

DIDATTICA DELLE SCELTE ECONOMICHE E FINANZIARIE

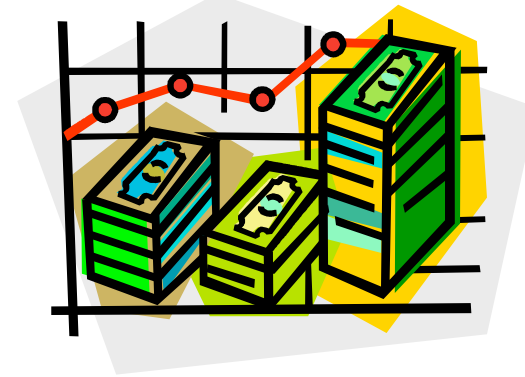
MODULO 4 Applicazioni finanziarie con VBA

S.S.I.S. Puglia
Classe 48/A - Indirizzo Fis.-Inf.-Mat.
Prof. Crescenzo Gallo
C.GALLO@UNIFG.IT

Sommario

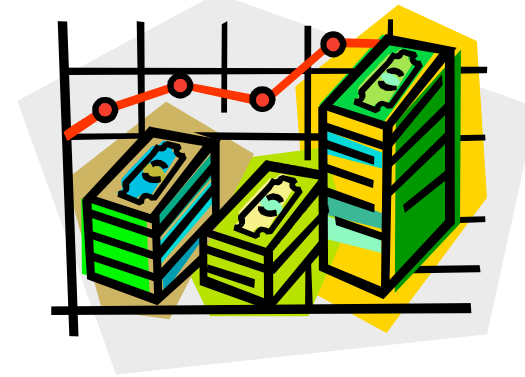
- 1. L'uso degli array nelle operazioni finanziarie**
- 2. Funzioni VBA per la determinazione delle rate di un'operazione finanziaria**
- 3. Piano di ammortamento di un mutuo**
- 4. Calcolo del tasso interno di rendimento**

1. L'uso degli array nelle operazioni finanziarie



Un'operazione finanziaria è definibile come un insieme di importi monetari che vengono pagati o riscossi (*flussi di cassa*) in certe date (*scadenze*).

1. L'uso degli array nelle operazioni finanziarie



È naturale definire un'operazione in cui si paga una somma in cambio di un flusso di somme future un'operazione di *investimento*, mentre definiamo operazione di *finanziamento* l'introito di una somma a fronte di esborsi futuri.

I flussi associati ad ogni operazione finanziaria sono destinati alla remunerazione ed alla restituzione del capitale impiegato o raccolto.

La struttura di base di ogni operazione finanziaria è composta da un insieme di scadenze:

$\{t_0 \ t_1 \ t_2 \ \dots \ t_N\}$ e da un insieme di flussi di cassa:

$\{c_0 \ c_1 \ c_2 \ \dots \ c_N\}$.



Lo scopo è quello di mettere in relazione il valore di somme di denaro ricevute in date diverse, e di poterne effettuare una valutazione aggregata rispetto ad un tempo di riferimento t .

Prima di procedere in questo senso, dobbiamo imparare a gestire insiemi di dati di questo tipo, detti ***array***, in VBA.



Un **array** è un insieme di variabili omogenee (come tipo di dati) identificate da un unico nome.

Un array ad una dimensione è anche detto *vettore*, un semplice elenco di elementi individuati da un indice, come ad es.:

Importi →

115,50	2500,00	350,45	735,55	1500,00
--------	---------	--------	--------	---------

In VBA tale vettore verrebbe dichiarato in questo modo:

Dim Importi(1 to 5) As Currency

in cui si intende che viene definito un “contenitore” di nome Importi composto di cinque elementi numerati da 1 a 5 e tutti di tipo “valuta”.

Il secondo importo, di 2500 Euro, dell'esempio è quindi identificato con **Importi(2)**.

Un array a due dimensioni è detto *matrice*, una tabella di elementi omogenei in cui ciascun elemento è individuato da una coppia di indici che specificano la riga e la colonna; ad es.:

Flussi →

115,50	2500,00	350,45	735,55	1500,00
264,30	8351,00	651,00	426,50	9350,25
815,50	7123,00	981,97	712,44	1098,00

in cui ogni riga individua un flusso di cassa per un'operazione diversa ed ogni colonna corrisponde ad un tempo.

In VBA tale matrice verrebbe dichiarata in questo modo:

Dim Flussi(1 to 3, 1 to 5) As Currency

in cui si intende che viene definito un “contenitore” di nome Flussi composto di cinque elementi numerati da 1 a 5 e tutti di tipo “valuta”.

Il secondo importo della terza riga, di 7123 Euro, dell'esempio è quindi identificato con **Flussi(3, 2)**.

Uno dei vantaggi principali dell'uso di array è la possibilità di utilizzare dei cicli per semplificare l'elaborazione dei diversi elementi che lo compongono.

Per esempio, se vogliamo inizializzare a zero tutti gli elementi del vettore **Importi** e del vettore **Flussi** dell'esempio precedente possiamo scrivere:

Dim i As Integer 'indice del ciclo

For i = 1 to 5 'ciclo su tutti gli elementi del vettore

Importi(i) = 0 'l'elemento viene azzerato

Next 'si passa all'elemento successivo

Dim riga As Integer, colonna as Integer 'indici dei cicli

For riga = 1 to 3 'ciclo sulle righe della matrice

For colonna = 1 to 5 'ciclo sulle colonne della matrice

Flussi(riga, colonna) = 0 'l'elemento viene azzerato

Next 'si passa alla colonna successiva

Next 'si passa alla riga successiva

Oppure possiamo costruire la tavola pitagorica sino al 10 nel seguente modo:

```
Dim Tavola(1 to 10, 1 to 10) As Integer
Dim riga As Integer, colonna As Integer
For riga = 1 to 10
    For colonna = 1 to 10
        Tavola(riga, colonna) = riga * colonna
    Next colonna
Next riga
```

Si noti che quando si utilizzano cicli **For** nidificati può essere utile (anche se non è obbligatorio) riportare l'indice del ciclo accanto all'istruzione **Next** che chiude il ciclo stesso.

Pulsanti di comando personalizzati e gestione degli eventi

Il concetto di “flusso di cassa” si presta ad illustrare l’uso degli array con una semplice applicazione, nella quale vedremo l’uso dei pulsanti di comando personalizzati che risultano molto utili in varie situazioni e rirenderemo il concetto di “evento”.

