Appendice

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

Matrici di Parametri

- In VB è possibile richiamare Sub o Function con un numero di parametri non fissato a priori usando le matrici di parametri
  - Il parametro è preceduto dalla parola chiave **`ParamArray`** ed è di tipo **`Variant`**
    - È un vettore che parte sempre da 0
  - è sempre l'ultimo elemento di una lista di parametri

Matrici di Parametri

```
Option Explicit
Function argoVar(uno As Integer, _
                    ParamArray vari() As Variant)
  Dim i As Integer, s As String
  s = ""
  For i = LBound(vari) To UBound(vari)
     s = s & i & "  " & vari(i) & vbNewLine
  Next
  MsgBox ("uno vale " & uno & vbNewLine & s)
  argoVar = UBound(vari)
End Function
Sub usaArgVar()
   Dim uno As Integer, due As Integer
   uno = argoVar(8, "wer", 890, 34.78)
   due = argoVar(800, 0.78, "casa")
   MsgBox ("-> " & uno & vbNewLine & " --> " & due)
End Sub
```


Esempio

```
Option Explicit
Option Base 1
Sub conta(ParamArray v() As Variant)
    Dim i As Integer
    'paramarray parte da 0 Sempre!
    For i = LBound(v) To UBound(v)
       Cells(2, i + 1) = v(i)
    Next
End Sub
Sub prv()
   Call conta(5, 34, 6, 7, "wrrew")
End Sub
```

Riferirsi ad intervalli di celle

- Per le routine può essere necessario riferirsi a blocchi di celle di un foglio di lavoro
  - Il parametro è di tipo **`Range`**
  - Si scorre l'intervallo con il ciclo **`for each`**

Riferirsi ad intervalli di celle

```
Option Explicit
Function interv(inte As Range) As Double
  Dim q As Variant

  interv = 0
  For Each q In inte
    If IsNumeric(q) Then
      interv = interv + q
    End If
  Next
End Function

Sub total()
   Range("F1").Value = interv(Range("a1:d5"))
   Range("F2") = interv(Range("a1"))
End Sub
```

Riferirsi ad intervalli di celle

- Più complessa è la seguente modalità
  - Per avere una interazione corretta basta ricordare:
    - Un intervallo di celle è SEMPRE una matrice bidimensionale
    - Una singola cella non è una matrice
      - Il predicato **`IsArray(`*`variabile`*`)`** restituisce **`True`** se la variabile è una matrice
    - Il parametro che indica l'intervallo va passato come **`Variant`** e va poi assegnato ad una variabile locale di tipo **`Variant`**

Riferirsi ad intervalli di celle

```
Option Base 1
Sub passa(xp As Variant)
  Dim X As Variant
  X = xp

  MsgBox (X)
End Sub

Sub test()
  passa (Cells(1, 2))
End Sub
```

Una sola cella va trattata come variabile

```
Option Base 1
Sub passa(xp As Variant)
  Dim X As Variant
  X = xp

  MsgBox (X(1, 1))
End Sub

Sub test()
 passa (Range(Cells(1, 1),_
              Cells(1, 2)))
End Sub
```

Intervallo di celle: va trattato come matrice

Riferirsi ad intervalli di celle

```
Function interv(inte As Variant) As Double
  Dim X As Variant
  Dim i As Integer, j As Integer
  X = inte
  interv = 0
  If IsArray(X) Then
    For i = LBound(X) To UBound(X)
      For j = LBound(X, 2) To UBound(X, 2)
        If IsNumeric(X(i, j)) Then
           interv = interv + X(i, j)
        End If
      Next
    Next
  Else
    interv = X
  End If
End Function

Sub total()
   Range("F1").Value = interv(Range("a1:d5"))
   Range("F2") = interv(Range("a1"))
End Sub
```

Si controlla se è stata passata una cella o un intervallo

Valutare Espressioni

- Se una stringa contiene una formula sintatticamente corretta per Excel con una variabile si può valutare la formula per diversi valori della variabile creando una tabella
  - Si deve usare la proprietà `Formula` di una cella insieme alla funzione `Replace`
    - Nella formula i numeri debbono avere come separatore il punto decimale
    - La variabile da sostituire è meglio che non sia un carattere o una sequenza di caratteri che può essere contenuta in funzioni di libreria
      - » Preferire ad esempio **_x** ad **x** (x è contenuto anche in funzioni come `Exp( )`)

