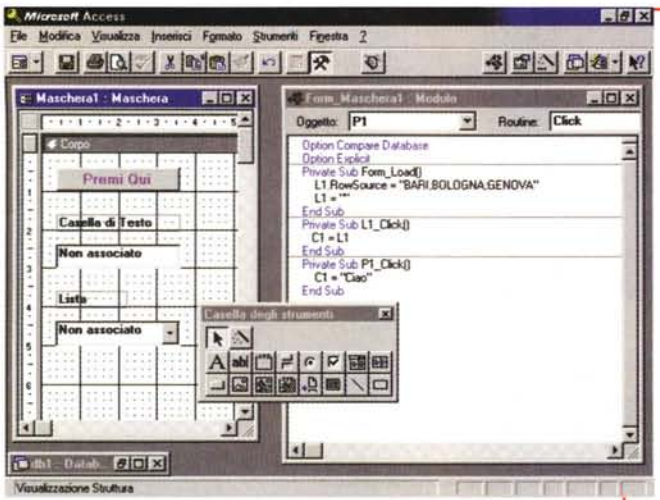

Figura 3 - I principi della Programmazione Object Based - Con MS Access 7.0.
Il Basic di Access 7.0, che si chiamava Access Basic fino alla versione 2.0 del prodotto, è finalmente diventato Visual Basic for Application con la versione 7.0, la prima per Windows 95. L'ambiente visuale, nel quale si costruiscono Maschere e Report, assomiglia moltissimo a quello del Visual Basic. La differenza principale è costituita dalle numerose proprietà (in più rispetto a quelle disponibili per gli oggetti del VB) che consentono di legare i vari oggetti direttamente ai dati di un Database.

esempio, per un casella di testo, possono essere impostate le proprietà:

- posizione nella finestra di appartenenza,
- dimensione,
- tipo, dimensione e attibuti del font con cui il contenuto della box viene visualizzato,
- estetica fine della casella, come colore, effetto rilievo o incassato, ecc.

E' chiaro che quando nell'ambiente visuale si scelgono gli oggetti, si posizionano e se ne configurano le caratteristiche, le varie impostazioni vengono convertite in un codice, più o meno nascosto, a seconda del prodotto. Alcuni di questi, ad esempio il dBase 5.5 della Borland, sono "double face" nel senso che dispongono di due ambienti paralleli, quello visuale e quello tradizionale. E' il programmatore che sceglie quello dei due che più gli conviene usare: disegnare nell'ambiente grafico oppure scrivere righe di codice, che provocano anche il disegno dell'oggetto.

Con il Visual Basic della Microsoft, invece, non è possibile vedere il codice generato quando si lavora nell'ambiente visuale. Si può al massimo leggere come viene convertito il lavoro nell'ambiente visuale nel file generato quando si salva, ma non lo si può scrivere direttamente.

Abbiamo detto che l'applicazione non si può ridurre a pura esteriorità. Oltre agli oggetti e alle proprietà occorre del codice, spezzettato in routine che vengono eseguite al verificarsi di un evento su un oggetto. Per questa affermazione, che può sembrare troppo semplificativa, non esistono eccezioni. Esempi di eventi di vario tipo:

- il lancio dell'Applicazione,
- l'entrata del cursore in una Casella di Testo,
- l'uscita del cursore da una Casella di Testo,
- la modifica del testo in una Casella di Testo,
- il fatto che, dall'interno della Casella di Testo, viene premuto il tasto Alt+F12,
- l'entrata nel record successivo quando si scorre un Tabella di un Database,
- il Click, o il doppio Click, con il tasto sinistro ma anche con il tasto destro, su un qualsiasi oggetto,
- il semplice passaggio del mouse su un qualsiasi oggetto.

Esistono eventi di mouse, eventi di tastiera, eventi di esecuzione, e tanti altri.

Se fino ad ora non avete capito nulla...

Compito del programmatore è dunque quello di scomporre l'applicazione, per quanto complessa, in una serie di oggetti elementari, ed è quello di individuare quali proprietà di ciascun oggetto debbano essere coinvolte e quali siano gli eventi da intercettare che modificano una o più proprietà degli oggetti.

La sua abilità consisterà conseguentemente nell'intuire quali siano gli elementi del gioco, quali gli oggetti, quali le proprietà, gli eventi e i metodi, e nell'assemblarli opportunamente.

Usiamo il termine Intuizione perché pensiamo che non sia possibile né necessario, al programmatore, conoscere a priori tutti gli oggetti e tutte le proprietà e tutti gli eventi degli oggetti. Al momento opportuno intuisce che deve esistere un certo oggetto, una certa proprietà di quell'oggetto, un certo evento che subisce quell'oggetto.

Ci rendiamo conto che una semplice elencazione delle regole teoriche del Modello Object Based può lasciare, nella migliore delle ipotesi, qualche dubbio, oppure può, e questa è la peggiore delle ipotesi... non far capire assolutamente nulla. Proviamo quindi dapprima a formulare una descrizione alternativa del Modello Object Based per poi esaminare una piccola serie di esercizi introduttivi, che hanno lo scopo di mettere i pratica la mezza dozzina di regole.

Gli Oggetti che si usano nella programmazione Object Based sono in pratica degli elementi "semilavorati", per realizzare i quali qualcuno ha scritto un bel po' di codice. Quando inseriamo uno di questi oggetti nella nostra applicazione sfruttiamo, anche se non possiamo in nessun modo vederlo, il codice scritto dal "programmatore sconosciuto". Il nostro compito è quello di aggiungere altro codice necessario per inglobare correttamente l'oggetto in questione nella nostra applicazione.

Ci sono oggetti più semplici, ad esempio un Pulsante o una Casella di Testo, e oggetti più complessi, che possono essere paragonati, per complessità, ad applicativi a tutti gli effetti. Si pensi ad un Oggetto di tipo Chart, che è un programma di grafica a tutti gli effetti, al quale vanno passati semplicemente i dati numerici e i dati testuali da graficare.

