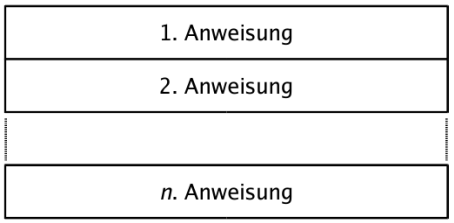
VBA vs. Nassi-Shneiderman

- Auf den folgenden Folien: Elemente aus den Nassi-Shneiderman-Diagrammen in VBA
  - Sequenz (Hintereinanderausführung)
  - Verzweigung (Fallunterscheidung)
  - Iteration (Schleife)

VBA: Sequenzen

- Anweisungen, die in Sequenz ausgeführt werden sollen, in VBA einfach hintereinander schreiben

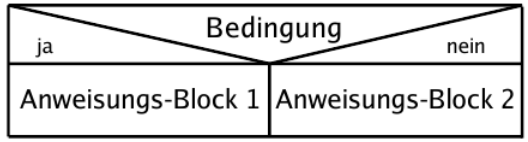
```
Sub TestSequenz ()
  Wert = Cells(1,1)
  Cells(1,2) = Wert
  MsgBox ("Habe Zelle A1
    nach B1 kopiert")
End Sub
```



VBA: Verzweigung: If Then Else

- Fallunterscheidung in VBA mit If-Then-Else:

```
If Bedingung Then
  Block1
Else
  Block2
End If
```

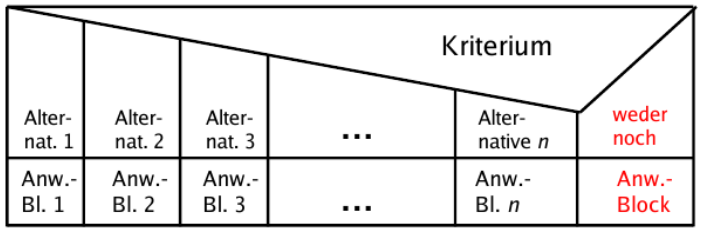



```
x = Int(InputBox("Wert für x"))
If x<3 Then
  y=y+x
  MsgBox ("Neuer y-Wert: " & y)
Else
  y=y+1
End If
```


VBA: Mehrfachverzweigung

- Mehrere Fälle: ElseIf für weitere Tests

```
If Bedingung 1 Then
  Block1
ElseIf Bedingung 2 Then
  Block2
ElseIf Bedingung 3 Then
  Block3
...
ElseIf Bedingung N Then
  Block N
Else
  Block
End If
```



VBA: Schleife (Iteration)

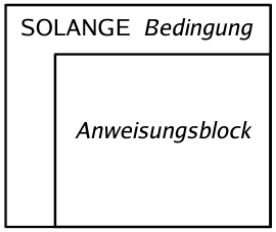
- VBA kennt verschiedene Schleifen
  - **For-Schleife** lässt eine Zählvariable nacheinander verschiedene Werte annehmen:

```
For Zaehler = Anfang to Ende
  Debug.Print Zaehler
Next
```

  - **While-Schleife** läuft solange, wie die Bedingung erfüllt ist

```
Zaehler = 1
While Zaehler < 5
  Debug.Print Zaehler
  Zaehler = Zaehler + 1
Wend
```

Variante 1: vorher prüfen

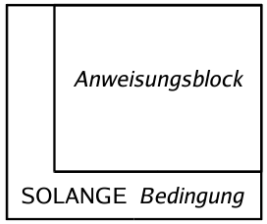


VBA: Schleife (Iteration)

- VBA kennt verschiedene Schleifen
  - **Do-Loop-Until-Schleife** läuft solange, bis Bedingung erfüllt ist

```
Zaehler = 1
Do
  Debug.Print Zaehler
  Zaehler = Zaehler + 1
Loop Until Zaehler >= 5
```

Variante 2: nachher prüfen

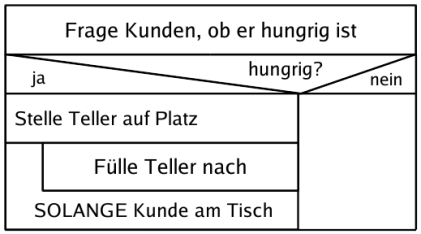


  - **Endlosschleife** läuft ewig – bis Abbruch mit Exit Do

```
Do
  Debug.Print Zaehler
  Zaehler = Zaehler + 1
  If Zaehler >= 5 Then Exit Do
Loop
```


VBA: Beispiel

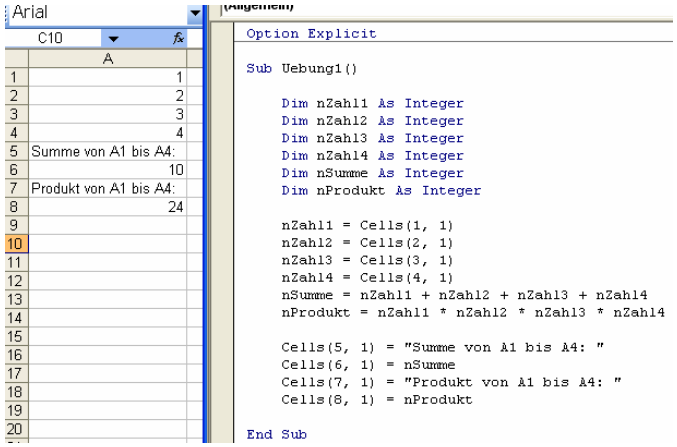
alle zusammen:




```
Sub BedieneKunden ()
  Antwort = InputBox ("Hungrig?")
  If Antwort = "Ja" Then
    StelleTellerAufPlatz
    Do
      FuelleTellerNach
    Loop Until KundeIstWeg
End Sub
```


