Esercizi VBA di preparazione

Quelli che seguono sono una serie di esercizi progressivi sull'uso di VBA. Si parte da esercizi molto semplici.

Parte 1: esercizi base

Esecuzioni a mano (traccia di esecuzione, trace)

Per gli esercizi che seguono dovete mostrare come cambiano le variabili istruzione per istruzione tenendo presente che le variabili dichiarate valgono inizialmente 0. Se il ciclo sembra proseguire all'infinito scrivere ciclo infinito (per verificarli usate il debugger)::

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 5
   B = 7

   A = A + B
   B = B - 3
   If (A > B) Then
     A = -9
   Else
     B = -5
   End If
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 9
   B = 12

   A = A - B
   B = B + 9
   If (A > B) Then
     A = 12
   Else
     B = -10
   End If
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 8
   B = -2
   While A > B
     B = B + 3
     A = A - 2
   Wend
   A = A - B
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 3
   B = 10
   While A < B
     B = B - 2
     A = A + 1
   Wend
   A = B - A
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 2
   B = 5
   While A < 9
     If B > 0 Then
        B = B - 6
     Else
        B = B + 3
     End If
     A = A + 3
   Wend
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 9
   B = 3
   While B < 10
     If A > 0 Then
        A = A - 6
     Else
        A = A + 3
     End If
     B = B + 3
   Wend
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 5
   B = 7

   A = A + B
   B = B - 3
   If (A > B) Then
     A = -9
   Else
     B = -5
   End If
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 9
   B = 12

   A = A - B
   B = B + 9
   If (A > B) Then
     A = 12
   Else
     B = -10
   End If
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 8
   B = -2
   While A > B
     B = B + 3
     A = A - 2
   Wend
   A = A - B
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 3
   B = 10
   While A < B
     B = B - 2
     A = A + 1
   Wend
   A = B - A
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 2
   B = 5
   While A < 9
     If B > 0 Then
        B = B - 6
     Else
        B = B + 3
     End If
     A = A + 3
   Wend
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 9
   B = 3
   While B < 10
     If A > 0 Then
        A = A - 6
     Else
        A = A + 3
     End If
     B = B + 3
   Wend
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 9
   B = 3
   While B < 10
     while  A > 6
          A = A - 2
     wend
     B = B + 3
     A = B - 1
   Wend
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 5
   B = 17

   A = A + B
   B = B - 3
   If (A > B) Then
     A = -9
   Else
     B = -5
   End If
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = -9
   B = 36

   A = A - B
   B = B + 9
   If (A > B) Then
     A = 12
   Else
     B = -10
   End If
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 3
   B = 10
   While B > 0
     B = B - 3
     A = A - 2
   Wend
   A = A - B
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = -4
   B = 2
   While A < B
     B = B - 2
     A = A + 1
   Wend
   A = B - A
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 6
   B = 5
   While A < 9
     If B > 0 Then
        B = B - 6
     Else
        B = B + 3
     End If
     A = A + 2
   Wend
End Sub
```

RICORDO che quando ci sono due cicli innestati si completa prima quello più interno poi si passa a verificare nuovamente il ciclo esterno, se è verificata la condizione si ritorna al ciclo interno.

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer, C as Integer
   A = 2
   B = 3
   While B < 10
     C = B
     while  A > 4
          A = A – 2
          C = C + 3
     wend
     B = B + 3
     A = B - 1
   Wend
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer, C as Integer
   A = 12
   B = 15
   While B < 10
     C = A
     while  A > 4
          A = A – 2
          C = C + 3
     wend
     B = B - 6
   Wend
   C = A + B
End Sub
```