Esempio

- La cella A1 contiene una formula ben formata per Excel con l'incognita **_x**
- La colonna B, a partire da B2, ha un elenco di valori per **_x**
- Scrivere nella colonna C il valore della formula dopo l'opportuna sostituzione di **_x** con il valore in B

Esempio

```
Option Explicit

Sub prova()
   Dim rg As Integer
   Dim Form As String

   Form = Cells(1, 1)
   rg = 2
   Do While Not IsEmpty(Cells(rg, 2))
      Cells(rg, 3).Formula = "=" & Replace(Form, "_x", Cells(rg, 2))
      rg = rg + 1
   Loop
End Sub
```

	A	B	C
1	3*_x+Exp(_x+2)	_x	
2		1,00	23,09
3		-2,00	-5,00
4		4,00	415,43
5		-3,00	-8,63

Valutazione di una Espressione

- Supponiamo che
  - la cella A2 contenga una formula con una incognita _x
  - La cella B2 il primo valore per _x
  - La cella C2 l'ultimo valore per _x
  - La cella D2 il passo
- Il codice nella pagina che segue permette di tabulare tutti i valori della formula

Valutazione di una Espressione

```
Sub Tabula()
   Dim x As Double, y As Double, iniX As Double
   Dim finX As Double, passo As Double
   Dim formula As String, i As Integer

   formula = Cells(2, 1).Value
   formula = Replace(formula, ",", ".")

   iniX = Cells(2, 2).Value
   finX = Cells(2, 3).Value
   passo = Cells(2, 4).Value
   i = 0

   For x = iniX To finX Step passo
     Cells(4 + i, 2).Value = x
    Cells(4 + i, 3).formula = "=" & _
           Replace(formula, "_x", _
                        CStr(Replace(x, ",", ".")))
    Cells(4 + i, 3).NumberFormat = ".##"
      i = i + 1
   Next
End Sub
```

Due cifre decimali

Seleziona il tipo di formato numerico

Esempio

– Scrivere il programma che legge i valori dal file dati.txt, li scrive nel foglio corrente a partire dalla colonna A e quindi su ciascuno applica la formula contenuta nella cella C1 (dove la variabile si chiama _x) e scrive nella colonna B il valore ottenuto sostituendo nella formula in C1 il valore in A

Esempio

```
Option Explicit
Sub leggiTabula()
   Dim val As Double, i As Integer
   Dim fml As String, fmlS As String

   fml = Range("C1").Value
   Open ThisWorkbook.Path & "\dati.txt" For Input As #1

   i = 1
   While Not EOF(1)
     Input #1, val

     Cells(i, 1) = val
     fmlS = Replace(fml, "_x", Replace(CStr(val), ",", "."))
     Cells(i, 2).Formula = "=" & fmlS

     i = i + 1
   Wend

   Close #1
End Sub
```

Oggetti

- Gli oggetti sono le componenti fondamentali di VB(A)
- Un oggetto è un contenitore di variabili, funzioni e subroutine che perseguono uno scopo comune
  - Gli oggetti sono all'interno di una gerarchia
- L'insieme è un oggetto che contiene al suo interno altri oggetti solitamente dell stesso tipo
  - L'insieme **`Workbooks`** di Microsoft Excel contiene, ad esempio, tutti gli oggetti **`Workbook`** aperti.

