabile con il comando Inserisci Macro Finestra di Dialogo. In questo ambiente si compone, aggiungendo a mano i vari elementi, la Finestra stessa. Proponiamo una sorta di automazione della produzione, nel senso che vogliamo scrivere su un foglio normale l'elenco degli elementi desiderati e delle caratteristiche desiderate e poi, con una Macro, vogliamo convertire l'elenco in oggetti della finestra di dialogo. Nella figura vediamo sul foglio l'elenco degli oggetti ed in primo piano il risultato ottenuto.

maniera più sintetica il pulsante stesso. In pratica è possibile con questo sistema alleggerire la dimensione del codice.

Nella figura 2 vediamo il visualizzatore degli oggetti di Excel ed in particolare gli oggetti pulsante.

Nell'esercizio successivo gestiamo la proprietà InternoColore di un Oggetto Rettangolo (figg. 3 e 4). Piazziamo tre Barre di Scorrimento sul foglio, ne definiamo i valori minimo e massimo (a 0 e a 255) e le leghiamo a tre celle del foglio. Queste proprietà si possono definire, dal quick menu, senza ricorrere alla programmazione. Allo scorrere di una qualsiasi delle tre barre eseguiamo la Subroutine (COLORA) che assegna al rettangolo il colore RGB, caratterizzato



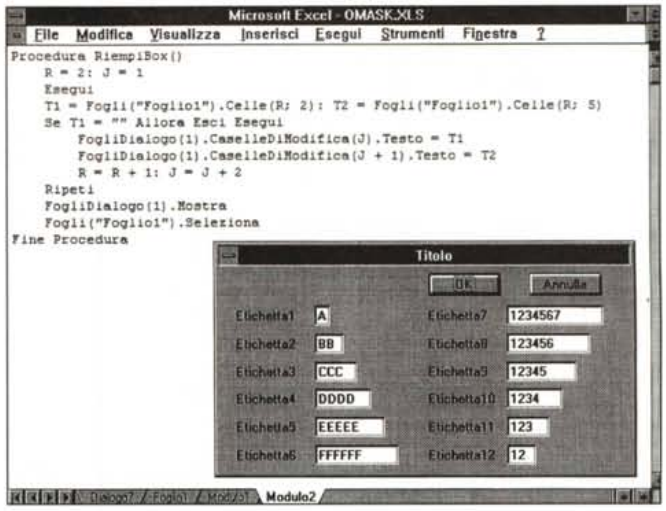


Figura 13 - Excel 5.0 - Visual Basic for Application - Automatizziamo la costruzione di una Dialog Box - Listato.
I vari comandi, o meglio, i vari Metodi che abbiamo utilizzato agiscono sul foglio attivo, anche se si tratta di un foglio di tipo Finestra di Dialogo. Nella parte superiore del listato agiamo sull'oggetto CorniceFinestraDiDialogo del foglio attivo, impostandone le varie caratteristiche dimensionali ed il Titolo, che viene preso dalla cella A1 del Foglio1. Nella parte inferiore il ciclo che aggiunge Etichette e Caselle di Modifica, dimensionate secondo le «misure» indicate sul Foglio1. È evidente che si tratta di un programma che può essere preso come spunto per un'applicazione più complessa (che prevede più tipi di oggetti) e più generalizzata.

Figura 14 - Excel 5.0 - Visual Basic for Application - Ed ora riempiamo la Dialog Box.
Qui vediamo come sia possibile riempire le Caselle di Modifica della nostra Finestra di Dialogo con un contenuto, supposto che questo sia già scritto da qualche parte sul Foglio1. In pratica il gioco consiste nel leggere il dato da una cella e riportarlo nella Finestra di Dialogo. In conclusione è possibile, con un paio di Macro relativamente semplici, gestire sia le caratteristiche strutturali della Finestra di Dialogo che i dati in essa contenuti.

dai tre valori componenti Rosso, Verde e Blu, letti dalle tre barre. A completamento piazziamo sul foglio anche cinque Pulsanti di Opzione che impostano cinque colori fondamentali. La seconda routine richiamata, che svolge questo compito, è la BASE.

