VBA: „Unsortierte Reste“ + Übungen

Abschluss des Themas VBA-Programmierung:

- Dinge aus verschiedenen VBA-Themenbereichen, die wir bisher ausgelassen hatten
  - String-Funktionen: `InStr, InStrRev, Mid, Left, Right, Len, Str, Val, Chr, Asc`
  - Felder re-dimensionieren: `ReDim, ReDim Preserve`
  - Zufallszahlen: `Rnd`
  - GUI-Anwendungen in Excel erstellen (grober Überblick)
- Beispiel-Aufgaben (teilw. mit Lösungsansätzen)

VBA: String-Funktionen (1/3)

- **`InStr([Start,] Zeichenfolge, Suchbegriff)`** – Position des ersten Auftretens von *Suchbegriff* innerhalb von *Zeichenfolge*
- **`InstrRev(Zeichenfolge, Suchbegriff [, start])`** – Position des letzten Auftretens von *Suchbegriff* innerhalb von *Zeichenfolge*
- **`Mid(Zeichenfolge, Start, Länge)`** – String, der *Länge* Zeichen aus *Zeichenfolge* ab Position *Start* enthält
- **`Left(Zeichenfolge, Länge)`** – Die ersten *Länge* Zeichen der *Zeichenfolge*
- **`Right(Zeichenfolge, Länge)`** – Die letzten *Länge* Zeichen der *Zeichenfolge*
- **`Len(Zeichenfolge)`** – Länge der *Zeichenfolge*

VBA: String-Funktionen (2/3)

- **`Str (n)`** – erzeugt aus der Zahl *n* einen String
- **`Val (s)`** – versucht(!), den String s in eine Zahl umzuwandeln:
  `Val("13")=13, Val("13 Uhr")=13, Val("13:00")=13, aber Val("EUR 13")=0`
- **`Chr (n)`** – gibt das ASCII-Zeichen Nr. *n* zurück:
  `Chr(32)=" ", Chr(65)="A", Chr(98)="b"`
- **`Asc (s)`** – gibt ASCII-Code zum Zeichen s zurück; besteht s aus mehreren Zeichen, dann vom ersten Zeichen:
  `Asc("A")=65, Asc("Aha")=65, Asc("b")=98`

VBA: String-Funktionen (3/3)

Beispiele

```
s="ABCDEFABCDEF123"
print InStr(s, "CDE")
 3
print InStrRev(s, "CDE")
 9
print InStr(s, "Neee")
 0
print Mid (s, 3, 6)
CDEFAB
print Mid (s, 9, 6)
CDEF12
print Left (s, 5)
ABCDE
print Right (s, 5)
EF123
print Len (s)
 15
```

```
s="HAL 345"
For i = 1 To 10 : _
print Chr(Asc(Mid(s,i,1))+1); : _
Next i
IBM!456

' ASCII-Tabelle
For i = 32 To 127 : Print Chr(i); : _
Next
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDE
FGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
lmnopqrstuvwxyz{|}~
```

VBA: Felder re-dimensionieren (1/4)

- Schon bekannt: Felder mit **`Dim`** dimensionieren
  `Dim nWerte(20) As Integer`
  ➞ reserviert 21 (!) Plätze, `nWerte(0)` bis `nWerte(20)`
- Index-Grenzen selber festlegen:
  `Dim nWerte(5 To 20) As Integer`
  ➞ jetzt nur `nWerte(5)` bis `nWerte(20)` möglich, Zugriff auf `nWerte(3)` ist Fehler
- Mit **`ReDim`**-Befehl: Größe des Felds nachträglich ändern oder festlegen

VBA: Felder re-dimensionieren (2/4)

- ReDim hat gleiche Syntax wie Dim, also z. B.
  `ReDim nWerte(20) As Integer`
  und ändert die Dimensionierung eines Felds – aber nur, wenn das Feld ursprünglich entweder
  - als Feld ohne Größenangabe definiert wurde:
    `Dim nWerte() as Integer`
    für dynamische Festlegung der Feldgröße
  - oder bereits mit ReDim statt Dim festgelegt wurde

VBA: Felder re-dimensionieren (3/4)