Oggetti

- I dati su cui lavorano gli oggetti sono detti **proprietà**.
- Le funzioni contenute in un oggetto sono dette **metodi**.
- Alle proprietà ed ai metodi di un oggetto si accede attraverso la variabile di tipo oggetto seguita dal carattere punto (`.`).
  - Ad esempio l'oggetto `ActiveCell` indica la cella al momento attiva, contiene fra gli altri
    - Le proprietà `Address`, `Formula` e `Value`
    - I metodi `Cells()` e `BorderAround()`

Oggetti

```
Sub provaOggetto()
   MsgBox ("cella attiva: " & ActiveCell.Address)
   MsgBox ("la cella contiene: " & ActiveCell.Value)
   MsgBox ("la formula nella cella è: " & _
                                ActiveCell.Formula)

   ActiveCell.BorderAround xlDouble, xlThick, _
                        Color:=RGB(255, 255, 0)
   ActiveCell.Cells(4, 5).Select
'si sposta relativamente alla cella precedente
    ActiveCell.Value = 90
End Sub
```


Oggetti

- L'istruzione **`with`** permette di evitare la ripetizione della catena di oggetti fino ai metodi ed alle proprietà dell'ultimo oggetto:

```
Sub provaOggetto()
 with ActiveCell
   MsgBox ("cella attiva: " & .Address)
   MsgBox ("la cella contiene: " & .Value)
   MsgBox ("la formula nella cella è: " & .Formula)

   .BorderAround xlDouble,xlThick,Color:=RGB(255, 0, 0)
   .Cells(4, 5).Select 'cambia cella attiva
   .Value = 90
 End With
End Sub
```

Oggetti

– Per assegnare ad una variabile o ad una proprietà un oggetto si deve far precedere l'istruzione di assegnamento dalla parola chiave **Set**.

```
Sub usoSet()
 Dim foglio As Worksheet

 Set foglio = Worksheets(1)
 foglio.Name = "ancora"

End Sub
```


Oggetto **Range**

- E' un oggetto particolarmente importante in VBA
  - le singole celle ed i gruppi di celle sono di tipo **Range**.
  - Alcune proprietà
    - **Rows** indica tutte le righe di **Range**
    - **Columns** indica tutte le colonne di **Range**

Oggetto **Range**

```
Option Explicit

Sub modificaRiga(rn As Range, val As Variant)
  Dim Cell As Range
  For Each Cell In rn.Rows
    Cell = val
  Next
End Sub

Sub richiama()
  modificaRiga Range("A1..D5"), 3
  Call modificaRiga(Range("E2", "G5"), "ciao")
End Sub
```


Oggetto **Range**

```
Option Explicit
Sub modificaRiga(rn As Range)
  Dim righe As Range, i As Integer

  Set righe = rn.Rows
  For i = 1 To righe.Count
    righe.Item(i) = -5
  Next
End Sub

Sub richiama()
  modificaRiga Range("A1..D5")
End Sub
```

Assegna le righe alla variabile righe

Indica il numero di celle nella riga

`Item` proprietà che permette di scorrere sugli elementi

	A	B	C	D
1	-5,00	-5,00	-5,00	-5,00
2	-5,00	-5,00	-5,00	-5,00
3	-5,00	-5,00	-5,00	-5,00
4	-5,00	-5,00	-5,00	-5,00
5	-5,00	-5,00	-5,00	-5,00

Moduli e Controlli

- In alcuni casi è utile facilitare l'utente nell'inserimento dei dati questo può avvenire attraverso i **moduli** (***Form***) e attraverso i **controlli** in essi contenuti.
  - I **controlli** sono degli oggetti contenuti in un oggetto di tipo **Form** o in un foglio di lavoro (p. es. i bottoni) e possono associare a dei precisi eventi (schiacciare un tasto, schiacciare il mouse, ...) una funzione appositamente scritta.

Uso di controlli

- Per usare i controlli attivare la barra dei controlli:
  - Visualizza -> Barre degli Strumenti -> Strumenti di Controllo

Modalità Progettazione

Visualizzatore Proprietà

Visualizzatore Codice

Controlli Standard (nell'ordine):
- Casella controllo
- Casella di testo
- Pulsante Comando (Bottone)
- Pulsante di opzione
- Casella riepilogo
- Casella combinata
- Interruttore
- Pulsante di selezione
- Barra di scorrimento
- Etichetta
- Immagine

