Neu: Word-Dokument mit Benutzerformular bereinigen

Volker Thormählen, 5. Mai 2021

Inhaltsverzeichnis

1 Aufgabenstellung

In diesem Beitrag werden Ergänzungen eines Benutzerformulars beschrieben, das vom Autor bereits im Jahr 2017 veröffentlicht wurde[1].

- Das ergänzte Benutzerformular enthält im Rahmen „Suchen und ersetzen“ (s. Abbildung 2) zwei zusätzliche Optionsfelder.
- Der VBA-Quellcode zu allen 13 Optionsfeldern (s. Abbildung 2) wird präsentiert.

2 Projekt

Das zugehörige Projekt enthält 1 Benutzerformular mit dem Namen *frmGetFile* (s. Abbildung 1) sowie 3 Module mit dem Namen *Cleaning, Setup* bzw. *Tools*:

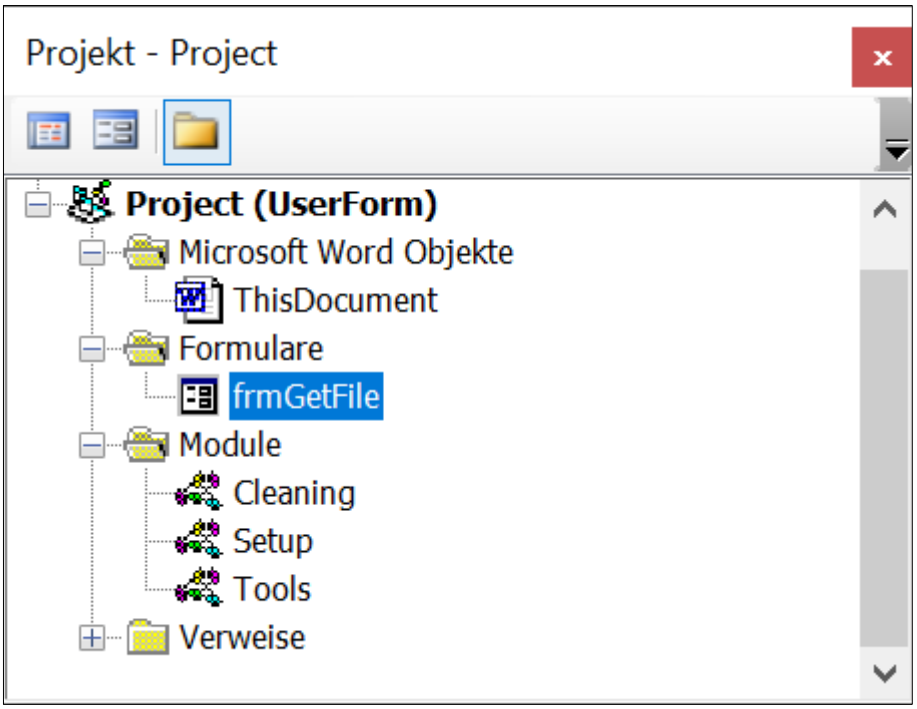

Abbildung 1: Das Projekt

3 Benutzerformular

Gruppiert in 5 Rahmen enthält das Benutzerformular *frmGetFile* insgesamt 13 Optionsfelder. In einem ersten Schritt muss zuvor das zu bereinigende Word-Dokument (sog. Zieldokument) interaktiv ausgewählt werden (s. Abbildung 2).

Außerdem enthält das Benutzerformular am unteren Rand vier weitere Schaltflächen zum Aufruf entsprechender Ereignisprozeduren:

[1] https://www.dr-thormaehlen.de/WORD/Benutzerformular.pdf

Bezeichnung der Schaltfläche	Aufgabe der Schaltfläche
1. Schritt: …	Interaktive Auswahl des zu bereinigenden Zieldokuments.
OK	Bewirkt die Ausführung von interaktiven Bereinigungs-prozeduren für die ausgewählten Optionsfelder.
Abbruch	Bewirkt den Abbruch bzw. das Ende des Benutzerformulars.
Alles zurücksetzen	Setzt der Wert aller Optionsfelder zurück auf den Anfangswert: False.
Alles auswählen	Setzt den Wert aller Optionsfelder auf „ausgewählt“: True.