VBA: Einfache Beispiele

- Makro, das vier Zahlen aus Zellen liest und die Summe sowie das Produkt aller vier Zahlen bildet:



	A
1	1
2	2
3	3
4	4
5	Summe von A1 bis A4:
6	10
7	Produkt von A1 bis A4:
8	24

```
Option Explicit

Sub Uebung1()

    Dim nZahl1 As Integer
    Dim nZahl2 As Integer
    Dim nZahl3 As Integer
    Dim nZahl4 As Integer
    Dim nSumme As Integer
    Dim nProdukt As Integer

    nZahl1 = Cells(1, 1)
    nZahl2 = Cells(2, 1)
    nZahl3 = Cells(3, 1)
    nZahl4 = Cells(4, 1)
    nSumme = nZahl1 + nZahl2 + nZahl3 + nZahl4
    nProdukt = nZahl1 * nZahl2 * nZahl3 * nZahl4

    Cells(5, 1) = "Summe von A1 bis A4: "
    Cells(6, 1) = nSumme
    Cells(7, 1) = "Produkt von A1 bis A4: "
    Cells(8, 1) = nProdukt

End Sub
```

Boole'sche Ausdrücke (1/n)

- Logische Bedingungen lassen sich u.a. mit „UND“ und „ODER“ verknüpfen
  - „Wenn es regnet ODER schneit, muss man vorsichtig fahren.“
  - „Wenn a>2 UND b>2, dann gilt a+b>4.“
- In VBA verknüpfen Sie entsprechend die Wahrheitswerte True und False mit „And“ / „Or“

Boole'sche Ausdrücke (2/n)

- „And“-Verknüpfung („Konjunktion“):

```
nWert1 = Cells (1,1)
nWert2 = Cells (2,1)
If nWert1 > 3 And nWert2 > 3 Then
  MsgBox ("Summe ist >6.")
```

- „Or“-Verknüpfung („Disjunktion“):

```
nWert1 = Cells (1,1)
nWert2 = Cells (2,1)
If nWert1 > 3 Or nWert2 > 3 Then
  MsgBox ("Einer der zwei Werte ist >3.")
```


Boole'sche Ausdrücke (3/n)

- Rechnen mit Wahrheitswerten

```
Print (3<4)
  Wahr
Print Not (3<4)
  Falsch
bBedingung1 = 3<4
Print bBedingung1
  Wahr
bBedingung2 = 3>4
Print bBedingung2
  Falsch
Print bBedingung1 And bBedingung2
  Falsch
Print bBedingung1 Or bBedingung2
  Wahr
```


Boole'sche Ausdrücke (4/n)

- „Negation“: den Wert umkehren mit „Not“

```
bBedingung = ...
If Not bBedingung Then
  ...
End If
```

- „Xor“-Verknüpfung (Entweder-Oder, „exklusives Oder“):
  A Xor B ist genau dann wahr, wenn entweder A oder B gilt (aber nicht beide!)

Boole'sche Ausdrücke (5/n)

- Wahrheitstabellen

AND	Wahr	Falsch
Wahr	Wahr	Falsch
Falsch	Falsch	Falsch

OR	Wahr	Falsch
Wahr	Wahr	Wahr
Falsch	Wahr	Falsch

	NOT
Wahr	Falsch
Falsch	Wahr

XOR	Wahr	Falsch
Wahr	Falsch	Wahr
Falsch	Wahr	Falsch

Übungen

- Für welche Wahrheitswerte A und B werden die folgenden Ausdrücke wahr?
  - NOT (A AND B)
  - A AND (A OR B)
  - (A XOR B) AND NOT (A AND B)
  - A AND (B OR NOT(B)) AND NOT(A)
- Betrachten Sie die Ausdrücke als Funktionen von A und B, also f(A,B), und schreiben Sie die vollständige Funktionstabelle auf

Übungen

- Anleitung für die erste Teilaufgabe:

Gesucht: NOT (A AND B)

A	B	A AND B	NOT(A AND B)
T	T	T	F
T	F	F	T
F	T	F	T
F	F	F	T

Modulo-Operation

- Die Moduloperation Mod
- Ergebnis = Operand1 Mod Operand2
- Gibt den Rest einer ganzzahligen Division zweier Zahlen zurück.
- 4 Mod 3 = 1, 22 Mod 9 = 4
- Zur Verdeutlichung: Eine Zahl, die Mod n gleich 0 ist, ist durch n teilbar!
- Hinweis: Das Ergebnis einer Mod-n-Operation liegt immer zwischen 0 und n-1

Übung zu Modulo-Operation

- Lesen Sie aus A1, A2 und A3 je eine Zahl in eine Variable ein.
- A1 ist der erste, A2 der zweite Operand einer Modulooperation – rechnen Sie also A1 Mod A2 aus.
- Wenn der Inhalt von A3 gerade ist, soll die Ausgabe in einer Messagebox erfolgen, anderenfalls in Zelle A4.
- Die Ausgabe soll in jedem Fall vernünftig lesbar sein, z.B. in der Form „12 Mod 5 ist 2“.
- Verwenden SIe „sprechende Bezeichner“ für Ihre Variablen, z. B. Zahl1

Übung

- Schreiben Sie einen Makro, der den Inhalt der Zellen A1 bis A10 im Arbeitsblatt 2 in Arbeitsblatt 1 in den Zellen A1 bis J1 anzeigt, wenn diese durch 13 ohne Rest teilbar sind und größer als 8 sind. Sollte die Bedingung nicht erfüllt sein, so soll die jeweilige Zelle den Wert "Unpassender Wert" bekommen.
- Tipp: Auf Zellen in anderem Arbeitsblatt zugreifen mit
  `Worksheets(Nummer).Cells(...)`