Uso delle funzioni con i parametri
Calcolo del massimo comune divisore (MCD)

In matematica la **funzione** è una relazione tra un insieme, detto degli **argomenti**, e un altro, detto dei **valori**. Analogamente in Visual Basic il termine **Function** sta ad indicare un modulo software, che ha una struttura simile ai sottoprogrammi *Sub*, ma che si comporta in modo del tutto differente.
La funzione è un sottoprogramma che riceve dal sottoprogramma o da un'altra funzione i valori assegnati come parametri e restituisce un valore calcolato. Quindi le *Function* vengono usate per rappresentare un procedimento di calcolo o, in generale, un'elaborazione che deve fornire un solo valore alla procedura chiamante.

La struttura generale di una *Function* è la seguente:

```
Function NomeFunzione (Parametri As Tipo) As Tipo
. . . . . .
. . . . . .
. . . . . .
End Function
```

Il nome della funzione è seguito, tra parentesi, dai parametri e dall'indicazione dei loro tipi.
Inoltre al termine della riga di intestazione della *Function* si deve indicare qual è il tipo del valore che viene restituito al termine dell'esecuzione della *Function*.
Questo comportamento della *Function* ne giustifica il nome: infatti in matematica la funzione $y = f(x)$ restituisce un valore ad y in corrispondenza del valore fornito per x.
Poiché al termine dell'esecuzione la *Function* deve uscire con un valore, all'interno del corpo della *Function* comparirà sicuramente un'istruzione di assegnazione contenente a sinistra il nome della *Function*.

L'uso dei sottoprogrammi e delle funzioni risponde all'esigenza di costruire programmi formati da moduli che svolgono una precisa funzionalità all'interno dell'applicazione software.
Un'altra esigenza è rappresentata dalla necessità di poter utilizzare più volte uno stesso gruppo di istruzioni in momenti diversi del programma, e magari con dati diversi. Questo giustifica l'uso dei parametri nell'esecuzione delle funzioni.

L'operazione, con la quale il sottoprogramma chiamante manda i valori alla funzione, assegnandoli ai parametri, si chiama **passaggio di parametri**.
Le variabili indicate nell'intestazione della funzione si chiamano **parametri formali**; le variabili che forniscono i valori ai parametri si chiamano **parametri attuali**.

Il passaggio dei parametri può avvenire in due modi diversi:

- **passaggio di parametri per indirizzo** (o **per variabile** o **per riferimento**), quando i parametri attuali e formali fanno riferimento allo stessa cella di memoria centrale, cioè allo stesso **indirizzo** di memoria; questo è il caso in cui il programmatore vuole che i cambiamenti di valore dei parametri formali, durante l'esecuzione della funzione, si riflettano anche sui valori delle variabili corrispondenti nel sottoprogramma chiamante;
- **passaggio di parametri per valore**, quando i valori delle variabili del sottoprogramma chiamante vengono ricopiati nei parametri della funzione; i cambiamenti effettuati sui parametri formali durante l'esecuzione della funzione non influenzano i valori delle variabili nel sottoprogramma chiamante.

In Visual Basic il passaggio di parametri per valore viene distinto dal passaggio per indirizzo, facendo precedere la parola **ByVal** alla lista dei parametri passati per valore nell'intestazione della funzione chiamata:

```
Function Calcola(ByVal a, ByVal b) As Integer
```

Il passaggio di parametri per indirizzo è rappresentato invece dalla parola chiave **ByRef**.
Se, quando si passa un argomento, non si specifica la parola chiave *ByVal* o *ByRef*, il valore viene passato per indirizzo, perché per *default* (cioè in mancanza di specificazione) il passaggio di parametri avviene per indirizzo: quindi nel caso di passaggio per indirizzo la parola *ByRef* può essere omessa.

Esempio

Costruire una funzione per calcolare il Massimo Comun Divisore di due numeri.

