Gestire gli input da tastiera in un controllo TextBox

Durante la stesura di programmi o verticalizzazioni in VBA si verifica molto spesso la necessità di fare in modo che un utente in una TextBox possa digitare<u>solo</u> un input specifico, come numeri interi, stringhe etc. sia per evitare errori di elaborazione del dato inserito che per una corretta scrittura di dati specifici nei relativi fogli di lavoro. E' possibile eseguire il controllo sul testo immesso anche successivamente all'immissione, ma risulta molto più comodo limitare la scelta del tipo di dati da inserire (numeri, stringhe etc.) direttamente nella TextBox utilizzando l'evento "*KeyPress*", che contiene il tasto digitato, e il suo codice Ascii per controllarlo. Prima di iniziare ad usare codice VBA per programmare il TextBox è consigliato stilare una lista delle opzioni e dei vincoli che la casella di testo (TextBox) deve avere, inoltre si deve aggiungere anche il nostro obiettivo finale, che è quello di scrivere il valore desiderato in una cella, in formato numerico, nel foglio di lavoro, pertanto la lista potrebbe essere come la seguente:

1. Accettare solo numeri, una virgola o un segno – (meno)
2. Accettare solo un segno meno, che si deve trovare all'inizio della stringa
3. Accettare solo un punto posizionato nella posizione desiderata.
4. Scrivere il valore del TextBox in un formato numerico nel foglio di lavoro

Infine, ci si deve porre una domanda significativa; *il nostro codice viene usato una sola volta oppure poco frequentemente o viene richiamato ripetutamente?* Se è così, possiamo costruire una funzione e richiamarla ogni volta che ne abbiamo bisogno.

Iniziamo con la costruzione di una UserFom denominata "*Form1*", in cui riponiamo una TextBox denominata "*Text1*" e un pulsante di comando denominato "*Command1*" che cliccando su di esso convaliderà l'assegnazione del valore della nostra TextBox nell'apposita cella del foglio di lavoro. Iniziamo con la verifica del testo inserito nella TextBox che sia di formato numerico e che possa contenere una virgola o un segno meno utilizzando il seguente codice

```
Private Sub Text1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
If InStr("1234567890,-", Chr(KeyAscii)) = 0 Then KeyAscii = 0
End Sub
```

***Analizziamo questa riga*:** If InStr("1234567890,-", Chr(KeyAscii)) = 0 Then KeyAscii = 0

a. La funzione **Chr** restituisce una stringa contenente il carattere associato al codice carattere specificato nella variabile KeyAscii che corrisponde al codice del tasto premuto nella nostra procedura.
b. La funzione **InStr** (stringa1, stringa2), restituisce un valore di tipo Variant, che indica la posizione della prima occorrenza di una stringa (stringa2) all'interno di un'altra stringa (stringa1), nel nostro codice abbiamo: stringa1 = "1234567890, -" e stringa2 = Chr (KeyAscii)).
c. La funzione = 0 rappresenta un avviso che non è stata trovata l'occorrenza nella stringa di riferimento e quindi restituisce 0.
d. L'enunciato KeyAscii = 0 è un valore nullo assegnato al pulsante premuto che equivale ad annullare il testo digitato.

Molto brevemente con il codice sopra esposto verifichiamo se non c'è un'occorrenza del carattere corrispondente al tasto premuto nella nostra stringa di riferimento, in tal caso non prendiamo in considerazione il testo digitato. Testando il codice nella UserForm otteniamo l'effetto desiderato, ma l'utente potrebbe avere la sensazione che la sua tastiera possa avere un problema (non scrive i caratteri), si dovrebbe rimandare un messaggio di errore, ma pesante a quanti avvisi potrebbe ricevere l'utente utilizzando un metodo come questo.