Uso delle risorse di Windows. Uso delle risorse di Excel ed altro

VBA, come del resto anche il Visual Basic «normale», permette di accedere alle risorse del sistema. Nel successivo esercizio, presentato nelle figure 5 e 6, vediamo come utilizzare la Finestra File Apri di Windows. Leggiamo un file grafico e lo piazziamo, come immagine, sul foglio, in corrispondenza di una data cella.

Nel successivo esercizio, le figure sono anche il questo caso due, la 7 e la 8, vediamo come sfruttare da VBA le numerose Funzioni disponibili nel foglio normale. Abbiamo tre fogli, uno normale (si chiama FOGLIO) che ha solo un pulsante che serve per richiamare la Finestra di Dialogo che è disegnata nel secondo foglio (DIALOGO), ed infine il foglio MODULO con il codice. Si lavora solo sulla Finestra di Dialogo. Non diciamo altro, vi rimandiamo alla figura e al listato.

Nelle due successive figure, siamo alla 9 e alla 10, vediamo come creare, in una Finestra di Dialogo personalizzata, una Casella di Testo in cui inserire un testo lungo letto da un file in formato testuale.

Nella figura 11 vediamo gli oggetti Costanti, che iniziano con le due lettere «xl», e che in pratica permettono di utilizzare nell'applicazione le varie Finestre di Dialogo proprie di Excel. Anche in questo caso, lasciamo a voi la sperimentazione, si possono ipotizzare decine di casi applicativi.

Benedette Dialog Box

Una delle difficoltà nell'uso del Visual Basic for Application consiste nel fatto che esiste una tipologia speciale di foglio nel quale vanno disegnate, una per foglio, le Finestre di Dialogo che servono nella nostra applicazione. La Finestra va richiamata (ovviamente è un oggetto anche lei), vanno gestiti gli oggetti presenti, va gestito il passaggio dei dati da e verso il foglio.

Nel successivo esercizio, anche questo ben documentato nelle varie figure, dalla 12 alla 14, a corredo, vediamo come sia possibile automatizzare non tanto l'uso delle Dialog Box personalizzate quanto addirittura la loro costruzione.

In pratica creiamo un nostro sistema di codifica degli oggetti che poi viene usato dalla Macro che produce la Box. Vediamo infine una routine che serve per riempire le varie Caselle di Testo con dei valori presenti nel foglio.

Problematica Filtro Avanzato: Soluzione in Foglio

Excel 5.0, come noto, presenta rispetto alla sua precedente versione un netto miglioramento nella gestione dei dati: l'interfaccia si è fatta più intuitiva e l'utente è in grado di eseguire semplici Interrogazioni senza troppa fatica. Il meccanismo di gestione delle Interrogazioni prende il nome di «Filtro Automatico» e si trova nel menu Dati/Filtro (lo vediamo in fig. 15). Praticamente, in presenza di un archivio, compaiono accanto ai nomi dei campi dei piccoli pulsanti-freccia che permettono di impostare i filtri di selezione su ciascun campo/colonna dell'archivio. Istantaneamente le righe dell'elenco che non soddisfano i criteri impostati vengono temporaneamente nascoste finché non viene impostato qualche nuovo criterio, oppure viene selezionato il comando Dati/Filtro/Mostra Tutto, oppure viene disattivato il «Filtro Automatico».