Tabelle 1: Schaltflächen im Benutzerformular zum Aufruf von Ereignisprozeduren

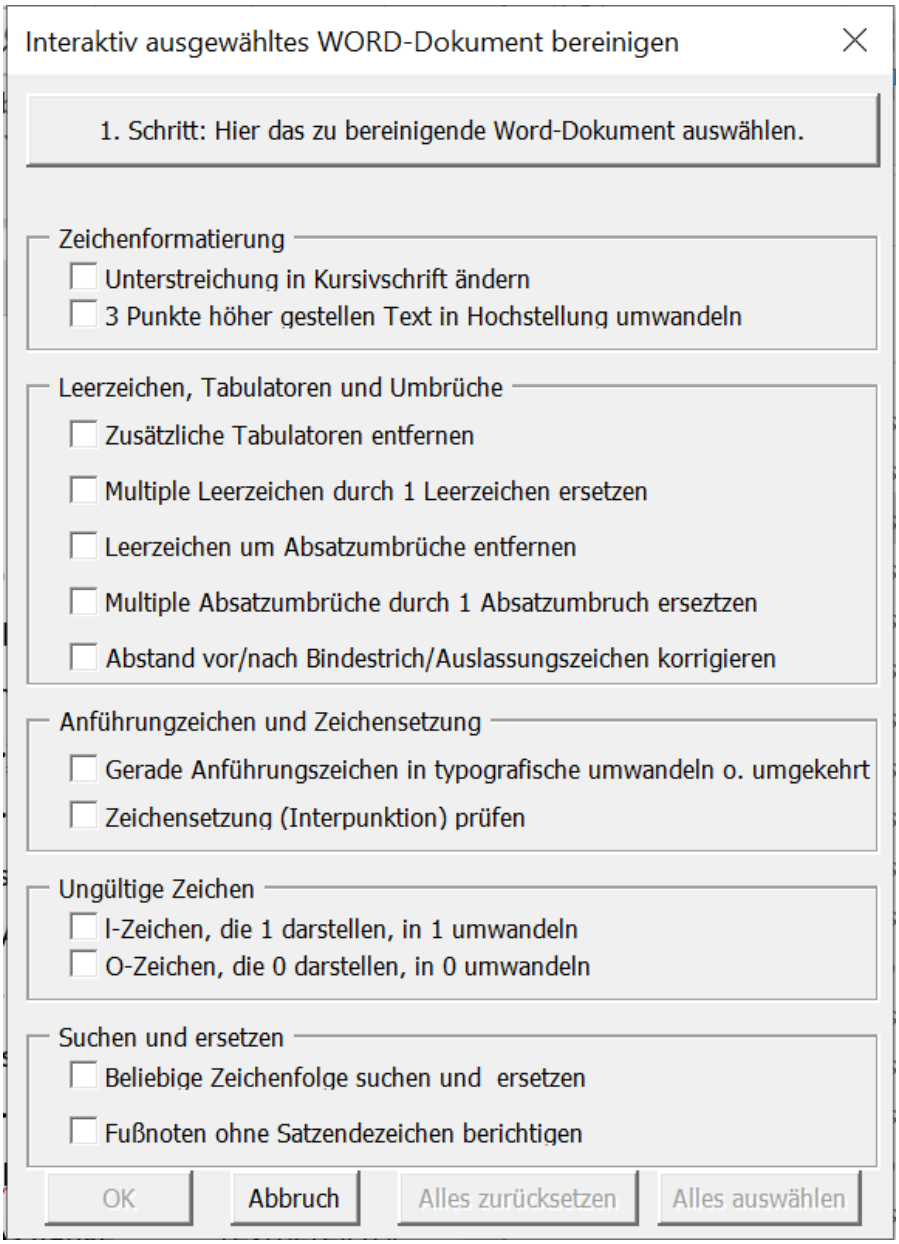

Abbildung 2: Benutzerformular

4 Module

Das Projekt enthält 3 Module (s. Tabelle 2):