Beispiele

```
Sub ReDimTest1()
   ReDim n(10) As Integer
   n(10) = 10
   n(9) = 9
   Debug.Print n(0);n(9);n(10)

   ReDim n(20) As Integer
   Debug.Print n(0);n(9);n(10)
End Sub
```

```
 0  9  10
 0  0  0
```

Achtung: Nach ReDim sind alte Werte weg!

```
Sub ReDimTest2()
    Dim nWerte() As Integer
    Dim i, nAnzahl As Integer
    nAnzahl = _
      InputBox("Wie groß ist das Feld?")
    ReDim nWerte(nAnzahl) As Integer
    For i = 1 To nAnzahl
      nWerte(i) = 2 ^ i
      Debug.Print nWerte(i);
    Next i
End Sub
```

```
 2  4  8  16  32  64  128  256
```

Nach Eingabe von „8“ in InputBox

VBA: Felder re-dimensionieren (4/4)

- Wenn Sie Feldinhalte bei **`ReDim`** behalten wollen, verwenden Sie zusätzlich das Schlüsselwort **`Preserve`**:

```
Sub ReDimTest1()
   ReDim n(10) As Integer
   n(10) = 10
   n(9) = 9
   Debug.Print n(0);n(9);n(10)

   ReDim Preserve n(20) As Integer
   Debug.Print n(0);n(9);n(10)
End Sub
```

➞

```
 0  9  10
 0  9  10
```

Bei ReDim Preserve bleiben die alten Werte erhalten

VBA: Zufallszahlen

- **Rnd**: erzeugt Zufallszahl zwischen 0 und 1 (aber ≠ 1)

```
Print Rnd
 0,7055475
```

- Um Integer-Zufallszahl zwischen 1 und 10 zu erzeugen:

```
Print Int(10*Rnd)+1
 5
```

Warum?

  - **10*Rnd** liefert Zahl zwischen 0 und 10, aber ≠ 10;
  - **Int** schneidet alle Nachkommastellen ab

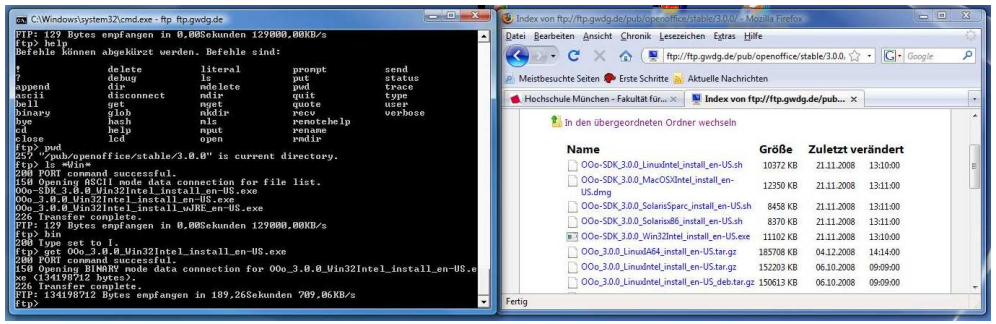
VBA: GUI-Programmierung (1/7)

- In VBA können Sie auch einfache GUI-Programme erstellen
  (**GUI** = **G**raphical **U**ser **I**nterface)
- Im Rahmen dieser Veranstaltung: Nur Begriffsklärung

VBA: GUI-Programmierung (2/7)

- Allgemeiner GUI-Begriff:
  - praktisch alle Anwendungen, die Sie unter Windows verwenden, sind GUI-Programme, laufen also in Programmfenstern und haben grafische Steuerelemente (Buttons, Scroll-Leisten, aufklappende hierarchische Menüs etc.)
  - Gegenkonzept: Textmodusanwendungen, d. h., Programme, die (unter Windows) in der **Eingabeaufforderung** (DOS-**Kommandozeileninterpreter**) benutzt werden

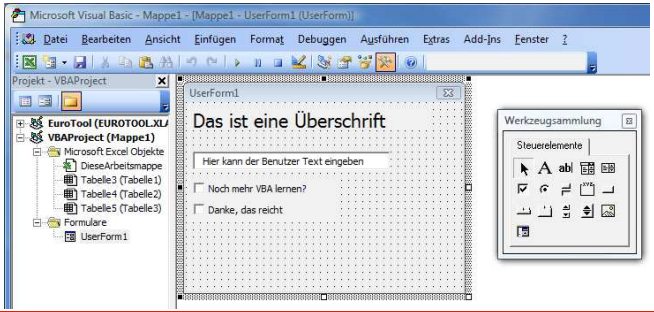
VBA: GUI-Programmierung (3/7)



