7. Textfelder zur Zahleneingabe

Bereits im 1. Teil dieses Manuskripts wurde in Kap. 6.5.4 beschrieben, welche Probleme bei der Umwandlung und dem Vergleich von Zahleneingaben in Textfeldern und den zugehörigen nummerischen Formaten ausftreten können. Insbesondere ist zu beachten, dass abweichend von der mathematisch üblichen Notierung als Trennzeichen zwischen Vor- und Nachkommastellen nicht ein Komma, sondern ein Punkt verwendet werden muss. In diesem Fall wird z.B. die Text-Eingabe "2.345" durch die Val-Funktion in das Zahlenformat 2,345 überführt, das für mathematische Operationen weiter verwendet werden kann, während bei der Eingabe von "2,345" die Val-Funktion als Ergebnis den Zahlenwert 2 liefert, die Nachkommastellen also abschneidet. Wird nun das Ergebnis einer solchen Operation in einem Textfeld gespeichert, erscheint wieder das Komma als Trennzeichen, was bei einer ggf. nachfolgenden Weiterverarbeitung nach Umwandlung mit der Val-Funktion zu einem Abschneiden der Nachkommastellen führt.

7.1 Arbeiten mit Zahlenwerten

Das oben beschriebene Umwandlungsproblem macht es erforderlich, bei der Verarbeitung von Zahlenwerten klar zwischen Text- und Zahlenformaten zu trennen. Dabei ist grundsätzlich folgendes Verfahren zu empfehlen:

Eingeben der Zahlenwerte in Textfelder
Überprüfen des verwendeten Trennzeichens
Umwandeln der Zahlenwerte in ein nummerisches Format
Verarbeiten der nummerischen Zahlenwerte
Ausgaben der Zahlenwerte in Textfeldern

Je nach gewünschtem Eingabekomfort können Textfelder zusätzlich vor der Verarbeitung manipuliert werden, um auch die Verwendung des aus der Mathematik gewohnten Kommas als Trennzeichen zu ermöglichen.

7.1.1 Eingeben der Zahlenwerte in Textfelder

Bei der Eingabe von Zahlenwerten muss sich der Benutzer an bestimmte Konventionen halten, so sollten z.B. keine Leerzeichen oder nicht nummerische Zeichen verwendet werden, sofern sie nicht zu einem definierten Zahlenformat (wie z.B. für die wissenschaftliche Notation) gehören. Verwendet der Benutzer den Punkt als Trennzeichen, dann können die eingegebenen Werte ohne weitere Maßnahmen umgewandelt werden.

7.1.2 Überprüfen des verwendeten Trennzeichens

In diesem Beispiel soll lediglich gezeigt werden, wie der Benutzer das Komma als Trennzeichen verwenden darf, ohne die nachfolgende Umwandlung in ein nummerisches Format zu verfälschen. Dabei ist lediglich zu prüfen, ob in der eingegebenen Zeichenfolge ein Komma auftritt, um dieses dann in einen Punkt umzuwandeln. Dazu dient als Beispiel folgende Funktion, die als Argument eine Zeichenkette zur Analyse erhält und die ggf. korrigierte Zeichenkette als Ergebnis zurückgibt.

```
Public Function KommaPrüfung(str as string) as string
   Dim i as integer
   For i = 1 to Len(str)
      If Mid(str,i,1) = "," Then Mid(str,i,1) = "."
   next i
   KommaPrüfung = str
End Function
```

In der Zählschleife werden alle Zeichen in der als str übergebenen Zeichenkettenvariablen überprüft; wenn das aktuelle Zeichen ein Komma ist, wird es durch einen Punkt ersetzt. Abschließend wir die ggf. modifizierte Zeichenkette als Ergebnis der Funktion zugewiesen und die Funktion beendet.

7.1.3 Umwandeln der Zahlenwerte in ein nummerisches Format

Für mathematische Operationen kommen im Regelfall nur Zahlentypen in Frage, die Kommazahlen verarbeiten können. Dazu gibt es in VBA die beiden Formate single und double, die sich hinsichtlich ihrer Stellenzahl (und damit ihrer Genauigkeit) unterscheiden.