Il limite di questo metodo sta nelle composizioni logiche del filtro (e quindi nell'uso degli AND e degli OR) in quanto il filtro automatico non permette di sommare più di due criteri. In pratica possiamo operare una selezione del tipo:

```
CITTA = ROMA OR CITTA = FIRENZE
```

se vogliamo selezionare solo i record di ROMA e FIRENZE, oppure:

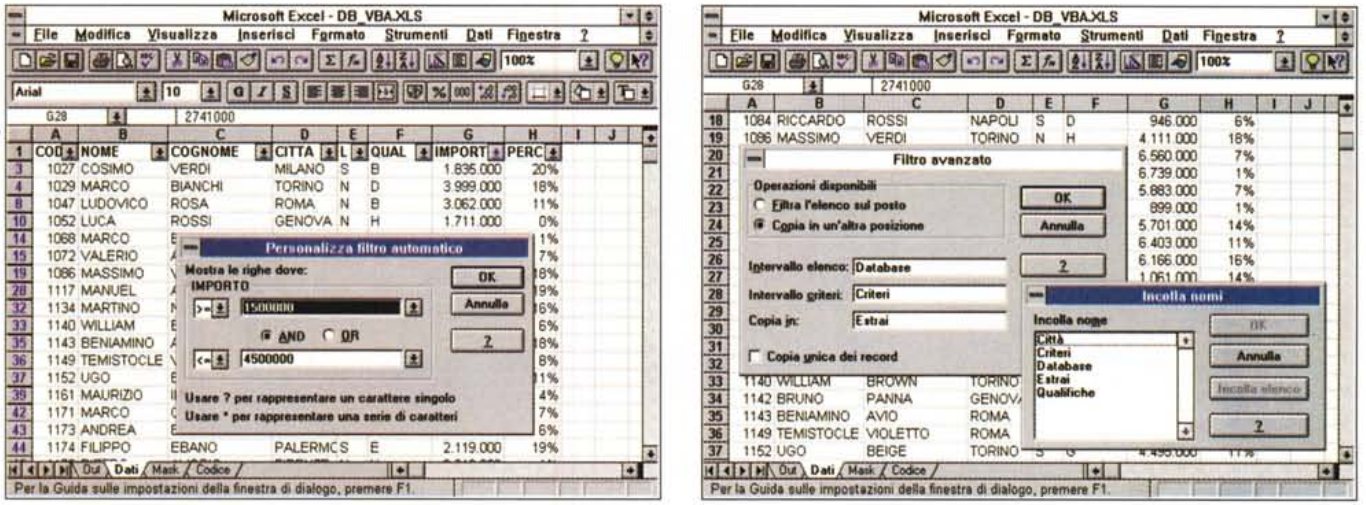


Figura 15 - Excel 5.0 - Il Filtro Automatico.
In Excel 5.0 è stato introdotto un nuovo sistema per eseguire estrazioni di dati letti da una tabella, organizzata in righe e colonne. Il nuovo sistema, che si chiama Filtro Automatico, ed è molto evoluto, si affianca al vecchio sistema, che ora si chiama Filtro Avanzato, e che invece permette, con uno sforzo leggermente maggiore, di eseguire operazioni anche molto complesse, non permesse dal Filtro Automatico.

Figura 16 - Excel 5.0 - Il Filtro Avanzato.
Nel prossimo esercizio, che chiameremo Applicazione Complessa, proponiamo una Box specializzata nella interrogazione dei Dati ed in altre cose, come la loro importazione da un file testuale, e la loro esportazione, dopo averli filtrati, verso Word. Prerequisito per mettere a punto efficacemente un sistema del genere è la conoscenza approfondita delle tecniche di manipolazione dei Dati usabili sul foglio. In pratica la conoscenza dei vari comandi presenti nel menu Dati.

