

Appunti Di Microsoft® Visual Basic 5.0



PREMESSA

Abbiamo assemblato questa guida attingendo dall'help in linea di **Microsoft® Visual Basic 5.0**, non con la pretesa di redigere un libro di testo ma con l'intento di creare uno strumento che integrasse l'operato del docente di Sistemi ed Automazione. Esistono in commercio testi riferiti a questo linguaggio di programmazione ma abbiamo preferito riunire in questa guida gli argomenti e gli esempi di maggior utilizzo per l'insegnamento della programmazione ad oggetti nell'ambito della nostra materia. Certamente chi insegna programmazione in istituti che fanno dell'informatica la materia principale o chi programma quotidianamente a livello professionale, troverà questo testo incompleto ma, ripetiamo: a noi serve un qualcosa che tratti solo gli argomenti di nostro interesse e che riporti i relativi esempi. È nostro intento apportare continue migliorie a quanto qui presente; migliorie che sicuramente verranno dettate dall'esperienza che andremo ad acquisire con il passare del tempo.

Indice delle categorie

OGGETTI	Sez. A
CONTROLLI	Sez. B
PROPRIETÀ	Sez. C
OPERATORI	Sez. D
ISTRUZIONI	Sez. E
FUNZIONI	Sez. F
METODI	Sez. G
EVENTI	Sez. H
TIPI DI DATI	Sez. I
ESEMPI	Sez.

Indice per categoria

OGGETTI

[3 MDIForm](#)

CONTROLLI

[3 CheckBox \(casella di controllo\)](#)

[3 CommandButton \(pulsante di comando\)](#)

[4 Frame \(cornice\)](#)

[3 Label \(etichetta\)](#)

[3 OptionButton](#)

[PictureBox \(casella immagine\)](#)

[3 TextBox \(casella di testo\)](#)

[4 Timer](#)

PROPRIETÀ

[3 Alignment](#)

[3 BorderStyle](#)

[3 Caption](#)

[4 ControlBox](#)

[3 Enabled](#)

[Font](#)

[Height \(e Width\)](#)

[Icon](#)

[4 Interval](#)

[Left \(e Top\)](#)

[4 MaxButton](#)

[4 MinButton](#)

[Moveable](#)

[3 MultiLine](#)

[3 Name](#)

[ScrollBars](#)

[3 TabIndex](#)

[3 Text](#)

[3 Value](#)

[3 Visible](#)

OPERATORI

[+](#)

[=](#)

[*](#)

[/](#)

[\](#)

[&](#)

[And](#)

[Eqv](#)

[Is](#)

[Like](#)

[Mod](#)

[Not](#)

[Or](#)

ISTRUZIONI

[Date](#)

[3 Dim](#)

[Do...Loop](#)

[3 For...Next](#)

[For Each...Next](#)

[GoTo](#)

[GoSub...Return](#)

[Let](#)

[Set](#)

[Time](#)

[While...Wend](#)

[With](#)

FUNZIONI

[4 Asc](#)

[4 Chr](#)

[Date](#)

[4 Hex](#)

[4 InputBox](#)

[4 Int \(e Fix\)](#)

[3 LCase](#)

[3 Left](#)

[3 Len](#)

[3 Ltrim \(Rtrim e Trim\)](#)

[3 Mid](#)
[4 MsgBox](#)
[3 Right](#)
[4 Shell](#)
[4 Str](#)
[_Time](#)
[_Timer](#)
[3 Ucase](#)
[4 Val](#)

METODI

[3 Refresh](#)
[3 SetFocus](#)

EVENTI

[3 Change](#)
[3 Click](#)
[_DbClick](#)
[_Load](#)
[_MouseDown \(e MouseUp\)](#)
[3 MouseMove](#)
[_Timer](#)
[_Unload](#)

TIPI DI DATI

[Boolean](#)
[Byte](#)
[Currency](#)
[Date](#)
[Decimal](#)
[Double](#)
[Integer](#)
[Long](#)
[Object](#)
[Single](#)
[String](#)
[Variant](#)

ESEMPI

Sez. A

Categoria Oggetti

OGGETTI

[MDIForm](#)

[Torna all'indice delle Categorie](#)

Oggetto MDIForm

Un form MDI (multiple-document interface) è la finestra che costituisce lo sfondo dell'applicazione e il contenitore dei form la cui proprietà `MDIChild` è impostata su `True`.

Sintassi

MDIForm

Osservazioni

È possibile creare un oggetto **MDIForm** scegliendo **Form MDI** dal menu **Inserisci**.

Un'applicazione può includere un solo oggetto **MDIForm**, ma molti form *MDI secondari*. Se è attivo un form MDI secondario che contiene menu, la relativa barra dei menu sostituirà automaticamente la barra dei menu dell'oggetto **MDIForm**. I form MDI secondari vengono visualizzati all'interno dell'oggetto **MDIForm** come icone.

Un oggetto **MDIForm** può contenere solo i controlli **Menu** e **PictureBox** e i controlli aggiuntivi per i quali è disponibile la proprietà **Align**. Se si desidera includere altri controlli, disegnare una casella immagine sul form e quindi i controlli desiderati all'interno di questa. È possibile utilizzare il metodo **Print** per visualizzare testo in una casella immagine in un oggetto **MDIForm**, ma non per visualizzare testo nell'oggetto **MDIForm** stesso.

L'oggetto **MDIForm** non può essere *a scelta obbligatoria*.

I form MDI secondari vengono creati indipendentemente dall'oggetto **MDIForm**, all'interno del quale tuttavia si trovano sempre in *fase di esecuzione*.

È possibile accedere all'insieme dei controlli di un oggetto **MDIForm** tramite l'insieme **Controls**. Per nascondere, ad esempio, tutti i controlli di un oggetto **MDIForm** è possibile utilizzare il seguente codice:

```
For Each Control in MDIForm1.Controls  
    Control.Visible = False  
Next Control
```

La proprietà **Count** dell'oggetto **MDIForm** indica il numero di controlli dell'insieme **Controls**.

Sez. B

Categoria Controlli

CONTROLLI

[CheckBox \(casella di controllo\)](#)
[CommandButton \(pulsante di comando\)](#)
[Frame \(cornice\)](#)
[Label \(etichetta\)](#)
[OptionButton](#)
[PictureBox \(casella immagine\)](#)
[TextBox \(casella di testo\)](#)
[Timer](#)

[Torna all'indice delle Categorie](#)

Controllo CheckBox (casella di controllo)

Il controllo **CheckBox** (casella di controllo) è costituito da una casella nella quale viene visualizzata una crocetta (X) quando il controllo è selezionato. La crocetta scompare nel momento in cui la casella viene deselezionata. Il controllo **CheckBox** fornisce all'utente opzioni di tipo True/False o Sì/No. È possibile utilizzare gruppi di controlli **CheckBox** per visualizzare le varie opzioni disponibili tra cui selezionare quelle desiderate. È inoltre possibile impostare il valore di un controllo **CheckBox** a livello di programmazione tramite la proprietà **Value**.

Sintassi

CheckBox

Osservazioni

I controlli **CheckBox** e **OptionButton** funzionano in modo simile, con la sola differenza che mentre è possibile selezionare più controlli **CheckBox** contemporaneamente, in un gruppo di controlli **OptionButton** è possibile selezionare un solo controllo alla volta.

Per visualizzare il testo accanto al controllo **CheckBox**, impostare la proprietà **Caption**. Per definire lo stato del controllo, ovvero selezionato, deselezionato o non disponibile, impostare la proprietà **Value**.

Controllo CommandButton (pulsante di comando)

Il controllo **CommandButton** (pulsante di comando) viene utilizzato per iniziare, interrompere o concludere un processo. Quando viene scelto cambia aspetto e appare incassato.

Sintassi

CommandButton

Osservazioni

Per visualizzare del testo in un controllo **CommandButton**, impostare la relativa proprietà **Caption**. È sempre possibile scegliere un controllo **CommandButton** facendo clic su di esso. Se si desidera che quando si preme INVIO venga scelto per impostazione predefinita questo controllo, impostare la proprietà **Default** su **True**. Se si desidera invece che venga scelto quando si preme ESC, impostare la proprietà **Cancel** su **True**.

Controllo Frame (cornice)

Il controllo *Frame* (cornice) consente di raggruppare controlli in modo visibile e identificabile. Può essere utilizzato per suddividere un *form* in modo funzionale, ad esempio separando gruppi di controlli *OptionButton*.

Sintassi

Frame

Osservazioni

Per raggruppare controlli, creare innanzitutto il controllo *Frame* e quindi all'interno di esso i vari controlli. Il controllo *Frame* e i controlli che contiene potranno essere spostati insieme. Se si disegna un controllo all'esterno di un controllo *Frame* e quindi si cerca di spostarlo all'interno di *Frame*, il controllo risulterà sovrapposto a *Frame* e sarà quindi necessario spostare i controlli separatamente.

Per selezionare più controlli in un controllo *Frame*, tenere premuto CTRL e trascinare il mouse per racchiudere i controlli in un riquadro.

Controllo Label (etichetta)

Il controllo *Label* (etichetta) è un controllo di tipo grafico e consente di visualizzare del testo che l'utente può modificare direttamente.

Sintassi

Label

Osservazioni

È possibile scrivere del codice che modifichi il testo visualizzato da un controllo *Label* in risposta a eventi in *fase di esecuzione*. Se, ad esempio, un'applicazione impiega qualche minuto per eseguire una modifica, è possibile visualizzare in un controllo *Label* un messaggio che indichi qual è lo stato dell'elaborazione in corso. Il controllo *Label* può inoltre essere utilizzato per identificare un controllo che non ha la proprietà *Caption*, come ad esempio *TextBox*.

Se si desidera che il controllo *Label* visualizzi correttamente righe di lunghezza variabile o un numero variabile di righe, impostare le proprietà *AutoSize* e *WordWrap*.

Il controllo *Label* può fungere da *destinazione* in una conversazione DDE. Impostare la proprietà *LinkTopic* per stabilire il collegamento, la proprietà *LinkItem* per specificare l'argomento della conversazione e la proprietà *LinkMode* per attivare il collegamento. Dopo aver impostato queste proprietà, verrà tentata l'inizializzazione della conversazione e verrà visualizzato un messaggio se il tentativo non riesce.

Se si desidera specificare un *tasto di scelta* nella proprietà *Caption* di *Label*, impostare la proprietà *UseMnemonic* su *True*. Sarà quindi possibile premere ALT+ il carattere specificato per spostare lo stato attivo sul controllo successivo nell'*ordine di tabulazione*.

Controllo OptionButton

Il controllo *OptionButton* (pulsante di opzione) consente di visualizzare un'opzione che può essere attivata o disattivata.

Sintassi

OptionButton

Osservazioni

In genere, i controlli ***OptionButton*** vengono utilizzati in un gruppo di opzioni tra cui è possibile selezionarne una. Per raggruppare i controlli ***OptionButton***, inserirli all'interno di un contenitore, come ad esempio un controllo ***Frame*** o ***PictureBox*** oppure un form. Per raggrupparli in un controllo ***Frame*** o ***PictureBox***, disegnare innanzitutto il controllo contenitore e quindi i vari controlli ***OptionButton*** al suo interno. Tutti i controlli ***OptionButton*** di uno stesso contenitore costituiscono un singolo gruppo.

I controlli ***OptionButton*** funzionano in modo simile ai controlli ***CheckBox***, con la sola differenza che, mentre è possibile selezionare più controlli ***CheckBox*** contemporaneamente, in un gruppo di controlli ***OptionButton*** è possibile selezionare un solo controllo alla volta. Quando si seleziona un controllo ***OptionButton***, infatti, tutti gli altri controlli del gruppo non sono più disponibili.

Controllo PictureBox (casella immagine)

Il controllo **PictureBox** (casella immagine) consente di visualizzare un elemento grafico da un file in formato **bitmap**, **icona**, **metafile**, **enhanced metafile**, **JPEG** o **GIF**. Se le dimensioni del controllo non sono sufficientemente grandi per visualizzare l'intera immagine, l'elemento grafico verrà ritagliato.

Sintassi

PictureBox

Osservazioni

Il controllo **PictureBox** può essere utilizzato per raggruppare controlli **OptionButton** e per visualizzare l'output di metodi grafici e di testo creato con il metodo **Print**.

Se si desidera che il controllo **PictureBox** venga automaticamente ridimensionato in modo che visualizzi l'intero elemento grafico, impostare la proprietà **AutoSize** del controllo su **True**.

È possibile creare un'animazione o una simulazione manipolando le proprietà e i metodi grafici nel codice. Proprietà e metodi grafici risultano utili per operazioni di stampa in fase di esecuzione, come ad esempio la modifica sullo schermo del formato di un **form** per la stampa.

Il controllo **PictureBox** può fungere da collegamento di destinazione in una conversazione **DDE**.

PictureBox e **Data** sono gli unici controlli standard di **Visual Basic** che possono essere inclusi nell'area interna di un **form** MDI, che è possibile utilizzare per raggruppare controlli lungo il margine superiore o inferiore dell'area interna per la creazione di una barra degli strumenti o di una barra di stato.

Controllo TextBox (casella di testo)

Il controllo **TextBox** (casella di testo), a volte definito campo di modifica o controllo di modifica, consente di visualizzare le informazioni immesse in *fase di progettazione*, immesse dall'utente o assegnate al controllo nel codice in *fase di esecuzione*.

Sintassi

TextBox

Osservazioni

Per visualizzare più righe di testo in un controllo **TextBox**, impostare la proprietà **MultiLine** su **True**. Se un controllo **TextBox** a più righe non include una barra orizzontale, al testo viene automaticamente applicato il ritorno a capo automatico anche quando il controllo viene ridimensionato. Per personalizzare la combinazione delle barre di scorrimento in un controllo **TextBox**, impostare la proprietà **ScrollBars**.

Le barre di scorrimento vengono visualizzate accanto alla casella di testo quando la proprietà **MultiLine** di quest'ultima è impostata su **True** e la proprietà **ScrollBars** è impostata su un valore diverso da **None** (0).

Se si imposta la proprietà **MultiLine** su **True**, è possibile utilizzare la proprietà **Alignment** per impostare l'allineamento del testo all'interno del controllo **TextBox**. Per impostazione predefinita il testo viene allineato a sinistra. Se la proprietà **MultiLine** è impostata su **False**, l'impostazione di **Alignment** non avrà alcun effetto.

Il controllo **TextBox** può fungere da collegamento di *destinazione* in una conversazione DDE.

Controllo Timer

Il controllo *Timer* consente di eseguire del codice a intervalli di tempo regolari generando un evento *Timer*.

Sintassi

Timer

Osservazioni

Il controllo *Timer*, non visibile all'utente, risulta utile per elaborazioni in background.

Non è possibile impostare la proprietà *Enabled* di un controllo *Timer* per la selezione multipla di controlli diversi da controlli *Timer*.

Non vi sono limiti al numero di controlli timer attivi in Visual Basic 5.0 in esecuzione in Windows 95 o Windows NT.

Sez. C

Categoria Proprietà

PROPRIETÀ

[Alignment](#)
[BorderStyle](#)
[Caption](#)
[ControlBox](#)
[Enabled](#)
[Font](#)
[Height \(e Width\)](#)
[Icon](#)
[Interval](#)
[Left \(e Top\)](#)
[MaxButton](#)
[MinButton](#)
[Moveable](#)
[MultiLine](#)
[Name](#)
[ScrollBars](#)
[TabIndex](#)
[Text](#)
[Value](#)
[Visible](#)

[Torna all'indice delle Categorie](#)

Proprietà Alignment

Restituisce o imposta un valore che specifica l'allineamento di un controllo *CheckBox* o *OptionButton*, del testo di un controllo o dei valori contenuti in una colonna di un controllo *DBGrid*. Per i controlli *CheckBox*, *OptionButton* e *TextBox* è di sola lettura in fase di esecuzione.

Sintassi

oggetto.*Alignment* [= numero]

La sintassi della proprietà *Alignment* è composta dalle seguenti parti:

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
numero	Intero che specifica il tipo di allineamento, come indicato nella sezione "Impostazioni".

Impostazioni

Le possibili impostazioni di numero per i controlli *CheckBox* e *OptionButton* sono:

Costante	Impostazione	Descrizione
<i>VbLeftJustify</i>	0	(Predefinita) Il testo è allineato a sinistra; il controllo è allineato a destra.
<i>VbRightJustify</i>	1	Il testo è allineato a destra; il controllo è allineato a sinistra.

Le possibili impostazioni di numero per i controlli *Label* e *TextBox* sono:

Costante	Impostazione	Descrizione
<i>vbLeftJustify</i>	0	(Predefinita) Il testo è allineato a sinistra.
<i>vbRightJustify</i>	1	Il testo è allineato a destra.
<i>vbCenter</i>	2	Il testo è allineato al centro.

Le possibili impostazioni di numero per una colonna *DBGrid* sono:

Costante	Impostazione	Descrizione
<i>dbgLeft</i>	0	Il testo è allineato a sinistra.
<i>DbgRight</i>	1	Il testo è allineato a destra.
<i>DbgCenter</i>	2	Il testo è allineato al centro.
<i>dbgGeneral</i>	3	(Predefinita) Il testo è allineato a sinistra, i numeri sono allineati a destra.

Osservazioni

È possibile visualizzare del testo a destra o a sinistra dei controlli *OptionButton* e *CheckBox*. Per impostazione predefinita il testo è allineato a sinistra.

La proprietà *MultiLine* in un controllo *Textbox* deve essere impostata su *True* per consentire il corretto funzionamento della proprietà *Alignment*. Se l'impostazione della proprietà *MultiLine* di un controllo *TextBox* è *False*, la proprietà *Alignment* verrà ignorata.

Proprietà **BorderStyle**

Restituisce o imposta un valore che specifica lo stile del bordo del controllo. La proprietà **BorderStyle** è di lettura-scrittura in fase di progettazione del controllo ed è di sola lettura in fase di esecuzione.

Sintassi

oggetto.**BorderStyle** [= enum]

La sintassi della proprietà **BorderStyle** è composta dalle seguenti parti:

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
enum	Valore enumerato che determina lo stile del bordo del controllo, come indicato nella sezione "Impostazioni".

Impostazioni

Le possibili impostazioni di *enum* sono:

Impostazione	Descrizione
0-None	Nessun bordo. Impostazione predefinita.
1-Fixed Single	Intorno al controllo viene tracciata una singola riga.

Proprietà Caption

- **Form**: determina il testo visualizzato sulla *barra del titolo* dell'oggetto **Form** o **MDIForm**. Quando il form è ridotto a icona, questo testo viene visualizzato sotto l'*icona* del form.
- **Controllo**: determina il testo visualizzato in un controllo o vicino a esso.
- **Oggetto MenuLine**: determina il testo visualizzato per un controllo **Menu** o per un oggetto dell'insieme **MenuItems**.

Per un controllo **Menu**, la proprietà **Caption** è in genere di lettura/scrittura in *fase di esecuzione*. La proprietà **Caption** è di sola lettura per i menu esposti o resi disponibili da Visual Basic alle aggiunte (add-in), come l'oggetto **MenuLine**.

Sintassi

oggetto.**Caption** [= stringa]

La sintassi della proprietà **Caption** è composta dalle seguenti parti:

Parte	Descrizione
oggetto	<i>Espressione oggetto</i> che definisce un oggetto dell'elenco "Si applica a". Se oggetto viene omesso, verrà utilizzato come oggetto il form associato al <i>modulo</i> attivo.
stringa	<i>Espressione stringa</i> che definisce il testo visualizzato come didascalia.

Osservazioni

Quando si crea un nuovo oggetto, la didascalia predefinita corrisponde all'impostazione predefinita della proprietà **Name**. Questa didascalia predefinita include il nome dell'oggetto seguito da un intero, ad esempio Command1 o Form1. Se si desidera un'etichetta più descrittiva, impostare la proprietà **Caption**.