Da sie sich lediglich im benötigten Speicherplatz unterscheiden, der in der Regel reichlich vorhanden ist, bietet es sich an, grundsätzlich den Typ double zu vewenden. Auch hier kommt eine Funktion zum Einsatz, der als

Argument eine Zeichenkette übergeben wird und die als Ergebnis einen Zahlenwert zurückgibt:

```
Public Function Str2Double (str as string) as double
   Str2Double = Val(str)
End Function
```

Vergleicht man die Abfolge der Verarbeitungsschritte in 7.1. mit den beiden obigen Funktionen, so wird deutlich, dass die Überprüpfung und Umwandlung immer nacheinander auszuführen sind. Deshalb bietet es sich an, die beiden Schritte in einer Funktion zusammenzufassen:

```
Public Function StrCheck2Double (str as string) as double
   Dim i as integer
   For i = 1 to Len(str)
      If Mid(str,i,1) = "," Then Mid(str,i,1) = "."
   next i
   StrCheck2Double = Val(str)
End Function
```

Hier wird zunächst (wie bereits beschrieben) das Trennzeichen überprüft und ggf. korrigiert, anschließend erfolgt die Umwandlung in das nummerische Datenformat. Einziger Nachteil dieses Verfahrens ist, dass man sich nun nicht mehr die modifizierte Zeichenkette anzeigen lassen kann.

7.1.4 Verarbeiten der nummerischen Zahlenwerte

Die so gewonnenen Zahlenwerte können nun beliebig verarbeitet werden, dabei muss allerdings darauf geachtet werden, dass alle ggf. verwendeten Variablen einen geeigneten Typ aufweisen - vorzugsweise also ebenfalls als double deklariert werden. Um z.B. das Produkt zweier in den Textfeldern TxtX und TxtY eingegebener Zahlenwerte, die als Zeichenketten übergeben werden, zu bestimmen und als Funktionswert zurückzugeben sind folgende Vorgehensweisen äquivalent:

```
Private Function ProduktBerechnen1(str1 as string, str2 as string) as double
   Dim x as double, y as double, result as double
   x = StrCheck2Double(str1)
   y = StrCheck2Double(str2)
   result = x * y
   ProduktBerechnen1 = result
End Function
```

bzw.

```
Private Function ProduktBerechnen2(str1 as string, str2 as string) as double
   ProduktBerechnen2 = StrCheck2Double(str1) * StrCheck2Double(str2)
End Function
```

Das zweite Beispiel benötigt keine nummerischen Variablen, führt aber zu dem gleichen Ergebnis, da auch hier das Produkt zweier Zahlenwerte gebildet wird (man beachte, dass die Funktion StrCheck2Double als Rückgabewert einen double-Wert zuweist).

Der Aufruf einer solchen Funktion aus einer Anweisungsfolge in einer beliebigen Prozedur erfolgt dann z.B. durch

```
Private Sub ...
Dim ergebnis as double
...
ergebnis = ProduktBerechnen1(TxtX, TxtY)
```

wobei der Variablen ergebnis der in der Funktion zuvor berechnete Zahlenwert zugeweisen wird.

7.1.5 Ausgeben der Zahlenwerte

Ohne weitere Maßnahmen würde eine Variable vom Typ double mit 15 signifikanten Ziffern ausgegeben werden, also liefert z.B. die Zahl

100 : 7 = 5,88235294117647

In manchen Fällen ist eine solche Ausgabe mit allen 15 Ziffern schlecht lesbar oder vermittelt den Eindruck einer lediglich scheinbaren Genauigkeit, die aber nicht den Eingangsdaten entspricht:

unformatierte Ausgabe:

Im obigen Schirmbildauszug sieht man z.B., dass die letzte Ziffer des Ergebnisses nicht mehr angezeigt wird, weil das Fenster zu klein ist.