Quando si costruisce un'applicazione Object Based si possono utilizzare, in generale, tre tipi di oggetti:

- oggetti messi a disposizione dal prodotto, sempre disponibili,
- oggetti messi a disposizione da "terze parti", caricabili via file (es. librerie OCX),
- oggetti messi a disposizione da altre applicazioni e attivabili sfruttando la tecnologia "OLE Automation".

Rimandiamo a dopo l'esame di questa distinzione.

L'influenza del prodotto di ambiente

Prima di passare a descrivere i nostri esercizi introduttivi dobbiamo capire in

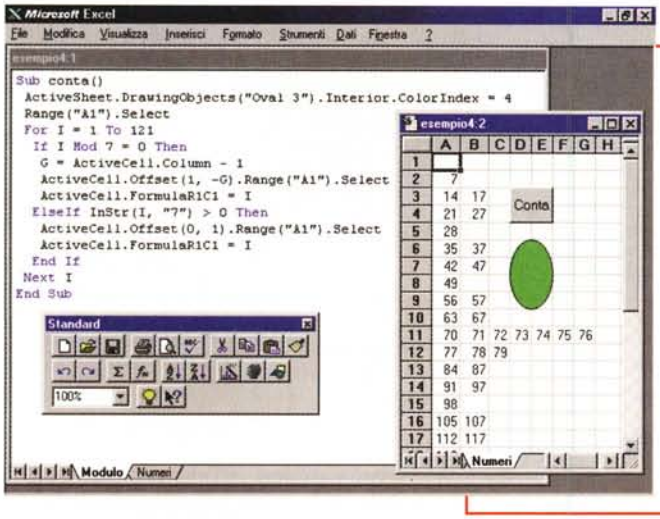

Figura 4 - I principi della Programmazione Object Based - Con MS Excel 7.0.
Si usa il linguaggio interno di Excel, il Visual Basic for Application, che è accomunato al Visual Basic "standard" per quanto riguarda le istruzioni di programmazione, e che permette anche di utilizzare la "caterva" di oggetti presenti nell'ambiente Excel, e quindi Cartelle di Lavoro, Fogli, Intervalli di Celle, Celle singole, Oggetti "volanti" sul Foglio, ecc.

che modo il prodotto che stiamo usando influenzi il modo di programmare per Oggetti, e lo faccia differire da quello praticabile con altri prodotti.

In ogni prodotto, che disponga di un ambiente di programmazione Object Based, esistono due gruppi di oggetti: quelli, che potremo definire Oggetti Windows, presenti in qualsiasi applicazione Windows, e quelli specifici del prodotto. Ad esempio, se il prodotto è uno Spreadsheet, ci sarà l'Oggetto Intervallo di Celle, se il prodotto è un DBMS, ci sarà l'Oggetto Tabella.

Fa eccezione il Visual Basic che non ha un ambiente di base e quindi utilizza, di base, solo oggetti "standard" Windows.

Le regole generali enunciate all'inizio sono indipendenti dal prodotto e quindi continuano a valere, solo che si applicano alle tipologie di oggetti proprie del prodotto che si sta usando.

Una prima serie di esercizi

Cominciamo con MS Visual Basic, prodotto "responsabile" più di ogni altro della nascita della tecnologia Object Based. Nella figura 1 vediamo il suo composito ambiente operativo multifinestra. C'è la finestra, o le finestre, che corrispondono ai Form dell'applicazione che si sta sviluppando, ci sono alcune finestre con i vari strumenti operativi, la Casella degli Oggetti (il sinonimo di Oggetto è Controllo) disponibili, al quale possono essere aggiunti i vari Oggetti su file OCX, la Tavolozza con i Colori, l'importantissima Finestra delle Proprietà, presente in tutti i prodotti Object Based, che serve per impostare le proprietà dell'Oggetto selezionato. In alto, e solo in alto, la cornice del Visual Basic, con il Menu e la Barra degli Strumenti. Altre finestre che affollano l'ambiente operativo visuale sono quella, detta Progetto, che mostra l'elenco delle Form e dei Moduli che compongono l'Applicazione, ogni voce del quale corrisponde ad un file specifico, e quella che mostra il Codice. Ogni porzione di programma è in pratica una Routine che viene eseguita al verificarsi dell'evento indicato sull'oggetto indicato. La sua sintassi generica è:

```
Sub Oggetto.Evento
    Oggetto.Proprietà=
        nuovovalore
End Sub
```

Il primo programma lo vediamo in figura 2.

Si tratta di un programma Visual Basic in cui usiamo alcuni oggetti standard Windows: un Pulsante, una Barra di Scorrimento e una Casella di Testo.

Nasce il problema della lingua. Nelle versioni originali i tre oggetti sono rispettivamente CommandButton, HorizontalScrollBar e TextBox.

Impostiamo solo due Proprietà del Pulsante (Caption: Pigia e Text: "" (vuoto)) e due Proprietà della ScrollBar (Min: 0, che è già il valore di default, e Max: 100).

Per impostare, o semplicemente per vedere, le proprietà di un Oggetto lo si seleziona e si preme F4, oppure si preme il Pulsante Properties sulla Barra degli Strumenti. E' buona norma esplorare le proprietà di ciascun oggetto ogni volta che capita l'occasione. Bisogna arrivare al punto di riuscire ad intuire quali siano le proprietà di ciascun oggetto, e quali di queste siano quelle che servono volta per volta.

Gestiamo poi l'evento Click del Pulsante (per il quale conserviamo il nome di Default: Command1) e l'evento Change della ScrollBar (nome: HScroll1).

Al verificarsi del Click sul pulsante Command1 vogliamo che la Casella di Testo mostri la stringa Buongiorno:

```
Sub Command1.Click
    Text1.Text = "Buongiorno"
End Sub
```

Poi, all'evento Change della ScrollBar, vogliamo che la Casella di Testo mostri il valore numerico assunto dalla ScrollBar:

```
Sub HScroll1.Change
    Text1.Text =HScroll1.Value
End Sub
```

Per scrivere queste due routines basta fare un doppio Click sull'oggetto del quale gestiamo l'evento. Appare l'Editor che propone in alto Oggetto ed Evento (rimanendo nell'Editor si può scegliere qualsiasi combinazione di Oggetto ed Evento) e che già propone le righe iniziale e finale della routine.