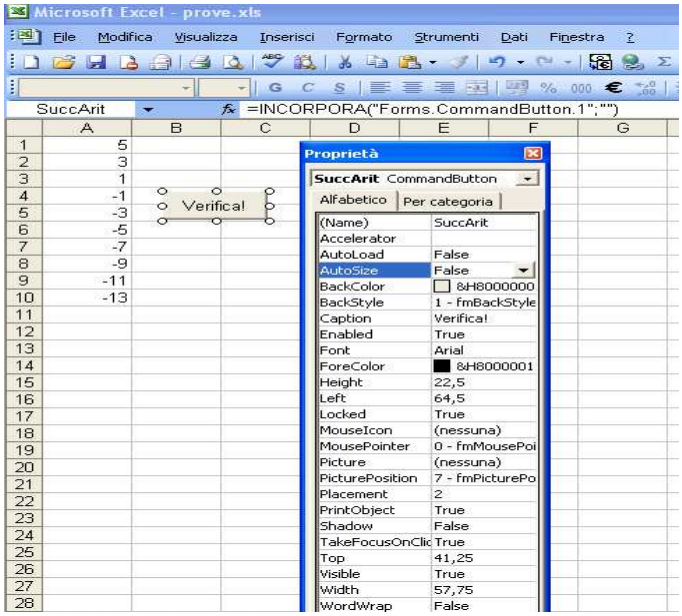
Altri Controlli

Moduli e Controlli

– Fra i controlli quello più utile da inserire direttamente in un foglio Excel è il *`Pulsante di comando`* che può attivare una routine per effettuare dei controlli o per elaborare dei valori presenti nel foglio.
  - Creare un bottone che permette di verificare se i valori presenti nel foglio elettronico dalle cella A1 alla cella A10 sono in progressione aritmetica (la differenza fra due valori consecutivi è costante). L'esito della verifica è scritto nella cella B1

Moduli e Controlli

- Innanzitutto nel foglio di lavoro
  - Si attiva la modalità progettazione
  - si sceglie il controllo bottone e si posiziona il mouse dove si desidera che sia inserito
- Con il tasto destro si seleziona la voce proprietà e si modificano quelle desiderate, ad esempio:
  - **`(Name)`**: il nome dell'oggetto in VBA
  - **`Caption`**: indica il testo contenuto nel bottone
- Si schiaccia due volte il bottone si apre l'editor di VBA su di una Sub chiamata con il nome immesso nel campo **`(Nome)`** seguita da **`_Click()`**
  - Questo è il codice che sarà richiamato quando si schiaccia sul bottone
- Per attivare il bottone bisogna uscire dalla modalità progettazione



Moduli e Controlli

```
Private Sub SuccArit_Click()
   Dim diff As Integer, x As Range
   Dim progAr As Boolean, prec As Integer

   progAr = True

   diff = Range("A1") - Range("A2")
   prec = Range("A2").Value
   For Each x In Range("A3..A10")
     If (prec - x.Value <> diff) Then
        progAr = False
     End If
     prec = x.Value
   Next

   If progAr Then
     Range("B1") = "in progressione aritmetica"
   Else
     Range("B1") = "non in progressione aritmetica"
   End If

End Sub
```

Moduli e Controlli

```
Private Sub Svuotamento_Click()
   Dim x As Range

   For Each x In Range("A1..D25")
      x.Value = ""
   Next
End Sub
```

In un foglio elettronico far richiamare con un pulsante una funzione che svuota le celle da A1 a D25

Moduli e Controlli

- In un foglio elettronico si desiderano calcolare i valori della funzione tasso di Poisson (K) e C di Erlang a partire
  - dai valori N > 0, numero di servitori
  - 0<=ρ<=1 utilizzazione usando le formule:

$$K = \frac{\sum_{h=0}^{N-1} \frac{(N\rho)^h}{h!}}{\sum_{h=0}^{N} \frac{(N\rho)^h}{h!}} \qquad C = \frac{1-K}{1-\rho K}$$

