29/10/2013

Informatica per la Finanza
Informatica per le Scienze Economiche
Informatica e Programmazione per il Turismo

Le routine in VBA

- Il termine *`routine`* in VB racchiude tre modi diversi di gestire sottoprogrammi:
  - Subroutine o procedure (**`Sub`**)
    - sequenze di istruzioni parametrizzabili che non restituiscono alcun valore se non attraverso i parametri (equivalgono ad una *`void function()`* del C)
  - Function o funzioni (**`Function`**)
    - Simili alle procedure ma restituiscono un valore ed hanno un tipo (come le funzioni in C)
  - *Property o proprietà (**`Property`**)*
    - *Usate per definire e manipolare le proprietà degli oggetti*

Le routine in VBA

- Quando viene richiamata una routine il collegamento fra parametri formali (quelli con cui si è dichiarata la funzione) e parametri attuali (quelli usati per richiamare la funzione) può avvenire:
  - In maniera posizionale come in C
  - Con la sintassi

    nomeParametro := valore

Le routine in VBA

- Gli argomenti di una routine vanno dichiarati uno dopo l'altro separati da virgola come segue:
  - Per ogni parametro per valore

    ByVal *nomeParametro* As *Tipo*
  - Per ogni parametro per riferimento

    ByRef nomeParametro* As *Tipo
    - La parola chiave **ByRef** si può omettere

Le routine in VBA

- Alle routine è possibile passare dei parametri
- Il passaggio dei parametri è molto flessibile in VB:
  - passaggio per riferimento (indirizzo): **`ByRef`**
    - Metodo usato se non diversamente indicato
  - passaggio per valore: **`ByVal`**
  - i parametri possono essere anche facoltativi (**`Optional`**)

Procedure

```
Sub nomeSub (parametri)
 dichiarazione variabili locali
 istruzioni
End Sub
```

- Per terminare una procedura senza giungere a **`End Sub`** si usa **`Exit Sub`**
- È meglio, non obbligatorio, dichiarare le variabili locali all'inizio della procedura

Procedure

- Per richiamare una procedura si possono usare due sintassi alternative:
  - **`Call nomeProcedura (`*`argomenti`*`)`**
  - **`nomeProcedura `*`argomenti`***
- Dove ***`argomenti`*** è l'elenco degli argomenti specificato in maniera posizionale o con la sintassi ***`nomeParametro := valore`***
- Si noti la mancanza delle parentesi nel secondo modo
  - Aggiungere le parentesi nel secondo modo provoca un comportamento anomalo:
    - Se vi è un solo parametro è comunque passato ByVal
    - Se vi è più di un parametro è generato un errore di sintassi

Esempio

```
Sub primaProc(v1 As Integer, ByVal v2 As Integer)
    Dim L As Integer
    v1 = v1 * 3
    v2 = v2 * 2
    L = v1 + v2
   MsgBox ("primaProc: " & vbCrLf & "v1=" & v1 & " v2=" & v2 & _
" l=" & l)
End Sub
```

Parametri Formali

- Si è dichiarata una procedura di nome ***`primaProc`***
- Il parametro **`v1`** è di tipo **`Integer`** passato per riferimento
- Il parametro **`v2`** è di tipo **`Integer`** passato per valore
- **`L`** è una variabile locale
- Il carattere **`vbCrLf`** indica il carattere di a capo.

Esempio

```
Sub richiama()
    Dim x As Integer, y As Integer
    x = 8
    y = 100
MsgBox ("richiama: " & vbCrLf & "x=" & x & " y=" & y)
    primaProc x, y
MsgBox ("richiama: " & vbCrLf & "x=" & x & " y=" & y)
End Sub
```

Parametri attuali

- La procedura ***richiama*** non ha parametri
- Vi sono due variabili locali di tipo **Integer**
- Osservare i valori di **x** ed **y** prima e dopo il richiamo di ***primaProc***

Nomenclatura

```
Option Explicit

Sub dati(ByRef x As Integer, ByVal y As Integer)
    x = 5
    y = 6
End Sub

Sub prova()
   Dim a As Integer, b As Integer

   a = 8
   b = 10

   Call dati(a, b)

   Range("A1") = a
   Range("B1") = b
End Sub
```

Parametro formale

Parametro attuale

prova()				
a	b	dati()		
0	0			
8	0			
8	10	x	y	
8	10	(>$a_{prova()}$)	10	
5	10	(>a)	10	x=5
5	10	(>a)	6	y=6
5	10			