Abhilfe schafft hier die Format-Funktion von VBA, die es erlaubt, ein bestimmtes Zahlenformat für die Ausgabe festzulegen. Diese Funktion benötigt zwei Argumente und gibt als Ergebnis eine Zeichenkette zurück. Das erste Argument enthält den auszugebenden Zahlenwert (das kann aber auch ein nummerischer Ausdruck sein), das zweite eine Folge bestimmter Zeichen (den sog. Formatstring), die das Ausgabeformat festlegen. Die Wirkung des Formatstrings auf die Ausgabe zeigen die folgenden Beispiele:

Format(100:7, "0.000")	100 : 7	5,882
Format(100:7, "#000.000")	100 : 7	005,882
Format(100:7, "#.000E+#")	100 : 7	5,882e+0
Format(100:7, "#.000E+#0")	100 : 7	58,824E-01

7.2 Arbeiten mit Teilbereichen aus Textfeldern

Mitunter kann es notwendig sein, nur mit Teilbereichen eines Textfeldes zu arbeiten, die einen Zahlenwert enthalten sei es, dass neben der Zahl noch weitere Angaben enthalten sind oder dass nummerische Operationen durchgeführt werden sollen, die eine höhere Stellenzahl verarbeiten sollen als sie durch die vorhandenen Datentypen unterstützt werden. Als Beispiel soll hier die Subtraktion zweier Zahlen beliebiger Stellenzahl (ggf. lediglich begrenzt durch die Länges des Eingabefeldes) dienen.

7.2.1 Entwicklung eines Subtraktions-Algorithmus

Wie aus der schriftlichen Subtraktion bekannt, werden die beiden Zahlen übereinander geschrieben und "von hinten nach vorne" abgearbeitet, dabei wird ziffernweise bestimmt, welche Ziffer addiert werden muss, um von der unteren Ziffer die obere zu erhalten. Ist die obere kleiner, wird ein Übertrag gebildet, der dann zur nächsten Ziffer addiert werden muss, bevor man weiter verfährt. Das zugehörige Struktogramm ist in Abb. 1 abgebildet.

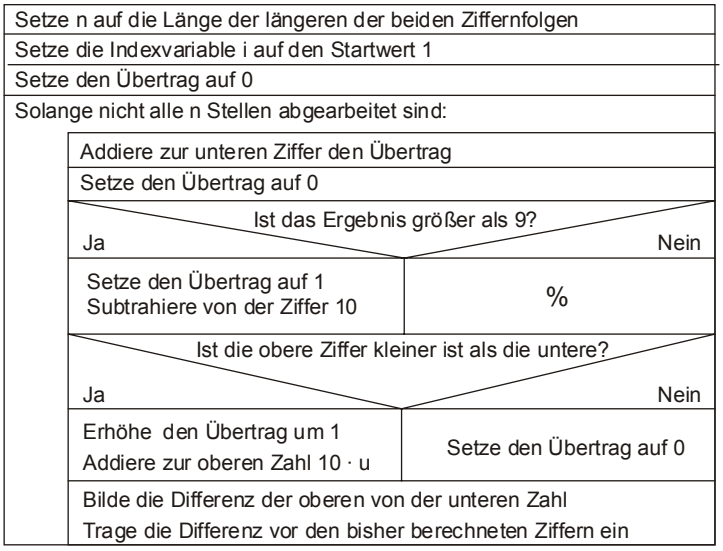

Abb. 1: Struktogramm zur Subtraktion

Ist die erste Zahl größer als die zweite, so erhält man direkt das richtige Ergebnis, z.B.:

```
  5 4 3
- 2 3 4
-------
  3 0 9
```

Ist dagegen die obere Zahl kleiner als die erste, erhält man ein zunächst verblüffendes Ergebnis:

```
    2 3 4
  -   5 4 3
-----------
  . . 9 6 9 1
```

Dieses Ergebnis ist nach dem gleichen Algorithmus entstanden, liefert aber nicht wie erwartet eine negative Zahl, sondern ein unendliche Folge von vorangestellten Neunen. Man nennt eine solche Zahl deswegen das **Neuner-Komplement**. Aus diesem kann man durch einen weiteren Schritt das tatsächliche Ergebnis gewinnen.