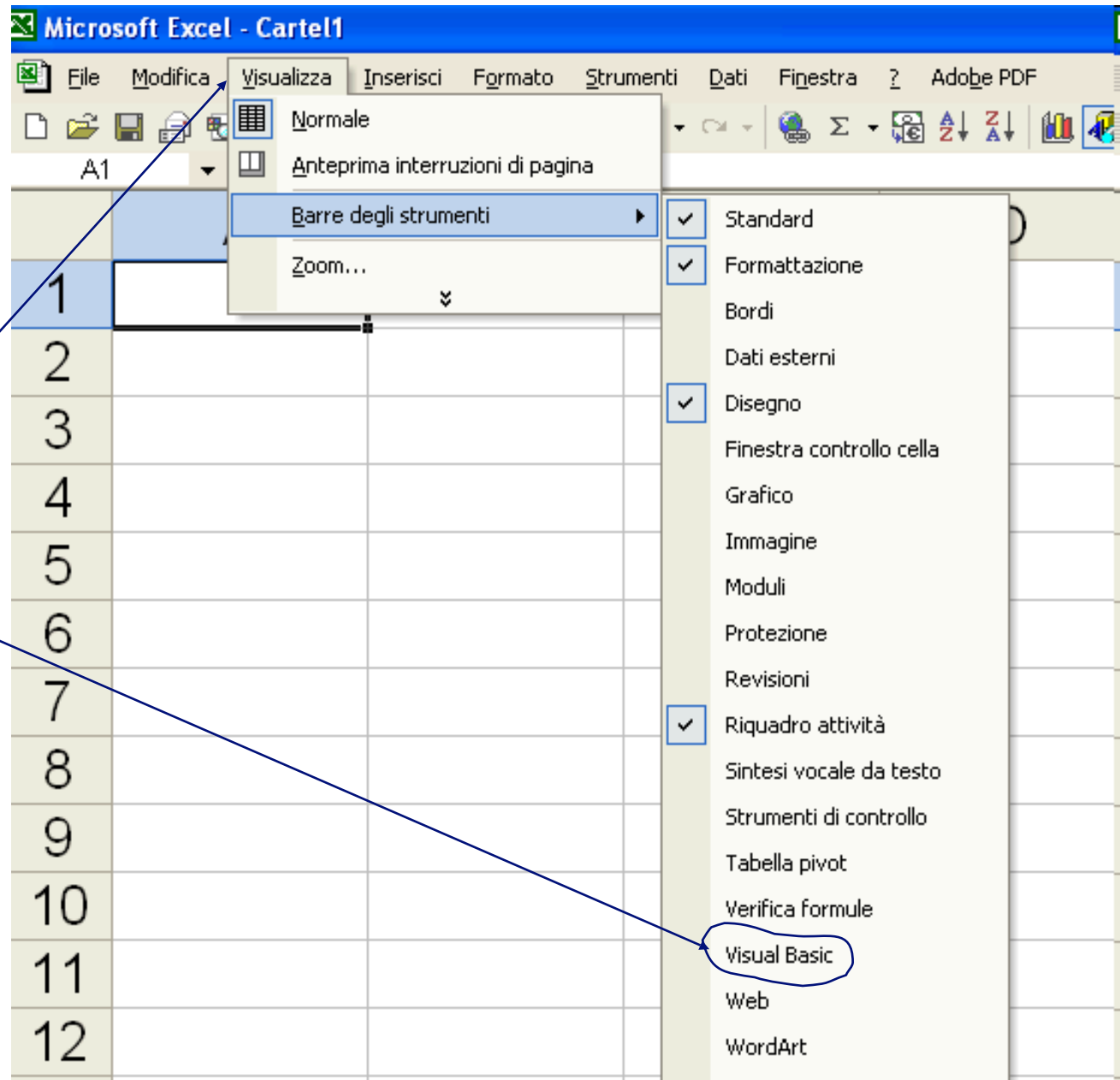
Pulsanti di comando personalizzati e gestione degli eventi

I pulsanti non solo possono attivare un evento quando vengono premuti, ma rispondono anche ad altri eventi come ad es. il passaggio del mouse sopra di essi.

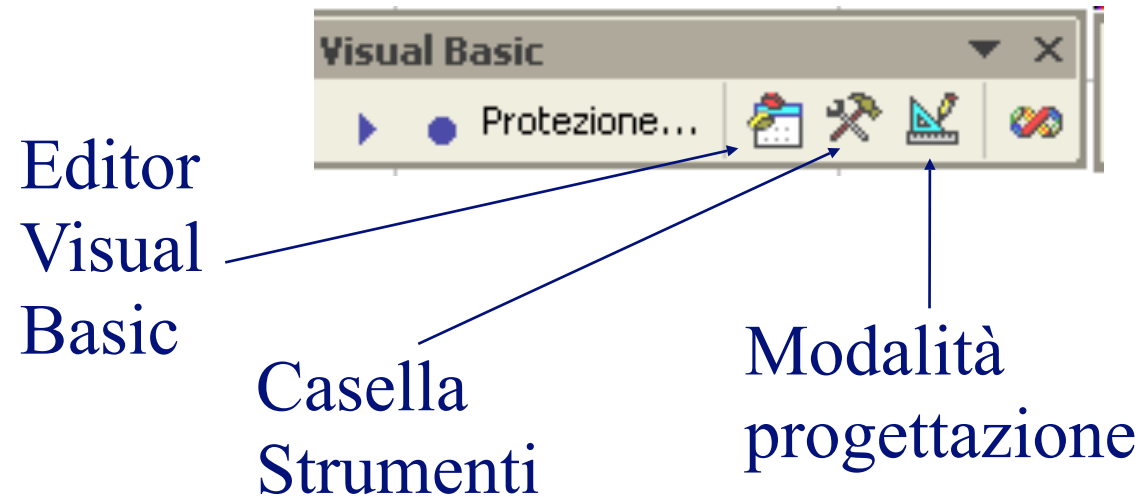
Diventa quindi importante poter *associare* una macro ad un pulsante: ciò è possibile mediante le **routine di gestione degli eventi**.

Le routine di gestione degli eventi sono speciali macro, collegate ad un oggetto (ad es. un pulsante di comando): vediamo un esempio.

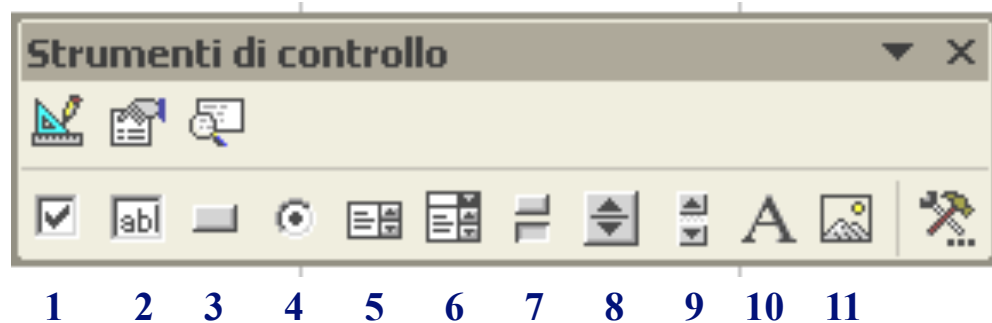
Apriamo Excel e attiviamo l'opzione *Visual Basic* dal menu *Visualizza \ Barre degli Strumenti*.