Scritte, a mano, le istruzioni delle routine si può provare direttamente il programma premendo il comando di menu Run o il pulsante Run o il tasto funzione F5.

Abbiamo in questa maniera gestito, per la prima volta, Oggetti, Proprietà ed Eventi.

Abbiamo imparato la Programmazione Object Based: infatti, dal punto di vista concettuale, la programmazione Object Based è tutta qui.

Le difficoltà nascono quando si maneggiano tanti oggetti e oggetti complessi, quando si trattano proprietà ostiche, quando si gestiscono eventi poco intuitivi, ma le regole fondamentali rimangono invariate.

Occorre comunque fare una serie di precisazioni.

In linea di massima, in una generica routine, vengono coinvolti due oggetti, quello sul quale si genera l'evento e quello che subisce l'evento. Si fa Click sul Pulsante, ma chi cambia è la Casella di Testo. I principianti si confondono facilmente.

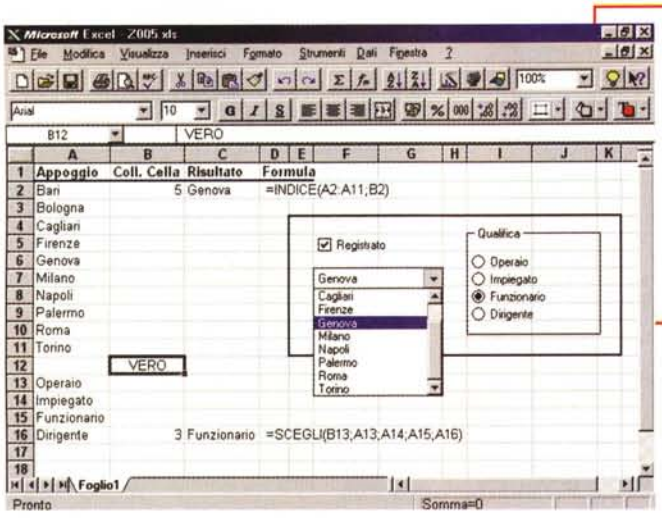

Figura 5 - MS Excel 7.0 - Lavorare con oggetti senza programmare.
La programmazione Object Based si poggia su principi generali semplici da enunciare e che prescindono dal prodotto che poi si intende usare. La vera difficoltà nel suo utilizzo sta nel fatto che i modi per mettere in pratica tali principi sono tantissimi, e cambiano, anche sostanzialmente, tra i vari prodotti che la utilizzano. Qui vediamo come, con Excel, sia possibile inserire Oggetti in un foglio di lavoro, legarne alcune proprietà direttamente a celle del foglio senza dover necessariamente scrivere del codice. Si tratta di un primo livello, assolutamente rudimentale, di programmazione Object Based.

nostra applicazione.

Inseriamo ora nella Form anche un Oggetto Timer, del quale impostiamo la proprietà Interval a 10000, che significa "ogni 10 secondi". Il relativo Evento Timer ci permette di indicare cosa deve fare la nostra applicazione "ogni dieci secondi":

```
Sub Timer1.Timer
    Text1.Text = "Sono passati altri 10 secondi"
End Sub
```

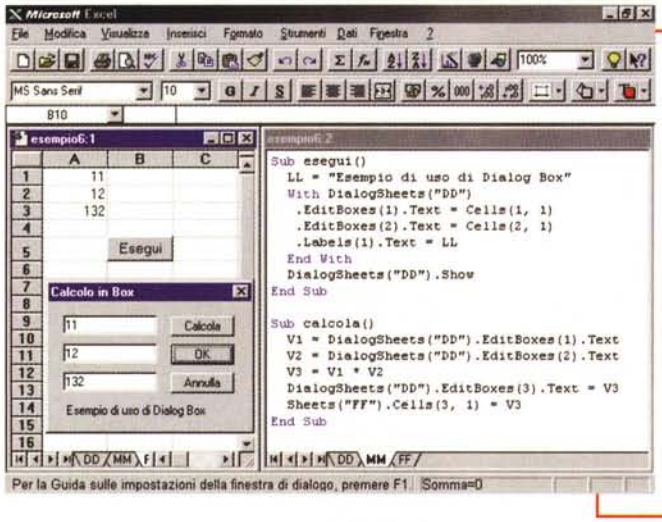

Figura 6 - MS Excel 7.0 - Uso di un foglio di tipo Finestra di Dialogo.
Nella figura precedente abbiamo visto come si possano appoggiare alcuni oggetti standard Windows direttamente sul foglio di lavoro: ComboBox, ScrollBar, Pulsanti, ecc. Non si possono inserire sul foglio Caselle di Testo, sostituibili dalle normalissime celle. Una strada alternativa, più adatta ad un programmatore, e' quella di usare una Finestra di Dialogo, che può ospitare tutti i tipi di oggetti Windows. Il problema più importante diventa quello del passaggio dei dati dal foglio alla finestra di dialogo e viceversa.

Questo è un esempio di Oggetto meno intuitivo, soprattutto per il fatto che in fase di esecuzione scompare (giustamente scompare).

Passiamo ad Access con il secondo esercizio che vediamo in figura 3.

Anche in questo caso abbiamo una Form (che in Access 7.0 italiano si chiama Maschera) nel cui interno abbiamo piazzato alcuni oggetti Windows: la solita Casella di Testo, tuttofare, in quanto la usiamo per ricevere e visualizzare qualche cosa e una ComboBox, in italiano Casella Combinata e un Pulsante.

Facciamo una breve pausa per lamentarci del fatto che il nome degli oggetti tipici di Windows, ad esempio la ComboBox o la ListBox, varia da prodotto a prodotto e da versione nazionale a versione nazionale. Non è che questo sia un problema per l'esperto, lo diventa solo per chi deve parlare, come in questo caso, un po' genericamente, di tutti i prodotti in circolazione.

Ritornando al nostro esercizio vediamo che l'unica novità, rispetto a quanto realizzato nell'esercizio precedente, consiste nell'uso della ComboBox che va alimentata, in qualche maniera e prima del suo uso, di dati. Si può fare in due modi, il più banale dei quali è il passaggio diretto dei dati, tramite la proprietà RowSource, ed è quello che abbiamo utilizzato. Il sistema più nobile invece è quello di alimentare la Lista con i dati di una Tabella.