Problemi base con istruzioni di assegnamento

1. Scrivere il codice VBA che permette di moltiplicare fra loro il contenuto delle celle da A1 ad A3 e di scrivere il risultato in C1. (il problema potrebbe essere risolto anche con una istruzione ciclica come il while)
2. Supponete che il foglio di lavoro contenga nella riga 1 dalla cella A1 alla cella D1 dei numeri così come nella riga 2 dalla cella A2 alla cella D2. Scrivere:
   - nella riga 3 dalla cella A3 alla cella D3 la somma degli elementi contenuti nelle righe 1 e 2 delle rispettive colonne (ad esempio in A3 va la somma di A1 e A2)
   - nella riga 4 dalla cella A4 ad D4 il prodotto degli elementi contenuti nelle righe 1 e 2 delle rispettive colonne
   - nella riga 5 dalla cella A5 alla cella D5 il resto della divisione fra la riga 1 e la riga 2 (il resto si ottiene con l'operatore mod)
3. Supponete che nella cella A1 sia indicata una velocità in Km/h e nella cella A3 una distanza in m. Scrivete nella cella A2 il tempo in minuti impiegato a percorrere la distanza (occorre gestire anche le conversioni fra unità di misura)

Problemi con istruzioni condizionali

1. scrivere nella cella A4 il valore maggiore fra quello delle celle A3 ed A2
2. scrivere nella cella A4 il valore maggiore fra quello delle celle A1, A2 ed A3
3. Supponete di dover risolvere l'equazione (a-1)x = b dove il valore di a è nella cella A1 ed il valore di b è nella cella B1. Scrivere nella cella C1 il risultato dell'equazione dopo aver verificato che a è diverso da 1. Nel caso in cui a=1 in C1 non si scrive nulla (non si fa il ramo else)

Problemi con sommatorie/produttorie

Nei problemi che seguono supponete che nella cella A1 vi sia il valore di a e nella cella B1 il valore di b. Ricordate che se il valore dell'indice di partenza (nei nostri esempi a) è maggiore dell'indice di arrivo (nei nostri esempi b) la sommatoria vale 0, la produttoria 1. Se volete potete scrivere un codice che eviti le divisioni per 0 saltando i termini per i quali il divisore vale 0 (occorre un if nel ciclo)

$s=\sum_{i=a}^{b}(i+3)$	$s=\sum_{i=a}^{b}\frac{i}{i-5}$	$s=\sum_{i=a}^{b}\left(4\cdot\frac{i}{i-5}\right)$	$s=\sum_{i=a}^{b}\left(i-\cdot\frac{i}{i+2}\right)$
$p=\prod_{i=a}^{b}\left(\frac{i+2}{4}\right)$	$p=\prod_{i=a}^{b}\left(\frac{i-2}{i}\right)$	$p=\prod_{i=a}^{b}\left(\frac{i}{i+1}\right)$	$p=\prod_{i=a}^{b}i$

Provate a scrivere il codice per questa doppia sommatoria:

$$s=\sum_{i=a}^{b}\left(\sum_{j=i+1}^{b}j\right)$$

Esercizi con i cicli

1. scrivere la funzione VBA che legge i valori nella colonna A e scrive nella colonna B, consecutivamente, senza lasciare spazi bianchi quei valori che sono maggiori di 8. La scrittura termina quando nella colonna A si trova una cella vuota.

2. Scrivere il programma VBA che legge il valore nella cella A1 e quindi scrive nella colonna B, uno per riga,
   - se il valore in A1 è positivo, tutti i numeri divisibili per 3 compresi fra 1 ed il valore in A1;
   - se il valore di A1 è negativo scrive nella colonna C tutti i valori compresi fra -1 ed il valore in A1; se il valore di A1 \`e nullo scrive in C1 il valore 0.

3. Scrivere il programma VBA che dati i valori nelle celle da A1 e C3 scrive nella colonna D, senza lasciare righe vuote i valori pari contenuti nelle celle da A1 a C3.

4. Scrivere il programma VBA che dati i valori nell'intervallo da A1 a D6 somma fra loro tutti i valori numerici (per verificare se una cella contiene un valore numerico usare la funzione IsNumeric() che ha come argomento la cella indicata o con Range() o con Cells()) e scrive il risultato nella cella E1.