Premiamo ora il pulsante *Casella Strumenti*:



Si aprirà una finestra contenente gli “attrezzi di lavoro”, in questo caso i *controlli* (oggetti dell'interfaccia utente grafica) inseribili in un foglio:

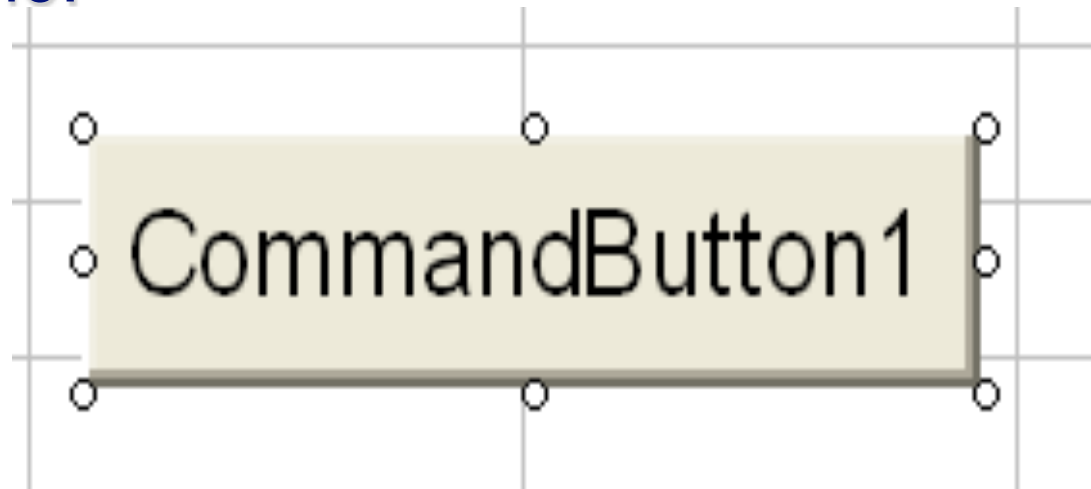


- 1) Casella di controllo
- 2) Casella di testo
- 3) Pulsante di comando
- 4) Pulsante di opzione
- 5) Casella di riepilogo
- 6) Casella combinata
- 7) Interruttore
- 8) Pulsante di selezione
- 9) Barra di scorrimento
- 10) Etichetta
- 11) Immagine

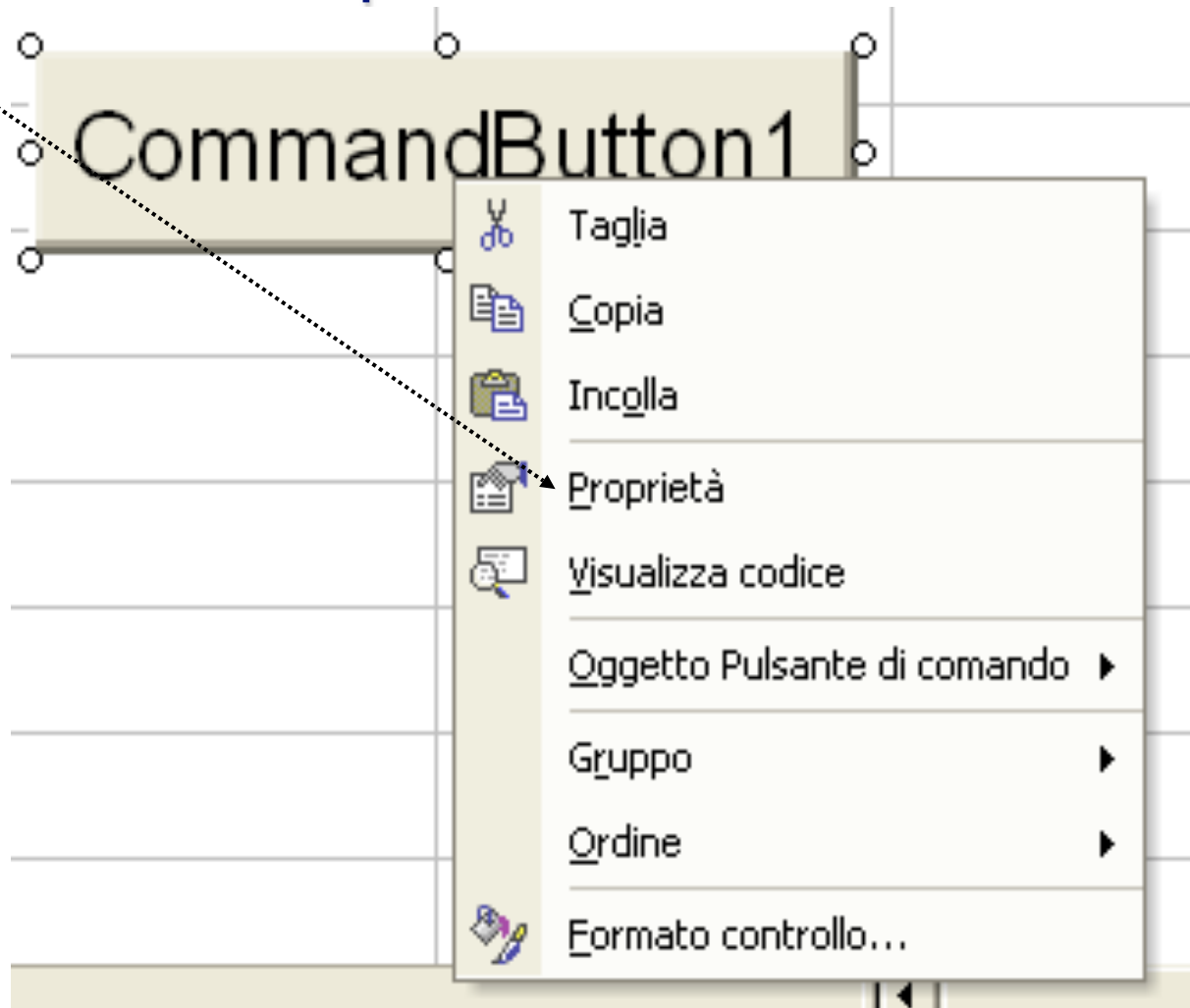
Opereremo con il controllo n.3 “pulsante di comando”; quando inseriremo il controllo, l’ambiente si troverà in “modalità progettazione”, nella quale i controlli non rispondono direttamente agli eventi ma consentono appunto di “progettare” un’applicazione in maniera visuale.