Risulta più conveniente mettere un "**Beep**" tramite l'altoparlante di sistema per comunicare all'utente che il testo digitato non è permesso, pertanto il codice diventa:

```
Private Sub Text1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
If InStr("1234567890,-", Chr(KeyAscii)) = 0 Then KeyAscii = 0 : Beep
End Sub
```

A questo punto è possibile modificare il codice per accettare un segno – (meno) che sarà posto all'inizio della stringa con l'aggiunta di un operatore (Or) nella seguente forma:

```
Private Sub Text1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
If InStr("1234567890,-", Chr(KeyAscii)) = 0 Or Text1.SelStart > 0 And Chr(KeyAscii) = "-"  _
Then KeyAscii = 0 : Beep
End If
```

Analizziamo questa parte: Or Text1.SelStart> 0 And Chr (KeyAscii) = '' - ''

a. La proprietà **SelStart** indica l'inizio del testo selezionato, o il punto di inserimento e se non viene selezionato nessun testo, l'intervallo di valori validi è compreso tra 0 e il numero totale di caratteri nell'area di modifica del controllo e pertanto restituisce la posizione che è occupata dal carattere premuto, se questa non corrisponde al primo (0), l'inserimento viene cancellato
b. And Chr (KeyAscii) = "-" assicura che SelStart venga applicata solo a un carattere.

A questo punto, è possibile consentire di poter digitare una virgola in qualsiasi punto tra due numeri, ma una sola volta. Fondamentalmente abbiamo bisogno di un nuovo operatore (And) nella seguente forma:

```
Private Sub Text1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
If InStr("1234567890,-", Chr(KeyAscii)) = 0 Or Text1.SelStart > 0 And Chr(KeyAscii) = "-" _
Or InStr(Text1.Value, ",") <> 0 And Chr(KeyAscii) = ","  Then KeyAscii = 0 : Beep
End If
```

Analizziamo questa parte: Or InStr (Text1.Value, ",") <> 0 And Chr (KeyAscii) = ","

a. Viene utilizzato, nuovamente, per verificare se *InStr* ha già un punto esistente nella stringa.
b. Se il carattere è un punto (che sarebbe troppo), viene rifiutato.

In sostanza, se c'è già un punto nella stringa, e se il carattere digitato è un punto, vale a dire che stiamo cercando di inserire un secondo punto, viene rifiutato. A questo punto si dovrebbe cercare di vietare il copia e incolla, compreso il controllo della clipboard, ma questo aspetto è più difficile in quanto non possiamo semplicemente prendere in considerazione solo il primo livello degli Appunti di Excel, ma dobbiamo anche considerare Windows, e questo è oltre la portata di questo tutorial, pertanto per il momento riteniamo che non c'è nessun evento copia/Incolla diretto al TextBox, quindi proseguiamo con l'analisi degli altri vincoli da rispettare che sono i seguenti:

1. Verificare che la stringa non è vuota.
2. Che non ha punti.
3. Se ha un solo carattere, e che non è parte di una delle 10 cifre, viene rifiutato.
4. Che si tratta di una stringa che rappresenta un numero come definito inizialmente.

Quindi useremo l'evento *Private Sub Text1_BeforeUpdate (ByVal Cancel As MSForms.ReturnBoolean)*, che si verifica prima di modificare i dati in un controllo, per controllare il valore della casella di testo prima che venga convalidato.

Poco sopra abbiamo detto che se il codice venisse utilizzato di frequente sarebbe indicato utilizzare, o meglio, creare una funzione personale sia per fornire ergonomia professionale alla futura applicazione, che per comodità. Quindi possiamo sfruttare l'opportunità per costruire una funzione a questo livello con il seguente codice:

```
Private Sub Text1_BeforeUpdate(ByVal Cancel As MSForms.ReturnBoolean)
Dim stringa1 As String
Stringa1 = Text1.Value
If FunOk(stringa1) = True Then Cancel = True: Text1.Value = "": Beep: MsgBox _
"Input non valido !"
End Sub
```

La nostra Funzione è stata chiamata "*FunOK*" e restituisce un valore booleano (True o False) che ci dirà se la stringa proposta dovrebbe essere respinta o no. Per analizzare la funzione useremo la variabile stringa1 a cui viene assegnato il valore della TextBox. La funzione "FunOK", presenta di default il valore False, mentre se il valore è True, la stringa viene scartata.

```
FunOK If (stringa1) = True Then Cancel = True: Text1.Value = " "Beep: MsgBox" Input non valido "
```