5. Nella colonna A del foglio di lavoro vi sono delle righe con dei numeri e delle righe con dei valori non numerici. Scrivere nella colonna B consecutivamente tutte le righe con dei valori numerici e nella colonna C, sempre consecutivamente, tutte le righe con dei valori non numerici

6. Scrivere il programma VBA che ha nelle colonne A e B dei valori numerici e scrive nella colonna C, per ogni riga il valore più grande fra quello della colonna A e B

7. Scrivere il programma VBA che nella colonna A contiene dei valori numerici che rappresentano delle distanze, nella colonna B l'unità di misura della distanza secondo la seguente codifica:
   - 1 distanza espressa in Km
   - 2 distanza espressa in m

   Scrivere nella cella D1 la distanza totale in Km e nella cella D2 la distanza totale in m

8. provare a colorare lo sfondo delle celle di una colonna del foglio di lavoro usando la proprietà

   ```
   Cells(x,y).Interior.ColorIndex = valore
   ```

   dove X e y sono le coordinate della cella e valore è un numero fra 1 e 56 (scrivete uno per riga nella colonna A i diversi valori)
   In alternativa usate la proprietà

   ```
   Cells(x,y).Interior.Color = RGB(R,G,B)
   ```

   dove R, G e B sono dei valori numerici compresi fra 0 e 255 ed indicano la quantità di ognuno dei colori fondamentali (Red Green Blue).

Parte 2: un po' più complessi

Esecuzioni a mano (traccia di esecuzione, trace)

Per gli esercizi che seguono dovete mostrare come cambiano le variabili istruzione per istruzione tenendo presente che le variabili dichiarate valgono inizialmente 0. Se il ciclo sembra proseguire all'infinito scrivere ciclo infinito[1] (per verificare i risultati si può usare il Debugger):

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 5
   B = 7

  For A = 3 To 10 Step 4
     B = B + 2
     A = A - 1
  Next
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 4
   For A = 10 To 8 Step 3
    B = B + 3
   Next
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 4
   For A = 10 To 8 Step -2
    B = B + 3
   Next
End Sub
```

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = 4
   For A = 10 To 8 Step -1
    B = B + 3
   Next
End Sub
```

[1] Si ricorda che il ciclo **For** equivale ad un ciclo **While** secondo la seguente corrispondenza di tipo generale di cui di seguito si fornisce un esempio:

Ciclo For	Ciclo While

`For A = 1 To 10 Step 3`
`  Istruzioni ciclo`
`Next`
	`A = 1`
	`While A <= 10`
	`  Istruzioni ciclo`
	`  A = A + 3`
	`Wend`

Se la parte **Step** ***valore*** non è indicata l'incremento è unitario. Inoltre se il valore iniziale della variabile che governa il ciclo è maggiore del valore finale (nell'esempio se fosse stato **For A = 10 To 1**) l'istruzione non viene eseguita a meno di un valore di **Step** negativo, ma alla variabile che gestisce il ciclo (**A** nell'esempio) è comunque assegnato il valore iniziale.

Per l'esercizio che segue dare in ingresso le seguenti coppie di valori e quindi procedere con l'esecuzione:

A (prima Input box)	B (seconda InputBox)
5	7
8	12
7	7
-3	-8
-5	-9

```
Option Explicit
Sub pr()
  Dim A As Integer, B As Integer
   A = InputBox("dammi un valore: ")
   B = InputBox("dammi un valore: ")
      If A > B Then
     A = A + 4
     While A > 0
        A = A - B
        B = B + 1
     Wend
   Else
     B = B + 1
     For A = 4 To 8 Step 4
       B = B - 2
       A = A - 1
     Next
   End If
   A = A + B
End Sub
```

Esercizi[2]

1. Scrivere la sub px() che pero ogni cella della colonna A scrive nella colonna B la lunghezza della stringa in essa contenuta
2. Scrivere la sub VBA prv() che legge i valori contenuti nella colonna A del foglio di lavoro e scrive:
   - nella colonna B i primi tre caratteri della stringa nella colonna A,
   - nella colonna C gli ultimi 4 caratteri e

[2] Si ricorda che una sequenza ordinata di caratteri in Informatica si chiama stringa. In VBA esiste un tipo di dato apposito (**String**) per dichiarare variabili di questo tipo; una costante di tipo stringa è racchiusa fra doppi apici (**"**). Due stringhe si uniscono attraverso l'operatore **&**.
Per carattere si intendono lettere, cifre, segni di interpunzione, parentesi, in generale tutti i simboli di un alfabeto convenzionale come l'alfabeto ASCII o Unicode.
Al tipo String, come visto in classe, sono associate diverse funzioni, ecco le principali (quelle viste in classe), dove al posto di **S** può esserci una variabile di tipo **String**, il contenuto di una cella o una costante (come in tutti gli esempi proposti):