La Programmazione Visuale realizzata con Visual Basic è assolutamente

Altra classica confusione è quella tra la proprietà Name e la proprietà Text della Casella di Testo. Text è quello che appare nelle Casella, Name è il suo nome "da programma".

Esistono Proprietà di default ed Eventi di default. Se si fa doppio click su un Pulsante viene attivato l'evento Click, se si fa doppio click sulla ScrollBar viene attivato l'evento Change. Sono gli eventi di Default. Se non si controlla che l'evento sia quello giusto si rischia di programmare un evento sbagliato.

Inoltre l'espressione:

```
Text1 = "Buongiorno"
```

sottointende la proprietà Text, che è la proprietà di Default.

Nella nostra applicazione Visual Basic ci sono, fino ad ora, quattro oggetti: la Form, il Pulsante, la Casella di Testo, la ScrollBar. Ognuno di questi ha decine di Proprietà e può subire decine di Eventi. Compito del programmatore è quello di gestire solo le proprietà e gli eventi di suo interesse. Quelli non movimentati non danno nessun fastidio all'applicazione, anche se entrano nell'applicazione.

Consideriamo la facilità con la quale abbiamo inserito nella nostra applicazione un oggetto complesso, che ha un suo aspetto, una sua operatività, come la ScrollBar. Il nostro compito è solo quello di prelevarla dal Pannello degli Strumenti, di piazzarla nella Form, e poi quello di gestire quelle due o tre proprietà di nostro interesse. E' chiaro che dentro un oggetto "ScrollBar" ci sono decine di righe di programma che in tal modo entrano, in un certo senso, nella

analoga a quella realizzata con MS Access. Variano un po', ma neanche tanto, i metodi di accesso ai Dati di un Database, sui quali torneremo tra pochissimo.

Il terzo esercizio è stato realizzato con MS Excel (figura 4).

In questo caso il linguaggio risente pesantemente dell'ambiente che, come noto, è costituito di Cartelle di Lavoro, di Fogli, di Intervalli di Celle e di Celle singole, che sono elementi che contengono i dati dell'applicazione. Anche questi sono oggetti, e quindi come tali vanno identificati, come tali hanno proprietà, subiscono eventi, ecc.

Dovendo utilizzare, per l'applicazione che si sta costruendo, degli oggetti, si possono "appoggiare" direttamente sul foglio oppure si possono appoggiare su una Form, che, in Excel, si realizza usando uno speciale tipo di foglio che si chiama Finestra di Dialogo.

In particolare il nostro programma conta da 1 a 121, visualizza i numeri divisibili per 7 e quelli che contengono un 7, posizionandoli nelle celle del foglio (si va a riga nuova quando si trova un numero divisibile per 7). Il programma inoltre colora un Oggetto Grafico, di forma ovale, piazzato sul foglio.

Nel listato, che si vede in secondo piano, notiamo le istruzioni (un po' complicate) che servono per identificare l'Oggetto Grafico, e quelle che servono per puntare una data cella. Le istruzioni e le funzioni che servono per contare e per testare i numeri (che devono essere divisibili per 7 e devono contenere un 7) sono più tradizionalmente Basic.

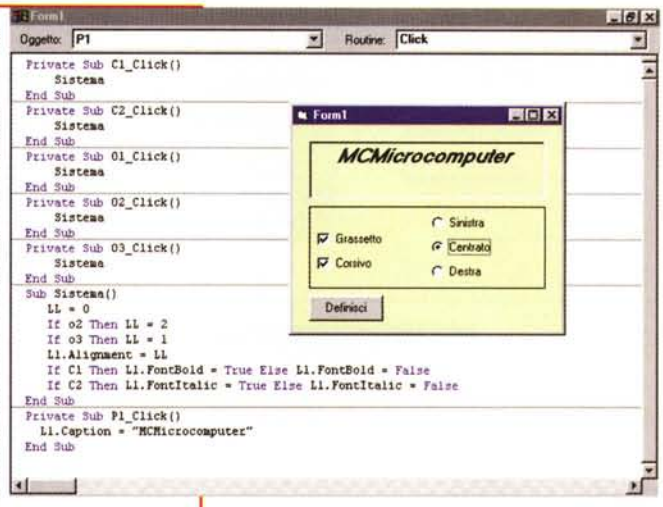

Figura 7 - Visual Basic e Visual Basic for Application - Esempio di procedura.
L'uso del linguaggio è relegato nei meandri della programmazione degli eventi. I prodotti Object Based della Microsoft utilizzano tutti, come base, il vecchio Basic, opportunamente aggiornato per metterlo in grado di gestire Oggetti, Proprietà ed Eventi. Nell'esercizio vediamo due microprocedure. La prima, legata all'evento Click sul Pulsante P1, è la semplice definizione della Proprietà Caption dell'Oggetto Label, chiamato L1. La seconda è più complessa in quanto contiene una serie di strutture di controllo, in base alle quali la scritta della Label viene sistemata all'interno della sua Cornice.

A proposito di Excel

Per Excel va spesa qualche parola in più.

Fino alla versione 4.0 Excel disponeva di un suo linguaggio di programmazione Macro sufficientemente completo, al punto da essere usato per la realizzazione di vere e proprie applicazioni "chiuse" basate sul foglio elettronico, che comunque è la base di Excel.

Porzioni di programma potevano poi essere costruite sfruttando la comoda tecnica del Registratore di Macro, che permette di trasformare in istruzioni di programmazione tutte le operazioni che si eseguono lavorando normalmente su foglio.

L'introduzione, avvenuta con Excel 5.0, della Programmazione Object Based e del Linguaggio Visual Basic for Application ha comportato una vera e propria rivoluzione, per il fatto che tra vecchio modo di programmare e nuovo modo di programmare non c'è nessuna relazione. Non esiste neanche un convertitore di procedure che permetta di utilizzare in VBA le procedure Macro 4.0.

