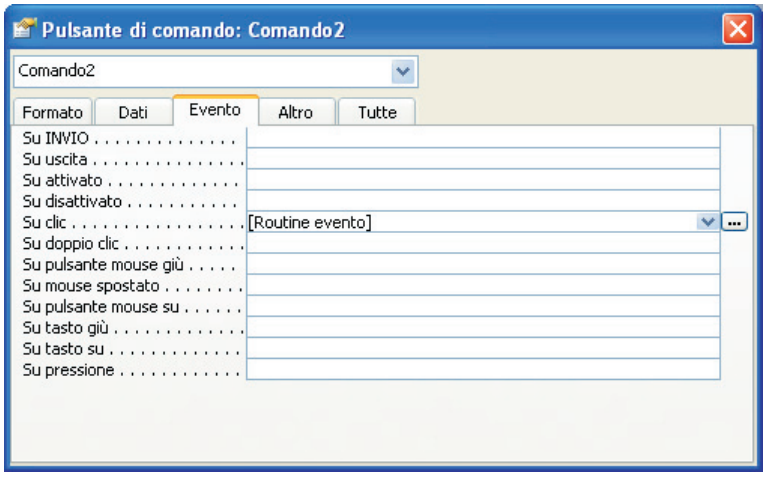
3. Le routine evento in Access 2000/2003

Le routine evento possono essere associate a un singolo controllo grafico (caselle di testo, pulsanti di comando, ecc.) presente all'interno di una maschera o di un report oppure agli eventi della maschera o del report medesimi.



Una routine evento viene associata a un controllo o a una maschera o a un report scegliendo, nella finestra **Proprietà** dell'oggetto, **[Routine evento]** dall'elenco che si apre facendo clic sulla freccia nera verso il basso, sulla riga corrispondente all'evento**.** Occorre poi fare clic sui tre puntini a destra per attivare l'ambiente di programmazione Visual Basic.

Il codice di una routine evento è organizzato come sottoprogramma, detto **Sub**, secondo questo schema generale:

```
Public Sub NomeS(parametri)
. . . .
End Sub
```

Tra *Sub* e *End Sub* vengono scritte le istruzioni del sottoprogramma.

PROGETTO

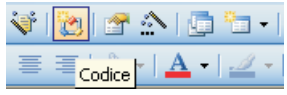
Con riferimento al database per la gestione dei prodotti di un magazzino, aggiungere nella maschera di gestione dei prodotti la funzionalità che consente di creare un nuovo fornitore durante l'inserimento di un nuovo prodotto.

Apriamo la maschera *Prodotti* in *Visualizzazione Struttura* e apriamo la finestra **Proprietà** del pulsante di comando *NuovoFornitore* (tasto *F4*, oppure tasto destro del mouse e scelta *Proprietà*).
Nella scheda **Evento** selezioniamo l'evento **Su clic** e, nell'elenco che si apre facendo clic sulla freccia nera verso il basso, sostituiamo il nome della macro con **[Routine evento]**.