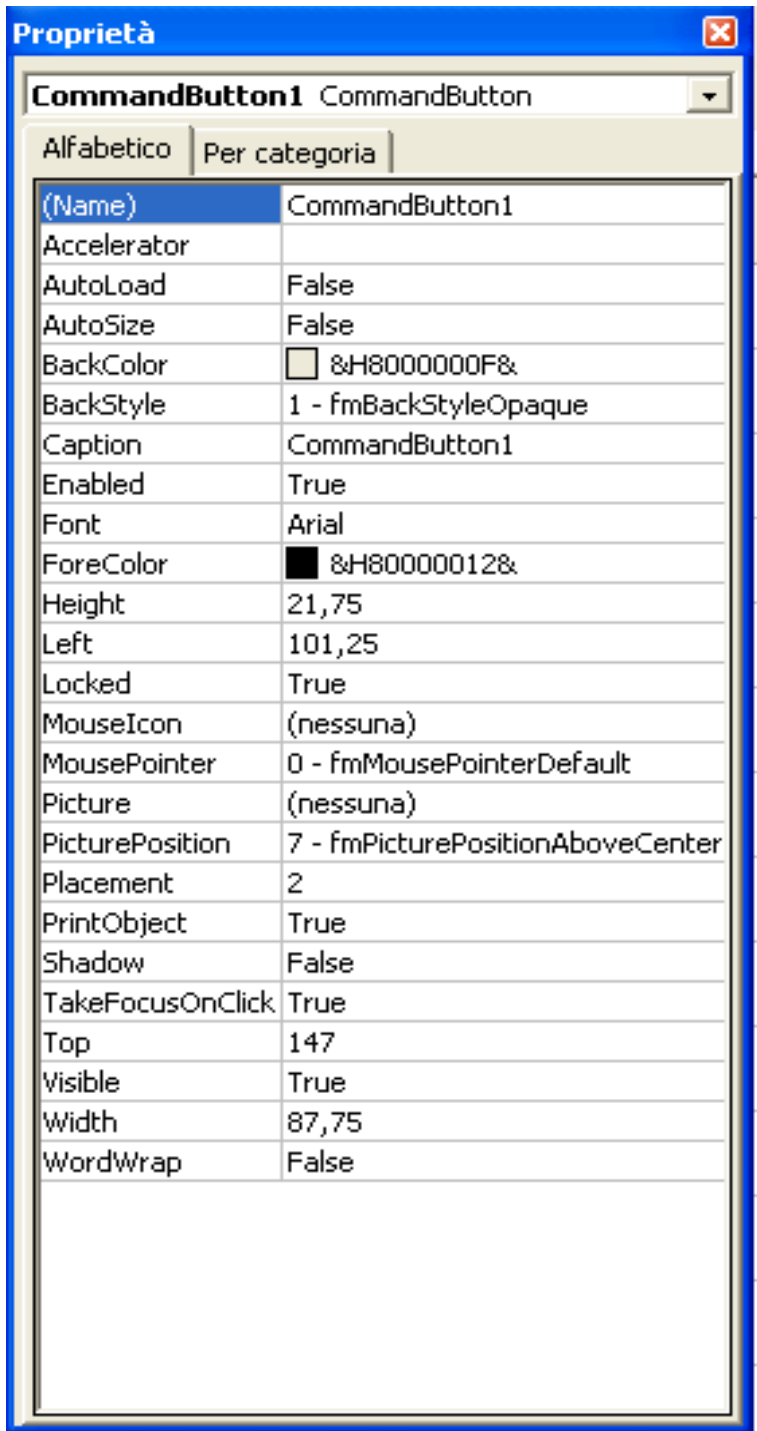


Premiamo quindi sul terzo controllo nella casella degli strumenti, portiamo il cursore del mouse sul foglio di lavoro e – tenendo premuto il tasto sinistro – trasciniamo un rettangolo sul foglio da un punto qualsiasi all'angolo opposto: apparirà un pulsante di comando, circondato da “punti” bianchi sui bordi che stanno ad indicare che il comando è attualmente selezionato in modalità progettazione:

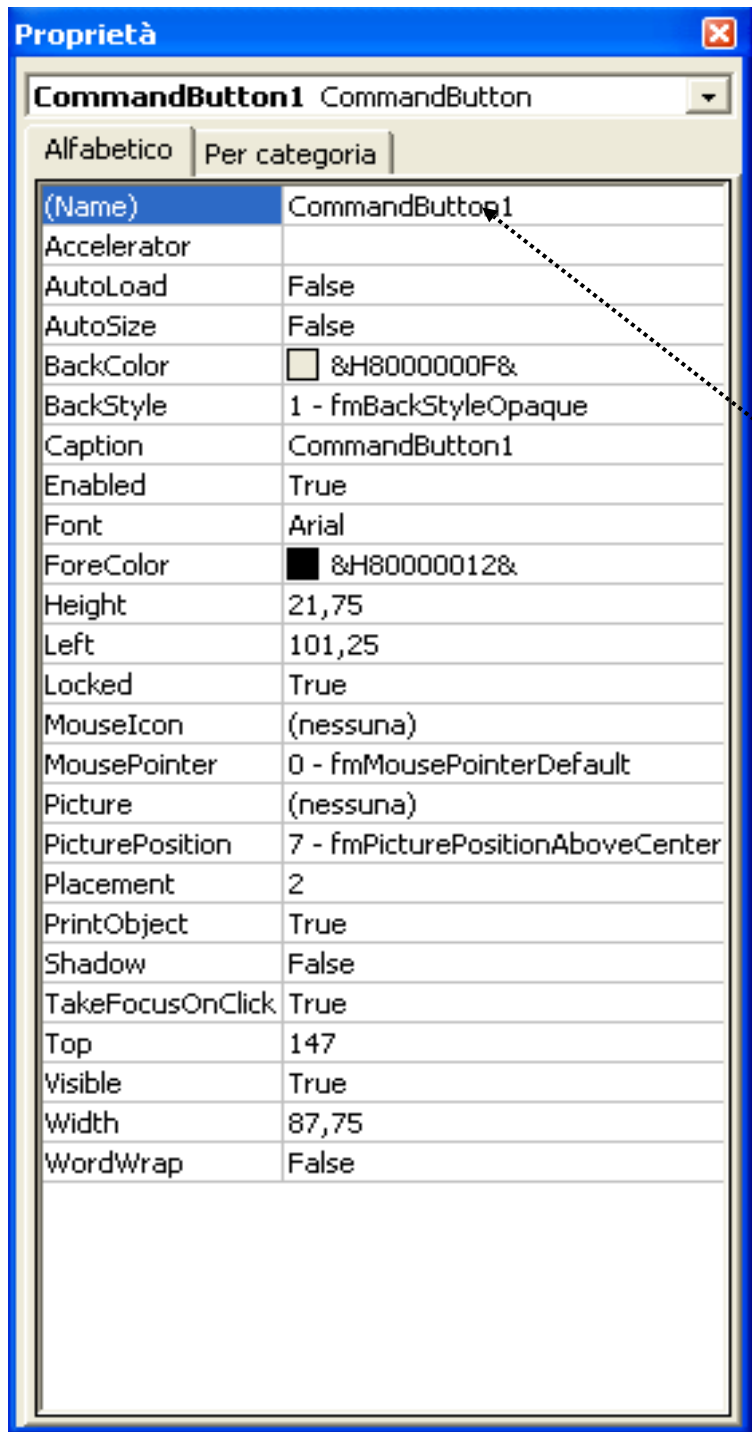


Premendo con il tasto destro del mouse sul controllo, si aprirà una finestra di menu: selezioniamo la voce *Proprietà* per far comparire la relativa finestra:

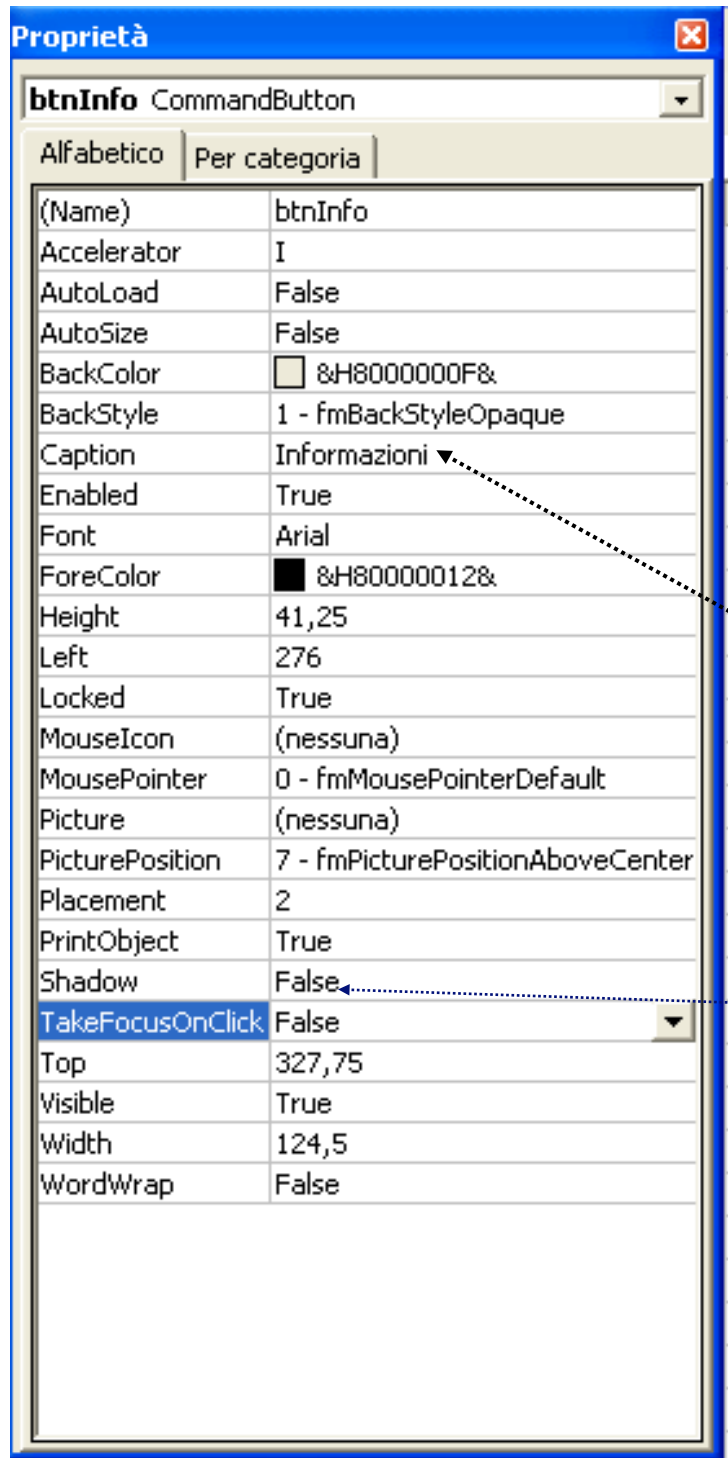




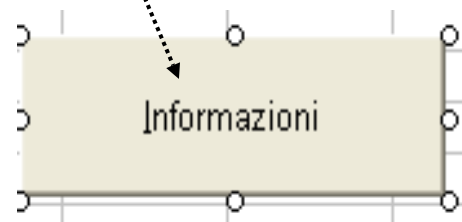
La casella superiore della finestra mostra quali proprietà dell'oggetto stanno per essere visualizzate; in questo caso l'oggetto è **CommandButton1** (che è un oggetto di tipo "CommandButton", cioè *pulsante di comando*).