Inoltre va considerato il fatto che mentre la programmazione con Access e con Visual Basic è, in un certo senso, obbligata in quanto ci sono Form ed Oggetti, con Excel vi sono, in fase di impostazione dell'applicazione, numerose alternative. Si possono appoggiare gli oggetti direttamente sul foglio (lo vediamo in figura 5) ed in questo caso, specificando opportunamente le proprietà dei vari oggetti e inserendo opportune formule sul foglio, non è necessario scrivere righe di programma, oppure si possono creare delle Form, dette Finestre di Dialogo, nelle quali si possono inserire i vari oggetti (tutti quelli standard Windows). Il problema principale, e si risolve da procedura, diventa quello di collegare i vari oggetti della Box alle Celle del Foglio di Lavoro (lo vediamo in figura 6).

Anche nel caso del VBA di Excel ogni oggetto ha un suo nome. C'è però una difficoltà dovuta al fatto che in Excel è molto spinta la gerarchizzazione degli oggetti, per cui le espressioni per indicare, ad esempio, una data Cella di un dato Foglio di una data Cartella, possono essere complesse.

Altro elemento, abbastanza complicato, che entra in gioco quandi si programma con Excel, è la "Selezione", ovvero il fatto che oltre alla possibilità di indicare nelle espressioni direttamente delle celle, si può sia indicare la selezione attiva, sia selezionare, da programma, le celle desiderate.

In pratica la programmazione si complica un bel po', rispetto a quella molto lineare del VB semplice, per due motivi: per il maggior numero di elementi in gioco e per la maggior difficoltà nel referenziare gli oggetti in gioco.

Il problema del Linguaggio

Come chiarito in precedenza la programmazione è relegata "in seconda linea". Anzi molte delle funzionalità di un' applicazione sono risolte direttamente a livello di proprietà per cui non è necessario programmare, oppure al massimo occorre una sola istruzione per assegnare un nuovo valore ad un proprietà di un oggetto.

Fermo rimanendo il principio fondamentale che il codice è associato ad un evento, può essere necessario che tale codice sia più complesso, che richieda più istruzioni: occorre quindi usare un Linguaggio.

La Microsoft, legata per motivi storici ad affettivi al Basic, lo ha "riesumato" e lo ha integrato nella tecnologia Object Based, realizzando il Visual Basic, che comprende in pratica sia le classiche istruzioni e funzioni del Basic (nelle sua numerose varianti GW Basic, Quick Basic, Turbo Basic, ecc.) sia le istruzioni necessarie per gestire gli oggetti e le proprietà.

Quindi il principiante che decidesse

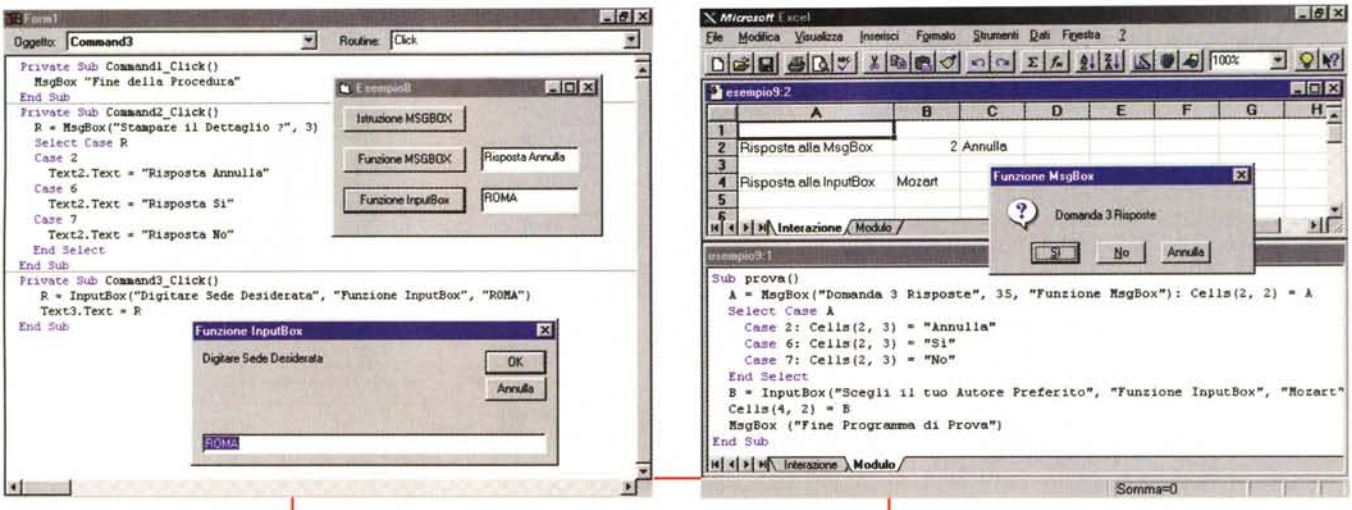

Figure 8, 9 - Visual Basic e Visual Basic for Application - Istruzioni di interazione.
La programmazione, che si concretizza, al solito, nella scrittura di righe di codice, va dunque associata al singolo evento del singolo oggetto. Il linguaggio con il quale vanno scritte le istruzioni è il Visual Basic, lontanissimo erede dei primi Basic, che, nelle primissime generazioni dei PC, erano addirittura residenti in ROM. Molte istruzioni sono rimaste invariate, molte sono del tutto nuove, altre sono state modificate anche per tener conto del nuovo ambiente grafico. Vediamo le istruzioni MessageBox e InputBox, evoluzioni della vecchia istruzione Input.

di dedicarsi al Visual Basic dovrebbe "studiare" anche un po' di Basic, le sue istruzioni fondamentali che servono per realizzare dei cicli (FOR .. NEXT, DO .. WHILE, ecc.) per eseguire dei test (IF .. ELSE .. ENDIF, SELECT CASE .. END SELECT, ecc.), le sue istruzioni di interazione, le sue istruzioni di Input/Output di file, ecc.

E' chiaro che le due componenti del Visual Basic, la parte Basic e la parte Object, vanno integrate. Vediamo un esempio classico di tale integrazione in figura 7: con tre OptionBox e due CheckBox si scelgono le modalità di visualizzazione di un'Etichetta nella sua Cornice.