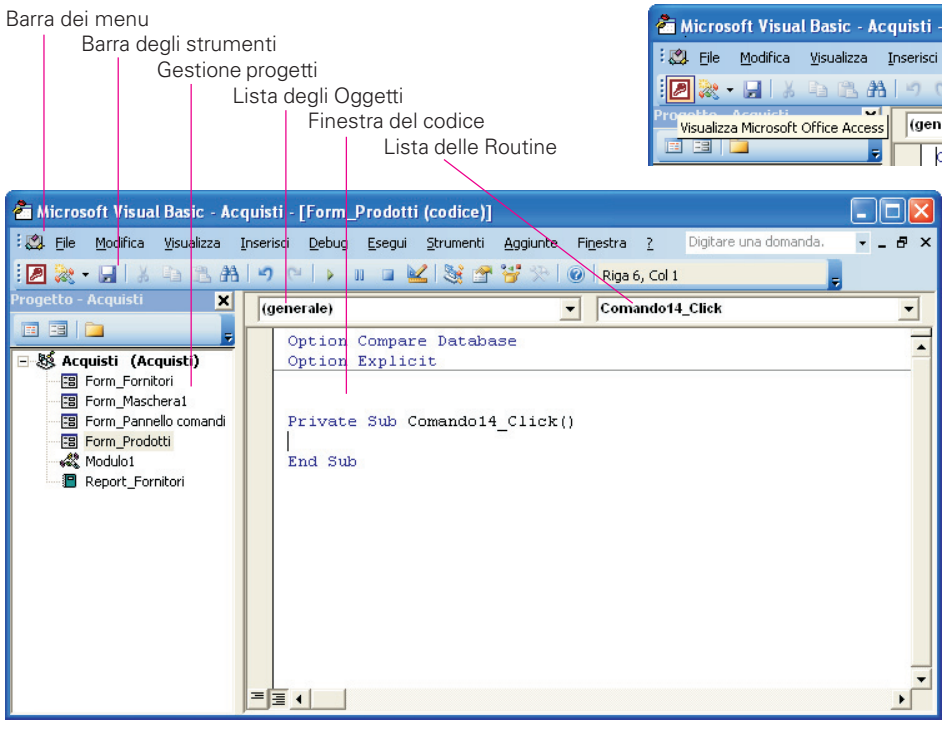
Facendo poi clic sui tre puntini a destra, si apre la finestra del codice **Visual Basic**.
Si può passare velocemente dalla finestra Visual Basic agli oggetti di Access e viceversa attraverso la combinazione di tasti **ALT + F11**.

In alternativa si possono usare le icone della barra degli strumenti:

- l'icona *Codice* per passare da Access a Visual Basic:



- l'icona *Microsoft Access* per passare da Visual Basic ad Access:



Il riquadro a destra rappresenta la finestra di **Visual Basic Editor**, dove si possono scrivere o modificare le istruzioni del codice.

Nella finestra vengono create automaticamente le seguenti righe di codice:

```
Option Compare Database
Option Explicit
```

```
Private Sub Comando14_Click()

End Sub
```

L'istruzione **Option Compare** imposta il modo con cui le stringhe vengono confrontate tra loro. Per il codice Visual Basic scritto in Access, si può usare l'opzione **Database** per indicare che il confronto tra stringhe deve essere fatto secondo l'ordinamento determinato dall'impostazione dell'identificativo del Paese nel database di Access.

La seconda riga **Option Explicit** è l'istruzione che imposta la dichiarazione obbligatoria per tutte le variabili utilizzate nel codice. Questo consente di evitare problemi causati da nomi di identificatori digitati in modo errato.
Sub e **End Sub** indicano l'inizio e la fine del sottoprogramma (*subprogram*). L'intestazione del *Sub* è preceduta dalla parola **Private**, per indicare una routine locale della maschera, cioè un codice che deve essere visibile soltanto all'interno della maschera. In caso contrario, si deve usare la dichiarazione **Public**.
Anche il nome del sottoprogramma viene impostato automaticamente come:

```
Comando14_Click
```

dove *Comando14* è il nome assegnato al pulsante di comando (il nome può essere cambiato impostando la proprietà **Nome elemento**) e *Click* è l'evento provocato dall'utente quando fa un clic con il mouse sopra il pulsante di comando.
In altre parole, il codice tra *Sub* e *End Sub* specifica le istruzioni da eseguire quando accade l'evento clic sul pulsante di comando.
Il nome del controllo e il nome dell'evento sono separati dal carattere _.
Le parentesi tonde dopo il nome del *Sub* servono a contenere eventuali parametri passati al sottoprogramma.
Scriviamo quindi l'istruzione per aprire la maschera *Fornitori* nella modalità che consente l'aggiunta di un nuovo record:

```
DoCmd.OpenForm "Fornitori", , , , acFormAdd
```

DoCmd (*Do Command*) indica l'esecuzione di un comando, **OpenForm** specifica il comando da eseguire (*Form* è la traduzione di *Maschera*), "*Fornitori*" è il nome della maschera da aprire, **acFormAdd** è la modalità di apertura della maschera (aggiunta di un nuovo record). Le virgole indicano parametri non specificati che sono opzionali.
Più precisamente, *DoCmd* è l'**oggetto** del linguaggio Visual Basic che esegue le azioni di Access e *OpenForm* è un **metodo** applicato all'oggetto *DoCmd*. Oggetto e metodo sono separati dal punto.
In alternativa si possono scrivere le seguenti righe di codice:

```
DoCmd.OpenForm "Fornitori"
DoCmd.GoToRecord , , acNewRec
```

La prima istruzione apre la maschera *Fornitori*, la seconda effettua il posizionamento su un nuovo record per l'inserimento dei dati.