Modul	Inhalt	Prozeduren
Cleaning	Prozeduren zum Bereinigen des Zieldokuments.	13
Setup	Prozeduren zum Einrichten des Benutzerformulars *frmGetFile*.	6
Tools	Prozeduren zur interaktiven Auswahl des WORD-Zieldokuments.	2

Tabelle 2: Module des Projekts

4.1 Cleaning

Die 13 Prozeduren im Modul *Cleaning* führen folgende Aufgaben aus (s. Tabelle 3):

1.	Unterstreichung in Kursivschrift ändern u. Unterstreichung aufheben.
2.	Drei Punkte höher gestellten Text in Hochstellung umwandeln und Schriftgrad erhöhen.
3.	Zusätzliche Tabulatoren entfernen.
4.	Multiple Leerzeichen durch 1 Leerzeichen ersetzen.
5.	Leerstelle(n) vor/nach Absatzmarken entfernen.
6.	Ein oder mehr Leerzeichen vor/hinter Bindestrich entfernen.
7.	Leerstelle(n) vor/nach Auslassungszeichen entfernen.
8.	Gerade in typografische Anführungszeichen umwandeln oder umgekehrt.
9.	Zeichensetzung prüfen.
10.	l-Zeichen, die keine eins darstellen, in 1 umwandeln.
11.	O-Zeichen, die keine null darstellen, in 0 umwandeln.
12.	Zeichenfolge finden und ersetzen.
13.	Fehlendes Zeichen am Satzende in Fußnote einfügen.

Tabelle 3: Aufgaben der 13 Bereinigungsprozeduren im Modul Cleaning

Die 13 Prozeduren im Modul *Cleaning* benötigen folgende Deklarationen von Konstanten bzw. Variablen (s. Tabelle 4):

Konstante/Variable	Datentyp	Kommentar
Const conLast	String = ".?!"	Zeichen für Satzende
Dim bolOption	Boolean	Optionswert
Dim CurrentColor	WdColorIndex	Farbindex
Dim ftIdx	Integer	Schleifenzähler f. Fußnoten
Dim ftRng	Range	Fußnotenbereich
Dim intChanges	Integer	Zähler f. Ersetzungen
Dim intChoice	Integer	Auswahl d. Benutzers
Dim intHits	Integer	Trefferzähler
Dim intLoop	Integer	Schleifenzähler
Dim intUpdCnt	Integer	Zähler f. Änderungen
Dim lngAsk	Long	Antwort auf Abfrage
Dim lngChrCnt	Long	Anfängliche Zeichenzahl im Dokument
Dim lngCnt	Long	Zähler f. entfernte Zeichen
Dim lngHighLight	Long	Farbwert f. Hervorhebungen
Dim lngLoop	Long	Schleifenzähler
Dim lngRetVal	Long	Rückgabewert d. MsgBox
Dim objRng	Range	Textbereich
Dim rngDoc	Range	Suchbereich
Dim strFindTxt	String	Suchbegriff
Dim strLast *)	String * 1	Letztes Zeichen einer Fußnote
Dim strListSep *)	String * 1	Listentrennzeichen
Dim strReplTxt	String	Ersatzbegriff
Dim StyleNormal	Style	Normaler Schreibstil
Dim varFind	Variant	Suchbegriff
Dim varFindList	Variant	Suchliste
Dim varRepl	Variant	Ersatzbegriff
*) Zeichenfolgevariable der Länge 1 Byte.		

Tabelle 4: Konstanten und Variablen der 13 Prozeduren im Modul Cleaning

4.2 Stetup

In diesem Modul werden mit der Subprozedur *UserForm_Initialize* die Anfangswerte des Benutzerformulars mit dem Namen *frmSetFile* gesetzt. Die übrigen Ereignisprozeduren werden durch die Schaltflächen des Benutzerformulars ausgelöst.

4.3 Tools

Das Modul *Tools* (dt. Werkzeuge) enthält nur zwei Prozeduren:

- Mit der Subprozedur *OpenWordFile* wird das WORD-Zieldokument durch den Benutzer interaktiv ausgewählt.
- Mit der Funktionsprozedur *ProcExists* wird das Vorkommen einer Subprozedur im Modul *Cleaning* überprüft (s. Tabelle 2).