Nome	Significato	Esempio
Len(S)	Restituisce il numero di caratteri di cui è composta la stringa **S** (anche gli spazi vengono contati)	**Len("ciao bello")** restituisce il numero: **10**
Left(S,n)	Restituisce i primi **n** caratteri di **S**	**Left("ciao bello", 3)** Restituisce la stringa: ***cia***
Right(S,n)	Restituisce gli ultimi **n** caratteri di **S**	**Right("ciao bello", 4)** Restituisce la stringa: ello
Mid(S,p,n)	Restituisce **n** caratteri dalla stringa **S** a partire dal carattere di posto **p** compreso (i caratteri si contano da sinistra verso destra. Il primo carattere è nella posizione 1)	**Mid("ciao bello", 6, 3)** Restituisce la stringa: ***bel***
StrReverse(S)	Restituisce la stringa **S** scritta al contrario	**Mid("ciao bello", 6, 3)** Restituisce la stringa: ***olleb oaic***

- nella colonna D i tre caratteri successivi al secondo nella colonna A.

Supponete che la colonna A contenga solo stringhe di caratteri.

3. Le celle della colonna A contengono dei numeri o delle stringhe. Scrivere la sub qx() che per ogni cella della colonna A scrive nella colonna B il valore della cella A raddoppiato se A contiene un valore numerico, il contenuto della cella A scritto al contrario se la cella contiene una stringa (usare per discriminare il contenuto la funzione VBA **`IsNumeric()`**)
4. Nella colonna A del foglio di lavoro sono contenute delle sequenze di caratteri. Scrivere la sub prq() che per ogni valore di A se la stringa è palindroma scrive nella colonna B il valore 1, altrimenti il valore 0 (per il significato di palindromo vedere sul vocabolario/Internet, per risolvere il problema aiutarsi con la funzione **`StrReverse()`**)
5. Nella colonna A del foglio di lavoro sono contenute in ogni cella delle stringhe composte da 4 cifre, seguite dal carattere . e da altri tre caratteri (ad esempio **`7853.qxc`**). Scrivere nella colonna B la stringa che si ottiene prendendo i primi quattro caratteri contenuti nella cella A cui si aggiunge la sequenza di caratteri ***FUNE***.
6. Scrivere il codice VBA che prende tutti i valori contenuti nelle colonna A e nella colonna B e scrive nella colonna C il contenuto concatenato (unito) delle celle nella colonna A e B
7. Il foglio di lavoro contiene nella colonna A delle stringhe nella colonne B e C dei numeri interi. Scrivere il programma VBA che per ogni stringa nella colonna A scrive nella colonna D il valore ottenuto applicando al funzione **`Mid()`** con parametri rispettivamente il contenuto della colonna a, B e C.
8. Scrivere il codice VBA che permette di scrivere nelle celle da A1 a E35 il valore 0 se queste contengono un numero (verificare con `IsNumeric()`), nulla (scrivere nella cella "") in caso contrario.
9. Scrivere il codice VBA che permette di scrivere nelle celle da A1 a E35 il valore 5 se queste sono vuote (verificare con `IsEmpty()`), lascia immodificato il contenuto della cella in caso contrario.

Parte 3: funzioni e sub

Una funzione si può vedere come un sistema comodo per evitare di riscrivere più volte lo stesso codice. Per renderla più generale può avere dei parametri. In un certo modo è una sorta di funzione matematica.