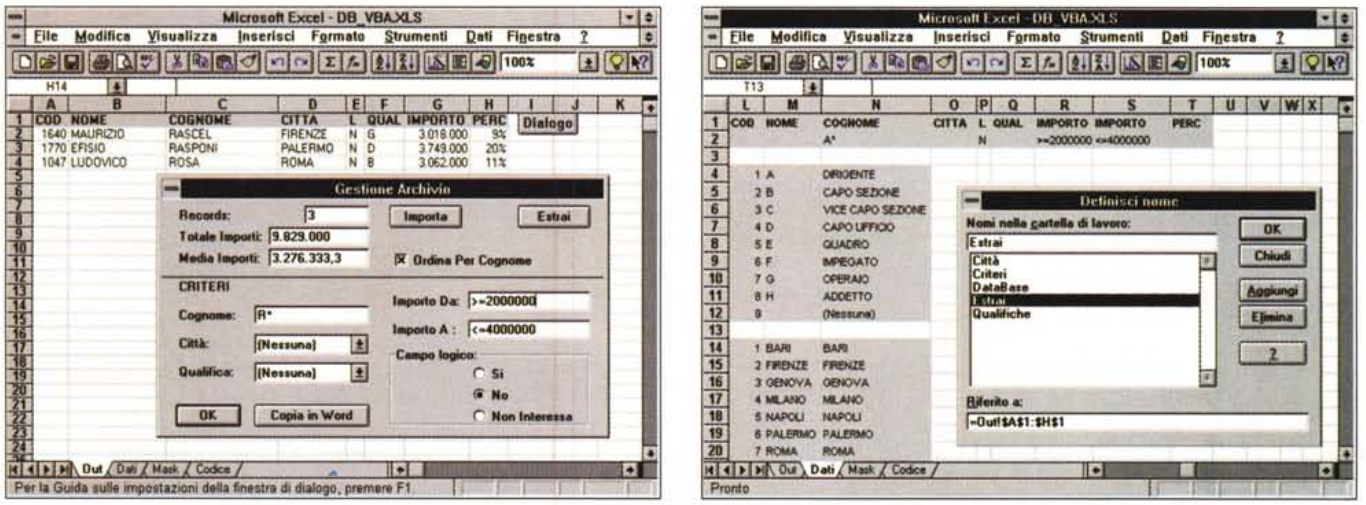


Figura 17 - Excel 5.0 - Visual Basic for Application - La Dialog Box dell'Applicazione Complessa.
Si tratta dell'esercizio finale, più complesso dei precedenti, ma i cui listati, stampati, non superano comunque una paginetta. L'obiettivo è quello di costruire una Finestra di Dialogo dalla quale eseguire una serie di operazioni di tipo Database sui dati presenti e/o caricati su un foglio apposito. I dati si possono caricare da un file testuale e, una volta caricati, si possono eseguire estrazioni e calcoli statistici. Il risultato dell'estrazione può essere scaricato su un documento Word.

Figura 18 - Excel 5.0 - Visual Basic for Application - Le zone d'appoggio dell'applicazione Complessa.
I fogli usati sono quattro. Si lavora solo sul foglio Out che contiene i dati risultanti dall'operazione di estrazione e contiene il pulsante che richiama la Finestra di Dialogo. C'è il foglio Dati che contiene la base dati e le varie zone di appoggio (le vediamo nella figura). C'è il foglio di tipo Finestra di Dialogo, che si chiama Mask, che contiene il disegno della stessa e c'è il foglio Modulo che contiene le varie subroutine.

```
IMPORTO >= 1500000 AND IMPORTO =<
4500000
```

se vogliamo selezionare solo i record il cui IMPORTO sia compreso fra 1.500.000 e 4.500.000.

Non possiamo però effettuare selezioni del tipo:

```
CITTA = ROMA OR CITTA = FIRENZE
OR CITTA = NAPOLI
IMPORTO >= 1500000 AND IMPORTO =<
4500000 AND IMPORTO <> 3000000
```

In questi ed in altri casi si impone l'utilizzo del «Filtro Avanzato» (lo vediamo in fig. 16), ovvero lo stesso metodo che fino alla precedente versione di Excel era l'unico.

Si tratta di creare (con il comando Inserisci/Nome/Definisci) tre zone di celle: la zona «DataBase», la zona «Criteri», la zona «Estrai». Esaminiamole in dettaglio.