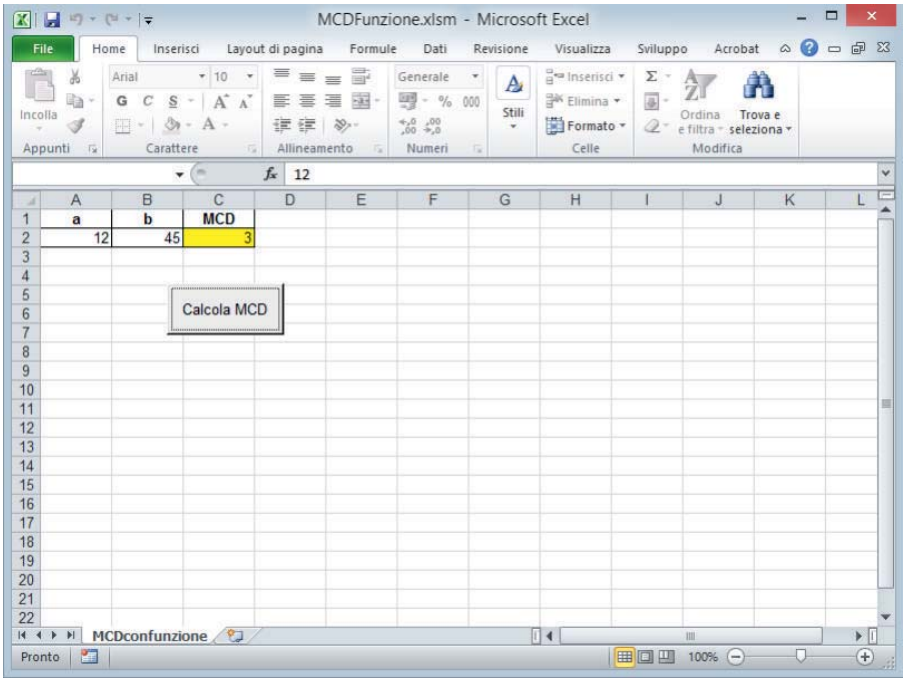
Descriviamo un algoritmo che si basa sull'idea intuitiva di MCD, come il più grande tra i divisori comuni tra due numeri.

- *Dati di input*:
  due numeri a, b.
- *Dati di output*:
  MCD dei due numeri.
- *Risoluzione*: L'algoritmo all'inizio stabilisce quale tra i due numeri sia il minore. Viene poi assegnato al MCD il valore iniziale 1, che diventerà il MCD nel caso in cui i due numeri non abbiano divisori comuni. Con una ripetizione, poi, da 2 fino al numero minore, l'algoritmo cerca i numeri che possono essere divisori comuni; se ne trova, questi diventano il nuovo MCD. Al termine della ripetizione, l'ultimo valore designato come divisore comune è il MCD.

```
algoritmo CalcolaMCD                    riga di intestazione
variabili                               sezione dichiarativa
      dichiara a, b,
      i, MCD come numeri interi
inizio                                  sezione esecutiva
      immetti a, b
      se a > b
      allora
          scambia a con b
      fine se
      assegna MCD = 1
                - - - esamina i possibili divisori - - -
      per i = 2 fino al valore di a
          se (i è divisore di a) e (i è divisore di b)
          allora
              assegna MCD = i
          fine se
      ripeti
      scrivi MCD
fine
```

Interfaccia utente

Il foglio di calcolo contiene i due numeri di cui si vuole calcolare il MCD. Il pulsante di comando attiva il sottoprogramma che legge i due numeri e li passa alla funzione come parametri.



Codice

L'algoritmo per il calcolo del MCD viene codificato con una *Function*, in modo da avere a disposizione un modulo software che può essere utilizzato in programmi diversi che richiedano questo tipo di calcolo. I valori di *a* e *b* non sono quindi immessi da tastiera, ma passati come parametri alla *Function* dal programma chiamante.

```
Function CalcolaMCD(ByVal a, ByVal b) As Integer
' dichiarazione delle variabili
Dim temp, i, MCD As Integer
' scambia i numeri se non sono ordinati
If a > b Then
 temp = a
 a = b
 b = temp
End If
MCD = 1
' ripetizione sui possibili divisori
For i = 2 To a
 If (a Mod i = 0) And (b Mod i = 0) Then
      MCD = i
 End If
Next i
' assegna il valore calcolato alla funzione
CalcolaMCD = MCD
End Function
```

All'inizio della funzione si effettua lo scambio tra i due parametri, utilizzando una variabile temporanea, nel caso in cui il primo numero sia maggiore del secondo:

```
temp = a
a = b
b = temp
```

Per controllare se un numero è divisore, si controlla che il resto della divisione sia 0, utilizzando l'operatore **Mod** che calcola il resto di una divisione intera. Poiché il divisore deve essere comune la condizione è composta con l'operatore **And**:

```
If (a Mod i = 0) And (b Mod i = 0) Then
```

Si noti che prima della fine della funzione, il valore calcolato viene assegnato al nome della funzione:

```
CalcolaMCD = MCD
```

Il sottoprogramma, associato al pulsante di comando *Esegui*, chiama l'esecuzione della funzione passando i parametri per valore: nell'intestazione delle funzione, prima dei parametri, è indicata la specifica **ByVal**.
Il passaggio è per valore perché all'interno della funzione le variabili *a* e *b* potrebbero scambiare i loro valori e questo non deve riflettersi sui valori di partenza nel sottoprogramma chiamante.
Nel sottoprogramma il valore restituito dalla funzione *CalcolaMCD* è assegnato alla variabile *num3*.

```
Private Sub Esegui_Click()
Dim num1, num2, num3 As Integer
num1 = Range("A2")
num2 = Range("B2")
' chiama la funzione passando i parametri
num3 = CalcolaMCD(num1, num2)
Range("C2") = num3
End Sub
```