Ad esempio calcolare il valore di:

$$s = \sum_{i=6}^{18} (i+1)/i$$

non è diverso da:

$$s = \sum_{i=3}^{5} (i+1)/i$$

infatti se al posto del valore dell'indice inferiore e dell'indice superiore usiamo due parametri rispettivamente inf e sup possiamo generalizzare la sommatoria:

$$s = \sum_{i=inf}^{sup} (i+1)/i$$

Allo stesso modo invece di scrivere due volte un codice molto simile possiamo scriverne uno più generale:

```
For i=inf to sup
   s = s + (i+1)/i
Next
```

La cosa si può generalizzare usando il concetto di funzione[3]:

```
Function somma(ByVal inf As Integer, ByVal sup As Integer) As
Double
  Dim i As Integer, s As Double
  For i = inf To sup
    s = s + (i + 1) / i
  Next
  somma = s
End Function
```

La funzione sarà richiamata (non può partire da sola perché non conosce i valori dei parametri) ad esempio così:

```
Sub pr()
  Range("A1") = somma(6, 18)
  Range("A2") = somma(3, 5)
End Sub
```

La funzione **`somma()`** nel file deve precedere la **`Sub pr()`**.

Riassumendo da un punto di vista sintattico la funzione si compone in questo modo:

```
Function nomeFunzione (ByVal nomePar1 as Tipo, ...) as Tipo
     codice VBA
End Function
```

[3] In realtà anche una sub ha lo stesso scopo solo che non restituisce direttamente dei valori.

Dove:

- La prima riga viene detta **intestazione** /prototipo della funzione
- `nomeFunzione` è il nome con il quale sarà chiamata la funzione
    - `nomeFunzione` è anche il nome della variabile in cui sarà messo il risultato per poter essere usato nella sub che richiama la funzione.
- `nomePar1` indica il nome del **parametro formale** usato per scambiare i dati con la sub che lo richiama. Si deve notare
    - che ha un tipo;
    - è preceduto dalla parola chiave `ByVal`.
    - ci possono essere più parametri e nel caso questi sono separati dal carattere `,`
    - la funzione può non avere parametri ma può restituire comunque un valore.
- la funzione termina con l'indicazione del *Tipo restituito*.

Per essere richiamata si procede come segue[4]:

```
variabile = nomeFunzione(p1, ...)
```

Dove:

- `variabile` è una variabile definita nella sub che richiama la funzione; deve essere di tipo compatibile con il tipo restituito dalla funzione
- `p1` è il **parametro attuale** (variabile o costante) con cui richiamare la funzione. Si noti che:
    - i parametri attuali sono separati da ,
    - il numero dei parametri attuali coincide con il numero dei parametri formali;
    - il collegamento è effettuato su base posizionale (il primo parametro attuale è collegato al primo parametro formale, il secondo parametro attuale al secondo formale e così via) e prende il nome di *istanziazione* dei parametri.

Il richiamo della funzione richiede la creazione di un nuovo spazio nella memoria centrale per gestire

- i parametri
- la variabile di restituzione (quella con lo stesso nome della funzione)
- le **variabili locali** alla funzione (quelle contenute nella funzione, nell'esempio iniziale `i` ed `s`)

[4] Esistono anche altri modi, noi vedremo solo questo.

Una volta richiamata la funzione il controllo passa al codice in essa contenuto finché non viene raggiunta l'istruzione `End Function` che restituisce il controllo alla Sub chiamante (la restituzione del valore si è evidenziata con una <- che precede il valore restituito).

```
Option Explicit
Function somma(ByVal inf As Integer, _
        ByVal sup As Integer) _
                              As Double
  Dim i As Integer, s As Double
  For i = inf To sup
    s = s + i
  Next
  somma = s
End Function

Sub pr()
  Dim X As Double, Y As Double
  Dim v As Integer

  v = 9
  X = somma(6, v)
  Y = somma(3, 5)
End Sub
```

```
   pr()
X  Y  v
0  0  0          somma()
0  0  9    inf    sup   somma   i    s
0  0  9     6      9      0     0    0
0  0  9     6      9      0     6    0
0  0  9     6      9      0     6    6
0  0  9     6      9      0     7    6
0  0  9     6      9      0     7    13
0  0  9     6      9      0     8    13
0  0  9     6      9      0     8    21
0  0  9     6      9      0     9    21
0  0  9     6      9      0     9    30
0  0  9     6      9      0     10   30
0  0  9     6      9     30     10   30
                        30<-
30 0  9
30 0  9     3      5      0     0    0
30 0  9     3      5      0     3    0
30 0  9     3      5      0     3    3
30 0  9     3      5      0     4    3
30 0  9     3      5      0     4    7
30 0  9     3      5      0     5    7
30 0  9     3      5      0     5    12
30 0  9     3      5      0     6    12
30 0  9     3      5     12     6    12
                        12<-
30 12 9
```

Si noti che:

- il parametro attuale può essere sia una variabile che una costante
- quando la funzione viene richiamata una seconda volta le variabili locali (`i` ed `s`) e la variabile di restituzione valgono 0 e quindi da un richiamo all'altro i valori <u>non</u> sono mantenuti.