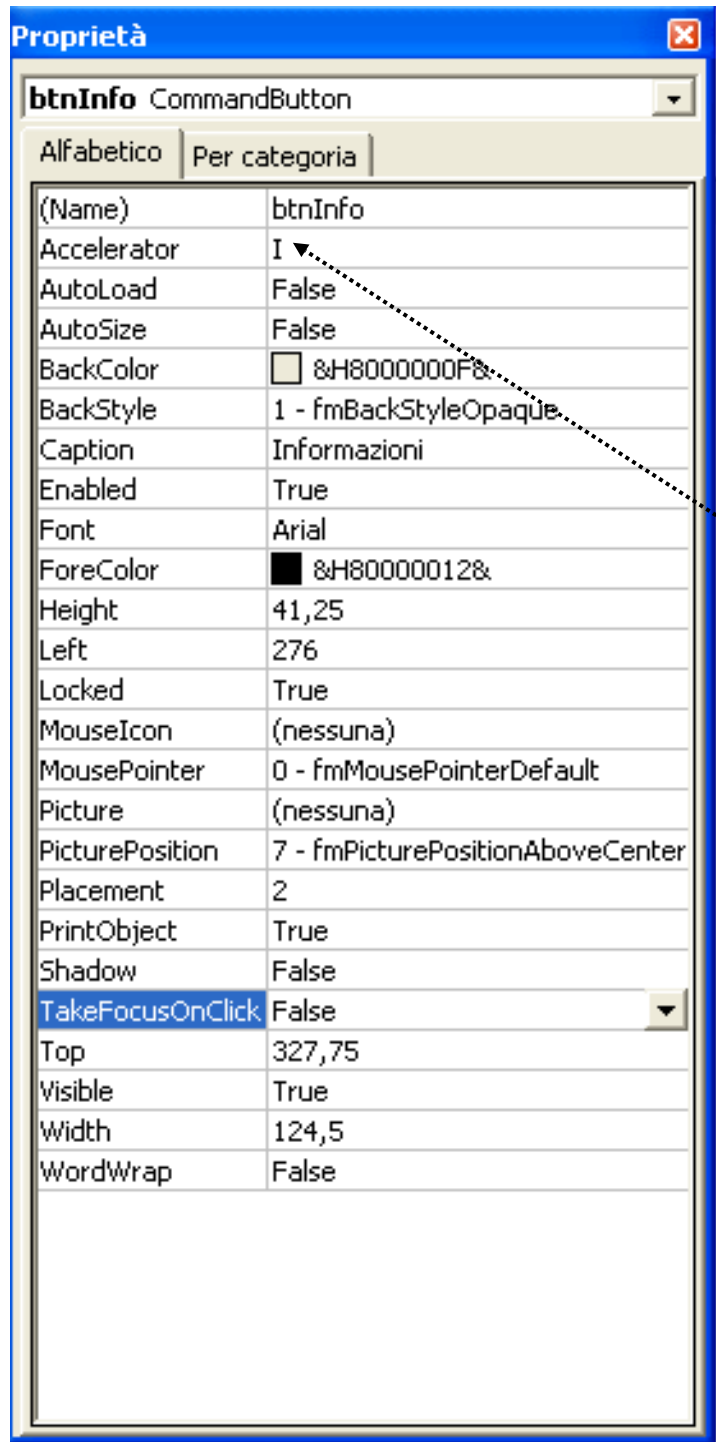


La finestra mostra varie proprietà del pulsante di comando; una particolarmente importante è “(Name)”, che serve ad assegnare il nome al controllo: è buona norma cambiare il nome standard in uno specifico per l'attività che si intende realizzare, come ad es. **bntInfo** (*btn*=bottone, pulsante; *Info* sta per Informazioni).

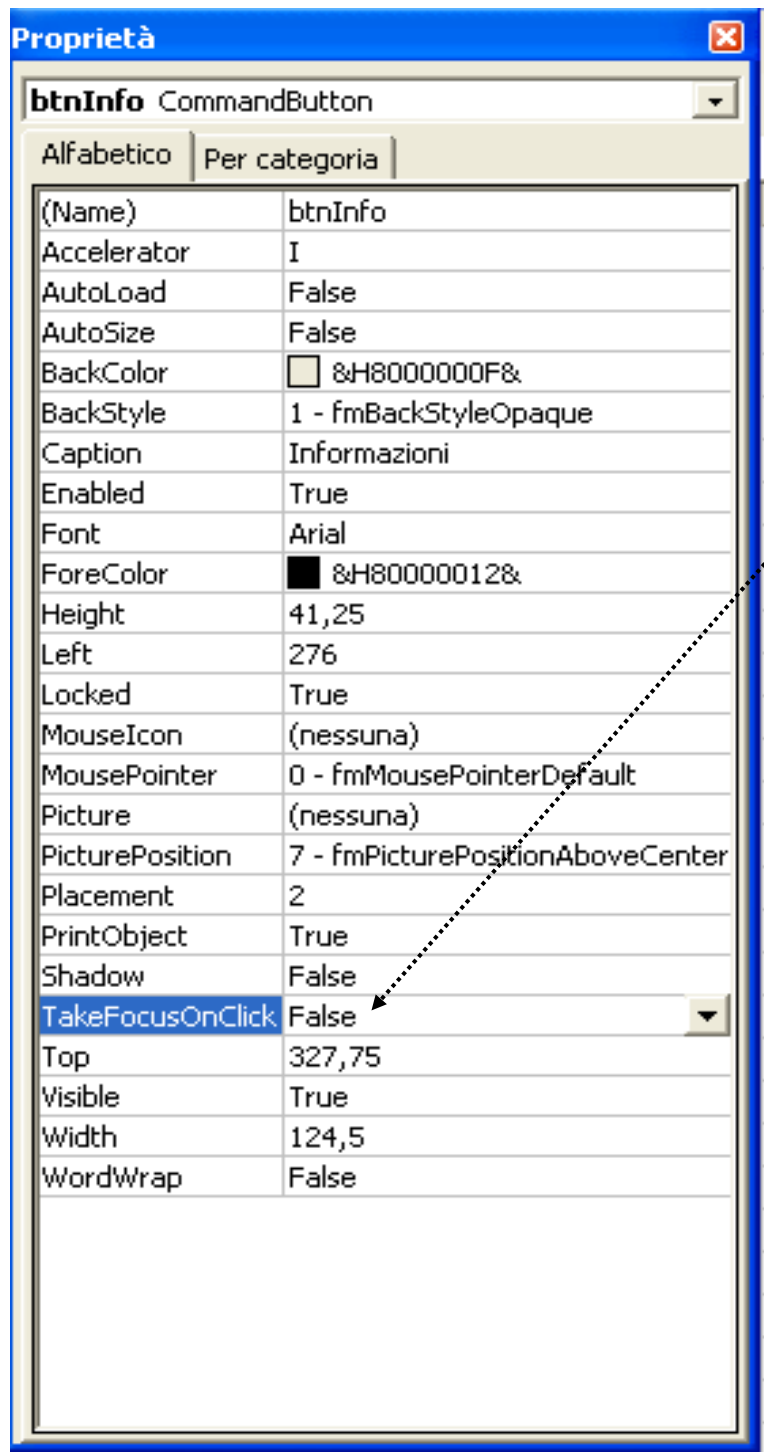


La didascalia sul pulsante invece corrisponde alla proprietà **Caption**; impostiamola a **Informazioni**.



