Unterprogramme und Funktionen

- Idee: Komplexere Aufgaben in Teilaufgaben untergliedern
- Dafür bietet VBA zwei Mechanismen:
  - Unterprogramme (**Sub** ...)
  - Funktionen (**Function** ...)

Unterprogramme (Subs)

- Unterprogramme haben Sie schon kennen gelernt:

```
Sub Name()
  ...
End Sub
```

  definiert das Unterprogramm (sub procedure) namens **Name**.
- Solche „Sub"s können Sie aus dem Makro-Editor heraus mit [F5] starten – aber auch aus anderen Makros heraus

Unterprogramme – verschachtelt

- Betrachten Sie den folgenden Code:

```
Sub HalloAusgeben()
  Debug.Print "Hallo"
End Sub

Sub TestMakro()
  Call HalloAusgeben
  Call HalloAusgeben
End Sub
```

- Wenn Sie **TestMakro** starten, erscheint:

```
Hallo
Hallo
```


Unterprogramme mit Argumenten

- Unterprogramme beim Aufruf noch beeinflussen mit „Argumenten" bzw. „Aufrufparametern"

```
Sub HalloAusgeben(nAnzahl As Integer)
  Dim i As Integer
  For i = 1 To nAnzahl
    Debug.Print "Hallo"
  Next
End Sub

Sub TestMakro()
  Call HalloAusgeben(3)
End Sub
```

Ausgabe:

```
Hallo
Hallo
Hallo
```

Übung (auf Papier!)

Was passiert, wenn Sie das Makro **`TestMe()`** aufrufen?

```
Sub ZweiteStufe()
  Debug.Print "ZweiteStufe"
End Sub

Sub ErsteStufe(n As Integer)
  Debug.Print "Anfang ErsteStufe"
  For i = 1 To n
    Call ZweiteStufe
  Next
  Debug.Print "Ende ErsteStufe"
End Sub

Sub TestMe()
  Debug.Print "Anfang"
  Call ErsteStufe(2)
  Call ZweiteStufe
  Debug.Print "Ende"
End Sub
```


Argumente an Sub übergeben (1)

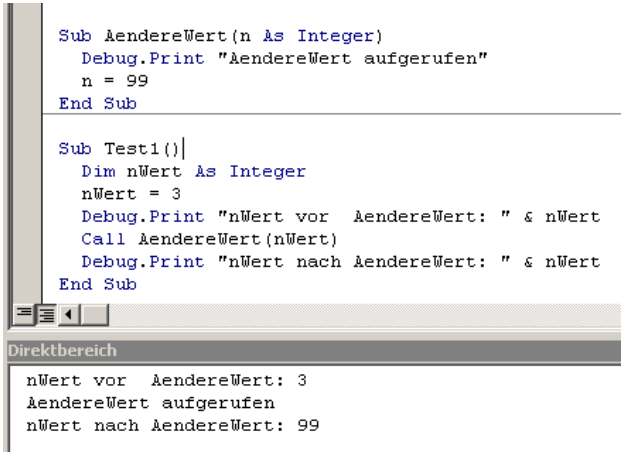
- Was passiert mit übergebenen Argumenten?

```
Sub AendereWert(n As Integer)
  Debug.Print "AendereWert aufgerufen"
  n = 99
End Sub

Sub Test()
  Dim nWert As Integer
  nWert = 3
  Debug.Print "nWert vor  AendereWert: " & nWert
  Call AendereWert(nWert)
  Debug.Print "nWert nach AendereWert: " & nWert
End Sub
```

- Welche Ausgabe erwarten Sie? 2x „3“?

Argumente an Sub übergeben (2)



- verwirrend?

Call by Reference

- Bei einem Sub-Aufruf der Form
  `Call Prozedur (Variable)`
  erhält die Prozedur die Adresse (reference) der als Parameter vergebenen Variable im Speicher
- Diese Art von Variablenübergabe nennt man **Call by Reference** (kurz: **ByRef**)
- Änderungen an der Variablen sind anschließend im aufrufenden Code sichtbar

Call by Value

- Um das Verändern von Variablen in Unterprozeduren zu vermeiden, Variablen nur als Wert übergeben
- Das nennt sich **Call by Value** (kurz: **ByVal**)
- Sub-Definition mit Schlüsselwort „ByVal“:

```
Sub AendereWertNicht(ByVal n As Integer)
  Debug.Print "AendereWertNicht aufgerufen"
  n = 99
End Sub
```

