Prendere decisioni – Ciclo If e Select Case

Finora abbiamo visto delle procedure che sono in grado di portare a termine i compiti assegnati, ma però non sono in grado di prendere delle decisioni che permettano di eseguire diverse azioni in circostanze differenti, operazione che risultano necessarie in molte situazioni. A volte è necessario che sia la procedura stessa ad offrire la possibilità di poter scegliere quale azione intraprendere al verificarsi di un determinato evento, per esempio possiamo scrivere una procedura che controlli una colonna in un foglio di lavoro per verificare se tutti i numeri sono compresi tra 1 e 10, inoltre la procedura potrebbe poi esaminare ogni elemento della colonna separatamente ed eseguire azioni particolari se incontra un elemento non compreso nell'intervallo specificato.

Il Ciclo IF

Poichè le istruzioni per l'esecuzione di scelte modificano il flusso di esecuzione del programma, vengono spesso chiamate istruzioni di *controllo di flusso* o di *controllo di programma*, ma sono più note tecnicamente agli addetti ai lavori come istruzioni *condizionali* e *incondizionali*.
Un'istruzione *condizionale* è una struttura per l'esecuzione di scelte, che sceglie un blocco di istruzioni del codice del programma basandosi su una condizione o su un gruppo di condizioni predefinite, mentre un'istruzione *incondizionale* è un'istruzione che modifica semplicemente il flusso di esecuzione della procedura senza dipendere da nessuna condizione specifica. Vediamo in dettaglio questo concetto. Per eseguire un'istruzione condizionale usiamo la funzione **If** ed è rappresentata in due modi

If Condizione Then
Istruzioni
End If

Oppure a riga singola

If Condizione Then Istruzioni

Quando VBA incontra un'istruzione condizionale come If ....Then, prima valuta l'espressione logica che descrive le condizioni, in base alle quali deve essere eseguita una particolare azione, se l'espressione è *True* (cioè vera) le condizioni predefinite sono state soddisfatte e vengono eseguite le istruzioni indicate, mentre ***End If*** indica la fine del ciclo decisionale. Vediamo questa procedura con degli esempi

```
Sub prova()
 Dim var1 As Integer
 var1 = "1"
 If var1 = "1" Then MsgBox ("Bravi")
End Sub
```

In pratica la funzione fa questa valutazione: *se la variabile var1 è uguale a "1"*, [Valutazione delle condizioni], e la valutazione è *True* (cioè è vera), allora fai apparire un messaggio con la scritta *Bravi*, (Esegui l'istruzione), possiamo anche specificare più di un'azione da intraprendere in base alle condizioni usando il seguente enunciato

If Condizione Then
Istruzioni
Else
Istruzioni per Else
End If

In pratica da quanto sopra esposto aggiungiamo altre istruzioni nel caso che la condizione non venga soddisfatta, lo possiamo capire meglio con questo esempio

```
Sub prova1()
Dim var1 As Integer
var1 = "2"
If var1 = "1" Then
 MsgBox ("Bravi")
Else
 MsgBox ("Condizione non soddisfatta")
End If
End Sub
```

Oppure usando l'enunciato a riga singola

```
Sub prova()
Dim var1 As Integer
var1 = "2"
If var1 = "1" Then MsgBox ("Bravi") Else MsgBox ("Condizione non soddisfatta")
End Sub
```

Finora abbiamo visto delle istruzioni condizionali capaci di scegliere un singolo blocco di istruzioni alternativo per l'esecuzione della procedura, in molti casi però abbiamo bisogno di fare delle scelte più complesse scegliendo fra tre o quattro o più blocchi di istruzioni da eseguire, possiamo in questo caso inserire delle istruzioni *If ... Then* o *If ... Then .. Else* all'interno di altre istruzioni If ... Then o If ... Then .. Else, questa operazione si chiama **"Nidificare"** le istruzioni (nidificare significa mettere un tipo di struttura di controllo del flusso all'interno di un'altra), per usare questa sintassi è meglio usare il formato a blocchi, per una maggiore chiarezza e semplicità di lettura, l'enunciato è espresso in questa forma

If Condizione Then
Istruzioni
Else
If Condizione1 Then
Istruzioni1
Else
Istruzioni 2
End If
End If