Ci sono due aspetti interessanti da osservare: quello della routine (il suo nome è Sistema) che sistema l'Etichetta nella Cornice e quello dell'individuazione dell'Evento che deve richiamare la routine. Gli eventi sono tanti, sono tutti quelli che modificano la situazione delle Option o delle Check, sono cinque e richiamano tutti la stessa Routine Sistema.

In figure 8 e 9 vediamo una serie di istruzioni di Interazione, che, grosso modo, sostituiscono la vecchia istruzione INPUT del Basic classico. Sono tre:

```
Msgbox(messaggio)
A = Msgbox(..)
B = InputBox(..)
```

Queste tre istruzioni sono le stesse in tutti e tre i prodotti di cui stiamo parlando. In Excel italiano diventano:

```
FinestraMessaggio(..)
A = FinestraMessaggio(..)
B = FinestraInput(..)
```

Chiariamo la differenza tra i primi due comandi. La prima è un' Istruzione che presenta un messaggio secco, che blocca l'esecuzione del programma che può essere ripresa cliccando sul Pulsante OK presente nella Box. La seconda è una Funzione che presenta un Messaggio di tipo Richiesta che propone due o tre pulsanti di uscita, esempio:

```
Stampa - Sì, No
Visualizza su Video - Sì, No,
Annulla
```

La variabile A assume un valore numerico corrispondente al particolare pulsante premuto.

Il terzo comando, siamo arrivati alla InputBox, serve per digitare una stringa che viene assegnata, dal comando stesso, alla variabile B.

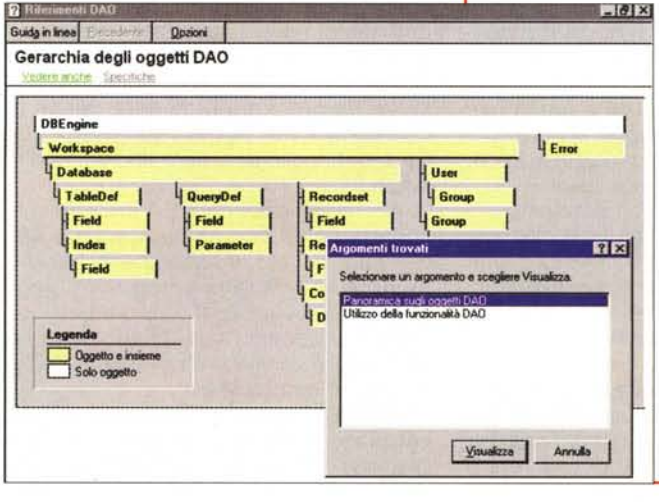

Figura 10 - MS Excel 7.0 e MS Access 7.0 - Il Modello Object Based applicato all'accesso ai Dati (DAO).
Il Modello Object Based è stato all'inizio (parliamo delle primissime versioni del Visual Basic) utilizzato solo per maneggiare i semplici oggetti standard di Windows. Poi, data la sua indubbia efficacia, è stato esportato in altri ambiti. L'esempio più importante di tale generalizzazione è costituito da DAO, la tecnologia con la quale anche un Database e tutti i suoi componenti possono essere visti e trattati come Oggetti. Importantissimo è il fatto che la tecnologia DAO (che significa Data Access Object) può essere sfruttata, nello stesso identico modo, da tutti e tre i prodotti di cui stiamo parlando.

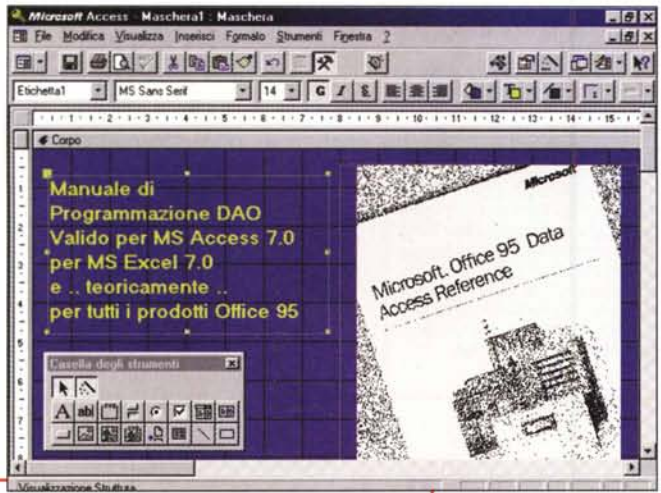

Figura 11 - Il Manuale di programmazione DAO. Chi si interessa di programmazione Object Based, con qualsiasi prodotto stia lavorando, dovrebbe procurarsi questo manuale della Microsoft. E' "bivalente" nel senso che vale sia per MS Access che per MS Excel. Integra gli altri manuali dei due prodotti che sono, come purtroppo noto, abbastanza carenti su questi argomenti un po' più evoluti. I manuali disponibili con Visual Basic sono invece approfonditi anche nella trattazione degli argomenti DAO. Va ribadito il fatto che comunque le istruzioni per programmare gli oggetti DAO sono le stesse in tutti e tre i prodotti.

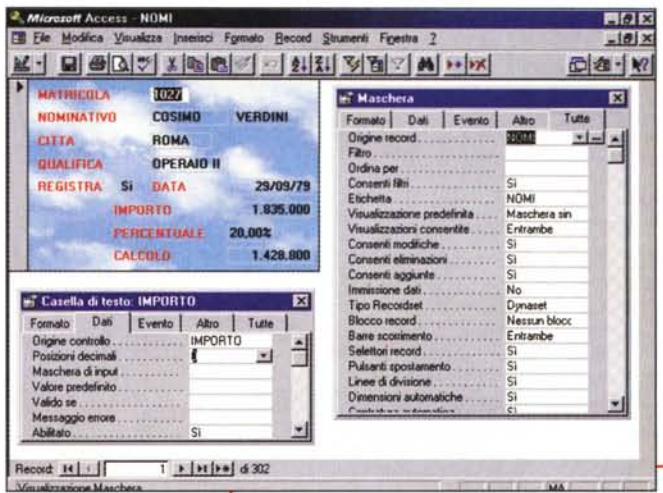

Figura 12 - MS Access 7.0 - Collegamento diretto degli Oggetti ai Dati. Il primo livello di utilizzo di un Database in un'applicazione Access è costituito dal collegamento diretto tra gli oggetti, Caselle di Testo, Combo Box, ecc. e la struttura delle Tabelle o delle Query. In pratica occorre fare due cose, collegare la Form ad un'Origine dei Record (una Tabella o una Query, appunto) e gli oggetti elementari ad un Campo. Inoltre, per i vari oggetti, nascono una serie di Eventi dipendenti da operazioni sui Dati. Il Modello Object Based trova in questo utilizzo un'ulteriore conferma.