5 Quellcode

5.1 Modul *Cleaning*

```
Option Explicit
Sub ChangeUnderlingToItalic(docTgt As Document, strCaption As String)
    ' 1. Unterstreichung in Kursivschrift ändern u. Unterstreichung aufheben.
    Dim intChanges  As Integer  ' Zähler f. Ersetzungen
    Dim intHits     As Integer  ' Trefferzähler
    Dim lngRetVal   As Long     ' Rückgabewert d. MsgBox
    Dim objRng      As Range    ' Suchbereich
    Application.ScreenUpdating = True
    Set objRng = docTgt.Range(Start:=0, End:=0)
    With objRng.Find
        .Text = ""
        .ClearFormatting
        With .Font
            .Underline = wdUnderlineSingle
            .Italic = False
        End With
        .Forward = True
        .Wrap = wdFindContinue
        .Format = True
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            With objRng
                .HighlightColorIndex = wdTurquoise
                .Duplicate.Select
            End With
            lngRetVal = MsgBox("Ersetzen?" & vbCr & _
               objRng.Text, vbYesNo + vbQuestion, strCaption)
            If lngRetVal = vbYes Then
                intChanges = intChanges + 1
                With objRng.Font
                    .Underline = wdUnderlineNone
                    .Italic = True
                End With
            End If
            objRng.Collapse direction:=wdCollapseEnd
            .Execute
        Loop
    End With
```

```
    MsgBox CStr(intHits) & " Fundstelle(n), davon " & CStr(intChanges) & " ersetzt!",
vbExclamation, strCaption
    Application.ScreenUpdating = True
End Sub

Sub RaisedTextToSuperscript(docTgt As Document, strCaption As String)
  ' 2. Drei Punkte höher gestellten Text in Hochstellung umwandeln u. Schriftgrad erhöhen.
  Dim intChanges    As Integer  ' Zähler f. Ersetzungen
  Dim intHits       As Integer  ' Trefferzähler
  Dim lngRetVal     As Long     ' Rückkehrcode d. MsgBox
  Dim objRng        As Range    ' Suchbereich
  Application.ScreenUpdating = True
  Set objRng = docTgt.Range(Start:=0, End:=0)
  With objRng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = ""
    With .Font
        .Position = 3
    End With
    .Forward = True
    .Wrap = wdFindContinue
    .Format = True
    .MatchCase = False
    .MatchWholeWord = False
    .MatchWildcards = False
    .MatchSoundsLike = False
    .MatchAllWordForms = False
    .Execute
    Do While .Found = True
        intHits = intHits + 1
        With objRng
            .HighlightColorIndex = wdTurquoise
            .Duplicate.Select
        End With
        lngRetVal = MsgBox("Ersetzen?" & vbCr & _
           objRng.Text, vbYesNo + vbQuestion, strCaption)
        If lngRetVal = vbYes Then
            intChanges = intChanges + 1
            With objRng.Font
                .Size = 14
                .Position = 0
                .Superscript = True
            End With
        End If
        objRng.Collapse direction:=wdCollapseEnd
        .Execute
    Loop
  End With
  Application.ScreenUpdating = True
  MsgBox CStr(intHits) & " Fundstelle(n), davon " & CStr(intChanges) & " ersetzt!",
vbExclamation, strCaption
End Sub

Sub RemoveExtraneousTabs(docTgt As Document, strCaption As String)
    ' 3. Zusätzliche Tabulatoren entfernen
    Dim intChanges  As Integer  ' Zähler f. Ersetzungen
    Dim intHits     As Integer  ' Trefferzähler
    Dim lngRetVal   As Long     ' Rückkehrcode d. MsgBox
    Dim objRng      As Range    ' Suchbereich
    Application.ScreenUpdating = True
    Set objRng = docTgt.Range(Start:=0, End:=0)
    With objRng.Find
        .ClearFormatting
        .Text = "^t{2;}"
        .Forward = True
        .Wrap = wdFindContinue
```

```
14          .Format = True
15          .MatchCase = False
16          .MatchWholeWord = False
17          .MatchWildcards = True
18          .MatchSoundsLike = False
19          .MatchAllWordForms = False
20          .Execute
21          Do While .Found = True