Che sta a indicare: Se il valore di ritorno è True, quindi il parametro Annulla della macro è vero, vuol dire che hai inserito una stringa vuota nel controllo TextBox, emetti un segnale acustico e visualizza il messaggio "Input non valido !". Per verificare se la stringa è vuota non c'è bisogno di avviare un processo, per questo, utilizziamo una nuova istruzione IF Then Else auto-esplicativa come la seguente:

```
If stringa1 = "" Then Exit Function
```

Mentre invece se non include nessun segno possiamo usare un'altra istruzione If Then Else nella forma:

```
If Len(Replace(stringa1, ".", "")) <> Len(stringa1) Then FunOK = True: Exit Function
```

Da ricordare che la funzione **Len** restituisce un valore long che contiene il numero di caratteri di una stringa, pertanto se la lunghezza della stringa ottenuta sostituendo il punto con una stringa vuota è diversa dalla lunghezza della stringa di base, significa che la nostra stringa ha un punto e quindi viene assegnato il valore di riferimento come True alla nostra funzione e il valore sarà respinto. Se invece la stringa ha un solo carattere, che non è parte di una delle 10 cifre permesse, allora viene rifiutata in questo modo:

```
If Len(stringa1) = 1 And InStr("1234567890", stringa1) = 0 Then FunOK = True: Exit Function
```

Il codice sopra esposto sta a indicare che: Se la lunghezza della stringa è uguale a 1 e non troviamo nessuna corrispondenza con le 10 cifre permesse, allora si assegna il valore True come valore di ritorno alla funzione e il valore verrà respinto. Dopo le prime operazioni eseguite, resta ora quello di assicurarsi che la stringa corrisponda a un numero secondo i nostri desideri (numeri da 0 a 9, un possibile punto positivo o negativo). Per fare questo, sarebbe conveniente utilizzare la funzione **Val** che restituisce il numero contenuto in una stringa come un valore numerico di tipo appropriato e se non viene trovato un numero restituisce 0 e non un messaggio di errore.

A sfavore di questa funzione c'è che le virgole non sono riconosciute, infatti l'editor di VBA riconosce solo il punto come separatore numerico. Quindi dovremo trattare la stringa per testare la sostituzione (solo per le prove) della virgola, potenzialmente esistente, con il punto che ci consente di utilizzare la funzione Val con il seguente codice:

```
stringa1 = Replace(stringa1, ",", ".")
```

A questo livello, ci si baserà su questo ulteriore caratteristica della funzione Val, in cui la funzione interrompe la lettura della stringa in corrispondenza del primo carattere che non è una parte evidente di un numero. E' questo quello che ci interessa, infatti, se si tratta di una sequenza eterogenea di caratteri che comprende lettere nel mezzo di numeri, la lettura della funzione *Val* si ferma al primo carattere non permesso incontrato. Allora possiamo costruire, come sopra, il metodo per confrontare la lunghezza della stringa risultante con l'originale e vedere, ovviamente, la differenza.

Si deve ricordare che la funzione *Len* si occupa solo di lunghezze di stringhe, poi attraverso l'utilizzo della funzione Val viene convertita in un'espressione numerica, e in seguito con la funzione **CStr** si converte un'espressione numerica in una stringa. Possiamo utilizzare un codice come il seguente:

```
If Len(CStr(Val(stringa1))) <> Len(stringa1) Then FunOK = True
```

Che consiste in: Se la lunghezza della stringa costituita dal primo carattere numerico incontrato, prima di un carattere nella stringa da testare, è diversa dalla lunghezza di quest'ultimo, vuol dire che c'è un carattere indesiderato, quindi influisce sul valore restituito, cioè True alla funzione e l'inserimento verrà respinto. A questo punto resta da scrivere il valore nell'apposita cella del foglio di lavoro che possiamo farlo usando il seguente codice:

```
Private Sub command1_Click()
Cells(5, 1) = Text1.Value
End Sub
```

E otteniamo:

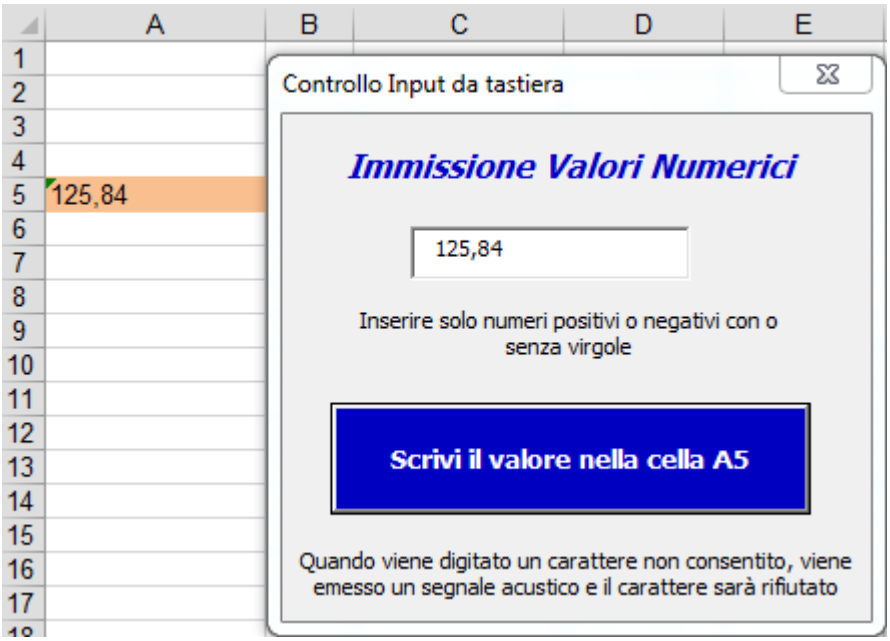

Fig. 1

L'allineamento predefinito a sinistra del nostro valore nella casella A5 indica che i dati scritti sono stati considerati come una stringa, come conferma l'attivazione del pop-up:

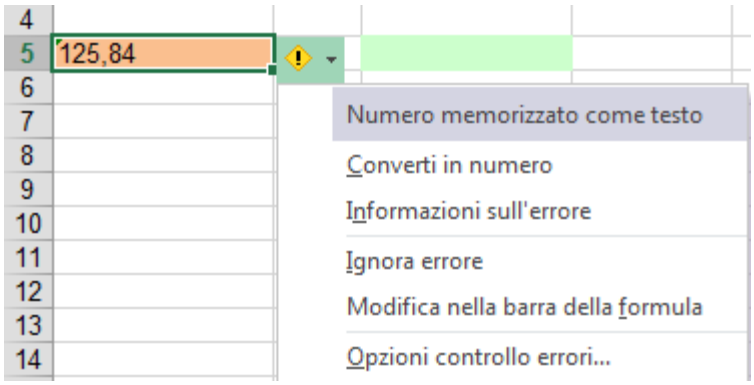

Fig. 2

Si prega di notare questo comportamento, che è la causa di molti malintesi, perché se includiamo quella cella in un calcolo come "*= A5 + A7*" con un numero in A7, avremo un risultato, mentre se si utilizza la funzione SOMMA "*= Somma (A5 : A7)*", il valore di A5 non sarà preso in considerazione, perchè, ancora una volta, è normale e segue le regole di interpretazione di Excel. Finora abbiamo visto un TextBox che restituisce valori di stringa, ma possiamo convertire il valore nel formato numerico oppure secondo i criteri della cella di destinazione usando il seguente codice:

```
Private Sub scrivi1_Click()
Cells(5, 3) = CDbl(Text1.Value)
End Sub
```

La funzione **CDbl** converte un'espressione (che può essere qualsiasi espressione stringa o un'espressione numerica) in una variabile di tipo Double, possiamo verificare scrivendo il valore nella cella C5 (verde) per visualizzare il risultato.