7.2.2 Umwandeln des Neuner-Komplements

In dem oben beschriebenen Algorithmus erkennt man das Auftreten des Neunerkomplements daran, dass nach Abarbeiten der letzten Ziffer ein Übertrag verbleibt, der größer als 0 ist. Nun bildet man zu jeder Ziffer des Ergebnisses die Differenz zur Ziffer 9 und erhält daraus eine neue Zahl. Zu dieser addiert man 1 und ergänzt das

negative Vorzeichen. Abb. 2 zeigt das zugehörige Struktogramm. Dieses neue Ergebnis ist dann das gesuchte:

```
  9999
..9691
------
  0308
+    1
------
 - 309
```

Setze n auf die Länge des Ergebnisses
Setze die Indexvariable i auf den Startwert 1
Solange nicht alle n Stellen abgearbeitet sind ...
Bilde die Differenz der aktuellen Stelle zu 9 Trage die Differenz vor den bisher berechneten Ziffern ein
Addiere zu dem Ergebnis 1
Füge vor dem Ergebnis ein Minuszeichen ein

Abb. 2: Bilden des Neuner-Komplements

Man beachte, dass es hier nicht genügt, einfach nur die letzte Ziffer um 1 zu erhöhen, da dabei ggf. ein Übertrag auftreten könnte, der auch bei den höheren Stellen berücksichtigt werden muss. Hier muss für Zahlen mit beliebig vielen Stellen ein der Subtraktion entsprechender Additions-Algorithmus verwendet werden, der hier jedoch nicht näher beschrieben werden soll, da er dem für die Subtraktion weitgehend ähnelt.

7.2.3 Implementation des Subtraktionsalgorithmus

Das folgende Programmmodul berechnet nach dem Struktogramm aus Abb. 1 die Differenz zweier Zahlen, die in den Zeichenketten str1 und str2 an die Funktion übergeben werden, das Ergebnis wird ebenfalls als Zeichenkette an die aufrufende Stelle zurückgegeben. Bei Bedarf wird das Neuner-Komplement gebildet.

```
Public Function DifferenzBerechnen(str1 As String, str2 As String) As String
    Dim i As Integer, n As Integer, u As Integer, x As Integer, y As Integer, tmp As String
    tmp = ""
    If Len(str2) > Len(str1) Then n = Len(str2) Else n = Len(str1)
    While Len(str2) < n                             ' Zeichenketten auf gleiche Länge bringen
        str2 = "0" & str2
    Wend
    While Len(str1) < n                             ' Zeichenketten auf gleiche Länge bringen
        str1 = "0" & str1
    Wend
    u = 0                                           ' Setze den Übertrag auf 0
    For i = 0 To n - 1
        x = Val(Mid(str1, n - i, 1))                ' Zahlenwert der oberen Ziffer
        y = Val(Mid(str2, n - i, 1)) + u            ' Zahlenwert untere Ziffer + Übertrag
        u = 0                                       ' Setze Übertrag auf 0
        If y > 9 Then                               ' Ergebnis größer als 9?
            u = 1                                   ' Übertrag auf 1 setzen
            y = y - 10                              ' Subtraktion von 10
        End If
        If x < y Then                               ' Oberer Zahlenwert kleiner als unterer?
            u = u + 1                               ' Übertrag um 1 erhöhen
            x = x + 10 * u                          ' zur oberen Zahl 10 addieren
        End If
        tmp = Right(str(x - y), 1) & tmp            ' Einerziffer der Differenz ausgeben
    Next i
    If u > 0 Then tmp = KomplementBilden(tmp)       ' ggf. Komplement bilden
    DifferenzBerechnen = tmp                        ' Ergebnis als Zeichenkette zurückgeben
End Function
```

7.2.4 Implementation des Neuner-Komplements

Der nachfolgende Programmauszug bestimmt entsprechend dem Struktogramm nach Abb. 2 das Neuner-Komplement. Die zugehörige Zeichenkette wird als str1 übergeben, das Ergebnis wird auch hier als Zeichenkette zurückgegeben.

```
Public Function KomplementBilden(str1 As String) As String
    Dim i As Integer, n As Integer, x As Integer, tmp As String
    tmp = ""
    n = Len(str1)                                   ' Setze n auf die Länge der Zeichenkette
    For i = 0 To n - 1                              ' Beginne bei der letzten Ziffer
        x = 9 - Val(Mid(str1, n - i, 1))            ' Berechne die Differenz zu 9
        tmp = Right(str(x), 1) & tmp                ' Trage das Ergebnis ein
    Next i
    tmp = SummeBerechnen(tmp, "1")                  ' Addiere zum Ergebnis 1
    KomplementBilden = "-" & tmp                    ' Ergänze Vorzeichen, Ergebnisrückgabe
End Function
```

7.2.5 Entfernen überflüssiger führender Nullen

Bei der Bildung einer Differenz nach den vorbeschriebenen Algorithmen kann es vorkommen, dass das Ergebnis führende Nullen aufweist. Um diese zu entfernen, ist die Zeichenkette von links nach rechts abzuarbeiten. Dabei werden die Nullen solange entfernt, bis eine von Null verschiedene Ziffer auftritt, die letzte Ziffer bleibt jedoch erhalten, auch wenn sie den Wert 0 hat, damit das Ergebnis keine leere Zeichenkette werden kann. Ein eventuell auftretendes Vorzeichen muss natürlich erhalten bleiben.