Moduli e Controlli

```
Private Sub Calcolo_Click()
   Dim N As Integer, rho As Double, h As Integer
   Dim prec As Double, num As Double, den As Double


   'verifica coerenza
    If IsNumeric(Range("B9").Value) = False Then
     MsgBox ("fornire un valore numerico per N!")
     Range("B9").Value = ""
     Exit Sub
    Else
     If Int(Range("B9").Value) <> Range("B9").Value Then
       MsgBox ("fornire un numero intero per N!")
       Range("B9").Value = ""
       Exit Sub
     End If
    End If

    N = Range("B9").Value
    If N <= 0 Then
      MsgBox "Il valore di N deve essere positivo", vbCritical
      Range("B9").Value = ""
      Exit Sub
    End If
```

Il valore di N in B9
di rho in B10
di K in B14
di C in B15

Moduli e Controlli

```
   If IsNumeric(Range("B10").Value) = False Then
     MsgBox ("fornire un valore numerico per rho!")
     Range("B10").Value = ""
     Exit Sub
    End If
    rho = Range("B10").Value
    If rho <= 0 Or rho >= 1 Then
      MsgBox "rho fra 0_ ed 1 E.E", vbCritical
      Range("B10").Value = ""
      Exit Sub
    End If

    'calcolo K
    prec = 1
    num = 1
    For h = 1 To N - 1
      prec = prec * N * rho / h
      num = num + prec
    Next
    den = num + prec * N * rho / N

    Range("B14").Value = num / den
    Range("B15").Value = (1 - Range("B14").Value) / (1 - Range("B14").Value * rho)

End Sub
```

Il valore di N in B9
di rho in B10
di K in B14
di C in B15

Moduli e Controlli

- Il *modulo* è un metodo comodo per permettere la selezione e la validazione di un certo numero di voci
- Per creare un *modulo* (*UserForm* o *Form*) occorre:
  - Entrare nell'editor VBA
  - Selezionare la voce UserForm dal menù Inserisci
    - In alternativa si può selezionare l'icona
  - Aggiungere i controlli desiderati

Moduli e Controlli

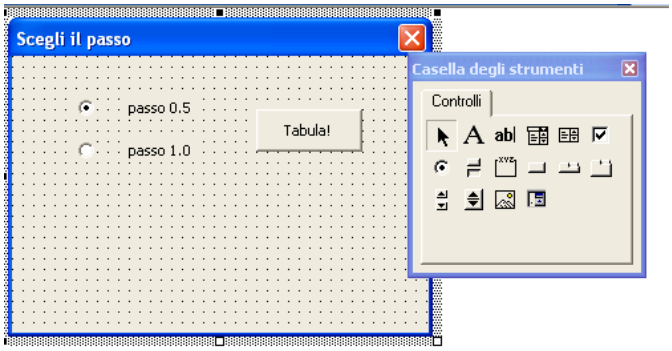
- Vediamo attraverso alcuni esempi i principali controlli e alcune loro proprietà
  - I moduli saranno richiamati tramite
    - un pulsante di comando
    - Una (nuova) macro di Excel selezionabile da Strumenti->Macro

Moduli e Controlli

– Costruiamo un semplice esempio di Modulo usato per creare una sequenza di 10 valori ordinati a partire di valore 0 con un incremento scelto dall'utente.
  - Useremo una UserForm contenente
    – Due **OptionButton** uno per ciascuno dei possibili passi (0,5; 1)
      » Selezionando l'opzione si attiva il passo
      » Il primo valore è quello usato in caso di mancata scelta
    – Due **Label** con la legenda di ogni opzione
      » Schiacciando sull'etichetta si attiva il passo scelto
    – Un pulsante di comando per attivare la creazione della sequenza a partire dalla cella attiva

Moduli e Controlli

– Con l'editor dei form si crea la seguente situazione:



Moduli e Controlli

```
Option Explicit
Dim Passo As Double

Private Sub Label1_Click()
  Passo = 0.5
  OptionButton1.Value = True
End Sub

Private Sub Label2_Click()
  Passo = 1#
  OptionButton2.Value = True
End Sub

Private Sub OptionButton1_Click()
  Passo = 0.5
End Sub

Private Sub OptionButton2_Click()
  Passo = 1#
End Sub
```

```
Private Sub Tabulazione_Click()
  Dim i As Double, j As Integer

  part = 0
  If Passo = 0 Then
   Passo = 0.5
  End If
  i = 0
 For j = 1 To 10
  ActiveCell.Offset(j - 1, 0).Value = i
  i = i + Passo
 Next j
  Me.Hide
End Sub

Private Sub UserForm_Click()
  Me.Hide
End Sub
```