La prima zona del foglio identifica

l'archivio che intendiamo gestire, comprendendo tutte le righe e le colonne nonché la riga delle intestazioni dei campi. La seconda zona deve essere composta da una riga in cui sono riportati esattamente i nomi di quei campi su cui imposteremo i criteri di filtro, ed almeno una riga vuota al di sotto della prima. La zona «Estrai» riporta i nomi dei campi di cui vogliamo estrarre il contenuto.

Effettuati questi adempimenti preliminari, nella riga vuota della zona «Criteri» inseriamo il nostro filtro, selezioniamo il comando Dati/Filtro/Filtro Avanzato e specifichiamo nella finestra di dialogo i nomi delle zone che abbiamo creato.

Se l'opzione «Copia in un'altra posizione» non viene attivata, si avrà un effetto di compressione delle righe analogo a quello ottenuto con il «Filtro Avanzato», altrimenti le righe che soddisfano i criteri verranno automaticamente ricopiate al di sotto della zona «Estrai».

Volendo risolvere il problemino precedente, quello dei tre criteri di selezione in fila, si hanno due soluzioni diverse. Per sommare le condizioni «OR» dobbiamo aggiungere alla zona «Criteri» tante righe quanti sono gli OR aggiuntivi, per sommare le condizioni «AND», invece che righe dobbiamo aggiungere colonne, ma con la stessa intestazione del campo interessato (entrambe le operazioni richedono la ridefinizione della zona «Criteri»).

Problematica Filtro Avanzato: Soluzione in VBA

L'esercizio che proponiamo mette in gioco molte delle possibilità offerte dal VBA in termini di automatizzazione di eventi e di controllo dei dati.

Le premesse: abbiamo un archivio su file in formato .TXT, magari proveniente da host, che vogliamo importare in un foglio Excel, ridistribuirlo in colonne e su di esso lanciare qualche piccola query attraverso una Finestra di Dialogo creata per l'occasione. I dati filtrati potranno poi essere scaricati in un documento di MS Word.

I vari «pezzi» che compongono questa piccola procedura li vedete nelle figure, dalla 17 alla 19, insieme al listato completo (nella 20). Analizziamone ora le parti più interessanti.

L'inserimento dell'archivio in formato testo nel foglio di lavoro che battezziamo «Dati» avviene aprendo il file in modalità di lettura sequenziale e scaricandolo una linea alla volta (da ritorno a capo a ritorno a capo) nelle celle della prima colonna del foglio:

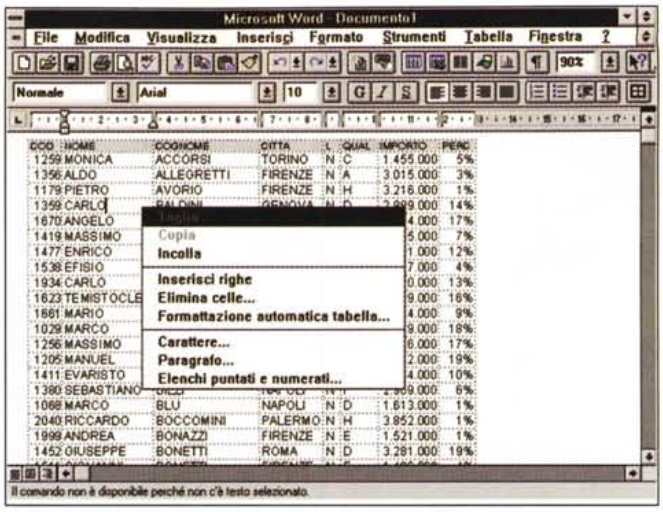


Figura 19 - Excel 5.0 - Visual Basic for Application - Applicazione Complessa: esportazione in Word. L'esportazione verso Word 6.0 viene realizzata sfruttando il comando SendKeys (tradotto in InviaTasti). Vengono esportati i dati risultanti dell'Estrazione che, in Word 6.0, assumono la forma di Tabella, gestibile operativamente ed esteticamente con gli specifici comandi.

```
ApriFile «PERSONE.TXT» Per Input
Come #1
K = 0
Esegui Mentre Not FineFile(1)
  Riga Input #1; X
  K = K + 1
  Fogli(«Dati»).Celle(K;1) = X
Ripeti
Chiudi #1
```