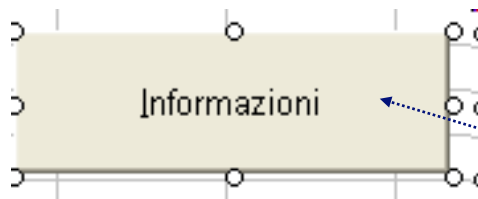
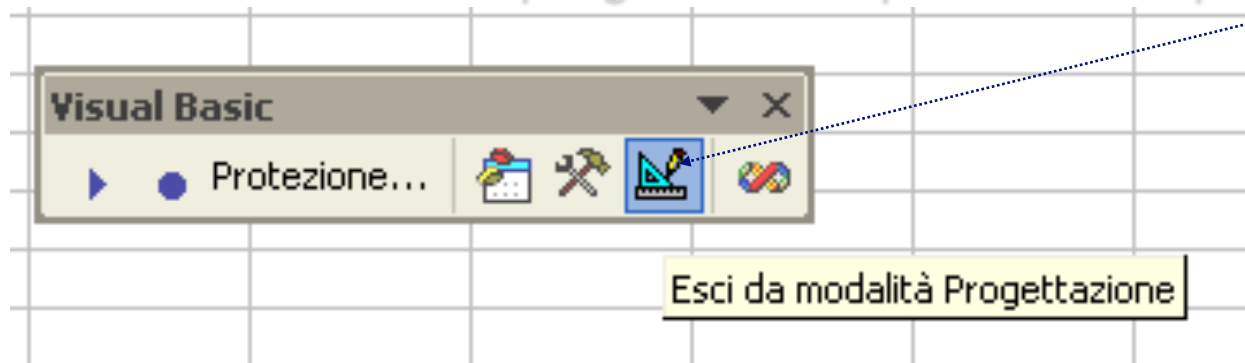


È inoltre possibile specificare un tasto di scelta rapida (ad es. la lettera **I**) tramite la proprietà **Accelerator**; in tal modo la **I** di **Informazioni** nella didascalia diventerà sottolineata e sarà possibile attivare il pulsante anche con la sequenza di tasti **Alt-I**.

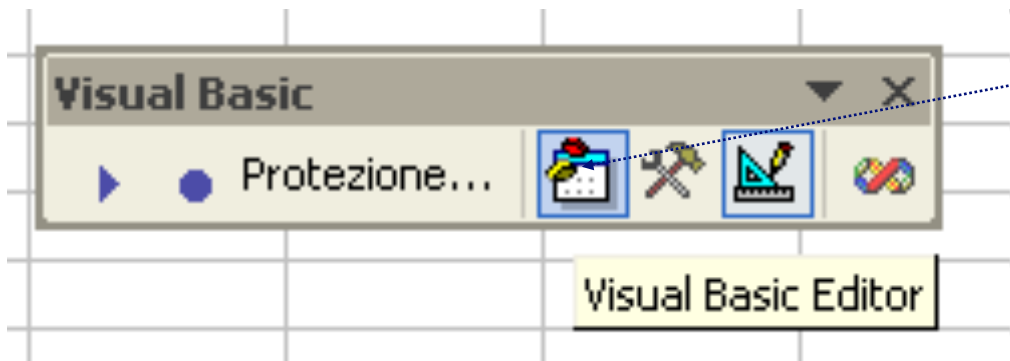


Impostiamo inoltre la proprietà ***TakeFocusOnClick*** a ***False*** per evitare che la cella attiva venga deselezionata alla pressione del pulsante di comando.

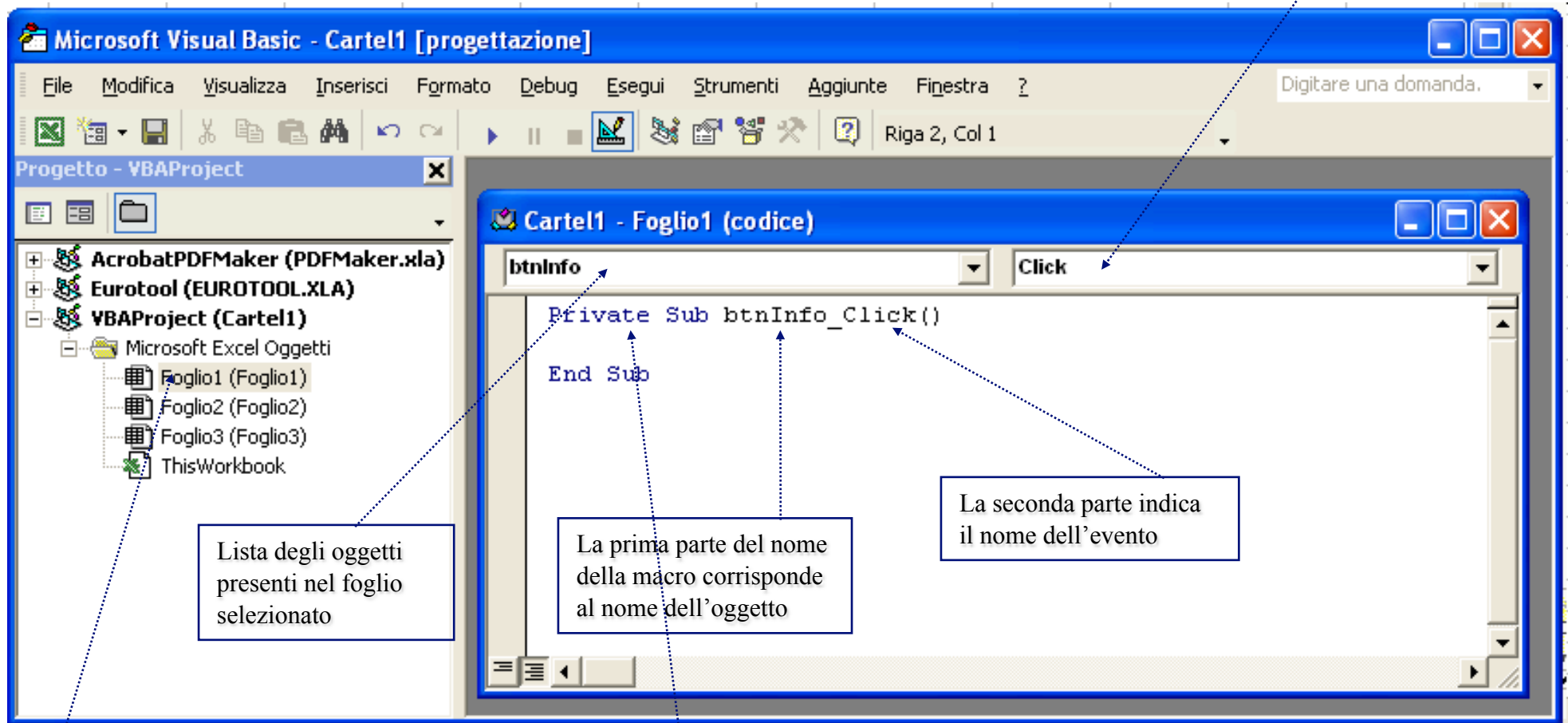
Usciamo dalla modalità di progettazione premendo il pulsante sulla barra VBA:



Premendo il pulsante non accade nulla perché dobbiamo ancora associarvi una macro. Per fare questo, riattiviamo la modalità progettazione ed entriamo nell'Editor Visual Basic facendo semplicemente doppio click sul pulsante.



Si attiverà l'Editor di Visual Basic con una la nuova macro **btnInfo_Click()**:



Il foglio correntemente selezionato

La parola chiave **Private** indica che questa macro non comparirà nella finestra di dialogo *Esegui Macro*; si tratta cioè di una funzionalità esclusivamente legata all'oggetto pulsante di comando **btnInfo**.