22              intHits = intHits + 1
23              With objRng
24                  .HighlightColorIndex = wdTurquoise
25                  .Duplicate.Select
26              End With
27              lngRetVal = MsgBox("Ersetzen?" & vbCr & _
28                 objRng.Text, vbYesNo + vbQuestion, strCaption)
29              If lngRetVal = vbYes Then
30                  intChanges = intChanges + 1
31                  objRng.Text = Chr(9)
32              End If
33              objRng.Collapse direction:=wdCollapseEnd
34              .Execute
35          Loop
36       End With
37      MsgBox CStr(intHits) & " Fundstelle(n), davon " & CStr(intChanges) & " ersetzt!",
38  vbExclamation, strCaption
39      Application.ScreenUpdating = True
40  End Sub

 1  Sub ReplaceMultipleSpaces(docTgt As Document, strCaption As String)
 2      ' 4. Multiple Leerzeichen durch 1 Leerzeichen ersetzen.
 3      Dim intChanges  As Integer      ' Zähler f. Ersetzungen
 4      Dim intHits     As Integer      ' Trefferzähler
 5      Dim lngRetVal   As Long         ' Rückkehrcode d. MsgBox
 6      Dim objRng      As Range        ' Suchbereich
 7      Dim strListSep  As String * 1   ' Listentrennzeichen
 8      strListSep = Application.International(wdListSeparator) 'Listentrennzeichen ermitteln
 9      Application.ScreenUpdating = True
10      Set objRng = docTgt.Range(Start:=0, End:=0)
11      With objRng
12          With .Find
13              .ClearFormatting
14              .Text = "[ ]{2" & strListSep & "}"
15              .Forward = True
16              .Wrap = wdFindContinue
17              .Format = True
18              .MatchCase = False
19              .MatchWholeWord = False
20              .MatchWildcards = True
21              .MatchSoundsLike = False
22              .MatchAllWordForms = False
23              .Execute
24              Do While .Found = True
25                  intHits = intHits + 1
26                  With objRng
27                      .HighlightColorIndex = wdTurquoise
28                      .Duplicate.Select
29                  End With
30                  lngRetVal = MsgBox("Ersetzen?" & vbCr & _
31                     objRng.Text, vbYesNo + vbQuestion, strCaption)
32                  If lngRetVal = vbYes Then
33                      intChanges = intChanges + 1
34                      objRng.Text = Space(1)
35                  End If
36                  objRng.Collapse direction:=wdCollapseEnd
37                  .Execute
38              Loop
39          End With
```

```
    End With
    MsgBox CStr(intHits) & " Fundstelle(n), davon " & CStr(intChanges) & " ersetzt!",
vbExclamation, strCaption
    Application.ScreenUpdating = True
End Sub

Sub DeleteWhiteSpacesAroundReturns(docTgt As Document, strCaption As String)
    ' 5. Leerstelle(n) vor/nach Absatzmarken entfernen.
    Dim intChanges      As Integer  ' Zähler f. Ersetzungen
    Dim intHits         As Integer  ' Trefferzähler
    Dim intLoop         As Integer  ' Schleifenzähler
    Dim lngHighLight    As Long     ' Farbwert f. Ersetzungen
    Dim lngRetVal       As Long     ' Rückgabewert d. MsgBox
    Dim objRng          As Range    ' Suchbereich
    Dim varFindList     As Variant  ' Suchliste
    Application.ScreenUpdating = True
    ' Leerstelle(n) vor/nach Absatzmarken entfernen.
    varFindList = Array("^w^13", "^13^w") ' Suchbegriffe
    lngHighLight = Options.DefaultHighlightColorIndex
    For intLoop = LBound(varFindList) To UBound(varFindList)
        Set objRng = docTgt.Range(Start:=0, End:=0)
        With objRng.Find
            .ClearFormatting
            .Text = varFindList(intLoop)
            .Forward = True
            .Wrap = wdFindContinue
            .Format = True
            .MatchCase = False
            .MatchWholeWord = False
            .MatchWildcards = False
            .MatchSoundsLike = False
            .MatchAllWordForms = False
