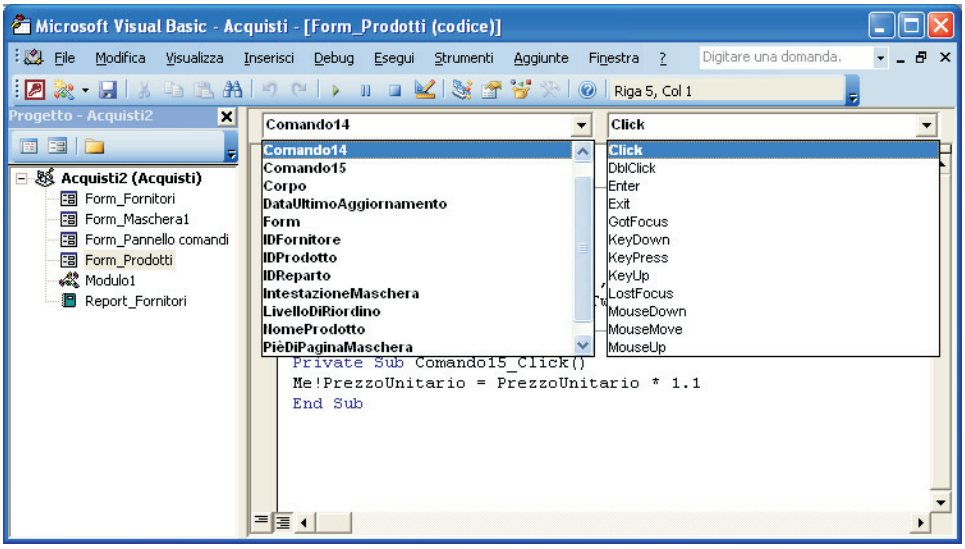
4. I moduli in Access 2000/2003

Il **modulo** è uno degli oggetti del database di Access e rappresenta un insieme di dichiarazioni e routine scritte con il linguaggio **Visual Basic**, memorizzate come una singola unità.
I **moduli di maschera e di report** sono associati a una maschera, a un report oppure a un controllo grafico contenuto nella maschera o nel report: in genere questi moduli contengono routine di gestione di eventi**.**
Quando si crea la prima routine evento per una maschera o per un report, viene automaticamente creato il modulo di maschera o di report associato.

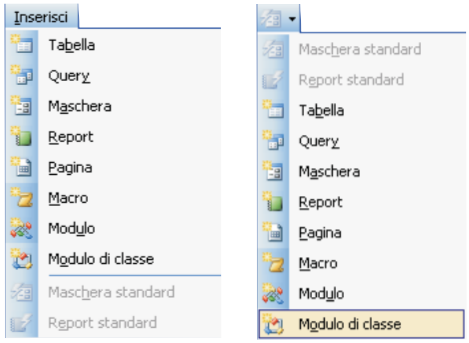


Quando una maschera è aperta in *Visualizzazione Struttura*, nella finestra del codice Visual Basic la casella in alto a sinistra (**Oggetto**) riporta l'elenco dei controlli e delle sezioni della maschera, mentre la casella in alto a destra (**Routine**) contiene l'elenco degli eventi che possono essere associati agli oggetti della maschera.
I nomi degli eventi, ai quali sono già associate routine evento, sono visualizzati in grassetto.

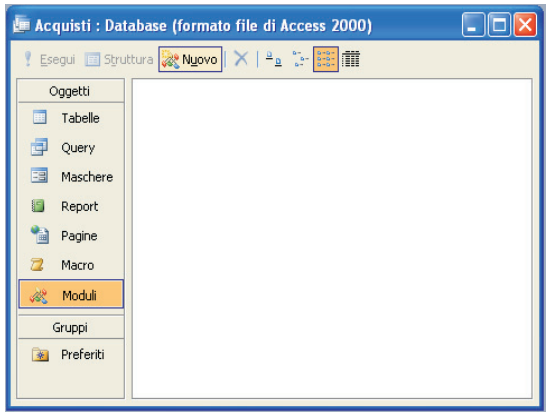
Considerazioni analoghe valgono anche per i report.

Nella finestra del codice, in basso a sinistra, ci sono due icone che attivano due diverse modalità di visualizzazione dei moduli:
- icona **Visualizza routine**
- icona **Visualizza modulo intero**.

I **moduli di classe** contengono la definizione di nuovi oggetti personalizzati (secondo la terminologia della programmazione a oggetti, l'*oggetto* è l'istanza di una *classe*). Le routine definite nel modulo di classe diventano le proprietà e i metodi dell'oggetto.
Per creare un modulo di classe occorre scegliere **Modulo di classe** nel menu **Inserisci** oppure nell'elenco che si apre facendo clic sulla freccia accanto all'icona **Nuovo oggetto** nella *Barra degli strumenti.*



I **moduli standard**, invece, sono definiti nel database in modo indipendente dalle maschere e dai report e consentono di creare codice utilizzabile in altri moduli o in altre routine evento. Per creare un modulo standard basta fare clic su **Moduli** e poi su **Nuovo** nella finestra degli oggetti di Access: anche in questo caso si apre automaticamente l'ambiente di programmazione Visual Basic.