Moduli e Controlli

- Le subroutine che terminano con *`_Click()`* si attivano quando si schiaccia/seleziona l'oggetto
- Al termine dell'esecuzione si deve chiudere il modulo con l'istruzione `Me.hide`
  - **`Tabulazione_click()`** è richiamata dal pulsante di comando con scritto *Tabula!*
  - **`UserForm_Click()`** è richiamata quando si seleziona il bottone chiudi del UserForm
- Per poter utilizzare la user form si crea nella sezione Moduli del progetto VBA la sub:

```
Sub Tabula()
  UserForm1.Show
End Sub
```

1

Moduli e Controlli

- Creiamo una UserForm per scegliere fra tre tipi di funzioni e quindi prelevando il valore dalla cella attiva si applica a questo la funzione scelta e si scrive il risultato nella cella accanto
  - Si utilizza una ComboBox
    - Si crea l'elenco delle voci in un foglio di lavoro compreso nel file Excel si dà un nome all'elenco (Inserisci->Nome->crea)
    - si mette questo nome nella proprietà RowSource
    - Infine si scrive il codice da utilizzare quando l'utente sceglie (Sub ComboBox1_Change())

Moduli e Controlli

```
Option Explicit
Private Sub ComboBox1_Change()
   If Me.ComboBox1.Value = 0 Then
      ActiveCell.Offset(0, 1).Value = Sin(ActiveCell.Value)
   Else
     If Me.ComboBox1.Value = 1 Then
        ActiveCell.Offset(0, 1).Value = Cos(ActiveCell.Value)
     Else
        ActiveCell.Offset(0, 1).Value = Tan(ActiveCell.Value)
     End If
   End If
   Me.ComboBox1.Value = ""
   Me.Hide
End Sub
Private Sub UserForm_Click()
    Me.ComboBox1.Value = ""
    Me.Hide
End Sub
```

Diapositiva 41

l1 RIVEDI IL FUNZIONAMENTO!
luca; 13/12/2010

Moduli e Controlli

– Creiamo un esempio nel quale dai dati contenuti in una tabella calcoliamo
  - Coefficiente di correlazione lineare

$$r = \frac{n \cdot \sum(x \cdot y) - (\sum x) \cdot (\sum y)}{\sqrt{(n \cdot \sum x^2 - (\sum x)^2) \cdot (n \cdot \sum y^2 - (\sum y)^2)}}$$

  - Quindi se è compreso fra -1 e -0,3 o fra 0,3 ed 1 calcoliamo i valori per la retta di interpolazione lineare $y=a_0+a_1x$

$$a_0 = \frac{\sum(x) \cdot \sum(x^2) - \sum(x) \cdot \sum(x \cdot y)}{(n \cdot \sum x^2 - (\sum x)^2)} a_1 = \frac{n \cdot \sum(x \cdot y) - \sum(x) \cdot \sum(y)}{(n \cdot \sum x^2 - (\sum x)^2)}$$

Moduli e Controlli

```
option Explicit
Private Sub calcoloCoeff_Click()
   Dim r As Double, a0 As Double, a1 As Double
   Dim Sx As Double, Sy As Double, Sx2 As Double, Sy2 As Double
   Dim Sxy As Double
   Dim i As Integer, j As Integer

   Sx = 0
   Sy = 0
   Sx2 = 0
   Sy2 = 0
   Sxy = 0

   i = 2
   j = 0
   While IsNumeric(Cells(i, 1).Value) And Not IsEmpty(Cells(i, 1))
     Sx = Sx + Cells(i, 1).Value
     Sy = Sy + Cells(i, 2).Value
     Sx2 = Sx2 + Cells(i, 1).Value ^ 2
     Sy2 = Sy2 + Cells(i, 2).Value ^ 2
     Sxy = Sxy + Cells(i, 1).Value * Cells(i, 2).Value
     j = j + 1
     i = i + 1
   Wend
```