In pratica si tratta di piccolissime DialogBox nelle quali è presente una sola TextBox.

Il Modello Object Based applicato al Database

Il modello concettuale, basato su oggetti, proprietà, eventi e metodi, può essere applicato anche ad altre entità gestibili in un'applicazione sviluppata in Windows. Una grande e decisiva intuizione, da parte della Microsoft, è stata quella di associare il modello Object Based al Database, che può essere quindi scomposto in oggetti e manipolato per mezzo degli eventi legati ai dati e intervenendo sulle loro proprietà.

L'introduzione di questa tecnologia, che si chiama DAO, presenta una serie di vantaggi. Se si padroneggia la filosofia del Modello Object Based diventa più facile anche maneggiare i Database. Le istruzioni di manipolazione degli Oggetti (i famosi Metodi) che costituiscono il Database sono le stesse in tutti i prodotti Obejct Based della Microsoft.

In figura 10 vediamo l'organizzazione gerarchica degli oggetti che costituiscono un Database, importantissima per capire quale oggetto può essere manipolato, e come, in un Database.

In figura 11 invece vediamo la foto della copertina del Manuale di Programmazione DAO, in dotazione al prodotto MS Access Developer's ToolKit, primo esempio di manuale "interprodotto". E' in comune tra Access ed Excel (in pratica gli esempi in esso riportati fanno riferimento o ad Access o ad Excel).

In concreto, quando si ha a che fare con i dati

Di DAO abbiamo parlato moltissime volte in numerosi articoli di questa rubrica. Dato il taglio introduttivo di questo articolo preferiamo ribadire solamente le semplici regole da rispettare quando si voglia accedere ai dati di un Database, supposto in formato MDB, rispettivamente da Access, da Visual Basic e da Excel.

Con Access per fare in modo che su una Maschera (una Scheda, se la versione è la 2.0) appaiano, in una serie di Caselle di Testo, i campi di una Tabella occorre che siano impostate una serie di proprietà della Maschera e delle varie Caselle di Testo. Per la Maschera la proprietà principale è la Origine dei Record che indica quale è la Tabella, o la Query, che rifornisce di dati la Maschera stessa.

A questo punto si è collegata la Maschera ai Dati e conseguentemente quando si inseriscono oggetti, ad esempio le Caselle di Testo, vengono automaticamente proposti, per la proprietà Origine del Controllo, i campi messi a disposizione della Tabella (lo vediamo concretamente nel fotomontaggio di figura 12).

Un'altra proprietà della Maschera da tenere sott'occhio è la Mostra Pulsanti di Trasferimento. E' quella che visualizza, sul bordo inferiore sinistro della Maschera, i quattro pulsanti VCR-like che servono per muoversi tra i record.

Tutte queste impostazioni citate vengono generate automaticamente quando si fa uso dell'Autocomposizione della Maschera, per cui non ci se ne accorge. Se si crea a mano la Maschera occorre provvedere personalmente.

Altra possibilità è quella di rinunciare a queste impostazioni e di costruire proprie routine di visualizzazione dei Dati e di movimento tra i Record, sfruttando la programmazione DAO (della quale vedremo tra un po' l'esempio in Excel).

In Visual Basic (l'esempio è in figura 13) il delicato compito di collegare i Dati, ad esempio i dati di una Tabella di un Database, con l'applicazione è svolto dal DataControl, le cui proprietà principali sono quattro:

- Connect, ovvero il tipo di database cui ci si collega (se si omette viene assunto come tipo di default il Database in formato MDB, e quindi Access),

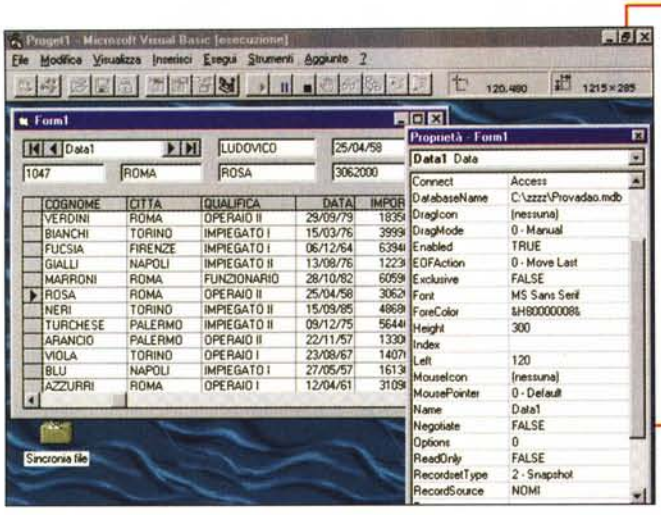

Figura 13 - MS Visual Basic 4.0 - Collegamento diretto degli Oggetti ai Dati.
Con Visual Basic si può ottenere lo stesso risultato visto or ora con Access. La differenza sta nel fatto che occorre introdurre un particolare Oggetto che fa da intermediario tra Database e Oggetti: il Data Control. In pratica mentre con Access l'Origine dei Record è una proprietà della Form, con Visual Basic l'Origine dei Record è una proprietà del Data Control. Al Data Control poi possono essere collegati tanti tipi di Oggetti che visualizzano, in varia forma, i Dati.

- DatabaseName, il nome del Database al quale si sta accedendo (in caso di file MDB il nome del file, in caso di file DBF, il nome della Cartella in cui il file è memorizzato),
- RecordsetType, va indicato se si tratta di una Tabella, di una Query i cui dati debbano essere aggiornabili, di una Query i cui dati non debbano essere aggiornabili,
- RecordSet, la Tabella o la Query, espressa in SQL.