            .Execute
            Do While .Found = True
                intHits = intHits + 1
                With objRng
                    .HighlightColorIndex = wdYellow
                    .Duplicate.Select
                End With
                lngRetVal = MsgBox("Ersetzen?" & vbCr & _
                   objRng.Text, vbYesNo + vbQuestion, strCaption)
                If lngRetVal = vbYes Then
                    intChanges = intChanges + 1
                    objRng.Text = Chr(13)
                End If
                With objRng
                    .HighlightColorIndex = wdNoHighlight
                    .Collapse direction:=wdCollapseEnd
                End With
                .Execute
            Loop
        End With
    Next intLoop
    Options.DefaultHighlightColorIndex = lngHighLight
    MsgBox CStr(intHits) & " Fundstelle(n), davon " & _
           CStr(intChanges) & " ersetzt!", vbExclamation, strCaption

    ' Zwei weitere Prozeduren aufrufen.
    If ProcExists(docTgt, "Cleaning", "RemoveSpacesBeforeParagraphBreak") Then
        RemoveSpacesBeforeParagraphBreak    ' Leerzeichen VOR Absatzumbruch entfernen
    End If

    If ProcExists(docTgt, "Cleaning", "RemoveSpacesAfterParagraphBreak") Then
        RemoveSpacesAfterParagraphBreak     ' Leerzeichen NACH Absatzumbruch entfernen
    End If
    Application.ScreenUpdating = True
End Sub
```

```
Sub RemoveSpacesBeforeParagraphBreak()
    ' 5.1 Leerzeichen VOR Absatzumbruch entfernen
    Dim lngChrCnt   As Long  ' anfängliche Zeichenzahl im Dokument
    Dim lngCnt      As Long  ' Zähler f. entfernte Zeichen
    lngChrCnt = ActiveDocument.Characters.Count
    Application.ScreenUpdating = False
    With ActiveDocument.Content.Find
        .ClearFormatting
        .Text = "^w^p"
        With .Replacement
            .ClearFormatting
            .Text = "^p"
        End With
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .Execute Replace:=wdReplaceAll
    End With
    Application.ScreenUpdating = True
    MsgBox ActiveDocument.Characters.Count
    lngCnt = ActiveDocument.Characters.Count - lngChrCnt
    MsgBox "Fertig!", vbExclamation, "RemoveSpacesAfterParagraphBreak: " & CStr(lngCnt)
End Sub
```

```
Sub RemoveSpacesAfterParagraphBreak()
    ' 5.2 Leerzeichen NACH Absatzumbruch entfernen
    Dim lngChrCnt As Long   ' anfängliche Zeichenzahl im Dokument
    Dim lngCnt    As Long   ' Zähler f. entfernte Zeichen
    lngChrCnt = ActiveDocument.Characters.Count
    Application.ScreenUpdating = False
    With ActiveDocument.Content.Find
        .ClearFormatting
        .Text = "^p^w"
        .Style = ActiveDocument.Styles(wdStyleNormal)
        .Replacement.ClearFormatting
        .Replacement.Text = "^p"
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .Execute Replace:=wdReplaceAll
    End With
    Application.ScreenUpdating = True
    lngCnt = ActiveDocument.Characters.Count - lngChrCnt
    MsgBox "Fertig!", vbExclamation, "RemoveSpacesAfterParagraphBreak: " & CStr(lngCnt)
End Sub
```

```
Sub RemoveSpacesAroundHyphens(docTgt As Document, strCaption As String)
    ' 6. Ein oder mehr Leerzeichen vor/hinter Bindestrich entfernen.
    Dim CurrentColor    As WdColorIndex ' Farbindex
    Dim intChanges      as Integer      ' Zähler f. Ersetzungen
    Dim intHits         As Integer      ' Trefferzähler
    Dim lngRetVal       As Long         ' Rückgabewert d. MsgBox
    Dim objRng          As Range        ' Suchbereich
    Dim strListSep      As String * 1   ' Listentrennzeichen
    On Error GoTo ErrorHandler
    With Application
        .ScreenUpdating = True
        strListSep = .International(wdListSeparator) ' Listentrennzeichen ermitteln
        CurrentColor = .Options.DefaultHighlightColorIndex
        .Options.DefaultHighlightColorIndex = wdYellow
    End With
    Set objRng = docTgt.Range(0, 0)
    With objRng
        With .Find