Vediamo ora come si possa tradurre in Visual Basic l'azione *SpostaRidimensiona* per una maschera.
Il comando per spostare e ridimensionare una maschera è **MoveSize** che ha 4 argomenti: a destra, giù, larghezza e altezza. Tutti gli argomenti sono espressi in **twip**, un'unità di misura equivalente a 1/1440 di pollice oppure a 1/567 centimetri (un centimetro contiene 567 twip).
I parametri utilizzati nell'esempio del paragrafo 1 (*A destra*: 2 cm, *Giù*: 6 cm, *Larghezza*: 15 cm, *Altezza*: 10 cm) possono essere rappresentati nel codice Visual Basic nel seguente modo:

```
Dim T As Integer
T = 567
DoCmd.MoveSize 2 * T, 6 * T, 15 * T, 10 * T
```

Il valore di *T* viene usato come moltiplicatore per i valori espressi in cm. Il carattere * indica l'operazione di moltiplicazione.
La modalità usata non è ovviamente obbligatoria (si potrebbe scrivere direttamente 1134 anziché 2 * T), ma mostra un esempio delle modalità per dichiarare le variabili e per assegnare ad esse un valore. Inoltre fornisce maggiore chiarezza al codice.
L'istruzione **Dim** dichiara il nome *T* e il tipo di variabile (il tipo **Integer**, cioè numero intero, è scritto dopo la parola **As**).
L'assegnazione è realizzata dall'**operatore =** (la variabile a sinistra del segno =, il valore assegnato a destra).
La routine completa è la seguente:

```
Private Sub Comando14_Click()
Dim Twip As Integer
Twip = 567
DoCmd.OpenForm "Fornitori", , , , acFormAdd
DoCmd.MoveSize 2 * Twip, 6 * Twip, 15 * Twip, 10 * Twip
End Sub
```

Completiamo la traduzione delle azioni aggiungendo alla maschera *Fornitori* la routine evento che gestisce il clic sul pulsante di comando *ChiudiFornitore*

```
Private Sub Comando16_Click()
DoCmd.Close
Forms!Prodotti!IDFornitore.Requery
End Sub
```

La prima istruzione esegue il metodo **Close** sulla maschera; la seconda aggiorna i dati sui fornitori nell'elenco della casella combinata di *IDFornitore* nella maschera *Prodotti*.
La sintassi:

```
Forms!Prodotti!IDFornitore.Requery
```

significa che l'elenco del controllo *IDFornitore* della maschera *Prodotti*, che si trova nell'insieme **Forms** delle maschere nel database, viene aggiornato attraverso il metodo **Requery**.
I punti esclamativi sono elementi di concatenazione tra gli identificativi degli oggetti.
Se il nome della maschera o del controllo include uno spazio, il nome deve essere racchiuso tra una coppia di parentesi quadre [ ].

Quando l'operazione svolta all'interno di una maschera riguarda gli oggetti della maschera stessa, il nome della maschera viene sostituito con **Me** (traduzione inglese di *me stessa*). Per esempio il seguente sottoprogramma, associato al pulsante di comando *Comando15* nella maschera *Prodotti*, aggiorna il prezzo del prodotto aumentandolo del 10%:

```
Private Sub Comando15_Click()
Me!PrezzoUnitario = PrezzoUnitario * 1.1
End Sub
```

Il nome *Me* identifica la maschera stessa, *Prodotti*, contenente la routine.
Per l'apertura di un report un anteprima di stampa si può usare il metodo **OpenReport**, con modalità e sintassi analoghe a quelle viste per il metodo *OpenForm* nell'esempio precedente.