È possibile utilizzare la proprietà **Caption** per assegnare un *tasto di scelta* a un controllo. Nella didascalia, includere una e commerciale (&) immediatamente prima del carattere che si desidera designare come tasto di scelta. Il carattere verrà sottolineato. Premere ALT in combinazione con il carattere sottolineato per spostare lo *stato attivo* sul controllo desiderato. Per includere una e commerciale in una didascalia senza creare un tasto di scelta, includere due e commerciali (&&). La didascalia conterrà una singola e commerciale e nessun carattere sottolineato.

Le dimensioni della didascalia di un controllo **Label** non hanno limiti. Per i form e tutti gli altri controlli che dispongono di didascalie, il limite è di 255 caratteri.

Per visualizzare la didascalia per un form, impostare la proprietà **BorderStyle** su 1 (singolo fisso) o **vbFixedSingle**, 2 (ridimensionabile) o **vbSizable** o 3 (finestra di dialogo fissa) o **vbFixedDialog**. Se alla barra del titolo del form viene assegnata una didascalia di lunghezza eccessiva, la didascalia verrà tagliata. Se si ingrandisce un form **MDI secondario** all'interno di un oggetto **MDIForm**, la didascalia del form secondario verrà inclusa nella didascalia del form principale.

Suggerimento Per un'etichetta, impostare la proprietà **AutoSize** su **True** in modo che il controllo venga ridimensionato automaticamente e adattato alla didascalia.

Proprietà ControlBox

Restituisce o imposta un valore che indica se su un *form* è visualizzata una casella del menu di controllo in fase di esecuzione. Di sola lettura in fase di esecuzione.

Sintassi

oggetto.*ControlBox*

Il segnaposto oggetto rappresenta un'espressione oggetto che definisce un oggetto dell'elenco "Si applica a".

Impostazioni

Le possibili impostazioni della proprietà *ControlBox* sono:

Impostazione	Descrizione
<i>True</i>	(Predefinita) Visualizza la casella del menu di controllo.
<i>False</i>	Rimuove la casella del menu di controllo.

Osservazioni

Per visualizzare una casella del menu di controllo, è necessario impostare anche la proprietà *BorderStyle* del form su **1** (semplice fisso), **2** (ridimensionabile) o **3** (finestra di dialogo fissa).

La casella del menu di controllo può essere inclusa sia in finestre a scelta obbligatoria che in finestre non a scelta obbligatoria.

I comandi disponibili in fase di esecuzione dipendono dalle impostazioni delle proprietà correlate. Impostando *MaxButton* e *MinButton* su *False*, ad esempio, vengono disattivati i comandi *Ingrandisci* e *Riduci a icona* del menu di controllo, mentre i comandi *Sposta* e *Chiudi* rimangono disponibili.

Nota Le impostazioni delle proprietà *ControlBox*, *BorderStyle*, *MaxButton* e *MinButton* non hanno alcun effetto sull'aspetto del *form* se non in fase di esecuzione.

Proprietà Enabled

Restituisce o imposta un valore che determina se un form o un controllo è in grado di rispondere agli eventi generati dall'utente.

Sintassi

oggetto.**Enabled** [= booleano]

La sintassi della proprietà **Enabled** è composta dalle seguenti parti:

Parte	Descrizione
oggetto	<i>Espressione oggetto</i> che definisce un oggetto dell'elenco "Si applica a". Se oggetto viene omissso, verrà utilizzato come oggetto il form associato al modulo attivo.
booleano	<i>Espressione booleana</i> che specifica se oggetto è in grado di rispondere agli eventi generati dall'utente.

Impostazioni

Le possibili impostazioni di *booleano* sono:

Impostazione	Descrizione
<i>True</i>	(Predefinita) Consente a oggetto di rispondere agli eventi.
<i>False</i>	Impedisce a oggetto di rispondere agli eventi.

Osservazioni

La proprietà **Enabled** consente di attivare o disattivare form e controlli in *fase di esecuzione*. Ad esempio, è possibile disattivare gli oggetti che non si applicano allo stato corrente dell'applicazione. È inoltre possibile disattivare un controllo utilizzato unicamente a fini di visualizzazione, come una casella di testo che fornisca informazioni di sola lettura.

La disattivazione di un controllo **Timer** tramite l'impostazione di **Enabled** su **False** annulla il conto alla rovescia impostato dalla proprietà **Interval** del controllo.

Per un controllo **Menu**, la proprietà **Enabled** è in genere di lettura/scrittura in fase di esecuzione. La proprietà **Enabled** è invece di sola lettura per le voci di menu esposte o rese disponibili da Visual Basic alle aggiunte (add-in), come il comando Gestione aggiunte del menu Aggiunte.

Proprietà Font

Restituisce un oggetto Font.

Sintassi

oggetto.**Font**

Il segnaposto oggetto rappresenta un'espressione oggetto che definisce un oggetto dell'elenco "Si applica a".

Osservazioni

Utilizzare la proprietà **Font** di un oggetto per identificare un oggetto **Font** specifico di cui si desidera utilizzare le proprietà. Ad esempio, nel codice seguente viene modificata l'impostazione della proprietà **Bold** di un oggetto **Font** identificato dalla proprietà **Font** di un oggetto **TextBox**.

```
txtFirstName.Font.Bold = True
```

Proprietà Height (e Width)

Restituiscono o impostano le dimensioni di un oggetto o la larghezza di un oggetto *Columns* di un controllo *DBGrid*. Per gli oggetti *Printer* e *Screen* non sono disponibili in fase di progettazione.

Sintassi

oggetto.*Height* [= numero]

oggetto.*Width* [= numero]

La sintassi delle proprietà *Height* e *Width* è composta dalle seguenti parti:

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
numero	Espressione numerica che specifica le dimensioni di un oggetto come indicato nella sezione "Impostazioni".

Impostazioni

Le dimensioni vengono calcolate nel modo seguente:

- **Form:** vengono calcolate l'altezza e la larghezza esterne del form, compresi i bordi e la barra del titolo.
- **Control:** le dimensioni vengono misurate dal centro del bordo del controllo in modo che i controlli con bordi di spessore diverso vengano allineati correttamente. Queste proprietà utilizzano le unità di scala di un contenitore del controllo.
- Oggetto **Printer:** vengono calcolate le dimensioni fisiche del foglio impostato per la periferica di stampa; non disponibile in fase di progettazione. Se le proprietà vengono impostate in fase di esecuzione, i rispettivi valori verranno utilizzati al posto dell'impostazione della proprietà *PaperSize*.
- Oggetto **Screen:** vengono calcolate l'altezza e la larghezza dello schermo; non disponibile in fase di progettazione e di sola lettura in fase di esecuzione.
- Oggetto **Picture:** vengono calcolate l'altezza e la larghezza dell'immagine espresse in unità HiMetric.

Osservazioni

Per gli oggetti *Form*, *Printer* e *Screen* queste proprietà vengono sempre misurate in twip. Per un **form** o un controllo, i valori di queste proprietà cambiano quando l'oggetto viene ridimensionato dall'utente o mediante il codice. In tutti gli oggetti i valori massimi di queste proprietà dipendono dal sistema.

Se si impostano le proprietà *Height* e *Width* per un driver della stampante che non le accetta, non vengono generati errori e le dimensioni del foglio rimangono invariate. Se invece si impostano queste proprietà per un driver della stampante che accetta solo determinati valori, non vengono generati errori e le proprietà vengono impostate su uno dei valori validi per il driver. Se, ad esempio, si imposta *Height* su 150, la proprietà potrebbe essere impostata su 144.

Utilizzare le proprietà *Height*, *Width*, *Left* e *Top* per operazioni o calcoli basati sull'area totale di un oggetto, come ad esempio il ridimensionamento e lo spostamento. Utilizzare le proprietà *ScaleLeft*, *ScaleTop*, *ScaleHeight* e *ScaleWidth* per operazioni o calcoli basati sull'area interna di un oggetto, come ad esempio il disegno o lo spostamento di oggetti all'interno di un altro oggetto.

Nota Non è possibile modificare la proprietà *Height* per i controlli *DriveListBox* o *ComboBox* la cui proprietà *Style* è impostata su **0** (elenco a discesa combinato) o **2** (elenco a discesa).

Per l'oggetto *Columns* del controllo *DBGrid*, la proprietà *Width* è specificata nell'unità di misura dell'oggetto contenente il controllo *DBGrid*. Il valore predefinito per la proprietà *Width* corrisponde al valore della proprietà *DefColWidth* del controllo *DBGrid*.

Per l'oggetto *Picture*, utilizzare i metodi *ScaleX* e *ScaleY* per convertire le unità HiMetric nella scala desiderata.

Proprietà Icon

Restituisce l'icona visualizzata quando un *form* viene ridotto a icona in fase di esecuzione.

Sintassi

oggetto.*Icon*

Il segnaposto oggetto rappresenta un'espressione oggetto che definisce un oggetto dell'elenco "Si applica a".

Osservazioni

Utilizzare questa proprietà per specificare un'icona per i *form* che possono essere ridotti a icona in fase di esecuzione.

È possibile, ad esempio, assegnare un'icona univoca a un *form* per indicarne la funzione. Specificare l'icona caricandola nella finestra Proprietà in fase di progettazione. Il file che viene caricato deve essere un file in formato *.ico*. Se non si specifica alcuna icona, verrà utilizzata quella predefinita per i *form* di *Visual Basic*.

È possibile scegliere l'icona da utilizzare nella libreria delle icone di Visual Basic (nella sottodirectory Icons). Quando si crea un file eseguibile, è possibile assegnare una icona all'applicazione utilizzando la proprietà *Icon* di un *form* dell'applicazione.

Nota È possibile vedere l'icona di un *form* nell'angolo superiore sinistro del *form* in Windows 95, oppure riducendo il *form* a icona sia in Windows 95 che in Windows NT. Se il *form* è ridotto a icona, per poter vedere l'icona la proprietà *BorderStyle* deve essere impostata su *1* (semplice fisso) o su *2* (ridimensionabile) e la proprietà *MinButton* deve essere impostata su *True*.

In fase di esecuzione, è possibile assegnare la proprietà *Icon* di un oggetto alla proprietà *DragIcon* o *Icon* di un altro oggetto. È inoltre possibile assegnare l'icona restituita dalla funzione *LoadPicture*. Se non si specifica alcun argomento per *LoadPicture*, al *form* verrà assegnata un'icona vuota, alla quale sarà possibile assegnare un'immagine in fase di esecuzione.

Proprietà Left (e Top)

- *Left* restituisce o imposta la distanza tra il margine interno sinistro di un oggetto e il margine sinistro del relativo contenitore.
- *Top* restituisce o imposta la distanza tra il margine interno superiore di un oggetto e il margine superiore del relativo contenitore.

Sintassi

oggetto.*Left* [= valore]

oggetto.*Top* [= valore]

La sintassi delle proprietà *Left* e *Top* è composta dalle seguenti parti:

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
valore	Espressione numerica che specifica la distanza.

Osservazioni

Per un *form*, le proprietà *Left* e *Top* vengono sempre espresse in twip; per un controllo, invece, vengono espresse in base all'unità di misura del sistema di coordinate del contenitore del controllo. I valori di queste proprietà cambiano quando l'oggetto viene spostato dall'utente o mediante il codice. Per i controlli *CommonDialog* e *Timer* queste proprietà non sono disponibili in fase di esecuzione.

Per una qualsiasi di queste proprietà è possibile specificare un numero a precisione singola.

Utilizzare le proprietà ***Left***, ***Top***, ***Height*** e ***Width*** per operazioni basate sulle dimensioni esterne di un oggetto, come ad esempio il ridimensionamento e lo spostamento. Utilizzare le proprietà ***ScaleLeft***, ***ScaleTop***, ***ScaleHeight*** e ***ScaleWidth*** per operazioni basate sulle dimensioni interne di un oggetto, come ad esempio il disegno o lo spostamento di oggetti all'interno di un altro oggetto. Le proprietà relative alle proporzioni vengono applicate solo ai controlli ***PictureBox*** e agli oggetti ***Form*** e ***Printer***.

Proprietà Interval

Restituisce o imposta il numero di millisecondi che intercorrono tra le varie chiamate dell'evento **Timer** del controllo **Timer**.

Sintassi

oggetto.**Interval** [= millisecondi]

La sintassi della proprietà **Interval** è composta dalle seguenti parti:

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
millisecondi	Espressione numerica che specifica il numero di millisecondi come indicato nella sezione "Impostazioni".

Impostazioni

Le possibili impostazioni di millisecondi sono:

Impostazione	Descrizione
0	(Predefinita) Disattiva un controllo Timer.
Da 1 a 65.535	Imposta un intervallo (in millisecondi) che ha effetto quando la proprietà Enabled del controllo Timer è impostata su True . Ad esempio, un valore pari a 10.000 millisecondi è uguale a 10 secondi. Il numero massimo di millisecondi, ovvero 65.535, è equivalente a poco più di 1 minuto.

Osservazioni

È possibile impostare la proprietà **Interval** di un controllo **Timer** sia in fase di progettazione che in fase di esecuzione. Quando si utilizza la proprietà **Interval**, tenere presente quanto segue:

- La proprietà **Enabled** del controllo **Timer** determina se il controllo risponde al trascorrere del tempo. Impostare **Enabled** su **False** per disattivare il controllo **Timer**, impostarla su **True** per attivarlo. Quando un controllo **Timer** è attivo, il relativo conto alla rovescia inizia sempre dal valore impostato per la proprietà **Interval**.
- È necessario creare una routine di eventi **Timer** per indicare quale operazione deve essere eseguita allo scadere di ciascun intervallo di tempo specificato da **Interval**.

Proprietà MaxButton

Restituisce un valore che indica se in un form è disponibile un *pulsante di ingrandimento*.

Sintassi

oggetto.**MaxButton**

Il segnalibro oggetto rappresenta un'espressione oggetto che definisce un oggetto dell'elenco "Si applica a".

Impostazioni

Le possibili impostazioni della proprietà **MaxButton** sono:

Impostazione	Descrizione
<i>True</i>	(Predefinita) Nel form è disponibile un pulsante di ingrandimento.
<i>False</i>	Nel form non è disponibile un pulsante di ingrandimento.

Osservazioni

Il pulsante di ingrandimento consente di ingrandire la finestra di un *form* alle dimensioni dello schermo. Per visualizzarlo, è necessario impostare anche la proprietà **BorderStyle** del *form* su **1** (semplice fisso), su **2** (ridimensionabile) o su **3** (doppio fisso).

Quando si ingrandisce una finestra, il pulsante di ingrandimento diventa il pulsante di ripristino. Quando si ripristinano le dimensioni originali della finestra o la si riduce a icona, il pulsante di ripristino diventa nuovamente il pulsante di ingrandimento.

Le impostazioni delle proprietà **MaxButton**, **MinButton**, **BorderStyle** e **ControlBox** non influiscono sull'aspetto del form se non in fase di esecuzione.

Nota L'ingrandimento di un *form* in fase di esecuzione genera un evento **Resize**. La proprietà **WindowState** riflette lo stato corrente della finestra. Se viene impostata su **2** (ingrandito), il *form* viene ingrandito indipendentemente dalle impostazioni delle proprietà **MaxButton** e **BorderStyle**.

Proprietà MinButton

Restituisce un valore che indica se in un *form* è disponibile un *pulsante di riduzione a icona*.

Sintassi

oggetto.**MinButton**

Il segnaposto oggetto rappresenta un'espressione oggetto che definisce un oggetto dell'elenco "Si applica a".

Valori restituiti

I valori restituiti da *MinButton* sono:

Impostazione	Descrizione
<i>True</i>	(Predefinita) Nel <i>form</i> è disponibile un pulsante di riduzione a icona.
<i>False</i>	Nel form non è disponibile un pulsante di riduzione a icona.

Osservazioni

Il pulsante di riduzione a *icona* consente di ridurre a icona la finestra di un *form*. Per visualizzarlo, è necessario impostare anche la proprietà *BorderStyle* del *form* su **1** (semplice fisso), su **2** (ridimensionabile) o su **3** (doppio fisso).

Le impostazioni delle proprietà *MaxButton*, *MinButton*, *BorderStyle* e *ControlBox* non influiscono sull'aspetto del *form* se non in fase di esecuzione.

Nota La riduzione a icona di un *form* in fase di esecuzione genera un evento *Resize*. La proprietà *WindowState* riflette lo stato corrente della finestra. Se viene impostata su **2** (ingrandito), il *form* viene ingrandito indipendentemente dalle impostazioni delle proprietà *MaxButton* e *BorderStyle*.

Proprietà Moveable

Restituisce o imposta un valore che specifica se è possibile spostare l'oggetto.

Sintassi

oggetto.**Moveable** = booleano

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
booleano	Espressione booleana che specifica se è possibile spostare l'oggetto.

Impostazioni

Le possibili impostazioni di booleano sono:

Costante	Valore	Descrizione
True	-1	L'oggetto può essere spostato.
False	0	L'oggetto non può essere spostato.

Proprietà MultiLine

Restituisce o imposta un valore che indica se un controllo **TextBox** può accettare e visualizzare più righe di testo. Di sola lettura in *fase di esecuzione*.

Sintassi

oggetto.**MultiLine**

Il segnaposto oggetto rappresenta un'*espressione oggetto* che definisce un oggetto dell'elenco "Si applica a".

Impostazioni

Le impostazioni della proprietà **MultiLine** sono:

Impostazione	Descrizione
<i>True</i>	Consente la presenza di più righe di testo.
<i>False</i>	(Predefinita) Ignora i ritorni a capo e colloca i dati su un'unica riga.

Osservazioni

In un controllo **TextBox** a più righe il testo va a capo automaticamente se si estende oltre al bordo della casella di testo.

È possibile aggiungere barre di scorrimento a controlli **TextBox** di grandi dimensioni, utilizzando la proprietà **ScrollBars**. Se non viene specificato alcun controllo **HScrollBar** (barra di scorrimento orizzontale), il testo nel controllo **TextBox** a più righe andrà a capo automaticamente.

Nota In un *form* senza pulsanti predefiniti, premendo INVIO in un controllo **TextBox** a più righe lo stato attivo verrà spostato sulla riga successiva. Se il *form* contiene un pulsante predefinito, per passare alla riga successiva sarà necessario premere CTRL+INVIO.

Proprietà Name

Imposta o restituisce un nome definito dall'utente per un oggetto **DAO**. Per un oggetto non accodato ad un'*insieme*, questa proprietà è di lettura/scrittura.

Impostazioni

L'impostazione o il valore restituito è un tipo di dati **String** che specifica un nome. Il nome deve iniziare con una lettera. Il numero massimo dei caratteri dipende dal tipo di oggetto a cui **Name** si riferisce, come mostrato nella sezione.

Osservazioni

Può includere numeri e caratteri di sottolineatura () ma non caratteri di punteggiatura o spazi.

Proprietà ScrollBars

Restituisce o imposta un valore che indica se un oggetto dispone di barre di scorrimento orizzontali o verticali.
Di sola lettura in fase di esecuzione.

Sintassi

oggetto.**ScrollBars**

Il segnaposto oggetto rappresenta un'espressione oggetto che definisce un oggetto dell'elenco "Si applica a".

Impostazioni

Per un oggetto **MDIForm**, le possibili impostazioni della proprietà **ScrollBars** sono:

Impostazione	Descrizione
True	(Predefinita) Il form dispone di una barra di scorrimento orizzontale o verticale o di entrambe.
False	Il form non dispone di alcuna barra di scorrimento.

Per un controllo **TextBox**, le possibili impostazioni della proprietà **ScrollBars** sono:

Costante	Impostazione	Descrizione
VbSBNone	0	(Predefinita) Nessuna
VbHorizontal	1	Orizzontale
VbVertical	2	Verticale
VbBoth	3	Entrambe

Osservazioni

Per un controllo **TextBox** con impostazione **1** (orizzontale), **2** (verticale) o **3** (entrambe), la proprietà **MultiLine** deve essere impostata su **True**.