            .ClearFormatting
            .Text = "(*)([ ]{1" & strListSep & "})(-)"   ' vor Bindestrich
```

```
            With .Replacement
                .ClearFormatting
                .Text = "\1\3"
            End With
            .Wrap = wdFindStop
            .Forward = True
            .Format = False
            .MatchCase = False
            .MatchWholeWord = False
            .MatchAllWordForms = False
            .MatchSoundsLike = False
            .MatchWildcards = True
            .Execute
            Do While .Found = True
                intHits = intHits + 1
                objRng.Duplicate.Select
                lngRetVal = MsgBox("Leerstelle(n) VOR Bindestrich ersetzen?", vbYesNo +
vbQuestion, strCaption)
                If lngRetVal = vbYes Then
                    intChanges = intChanges + 1
                    .Execute Replace:=wdReplaceOne
                    objRng.Collapse direction:=wdCollapseEnd
                End If
                .Execute
            Loop
        End With
    End With
    Set objRng = docTgt.Range(0, 0)
    With objRng
        With .Find
            .ClearFormatting
            .Text = "(*-)[ ]{1" & strListSep & "}"    ' nach Bindestrich
            With .Replacement
                .ClearFormatting
                .Text = "\1"
            End With
            .Wrap = wdFindStop
            .Forward = True
            .Format = False
            .MatchCase = False
            .MatchWholeWord = False
            .MatchAllWordForms = False
            .MatchSoundsLike = False
            .MatchWildcards = True
            .Execute
            Do While .Found = True
                intHits = intHits + 1
                objRng.Duplicate.Select
                lngRetVal = MsgBox("Leerstelle(n) NACH Bindestrich ersetzen?", vbYesNo +
vbQuestion, strCaption)
                If lngRetVal = vbYes Then
                    intChanges = intChanges + 1
                    .Execute Replace:=wdReplaceOne
                    objRng.Collapse direction:=wdCollapseEnd
                End If
                .Execute
            Loop
        End With
    End With
    MsgBox "Fertig! " & CStr(intHits) & " Fundstelle(n), davon " & _
           CStr(intChanges) & " ersetzt!", vbExclamation, strCaption
ExitPoint:
    On Error Resume Next
    With Application
        .ScreenUpdating = True
        .Options.DefaultHighlightColorIndex = CurrentColor
    End With
```

```
    Set objRng = Nothing
    Exit Sub
ErrorHandler:
    MsgBox "Laufzeitfehler: " & Err.Number & ", " & Err.Description, vbExclamation,
"Prozedur: 'RemoveSpacesAroundHyphens'"
    Resume ExitPoint
End Sub

Sub RemoveSpacingAroundEllipses(docTgt As Document, strCaption As String)
    ' 7. Leerstelle(n) vor/nach Auslassungszeichen entfernen
    Dim intChanges      As Integer      ' Zähler f. Ersetzungen
    Dim intHits         As Integer      ' Trefferzähler
    Dim intLoop         As Integer      ' Schleifenzähler
    Dim lngHighLight    As Long         ' Farbwert f. Hervorhebungen
    Dim lngRetVal       As Long         ' Rückgabewert d. MsgBox
    Dim objRng          As Range        ' Suchbereich
    Dim strListSep      As String * 1   ' Listentrennzeichen
    Dim varFindList     As Variant      ' Suchliste
    Application.ScreenUpdating = True
    strListSep = Application.International(wdListSeparator)
    varFindList = Array("[ ]{1" & strListSep & "}^0133", "^0133[ ]{1" & strListSep & "}") '
Suchbegriffe
    lngHighLight = Options.DefaultHighlightColorIndex
    For intLoop = LBound(varFindList) To UBound(varFindList)
        Set objRng = docTgt.Range(Start:=0, End:=0)
        With objRng.Find
            .ClearFormatting
            .Text = varFindList(intLoop)
            .Forward = True
            .Wrap = wdFindStop   ' wdFindContinue
            .Format = True
            .MatchCase = False
            .MatchWholeWord = False