Moduli e Controlli

```
  r = (j * Sxy - Sx * Sy)/((j * Sx2 - Sx ^ 2) * (j * Sy2 - Sy ^ 2))^(0.5)

  If r >= -0.3 And r <= 0.3 Then
    MsgBox ("non esiste correlazione far i dati")
  Else
    a1 = (Sx * Sx2 - Sx * Sxy) / (j * Sx2 - Sx ^ 2)
    a0 = (j * Sxy - Sx * Sy) / (j * Sx2 - Sx ^ 2)
    For i = j + 1 To 2 Step -1
      Cells(i, 3).Value = Cells(i, 1).Value * a1 + a0
    Next
    Cells(3, 6).Value = r
    Cells(4, 6).Value = a0
    Cells(5, 6).Value = a1
  End If

End Sub
```


Aggiungere funzioni

- E' possibile aggiungere alle funzioni predefinite in Excel e richiamabili nel foglio elettronico delle funzioni scritte dall'utente
  - Le nuove funzioni vanno aggiunte in un modulo dall'editor di VBA:
    - Inserisci->Modulo
  - Si scrivono le funzioni che compariranno nell'insieme delle funzioni selezionabili sotto la voce Definite dall'utente

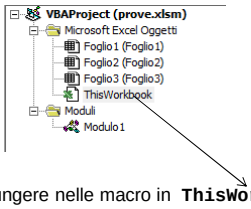
Esempio

```
Option Explicit
'restituisce quanti valori positivi sono in r
Function quantiPos(r As Range)
  Dim X As Variant

  quantiPos = 0
  For Each X In r
        If Not IsEmpty(X) Then
          If IsNumeric(X) Then
            If X > 0 Then
             quantiPos = quantiPos + 1
            End If
          End If
        End If
  Next

End Function
```

	A	B	C	D	E	F	G
1	-5,00	-5,00	-5,00	-5,00		cvnvbmn	
2	-5,00	-5,00	-5,00	-5,00			
3	-5,00	-5,00	-5,00	-5,00	234,00		
4	-5,00	-5,00	-5,00	-5,00			1,00
5	-5,00	-5,00	-5,00	-5,00			



Per inserire la descrizione della funzione occorre aggiungere nelle macro in **ThisWorbook** la funzione **WorkBook_Open()che è** attivata quando si apre il file (qui si possono aggiungere i comandi da eseguire all'apertura del file):

```
Sub WorkBook_Open()

    Application.MacroOptions Macro:="quantiPos", Description:="Conta i numeri positivi nell'intervallo"

End Sub
```


Aggiungere funzioni

– Creare la funzione progGeom che ha come argomento un intervallo di valori e restituisce True se i valori in esso contenuto sono in progressione geometrica (il rapporto fra due valori consecutivi è costante), altrimenti False

```
Option Explicit
Function progGeom(r As Range) As Boolean
  Dim x As Range, i As Integer

  progGeom = True
  If (r.Count > 2) Then
    For i = 1 To (r.Count - 3)
      If (r.Item(i).Value / r.Item(i + 1).Value _
           <> r.Item(i + 1).Value / r.Item(i + 2).Value) Then
         progGeom = False
      End If
    Next
  End If
End Function
```

Aggiungere Funzioni

```
Function progArit(r As Range) As Boolean
  Dim i As Integer
  progArit = True

  With r
   If (.Count > 2) Then
    For i = 1 To (r.Count - 3)
      If (.Item(i).Value - .Item(i + 1).Value _
             <> .Item(i + 1).Value - .Item(i + 2).Value) Then
        progArit = False
      End If
    Next
   End If
  End With

End Function
```


Un appunto sui colori

- I colori si ottengono combinando i tre colori base della sintesi sottrattiva:
  - Rosso (Red)
  - Verde (Green)
  - Blu (Blue)
- Si usa la funzione `RGB(RR, VV, BB)` che ha come argomento le quantità di ciascuno dei tre colori base mediante un valore fra 0 e 255