Esempio

```
Option Explicit
Sub prova(ByRef a As_
 Integer, _
    ByVal b As Integer)
    a = a * 2
    b = b * 2
End Sub
Sub richiama()
   Dim x As Integer, _
       y As Integer

   x = 2
   y = 10
   prova x, y
End Sub
```

```
richiama()
  x    y
  0    0
  2    0      prova()
  2   10       a      b
  2   10     (^x)    10
  4   10     (^x)    10
  4   10     (^x)    20
  4   10
```

tempo

Esempio

```
Option Explicit
Sub parametri(ByVal x As Double, _
   ByRef y As Double)
  Dim z As Double
   x = x + 5
   y = y – x  'parametro x
   z = x - y
End Sub

Sub prv()
  Dim x As Double, y As Double

  x = 10
  y = -3

  Call parametri(x, y)
  Call parametri(y, x)
End Sub
```

```
prv()
x    y
10   -3
10   -3
10   -3
10   -18
10   -18

10 -18
10 -18
10 -18
23 -18
23 -18

23 -18
```

```
parametri()
x     y          z
10   (>yprv())   0
10   (>yprv())   0
15   (>yprv())   0
15   (>yprv())   0
15   (>yprv())   33
parametri()
x     y          z
-18  (>xprv())   0
-13  (>xprv())   0
-13  (>xprv())   0
-13  (>xprv())   -36
```

Esempio

```
Sub passa(ByRef x As Integer, _
   ByRef y As Integer, _
 ByVal z As Integer)

   z = x
   x = z - y
   y = z + 3
End Sub

Sub prova()
   Dim a As Integer,
   Dim b As Integer

   a = 3
   b = 5
   Call passa(b, a, 45)

End Sub
```

prova()		passa()		
a	b	x	y	z
3	5	(>b)	(>a)	45
3	5	(>b)	(>a)	5
3	2	(>b)	(>a)	5
8	2	(>b)	(>a)	5
8	2			

Esempio

```
Sub primaProc(v1 As Integer, _
           ByVal v2 As Integer)
    Dim L As Integer
    v1 = v1 * 3
    v2 = v2 * 2
    L = v1 + v2
End Sub

Sub richiama()
  Dim x As Integer, y As Integer
    x = 8
    y = 100

    primaProc x, y
End Sub
```

richiama()		primaProc()		
x	y	v1	v2	L
8	100	>x	100	0
8	100	>x	100	0
24	100	>x	200	0
24	100	>x	200	224
24	100			
24	100			

Esempio

```
Sub prima(v1 As Integer, _
          ByVal v2 As Integer)
    Dim L As Integer
    v1 = v1 * 5
    v2 = v2 - 10
    L = v1 + v2
    MsgBox ("prima: " & v1 _
         & " " & v2 & " " & L)
End Sub

Sub richiama()
   Dim g As Integer, _
                q As Integer
   g = 10
   q = -10
   Call prima(g, q)
   MsgBox (g & "  " & q)
   prima g, 78
   MsgBox (g & "  " & q)
End Sub
```

richiama()

g	q
10	-10
10	-10
50	-10
50	-10
50	-10
50	-10
50	-10

prima()

v1	v2	L
(^g)	-10	/
(^g)	-10	/
(^g)	-20	/
(^g)	-20	30

prima()

v1	v2	L
(^g)	78	/
(^g)	78	/
(^g)	68	/
(^g)	68	318

Esempio

```
Option Explicit

Sub scambia(ByVal x As Integer, _
            ByVal y As Integer)
   Dim app As Integer
   app = x
   x = y
   y = app
End Sub

Sub prova()
   Dim a As Integer, b As Integer

   a = 8
   b = 10

   Range("A1") = a
   Range("B1") = b
   scambia a, b
   Range("A2") = a
   Range("B2") = b
End Sub
```

prova()

a	b
8	10
8	10
8	10
8	10
8	10
8	10

scambia()

x	y	app
8	10	0
8	10	8
10	10	8
10	8	8

Esempio

```
Option Explicit

Sub scambia(ByRef x As Integer, _
                ByRef y As Integer)
   Dim app As Integer
   app = x
   x = y
   y = app
End Sub

Sub prova()
   Dim a As Integer, b As Integer

   a = 8
   b = 10

   Range("A1") = a
   Range("B1") = b
   scambia a, b
   Range("A2") = a
   Range("B2") = b
End Sub
```

prova()		scambia()		
a	b	x	y	app
8	10	(>a)	(>b)	0
8	10	(>a)	(>b)	8
10	10	(>a)	(>b)	8
10	8	(>a)	(>b)	8
10	8			