Vediamo con un esempio come nidificare più istruzioni ed usiamo anche la funzione InputBox che abbiamo visto all'inizio

```
Sub nidifica()
Dim var1
var1 = InputBox(prompt:="Inserisci a quanti gradi metti il termostato del riscaldamento: ",
Title:="Misura la temperatura di casa")
If Pianeta > 20 Then
 MsgBox "Troppo caldo, vedrai che bolletta"
Else
  If var1 > 18 Then
  MsgBox "Temperatura giusta"
  Else
  MsgBox "Temperatura troppo bassa, ti prendi un raffreddore"
  End If
End If
End Sub
```

Eseguendo questa macro e inserendo il valore 30 otteniamo questo:

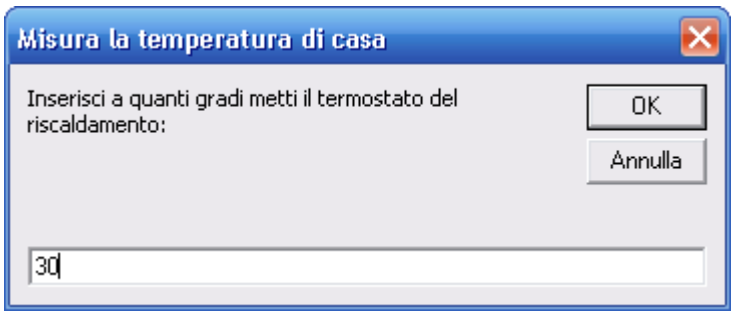

Fig. 1

Fig. 2

Se invece inseriamo il valore 19 otteniamo questi messaggi

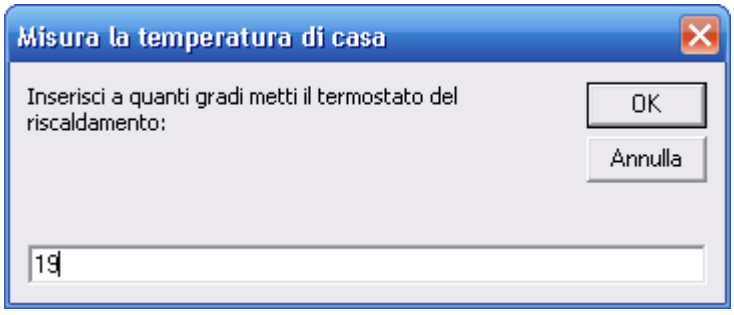

Fig. 3

Fig. 4

Inserendo invece il valore 17 ci viene mostrato questo avviso

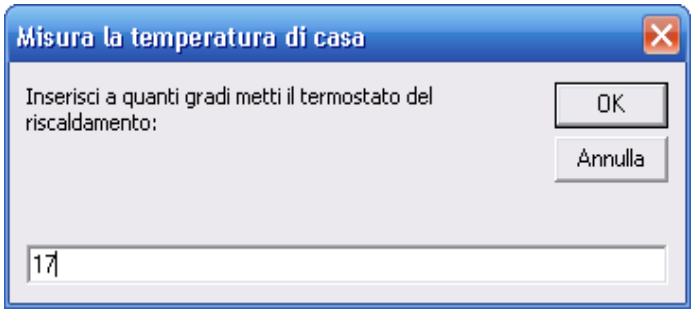

Fig. 5

Fig. 6

Vediamo nel dettaglio casa abbiamo fatto, per comprendere meglio la nidificazione usiamo anche i colori, all'inizio abbiamo dichiarato la variabile var1 ed abbiamo omesso di specificare il tipo di dati (ricordate che nella lezione 3 abbiamo detto che non dichiarando il tipo di dati la variabile assumeva per default il tipo Variant, in questo esempio è di scarsa importanza il tipo di dati), poi tramite la funzione InputBox abbiamo richiesto un dato dall'utente e di seguito abbiamo elencato le condizioni.

La prima *If var1 > 20 Then* confronta il valore della variabile var1 e avendole assegnato il valore 30 tramite InputBox viene soddisfatta la prima condizione che abbiamo posto, cioè, se var1 è maggiore di 20, porta a video il messaggio *Troppo caldo, vedrai che bolletta* a questo punto che la condizione è stata verificata e soddisfatta l'esecuzione continua dopo la parola chiave *End If*, *in questo caso segue la procedura di colore blu* e ci riporta a fine codice.