In fase di esecuzione, l'ambiente operativo Microsoft Windows implementa automaticamente un'interfaccia di tastiera standard per consentire lo spostamento all'interno di controlli **TextBox** con i tasti di DIREZIONE (freccia SU, freccia GIÙ, freccia SINISTRA e freccia DESTRA), i tasti HOME e FINE e così via.

In un oggetto, le barre di scorrimento vengono visualizzate soltanto se il contenuto dell'oggetto oltrepassa i bordi. Se, ad esempio, in un oggetto **MDIForm** una parte di un **form** secondario è nascosta dietro il bordo del **form** MDI principale, verrà visualizzata una barra di scorrimento orizzontale (controllo **HScrollBar**). In modo analogo, verrà visualizzata una barra di scorrimento verticale su un controllo **TextBox** quando non è possibile visualizzare tutte le righe di testo. Se la proprietà **ScrollBars** è impostata su **False**, l'oggetto non avrà barre di scorrimento, indipendentemente dal contenuto.

Proprietà *TabIndex*

Restituisce o imposta l'ordine di tabulazione della maggior parte degli oggetti contenuti in un *form*.

Sintassi

oggetto.*TabIndex* [= indice]

La sintassi della proprietà *TabIndex* è composta dalle seguenti parti:

Parte	Descrizione
Oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
Indice	Numero intero compreso tra 0 e n-1, dove n è il numero di controlli del <i>form</i> che hanno la proprietà <i>TabIndex</i> . Se si assegna un valore di <i>TabIndex</i> minore di 0 viene generato un errore.

Osservazioni

Vengono inclusi nell'ordine di tabulazione tutti i controlli creati in un *form*, a eccezione di *Menu*, *Timer*, *Data*, *Image*, *Line* e *Shape*. In fase di esecuzione, i controlli non visibili o disattivati e i controlli che non possono ricevere lo stato attivo (i controlli *Frame* e *Label*) rimangono nell'ordine di tabulazione, ma vengono ignorati quando si utilizza TAB per passare da un controllo all'altro.

I vari controlli vengono via via aggiunti alla fine dell'ordine di tabulazione. Se si modifica la proprietà *TabIndex* di un controllo per modificare l'ordine di tabulazione predefinito, la proprietà *TabIndex* di altri controlli verrà automaticamente rinumerata in base a eventuali aggiunte o eliminazioni. È possibile apportare modifiche in fase di progettazione mediante la finestra Proprietà o in fase di esecuzione direttamente nel codice.

Il metodo *ZOrder* non influisce sulla proprietà *TabIndex*.

Nota L'ordine di tabulazione di un controllo non influisce sul tasto di scelta associato al controllo. Se si utilizza il tasto di scelta dei controlli *Frame* o *Label*, lo stato attivo viene spostato sul successivo controllo nell'ordine di tabulazione che può riceverlo.

Quando si caricano *form* salvati come testo ASCII, viene automaticamente assegnato un valore alla proprietà *TabIndex* per i controlli che hanno questa proprietà e non sono elencati nella descrizione del *form*. Se i valori esistenti di controlli caricati successivamente sono in conflitto con valori assegnati in precedenza, ai controlli verranno assegnati automaticamente nuovi valori.

Quando si eliminano uno o più controlli, è possibile utilizzare il comando Annulla per ripristinare i controlli e tutte le relative proprietà eccettuata la proprietà *TabIndex*, che non è possibile ripristinare. Quando si utilizza Annulla, *TabIndex* viene reimpostata alla fine dell'ordine di tabulazione.

Proprietà Text

- Con il controllo **ComboBox** (proprietà **Style** impostata su **0** [elenco a discesa combinato] o su **1** [casella combinata semplice]) e il controllo **TextBox**, restituisce o imposta il testo contenuto nell'area di modifica.
- Con il controllo **ComboBox** (proprietà **Style** impostata su **2** [elenco a discesa]) e il controllo **ListBox**, restituisce l'elemento selezionato nella casella di riepilogo. Il valore restituito è sempre equivalente al valore restituito dall'espressione List(ListIndex). Di sola lettura sia in *fase di progettazione* che in *fase di esecuzione*.

Sintassi

oggetto.**Text** [= stringa]

La sintassi della proprietà **Text** è composta dalle seguenti parti:

Parte	Descrizione
oggetto	<i>Espressione oggetto</i> che definisce un oggetto dell'elenco "Si applica a".
stringa	<i>Espressione stringa</i> che specifica il testo.

Osservazioni

Solo in fase di progettazione, le impostazioni predefinite della proprietà **Text** sono:

- Per i controlli **ComboBox** e **TextBox**, la proprietà **Name** del controllo.
- Per il controllo **ListBox**, una stringa vuota ("").

In un controllo **ComboBox** con la proprietà **Style** impostata su **0** (elenco a discesa combinato) o su **1** (casella combinata semplice) e in un controllo **TextBox**, questa proprietà risulta utile per leggere la stringa contenuta nell'area di modifica del controllo.

In un controllo **ComboBox** o **ListBox** con la proprietà **Style** impostata su **2** (elenco a discesa), è possibile utilizzare la proprietà **Text** per determinare l'elemento selezionato.

L'impostazione di **Text** per un controllo **TextBox** può contenere al massimo 2048 caratteri, a meno che la proprietà **MultiLine** non sia **True**. In quest'ultimo caso il limite massimo è di circa 32K.

Proprietà Value

- Con i controlli **CheckBox** e **OptionButton**, restituisce o imposta lo stato del controllo.
- Con il controllo **CommandButton**, restituisce o imposta un valore che indica se il pulsante è scelto; non disponibile in *fase di progettazione*.
- Con l'oggetto **Field**, restituisce o imposta il contenuto di un campo; non disponibile in fase di progettazione.
- Con i controlli **HScrollBar** e **VScrollBar** restituisce o imposta la posizione corrente della barra di scorrimento. Il valore restituito è sempre compreso tra i valori delle proprietà **Max** e **Min** inclusi.

Sintassi

oggetto.**Value** [= valore]

La sintassi della proprietà **Value** è composta dalle seguenti parti:

Parte	Descrizione
oggetto	<i>Espressione oggetto</i> che definisce un oggetto dell'elenco "Si applica a".
valore	Valore che specifica lo stato, il contenuto o la posizione di un controllo come indicato nella sezione "Impostazioni".

Impostazioni

Le possibili impostazioni di valore sono:

- Controllo **CheckBox**. L'impostazione predefinita **0** indica che il controllo non è selezionato, **1** che è selezionato e **2** che non è disponibile.
- Controllo **CommandButton**. **True** indica che il pulsante è scelto, **False** (impostazione predefinita) che il pulsante non è scelto. Impostando la proprietà **Value** su **True** nel codice viene generato l'evento Click del pulsante.
- Controllo **CommandButton**. **True** indica che il pulsante è scelto, **False** (impostazione predefinita) che il pulsante non è scelto. Impostando la proprietà **Value** su **True** nel codice viene generato l'evento Click del pulsante.
- Oggetto **Field**. L'unico limite è rappresentato dai tipi di dati **Field**.
- Controlli **HScrollBar** e **VScrollBar**. Sono disponibili valori compresi tra **-32.768** e **32.767** per il posizionamento della casella di scorrimento.
- Controllo **OptionButton**. **True** indica che il pulsante è scelto, **False** (impostazione predefinita) che il pulsante non è scelto.

Osservazioni

Viene utilizzata una delle proprietà predefinite dell'oggetto. Non è necessario specificarla nel codice. **Field**, ad esempio, è la proprietà predefinita di un **Recordset**, mentre **Value** è la proprietà predefinita di un oggetto **Field**. Le due istruzioni che seguono sono pertanto equivalenti.

```
Dn.Fields("PubID").Value = X  
Dn("PubID") = X
```

Nella prima istruzione le proprietà predefinite vengono specificate in modo esplicito, nella seconda vengono considerate *implicite*.

Proprietà Visible

Restituisce o imposta un valore che indica se un oggetto è visibile o nascosto.

Sintassi

oggetto.*Visible* [= booleano]

La sintassi della proprietà *Visible* è composta dalle seguenti parti:

Parte	Descrizione
Oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
booleano	Espressione booleana che specifica se l'oggetto è visibile o nascosto.

Impostazioni

Le possibili impostazioni di booleano sono:

Impostazione	Descrizione
<i>True</i>	(Predefinita) L'oggetto è visibile.
<i>False</i>	L'oggetto è nascosto.

Osservazioni

Se si desidera che un oggetto sia invisibile all'avvio, impostare la proprietà *Visible* su *False* in fase di progettazione. L'impostazione di questa proprietà nel codice consente di nascondere e successivamente rivisualizzare un controllo in fase di esecuzione in risposta a un determinato evento.

Nota L'utilizzo dei metodi *Show* e *Hide* in un *form* equivale a impostare nel codice la proprietà *Visible* del *form* rispettivamente su *True* e *False*.

Sez. D

Categoria Operatori

OPERATORI

[+](#)
[-](#)
[*](#)
[/](#)
[\](#)
[&](#)
[And](#)
[Eqv](#)
[Is](#)
[Like](#)
[Mod](#)
[Not](#)
[Or](#)

[Torna all'indice delle Categorie](#)

Operatore +

Utilizzato per sommare due numeri.

Sintassi

risultato = espressione1+espressione2

La sintassi dell'operatore + è composta dalle seguenti parti:

Parte	Descrizione
risultato	Obbligatoria; qualsiasi variabile numerica.
espressione1	Obbligatoria; qualsiasi espressione.
espressione2	Obbligatoria; qualsiasi espressione.

Osservazioni

L'utilizzo dell'operatore + non consente di determinare se verrà eseguita un'operazione di addizione o di concatenamento di stringhe. Per evitare ambiguità nell'esecuzione e per fornire del codice autoesplicativo, si consiglia di utilizzare a questo scopo l'operatore & per il concatenamento.

Se almeno una espressione non è di tipo *Variant*, saranno valide le seguenti regole:

<u>Se</u>	<u>Operazione eseguita</u>
Entrambe le espressioni sono costituite da tipi di dati numerici (<i>Byte, Boolean, Integer, Long, Single, Double, Date, Currency</i> o <i>Decimal</i>)	Addizione
Entrambe le espressioni sono di tipo <i>String</i>	Concatenamento
Un'espressione contiene dati di tipo numerico e l'altra è di tipo <i>Variant</i> (diversa da <i>Null</i>)	Addizione
Un'espressione è di tipo <i>String</i> e l'altra è di tipo <i>Variant</i> (diversa da <i>Null</i>)	Concatenamento
Un'espressione è un valore <i>Variant</i> di tipo <i>Empty</i>	Restituisce come risultato l'altra espressione invariata
Un'espressione contiene dati di tipo <i>numerico</i> e l'altra è di tipo <i>String</i>	Viene generato l'errore Tipo non corrispondente
Entrambe le espressioni sono <i>Null</i>	risultato sarà <i>Null</i>

Se entrambe le espressioni sono di tipo *Variant*, saranno valide le seguenti regole:

<u>Se</u>	<u>Operazione eseguita</u>
Entrambe le espressioni <i>Variant</i> sono di tipo <i>numerico</i>	Addizione.
Entrambe le espressioni <i>Variant</i> sono di tipo <i>stringa</i>	Concatenamento.
Un'espressione <i>Variant</i> è di tipo <i>numerico</i> e l'altra è di tipo <i>stringa</i>	Addizione.

Nel caso di una semplice addizione aritmetica, riguardante soltanto espressioni numeriche, il tipo di dati di risultato è in genere quello dell'espressione più precisa. L'ordine crescente di precisione è *Byte, Integer, Long, Single, Double, Currency* e *Decimal*. Esistono tuttavia alcune eccezioni a tale ordine:

<u>Se</u>	<u>Risultato sarà</u>
Un valore <i>Single</i> e uno <i>Long</i> vengono sommati	un <i>Double</i>
Il tipo di dati di risultato è un valore <i>Variant</i> di tipo <i>Long, Single</i> o <i>Date</i> che non rientra nell'intervallo consentito	convertito in un valore <i>Variant</i> di tipo <i>Double</i>

Il tipo di dati di risultato è un valore *Variant* di tipo *Byte* che non rientra nell'intervallo consentito

convertito in un valore Variant di tipo Integer

Il tipo di dati di risultato è un valore *Variant* di tipo *Integer* che non rientra nell'intervallo consentito

convertito in un valore Variant di tipo Long

Un tipo di dati *Date* viene aggiunto a qualsiasi tipo di dati

Date

Se una o entrambe le espressioni sono *Null*, risultato sarà *Null*. Se entrambe le espressioni hanno valore *Empty*, risultato sarà *Integer*. Se tuttavia una sola espressione ha valore *Empty*, l'altra espressione verrà restituita invariata come valore di risultato.

Nota L'ordine di precisione utilizzato per l'addizione e la sottrazione non è lo stesso utilizzato per la moltiplicazione.

Operatore –

Utilizzato per calcolare la differenza tra due numeri o indicare il valore negativo di una espressione numerica.

Sintassi 1

risultato = numero1–numero2

Sintassi 2

–numero

La sintassi dell'operatore - è composta dalle seguenti parti:

Parte	Descrizione
risultato	Obbligatoria; qualsiasi variabile numerica.
numero	Obbligatoria; qualsiasi espressione numerica.
numero1	Obbligatoria; qualsiasi espressione numerica.
numero2	Obbligatoria; qualsiasi espressione numerica.

Osservazioni

Nella sintassi 1, l'operatore - è l'operatore aritmetico di sottrazione utilizzato per calcolare la differenza tra due numeri. Nella sintassi 2, l'operatore - viene utilizzato come operatore di negazione unario per indicare il valore negativo di un'espressione.

Il tipo di dati di risultato è in genere quello dell'espressione più precisa. L'ordine di precisione crescente è **Byte**, **Integer**, **Long**, **Single**, **Double**, **Currency** e **Decimal**. Esistono tuttavia alcune eccezioni a tale tipo di ordine:

<u>Se</u>	<u>Risultato sarà</u>
La sottrazione viene eseguita tra valori Single e Long	convertito in Double
Il tipo di dati di risultato è un valore Variant di tipo Long , Single o Date che non rientra nell'intervallo consentito	convertito in un valore Variant che contiene un valore Double
Il tipo di dati di risultato è un valore Variant di tipo Byte che non rientra nell'intervallo consentito	convertito in un valore Variant di tipo Integer
Il tipo di dati di risultato è un valore Variant di tipo Integer che non rientra nell'intervallo consentito	convertito in un valore Variant di tipo Long
La sottrazione viene eseguita tra un tipo di dati Date e un qualsiasi altro tipo di dati	Date
La sottrazione viene eseguita tra due espressioni Date	Double

Se una o entrambe le espressioni sono espressioni **Null**, risultato sarà **Null**. Qualsiasi espressione con valore **Empty** sarà considerata uguale a **0**.

Nota L'ordine di precisione utilizzato per l'addizione e la sottrazione non è lo stesso utilizzato per la moltiplicazione.

Operatore *

Utilizzato per moltiplicare due numeri.

Sintassi

risultato = numero1 * numero2

La sintassi dell'operatore * è composta dalle seguenti parti:

Parte	Descrizione
risultato	Obbligatoria; qualsiasi variabile numerica.
numero1	Obbligatoria; qualsiasi espressione numerica.
numero2	Obbligatoria; qualsiasi espressione numerica.

Osservazioni

Il tipo di dati di risultato è in genere quello dell'espressione più precisa. L'ordine di precisione crescente è *Byte*, *Integer*, *Long*, *Single*, *Double*, *Currency* e *Decimal*. Esistono tuttavia alcune eccezioni a tale tipo di ordine:

<u>Se</u>	<u>Risultato sarà</u>
La moltiplicazione viene eseguita tra valori <i>Single</i> e <i>Long</i>	convertito in Double
Il tipo di dati di risultato è un valore <i>Variant</i> di tipo <i>Long</i> , <i>Single</i> o <i>Date</i> che non rientra nell'intervallo consentito	convertito in un valore Variant che contiene un valore Double
Il tipo di dati di risultato è un valore <i>Variant</i> di tipo <i>Byte</i> che non rientra nell'intervallo consentito	convertito in un valore Variant di tipo Integer
Il tipo di dati di risultato è un valore <i>Variant</i> di tipo <i>Integer</i> che non rientra nell'intervallo consentito	convertito in un valore Variant di tipo Long
Se una o entrambe le espressioni sono espressioni <i>Null</i> , risultato sarà <i>Null</i> . Qualsiasi espressione con valore <i>Empty</i> sarà considerata uguale a <i>0</i> .	

Nota L'ordine di precisione utilizzato per la moltiplicazione non è lo stesso utilizzato per l'addizione e la sottrazione.

Operatore /

Utilizzato per dividere due numeri e restituire un risultato con virgola mobile.

Sintassi

risultato = numero1/numero2

La sintassi dell'operatore / è composta dalle seguenti parti:

Parte	Descrizione
risultato	Obbligatoria; qualsiasi variabile numerica.
numero1	Obbligatoria; qualsiasi espressione numerica.
numero2	Obbligatoria; qualsiasi espressione numerica.

Osservazioni

Il tipo di dati di risultato è in genere **Double** o un valore **Variant** di tipo **Double**. Esistono tuttavia alcune eccezioni a tale tipo di ordine:

<u>Se</u>	<u>Risultato sarà</u>
Entrambe le espressioni sono Byte, Integer o Single	un valore Single a meno che non superi l'intervallo ammesso. In quest'ultimo caso, verrà generato un errore
Entrambe le espressioni sono Variant di tipo Byte, Integer o Single	un valore Variant di tipo Single a meno che non superi l'intervallo ammesso. In quest'ultimo caso, risultato sarà un valore Variant che contiene un valore Double
La divisione riguarda il tipo di dati Decimal e qualsiasi altro tipo di dati	un tipo di dati Decimal
Se una o entrambe le espressioni sono espressioni Null , risultato sarà Null . Qualsiasi espressione con valore Empty sarà considerata uguale a 0 .	

Operatore \

Utilizzato per dividere due numeri e restituire un risultato intero.

Sintassi

risultato = numero1 \ numero2

La sintassi dell'operatore \ è composta dalle seguenti parti:

Parte	Descrizione
Risultato	Obbligatoria; qualsiasi variabile numerica.
numero1	Obbligatoria; qualsiasi espressione numerica.
numero2	Obbligatoria; qualsiasi espressione numerica.

Osservazioni

Prima di eseguire la divisione, le espressioni numeriche vengono arrotondate in espressioni di tipo *Byte*, *Integer* o *Long*.

Il tipo di dati di risultato è in genere *Byte*, un valore *Variant* con sottotipo *Byte*, *Integer*, *Variant Integer*, *Long* o *Variant Long* indipendentemente dal fatto che risultato sia o meno un numero intero. La parte frazionaria verrà eliminata. Se tuttavia un'espressione è *Null*, risultato sarà *Null*. Qualsiasi espressione con valore *Empty* sarà considerata uguale a *0*.

Operatore &

Utilizzato per concatenare le stringhe di due espressioni.

Sintassi

risultato = espressione1 & espressione2

La sintassi dell'operatore & è composta dalle seguenti parti:

Parte	Descrizione
Risultato	Obbligatoria; qualsiasi variabile String o Variant.
espressione1	Obbligatoria; qualsiasi espressione.
espressione2	Obbligatoria; qualsiasi espressione.