L'operazione di conversione del testo riportato nella colonna «A» in celle e colonne separate, l'abbiamo agevolmente registrata usando il potente comando Dati/Testo in Colonne. Successivamente abbiamo battezzato la zona risultante con il nome «DataBase». In seguito (importando nuove versioni del file, a lunghezza variabile) se ne occuperà (del battesimo) direttamente la procedura stessa con l'istruzione

```
CartellaDiLavoroAttiva.Nomi.Aggiungi
```

Da notare il fatto che, una volta definito un nome di zona, tutte le procedure VBA lo potranno sfruttare con la semplice istruzione Intervallo, senza ulteriori referenziazioni. Ad esempio, per cancellare la zona «DataBase» prima di un nuovo input da file .TXT:

```
Intervallo(«DataBase»).Cancella
```

Sempre nel foglio «Dati», a partire dalla cella L1, impostiamo la zona «Criteri», operazione fondamentale per poter lanciare le Query di cui sopra. I campi sono gli stessi che vengono riportati dal file «PERSONE.TXT». Tra questi, in particolare, useremo per le interrogazioni «COGNOME», «CITTA», «L» (campo logico i cui valori sono solo «S» o «N»), «QUAL» (qualifiche ipotetiche, codificate con lettere da A ad H), «IMPORTO» (che riportiamo due volte in quanto su questo campo opereremo selezioni del tipo «AND»).

Nel Foglio di Lavoro, che chiamiamo «Out», definiamo la zona «Estrai», sotto cui dovranno via via apparire i record che soddisfano i «Criteri» impostati.

È ora di creare la Finestra di Dialogo personalizzata il cui risultato finale è visibile in figura. Sarà da questa finestra che lanceremo tutte le nostre interrogazioni senza dover mai passare al foglio.

Le caselle «Cognome», «Città», «Qualifica», «Importo Da», «Importo A» ed i pulsanti di opzione «Si», «No», «Non Interessa» sono tutti collegati alla procedura «Aggiorna», che si occupa di riportare i dati inseriti nella finestra di dialogo direttamente nelle corrispondenti celle della zona dei «Criteri» (foglio di lavoro «Dati»): «Cognome» in N2, «Città» in O2, e così via.

Le prime tre caselle di modifica nella Box, in realtà sono alimentate dalla procedura. Il numero totale dei record, la somma e la media della colonna «IMPORTO» sono elegantemente fornite dalle funzioni standard del foglio Excel DB.CONTA.VALORI(), DB.SOMMA() e DB.MEDIA(). Queste funzioni restituiscono istantaneamente il risultato non appena variano i dati nella zona dei criteri. Pertanto la finestra di dialogo ci riporta, subito, il numero dei record che soddisfano i criteri appena specificati, oltre alla somma e alla media dei loro importi.

La procedura VBA richiama tali calcoli con le istruzioni:

```
RC=Applicazione.Db.ContaValori(Intervallo(«DataBase»);«COGNOME»;Intervallo(«Criteri»))
SOM=Applicazione.Db.Somma(Intervallo(«DataBase»);«IMPORTO»;Intervallo(«Criteri»))
MED=Applicazione.Db.Media(Intervallo(«DataBase»);«IMPORTO»;Intervallo(«Criteri»))
```