Se nell'InputBox inseriamo un valore diverso (ad esempio 20) la prima condizione *If var1 > 20 Then* **non** viene soddisfatta ed allora passiamo alle istruzioni *Else* dove, in questo blocco di istruzioni, abbiamo posto 2 condizioni, la prima che viene verificata è *If var1 > 18 Then* (Se var1 è maggiore di 18), porta a video il messaggio *Temperatura giusta*, nel nostro caso avendo inserito il valore 20 viene soddisfatta questa condizione in quanto è maggiore di 18. Nello stesso blocco *Else* abbiamo anche posto un'altra condizione senza nessun valore, che significa, semplicemente che se il valore che introduciamo con InputBox non soddisfa la condizione If var1 > 18 Then allora il flusso del programma esegue le istruzioni di questa istruzione, infatti se inseriamo il valore 15 ci viene riportato a video il messaggio "Temperatura troppo bassa, ti prendi un raffreddore".

Ma perchè queste scelte avvengono così? perchè le istruzioni *Else* sono contenute completamente all'interno dell'istruzione più esterna, e quando viene verificato il valore della variabile Var1 viene mandato in esecuzione il blocco di codice quando il valore della variabile è *True* (vera) ne consegue che inserendo il valore 30 Var1 diventa *True* alla prima condizione ed esegue il flusso di *codice di colore blu*, invece se assegniamo a Var1 il valore 20 diventa *True* alla prima istruzione delle condizioni nidificate nel blocco *Else*, ogni altro valore assegnato [da 1 a 18] alla variabile Var1 viene eseguita l'ultima istruzione del blocco Else.

Possiamo semplificare il listato della nidificazione usando un'abbreviazione che si presenta con If ... Then .. ElseIf e la possiamo rappresentare con il seguente enunciato

```
If Condizione Then
Istruzioni
ElseIf Condizione1 Then
Istruzioni1
Else
Istruzioni 2
End If
```

Consideriamo questo enunciato come un'abbreviazione, il concetto sopra esposto non varia, la nostra macro diventa così


```
Sub nidifica_abbreviato()
Dim var1
var1 = InputBox(prompt:="Inserisci a quanti gradi metti il termostato del riscaldamento: ",
Title:="Misura la temperatura di casa")
If var1 > 20 Then
 MsgBox "Troppo caldo, vedrai che bolletta"
ElseIf var1 > 18 Then
 MsgBox "Temperatura giusta"
Else
 MsgBox "Temperatura troppo bassa, ti prendi un raffreddore"
End If
End Sub
```

Funzione Select Case

Le procedure che abbiamo visto finora sono moto utili con un numero ristretto di scelte da effettuare, ma presentano un problema quando ci sono molte condizioni da verificare, tale problema è di natura interpretativa in quanto diventa difficile leggere ed interpretare il listato del codice. VBA ci offre però un'istruzione condizionale da usare quando dobbiamo scegliere tra un gran numero di possibili scelte.

L'istruzione **Select Case** funziona allo stesso modo di più istruzioni **IF** indipendenti ma è più facile da eseguire ed interpretare, usando la parola chiave *Select Case* con più istruzioni *Case*, dove ogni istruzione Case verifica la presenza di una condizione e se saranno soddisfatte le condizioni, verrà eseguito il codice di un solo blocco Case, inoltre un blocco Case può contenere nessuna, una o più istruzioni, pertanto da questa breve introduzione possiamo affermare che in presenza di varie scelte da effettuare risulta un metodo molto utile e versatile oltre che facilmente leggibile. L'istruzione Select Case ha questa sintassi

Select Case espressione
Case elencoespressione1
Istruzioni1
Case elencoespressione2
Istruzioni2
etc...
Case elencoespressione n
[Case Else
IstruzioniElse]
End Select

Come già citato il concetto e l'utilizzo è uguale a più istruzioni IF, Select Case in più offre la possibilità di poter operare varie scelte e anche di utilizzare un operatore per specificare un intervallo di valori nell'elenco espressioni, tale operatore è **To** ed è espresso in questa forma: *espressione1 To espressione2*, per esempio possiamo specificare un intervallo di numeri da 1 a 10 in un elenco Case usando la seguente dicitura

Case 1 To 10

è inoltre possibile usare degli operatori di confronto per selezionare dei blocchi di istruzioni a seconda se espressione è maggiore, minore o uguale di espressione, la sintassi è la seguente:

Is operatore_di_confronto espressione, che semplificato prende questa forma: *Case Is < 10*

Vediamo ora con un esempio pratico di semplificare ulteriormente quanto esposto e trasformiamo la routine utilizzata con l'operatore *IF* con *Select Case*

```
Sub Selec_cast()
Dim var1
var1 = InputBox(prompt:="Inserisci a quanti gradi metti il termostato del riscaldamento: ",
Title:="Misura la temperatura di casa")
Select Case var1
  Case Is > 25
    MsgBox "Troppo caldo, vedrai che bolletta"
  Case 21 To 23
    MsgBox "Bello caldo, si stà bene"
  Case 17 To 20
    MsgBox "Temperatura giusta"
  Case > 15
    MsgBox "Raffredore assicurato"
  Case Else
    MsgBox "Troppo freddo si ghiaccia"
End Select
End Sub
```

Diamo una breve spiegazione a conclusione di questa lezione, sul listato appena esposto, avrete notato che è molto intuitivo, ma vediamo assieme come vengono interpretate le varie condizioni.

Abbiamo dichiarato una variabile [var1] ed abbiamo assegnato alla stessa un valore fornito dall'utente tramite la funzione InputBox, all'inizio del ciclo Select Case notiamo che è presente una sola variabile [var1], come abbiamo citato nella lezione sulle variabili il risultato di un'espressione contenente una singola variabile è il valore memorizzato nella variabile stessa, di conseguenza VBA confronta il valore memorizzato nella variabile Var1 con le condizioni specificate in ogni blocco *Case* dell'istruzione *Select Case*.

Innanzi tutto VBA controlla il valore della variabile var1 partendo dalla prima clausola Case, se assegniamo alla variabile il valore 28 viene subito soddisfatta la prima condizione Case Is > 25 *è maggiore di 25?* in questo caso si, la condizione diventa *True* e vengono eseguite le istruzioni di quel blocco Case, allo stesso modo se la variabile assume il valore 22, la prima condizione non viene soddisfatta [non è maggiore di 25] e VBA passa alla seconda [il valore è compreso tra 21 e 23?], Si, pertanto la condizione diventa *True* al secondo blocco Case ed esegue le relative istruzioni.
In buona sostanza possiamo mettere diverse condizioni, sia singole che intervalli,
nell'enunciato *Else* vengono invece inserite le istruzioni nel caso nessuna delle condizioni elencate venga soddisfatta, in questo caso vengono eseguite le istruzioni presenti nel blocco Else

Utilizzare la parola chiave To

Si può utilizzare la parola chiave To nell'espressione da valutare per specificare l'intervallo superiore e inferiore dei valori corrispondenti, come illustrato di seguito. Il valore a fianco della parola chiave To deve essere minore o uguale al valore a destra della parole chiave To.

```
Sub Test1 ()
Dim voti As Integer
voti = InputBox ("Inserisci Voto")
Select Case voti
Caso 70 To 100
MsgBox "Buono"
Caso 40 To 69
MsgBox "Medio"
Caso 0 To 39
MsgBox "Bocciato"
Case Else
MsgBox "Fuori Valutazione"
End Select
End Sub
```

Utilizzare la parola chiave Is

Per includere un operatore di confronto (=, <>, <,>, <= o> =) nell'espressione da valutare si utilizza la parola chiave Is, che viene inserita automaticamente prima di un operatore di confronto, se non espressamente incluso.

```
Sub Test2 ()
'se temperatura è uguale a 39,5, verrà restituito il messaggio "Moderatamente caldo"
Dim temp As Single
temp = 39.5
Select Case temp
Caso Is > = 40
MsgBox "Troppo Caldo"
Caso Is > = 25
MsgBox "Moderatamente caldo"
Caso Is > = 0
MsgBox "Troppo Freddo"
Caso Is <0
MsgBox "Molto Freddo"
End Select
End Sub
```

Utilizzare una virgola per separare più espressioni

Si possono specificati più espressioni o intervalli in ogni istruzione Case, separando ogni espressione con una virgola, che ha l'effetto dell'operatore OR. Più espressioni o intervalli possono essere specificati per stringhe di caratteri.

```
Sub Test3 ()
Dim var As Variant
var = "Ciao"
Select Case var
Case a, e, i, o, u
MsgBox "Vocali"
Cas2 2, 4, 6, 8
MsgBox "Numeri"
Caso 1, 3, 5, 7, 9, "Ciao"
MsgBox "Numeri o Ciao"
Case Else
MsgBox "Non Valutabile"
End Select
End Sub
```