A questo punto sono tantissimi i tipi di Oggetto associabili al DataControl. Se si usa il più semplice tra questi, ovvero la Casella di Testo, occorre impostare solo due proprietà:

- DataSource, e quindi il nome del DataControl,
- DataField, e quindi in nome del campo da visualizzare nella Casella.

Se si usa il più complicato, che è il DBGrid, occorre impostare una sola proprietà per indicare che la Griglia mostra i dati forniti dal DataControl.

Con queste semplici impostazioni si genera il collegamento tra applicazione e dati e questi si cominciano a vedere nelle caselle di Testo e nella Griglia (figura 13).

Come ultimo esercizio (per favore osservare la figura 14) vediamo come, sfruttando gli oggetti DAO, si possa da Excel accedere facilissimamente ad una Tabella di un Database MDB. Proponiamo due alternative:

- la prima è quella che comporta il semplice riversamento di una Tabella MDB in un foglio Excel sfruttando il formidabile comando CopyRecordSet,
- la seconda invece consiste in una procedura che legge uno per uno tutti i dati della Tabella e li visualizza sempre nelle stesse celle. Questo secondo esempio serve solo per studiare la routine generica di scorrimento. Dipenderà dal programma che si sta realizzando cosa fare dei dati che via via vengono letti.

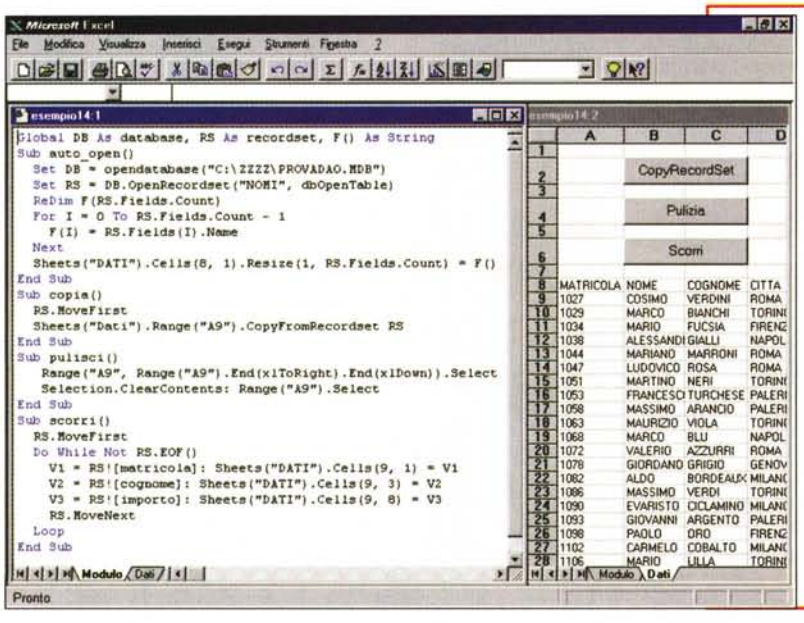

Figura 14 - MS Excel 7.0 - Esempio di programmazione DAO.
In pratica si definisce l'Oggetto RecordSet (nel nostro caso si chiama RS) che è un insieme di Dati provenienti da un Database. Un RecordSet può essere una Tabella "reale", fatta di righe e di colonne (Record e Campi), o una Tabella "virtuale", originata dinamicamente da un'interrogazione SQL. Su questo "oggetto", caratterizzato da una serie di proprietà e composto di numerosi "sottooggetti" (ad esempio i Nomi dei Campi) si può agire con tanti "metodi". C'è il CopyRecordSet, che scarica l'intero set di Dati a partire da una data cella del Foglio, oppure i vari comandi di movimento (es. MoveNext) e di lettura del singolo Campo di un singolo Record.

Conclusioni

All'inizio dell'articolo avevamo detto che avremmo accennato anche alla tecnologia OLE Automation, quella che consente di utilizzare come Oggetti in una propria Applicazione altre applicazioni o parti di esse (OLE Server e Mini OLE Server).

Anche in questo caso si tratta di una tecnologia facile da spiegare dal punto di vista teorico ma abbastanza complessa da sfruttare in tutte le sue sfaccettature e spesso critica nel suo utilizzo. Ne parleremo nel prossimo articolo. In quella sede faremo anche una serie di considerazioni finali sull'argomento Programmazione Object Based.

Intanto suggeriamo a chi voglia incominciare a studiare la "materia" Programmazione Object Based di utilizzare dapprima il Visual Basic, nel quale gli aspetti teorici trovano immediata traduzione in operazioni pratiche. Invece con Access e soprattutto con Excel si crea un po' di confusione quando occorra riferirsi ad un oggetto tipico del prodotto. In particolare con Excel, in cui gli oggetti sono fortemente gerarchicizzati (Applicazione, Cartella di Lavoro, Foglio di Lavoro, Intervallo, Cella, ecc.), l'individuazione esatta dell'Oggetto risulta in certi casi estremamente macchinosa e poco intuitiva soprattutto per chi è abituato alla facilità operativa delle normali operazioni sul foglio, sul quale sono anche disponibili potenti strumenti di aiuto assenti nell'ambiente programmazione. Au revoir. MC

Hobbytronica, emozioni di un altro mondo.

Volate al primo salone della multimedialità per la famiglia.

Viaggi nello spazio cibernetico, esplorazioni nel mondo dell'informatica, spedizioni sul pianeta della comunicazione telematica. Finalmente un salone dove vedere, provare e scegliere le nuove tecnologie multimediali: dalla realtà virtuale ai video game, dalle telecomunicazioni satellitari ai CD-ROM interattivi.

A Hobbytronica il futuro è già presente.

TORINO-LINGOTTO FIERE
20-24 NOVEMBRE 1996
DALLE 10 ALLE 23

Lingotto Fiere

Organizzazione: Expo 2000 S.p.A.
Tel. 011/664.4111 - Fax 011/664.6642
E Mail: dev @ lingottofiere.it
http://W W W.lingottofiere.it
Lingotto Fiere