Una semplice applicazione sull'uso degli array

La procedura che descriveremo serve ad illustrare alcune delle istruzioni e delle modalità di utilizzo degli array. Posizioniamoci all'interno della macro **btnInfo_Click()** ed inseriamo il codice seguente:

```
Const MAX_FLUSSI = 12 'n.ro massimo di flussi da gestire
Dim Flussi(1 to MAX_FLUSSI) As Currency 'flussi di cassa
Dim Scadenze(1 to MAX_FLUSSI) As Date 'le date delle scadenze
Dim ScadenzaScelta As Date, FlussoScelto As Currency
Dim i As Integer, n As Integer, Messaggio As String
' In questo esempio il numero di scadenze è costante e prefissato a 12.
' Il valore delle scadenze è riportato nella prima colonna del foglio a partire dalla
' prima riga, mentre quello dei flussi si trova nella seconda colonna in corrispon-
' denza di ciascuna scadenza.
```

```
For i = 1 To MAX_FLUSSI
```

```
    Scadenze(i) = Cells(i,1).Value
```

```
    Flussi(i) = Cells(i,2).Value
```

```
Next i
```

‘ Questo ciclo viene eseguito finché l’utente non inserisce il numero 0 o preme Annulla
Do

```
    Messaggio = “Inserire il numero della riga (0=fine):”
```

```
    n = Val(InputBox(Messaggio, “Input”))
```

```
    If n >= 1 And n <= MAX_FLUSSI Then
```

‘la condizione controlla che **n** sia nell’intervallo consentito;

‘quindi si assegna il valore dell’elemento **n**-esimo dei vettori

‘**Flussi** e **Scadenze** alle due variabili **FlussoScelto** e **ScadenzaScelta**

```
        FlussoScelto = Flussi(n)
```

```
        ScadenzaScelta = Scadenze(n)
```



```
Messaggio = "Scadenza = " & Str(ScadenzaScelta) & _  
    " – Flusso = " & Str(FlussoScelto)  
MsgBox Messaggio, vbOKOnly, "Risultati riga n.ro " & Str(n)  
Elseif n > MAX_FLUSSI Then  
    ' se il numero inserito è maggiore di 12 viene segnalato un errore  
    MsgBox "Inserire solo numeri compresi tra 1 e " & MAX_FLUSSI _  
        & "!", vbExclamation + vbOKOnly, "Attenzione!"  
End If  
Loop Until n = 0 'n=0 è la condizione di termine
```

I **vettori** Flussi e Scadenze così definiti sono detti **statici** perché il numero di elementi di cui sono composti (dimensione) non può essere modificato durante l'elaborazione.

È anche possibile definire **vettori *dinamici***, per i quali è possibile cambiare la dimensione durante l'esecuzione.

In tal caso, la definizione dei due vettori sarebbe la seguente:

Dim **Flussi**() As Currency

Dim **Scadenze**() As Date

‘ Si noti che i vettori sono stati dichiarati, ma senza dimensione.

...

ReDim Preserve **Flussi**(1 to n) As Currency

Redim Preserve **Scadenze**(1 to n) As Date

‘ Non appena siamo sicuri di voler “allocare” **n** elementi, effettuiamo il ridimensionamento dei vettori a tale ampiezza.

‘ Osserviamo l'uso della parola-chiave **Preserve** qualora si desideri mantenere il precedente contenuto del vettore.

Oggetti *Range*

Abbiamo visto che i vettori vengono utilizzati per memorizzare elenchi di valori di tipo omogeneo. È altrettanto utile esaminare gli oggetti ***Range***, che individuano un “insieme” di celle e sono probabilmente gli oggetti più importanti in Excel. Infatti, sono oggetti Range gli insiemi ***Cells***, ***Rows*** e ***Columns***.

Oggetti *Range*

Possiamo riferirci alla cella A2 ad es. come **Cells(2, 1)**, analogamente a come abbiamo visto nel codice precedente:

```
For i = 1 to MAX_FLUSSI  
    Scadenze(i) = Cells(i, 1).Value  
    Flussi(i) = Cells(i, 2).Value  
Next i
```

dove la proprietà **Value** è quella di default dell'oggetto e ne indica il contenuto.

Il risultato finale dell'esempio precedente sarà del tipo:

The screenshot shows a Microsoft Excel window titled "Microsoft Excel - Cartel1.xls". The menu bar includes File, Modifica, Visualizza, Inserisci, Formato, Strumenti, Dati, Finestra, and Adobe PDF. The toolbar shows various icons for file operations, editing, and formatting. The active sheet is named "I6". The formula bar shows "fx".

	A	B	C	D	E	F
1	01/01/2004	1000				
2	01/02/2004	2300				
3	01/03/2004	-500				
4	01/04/2004	1200				
5	01/05/2004	400				
6	01/06/2004	-670				
7	01/07/2004	1000				
8	01/08/2004	1230				
9	01/09/2004	300				
10	01/10/2004	400				
11	01/11/2004	500				
12	01/12/2004	350				
13						

A text box labeled "Informazioni flussi" is positioned over the table.

Two VBA dialog boxes are overlaid on the spreadsheet:

- Risultati riga n.ro 9**: Displays "Scadenza = 01/09/2004 - Flusso = 300" and an "OK" button.
- Input**: Prompts the user to "Inserire il numero della riga (0=fine):" with an input field and "OK" and "Annulla" buttons.

Le funzioni *LBound* e *UBound*

Per tenere traccia delle dimensioni di un *array* (statico o dinamico) sono previste le funzioni **LBound** e **UBound** che restituiscono il valore minimo (di solito 1) e massimo per le dimensioni di un vettore.

Le funzioni *LBound* e *UBound*

Ad esempio, per ottenere la dimensione massima del vettore **Scadenze** dovremo scrivere:

UBound(Scadenze)

mentre per ottenere il numero di colonne allocate ad una matrice bidimensionale occorre aggiungere il parametro **dimensione** dopo il nome dell'array:

UBound(TavolaPitagorica, 2)

Tassi d'interesse e di sconto

Sia i il *tasso d'interesse* e V_0 il *capitale investito* tra il tempo t_0 e t_1 ; alla fine del periodo il valore del capitale (*montante*) sarà:

$$V_1 = V_0 + iV_0 = V_0(1 + i)$$

essendo $(1 + i)$ il *fattore di capitalizzazione*.

Se invece vogliamo acquistare V_1 unità di capitale, disponibile al tempo t_1 , e calcolarne il valore al tempo $t_0 < t_1$ otteniamo:

$$V_0 = V_1 / (1 + i) = vV_1$$

dove v è il *fattore di sconto*.