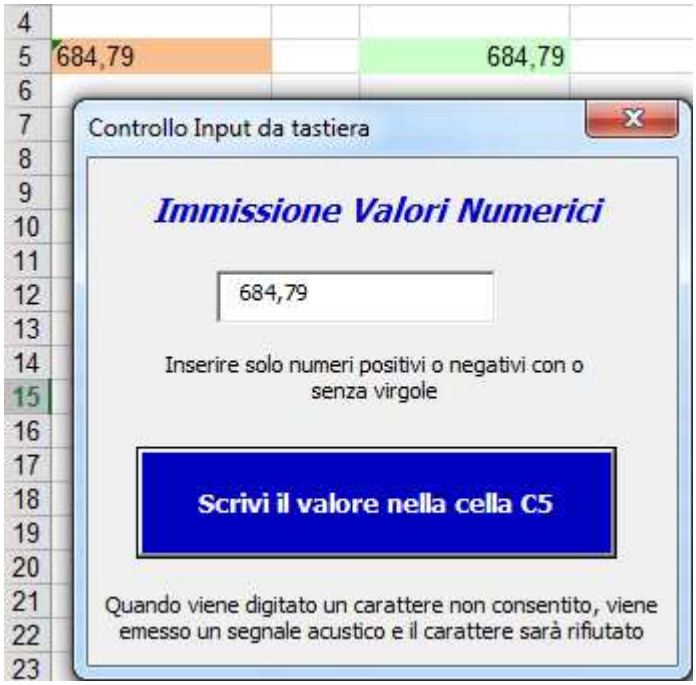

Fig. 3

Possiamo completare il codice per raggiungere i due tipi di dati da scrivere nella cella e vederne i risultati in questo modo:

```
Private Sub scrivi1_Click()
   MsgBox "Questo tipo di dati è :  " & TypeName(Text1.Value) & vbCr & vbLf _
   & "e abbiamo il testo nella cella A5 con allineamento predefinito a sinistra."
   Worksheets("Foglio1").Cells(5, 1) = Text1.Value

   Worksheets("Foglio1").Cells(5, 3) = CDbl(Text1.Value)
   MsgBox "Questo tipo di dati è :  " & TypeName(CDbl(Text1.Value)) & vbCr & vbLf _
   & "e abbiamo un numero in virgola mobile nella cella C5 con allineamento predefinito a destra"
End Sub
```

Il fatto di avere le colonne A e C estese, mette in evidenza le differenze di allineamento a causa del formato, è molto meno visibile (e fuorviante) sotto le colonne di base in funzione del numero di cifre!. In questa ultima fase abbiamo impiegato la funzione **TypeName** per visualizzare nel messaggio informativo che restituisce se una stringa corrisponde al tipo di una variabile. A questo punto il listato finale è il seguente:

```
Option Explicit
Private Sub scrivi1_Click()
   MsgBox "Questo tipo di dati è :  " & TypeName(Text1.Value) & vbCr & vbLf _
   & "e abbiamo il testo nella cella A5 con allineamento predefinito a sinistra."
   Worksheets("Foglio1").Cells(5, 1) = Text1.Value

   Worksheets("Foglio1").Cells(5, 3) = CDbl(Text1.Value)
   MsgBox "Questo tipo di dati è :  " & TypeName(CDbl(Text1.Value)) & vbCr & vbLf _
   & "e abbiamo un numero in virgola mobile nella cella C5 con allineamento predefinito a destra"
End Sub

Private Sub Text1_BeforeUpdate(ByVal Cancel As MSForms.ReturnBoolean)
Dim stringa1 As String
   stringa1 = Text1.Value
   If FunOK(stringa1) = True Then Cancel = True: Text1.Value = "": Beep: MsgBox "Input non
Valido !"
End Sub

Private Function FunOK(stringa1 As String) As Boolean
   If stringa1 = "" Then Exit Function
   If Len(Replace(stringa1, ".", "")) <> Len(stringa1) Then FunOK = True: Exit Function
   If Len(stringa1) = 1 And InStr("1234567890", stringa1) = 0 Then FunOK = True: Exit Function
   stringa1 = Replace(stringa1, ",", ".")
   If Len(CStr(Val(stringa1))) <> Len(stringa1) Then FunOK = True
End Function

Private Sub Text1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
   If InStr("1234567890,-", Chr(KeyAscii)) = 0 Or Text1.SelStart > 0 And Chr(KeyAscii) = "-" _
     Or InStr(Text1.Value, ",") <> 0 And Chr(KeyAscii) = "," Then
      KeyAscii = 0: Beep
   End If
End Sub
```