Osservazioni

Quando espressione non è una stringa viene convertita in un valore *Variant* di tipo *String*. Il tipo di dati di risultato è *String* se entrambe le espressioni sono espressioni stringa, altrimenti risultato è un valore *Variant* di tipo *String*. Se entrambe le espressioni sono *Null*, risultato sarà *Null*. Se, tuttavia, solo un'espressione è *Null*, tale espressione sarà considerata come una stringa di lunghezza zero quando viene concatenata all'altra espressione. Qualsiasi espressione con valore *Empty* sarà considerata equivalente a una stringa di lunghezza zero.

Operatore And

Utilizzato per eseguire un collegamento logico tra due espressioni.

Sintassi

risultato = espressione1 *And* espressione2

La sintassi dell'operatore *And* è composta dalle seguenti parti:

Parte	Descrizione
risultato	Obbligatoria; qualsiasi variabile numerica.
espressione1	Obbligatoria; qualsiasi espressione.
espressione2	Obbligatoria; qualsiasi espressione.

Osservazioni

Solo nel caso in cui entrambe le espressioni diano come risultato *True*, risultato sarà *True*. Se una delle espressioni dà come risultato *False*, risultato sarà *False*. La seguente tabella mostra come viene determinato risultato:

<u>Valore di espressione1</u>	<u>Valore di espressione2</u>	<u>Valore di risultato</u>
True	True	True
True	False	False
True	Null	Null
False	True	False
False	False	False
False	Null	False
Null	True	Null
Null	False	False
Null	Null	Null

L'operatore *And* esegue un confronto bit per bit dei bit collocati nella stessa posizione in due diverse espressioni numeriche e imposta il bit corrispondente in risultato in base alla seguente tabella:

<u>Valore del bit in espressione1</u>	<u>Valore del bit in espressione2</u>	<u>Valore di risultato</u>
0	0	0
0	1	0
1	0	0
1	1	1

Operatore Eqv

Utilizzato per stabilire un'equivalenza logica tra due espressioni.

Sintassi

risultato = espressione1 *Eqv* espressione2

La sintassi dell'operatore *Eqv* è composta dalle seguenti parti:

Parte	Descrizione
Risultato	Obbligatoria; qualsiasi variabile numerica.
espressione1	Obbligatoria; qualsiasi espressione.
espressione2	Obbligatoria; qualsiasi espressione.

Osservazioni

Quando una delle due espressioni è un valore *Null*, anche risultato sarà *Null*. Quando nessuna delle due espressioni è *Null*, risultato sarà determinato in base alla seguente tabella:

<u>Valore di espressione1</u>	<u>Valore di espressione2</u>	<u>Valore di risultato</u>
True	True	True
True	False	False
False	True	False
False	False	True

L'operatore *Eqv* esegue un confronto bit per bit dei bit collocati nella stessa posizione in due diverse espressioni numeriche e imposta il bit corrispondente in risultato in base alla seguente tabella:

<u>Valore di espressione1</u>	<u>Valore di espressione2</u>	<u>Valore di risultato</u>
0	0	1
0	1	0
1	0	0
1	1	1

Operatore Is

Utilizzato per confrontare due variabili che fanno riferimento a oggetti.

Sintassi

risultato = oggetto1 *Is* oggetto2

La sintassi dell'operatore *Is* è composta dalle seguenti parti:

Parte	Descrizione
risultato	Obbligatoria; qualsiasi variabile numerica.
oggetto1	Obbligatoria; qualsiasi espressione.
oggetto2	Obbligatoria; qualsiasi espressione.

Osservazioni

Se oggetto1 e oggetto2 si riferiscono allo stesso oggetto, risultato sarà *True*; in caso contrario, risultato sarà *False*. Due variabili possono fare riferimento allo stesso oggetto in diversi modi.

Nel seguente esempio, A è stata impostata in modo che faccia riferimento allo stesso oggetto di B:

Set A = B

Nel seguente esempio A e B sono state impostate in modo che facciano riferimento allo stesso oggetto di C:

Set A = C

Set B = C

Operatore Like

Viene utilizzato per confrontare un'espressione stringa con un criterio di ricerca contenuto in un'espressione SQL.

Sintassi

espressione *Like* "criteriodiricerca"

La sintassi dell'operatore *Like* è composta dalle parti descritte di seguito.

Parte	Descrizione
espressione	Espressione SQL utilizzata in una proposizione WHERE.
criteriodiricerca	Stringa o carattere visualizzato con il quale viene confrontata espressione.

Osservazioni

È possibile utilizzare l'operatore *Like* per ricercare in un campo di valori che corrispondono al criterio di ricerca specificato. Per criteriodiricerca è possibile specificare il valore completo come, ad esempio, *Like* "Rossi" oppure si possono utilizzare i caratteri jolly per ricercare un intervallo di valori come, ad esempio, *Like* "Ro*". In un'espressione, è possibile utilizzare l'operatore *Like* per confrontare un valore di campo con un'espressione stringa. Se, ad esempio, si immette *Like* "C*" in una query SQL, essa restituirà tutti i valori di campo che iniziano con la lettera C. In una query con parametri, è possibile richiedere all'utente di specificare un criterio di ricerca.

Nell'esempio seguente vengono restituiti i dati che iniziano con la lettera P seguita da qualsiasi lettera compresa tra A e F e tre cifre:

Like "P[A-F]###"

Nella tabella seguente viene mostrato come utilizzare l'operatore *Like* per valutare le espressioni in base a differenti criteri di ricerca.

<u>Tipo di Corrispondenza</u>	<u>Criterio di ricerca</u>	<u>Corrispondenza (restituisce True)</u>	<u>Nessuna corrispondenza (restituisce False)</u>
Caratteri multipli	a*a *ab*	aa, aBa, aBBBa abc, AABB, Xab	aBC aZb, bac
Carattere speciale	a[*]a	a*a	aaa
Caratteri multipli	ab*	abcdefg, abc	cab, aab
Carattere singolo	a?a	aaa, a3a, aBa	aBBBa
Cifra singola	a#a	a0a, a1a, a2a	aaa, a10a
Intervallo di caratteri	[a-z]	f, p, j	2, &
Non compresa in un intervallo	[!a-z]	9, &, %	b, a
Non è una cifra	[!0-9]	A, a, &, ~	0, 1, 9
Combinata	a[!b-m]#	An9, az0, a99	abc, aj0

Operatore Mod

Utilizzato per dividere due numeri e restituire soltanto il resto della divisione.

Sintassi

risultato = numero1 *Mod* numero2

La sintassi dell'operatore *Mod* è composta dalle seguenti parti:

Parte	Descrizione
risultato	Obbligatoria; qualsiasi variabile numerica.
numero1	Obbligatoria; qualsiasi espressione numerica.
numero2	Obbligatoria; qualsiasi espressione numerica.

Osservazioni

L'operatore *Mod*, o resto, divide numero1 per numero2, arrotondando i numeri a virgola mobile in numeri interi e restituisce come risultato soltanto il resto della divisione. Come, ad esempio, nell'espressione che segue in cui A (risultato) è uguale a 5.

A = 19 *Mod* 6.7

Il tipo di dati di risultato è in genere *Byte*, *Variant Byte*, *Integer*, *Variant Integer*, *Long* o *Variant* che contiene un valore *Long*, indipendentemente dal fatto che risultato sia o meno un numero intero. La parte frazionaria verrà troncata. Se tuttavia un'espressione è *Null*, risultato sarà *Null*. Qualsiasi espressione con valore *Empty* sarà considerata uguale a 0.

Operatore Not

Utilizzato per eseguire una negazione logica di un'espressione.

Sintassi

risultato = *Not* espressione

La sintassi dell'operatore *Not* è composta dalle seguenti parti:

Parte	Descrizione
risultato	Obbligatoria; qualsiasi variabile numerica.
espressione	Obbligatoria; qualsiasi espressione.

Osservazioni

La seguente tabella mostra come viene determinato risultato:

<u>Valore di espressione</u>	<u>Valore di risultato</u>
True	False
False	True
Null	Null

L'operatore *Not* inverte inoltre i valori dei bit in qualsiasi variabile e imposta il bit corrispondente in risultato in base alla seguente tabella:

<u>Bit in espressione</u>	<u>Bit in risultato</u>
0	1
1	0

Operatore Or

Utilizzato per eseguire una disgiunzione logica tra due espressioni.

Sintassi

risultato = espressione1 **Or** espressione2

La sintassi dell'operatore **Or** è composta dalle seguenti parti:

Parte	Descrizione
risultato	Obbligatoria; qualsiasi variabile numerica.
espressione1	Obbligatoria; qualsiasi espressione.
espressione2	Obbligatoria; qualsiasi espressione.

Osservazioni

Se una o entrambe le espressioni rappresentano un valore **True**, risultato sarà **True**. La seguente tabella mostra come viene determinato risultato:

<u>Valore di espressione1</u>	<u>Valore di espressione2</u>	<u>Valore di risultato</u>
True	True	True
True	False	True
True	Null	True
False	True	True
False	False	False
False	Null	Null
Null	True	True
Null	False	Null
Null	Null	Null

L'operatore **Or** esegue un confronto bit per bit dei bit collocati nella stessa posizione in due diverse espressioni numeriche e imposta il bit corrispondente in risultato in base alla seguente tabella:

<u>Valore del bit in espressione1</u>	<u>Valore del bit in espressione2</u>	<u>Valore di risultato</u>
0	0	0
0	1	1
1	0	1
1	1	1

Sez. E

Categoria Istruzioni

ISTRUZIONI

[_Date](#)
[Dim](#)
[Do...Loop](#)
[For...Next](#)
[For Each...Next](#)
[GoTo](#)
[GoSub...Return](#)
[Let](#)
[Set](#)
[_Time](#)
[While...Wend](#)
[With](#)

[Torna all'indice delle Categorie](#)

Istruzione Date

Imposta la data di sistema corrente.

Sintassi

Date = data

Nei computer che eseguono Microsoft Windows 95 l'argomento data deve essere una data compresa tra l'1 gennaio 1980 e il 31 dicembre 2099. Nei computer che eseguono Microsoft Windows NT, l'argomento data deve essere una data compresa tra l'1 gennaio 1980 e il 31 dicembre 2079.

Istruzione Dim

Dichiara le *variabili* e assegna lo spazio di archiviazione.

Sintassi

Dim [*WithEvents*] *nomevariabile*[(*[indici]*)] [*As* [*New*] *tipo*] [, [*WithEvents*] *nomevariabile*[(*[indici]*)] [*As* [*New*] *tipo*]] . . .

La sintassi dell'istruzione *Dim* è composta dalle seguenti parti:

Parte	Descrizione
WithEvents	Facoltativa. <i>Parola chiave</i> che specifica che <i>nomevariabile</i> è una <i>variabile oggetto</i> utilizzata per rispondere agli eventi generati da un <i>oggetto ActiveX</i> . Valida solo nei <i>moduli di classe</i> . È possibile dichiarare il numero desiderato di variabili con <i>WithEvents</i> , ma non è possibile creare <i>matrici</i> con <i>WithEvents</i> . Non è consentito utilizzare <i>New</i> con <i>WithEvents</i> .
<i>nomevariabile</i>	Obbligatoria. Nome della variabile, espresso in base alle convenzioni di denominazione standard delle variabili.
indici	Facoltativa. Dimensioni di una variabile matrice. È possibile dichiarare fino a 60 dimensioni multiple. La sintassi dell'argomento <i>indici</i> è descritta di seguito: [<i>inferiore To</i>] <i>superiore</i> [, [<i>inferiore To</i>] <i>superiore</i>] . . . Se non specificato in modo esplicito con inferiore, il limite inferiore di una matrice viene controllato dall'istruzione <i>Option Base</i> . Se non è presente un'istruzione <i>Option Base</i> il limite inferiore sarà zero.
New	Facoltativa. Parola chiave che consente di creare un oggetto in modo implicito. Se si utilizza <i>New</i> per la dichiarazione della variabile oggetto, verrà creata una nuova istanza dell'oggetto in occasione del primo riferimento a tale oggetto e pertanto non è necessario utilizzare l'istruzione <i>Set</i> per assegnare il riferimento all'oggetto. La parola chiave <i>New</i> non può essere utilizzata per dichiarare variabili di qualsiasi <i>tipo di dati</i> intrinseco, per dichiarare istanze di oggetti dipendenti e con <i>WithEvents</i> .
<i>tipo</i>	Facoltativa. Tipo di dati della variabile; può essere <i>Byte</i> , <i>Boolean</i> , <i>Integer</i> , <i>Long</i> , <i>Currency</i> , <i>Single</i> , <i>Double</i> , <i>Decimal</i> (non ancora supportato), <i>Date</i> , <i>String</i> (per stringhe di lunghezza variabile), <i>String</i> * lunghezza (per stringhe di lunghezza fissa), <i>Object</i> , <i>Variant</i> , un <i>tipo definito dall'utente</i> oppure un <i>tipo di oggetto</i> . Utilizzare una proposizione <i>As tipo</i> distinta per ciascuna variabile da dichiarare.

Osservazioni

Le variabili dichiarate con **Dim** a *livello di modulo* sono disponibili per tutte le routine incluse in tale *modulo*. A *livello di routine*, le variabili sono disponibili solo all'interno della routine.

Utilizzare un'istruzione **Dim** a livello di modulo o di routine per dichiarare il tipo di dati di una variabile. L'istruzione che segue, ad esempio, dichiara una variabile di tipo **Integer**.

Dim NumberOfEmployees **As** Integer

L'istruzione **Dim** può essere inoltre utilizzata per dichiarare il tipo di oggetto di una variabile. L'istruzione che segue dichiara una variabile per una nuova istanza di un foglio di lavoro (Worksheet).

Dim X **As** New Worksheet

Se la parola chiave **New** non viene utilizzata nella dichiarazione di una variabile oggetto, alla variabile che fa riferimento all'oggetto deve essere assegnato un oggetto esistente tramite l'istruzione **Set** prima di poter essere utilizzata. Fino a quando non le viene assegnato un oggetto, la variabile oggetto dichiarata avrà il valore speciale **Nothing**, che indica che tale variabile non si riferisce a nessuna istanza particolare di un oggetto.

È inoltre possibile utilizzare l'istruzione **Dim** con parentesi vuote per dichiarare matrici dinamiche. Dopo aver dichiarato una matrice dinamica, utilizzare l'istruzione **ReDim** all'interno di una routine per definire il numero di dimensioni e di elementi della matrice. Se si prova a ridichiarare una dimensione di una variabile matrice di cui sono già state dichiarate le dimensioni con un'istruzione **Private**, **Public** o **Dim**, verrà generato un errore.

Se non si specifica un tipo di dati o un tipo di oggetto e il modulo non include un'istruzione **Deftipo**, la variabile sarà di tipo **Variant** per impostazione predefinita.

Le variabili numeriche vengono inizializzate su 0. Le stringhe di lunghezza variabile vengono inizializzate come stringhe di lunghezza zero e le stringhe di lunghezza fissa vengono riempite con zeri. Le variabili **Variant** vengono inizializzate su **Empty**. Gli elementi di tipo definito dall'utente vengono inizializzati come se fossero variabili distinte.

Nota Quando si utilizza l'istruzione **Dim** in una routine, in genere l'istruzione **Dim** viene inserita all'inizio della routine stessa.

Istruzione Do...Loop

Ripete un blocco di istruzioni finché la valutazione di una condizione dà come risultato **True** o fino a quando non dà come risultato **True**.

Sintassi

```
Do [{While | Until} condizione]  
[istruzioni]  
[Exit Do]  
[istruzioni]  
Loop
```

In alternativa, è possibile utilizzare la seguente sintassi equivalente:

```
Do  
[istruzioni]  
[Exit Do]  
[istruzioni]  
Loop [{While | Until} condizione]
```

La sintassi dell'istruzione **Do Loop** è composta dalle seguenti parti:

Parte	Descrizione
condizione	Facoltativa. Espressione numerica o espressione stringa che può dare come risultato True o False . Se condizione è Null , viene considerata come False .
istruzioni	Una o più istruzioni ripetute finché la valutazione di condizione è True .

Osservazioni

È possibile inserire un qualsiasi numero di istruzioni **Exit Do** in qualsiasi punto dell'istruzione **Do...Loop** come metodo alternativo per uscire dall'istruzione **Do...Loop**. **Exit Do** viene spesso utilizzata dopo la valutazione di una condizione, ad esempio **If...Then**, per trasferire il controllo all'istruzione immediatamente successiva all'istruzione **Loop**.

Quando le istruzioni **Do...Loop** sono nidificate, il controllo viene trasferito al ciclo che si trova al livello di nidificazione immediatamente superiore al ciclo in cui ha luogo **Exit Do**.

Istruzione For...Next

Ripete un gruppo di istruzioni per il numero di volte specificato.

Sintassi

```
For contatore = inizio To fine [Step incremento]
[istruzioni]
[Exit For]
[istruzioni]
Next [contatore]
```

La sintassi dell'istruzione *For...Next* è composta dalle seguenti parti:

Parte	Descrizione
contatore	<u>Obbligatoria</u> . Variabile numerica utilizzata come contatore di ciclo. La variabile non può essere <i>Boolean</i> o un elemento di una matrice.
inizio	<u>Obbligatoria</u> . Valore iniziale del contatore.
fine	<u>Obbligatoria</u> . Valore finale del contatore.
incremento	<u>Facoltativa</u> . Quantità di cui viene incrementato il contatore al compimento di ciascun ciclo. Se non viene specificato, incremento assume per impostazione predefinita il valore <i>I</i> .
istruzioni	<u>Facoltativa</u> . Una o più istruzioni comprese tra <i>For</i> e <i>Next</i> da eseguire per il numero di volte specificato.

Osservazioni

L'argomento incremento può essere sia positivo che negativo. Il valore dell'argomento incremento determina l'esecuzione del ciclo come di seguito indicato:

Valore	Il ciclo viene eseguito se
--------	----------------------------

Positivo o 0	contatore <= fine
Negativo	contatore >= fine

Dopo che si è entrati nel ciclo e che tutte le istruzioni del ciclo sono state eseguite, incremento viene sommato a contatore. A questo punto verranno eseguite di nuovo le istruzioni del ciclo, in base allo stesso test che ha causato la prima esecuzione del ciclo, oppure il ciclo terminerà e l'esecuzione continuerà con l'istruzione successiva a *Next*.

Suggerimento La modifica del valore di un contatore dall'interno di un ciclo può complicare la lettura e il *debug* del codice.

È possibile inserire un qualsiasi numero di istruzioni *Exit For* nel ciclo, come metodo d'uscita alternativo. *Exit For* viene spesso utilizzata dopo la valutazione di una condizione, ad esempio *If...Then*, per trasferire il controllo all'istruzione immediatamente successiva all'istruzione *Next*.

È possibile nidificare cicli *For...Next* inserendo un ciclo *For...Next* all'interno di un altro. Assegnare come contatore a ciascun ciclo un nome di variabile univoco. La seguente costruzione è corretta:

```
For I = 1 To 10
  For J = 1 To 10
    For K = 1 To 10
      Istruzioni...
    Next K
  Next J
Next I
```

Nota Se si omette contatore in un'istruzione *Next*, l'istruzione verrà eseguita come se contatore fosse incluso. Se l'istruzione *Next* viene incontrata prima dell'istruzione *For* corrispondente, verrà generato un errore.

Istruzione For Each...Next

Ripete un gruppo di istruzioni per ogni elemento di una matrice o di un insieme.

Sintassi

For Each elemento In gruppo
[istruzioni]
[Exit For]
[istruzioni]
Next [elemento]

La sintassi dell'istruzione *For...Each...Next* è composta dalle seguenti parti:

Parte	descrizione
elemento	<u>Obbligatoria</u> . Variabile utilizzata per l'interazione tra gli elementi dell'insieme o della matrice. Negli insiemi, elemento può essere soltanto una variabile <i>Variant</i> , una variabile oggetto generica o una variabile oggetto di automazione specifica. Nelle matrici, elemento può essere soltanto una variabile <i>Variant</i> .
gruppo	<u>Obbligatoria</u> . Nome di un insieme o di una matrice di oggetti, a eccezione delle matrici di tipi definiti dall'utente.
istruzioni	<u>Facoltativa</u> . Una o più istruzioni che vengono eseguite per ciascun elemento del gruppo.