Mostrare la traccia di esecuzione dei seguenti programmi (si noti che il valore di una funzione può essere utilizzato in una espressione come nei casi evidenziati in rosso. In questi casi si esegue la funzione ed il risultato lo si usa nell'espressione)

```
Option Explicit
Function fz(ByVal y As Integer) _
                          As Integer
   Dim x As Integer
   x = y + 2
   y = y - 1
   fz = x
End Function
Sub d1()
   Dim x As Integer, y As Integer
   Dim z As Integer

   x = 8
   y = 9

   For x = 5 To 13 Step 3
      y = x + 1
      z = fz(y)
   Next
End Sub
```

```
Option Explicit
Function fz(ByVal y As Integer, _
         ByVal   z   As   Integer)   _
                           As Integer
   y = y + 8
   z = z - 3
   fz = y - z
End Function
Sub d1()
   Dim x As Integer, y As Integer
   Dim z As Integer

   x = 10
   y = -10
   z = 8

   x = fz(24, y)
   z = fz(x, y)
   y = z + fz(9, x)
End Sub
```

```
Option Explicit
Function  fz(ByVal  x  As  Integer)  _
                           As Integer
   Dim y As Integer
   x = y + 2
   y = y - 1
   fz = y
End Function
Sub d1()
   Dim x As Integer, y As Integer
   Dim z As Integer

   x = 10
   y = 2

   For x = 6 To 13 Step 3
      y = x + 1
      z = fz(y)
   Next

End Sub
```

```
Option Explicit
Function fz(ByVal y As Integer, _
       ByVal z As Integer) As Integer
```

```
   y = y + 8
   z = z - 3
   fz = y - z
End Function
Sub d1()
   Dim x As Integer, y As Integer
   Dim z As Integer

   x = 10
   y = -10
   z = 8

   x = fz(z, 20)
   z = fz(x, y)
   y = z + fz(x, 10)

End Sub
```

Come esercizio trasformare in funzioni gli esercizi già proposti le volte precedenti. Ad esempio:

- scrivere una funzione che ha in ingresso un numero intero e restituisce 1 se è primo 0 altrimenti;
- scrivere una funzione ha in ingresso due numeri interi A e B e restituisce la somma di tutti i numeri divisibili per 3 compresi fra A e B
- scrivere una funzione (senza parametri) che legge (con una InputBox) dei numeri finché non viene inserito un numero positivo; la funzione restituisce il numero positivo
- scrivere una funzione che come parametro un numero X e che legge dei numeri finché non viene inserito un numero maggiore di X; la funzione restituisce il numero maggiore di X
- scrivere una funzione che ha come parametri due valori Riga e Colonna e che conta nel foglio di lavoro in tutte le celle definite dal rettangolo che ha come diagonale principale le celle da (1,1) a (Riga,Colonna) i valori numerici.

Per provare le funzioni occorre scrivere una sub che le richiama.

Parte 4: funzioni

Passaggio per indirizzo (o riferimento): ByRef

Nella modalità di passaggio dei parametri per riferimento (parola chiave **ByRef**), invece di creare una copia dei valori del parametro attuale nel parametro formale, si crea un collegamento diretto al contenuto del parametro attuale. In questo modo le modifiche vengono fatte sul parametro attuale[5].
Il vantaggi pratici sono:

- maggiore efficienza e minor spreco di memoria nello stack (la zona in cui sono posti i parametri e le variabili locali);
- possibilità di far restituire più di un valore ad una funzione;
- possibilità di usare le sub per far restituire dei valori.

Una funzione o una sub possono avere più parametri passati **ByVal** o **ByRef** in un ordine qualunque infatti ogni parametro va preceduto dalla parola chiave **ByVal** o **ByRef**[6].