Le zone N4:N12 e N14:N22 del foglio «Dati» servono ad alimentare le box

```
Procedura ApriTXT()
Con Fogli("Dati")
    .Seleziona
    Intervallo("Database").Cancella
    K = 0: ApriFile "PERSONE.TXT" Per Input Come #1
    Esegui Mentre Not FineFile(1)
        Riga Input #1; X: K = K + 1
        .Celle(K; 1) = X
    Ripeti
    .Colonne("A:A").Seleziona
    Selezione.TestoInColonne Destinazione:=Intervallo("A1"); TipoDati:= _
        xlDelimitate; QualificatoreTesto:=xlVirgolette Doppie; _
        DelimitatoreConsecutivo:=Falso; Tabulazione:=Vero; PuntoEVirgola _
        :=Falso; Virgola:=Falso; Spazio:=Falso; Altro:=Falso; _
        InfoCampo:=Matrice(Matrice(1; 1); Matrice(2; 1); Matrice(3; 1); Matrice( _
        4; 1); Matrice(5; 1); Matrice(6; 1); Matrice(7; 1); Matrice(8; 1))
    .Celle(1; 1).Seleziona
    Y = ElimSpazioSx(Str(K))
    CartellaDiLavoroAttiva.Nomi.Aggiungi _
    Nome:="DataBase"; RiferitoAR1C1:="=R1C1:R" & Y & "C8"
    Fogli("Out").Seleziona
Fine Con
    Chiudi #1
    Aggiorna
Fine Procedura

Procedura Gestione_DB()
    Aggiorna
    FogliDialogo("Mask").Mostra
Fine Procedura

Procedura Aggiorna()
    Con FogliDialogo("Mask")
    Imposta D = Fogli("Dati")
    D.Celle(2; 14).Valore = .CaselleDiModifica(4).Testo
    D.Celle(2; 18).Valore = .CaselleDiModifica(5).Testo
    D.Celle(2; 19).Valore = .CaselleDiModifica(6).Testo
    QUAL = Applicazione.CercaVert(.PiuADiscesa(1); Intervallo("Qualifiche"); 2)
    CIT = Applicazione.CercaVert(.PiuADiscesa(2); Intervallo("Città"); 2)
    LGC = ""
    Se .PulsantiDiOpzione(1) = 1 Allora LGC = "S"
    Se .PulsantiDiOpzione(2) = 1 Allora LGC = "N"
    D.Celle(2; 17).Valore = QUAL
    D.Celle(2; 15).Valore = CIT
    D.Celle(2; 16).Valore = LGC
    RC = Applicazione.DbContaValori(Intervallo("Database"); "COGNOME"; Intervallo("Criteri"))
    SOM = Applicazione.DbSomma(Intervallo("Database"); "IMPORTO"; Intervallo("Criteri"))
    Se RC > 0 Allora
        MED = Applicazione.DbMedia(Intervallo("Database"); "IMPORTO"; Intervallo("Criteri"))
    Altrimenti
        MED = 0
    Fine Se
    .CaselleDiModifica(1).Testo = RC
    .CaselleDiModifica(2).Testo = Formato(SOM; "#.##0")
    .CaselleDiModifica(3).Testo = Formato(MED; "#.##0,0")
Fine Con
Fine Procedura

Procedura Estrazione()
Con Fogli("Dati")
     .Intervallo("DataBase").FiltroAvanzato Azione:=xlFiltroCopia; _
        IntervalloCriteri:=.Intervallo("Criteri"); CopiaAIntervallo:= _
        Fogli("Out").Intervallo("Estrai"); Unico:=Falso
     Se FogliDialogo("Mask").CaselleDiControllo(1) = 1 Allora
        Fogli("Out").Intervallo("C1").Ordina Chiave1:=Intervallo("C1"); Ordine1:=xlCrescente; _
            Intestazione:=xlIpotesi; OrdinePersonalizzato:=1; _
            RilevaMaiuscole:=Falso; Orientamento:=xlDaAltoInBasso
     Fine Se
Fine Con
Fine Procedura

Procedura A_Word()
    TOT = ElimSpazioSx(Str(FogliDialogo("Mask").CaselleDiModifica(1).Testo + 1))
    Fogli("Out").Intervallo("A1:H" & TOT).Copia
    X = EseguiProgr("c:\WINWORD6\WINWORD.EXE"; 1)
    AttivaApp X
    InviaTasti "^v"
Fine Procedura
```