Osservazioni

Il blocco *For...Each* viene utilizzato in presenza di almeno un elemento indicato in gruppo. Una volta avviato il ciclo, tutte le istruzioni del ciclo verranno eseguite per il primo elemento indicato in gruppo. Quindi le istruzioni del ciclo verranno eseguite per gli altri elementi, se presenti. Nel caso non vi siano altri elementi, si uscirà dal ciclo e verrà eseguita l'istruzione successiva all'istruzione *Next*.

È possibile inserire un qualsiasi numero di istruzioni *Exit For* nel ciclo, come metodo d'uscita alternativo. *Exit For* viene spesso utilizzata con la valutazione di una condizione, ad esempio *If...Then*, per trasferire il controllo all'istruzione immediatamente successiva all'istruzione *Next*.

È possibile nidificare cicli *For...Each...Next* inserendo un ciclo *For...Each...Next* all'interno di un altro. Ogni elemento del ciclo deve essere tuttavia univoco.

Nota Se si omette elemento in un'istruzione *Next*, l'istruzione verrà eseguita come se elemento fosse incluso. Se l'istruzione *Next* viene incontrata prima dell'istruzione *For* corrispondente, verrà generato un errore.

Non è possibile utilizzare un'istruzione *For...Each...Next* con matrici di tipi definiti dall'utente perché una variabile *Variant* non può includere un tipo definito dall'utente.

Istruzione GoTo

Passa incondizionatamente a una riga specifica all'interno di una routine.

Sintassi

GoTo riga

L'argomento riga può essere una qualsiasi etichetta di riga o un qualsiasi numero di riga.

Osservazioni

L'istruzione *GoTo* consente di passare soltanto alle righe della routine in cui è inclusa.

Nota La presenza di troppe istruzioni *GoTo* può rendere difficile la lettura e il debug del codice. Quando possibile, è consigliabile utilizzare istruzioni di controllo strutturate (*Do...Loop*, *For...Next*, *If...Then...Else*, *Select Case*).

Istruzione GoSub...Return

In una routine, questa istruzione consente di passare a una subroutine e di tornare al punto di partenza.

Sintassi

GoSub riga

...

riga

...

Return

L'argomento riga può essere una qualsiasi etichetta di riga o un qualsiasi numero di riga.

Osservazioni

GoSub e *Return* possono essere utilizzate in un punto qualsiasi della routine, ma devono essere entrambe incluse nella stessa routine. Una subroutine può includere più istruzioni *Return*, ma la prima istruzione *Return* incontrata comporterà il ritorno del programma all'istruzione che segue l'istruzione *GoSub* più recente.

Nota Non è possibile entrare o uscire da routine Sub tramite l'istruzione *GoSub...Return*.

Suggerimento Una valida alternativa all'utilizzo dell'istruzione *GoSub...Return* è la creazione di routine separate richiamabili.

Istruzione Let

Assegna il valore di un'espressione a una variabile o a una proprietà.

Sintassi

[*Let*] nomevariabile = espressione

La sintassi dell'istruzione *Let* è composta dalle seguenti parti:

Parte	Descrizione
Let	<u>Facoltativa</u> . L'uso esplicito della parola chiave <i>Let</i> è una questione di stile; viene in genere omessa.
nomevariabile	<u>Obbligatoria</u> . Nome della variabile o della proprietà, espresso in base alle convenzioni di denominazione standard delle variabili.
espressione	<u>Obbligatoria</u> . Valore assegnato alla variabile o alla proprietà.

Osservazioni

Il valore di un'espressione può essere assegnato a una variabile o a una proprietà soltanto se il tipo di dati è compatibile con la variabile. Non è possibile assegnare un'espressione stringa alle variabili numeriche, né assegnare espressioni numeriche alle variabili stringa. In caso di assegnazione errata, in fase di compilazione verrà generato un errore.

Alle variabili di tipo *Variant* possono essere assegnate espressioni sia di tipo stringa che numeriche. Non sempre è possibile eseguire l'operazione inversa. A una variabile di tipo stringa può essere assegnato qualsiasi tipo di *Variant* a eccezione di *Null*, mentre a una variabile numerica può essere assegnata soltanto una variabile di tipo *Variant* che contiene un valore che può essere interpretato come numero. Utilizzare la funzione *IsNumeric* per determinare se il tipo di dati *Variant* può essere convertito in numero.

Attenzione Se si assegna un'espressione di un tipo di dati numerico a una variabile di un diverso tipo di dati numerico, il valore dell'espressione assumerà il tipo di dati della variabile risultante.

Le istruzioni **Let** possono essere utilizzate per assegnare una variabile record a un'altra soltanto se entrambe le variabili sono dello stesso tipo definito dall'utente. Utilizzare l'istruzione **LSet** per assegnare variabili record di tipi definiti dall'utente diversi. Utilizzare l'istruzione **Set** per assegnare riferimenti di oggetto a variabili.

Istruzione Set

Assegna un riferimento a un oggetto a una variabile o a una proprietà.

Sintassi

Set variabileoggetto = {[New] espressioneoggetto | Nothing}

La sintassi dell'istruzione **Set** è composta dalle seguenti parti:

Parte	Descrizione
variabileoggetto	<u>Obbligatoria</u> . Nome della variabile o proprietà, espresso in base alle convenzioni di denominazione standard delle variabili.
New	<u>Facoltativa</u> . New viene in genere utilizzata nelle dichiarazioni per consentire la creazione di oggetti in modo implicito. Quando si utilizza New in combinazione con Set , viene creata una nuova istanza della classe. Se variabileoggetto contiene un riferimento a un oggetto, il riferimento viene liberato quando ne viene assegnato uno nuovo. La parola chiave New non può essere utilizzata per dichiarare variabili di qualsiasi tipo di dati intrinseco e per creare oggetti dipendenti.
espressioneoggetto	<u>Obbligatoria</u> . Espressione composta dal nome di un oggetto, di un'altra variabile dichiarata dello stesso tipo di oggetto, oppure una funzione o un metodo che restituisce un oggetto dello stesso tipo.
Nothing	<u>Facoltativa</u> . Interrompe il collegamento di variabileoggetto con qualsiasi oggetto specifico. Se si assegna Nothing a variabileoggetto, diventeranno disponibili tutte le risorse di sistema e memoria associate all'oggetto di riferimento precedente se non vi sono altre variabili che si riferiscono a tale oggetto.

Osservazioni

Per essere valido, variabileoggetto deve essere un tipo di oggetto conforme all'oggetto che gli è stato assegnato. Le istruzioni **Dim**, **Private**, **Public**, **ReDim** e **Static** dichiarano soltanto la variabile che si riferisce a un oggetto. Non esisterà un effettivo riferimento all'oggetto finché non si utilizzerà l'istruzione Set per assegnare un oggetto specifico.

Nell'esempio che segue, l'istruzione **Dim** viene utilizzata per dichiarare una matrice di tipo **Form1**. Non esiste in effetti alcuna istanza di **Form1**. L'istruzione Set assegna quindi i riferimenti alle nuove istanze di **Form1** alla variabile **myChildForms**. Questo codice potrebbe essere utilizzato, ad esempio, per creare form secondari in un'applicazione MDI.

```
Dim myChildForms(1 to 4) As Form1
Set myChildForms(1) = New Form1
Set myChildForms(2) = New Form1
Set myChildForms(3) = New Form1
Set myChildForms(4) = New Form1
```

In genere, se viene utilizzata l'istruzione **Set** per assegnare a una variabile un riferimento a un oggetto, per tale variabile non verrà creata nessuna copia dell'oggetto. Verrà invece creato un riferimento all'oggetto. Più variabili oggetto possono fare riferimento allo stesso oggetto. Dato che queste variabili sono dei riferimenti e non copie dell'oggetto, qualsiasi modifica dell'oggetto verrà apportata anche alle variabili che si riferiscono all'oggetto. Se si utilizza la parola chiave New nell'istruzione **Set**, tuttavia, viene di fatto creata un'istanza dell'oggetto.

Istruzione Time

Imposta l'ora di sistema.

Sintassi

Time = orario

L'argomento obbligatorio orario può essere qualsiasi espressione numerica o espressione stringa in una qualsiasi combinazione che rappresenti un orario.

Osservazioni

Se orario è una stringa, *Time* prova a convertirlo utilizzando i separatori di ora impostati nel sistema. Nel caso in cui non possa essere convertito in un'orario valido, verrà generato un errore.

Istruzione While...Wend

Esegue una serie di istruzioni finché la valutazione di una determinata condizione dà come risultato *True*.

Sintassi

While condizione

[istruzioni]

Wend

La sintassi dell'istruzione *While...Wend* è composta dalle seguenti parti:

Parte	Descrizione
condizione	<u>Obbligatoria</u> . Espressione numerica o espressione stringa che può dare come risultato <i>True</i> o <i>False</i> . Se condizione è <i>Null</i> , viene considerata <i>False</i> .
istruzioni	<u>Facoltativa</u> . Una o più istruzioni eseguite finché la condizione è <i>True</i> .

Osservazioni

Se condizione è *True*, verranno eseguite tutte le istruzioni fino all'istruzione *Wend*. Il controllo tornerà quindi all'istruzione *While* e condizione verrà analizzata nuovamente. Se condizione è ancora *True*, il processo verrà ripetuto. Se non è *True*, l'esecuzione riprenderà con l'istruzione successiva all'istruzione *Wend*.

I cicli *While...Wend* possono essere nidificati a qualsiasi livello. Ciascuna istruzione *Wend* corrisponderà all'istruzione *While* più recente.

Suggerimento L'istruzione *Do...Loop* consente un'esecuzione dei cicli più strutturata e flessibile.

Istruzione With

Esegue una serie di istruzioni su un singolo oggetto o su un tipo definito dall'utente.

Sintassi

```
With oggetto  
[istruzioni]  
End With
```

La sintassi dell'istruzione **With** è composta dalle seguenti parti:

Parte	Descrizione
oggetto	<u>Obbligatoria</u> . Nome di un oggetto o di un tipo definito dall'utente.
istruzioni	<u>Facoltativa</u> . Una o più istruzioni da eseguire su oggetto.

Osservazioni

L'istruzione **With** consente di eseguire una serie di istruzioni su un oggetto specificato senza riqualificare il nome dell'oggetto. Se, ad esempio, si desidera modificare una serie di proprietà diverse per un singolo oggetto, risulta più agevole inserire le istruzioni di assegnazione di proprietà all'interno della struttura di controllo **With** facendo riferimento all'oggetto una sola volta, invece di dover includere il riferimento per ogni assegnazione di proprietà. In questo esempio viene descritto l'utilizzo dell'istruzione **With** per assegnare valori a diverse proprietà dello stesso oggetto.

```
With MyLabel  
    .Height = 2000  
    .Width = 2000  
    .Caption = "This is MyLabel"  
End With
```

Nota Dopo aver immesso un blocco **With**, non sarà possibile modificare oggetto. Ne risulta che non è possibile utilizzare una sola istruzione **With** per più oggetti diversi.

È possibile nidificare le istruzioni **With** inserendo un blocco **With** all'interno di un altro. I membri dei blocchi **With** esterni sono tuttavia nascosti dai blocchi **With** interni ed è pertanto necessario inserire in un blocco interno **With** un riferimento completo all'oggetto per qualsiasi membro di un oggetto incluso in un blocco **With** esterno.

Importante Non eseguire salti procedurali verso l'interno o l'esterno di blocchi **With**. Se le istruzioni di un blocco **With** vengono eseguite, ma non viene eseguita l'istruzione **With** o **End With** possono verificarsi errori o funzionamenti imprevedibili.

Sez. F

Categoria Funzioni

FUNZIONI

[Asc](#)
[Chr](#)
[_Date](#)
[Hex](#)
[InputBox](#)
[Int \(e Fix\)](#)
[LCase](#)
[Left](#)
[Len](#)
[Ltrim \(Rtrim e Trim\)](#)
[Mid](#)
[MsgBox](#)
[Right](#)
[Shell](#)
[Str](#)
[_Time](#)
[Timer](#)
[Ucase](#)
[Val](#)

[Torna all'indice delle Categorie](#)

Funzione Asc

Restituisce un valore **Integer** che rappresenta il codice di carattere corrispondente alla prima lettera di una stringa.

Sintassi

Asc(stringa)

L'argomento obbligatorio stringa può essere una qualsiasi espressione stringa valida. Se stringa non include alcun carattere, verrà generato un errore di run-time.

Osservazioni

L'intervallo valido dei risultati è compreso fra 0 e 255 in sistemi non DBCS e fra -32768 e 32767 in sistemi DBCS.

Nota La funzione *AscB* è utilizzata con i dati byte contenuti in una stringa. Aniché restituire il codice del primo carattere, *AscB* restituisce il primo byte. La funzione *AscW* restituisce il codice di carattere Unicode ad eccezione delle piattaforme che non supportano Unicode. In questi casi, le funzioni sono identiche a quelle della funzione *Asc*.

Funzione Chr

Restituisce un valore **String** che contiene il carattere associato al codice di carattere specificato.

Sintassi

Chr(codicecarattere)

L'argomento obbligatorio codicecarattere è un valore **Long** che identifica un carattere.

Osservazioni

I numeri da 0 a 31 corrispondono ai codici ASCII standard non stampabili. La funzione *Chr*(10), ad esempio, restituisce un carattere di avanzamento riga. L'intervallo valido per codicecarattere è in genere compreso fra 0 e 255. Nei sistemi DBCS l'intervallo valido per codicecarattere è compreso fra -32768 e 65536.

Nota La funzione *ChrB* viene utilizzata con i dati byte contenuti in un valore **String**. Aniché restituire un carattere, composto da uno o due byte, la funzione *ChrB* restituisce sempre un solo byte. La funzione *ChrW* restituisce un valore **String** contenente il carattere Unicode tranne che nelle piattaforme che non supportano Unicode. In questo caso le funzioni sono identiche a quelle della funzione *Chr*.

Funzione Date

Restituisce un valore **Variant** (Date) che contiene la data corrente di sistema.

Sintassi

Date

Osservazioni

Per impostare la data di sistema, utilizzare l'istruzione **Date**.

Funzione Hex

Restituisce un valore *String* che rappresenta il valore esadecimale di un numero.

Sintassi

Hex(numero)

L'argomento obbligatorio numero è qualsiasi espressione numerica o espressione stringa valida.

Osservazioni

Se numero non è già un numero intero, verrà arrotondato al numero intero più prossimo prima di essere valutato.

Se numero è

Hex restituisce

Null

Null

Empty

Zero (0)

Qualsiasi altro numero

Fino a otto caratteri esadecimale

È possibile rappresentare direttamente i numeri esadecimale facendo precedere i numeri nell'intervallo appropriato da **&H**. Ad esempio, **&H10** rappresenta il decimale 16 in notazione esadecimale.

Funzione InputBox

Visualizza un messaggio in una finestra di dialogo, attendendo che l'utente immetta del testo o scelga un pulsante, quindi restituisce un valore **String** che include il contenuto della casella di testo.

Sintassi

InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])

La sintassi della funzione **InputBox** è composta dai seguenti argomenti predefiniti:

Parte	Descrizione
prompt	Obbligatoria. Espressione stringa che costituisce il messaggio visualizzato nella finestra di dialogo. La lunghezza massima di prompt è di circa 1024 caratteri e dipende dalla larghezza dei caratteri utilizzati. Se prompt è suddiviso su più righe, è possibile includere, tra ciascuna coppia di righe, un ritorno a capo (Chr(13)), un carattere di avanzamento riga (Chr(10)) o una sequenza ritorno a capo-avanzamento riga (Chr(13) & Chr(10)).
title	Facoltativa. Espressione stringa visualizzata nella barra del titolo della finestra di dialogo. Se l'argomento title viene omissso, nella barra del titolo verrà indicato il nome dell'applicazione.
default	Facoltativa. Espressione stringa visualizzata nella casella di testo come risposta predefinita quando non sono forniti altri input. Se default è omissso, verrà visualizzata una casella di testo vuota.
xpos	Facoltativa. Espressione numerica che specifica la distanza orizzontale in twip del bordo sinistro della finestra di dialogo dal margine sinistro dello schermo. Se xpos è omissso, la finestra di dialogo apparirà centrata in senso orizzontale.
ypos	Facoltativa. Espressione numerica che specifica la distanza verticale in twip del bordo superiore della finestra di dialogo dal margine superiore dello schermo. Se ypos è omissso, la finestra di dialogo verrà posizionata verticalmente a circa un terzo dal margine superiore dello schermo.
helpfile	Facoltativa. Espressione stringa che identifica il file della Guida da utilizzare per ottenere informazioni di Guida sensibili al contesto relative alla finestra di dialogo. Se si indica helpfile è necessario indicare anche context .
context	Facoltativa. Espressione numerica che corrisponde al numero del contesto della Guida assegnato all'argomento corrispondente della Guida. Se si indica context è necessario indicare anche helpfile .

Osservazioni

Se vengono inclusi sia l'argomento **helpfile** che l'argomento **context** gli utenti potranno premere F1 per visualizzare l'argomento della Guida corrispondente a **context**. In alcune applicazioni host, ad esempio Microsoft Excel, verrà inoltre aggiunto automaticamente il pulsante della Guida (?) nella finestra di dialogo. Se l'utente sceglie **OK** o preme **INVIO**, la funzione **InputBox** restituirà il contenuto della casella di testo. Se l'utente sceglie il pulsante **Annulla**, la funzione restituirà una stringa di lunghezza zero ("").

Nota: Se si desidera specificare altri argomenti, oltre al primo predefinito, è necessario utilizzare **InputBox** in un'espressione. Se si desidera omettere alcuni argomenti di posizione, è necessario aggiungere la virgola di delimitazione corrispondente.

Funzioni Int e Fix

Restituisce un valore dello stesso tipo passato alla funzione che contiene la parte intera di un numero.

Sintassi

Int(numero)

Fix(numero)

L'argomento obbligatorio numero può essere un valore *Double* o una qualsiasi espressione numerica valida. Se numero contiene *Null*, il risultato sarà *Null*.

Osservazioni

Le funzioni *Int* e *Fix* eliminano la parte frazionaria di numero e ne restituiscono il valore intero.

La differenza tra le funzioni *Int* e *Fix* risiede nel fatto che se numero è negativo, *Int* restituisce il primo intero negativo minore o uguale a numero, mentre *Fix* restituisce il primo intero negativo maggiore o uguale a numero. *Int* ad esempio, converte -8,4 in -9, mentre *Fix* converte -8,4 in -8.

Fix(numero) equivale a:

Sgn(numero) * *Int*(*Abs*(numero))

Funzione LCase

Restituisce un valore *String* convertito in minuscole.

Sintassi

LCase(stringa)

L'argomento obbligatorio stringa è una qualsiasi espressione stringa valida. Se stringa contiene *Null*, verrà restituito *Null*.

Osservazioni

Solo le lettere maiuscole vengono convertite in minuscole; tutte le lettere minuscole e i caratteri non letterali rimangono invariati.

Funzione Left

Restituisce un valore *Variant* (String) che contiene un numero specificato di caratteri di una stringa a partire da sinistra.

Sintassi

Left(string, length)

La sintassi della funzione *Left* è composta dai seguenti argomenti predefiniti:

Parte	Descrizione
string	Obbligatoria. Espressione stringa dalla quale vengono restituiti i caratteri situati all'estrema sinistra. Se string contiene <i>Null</i> verrà restituito Null.
length	Obbligatoria; <i>Variant</i> (Long). Espressione numerica che indica quanti caratteri devono essere restituiti. Se è 0, verrà restituita una stringa di lunghezza zero (""). Se è maggiore o uguale al numero di caratteri presenti in <i>string</i> , verrà restituita l'intera stringa.