Esempio: Confronto tra stringhe "mele" A "uve" determina un valore compreso tra "mele" e "uve" in ordine alfabetico, e utilizza il metodo di confronto di testo predefinito in Binary (che è case-sensitive), perché Istruzione Option Compare è non specificata

```
Sub Test4 ()
'Option Compare non è specificato e quindi il confronto testo sarà case-sensitive
Dim var As Variant, risultato As String
var = InputBox ("Inserisci dati")
Select Case var
Case 1 To 10, 11 To 20: risultato = "Il numero è compreso tra 1 e 20"
Case "mele" To "uva", "mango", 98, 99: risultato = "Testo tra mele e uva, o mango, oppure
tra i numeri 98 o 99"
Case Else: risultato = "Non Valutabile"
End Select
MsgBox risultato
End Sub
```

Impostazione Option Compare

È possibile confrontare i dati stringa utilizzando metodi di confronto tra stringhe in binario, testo o database. (quest'ultimo viene utilizzato solo con Microsoft Access). Option Compare Binary rende confronti di stringhe sulla base di un ordinamento binario (in Microsoft Windows, la pagina di codice determina il tipo di ordinamento - in cui ANSI 1252 è utilizzato per l'inglese e molte lingue europee), inoltre Option Compare Text rende i confronti di stringhe che non si basano su un ordinamento testuale case-sensitive

L'istruzione Option Compare (cioè Option Compare Binary o Option Compare Text) può essere utilizzato per impostare il metodo di confronto e deve essere utilizzato a livello di modulo, prima di qualsiasi procedura. Se l'istruzione Option Compare non è specificata, il metodo di confronto testo predefinito è Binary.

```
Option Compare Binary
Sub Compare1 ()
Dim str As String
str = InputBox("Inserisci il testo ")
Select Case str
Case "Mele" To "Uva"
MsgBox " Il testo è tra mele e uva "
Case Else
MsgBox "Non Valutabile"
End Select
End Sub

Option Compare Text
Sub Compare2 ()
Dim str As String
str = InputBox ("Inserisci il testo")

Select Case str
Case "mele" To "uve"
MsgBox "Il testo è tra mele e uva"
Case Else
MsgBox "Non Valutabile"
End Select
End Sub
```

Select Case Annidati

Il blocco di istruzioni Select Case può essere nidificato all'interno di ogni altro ciclo, come If ... Then ... Else e Loop, senza alcun limite. Quando Select Case è nidificato dentro l'altro, deve essere un blocco completo e terminare con la propria End Select , all'interno di una specifica Case o Case Else

```
Sub select1 ()
Dim rng As Range, int1 As Integer
Set rng = ActiveSheet.Range ("A1")
Select Case IsEmpty (rng)
Case True
MsgBox rng.Address & "è vuota"
Case Else
Select Case IsNumeric (rng)
Case True
MsgBox rng.Address e "ha un valore numerico"
Select Case rng.HasFormula
Case True
MsgBox rng.Address & "ha anche una formula"
End Select
Case Else
Int1 = Len (rng)
MsgBox rng.Address & "ha una lunghezza di testo di" & int1
End Select
End Select
End Sub
```