Il codice di un modulo è organizzato come sottoprogramma, detto **Sub**, oppure come funzione, detta **Function**.
Gli schemi generali dei due tipi di codice sono:

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

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

```
Public Function NomeF(argomenti)As tipo
. . . .
End Function
```

Gli argomenti passati a una *Sub* o a una *Function* sono separati dalla virgola.
La differenza tra *Sub* e *Function* è la seguente: *Sub* è una routine che esegue un insieme di istruzioni, ma non restituisce alcun valore; la *Function* è una routine che restituisce un valore utilizzabile in un'espressione o nell'assegnazione di valore a una variabile. Il *tipo* del valore restituito è indicato nell'intestazione della *Function* dopo la parola **As**.
Proprio perché i moduli standard sono indipendenti dagli oggetti del database, è ragionevole pensare che essi contengano soprattutto *Function* pubbliche, che effettuano operazioni di calcolo o, in generale, elaborazioni utilizzabili da altri moduli o nelle routine evento.

PROGETTO

Scrivere il codice per controllare se una maschera è aperta.

Si consideri il caso della routine evento che consente di chiudere la maschera *Fornitori* e di aggiornare l'elenco di una casella combinata presente nell'altra maschera *Prodotti*. Perché l'aggiornamento si svolga correttamente, la maschera *Prodotti* deve essere aperta. In caso contrario si ha una situazione di errore.
Si deve quindi migliorare il codice aggiungendo la verifica sulla maschera *Prodotti*, in modo da eseguire l'istruzione di aggiornamento solo se la maschera *Prodotti* è aperta. Occorre anche controllare che la modalità di apertura non sia *Visualizzazione Struttura*.
Per risolvere il problema è stata usata come traccia la funzione *Caricata* presente nei moduli del database di esempio *Northwind* di Access.
La funzione riceve come parametro il nome di una maschera (*Maschera*) e restituisce il valore booleano **True** solo se la maschera è caricata e aperta in modalità diversa dalla *Visualizzazione Struttura*.
Creiamo quindi un nuovo modulo standard di Access e scriviamo nella finestra del codice Visual Basic la seguente funzione:

```
Function Caricata(ByVal Maschera As String) As Boolean
If SysCmd(acSysCmdGetObjectState,acForm,Maschera) <> 0 Then
  If Forms(Maschera).CurrentView <> 0 Then
    Caricata = True
  End If
End If
End Function
```

Il metodo **SysCmd** assume il valore 0 quando l'oggetto indicato come terzo argomento non è aperto o non esiste.
La proprietà **CurrentView** di una maschera assume il valore 0 se è aperta in *Visualizzazione Struttura*, secondo i valori presentati nella seguente tabella:

Valore di CurrentView	Modalità di visualizzazione
0	Visualizzazione Struttura
1	Visualizzazione Maschera
2	Visualizzazione Foglio dati

La funzione mostra anche un esempio di utilizzo della struttura di selezione in Visual Basic, secondo il seguente schema generale:

If *condizione* **Then**
  Istruzioni1
Else
  Istruzioni2
End If

Dopo aver salvato il modulo, la funzione *Caricata* può essere utilizzata nelle routine evento del database di Access.
Si osservi che l'uso del nome della maschera come argomento della funzione consente di applicare l'elaborazione della *Function* a qualsiasi maschera del database.

In particolare possiamo aggiungere un controllo all'interno della routine evento per verificare se la maschera *Prodotti* è aperta.

```
Private Sub Comando16_Click()
DoCmd.Close
If Caricata("Prodotti") Then
  Forms!Prodotti!IDFornitore.Requery
End If
End Sub
```

In questo modo la routine evento associata al clic sul pulsante *ChiudiFornitore* chiude comunque la maschera *Fornitori* e procede all'aggiornamento della casella combinata *IdFornitore* della maschera *Prodotti* solo se la maschera *Prodotti* è aperta.

La scrittura

```
If Caricata("Prodotti") Then . . .
```

è equivalente a

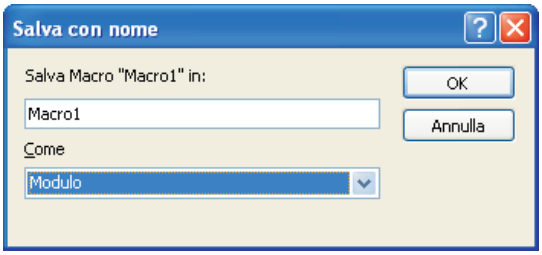
```
If Caricata("Prodotti") = True Then . . .
```

(valore restituito dalla funzione *Caricata* = True)

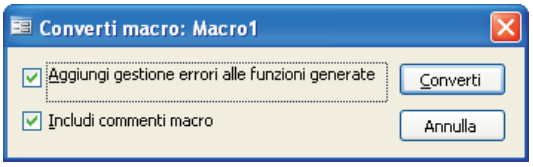
Per prendere confidenza con la sintassi del linguaggio può essere sicuramente utile provare a **convertire una macro in modulo**, sfruttando l'opzione presente in Access.

Occorre aprire una macro in *Visualizzazione Struttura* e dal menu **File** scegliere **Salva con nome**.

Nella finestra di dialogo che si apre, specificare un nome nella prima casella e scegliere **Modulo** nella casella inferiore (**Come**).



Nella finestra successiva si può chiedere di aggiungere i commenti e il controllo dell'errore nel codice generato dalla conversione.



Confermando con il pulsante **Converti**, negli oggetti di tipo *Modulo* del database si crea un nuovo modulo e nella finestra di Visual Basic si può leggere il codice generato.

Le righe in colore verde che iniziano con l'apice sono **righe di commento**, mentre l'intercettazione di eventuali situazioni di errore è rappresentata dalla struttura **On Error GoTo...**.