- VBA erzeugt dann „Kopie“ der Variablen

Call by Value

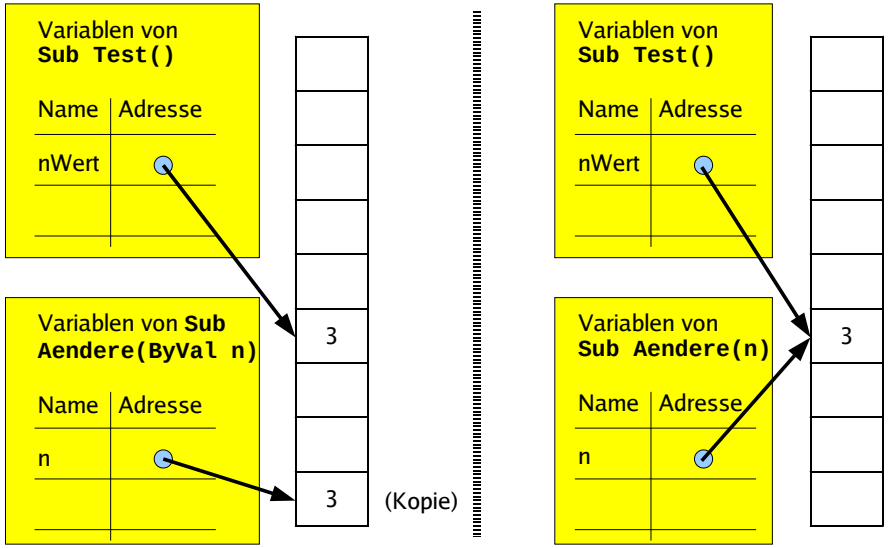
```
Sub AendereWertNicht(ByVal n As Integer)
  Debug.Print "AendereWertNicht aufgerufen"
  n = 99
End Sub

Sub Test1()
  Dim nWert As Integer
  nWert = 3
  Debug.Print "nWert vor  AendereWertNicht: " & nWert
  Call AendereWertNicht(nWert)
  Debug.Print "nWert nach AendereWertNicht: " & nWert
End Sub
```

```
Direktbereich
nWert vor  AendereWertNicht: 3
AendereWertNicht aufgerufen
nWert nach AendereWertNicht: 3
```

- jetzt keine Änderung!

Call by Value / Call by Reference



Call by Value / Call by Reference

- Manchmal ist es sinnvoll, wenn eine Prozedur die übergebene Variable verändern kann
  - Dann einfach Sub ohne weitere Angaben definieren:

    ```
    Sub Prozedur (nVariable As Integer)
    ```

  - Oder sogar explizit rein schreiben, dass es „by reference“ ist:

    ```
    Sub Prozedur (ByRef nVariable As Integer)
    ```

- Andernfalls nur Werte („by value“) übergeben:

  ```
  Sub Prozedur (ByVal nVariable As Integer)
  ```

Funktionen (1/5)

- Neben Subs gibt es noch eine andere Möglichkeit, das Programm zu unterteilen: Funktionen
- Wie in Mathematik: Funktion erwartet ein Argument und gibt ein (berechnetes) Ergebnis zurück
- Beispiel: Quadratfunktion $f(x) = x^2$
- In VBA:

```
Function f(x)
  f = x*x
End Function
```


Funktionen (2/5)

- Funktionen immer auf gleiche Weise definiert:

```
Function FName (Argumente)
  ... Berechnung mit den Argumenten
  Fname = Ergebnis
End Function
```

- Vor Rückgabe des Ergebnisses beliebige Befehle, wie in Sub-Prozedur; also auch Fallunterscheidungen, Schleifen etc. - das ganze Programm
- Mehrere Rückgabe-Anweisungen möglich – erste „gewinnt“

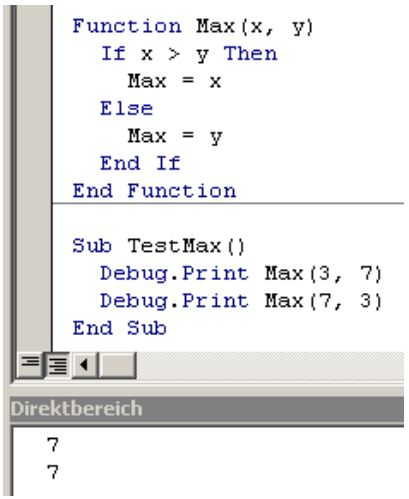
Funktionen (3/5)

- Beispiel: Maximum von zwei Werten berechnen
  - Mathematisch: $\max(x,y) = \begin{cases} x, & x > y \\ y, & x <= y \end{cases}$
  - VBA:

```
Function Max (x,y)
  If x > y Then
    Max = x
  Else
    Max = y
  End If
End Function
```


Funktionen (4/5)

- Funktionen können Sie (anders als Subs) nicht direkt im VBA-Editor mit [F5] starten, und Sie können sie auch nicht im Direktbereich aufrufen
- Zum Testen: Zu jeder Funktion eine Test-Sub schreiben, welche sie aufruft



Funktionen (5/5)

- Achtung: Auch bei Funktionen können Sie Variablen „by Reference“ übergeben, Standard für Funktionen aber „by Value“
- Schlüsselwort „ByRef“ verwenden, wenn Sie das ändern wollen

```
Function MaxMurks(ByRef x, ByRef y)
  If x > y Then
    y = 0
    Max = x
  Else
    x = 0
    Max = y
  End If
End Function

Sub TestMaxMurks()
  x = 3: y = 7
  Debug.Print "x/y: " & x & "/" & y
  Debug.Print "Max: " & MaxMurks(x, y)
  Debug.Print "x/y: " & x & "/" & y
End Sub
```

```
Direktbereich
x/y: 3/7
Max: 7
x/y: 0/7
```


Sub: ByRef

Sub

- Standard:
  Call by Reference
  (übergebene Variablen verändern)
- Definition für Call by Refer.:
  `Sub Name (Var)`
  `Sub Name (ByRef Var)`
- Definition für Call by Value:
  `Sub Name (ByVal Var)`

Function: ByVal

`Function`

- Standard:
  Call by Value
  (übergebene Variablen in Ruhe lassen)
- Definition für Call by Refer.:
  `Function Name (ByRef Var)`
- Definition für Call by Value:
  `Function Name (Var)`
  `Function Name (ByVal Var)`