Ecco un esempio di trace con il passaggio per riferimento.

```
Option Explicit
Function af(ByRef x As Integer)_
                                As Integer
   Dim y As Integer

   y = x + 1
   x = x * 2
   af = y - 5
End Function

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

  a = 9
  b = af(a)
End Sub
```

rch()		af()		
a	b	x	y	af
9	0	(>a_{rch})	0	0
9	0	(>a_{rch})	10	0
18	0	(>a_{rch})	10	0
18	0	(>a_{rch})	10	5
			5 <-	
18	5			

L'esecuzione parte, come al solito, da una sub che non ha parametri. Con la notazione **(>a_{rch})** si intende che il parametro formale x si riferisce alla variabile x della funzione rch; se è chiaro dal contesto si può evitare di indicare esplicitamente il nome della sub o function cui è collegata e quindi usare una notazione tipo **(>a)**. Si noti che l'effetto dell'istruzione `x = x * 2` avviene direttamente nelle variabili di **rch()**[7].

[5] E' evidente che il parametro attuale non può che essere una variabile a differenza dei parametri passati per valore (**ByVal**) dove il parametro attuale può essere una variabile, una costante o il risultato di una espressione. In realtà in VBA è possibile passare anche una costante o una espressione **ByRef** ma ciò va tassativamente evitato.
[6] Se davanti al parametro manca l'indicazione del tipo di passaggio, questo avviene **ByRef**; quindi non indicare il tipo di passaggio equivale ad un passaggio **ByRef**.
[7] Attenzione quando usate il Debug esso vi fa vedere il contenuto del parametro attuale!

Un ulteriore esempio è fatto utilizzando una sub con dei parametri:

```
Option Explicit
Sub aSb(ByRef x As Integer, _
            ByVal y As Integer)
  Dim t As Integer

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

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

  a = 9: b = 6
  Call aSb(a, b)
End Sub
```

```
 rch()
a     b          aSb()
9     6     x      y     t
9     6    (>a)    6     0
9     6    (>a)    6     15
12    6    (>a)    6     15
12    6    (>a)    3     15

12    6
```

Si noti che le sub vanno richiamate facendo precedere il loro nome dalla parola chiave **Call**.

Un ultimo esempio:

```
Option Explicit
Sub S1(ByVal x As Integer, _
           ByRef y As Integer)
  y = x \ 3
  y = y + 2
End Sub

Function f1(ByVal y As Integer) _
                        As Integer
  y = y + 7
  f1 = y - 5
End Function

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

  a = 8: b = 3
  b = f1((a + 7) * 2)
  Call S1(f1(b), a)
End Sub
```

```
 rch()
a     b     f1()
8     3    y    f1
8     3    30    0
8     3    37    0
8     3    37   32
                 32 <-
                f1()
8     32   y    f1
8     32   32    0
8     32   39    0
8     32   39   34
                 34<-
                S1()
                x     y
8     32   34    (>a)
11    32   34    (>a)
13    32   34    (>a)
13    32
```

Si noti che:

- **b = f1((a + 7) * 2)**: il parametro attuale è il risultato di un'espressione;
- **Call S1(f1(b), a)**: il parametro attuale per valore è il risultato della funzione **f1(b)** che è stata eseguita prima di richiamare **S1**. Il valore restituito da **f1(b)**, ovvero 34, è divenuto il primo parametro attuale di **S1**;
- l'operatore **** calcola il quoziente intero della divisione.