- links: FTP-Programm (ftp.exe) im Kommandozeilenfenster;
- rechts: FTP-Ansicht im Webbrowser Firefox (einem GUI-Programm)

VBA: GUI-Programmierung (4/7)

- GUI unter Excel:
  - neue Dialogbox („User Forms“) über *Einfügen / UserForm* im VBA-Fenster erzeugen
  - über Werkzeugsammlung neue GUI-Elemente:

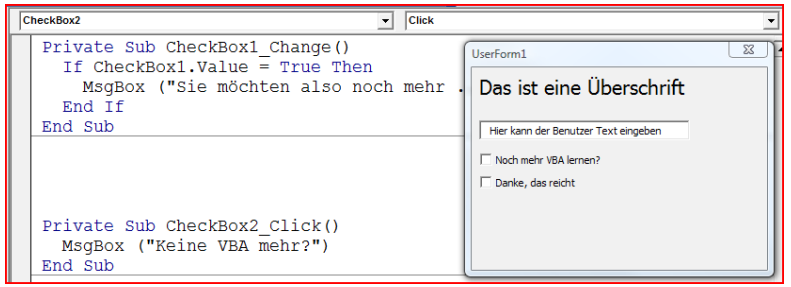


VBA: GUI-Programmierung (5/7)

- alle Elemente in GUI-UserForm sind frei programmierbar:
  - was tun, wenn ein Häkchen gesetzt/entfernt wird
  - was tun, wenn Inhalt eines Textfelds geändert wird
  - was tun, wenn eine Schaltfläche geklickt wird?
- zu jedem solchen **Ereignis** schreiben Sie eine **Ereignisprozedur** in VBA
  - das ist im Prinzip eine normale VBA-Sub-Prozedur, deren Namen VBA vorgibt, z. B. CheckBox_Change()
  - wird ausgeführt, wenn Ereignis „eintritt“

VBA: GUI-Programmierung (6/7)

- Beispiel: Schaltfläche „Noch mehr VBA lernen“
  - Doppelklick auf Schaltfläche öffnet Code-Fenster
  - rechts oben „Change“ (Änderung) auswählen
  - dann neue Prozedur „CheckBox1_Change()“ mit Inhalt füllen



VBA: GUI-Programmierung (7/7)

- Ereignisprozeduren
  - einige dieser Prozeduren sind mit (festen!) Parametern definiert, z. B. `Checkbox_Exit()`:

```
Private Sub Checkbox2_Exit(ByVal Cancel As
                           MSForms.ReturnBoolean)
  ...
End Sub
```

  - an diesen Definitionen dürfen Sie nichts ändern; wenn das Ereignis stattfindet, wird ein Wert übergeben, den die Prozedur auswerten kann
  - Wie normale Subs haben auch diese keinen Rückgabewert

Übungen: 1. Diagonale (1/2)

Eine neue Excel-Tabelle soll auf ihrer Diagonale von A1 über B2, D3, E4 usw. mit Werten besetzt werden, die jeweils doppelt so groß sind wie ihr jeweiliger Vorgänger.
Falls in A1 kein numerischer Wert steht soll dort eine 1 eingetragen werden. In diesem Falle erhält A1 also den Wert 1, B2 den Wert 2, B3 den Wert 4 usw.
Die Reihe soll beendet werden, bevor der nächste Wert den größten ganzzahligen Wert 32.767 überschreitet. Bei der wievielten Zelle ist dies der Fall?

alle Aufgaben von Prof. Klaus Teich übernommen

	A	B	C	D	E	F	G
1	1						
2		2					
3			4				
4				8			
5					16		
6						32	
7							64

Übungen: 1. Diagonale (2/2)

- Erstellen Sie ein Struktogramm für diese Aufgabenstellung.
- Erstellen Sie ein entsprechendes VBA-Programm.
- Zusatzaufgabe: Entfernen Sie die Werte ab B2 aus der Tabelle, bevor Sie einen neuen Durchlauf starten.