Osservazioni

Per determinare il numero di caratteri in string, utilizzare la funzione *Len*.

Nota Utilizzare la funzione *LeftB* con i dati byte contenuti in una stringa. Aniché specificare il numero di caratteri da restituire, length specifica il numero di byte.

Funzione Len

Restituisce un valore **Long** che contiene il numero di caratteri presenti in una stringa o il numero di byte necessari per memorizzare una variabile.

Sintassi

Len(stringa | nomevariabile)

La sintassi della funzione **Len** è composta dalle seguenti parti:

Parte	Descrizione
stringa	Qualsiasi espressione stringa valida. Se stringa contiene Null, verrà restituito Null.
nomevariabile	Qualsiasi nome di variabile valido. Se nomevariabile contiene Null, verrà restituito Null. Se l'argomento nomevariabile è Variant, Len lo considera come String e restituisce sempre il numero di caratteri contenuti.

Osservazioni

È necessario specificare uno solo dei due argomenti possibili. Per i tipi definiti dall'utente **Len** restituisce le stesse dimensioni che verranno scritte su file.

Nota Utilizzare la funzione **LenB** con i dati byte contenuti in una stringa. Anziché restituire il numero di caratteri di una stringa, **LenB** restituisce il numero di byte utilizzati per rappresentare la stringa. Per i tipi definiti dall'utente **LenB** restituisce le dimensioni in memoria, compresi i riempimenti tra gli elementi.

Nota È possibile che la funzione **Len** non consenta di determinare il numero effettivo di byte richiesti per la memorizzazione se utilizzata con stringhe di lunghezza variabile in tipi di dati definiti dall'utente.

Funzioni LTrim, RTrim e Trim

Restituisce un valore **Variant (String)** contenente una copia di una stringa senza spazi iniziali (**LTrim**), spazi finali (**RTrim**) o senza spazi iniziali e finali (**Trim**).

Sintassi

LTrim(stringa)

RTrim(stringa)

Trim(stringa)

L'argomento obbligatorio stringa è qualsiasi espressione stringa valida. Se stringa contiene **Null**, verrà restituito **Null**.

Funzione Mid

Restituisce un valore *Variant* (String) che contiene un numero specificato di caratteri di una stringa.

Sintassi

Mid(string, start[, length])

La sintassi della funzione **Mid** è composta dai seguenti argomenti predefiniti:

Parte	Descrizione
string	Obbligatoria. Espressione stringa dalla quale vengono restituiti i caratteri. Se <i>string</i> contiene <i>Null</i> , verrà restituito Null.
start	Obbligatoria; <i>Long</i> . Posizione di un carattere in string dalla quale inizia la parte che deve essere prelevata. Se <i>start</i> è superiore al numero di caratteri in string, <i>Mid</i> restituisce una stringa di lunghezza zero ("").
length	Facoltativa; <i>Variant</i> (Long). Numero di caratteri da restituire. Se omissso, o se nel testo vi sono meno caratteri che in <i>length</i> (compreso il carattere di start), vengono restituiti tutti i caratteri dalla posizione start alla fine della stringa.

Osservazioni

Per determinare il numero di caratteri in string, utilizzare la funzione *Len*.

Nota Utilizzare la funzione *MidB* con i dati byte contenuti in una stringa. Anziché specificare il numero di caratteri, gli argomenti specificano il numero di byte.

Funzione MsgBox

Visualizza un messaggio in una finestra di dialogo e attende che l'utente scelga un pulsante, quindi restituisce un valore *Integer* che indica quale pulsante l'utente ha scelto.

Sintassi

MsgBox(prompt[, buttons] [, title] [, helpfile, context])

La sintassi della funzione *MsgBox* è composta dai seguenti argomenti predefiniti:

Parte	Descrizione
prompt	Obbligatoria. Espressione stringa che costituisce il messaggio visualizzato nella finestra di dialogo. La lunghezza massima di prompt è di circa 1024 caratteri e dipende dalla larghezza dei caratteri utilizzati. Se <i>prompt</i> è suddiviso su più righe, è possibile includere, tra ciascuna coppia di righe, un ritorno a capo (<i>Chr(13)</i>), un carattere di avanzamento riga (<i>Chr(10)</i>) o una sequenza ritorno a capo-avanzamento riga (<i>Chr(13) & Chr(10)</i>).
buttons	Facoltativa. Espressione numerica che indica la somma dei valori che specificano il numero e il tipo di pulsante da visualizzare, lo stile di icona da utilizzare, il tipo di pulsante da utilizzare come impostazione predefinita e la modalità della finestra di messaggio. Se omesso il valore predefinito di <i>buttons</i> è 0 .
title	Facoltativa. Espressione stringa visualizzata nella barra del titolo della finestra di dialogo. Se l'argomento <i>title</i> viene omesso, nella barra del titolo verrà indicato il nome dell'applicazione.
helpfile	Facoltativa. Espressione stringa che identifica il file della Guida da utilizzare per ottenere informazioni di Guida sensibili al contesto relative alla finestra di dialogo. Se si indica <i>helpfile</i> è necessario indicare anche <i>context</i> .
context	Facoltativa. Espressione numerica che corrisponde al numero del contesto della Guida assegnato all'argomento corrispondente della Guida. Se si indica <i>context</i> è necessario indicare anche <i>helpfile</i> .

Impostazioni

Le possibili impostazioni dell'argomento *buttons* sono:

Costante	Valore	Descrizione
<i>VbOKOnly</i>	0	Visualizza solo il pulsante <i>OK</i> .
<i>VbOKCancel</i>	1	Visualizza i pulsanti <i>OK</i> e <i>Annulla</i> .
<i>VbAbortRetryIgnore</i>	2	Visualizza i pulsanti <i>Termina</i> , <i>Riprova</i> , e <i>Ignora</i> .
<i>VbYesNoCancel</i>	3	Visualizza i pulsanti <i>Sì</i> , <i>No</i> e <i>Annulla</i> .
<i>VbYesNo</i>	4	Visualizza i pulsanti <i>Sì</i> e <i>No</i> .
<i>VbRetryCancel</i>	5	Visualizza i pulsanti <i>Riprova</i> e <i>Annulla</i> .
<i>VbCritical</i>	16	Visualizza l'icona di messaggio critico.
<i>VbQuestion</i>	32	Visualizza l'icona di richiesta di avviso.
<i>VbExclamation</i>	48	Visualizza l'icona di messaggio di avviso.
<i>VbInformation</i>	64	Visualizza l'icona di messaggio di informazione.
<i>VbDefaultButton1</i>	0	Il primo pulsante è il predefinito.
<i>VbDefaultButton2</i>	256	Il secondo pulsante è il predefinito.
<i>VbDefaultButton3</i>	512	Il terzo pulsante è il predefinito.
<i>VbDefaultButton4</i>	768	Il quarto pulsante è il predefinito.
<i>VbApplicationModal</i>	0	L'utente deve rispondere alla finestra di messaggio prima di poter continuare a lavorare nell'applicazione corrente.
<i>VbSystemModal</i>	4096	Finestra di messaggio a scelta obbligatoria nel sistema. Tutte le applicazioni vengono sospese fino a quando l'utente non risponde alla finestra di messaggio.

Il primo gruppo di valori (0-5) descrive il numero e il tipo dei pulsanti visualizzati nella finestra di messaggio. Il secondo gruppo (16, 32, 48, 64) descrive lo stile dell'icona. Il terzo gruppo (0, 256, 512, 768) determina il pulsante predefinito e il quarto gruppo (0, 4096) determina se la finestra di messaggio è a scelta obbligatoria nell'applicazione o nel sistema. Quando si sommano dei

numeri per la creazione di un valore finale dell'argomento **buttons**, utilizzare soltanto un numero per ciascun gruppo.

Nota Di seguito sono elencate le costanti specificate o riconosciute da **Visual Basic**, Applications Edition utilizzabili in qualsiasi punto del codice in sostituzione dei valori effettivi.

Valori restituiti

Costante	Valore	Descrizione
<i>vbOK</i>	1	OK
<i>vbCancel</i>	2	Annulla
<i>vbAbort</i>	3	Termina
<i>vbRetry</i>	4	Riprova
<i>vbIgnore</i>	5	Ignora
<i>vbYes</i>	6	Sì
<i>vbNo</i>	7	No

Osservazioni

Se vengono inclusi sia l'argomento **helpfile** che l'argomento **context** gli utenti potranno premere F1 per visualizzare l'argomento della Guida corrispondente a **context**. In alcune applicazioni host, ad esempio Microsoft Excel, verrà inoltre aggiunto automaticamente il pulsante della Guida (?) nella finestra di dialogo.

Se nella finestra di dialogo è visualizzato un pulsante **Annulla**, il tasto **ESC** avrà la stessa funzione del pulsante **Annulla**. Se nella finestra di dialogo è incluso un pulsante della Guida (?), significa che per la finestra di dialogo è disponibile la Guida sensibile al contesto. Non verrà restituito comunque nessun valore, se non dopo aver scelto almeno uno degli altri pulsanti.

Nota: Se si desidera specificare altri argomenti, oltre al primo predefinito, è necessario utilizzare **MsgBox** in un'espressione. Se si desidera omettere alcuni argomenti di posizione, è necessario aggiungere la virgola di delimitazione corrispondente.

Funzione Right

Restituisce un valore **Variant** (String) che contiene un numero specificato di caratteri di una stringa a partire da destra.

Sintassi

Right(string, length)

La sintassi della funzione **Right** è composta dai seguenti argomenti predefiniti:

Parte	Descrizione
string	Obbligatoria. Espressione stringa dalla quale vengono restituiti i caratteri situati all'estrema destra. Se string contiene Null , verrà restituito Null .
length	Obbligatoria; Variant (Long) . Espressione numerica che indica quanti caratteri devono essere restituiti. Se pari a 0 viene restituita una stringa di lunghezza zero (""). Se maggiore o uguale rispetto al numero di caratteri in string, viene restituita l'intera stringa.

Osservazioni

Per determinare il numero di caratteri in **string** utilizzare la funzione **Len**.

Nota Utilizzare la funzione **RightB** con i dati byte contenuti in una stringa. Anziché specificare il numero di caratteri che devono essere restituiti, **length** specifica il numero di byte.

Funzione Shell

Avvia un programma eseguibile e restituisce un valore *Variant* (Double) che indica l'ID del task del programma in caso di esito positivo oppure zero in caso di esito negativo.

Sintassi

Shell(pathname[,windowstyle])

La sintassi della funzione *Shell* è composta dai seguenti argomenti predefiniti:

Parte	Descrizione
pathname	Obbligatoria; <i>Variant</i> (String). Nome del programma da eseguire e tutti gli argomenti o parametri della riga di comando. Può includere la directory o cartella e l'unità.
windowstyle	Facoltativa. <i>Variant</i> (Integer) corrispondente allo stile della finestra nella quale verrà eseguito il programma. Se <i>windowstyle</i> viene omesso, il programma verrà aperto come programma attivo ridotto a icona.

Per l'argomento predefinito *windowstyle* sono disponibili i seguenti valori:

Costante	Valore	Descrizione
<i>vbHide</i>	0	La finestra è nascosta e l'attivazione viene passata alla finestra nascosta.
<i>vbNormalFocus</i>	1	La finestra è attivata e vengono ripristinate la dimensione e la posizione originali.
<i>vbMinimizedFocus</i>	2	La finestra è ridotta a icona e attivata.
<i>vbMaximizedFocus</i>	3	La finestra è ingrandita e attivata.
<i>vbNormalNoFocus</i>	4	Vengono ripristinate le dimensioni e posizione precedenti della finestra. La finestra attiva resta attiva.
<i>vbMinimizedNoFocus</i>	6	La finestra è ridotta a icona. La finestra attiva resta attiva.

Osservazioni

Se la funzione *Shell* esegue in modo corretto il file indicato, restituirà l'ID del task del programma avviato. L'ID del task è un numero univoco, che identifica il programma in esecuzione. Se la funzione *Shell* non riesce ad avviare il programma specificato, verrà generato un errore.

Nota La funzione *Shell* consente di eseguire altri programmi in modo asincrono. Ciò significa che non è necessario attendere la fine dell'esecuzione di un programma avviato con *Shell* prima che possano essere eseguite le istruzioni che seguono la funzione *Shell* nell'applicazione.

Funzione Str

Restituisce una rappresentazione *Variant* (String) di un numero.

Sintassi

Str(numero)

L'argomento obbligatorio numero è un valore *Long* che contiene una qualsiasi espressione numerica valida.

Osservazioni

Quando i numeri vengono convertiti in stringhe, viene sempre riservato uno spazio iniziale al segno di numero. Se numero è positivo, la stringa restituita dalla funzione contiene uno spazio iniziale, mentre il segno più è sottinteso.

Utilizzare la funzione *Format* per convertire valori numerici che si desidera vengano formattati come data, ora o valuta o in altri formati definiti dall'utente. A differenza di *Str*, la funzione *Format* non prevede uno spazio iniziale per il segno di numero.

Nota La funzione *Str* riconosce solo il punto (.) come separatore decimale valido. Se è necessario utilizzare separatori decimali diversi, ad esempio in applicazioni internazionali, utilizzare *CStr* per convertire un numero in stringa.

Funzione Time

Restituisce un valore *Variant* (Date) che indica l'ora di sistema corrente.

Sintassi

Time

Osservazioni

Per impostare l'ora di sistema, utilizzare l'istruzione *Time*.

Funzione Timer

Restituisce un valore Single che rappresenta il numero di secondi trascorsi dalla mezzanotte.

Sintassi

Timer

Funzione Ucase

Restituisce un valore *Variant* (String) che contiene la stringa specificata, convertita in maiuscole.

Sintassi

UCase(stringa)

L'argomento obbligatorio stringa è qualsiasi espressione stringa valida. Se stringa contiene *Null*, verrà restituito *Null*.

Osservazioni

Solo le lettere minuscole vengono convertite in maiuscole. Tutte le lettere maiuscole e i caratteri non letterali rimangono invariati.

Funzione Val

Restituisce i numeri inclusi in una stringa sotto forma di valore numerico di tipo appropriato.

Sintassi

Val(stringa)

L'argomento obbligatorio stringa può essere una qualsiasi espressione stringa valida.

Osservazioni

La funzione *Val* interrompe la lettura della stringa in corrispondenza del primo carattere non riconosciuto come parte di un numero. I simboli e i caratteri spesso riconosciuti come facenti parte di un valore numerico, quali il segno di valuta e le virgole, non vengono riconosciuti. La funzione riconosce tuttavia i prefissi di radice **&O** (per ottale) e **&H** (per esadecimale). Gli spazi, le tabulazioni e i caratteri di avanzamento riga vengono eliminati dall'argomento.

In questo esempio la funzione restituirà il valore 1615198:

```
Val(" 1615 198th Street N.E.")
```

Nel seguente codice *Val* restituisce il valore decimale -1 per il valore esadecimale visualizzato:

```
Val("&HFFFF")
```

Nota La funzione *Val* riconosce solo il punto (.) come separatore decimale valido. Se si utilizzano separatori diversi (ad esempio in applicazioni internazionali) per convertire una stringa in numero è consigliabile utilizzare la funzione *Cdbl*.

Sez. G

Categoria Metodi

METODI

[Refresh](#)
[SetFocus](#)

[Torna all'indice delle Categorie](#)

Metodo Refresh

Forza l'aggiornamento completo di un *form* o di un controllo.

Sintassi

oggetto.*Refresh*

Il segnaposto oggetto rappresenta un'espressione oggetto che definisce un oggetto dell'elenco "Si applica a".

Osservazioni

Utilizzare il metodo *Refresh* se si desidera:

- Visualizzare completamente un *form* mentre viene caricato un altro *form*.
- Aggiornare il contenuto di una casella di riepilogo relativa al file system, come un controllo *FileListBox*.
- Aggiornare le strutture dati di un controllo *Data*.

Il metodo *Refresh* non può essere utilizzato su *form MDI*, mentre può essere utilizzato su *form MDI secondari*. Non è possibile utilizzare il metodo *Refresh* su controlli *Menu* o *Timer*.

In genere, il disegno di un *form* o di un controllo viene gestito automaticamente quando non si stanno verificando eventi. È possibile, tuttavia, che si desideri aggiornare immediatamente il *form* o il controllo. Se si utilizza, ad esempio, una casella di riepilogo dei file, delle directory o delle unità per visualizzare lo stato corrente della struttura delle directory, è possibile utilizzare il metodo *Refresh* per aggiornare l'elenco a ogni modifica della struttura delle directory.

È possibile utilizzare il metodo *Refresh* su un controllo *Data* per aprire o riaprire il database (se sono state modificate le impostazioni delle proprietà *DatabaseName*, *ReadOnly*, *Exclusive* o *Connect*) e ricreare il dynaset nella proprietà *Recordset* del controllo.

Metodo SetFocus

Sposta lo stato attivo sul controllo o sul *form* specificato.

Sintassi

oggetto.*SetFocus*

Il segnaposto oggetto rappresenta un'espressione oggetto che definisce un oggetto dell'elenco "Si applica a".

Osservazioni

È necessario che l'oggetto sia un oggetto *Form*, un oggetto *MDIForm* o un controllo che può ricevere lo stato attivo. Dopo aver richiamato il metodo *SetFocus*, tutti gli input dell'utente saranno diretti al *form* o al controllo specificato.

Lo stato attivo può essere spostato solo su un *form* o un controllo visibile. Poiché il *form* e i corrispondenti controlli non sono visibili fino al termine dell'evento *Load* del *form*, non è possibile utilizzare il metodo *SetFocus* per spostare lo stato attivo sul *form* mentre questo viene caricato, a meno che prima non sia stato utilizzato il metodo *Show* per visualizzare il *form* prima del termine della routine dell'evento *Form_Load*.

Non è possibile inoltre spostare lo stato attivo su un *form* o controllo se la proprietà *Enabled* è impostata su *False*. Se la proprietà *Enabled* è stata impostata su *False* in fase di progettazione, sarà necessario impostarla prima su *True* in modo che sia possibile spostare lo stato attivo utilizzando il metodo *SetFocus*.

Sez. H

Categoria Eventi

EVENTI

[Change](#)
[Click](#)
[DbClick](#)
[Load](#)
[MouseDown \(e MouseUp\)](#)
[MouseMove](#)
[Timer](#)
[Unload](#)

[Torna all'indice delle Categorie](#)

Evento Change

Indica che il contenuto di un controllo è stato modificato. Il momento e le modalità con cui questo evento viene generato dipendono dal controllo, come indicato di seguito:

- **ComboBox**: modifica il testo nella casella di testo del controllo. L'evento viene generato solo se la proprietà **Style** è impostata su **0** (casella combinata a discesa) o **1** (casella combinata semplice) e l'utente modifica il testo oppure viene modificata l'impostazione della proprietà **Text** nel codice.
- **DirListBox**: cambia la directory selezionata. L'evento viene generato quando si fa doppio clic su una directory diversa o quando viene modificata l'impostazione della proprietà **Path** nel codice.
- **DriveListBox**: cambia l'unità selezionata. L'evento viene generato quando viene selezionata un'unità diversa o modificata l'impostazione della proprietà **Drive** nel codice.
- **HScrollBar** e **VScrollBar** (barre di scorrimento orizzontale e verticale): spostano la casella di scorrimento nella barra di scorrimento. L'evento viene generato quando si esegue uno scorrimento o si modifica l'impostazione della proprietà **Value** nel codice.
- **Label**: modifica il contenuto di **Label**. L'evento viene generato quando viene eseguito un aggiornamento da un collegamento **DDE** o quando si modifica l'impostazione della proprietà **Caption** nel codice.
- **PictureBox**: modifica il contenuto di **PictureBox**. L'evento viene generato quando viene eseguito un aggiornamento da un collegamento **DDE** o quando si modifica l'impostazione della proprietà **Picture** nel codice.
- **TextBox**: modifica il contenuto della casella di testo. L'evento viene generato quando viene eseguito un aggiornamento mediante un collegamento **DDE**, quando l'utente modifica il testo o quando si modifica l'impostazione della proprietà **Text** nel codice.

