Ergänzung der UserForms (I)

1) With-Anweisung (With … End With)

Die With-Anweisung führt eine Reihe von Anweisungen (z. B. Eigenschaften) für ein einzelnes Objekt aus.
With-Anweisungen beschleunigen die Ausführung von Prozeduren und reduzieren wiederholte Eingaben.

Beispiel:
```
With Bezeichnungsfeld1
   .Height = 2000
   .Width = 2000
   .Caption = "Schönen Tag noch"
End With
```

Aufgabe: Finden Sie passende Beispiele in Ihrem Quelltext zur Umsetzung der With-Anweisung.

2) Die Message-Box und ihre MsgBox-Konstanten

Die Message-Box kann weitere Schaltflächen außerhalb der „o. k." Schaltfläche beinhalten. Hier die Wichtigsten:

vbOKOnly	Nur Schaltfläche **OK** (Voreinstellung)
vbOKCancel	Schaltflächen **OK** und **Abbrechen**
vbAbortRetryIgnore	Schaltflächen **Abbruch**, **Wiederholen** und **Ignorieren**
vbYesNoCancel	Schaltflächen **Ja, Nein** und **Abbrechen**
vbYesNo	Schaltflächen **Ja** und **Nein**
vbRetryCancel	Schaltflächen **Wiederholen** und **Abbrechen**
vbCritical	Meldung für kritischen Fehler
vbQuestion	Warnung mit Abfrage
vbExclamation	Warnmeldung
vbInformation	Informationsmeldung

Im Programmcode kann nun jede dieser Schaltflächen angesprochen werden. Da bei Aktivierung einer Schaltfläche ein Rückgabewert für diese Schaltfläche („true") gegeben wird, braucht diese einfach nur über eine zusätzliche Variable abgefragt werden.

Beispiel (vgl. online-Hilfe):

```
Dim Antwort, Text as  String
Antwort = Msgbox („Ist heute ein guter Tag?",vbYesNo)
If  Antwort = vbYes Then
Text = "dann viel Spaß dabei"
Else
Text = "macht doch nichts, wird schon wieder"
End If
Msxbox Text
```

Aufgabe: Finden Sie passende Beispiele in Ihrem Quelltext zur Umsetzung der erweiterten MSG-Box-Nutzung.

3) Fehlerbehandlungsroutinen mit der ON ERROR - Anweisung

Die ON ERROR-Anweisung (ON ERROR GOTO *ZEILE*) aktiviert eine Fehlerbehandlungsroutine, die z. B. Syntaxfehler (Werteeingaben außerhalb eines Gültigkeitsbereichs von gegebenen Variablen) aufdeckt.

Tritt danach ein Laufzeitfehler auf, so verzweigt die Programmsteuerung zu *Zeile* und aktiviert so die Fehlerbehandlungsroutine. Die angegebene Zeile muss sich in derselben Prozedur wie die On Error-Anweisung befinden, andernfalls tritt ein Fehler zur Kompilierungszeit auf. ***WICHTIG:*** Vor der Fehlerroutine muss die Anweisung ***EXIT SUB*** stehen!!!

Beispiel:
```
On Error GoTo errorhandler
…
Exit Sub
errorhandler:
MsgBox("Fehleingabe")
End Sub
```

Aufgabe: Finden Sie passende Beispiele in Ihrem Quelltext zur Umsetzung der Fehlerbehandlungsroutine.

Ergänzung der UserForms (II) – Exkurs „Arrays“

Arrays oder auch Datenfelder sind mit Matrizen in der Mathematik vergleichbar. Sie können sie sich als ein Art temporäre Tabelle, bestehend aus Zeilen und Spalten vorstellen. Der Zugriff auf die Felder des Arrays erfolgt über Indizes, die die Zeilen und Spaltennummern darstellen. Diese Vorstellung ist jedoch stark vereinfacht, da Datenfelder nicht nur aus zwei Dimensionen (also Zeilen und Spalten) bestehen, sondern bis zu 64 Dimensionen haben können. Für die meisten Anwendungen reichen jedoch zwei Dimensionen aus.

Im Unterschied zu einfachen Variablen können Sie in Datenfeldern eine ganze Menge gleichartiger Daten speichern, ohne dass Sie für jeden Werte eine einzelne Variable deklarieren müssen. Hier ein Beispiel.

```
Sub Namenliste()
Dim strNamen(3, 3) As String
 strNamen(0, 0) = "Vornamen"
 strNamen(0, 1) = "Nachnamen"
 strNamen(1, 0) = "Heinz"
 strNamen(1, 1) = "Müller"
 strNamen(2, 0) = "Gerard"
 strNamen(2, 1) = "Maier"
 MsgBox strNamen(0, 0) & " " & strNamen(0, 1) _
 & Chr(13) & strNamen(1, 0) & " " & strNamen(1, 1) _
 & Chr(13) & strNamen(2, 0) & " " & strNamen(2, 1)
End Sub
```

Das Ergebnis der Message-Box:

Die Array-Inhalte werden über die Indizes angesprochen

Die ARRAY-Funktion in VBA

VBA stellt außerdem die Funktion Array zur Verfügung, die die Handhabung von Arrays gleich in mehreren Beziehungen vereinfacht. Zum einen entfällt die Definition des Arrays. Der zweite Vorteil besteht darin, dass Sie den Feldern des Arrays nicht einzeln ihre Werte zuweisen müssen, das erledigt die Funktion selbst.

Sie benötigen dazu eine Variable Variant, die das Array aufnehmen soll. Die Werte des Arrays übergeben Sie als Parameter an die Funktion. Im folgenden Beispiel ist merkarray die Variable, in der das Ergebnis der Funktion Array gespeichert wird. Die Parameter, hier die Optionsfelder 1 bis 15, werden der Funktion übergeben und bilden den Inhalt des Arrays, das die Funktion zurückgibt. In unserem Fall entweder „WAHR“ für aktiviertes Optionsfeld oder „Falsch“ für nicht aktiviertes Optionsfeld.

Nach Ausführung dieser Array-Funktion können Sie auf die Felder des Arrays über die Variable mit dem entsprechenden Index zugreifen. Mit merkarray(0) würden Sie also den Werte des ersten Felds zurückgeben. In unserem Fall wird in einer zählergesteuerten Schleife jeder Array-Inhalt überprüft. Die Nummer des Array-Felds „Wahr“ wird an das Eingabefeld txtEingabe übergeben.

```
Private Sub cmdUebergabe_Click()
Dim merkarray As Variant
Dim Zaehler  As Integer
'Schreibe die Rückgabewerte der Optionsfelder in einen Array ("Falsch", "Wahr", "Falsch", "Falsch";....)
 merkarray = Array(optNote0, optNote1, optNote2, optNote3, optNote4, optNote5, optNote6, optNote7, _
 optNote8, optNote9, optNote10, optNote11, optNote12, optNote13, OptNote14, optNote15)
' Gehe den Array durch und falls arrayErgebnis = TRUE und übergib gib Schleifenzähler in Testbox
 For Zaehler  = 0 To 15
  If merkarray(Zaehler) = True Then
   txtEingabe = Zaehler
  End If
 Next Zaehler
End Sub
```