Figura 20 - Excel 5.0 - Visual Basic for Application - Applicazione complessa - Listati.
I listati delle varie routine sono relativamente corti in relazione alla complessità delle operazioni che vengono eseguite. Come specificheremo meglio nel testo, molte porzioni di questo codice sono state costruite sfruttando i servizi del registratore. In particolare vengono registrati i comandi, dalla sintassi particolarmente complessa, che eseguono operazioni di tipo DataBase.

«Città» e «Qualifica» della Finestra di Dialogo. Come è noto, però, le Caselle a Discesa restituiscono la posizione della voce di elenco selezionata, non la voce stessa. Scegliendo la qualifica di dirigente, ad esempio, il valore che si ottiene è 1 e non «DIRIGENTE».

Col metodo già visto in tante altre occasioni sfruttiamo la funzione CERCA.VERT() per trasformare in dati utilizzabili il risultato delle selezioni delle caselle a discesa. Definiamo, sempre nel foglio «Dati», due zone in cui ad un valore numerico progressivo corrispondono, per la zona «Qualifiche» (L4:M12), la decodifica necessaria per effettuare criteri di selezione nella colonna «QUAL», e per la zona «Città», (L14:M22) le città presenti nell'archivio.

Perciò, la qualifica e la città scelte diventano:

```
QUAL=Applicazione.CercaVert(FogliDialogo(«Mask»).PiuADiscesa(1);Intervallo(«Qualifiche»);2)
CIT =Applicazione.CercaVert(FogliDialogo(«Mask»).PiuADiscesa(2);Intervallo(«Città»);2)
```

Il pulsante «Estrai » sulla Dialog Box è collegato alla procedura «Estrazione», che ricopia dal foglio «Dati» al foglio «Out» i record che soddisfano i criteri impostati, eventualmente ordinandoli in base alla colonna «COGNOME» (cella C1) a seconda che la check box «Ordina Per Cognome» sia stata selezionata o meno.

Entrambe le operazioni, quella di estrazione e quella di ordinamento, la cui sintassi è particolarmente complessa, sono state registrate utilizzando i comandi Dati/Filtro/Filtro Avanzato e Dati/Ordina e poi ottimizzate.

Per passare i dati a Word si è scelta la via più semplice che è anche la più funzionale: copiare l'intervallo delle celle dei record estratti e scaricarlo direttamente in un foglio Word:

```
Fogli(«Out»).Intervallo(«A1:H» &
TOT).Copia
X = EseguiProgr(«C:\WINWORD6\WIN-
WORD.EXE»;1)
AttivaApp X
InviaTasti «^v»
```

In cui TOT è il numero di record estratti + 1 e in cui i tasti «^V» (Ctrl+V) eseguono il comando Incolla una volta giunti in Word.

Concludiamo segnalando i collegamenti degli oggetti posti nella finestra di dialogo alle varie procedure:

```
pulsante «Importa»        procedura APRITXT()
pulsante «Estrai»         procedura ESTRAZIONE()
pulsante «Copia in Word»  procedura _WORD()

caselle «Cognome:»
        «Città:»
        «Qualifica:»
        «Importo Da:»
        «Importo A:»
opzioni «Si»
        «No»
   «Non interessa» procedura AGGIORNA()
```

Conclusioni

Continuiamo ad essere perplessi sul VBA. In alcuni casi risulta essere semplicissimo ed immediato (e anche divertente) da usare, in altri casi sembra un po' contorto, ad esempio quando occorre riferirsi ad oggetti particolari. In alcuni casi probabilmente si tratta di peccati di gioventù, che speriamo, nell'immediato futuro, stiamo parlando di Windows 95, vengano in gran superati.

MC