Sintassi

Private Sub oggetto_ **Change**([index As Integer])

La sintassi dell'evento **Change** è composta dalle seguenti parti:

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica".
index	Intero che identifica in modo univoco un controllo se incluso in una matrice di controlli.

Osservazioni

La routine dell'evento **Change** consente di sincronizzare o coordinare la visualizzazione dei dati in più controlli. È possibile, ad esempio, utilizzare la routine dell'evento **Change** di una barra di scorrimento per aggiornare l'impostazione della proprietà **Value** della barra di scorrimento in un controllo **TextBox**. La routine può inoltre essere utilizzata per visualizzare i dati e le formule in una determinata area di lavoro e i risultati in un'area diversa.

Le routine di eventi **Change** risultano utili per eseguire l'aggiornamento delle proprietà nei controlli del file system, ovvero **DirListBox**, **DriveListBox** e **FileListBox**. È possibile, ad esempio, aggiornare l'impostazione della proprietà **Path** per il controllo **DirListBox** dopo aver modificato l'impostazione della proprietà **Drive** del controllo **DriveListBox**.

Nota Una routine dell'evento **Change** può a volte causare un evento a catena. Ciò si verifica quando l'evento **Change** del controllo altera il contenuto del controllo impostando nel codice una proprietà che determina il valore del controllo, come ad esempio la proprietà **Text** per il controllo **TextBox**. Per impedire che venga generato un evento a catena:

- Se possibile, evitare di scrivere routine di eventi **Change** per controlli che modificano il contenuto del controllo. Se è necessario scriverla, impostare un **flag** che impedisca ulteriori modifiche mentre è in corso la modifica corrente.
- Evitare di creare due o più controlli con routine di eventi **Change** che interferiscono l'uno con l'altro, ad esempio due controlli **TextBox** che si aggiornano a vicenda durante l'esecuzione dei relativi eventi **Change**.
- Evitare di utilizzare una funzione o istruzione **MsgBox** in questo evento per i controlli **HScrollBar** e **VScrollBar**.

Evento Click

Viene generato quando si preme e si rilascia un pulsante del mouse su un oggetto oppure quando si modifica il valore di un controllo.

In un oggetto **Form**, questo evento viene generato quando si fa clic su un'area vuota o su un controllo disattivato.

In un controllo l'evento viene generato quando l'utente...:

- ...fa clic su un controllo con il pulsante sinistro o destro del mouse. In un controllo **CheckBox**, **CommandButton**, **ListBox** o **OptionButton**, l'evento viene generato solo quando si fa clic con il pulsante sinistro.
- ...seleziona un elemento in un controllo **ComboBox** o **ListBox** premendo un tasto di DIREZIONE o facendo clic con il pulsante del mouse.
- ...preme la BARRA SPAZIATRICE quando lo stato attivo si trova su un controllo **CommandButton**, **OptionButton** o **CheckBox**.
- ...preme INVIO quando la proprietà **Default** del controllo **CommandButton** di un form è impostata su **True**.
- ...preme ESC in un **form** che contiene un pulsante Annulla, ovvero un controllo **CommandButton** con la proprietà **Cancel** impostata su **True**.
- ...preme il tasto di scelta di un controllo, ad esempio ALT+V, in un controllo **CommandButton** la cui didascalìa è "&Vai".

L'evento **Click** può inoltre essere generato intervenendo nel codice in uno dei seguenti modi:

- Se si imposta la proprietà **Value** di un controllo **CommandButton** su **True**.
- Se si imposta la proprietà **Value** di un controllo **OptionButton** su **True**.
- Se si modifica l'impostazione della proprietà **Value** di un controllo **CheckBox**.

Sintassi

```
Private Sub Form_Click()
```

```
Private Sub oggetto_Click([index As Integer])
```

La sintassi dell'evento **Click** è composta dalle seguenti parti:

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica".
index	Intero che identifica in modo univoco un controllo se questo è incluso in una matrice di controlli.

Osservazioni

In genere, si associa la routine dell'evento **Click** a un controllo **CommandButton**, a un oggetto **Menu** o a un controllo **PictureBox** per eseguire comandi o operazioni analoghe ai comandi. Per gli altri controlli, utilizzare questo evento per generare azioni in risposta alle modifiche apportate al controllo.

Per verificare lo stato di un controllo dal codice, è possibile utilizzare la proprietà **Value**. Quando si fa clic su un controllo, oltre all'evento **Click** vengono generati anche gli eventi **MouseDown** e **MouseUp**. L'ordine in cui questi eventi vengono generati varia a seconda del controllo. Ad esempio, per i controlli **ListBox** e **CommandButton**, gli eventi vengono generati nell'ordine **MouseDown**, **Click**, **MouseUp**, mentre per i controlli **FileListBox**, **Label** o **PictureBox**, gli eventi vengono generati nell'ordine **MouseDown**, **MouseUp** e **Click**. Se si aggiungono routine di eventi a questi eventi correlati, verificare che le relative azioni non creino conflitti. Se l'ordine degli eventi è importante nell'applicazione, sarà necessario provare il controllo per stabilire l'ordine corretto degli eventi.

Nota Per distinguere tra pulsante sinistro, destro e centrale del mouse, utilizzare gli eventi **MouseDown** e **MouseUp**.

Se l'evento **Click** contiene del codice, l'evento **DblClick** non verrà generato poiché l'evento **Click** è il primo dei due ad essere generato. Di conseguenza, il clic del mouse viene intercettato dall'evento **Click** e l'evento **DblClick** non viene generato.

Evento DbClick

Viene generato quando si preme e si rilascia due volte in rapida successione un pulsante del mouse puntato su un oggetto.

Per un form, l'evento **DbClick** viene generato quando si fa doppio clic su un controllo disattivato o su un'area vuota di un form. Per un controllo, l'evento **DbClick** viene generato quando l'utente:

- Fa doppio clic sul controllo con il pulsante sinistro del mouse.
- Fa doppio clic su un elemento di un controllo **ComboBox** la cui proprietà **Style** è impostata su **1** (casella combinata semplice) o di un controllo **FileListBox**, **ListBox**, **DBCombo** o **DBList**.

Sintassi

Private Sub Form_**DbClick** ()

Private Sub oggetto_**DbClick** (index As Integer)

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
index	Identifica il controllo se contenuto in una matrice di controlli.

Osservazioni

L'argomento **index** identifica in modo univoco un controllo se contenuto in una matrice di controlli. È possibile utilizzare una routine dell'evento **DbClick** per un'azione implicita, quale un doppio clic su un'icona per aprire una finestra o un documento. Questo tipo di routine consente inoltre di eseguire più passaggi in una singola azione, ad esempio un doppio clic per selezionare un elemento in una casella di riepilogo e per chiudere la finestra di dialogo.

Per creare queste scelte rapide in Visual Basic, è possibile utilizzare una routine dell'evento **DbClick** per una casella di riepilogo o una casella di riepilogo dei file insieme a un pulsante predefinito, ovvero un controllo **CommandButton** con la proprietà **Default** impostata su **True**. Per completare la routine dell'evento **DbClick** per la casella di riepilogo, è sufficiente richiamare l'evento **Click** del pulsante predefinito.

Per gli oggetti che ricevono gli eventi Mouse, gli eventi vengono generati con la sequenza **MouseDown**, **MouseUp**, **Click**, **DbClick** e **MouseUp**.

Se l'evento **DbClick** non viene generato entro il limite di tempo previsto dal sistema per il doppio clic, l'oggetto riconoscerà un altro evento **Click**. Il limite di tempo per il doppio clic è variabile in quanto è possibile impostare la velocità del doppio clic nel Pannello di controllo. Durante l'associazione delle routine per questi eventi correlati, verificare che le azioni corrispondenti non siano in conflitto. I controlli che non riconoscono gli eventi **DbClick** riconoscono due clic anziché l'evento **DbClick**.

Nota Per differenziare i pulsanti del mouse sinistro, destro e centrale, utilizzare gli eventi **MouseDown** e **MouseUp**.

Se l'evento **Click** contiene del codice, l'evento **DbClick** non verrà generato.

Evento Load

Viene generato quando si carica un form. Nel caso di un form di avvio, viene generato quando si avvia un'applicazione tramite un'istruzione **Load** o un riferimento a proprietà e controlli di un form non caricato.

Sintassi

```
Private Sub Form_Load()  
Private Sub MDIForm_Load()
```

Osservazioni

La routine dell'evento **Load** viene in genere utilizzata per includere il codice di inizializzazione di un form, ad esempio del codice che specifichi le impostazioni predefinite dei controlli, indichi il contenuto da caricare nei controlli **ComboBox** o **ListBox** e inizializzi le variabili a livello di form.

L'evento **Load** viene generato dopo l'evento **Initialize**.

Quando nel codice si fa riferimento a una proprietà di un form non caricato, il form verrà automaticamente caricato. Non viene tuttavia reso automaticamente visibile a meno che la proprietà **MDIChild** non sia impostata su **True**. Nel caso sia caricato solo il form MDI secondario e non l'oggetto **MDIForm**, entrambi verranno automaticamente caricati e resi visibili. Altri form verranno visualizzati solo quando si specifica il metodo **Show** o si imposta la proprietà **Visible** su **True**.

Il codice dell'evento **Load** dell'oggetto **MDIForm** riportato di seguito carica automaticamente un form secondario MDI, nell'ipotesi che la proprietà **MDIChild** di Form1 sia impostata su **True**.

```
Dim NewForm As New Form1  
NewForm.Caption = "Nuovo form" ' Carica il form per riferimento.
```

Dato che tutti i form secondari diventano visibili al momento del caricamento, il riferimento alla proprietà **Caption** carica il form e lo rende visibile.

Nota Quando si creano routine per eventi correlati, come ad esempio **Activate**, **GotFocus**, **Paint** e **Resize**, verificare che le relative azioni non siano in conflitto e che non causino eventi ricorsivi.

Eventi MouseDown e MouseUp

Vengono generati quando si preme (*MouseDown*) o si rilascia (*MouseUp*) un pulsante del mouse.

Sintassi

Private Sub Form_ *MouseDown*(button As Integer, shift As Integer, x As Single, y As Single)
Private Sub MDIForm_ *MouseDown*(button As Integer, shift As Integer, x As Single, y As Single)
Private Sub oggetto_ *MouseDown*([index As Integer,]button As Integer, shift As Integer, x As Single, y As Single)

Private Sub Form_ *MouseUp*(button As Integer, shift As Integer, x As Single, y As Single)
Private Sub MDIForm_ *MouseUp*(button As Integer, shift As Integer, x As Single, y As Single)
Private Sub oggetto_ *MouseUp*([index As Integer,]button As Integer, shift As Integer, x As Single, y As Single)

La sintassi degli eventi *MouseDown* e *MouseUp* è composta dalle seguenti parti:

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
index	Intero che identifica in modo univoco un controllo se questo è incluso in una matrice di controlli.
button	Intero che identifica il pulsante che è stato premuto (<i>MouseDown</i>) o rilasciato (<i>MouseUp</i>) generando quindi l'evento. L'argomento pulsante è un campo bit in cui bit 0 corrisponde al pulsante sinistro, bit 1 al pulsante destro e bit 2 al pulsante centrale. Questi bit corrispondono rispettivamente ai valori 1, 2 e 4. Viene impostato un solo bit ad indicare il pulsante che ha generato l'evento.
shift	Intero che corrisponde allo stato dei tasti MAIUSC, CTRL e ALT quando si preme o si rilascia il pulsante specificato dall'argomento <i>button</i> . Viene impostato un bit se il tasto è premuto. L'argomento <i>shift</i> è un campo bit in cui i bit meno significativi corrispondono a MAIUSC (bit 0), CTRL (bit 1) e ALT (bit 2). Questi bit corrispondono rispettivamente ai valori 1, 2 e 4. L'argomento <i>shift</i> indica lo stato dei tasti. È possibile impostare tutti i bit, uno solo o nessuno a seconda che si desideri indicare che tutti, uno solo o nessuno dei tasti viene premuto. Per indicare, ad esempio, che vengono premuti sia CTRL che ALT, si dovrà impostare l'argomento <i>shift</i> su 6.
x, y	Numero che specifica la posizione corrente del puntatore del mouse. I valori x e y vengono sempre espressi in relazione al sistema di coordinate definito mediante le proprietà <i>ScaleHeight</i> , <i>ScaleWidth</i> , <i>ScaleLeft</i> e <i>ScaleTop</i> dell'oggetto.

Osservazioni

Le routine di eventi *MouseDown* e *MouseUp* consentono di indicare l'azione che si verifica quando viene premuto o rilasciato un determinato pulsante del mouse. A differenza degli eventi *Click* e *DblClick*, con *MouseDown* e *MouseUp* è possibile distinguere tra pulsante destro, sinistro e centrale, nonché scrivere del codice per le combinazioni dei modificatori di tastiera MAIUSC, CTRL e ALT e il mouse.

Per entrambi gli eventi *Click* e *DblClick* sono valide le seguenti condizioni:

- Se un pulsante del mouse viene premuto mentre il puntatore si trova sopra un form o un controllo, questi "catturano" il mouse e ricevono tutti gli eventi relativi al mouse fino all'ultimo evento *MouseUp* incluso. Ciò significa che le coordinate x, y del puntatore restituite dall'evento mouse non si trovano sempre nell'area interna dell'oggetto che le riceve.
- Se i pulsanti del mouse vengono premuti in successione, l'oggetto che cattura il mouse dopo la prima pressione riceve tutti gli eventi del mouse che vengono generati fino al momento in cui si rilasciano tutti i pulsanti.

Se si desidera provare gli argomenti *button* o *shift* in modo da definirne i bit, è possibile utilizzare le costanti elencate nella libreria degli oggetti di Visual Basic (VB) nel Visualizzatore oggetti.

Costante (pulsante)	Valore	Descrizione
<i>vbLeftButton</i>	1	Viene premuto il pulsante sinistro
<i>vbRightButton</i>	2	Viene premuto il pulsante destro
<i>vbMiddleButton</i>	4	Viene premuto il pulsante centrale

Costante (tasto MAIUSC)	Valore	Descrizione
<i>vbShiftMask</i>	1	Viene premuto il tasto MAIUSC.
<i>vbCtrlMask</i>	2	Viene premuto il tasto CTRL.
<i>vbAltMask</i>	4	Viene premuto il tasto ALT.

Le costanti possono quindi essere utilizzate come maschere di bit per verificare combinazioni di pulsanti diverse senza dover identificare per ciascuna di esse il valore univoco del campo bit.

Nota È possibile utilizzare una routine dell'evento *MouseMove* in risposta ad un evento causato da uno spostamento del mouse. Mentre in *MouseDown* e *MouseUp* l'argomento *button* indica un pulsante specifico per ciascun evento, in *MouseMove* indica lo stato corrente dei pulsanti.

Evento MouseMove

Viene generato quando si sposta il mouse.

Sintassi

Private Sub Form_ *MouseMove*(button As Integer, shift As Integer, x As Single, y As Single)

Private Sub MDIForm_ *MouseMove*(button As Integer, shift As Integer, x As Single, y As Single)

Private Sub oggetto_ *MouseMove*([index As Integer,] button As Integer, shift As Integer, x As Single, y As Single)

La sintassi dell'evento *MouseMove* è composta dalle seguenti parti:

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica".
index	Intero che identifica in modo univoco un controllo se questo è incluso in una matrice di controlli.
button	Intero che identifica lo stato dei pulsanti del mouse. Quando un pulsante è premuto, viene impostato un bit. L'argomento button è un campo bit con bit corrispondenti al pulsante sinistro (bit 0), destro (bit 1) e centrale (bit 2). Questi bit corrispondono rispettivamente ai valori 1, 2 e 4. È possibile impostare tutti i bit, solo alcuni o nessuno ad indicare che sono premuti tutti i pulsanti, alcuni o nessuno.
shift	Intero che corrisponde allo stato dei tasti MAIUSC, CTRL e ALT quando si preme o si rilascia il pulsante specificato dall'argomento button. Viene impostato un bit se il tasto è premuto. L'argomento shift è un campo bit in cui i bit meno significativi corrispondono a MAIUSC (bit 0), CTRL (bit 1) e ALT (bit 2). Questi bit corrispondono rispettivamente ai valori 1, 2 e 4. L'argomento shift indica lo stato dei tasti. È possibile impostare tutti i bit, uno solo o nessuno a seconda che si desideri indicare che tutti, uno solo o nessuno dei tasti viene premuto. Per indicare, ad esempio, che vengono premuti sia CTRL che ALT, si dovrà impostare l'argomento shift su 6.
x, y	Numero che specifica la posizione corrente del puntatore del mouse. I valori x e y vengono sempre espressi in relazione al sistema di coordinate definito mediante le proprietà <i>ScaleHeight</i> , <i>ScaleWidth</i> , <i>ScaleLeft</i> e <i>ScaleTop</i> dell'oggetto.

Osservazioni

L'evento *MouseMove* viene generato più volte in successione mentre il puntatore del mouse si sposta sugli oggetti. Un oggetto riconosce un evento *MouseMove* ogni volta che il mouse si trova all'interno dei bordi dell'oggetto, a meno che il mouse non sia stato catturato da un altro oggetto.

Se si desidera provare gli argomenti button o shift in modo da definirne i bit, è possibile utilizzare le costanti elencate nella libreria degli oggetti di *Visual Basic* (VB) nel Visualizzatore oggetti.

Costante (pulsante)	Valore	Descrizione
<i>vbLeftButton</i>	1	Viene premuto il pulsante sinistro
<i>vbRightButton</i>	2	Viene premuto il pulsante destro
<i>vbMiddleButton</i>	4	Viene premuto il pulsante centrale

Costante (tasto MAIUSC)	Valore	Descrizione
<i>vbShiftMask</i>	1	Viene premuto il tasto <i>MAIUSC</i> .
<i>vbCtrlMask</i>	2	Viene premuto il tasto <i>CTRL</i> .
<i>vbAltMask</i>	4	Viene premuto il tasto <i>ALT</i> .

Le costanti possono quindi essere utilizzate come maschere di bit per verificare combinazioni di pulsanti diverse senza dover identificare per ciascuna di esse il valore univoco del campo bit.

Per verificare una condizione assegnare ogni risultato a una variabile intera temporanea e quindi confrontare gli argomenti *button* o *shift* con una maschera di bit. Utilizzare l'operatore *And* con ciascun argomento per verificare se la condizione è maggiore di zero, ad indicare che il tasto o il pulsante è premuto, come in questo esempio:

LeftDown = (Button And vbLeftButton) > 0

CtrlDown = (Shift And vbCtrlMask) > 0

È quindi possibile verificare in una routine qualsiasi combinazione di condizioni, come in questo esempio:

If *LeftDown And CtrlDown* Then

Nota È possibile utilizzare una routine dell'evento *MouseMove* in risposta ad un evento causato dalla pressione o dal rilascio del mouse.

Mentre in *MouseDown* e *MouseUp* l'argomento *button* indica un pulsante specifico per ciascun evento, in *MouseMove* indica lo stato corrente di tutti i pulsanti. Un singolo evento *MouseMove* può indicare che tutti, uno solo o nessuno dei pulsanti viene premuto.

Gli spostamenti di una finestra all'interno di un evento *MouseMove* possono causare un evento a catena. L'evento *MouseMove* viene generato quando la finestra viene spostata sotto il puntatore o, persino, quando il mouse è immobile.