Esempio

```
Option Explicit
Sub uno(a As Double)
   Dim b As Double

   b = a * 2
   a = a - 2
End Sub

Sub richi()
  Dim x As Double, a As Double

  x = 10
  a = 50
  uno x
  uno a
End Sub
```

richi()		
x	a	
0	0	
10	0	uno()
10	50	a b
10	50	(^x) 0
10	50	(^x) 20
8	50	(^x) 20
8	50	
8	50	(^a_{richi}) 0
8	50	(^a_{richi}) 100
8	48	(^a_{richi}) 100
8	48	

Esempio

```
Sub richiama2()
    Dim x As Integer, y As Integer

    x = 8
    y = 100
MsgBox ("richiama: " & vbCrLf & "x=" & x & " y=" & y)
    primaProc v2 := x, v1 := y
MsgBox ("richiama: " & vbCrLf & "x=" & x & " y=" & y)

End Sub
```

- Si mostra l'altra modalità di passaggio dei parametri

Esempio

```
Sub richiama3()
    Dim x As Integer, y As Integer

    x = 8
    y = 100
 MsgBox ("in richiama: " & vbCrLf & "x=" & x & " y=" & y)
    Call primaProc(x, y)
 MsgBox ("in richiama: " & vbCrLf & "x=" & x & " y=" & y)

End Sub
```

- per richiamare la procedura si è usata l'istruzione **`Call`**
  - Si noti che i parametri attuali sono fra parentesi tonde
  - Anche in questa modalità è possibile passare i parametri usando l'operatore `:=`

Esercizio 1/2

- nella celle A1,A2, A3 vi sono dei valori numerici che rappresentano
  - un valore iniziale (A1)
  - un valore finale (A2)
  - un incremento (A3)

  calcolare il valore della funzione (x^3+2x-1)/x

  con X che varia da A1 ad A2 con incremento A3
  - Verificare che X non sia 0 nel caso non calcolare la funzione nel punto
  - scrivere i valori della funzione nella colonna C
  - i valori di X nella colonna B
- Ripulire tutte le celle nella colonna B prima di iniziare

```
Option Explicit
Sub Pulisci()
  Dim riga As Integer, rigaX As Integer

  rigaX = 1
  Do While Not IsEmpty(Cells(rigaX, 2))
    Cells(rigaX, 2) = ""
    rigaX = rigaX + 1
  Loop

  For riga = 1 To 10000
    If (IsEmpty(Cells(riga, 3)) And riga > rigaX)
    Then
      Exit For
    Else
      Cells(riga, 3) = ""
    End If
  Next

End Sub
```


Esercizio 2/2

```
Private Sub Lavora_Click()
    Dim x As Double, riga As Integer

    Call Pulisci
    riga = 1
    For x = Range("A1").Value To Range("a2").Value Step Range("a3").Value
       Cells(riga, 2) = x
       Cells(riga, 2).NumberFormat = "0.0000"
       If (x <> 0) Then
         Cells(riga, 3) = (x ^ 3 + 2 * x + 1) / x
         Cells(riga, 3).NumberFormat = "0.0000"
       Else
         Cells(riga, 3) = ""
       End If
       riga = riga + 1
    Next

    riga = riga - 1
    Charts.Add
    ActiveChart.ChartType = xlLineMarkers
    ActiveChart.SetSourceData Source:=Sheets("Sheet1").Range("M18")
    ActiveChart.SeriesCollection.NewSeries
    ActiveChart.SeriesCollection(1).XValues = "=Sheet1!R1C2:R" & riga & "C2"
    ActiveChart.SeriesCollection(1).Values = "=Sheet1!R1C3:R" & riga & "C3"
    ActiveChart.Location Where:=xlLocationAsObject, Name:="Sheet1"
End Sub
```

Le istruzioni per disegnare il grafico sono state ricavate con il *registratore di macro* ed adattate (adattamenti in rosso)

Esempio 1/3

- Scrivere la sub che come parametri i valori di due interi e che restituisce in un terzo parametro il loro MCD

x	y	mcd	restoX	restoY
10	15	0	0	0
10	15	10	0	0
10	15	10	0	5
10	15	9	1	6
:				
10	15	6	4	3
10	15	5	0	0