Übungen: 2. Lottozahlen (1/3)

- Der folgende Code erzeugt Lottozahlen:

```
Sub Lottozahlen()
  Dim Werte(1 To 6) As Integer, i As Integer, s As String
  Randomize
  ' Der Befehl Randomize initilisiert den Zufallszahlengenerator.
  For i = 1 To 6
    Werte(i) = Round(48 * Rnd, 0) + 1
    ' Die Rnd-Funktion gibt einen Wert zurück, der kleiner
    ' als 1, aber größer als oder gleich Null ist.
    s = s & " " & Werte(i)
  Next i
  MsgBox "Lottozahlen: " & vbCrLf & s
End Sub
```

Übungen: 2. Lottozahlen (2/3)

a) Der Zufallsgenerator kann mehrfach gleiche Werte in der 6er-Reihe liefern (wie 14 in unserem Beispiel). Ändern Sie die Prozedur so ab, dass kein Wert öfter als 1 Mal vorkommt.

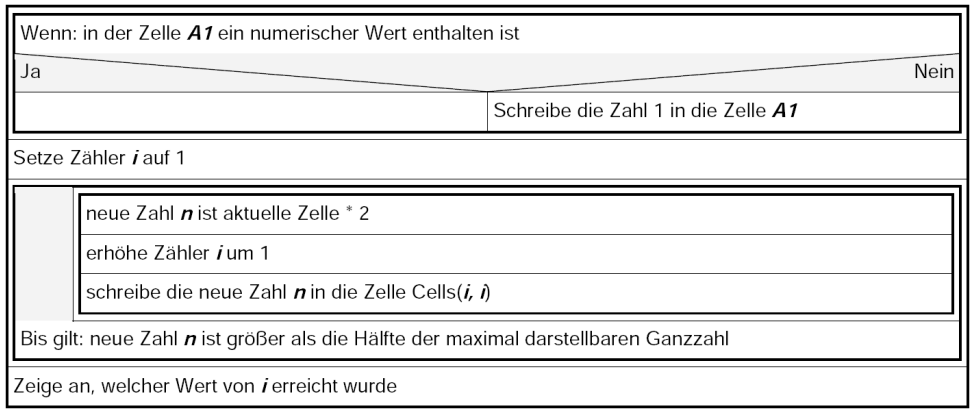
b) Sortieren Sie die Werte aufsteigend. Das Array „Werte“ soll also in `Werte(1)` den kleinsten und in `Werte(6)` den höchsten Wert enthalten. Nutzen Sie dazu den schon bekannten Bubblesort-Algorithmus.

Übungen: 2. Lottozahlen (3/3)

c) Eine neue Prozedur „Lottoschein“ soll 10 Tippreihen im aktuellen Excel-Arbeitsblatt erzeugen. Die Zeile 1 des Arbeitsblatts soll die Überschrift „Lottoschein“ enthalten. Die Tippreihen sollen ab Zeile 2 aufsteigend sortiert eingetragen sein:

	A	B	C	D	E	F
1	Lottoschein					
2	3	18	26	29	37	46
3	13	14	15	23	30	32
4	11	29	34	40	44	48
5	1	6	12	26	33	48
6	3	5	14	15	28	39
7	14	15	19	20	46	48
8	8	16	20	31	32	35
9	4	9	10	22	29	44
10	13	14	19	31	38	45
11	5	21	27	31	34	44

Übung 1: Anfang Musterlösung



Terminhinweis: 12.01. / 19.01.

- Die **Vorlesung am 13.01. entfällt** auf Wunsch von Prof. Schulz, der am 13.01. eine Prüfung im Fach Technisches Zeichnen durchführt.
- Ersatztermin für meine Veranstaltung am 13.01. ist **Montag 12.01., 10:00-11:30 in Raum BG.090**.
- Außerdem steht jetzt der Ersatztermin für die beiden ausgefallenen Vorlesungen im November fest: **Montag 19.01., 8:15-11.30 in Raum BG.090** – Achtung: Das ist dann eine **4-fach**-Stunde!

siehe auch Terminplan im VVK oder auf meiner Homepage