Esempio: Manipolazione del testo con le istruzioni condizionali nidificati

```
Funzione StringManipulation (str As String) As String
'Questo codice personalizza una stringa di testo come segue:
`1. rimuove i numeri da una stringa di testo;
'2. rimuove gli spazi iniziali e finali
'3. aggiunge uno spazio (se non presente) dopo ogni esclamativo, virgola, punto e interrogativo;
Dim iTxtLen As Integer, iStrLen As Integer, n As Integer, i As Integer, ansiCode As Integer
'Toglie i numeri
'Chr (48) chr (57) rappresentano i numeri da 0-9 in codici ASCII
For i = 48 To 57
'Rimuovere tutti i numeri dalla stringa di testo
str = Replace (str, Chr (i), "" )
Next i
'Toglie gli spazi con la funzione TRIM
str = Application.Trim (str)
 'Aggiunge uno spazio (se non presente) dopo ogni ! ; ? .
iTxtLen = Len (str)
For n = iTxtLen To 1 Step -1
'Chr spazio (32); Chr (33) esclamativo; Chr (44) virgola; Chr (46) punto; Chr (63) interrogativo;
If Mid (str, n, 1) = Chr (33) Or Mid (str, n, 1) = Chr (44) Or Mid (str, n, 1) = Chr (46) Or Mid (str, n, 1 )
= Chr (63) Then
'Controlla se lo spazio non è presente
If Mid (str, n + 1, 1) <> Chr (32) Then
` aggiungere lo spazio - notare che viene usato lunghezza della stringa corrente
str = Mid (str, 1, n) & Chr (32) & Right (str, iTxtLen - n)
'Update lunghezza della stringa - incrementa di 1 dopo l'aggiunta di uno spazio
iTxtLen = iTxtLen + 1
End If
End If
Next n
'Cancella gli spazi (se presenti) prima di ogni esclamativo etc e resetta la lunghezza della stringa
iTxtLen = Len (str)
For n = iTxtLen To 1 Step -1
'Chr spazio (32); Chr (33) esclamativo; Chr (44) virgola; Chr (46) punto; Chr (63) interrogativo
If Mid (str, n, 1) = Chr (33) Or Mid (str, n, 1) = Chr (44) Or Mid (str, n, 1) = Chr (46) Or Mid (str, n, 1 )
= Chr (63) Then
'Controlla se lo spazio è presente
If Mid (str, n - 1, 1) = Chr (32) Then
'eliminare uno spazio
str = Application.Replace (str, n - 1, 1, "" )
`ricontrollare nuovamente lo stesso carattere - la posizione di n spostamenti (diminuisce di 1) dovuto alla
cancellazione di un carattere
n = n - 1
End If
End If
Next n
```

Continua --→

```
'maiuscola la prima lettera della stringa e la prima lettera di una parola dopo ogni esclamazione, punto e punto interrogativo, mentre tutte le altre lettere sono minuscole
iStrLen = Len (str)
For i = 1 To iStrLen
'Determinare il codice ANSI di ogni carattere della stringa
ansiCode = Asc (Mid (str, i, 1))
Select Case ansiCode
'97 A 122 sono i codici ANSI equiparano a lettere piccole cap "a" alla "z"
Case 97 To 122
If i> 2 Then
'maiuscola la lettera la cui posizione è 2 caratteri dopo (1 carattere dopo, sarà il carattere di spazio aggiunto in precedenza) un punto esclamativo, punto e il punto interrogativo
If Mid (str, i - 2, 1) = Chr (33) Or Mid (str, i - 2, 1) = Chr (46) Or Mid (str, i - 2, 1) = Chr (63) Then
Mid (str, i, 1) = UCase (Mid (str, i, 1))
End If
'maiuscola la prima lettera della stringa
ElseIf i = 1 Then
Mid (str, i, 1) = UCase (Mid (str, i, 1))
End If
'Se è maiuscola, passare al carattere successivo (es. next i)
Case Else
GoTo salta
End Select
salta:
Next i
'Stringa manipolata
StringManipulation = str
End Function

Sub Str_Man ()
'specificare la stringa di testo per manipolare e ottenere la stringa manipolato
Dim strText As String
'Specificare la stringa di testo, che deve essere manipolato
strText = ActiveSheet.Range ( "A1" ). Valore
'La stringa di testo manipolata viene inserita in A5 del foglio attivo
ActiveSheet.Range ("A5"). Value = StringManipulation (strText)
End Sub
```

L'istruzione GoTo

Si utilizzare l'istruzione GoTo per passare a una linea all'interno della procedura e questa istruzione si compone di 2 parti:

- La dichiarazione GoTo che è la parola chiave GoTo seguita da una etichetta che è l'identificatore
- L'etichetta che è costituita dal nome della stessa seguita da due punti, e poi ha una riga di codice.

Se viene soddisfatta una condizione, con l'istruzione goto viene trasferito il controllo ad una linea separata del codice all'interno della procedura, identificato dall'etichetta. L' istruzione Goto è di solito evitabile se c'è una soluzione alternativa, molte volte è possibile utilizzare If ... Then ... Else o Select Case, in quanto GoTo rende il codice poco leggibile e un po' confuso. Si consiglia di utilizzarlo per la gestione degli errori, vale a dire. "On Error GoTo".