- 10 : 15 = 0
- 10

- 15:10 = 1
- 5

- 10 : 5 = 2
- 0

Esercizio 2/3

restoX <>0	restoY <> 0	resto nel ciclo?
V	V	sì (V)
V	F	sì (V)
F	V	sì (V)
F	F	no (F)

Esercizio 3/3

```
Option Explicit

Sub mcd(ByVal X As Integer, ByVal y As Integer, ByRef mc As Integer)
  Dim RestoX As Integer, RestoY As Integer

  If X < y Then
    mc = X
  Else
    mc = y
  End If

  Do
    RestoX = X Mod mc
    RestoY = y Mod mc
    mc = mc - 1
  Loop while (RestoX <> 0 Or RestoY <> 0)
  mc = mc + 1
End Sub

Sub provaO
   Dim riga As Integer, m As Integer

   'leggiamo i valori di cui calcolare MCD
   'dalle colonne A e B del foglio e
   'scriviamo il risultato in C

   riga = 1
   while (Not IsEmpty(Cells(riga, 1)) And Not IsEmpty(Cells(riga, 2)))
     Call mcd(Cells(riga, 1).Value, Cells(riga, 2).Value, m)
     Cells(riga, 3) = m
     riga = riga + 1
   Wend


End Sub
```


Funzioni

Function ***nomeFunz*** **(*parametri*)** ***As Tipo***
dichiarazione variabili locali
istruzioni
End Function

- Per terminare una funzione senza giungere a **End Function** si usa **Exit Function**
- È meglio, non obbligatorio, dichiarare le variabili locali all'inizio della procedura
- Il valore che deve restituire la funzione è assegnato al nome della funzione che in pratica è un ulteriore parametro
- Le funzioni da aggiungere all'ambiente Excel vanno dichiarate in un modulo (**module**)

Funzioni

- Per richiamare una funzione si possono usare tre sintassi alternative:
  - **`Call nomeFunzione (`*`argomenti`*`)`**
  - **`nomeFunzione `*`argomenti`***
  - ***`Val = nomeFunzione (argomenti)`***
- Dove ***`argomenti`*** è l'elenco degli argomenti specificato in maniera posizionale o con la sintassi

  `nomeParametro := valore`
- Con le prime due notazioni si perde il valore restituito

Funzioni

- L'uso delle parentesi per racchiudere i parametri attuali è diverso rispetto alle sub
  - Con l'istruzione **`Call`** è obbligatorio l'uso delle parentesi
  - Senza assegnare il valore restituito ad una variabile le parentesi vanno omesse; in caso contrario si ha un comportamento anomalo:
    - Se vi è un solo parametro è comunque passato ByVal
    - Se vi è più di un parametro è generato un errore di sintassi :
  - Quando si desidera assegnare il valore restituito ad una variabile le parentesi sono obbligatorie

Esempio

```
Function pari (x As Integer) As Boolean
   Dim y As Integer

   y = x Mod 2

   pari = (y = 0)
End Function

Sub richiama ()
  Dim r1 As Boolean, r2 As Boolean

  r1 = pari (5)
  r2 = pari (12)

  MsgBox (r1 & "   " & r2)
End Sub
```

Prepara il valore per la restituzione al chiamante

Operatore di confronto

Esempio 1/2

```
Option Explicit

Function mcd(ByVal X As Integer, ByVal y As Integer) As Integer
  Dim RestoX As Integer, RestoY As Integer

  If X < y Then
    mcd = X
  Else
    mcd = y
  End If

  Do
    RestoX = X Mod mcd
    RestoY = y Mod mcd
    mcd = mcd - 1
  Loop While (RestoX <> 0 Or RestoY <> 0)
  mcd = mcd + 1
End Function

Function mcm(ByVal X As Integer, ByVal y As Integer) As Integer
   mcm = X * y / mcd(X, y)
End Function
```

Esempio 2/2

```
Sub prova()
   Dim riga As Integer, m As Integer

   'leggiamo i valori di cui calcolare MCD
   'dalle colonne A e B del foglio e
   'scriviamo il risultato in C e in  D mcm

   riga = 1
   While (Not IsEmpty(Cells(riga, 1)) And Not IsEmpty(Cells(riga, 2)))
     Cells(riga, 3) = mcd(Cells(riga, 1).Value, Cells(riga, 2).Value)
     Cells(riga, 4) = mcm(Cells(riga, 1).Value, Cells(riga, 2).Value)
     riga = riga + 1
   Wend
End Sub
```