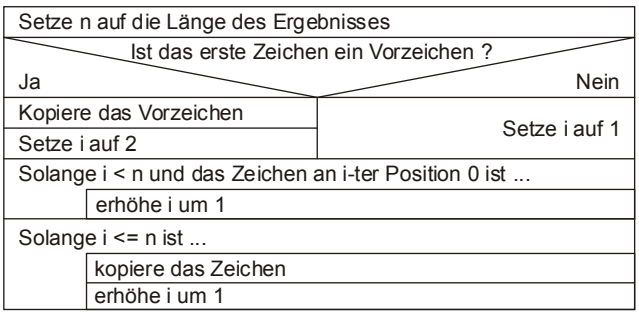

Abb. 3: Struktogramm zum Entfernen führender Nullen

7.2.6 Implementation der Entfernung führender Nullen

Der nachfolgende Codeausschnitt zeigt eine Funktion, die eine Zeichenkette mit der zu überprüfenden Zahl enthält, aus der unnnötige führende Nullen entfernt werden sollen. Auch hier wird das Ergebnis als Zeichenkette zurückgegeben.

```
Public Function NullEntfernen(str1 As String) As String
    Dim i As Integer, n As Integer, tmp As String
    n = Len(str1)
    If Left(str1, 1) = "-" Then
        i = 2
        tmp = "-"
    Else
        i = 1
        tmp = ""
    End If
    While (i < n) And (Mid(str1, i, 1) = "0")
        i = i + 1
    Wend
    While i <= n
        tmp = tmp & Mid(str1, i, 1)
        i = i + 1
    Wend
    NullEntfernen = tmp
End Function
```

7.3 In VBA verwendbare mathematische Funktionen

Visual Basic for Applications für Excel enthält eine gewisse Anzahl mathematischer Funktionen, die allerdings deutlich kleiner ist als die in Excel selbst implementierten. In VBA stehen folgende Funktionen zur Verfügung:

Abs	Absolutwert	Berechnen des Betrages einer Zahl
Atn	Arkustangens	Umkehrfunktion der Tangensfunktion (im Bogenmaß)
Cos	Cosinus	Berechnen des Cosinus eines Winkels (im Bogenmaß)
Exp	Exponent	Berechnen einer Potenz von e (e = 2,71828..., Eulersche Zahl)
Fix	Fixzahl	Bestimmen der nächstgelegenen ganzen Zahl
Int	Ganzzahl	Bilden des ganzzahligen Anteils einer Zahl durch Rundung
Log	Logarithmus	Berechnen des Logarithmus zur Basis e (natürlicher Log.)
Rnd	Random	Erzeugen einer Zufallszahl zwischen 0 und 1
Sgn	Signum	Bestimmen des Vorzeichens einer Zahl, ergibt +1, 0 oder -1
Sin	Sinus	Berechnen des Sinus eines Winkels (im Bogenmaß)
Sqr	Squareroot	Berechnen der Quadratwurzel einer Zahl
Tan	Tangens	Berechnen des Tangens eines Winkels (im Bogenmaß)

Bis auf die Funktionen Rnd (Single) und Sgn (integer) geben alle diese Funktionen einen Wert vom Typ double zurück. Zur Darstellung anderer - insbesondere trigonometrischer Funktionen - gibt die Hilfedatei von VBA unter dem Stichwort "Abgeleitete matehmatische Funktionen" eine Reihe von Formeln an, mit denen weitere, häufig benötigte Funktionen dargestellt werden können. Die für die Umrechnung von Bogen- in Winkelmaß benötigte Kreiszahl p erhält man übrigens durch den Ausdruck

```
pi = 4 * Atn(1)
```