Esercizi

Eseguire le trace dei seguenti codici

```
Function fx(ByVal a As Integer, _
        ByRef b As Integer) As Integer
   Dim c As Integer

   c = a + b
   b = b - 3
   a = a + 2
   fx = c - a
End Function

Sub pr()
   Dim x As Integer, y As Integer, _
          z As Integer

   x = -2
   y = 4
   z = fx(x, y)
   y = fx(y, z)
End Sub
```

```
Function fy(ByRef x As Integer, _
      ByVal y As Integer) As Integer
   Dim c As Integer

   c = x - y
   y = x - 3
   x = x + 2
   fy = c - y
End Function

Sub pr()
   Dim a As Integer, b As Integer, _
             z As Integer

   a = 5
   b = -2
   z = fy(a, b)
   a = fy(b, z)
End Sub
```

```
Function fy(ByRef x As Integer, _
      ByVal y As Integer) As Integer
    x = x - 7
   fy = x - y
End Function

Sub pr()
   Dim a As Integer, b As Integer, _
       z As Integer

   a = 6
   b = 5
   While a > 2
     z = fy(a, b)
     b = b - 1
     a = a - 1
   Wend
End Sub
```

```
Function ft(ByVal a As Integer, _
      ByRef b As Integer) As Integer
   b = a + 5
   ft = b - a
End Function

Sub pr()
   Dim x As Integer, y As Integer, _
      z As Integer

   x = 4
   y = 2
   While y < 7
     z = ft(x, y)
     x = y + 1
     y = y + 1
   Wend
End Sub
```

```
Function fz(ByVal a As Integer, _
     ByRef b As Integer) As Integer
   b = a + 5
   fz = b - a
End Function

Sub pr()
   Dim x As Integer, y As Integer, _
      k As Integer

   x = 3
   y = 5
   While y < 7
     k = fz(x, y)
     k = y - 1
     y = y + 1
   Wend
End Sub
```

```
Sub S1(ByVal x As Integer, _
           ByRef y As Integer)
  y = x \ 3
  y = y + 2
End Sub

Function f1(ByVal y As Integer) _
                          As Integer
  y = y + 7
  f1 = y - 5
End Function

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

  a = 5: b = 7
'prima si richiama f1(3), poi f1(a+b)
'i valori restituiti sono
' sommati ed inseriti in b
  b = f1(3) + f1(a + b)
  Call S1(a, b)
End Sub
```

```
Option Explicit
Sub S1(ByRef y As Integer)
  Dim x As Integer
  x = y \ 4
  y = y + 2
End Sub

Sub S2(ByRef y As Integer, _
         ByRef x As Integer)
  Dim t As Integer
  t = (x + y) \ 4
  y = x + 2
End Sub

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

  a = 1: b = 8
  Call S2(a, b)
  Call S1(a, b)
End Sub
```

Risolvere i seguenti esercizi

Per le sommatorie/produttorie scrivere la funzione che realizza la sommatoria/produttoria; la funzione ha come parametri l'indice iniziale e quello finale (ovviamente ByVal). Trascrivere le funzioni in sub, in questo caso il valore restituito sarà un parametro ByRef. In ogni caso scrivete un sub che richiama la funzione/sub leggendo i valori dei parametri attuali con delle InputBox

$s=\sum_{i=a}^{b}(i+3)$	$s=\sum_{i=a}^{b}\frac{i}{i-5}$	$s=\sum_{i=a}^{b}\left(4\cdot\frac{i}{i-5}\right)$	$s=\sum_{i=a}^{b}\left(i-\cdot\frac{i}{i+2}\right)$
$p=\prod_{i=a}^{b}\left(\frac{i+2}{4}\right)$	$p=\prod_{i=a}^{b}\left(\frac{i-2}{i}\right)$	$p=\prod_{i=a}^{b}\left(\frac{i}{i+1}\right)$	$p=\prod_{i=a}^{b}i$

1. Scrivere il codice VBA che:
   - definisce la funzione fx() che ha in ingresso un numero intero x passato per valore e che restituisce 1 se il numero è positivo e divisibile per 3, altrimenti la sua metà.
   - definisce la sub pr() che preleva dal foglio di lavoro nelle posizioni da A1 a C6 tutti i valori contenuti e se questi sono dei numeri richiama la funzione fx() con argomento il valore della cella e quindi somma alla cella D1 il valore restituito dalla funzione.

2. Scrivere il codice VBA che:
   - definisce la funzione fq() che ha in ingresso un numero intero x passato per valore e che restituisce 2 se il numero \`e positivo e divisibile per 5, altrimenti il suo doppio.
   - definisce la sub prv() che preleva dal foglio di lavoro nelle posizioni da A1 a B8 tutti i valori contenuti e se questi sono dei numeri richiama la funzione fq() con argomento il valore della cella e quindi somma alla cella C1 il valore restituito dalla funzione.