Esempio

- dati i valori nella colonna A calcolare la media e la varianza dei valori in essa contenuti

  Var(X) = (Σ x^2)/n –μ^2
- scrivere la funzione che restituisce:
  - come parametro il valore della media
  - come valore di ritorno della funzione la varianza
- Nella sub chiamante i valori vanno scritti rispettivamente in B1 e B2

Esempio

```
Option Explicit
Function varMed(ByRef media As Double) As Double
   Dim n As Integer, sommaVal As Double
   Dim sommaValQuad As Double, riga As Integer

   riga = 1
   While Not IsEmpty(Cells(riga, 1))
      sommaVal = sommaVal + Cells(riga, 1).Value
      sommaValQuad = sommaValQuad + Cells(riga, 1).Value ^ 2
      n = n + 1
      riga = riga + 1
   Wend

   media = sommaVal / n
   varMed = sommaValQuad / n - media ^ 2
End Function

Sub richiama()
  Dim m As Double, v As Double

  v = varMed(m)

  Range("B1") = m
  Range("B2") = v
End Sub
```


Esempio

- Scrivere una funzione che ha come parametro un valore numerico e lo restituisce raddoppiato
- utilizzare questa funzione in Sub che legge un valore dalla cella A3 lo passa alla funzione e quindi scrive il risultato nella cella C8

Esempio

```
Function raddoppia(val As Double)
     raddoppia = val * 2
End Function

Sub calcola()
    Dim v As Double
    v = Range("A3").Value
    Range("C8").Value = raddoppia(v)
End Sub
```


Esempio

```
Function fx(y As Double) As Double _
    Dim z As Double

  z = z - y
  y = y + 4
  fx = 5
End Function

Sub rch()
  Dim a As Double, b As Integer

  a = 5
  b = fx(a)

End Sub
```

```
   rch()
a      b    fx()
5      0   y      z    fx
5      0  (>a)    0    0
5      0  (>a)   -5    0
9      0  (>a)   -5    0
9      0  (>a)   -5    5
                  5 <-
9      5
```

Esempio

```
Option Explicit
Function fx(ByVal y As _
          Integer) As Integer

    If y <= 0 Then
      fx = 10
   Else
     fx = fx(y - 1)
   End If

End Function

Sub rch()
  Dim a As Integer, b As
   Integer

  a = 2
  b = fx(a)

End Sub
```

```
rch()
 a b        fx(2)
 2 0        y    fx     fx(1)
 2 0        2     0     y    fx     fx(0)
 2 0        2     0     1     0    y    fx
 2 0        2     0     1     0    0     0
 2 0        2     0     1     0    0    10
                                     10<-
 2 0        2     0     0    10
                            10<-
            10<-
2 10
```


Esempio

```
Option Explicit
Function raddoppia(A As Double) _
                    As Double
      raddoppia = A * 2
      A = A - 2
End Function
Sub rch()
    Dim x As Double, y As Double
    Dim z As Double, w As Double

    x = 10
    y = 20
    z = 30
    Call raddoppia(x)
    raddoppia y
    w = raddoppia(z)

End Sub
```

```
rch()
x  y  z  w
0  0  0  0     raddoppia()
10 20 30 0      A     raddoppia
10 20 30 0     (^x)       0
10 20 30 0     (^x)       20
 8 20 30 0     (^x)       20
                               20 <-
8 20  30 0
8 20  30 0     (^y)       0
8 20  30 0     (^y)      40
8 18  30 0     (^y)      40
                           40 <-
8 18  30 0
8 18  30 0     (^z)      0
8 18  30 0     (^z)      60
8 18  28 0     (^z)      60
                           60 <-
8 18  28 60
```

Esercizio

```
Option Explicit
Function raddoppia(A As Double) _
                        As Double
      raddoppia = A * 2
      A = A - 2
End Function
Sub rch()
    Dim x As Double, y As Double
    Dim z As Double, w As Double

 x = 10
 y = 20
 z = 30
 w = raddoppia(z) + raddoppia(y) / 5

End Sub
```

```
rch()
x  y  z  w
0  0  0  0
10 0  0  0
10 20 0  0     raddoppia()
10 20 30 0      A  raddoppia
10 20 30 0     (^z)     0
10 20 30 0     (^z)     60
10 20 28 0     (^z)     60
                            60<-
10 20 28 0     (^y)     0
10 20 28 0     (^y)     40
10 18 28 0     (^y)     40
                            40 <-
10 18 28 68
```