Evento Timer

Viene generato quando è trascorso l'intervallo di tempo predefinito impostato per un controllo *Timer*. La frequenza dell'intervallo viene registrata nella proprietà *Interval* del controllo che specifica la durata dell'intervallo in millisecondi.

Sintassi

Private Sub oggetto_ *Timer* ([index As Integer])

La sintassi dell'evento *Timer* è composta dalle seguenti parti:

Parte	Descrizione
oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
index	Intero che identifica in modo univoco un controllo se questo è incluso in una matrice di controlli.

Osservazioni

Questa routine dell'evento consente di specificare quale azione deve essere eseguita dopo che è trascorso ciascun intervallo di tempo del controllo *Timer*. Quando si utilizza l'evento *Timer*:

- La proprietà *Interval* specifica l'intervallo che intercorre tra i vari eventi *Timer* espresso in millisecondi.
- Quando la proprietà *Enabled* del controllo *Timer* è impostata su *True* e la proprietà *Interval* è maggiore di 0, l'evento *Timer* attende per il periodo di tempo specificato nella proprietà *Interval*.

Evento Unload

Viene generato quando un form sta per essere rimosso dallo schermo. Nel momento in cui il form viene ricaricato, il contenuto di tutti i relativi controlli viene reinizializzato. Questo evento viene generato dalla chiusura di un form utilizzando il comando Chiudi del menu di controllo o l'istruzione ***Unload***.

Sintassi

Private Sub oggetto_ ***Unload***(cancel As Integer)

La sintassi dell'evento ***Unload*** è composta dalle seguenti parti:

Parte	Descrizione
Oggetto	Espressione oggetto che definisce un oggetto dell'elenco "Si applica a".
cancel	Intero che determina se il form viene rimosso o meno dallo schermo. Se cancel è 0, il form viene rimosso. Per impedire che il form venga rimosso, impostare cancel su un valore diverso da zero.

Osservazioni

Impostando cancel su un valore diverso da zero, si impedisce che il form venga rimosso, ma non si interrompono altri eventi, quali ad esempio l'uscita dall'ambiente operativo Microsoft Windows. Per impedire l'uscita da Windows, utilizzare l'evento ***QueryUnload***.

La routine dell'evento ***Unload*** consente di verificare se il form deve essere scaricato o di specificare le azioni che si desidera vengano eseguite quando il form è stato scaricato. È inoltre possibile includere del codice di convalida a livello di form necessario per chiudere il form o per salvare i relativi dati in un file.

L'evento ***QueryUnload*** viene generato prima dell'evento ***Unload***, che a sua volta viene generato prima dell'evento ***Terminate***.

L'evento ***Unload*** può essere generato dall'istruzione ***Unload*** o scegliendo il comando Chiudi dal menu di controllo di un form, uscendo dall'applicazione con il pulsante Fine del task nell'Elenco dei task di Windows, chiudendo il form MDI di cui il form corrente è secondario o uscendo dall'ambiente operativo Microsoft Windows mentre l'applicazione è in esecuzione.

Sez. I

Categoria Tipi di Dati

TIPI DI DATI

[Boolean](#)
[Byte](#)
[Currency](#)
[Date](#)
[Decimal](#)
[Double](#)
[Integer](#)
[Long](#)
[Object](#)
[Single](#)
[String](#)
[Variant](#)

[Torna all'indice delle Categorie](#)

Tipo di dati Boolean

Le variabili di tipo **Boolean** sono memorizzate come numeri a 16 bit (2 byte) e possono essere solo **True** o **False**. Le variabili di tipo **Boolean** sono visualizzate come Vero o Falso quando si utilizza Print, oppure #TRUE# o #FALSE# quando si utilizza Write #. Utilizzare le parole chiave **True** e **False** per assegnare alle variabili di tipo **Boolean** uno dei due stati.

Quando si convertono in **Boolean** dati numerici, 0 diventa **False** e tutti gli altri valori diventano **True**. Quando i valori di tipo **Boolean** vengono convertiti in altri tipi di dati, **False** diventa 0 e **True** diventa -1.

Tipo di dati Byte

Le variabili di tipo **Byte** sono archiviate come unità da 8 bit (1 byte) singole e senza segno in un intervallo compreso tra 0 e 255.

Il tipo di dati **Byte** risulta utile per l'archiviazione di dati binari.

Tipo di dati Currency

Le variabili di tipo **Currency** sono memorizzate come numeri a 64 bit (8 byte) in formato intero, divise per 10.000 in modo da ottenere un numero a virgola fissa con 15 cifre alla sinistra della virgola decimale e 4 cifre alla sua destra. Con questa rappresentazione è possibile specificare un intervallo compreso tra -922.337.203.685.477,5808 e 922.337.203.685.477,5807. Il carattere di dichiarazione del tipo per **Currency** è il simbolo di chiocciola (@).

Il tipo di dati **Currency** è utile per i calcoli monetari e per calcoli a virgola fissa in cui la precisione riveste un'importanza particolare.

Tipo di dati Date

Le variabili di tipo **Date** sono memorizzate come numeri IEEE di 64 bit (8 byte) che rappresentano date comprese nell'intervallo fra l'1 gennaio 100 e il 31 dicembre 9999 e orari compresi fra le 0.00.00 e le 23.59.59. È possibile assegnare alle variabili **Date** qualsiasi valore letterale di data riconoscibile. I valori letterali di data devono essere indicate fra caratteri cancelletto (#), ad esempio #January 1, 1993# oppure #1 Jan 93#.

Le variabili **Date** visualizzano le date sulla base del formato di data breve riconosciuto dal sistema in uso. Gli orari vengono visualizzati, analogamente, sulla base del formato di ora riconosciuto dal sistema in uso (12 o 24 ore).

Quando altri tipi di dati numerici vengono convertiti in **Date**, i valori a sinistra della virgola decimale rappresentano le informazioni relative alla data, mentre i valori a destra della virgola rappresentano l'orario. Mezzanotte corrisponde a 0 e mezzogiorno a 0,5. I numeri interi negativi rappresentano le date antecedenti al 30 dicembre 1899.

Tipo di dati Single

Le variabili di tipo **Single** (con virgola mobile a precisione singola) sono memorizzate come numeri IEEE di 32 bit (4 byte) compresi nell'intervallo fra -3,402823E38 e -1,401298E-45 per i valori negativi e fra 1,401298E-45 e 3,402823E38 per quelli positivi. Il carattere di dichiarazione del tipo per **Single** è il punto esclamativo (!).

Tipo di dati String

Esistono due tipi di stringa:

- Stringhe di lunghezza variabile, che possono contenere fino a circa 2 miliardi (2^{31}) di caratteri.
- Stringhe di lunghezza fissa, che possono contenere da 1 a circa 64 KB (2^{16}) di caratteri.

Nota Una stringa di lunghezza fissa di tipo **Public** non può essere utilizzata in un modulo di classe.

I codici per i caratteri **String** sono compresi fra 0 e 255. I primi 128 caratteri (0-127) del set di caratteri corrispondono alle lettere e ai simboli di una tastiera standard americana. Questi primi 128 caratteri sono gli stessi definiti dal set di caratteri ASCII. I successivi 128 caratteri (128-255) sono caratteri speciali, quali lettere di alfabeti internazionali, accenti, simboli di valuta e frazioni. Il carattere di dichiarazione del tipo per **String** è il simbolo del dollaro (\$).

Tipo di dati Variant

Il tipo di dati **Variant** è il tipo di dati in cui vengono trasformate tutte le variabili se non sono dichiarate esplicitamente come tipo diverso utilizzando istruzioni quali **Dim**, **Private**, **Public** o **Static**. Il tipo di dati **Variant** non ha un carattere di dichiarazione del tipo specifico.

Variant è un tipo di dati speciale che può contenere qualsiasi tipo di dati a eccezione dei dati **String** a lunghezza fissa e dei dati di tipo definito dall'utente. **Variant** può inoltre contenere i valori speciali **Empty**, **Error**, **Nothing** e **Null**. Utilizzando la funzione **VarType** o **TypeName** è possibile determinare come vengono gestiti i dati nel tipo **Variant**.

I dati numerici possono essere valori numerici interi o reali compresi nell'intervallo fra -1,797693134862315E308 e -4,94066E-324 per i valori negativi, e fra 4,94066E-324 e 1,797693134862315E308 per quelli positivi. Per i dati numerici di tipo **Variant** viene in genere conservato il relativo tipo di dati originale all'interno di **Variant**. Se, ad esempio, si assegna un **Integer** a un tipo di dati **Variant**, le operazioni successive tratteranno **Variant** come se fosse un **Integer**. Tuttavia, se viene eseguita un'operazione aritmetica in un tipo **Variant** contenente dati di tipo **Byte**, **Integer**, **Long** o **Single** e il risultato non rientra nell'intervallo normale del tipo di dati originale, all'interno di **Variant** il risultato verrà trasformato nel successivo tipo di dati di maggiori dimensioni. **Byte** verrà trasformato in **Integer**, **Integer** verrà trasformato in **Long**, mentre **Long** e **Single** verranno trasformati in **Double**. Se le variabili di tipo **Variant** contenenti valori **Currency**, **Decimal** e **Double** non rientrano nei rispettivi intervalli, verrà generato un errore.

È possibile utilizzare il tipo di dati **Variant** al posto di qualsiasi tipo di dati per gestire i dati in modo più flessibile. Se il contenuto di una variabile **Variant** è rappresentato da cifre, queste possono essere una rappresentazione in forma di stringa delle cifre o il loro effettivo valore, a seconda del contesto. Ad esempio:

Dim MyVar As **Variant**

MyVar = 98052

Nell'esempio, **MyVar** contiene una rappresentazione numerica, ovvero il valore effettivo 98052. Gli operatori aritmetici funzionano nel modo previsto con le variabili di tipo **Variant** che contengono valori numerici o dati di stringa interpretabili come numeri. Utilizzando l'operatore + per aggiungere **MyVar** a un'altra **Variant** contenente un numero oppure a una variabile di un tipo di dati numerico, il risultato sarà una somma aritmetica.

Il valore **Empty** denota una variabile **Variant** che non è stata inizializzata, ovvero a cui non è stato assegnato un valore iniziale. Un tipo di dati **Variant** che contiene **Empty** è 0 se utilizzato in un contesto numerico, è invece una stringa di lunghezza zero (""), se utilizzato all'interno di una stringa.

Non confondere *Empty* con *Null*. *Null* indica che la variabile *Variant* non contiene volutamente dati validi.

In un tipo di dati *Variant*, *Error* è un valore speciale utilizzato per indicare che in una routine si è verificata una condizione di errore. Tuttavia, diversamente da altri tipi di errore, non è prevista una normale gestione degli errori a livello di applicazione. Questo consente al programmatore o all'applicazione stessa di scegliere un'alternativa basata sul valore di errore. I valori di *Error* vengono creati convertendo numeri reali in valori di errore tramite la funzione *CVErr*.

Sez . Categoria Esempi

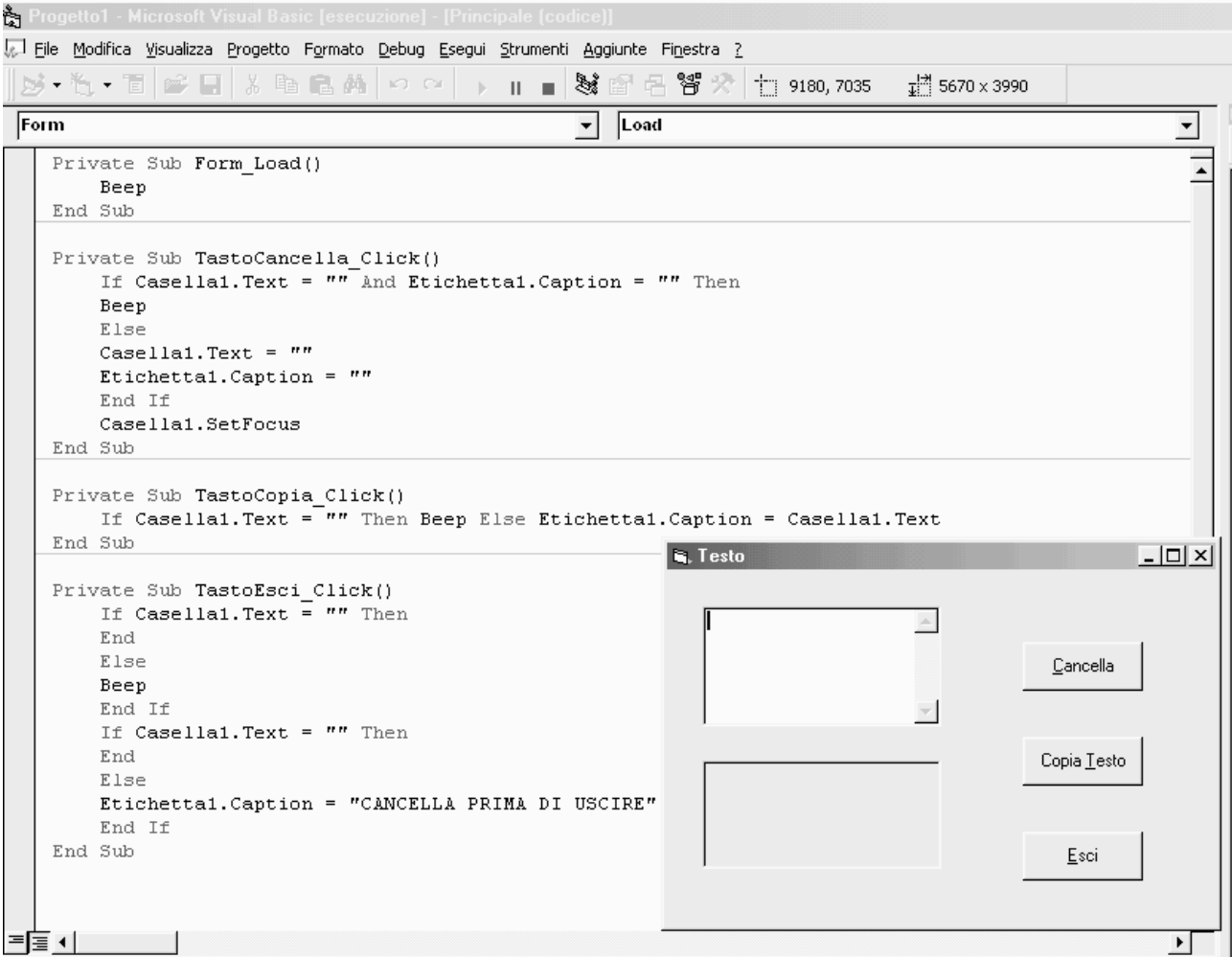
[Torna all'indice delle Categorie](#)

Esempio di programmazione N° 1

L'esempio seguente è costituito da un *Form* di partenza nel quale sono posizionati i seguenti oggetti:

- N° 1 **TextBox** (*Casella*)
- N° 1 **Label** (*Etichetta*)
- N° 3 **CommandButton** (*TastoCancella* – *TastoCopia* – *TastoEsci*)

Le righe di codice che seguono fanno sì che all'avvio del programma il cursore si posizioni nella *Casella* in attesa che l'utente digiti del testo. Al verificarsi dell'evento Click sul *TastoCopia*, il testo presente nella *Casella* viene copiato nell'*Etichetta*. Al verificarsi dell'evento Click sul *TastoCancella*, il testo presente nella *Casella* e nell'*Etichetta* viene cancellato; contemporaneamente il cursore viene riposizionato in *Casella*. Al verificarsi dell'evento Click sul *TastoEsci*, se non vi è nessun testo presente, termina l'esecuzione del programma; nel caso vi fosse del testo presente nella *Casella*, viene emesso un Beep e viene visualizzato il messaggio 'CANCELLA PRIMA DI USCIRE'.



The screenshot displays the Microsoft Visual Basic IDE in execution mode. The main window is titled 'Progetto1 - Microsoft Visual Basic [esecuzione] - [Principale [codice]]'. The code editor shows the following code for the 'Form' object:

```
Private Sub Form_Load()  
    Beep  
End Sub  
  
Private Sub TastoCancella_Click()  
    If Casella1.Text = "" And Etichetta1.Caption = "" Then  
        Beep  
    Else  
        Casella1.Text = ""  
        Etichetta1.Caption = ""  
    End If  
    Casella1.SetFocus  
End Sub  
  
Private Sub TastoCopia_Click()  
    If Casella1.Text = "" Then Beep Else Etichetta1.Caption = Casella1.Text  
End Sub  
  
Private Sub TastoEsci_Click()  
    If Casella1.Text = "" Then  
        End  
    Else  
        Beep  
    End If  
    If Casella1.Text = "" Then  
        End  
    Else  
        Etichetta1.Caption = "CANCELLA PRIMA DI USCIRE"  
    End If  
End Sub
```

Overlaid on the code editor is a dialog box titled 'Testo'. It contains two text boxes and three buttons: 'Cancella', 'Copia Testa', and 'Esci'.

Esempio di programmazione N° 2

L'esempio seguente è costituito da un **Form** di partenza nel quale sono posizionati i seguenti oggetti:

- N° 2 **TextBox** (*PrimoNumero* – *SecondoNumero*)
- N° 4 **Label** (*Risultato* – *Label1* – *Label2* – *Label3*)
- N° 6 **CommandButton** (*TastoSomma*– *TastoSottrai* – *TastoMoltiplica* – *TastoDividi* – *TastoCancella* – *TastoEsci*)

Le righe di codice che seguono, dichiarano le variabili *A* e *B* come *Integer* e fanno sì che all'avvio del programma il cursore si posizioni in *PrimoNumero* in attesa che l'utente digiti un numero. Successivamente, immettendo un numero anche in *SecondoNumero*, al verificarsi dell'evento Click su uno dei CommandButton delle quattro operazioni, viene visualizzato in *Risultato* il risultato dell'operazione relativa al CommandButton. Al verificarsi dell'evento Click sul *TastoCancella*, viene visualizzato '0' in *PrimoNumero*, in *SecondoNumero* ed in *Risultato*; contemporaneamente il cursore viene riposizionato in *PrimoNumero*. Al verificarsi dell'evento Click sul *TastoEsci*, termina l'esecuzione del programma.

The screenshot displays the Microsoft Visual Basic IDE with the following components:

- IDE Window:** Titled "Progetto1 - Microsoft Visual Basic [esecuzione] - [Form] [codice]". The menu bar includes "File", "Modifica", "Visualizza", "Progetto", "Formato", "Debug", "Esegui", "Strumenti", "Aggiunte", and "Finestra?". The toolbar shows various icons for file operations and execution. The status bar indicates coordinates "0, 0" and resolution "5550 x 4590".
- Code Editor:** Shows the code for the "TastoCancella" event. The code is as follows:

```
Dim A As Integer
Dim B As Integer
Private Sub PrimoNumero_Change()
    A = PrimoNumero.Text
End Sub
Private Sub SecondoNumero_Change()
    B = SecondoNumero.Text
End Sub
Private Sub TastoCancella_Click()
    PrimoNumero.Text = 0
    SecondoNumero.Text = 0
    Risultato.Caption = 0
    PrimoNumero.SetFocus
End Sub
Private Sub TastoDividi_Click()
    Risultato.Caption = A / B
End Sub
Private Sub TastoEsci_Click()
    End
End Sub
Private Sub TastoMoltiplica_Click()
    Risultato.Caption = A * B
End Sub
Private Sub TastoSomma_Click()
    Risultato.Caption = A + B
End Sub
Private Sub TastoSottrai_Click()
    Risultato.Caption = A - B
End Sub
```
- Form Preview:** A window titled "Operazioni" is shown. It contains:
 - Buttons for "+", "-", "x", and ":".
 - Text input fields: "Immetti il primo numero", "Immetti il secondo numero", and "Risultato".
 - Buttons for "Cancella" and "Esci".