Esercizio 1/2

- Scrivere la sub richiama() che:
  - Attraverso la funzione pos() legge dal foglio di lavoro i valori nella colonna A finché non incontra un valore positivo che restituisce
  - Attraverso la funzione somma() calcola il valore della somma di tutti gli interi compresi fra 1 ed il valore restituito da pos()
  - Scrive questo valore nella cella C3

Esercizio 2/2

```
Option Explicit
Function pos() As Integer
  Dim i As Integer

  i = 1
  'sono sicuro che ci sia un valore positivo
  While (Cells(i, 1).Value <= 0)
     i = i + 1
  Wend
  pos = Cells(i, 1).Value
End Function

Function somma(ByVal x As Integer)
   somma = x * (x + 1) / 2
End Function

Sub richiama()
   Dim vP As Integer

   vP = pos()
   Range("C3") = somma(vP)
End Sub
```


Esempio

- Scrivere una subroutine che preleva il valore della cella C7 e scrive da D7 in avanti tutti i divisori di C7

```
Sub divisori()
    Dim v As Integer, i As Integer
    Dim j As Integer
   v = Range("C7").Value
   j = 1
   For i = 1 To v
       If v Mod i = 0 Then
         Cells(7, (3 + j)) = i
         j = j + 1
       End If
   Next
   While Not IsEmpty(Cells(7,(3+j)))
        Cells(7, (3 + j)) = ""
         j = j + 1
   Wend
End Sub
```

Come richiamare le funzioni di libreria

```
Option Explicit


Sub richiama()
  Dim n As Integer

  n = 1
  While Not IsEmpty(Cells(n, 1))
    n = n + 1
  Wend
  n = n - 1

  Range("B1") = Application.Average(Range("A1:A" & n))
  Range("B2") = Application.Var(Range("A1:A" & n)) * (n - 1) / n

End Sub
```

Si possono usare solo i nomi in Inglese delle funzioni

Parametri facoltativi

- In VB è molto semplice passare ad una routine una serie di parametri facoltativi (*optional parameter*)
  - Si fa precedere dalla parola chiave **`Optional`**
    - **`ByRef`** o **`ByVal`** se presenti
    - altrimenti **`Optional`** precede il nome del parametro

    I parametri **`Optional`** devono essere gli ultimi della lista
  - Si dichiara il parametro di tipo **`Variant`**
    - Strettamente parlando il parametro potrebbe essere di qualsiasi tipo ma se è di tipo **`Variant`** è possibile usare la funzione **`IsMissing(`*`nomeParametro`*`)`** che restituisce **`True`** se il parametro è stato passato alla funzione

Parametri facoltativi

- Quando si richiama la routine i parametri facoltativi possono essere omessi
  - Usando la notazione posizionale si mette la virgola (**,**) per i parametri che non si vogliono fornire quando si debbono dare dei valori ai parametri successivi a quello da saltare
  - usando la notazione **:=** per indicare solo i parametri necessari

Parametri facoltativi

- E' buona norma dare un valore prefissato per i parametri facoltativi
  - Per i parametri di tipo **Variant**:
    - Usando la funzione **IsMissing()** per stabilire se il parametro è presente e quindi dando il valore in caso non lo sia
    - usando la notazione:

      Optional *nomeParametro* = *valoreDefault*
  - Per i parametri non di tipo **Variant**
    - usando la notazione:

      Optional *nomeParametro* As *Tipo* = *valoreDefault*

Esempio

```
Function facoltativi(x As Integer, _
                  Optional ByVal y = 2, Optional k)
    If IsMissing(k) Then k = 3
    facoltativi = x + y + k
End Function

Sub richiama ()
  MsgBox (facoltativi(8))
  MsgBox (facoltativi(8, 7))
  MsgBox (facoltativi(8, , 100))
  MsgBox (facoltativi(8, 10, 7))
  MsgBox (facoltativi(8, k:=7))
End Sub
```


Esercizio

```
Option Explicit
Sub facoltativi(x As Integer, Optional ByVal y, Optional ByVal z)
   If IsMissing(z) Then
       z = 12
    End If
   If IsMissing(y) Then
       y = -30
   End If
  x = z + y
End Sub

Sub rch()
  Dim a As Integer, b As Integer, c As Integer

  a = 8
  facoltativi a, ,900
End Sub
```