            .MatchWildcards = True
            .MatchSoundsLike = False
            .MatchAllWordForms = False
            .Execute
            Do While .Found = True
                intHits = intHits + 1
                With objRng
                    .HighlightColorIndex = wdYellow
                    .Duplicate.Select
                End With
                lngRetVal = MsgBox("Ersetzen?" & vbCr & _
                   objRng.Text, vbYesNo + vbQuestion, strCaption)
                If lngRetVal = vbYes Then
                    intChanges = intChanges + 1
                    With objRng
                        If intLoop = LBound(varFindList) Then
                            .Text = LTrim(.Text)    ' vor
                        Else
                            .Text = RTrim(.Text)    ' nach
                        End If
                    End With
                End If
                With objRng
                    .HighlightColorIndex = wdNoHighlight
                    .Collapse direction:=wdCollapseEnd
                End With
                .Execute
            Loop
        End With
    Next intLoop
    With Application
        .Options.DefaultHighlightColorIndex = lngHighLight
        .ScreenUpdating = True
    End With
```

```
60      MsgBox CStr(intHits) & " Fundstelle(n), davon " & _
61             CStr(intChanges) & " ersetzt!", vbExclamation, strCaption
62  End Sub

 1  Sub ConvertQuotes(docTgt As Document, strCaption As String)
 2      ' 8. Gerade in typografische Anführungsstriche umwandeln oder umgekehrt.
 3      Dim bolOption   As Boolean  ' Optionswert
 4      Dim lngLoop     As Long     ' Scheifenzähler
 5      Dim lngRetVal   As Long     ' Rückgabewert d. MsgBox
 6      Dim objRng      As Range    ' Suchbereich
 7      Dim varFind     As Variant  ' Suchbegriff
 8      Dim varRepl     As Variant  ' Ersatzbegriff
 9      Application.ScreenUpdating = False
10      lngRetVal = MsgBox("Auf 'Ja' klicken, um typografische in gerade Anführungszeichen
11  umzuwandeln." & vbCr & _
12                         "Auf 'Nein' klicken, um gerade in typografische Anführungszeichen
13  umzuwandeln.", _
14                         vbYesNoCancel, strCaption)
15      If lngRetVal = vbCancel Then
16          GoTo ExitPoint
17      Else
18          bolOption = Options.AutoFormatAsYouTypeReplaceQuotes
19          If lngRetVal = vbYes Then
20              varFind = Array(ChrW(132), ChrW(147), ChrW(148), ChrW(171), ChrW(130),
21  ChrW(145), ChrW(146))
22              varRepl = Array(Chr(34), Chr(34), Chr(34), Chr(34), Chr(39), Chr(39), Chr(39))
23              Options.AutoFormatAsYouTypeReplaceQuotes = False
24              For lngLoop = LBound(varFind) To UBound(varFind)
25                  Set objRng = docTgt.Range
26                  With objRng.Find
27                      Do While .Execute(varFind(lngLoop))
28                          With objRng
29                              .Text = varRepl(lngLoop)
30                              .Collapse wdCollapseEnd
31                          End With
32                      Loop
33                  End With
34              Next lngLoop
35          ElseIf lngRetVal = vbNo Then
36              Options.AutoFormatReplaceQuotes = True
37              docTgt.Range.AutoFormat
38          End If
39      End If
40      With Application
41          .Options.AutoFormatAsYouTypeReplaceQuotes = bolOption
42          .ScreenUpdating = True
43      End With
44      MsgBox "ConvertQuotes: Normales Ende!", vbExclamation, strCaption
45  ExitPoint:
46  End Sub

 1  Sub ProperPunctuation(docTgt As Document, strCaption As String)
 2  ' 9. Zeichensetzung prüfen
 3  Dim intUpdCnt       As Integer  ' Zähler f. Änderungen
 4  Dim lngAsk          As Long     ' Rückgabewert d. MsgBox
 5  Dim lngHighLight    As Long     ' Hervorhebungsfarbe f. Änderungen
 6  Dim rngDoc          As Range    ' Suchbereich
 7  Dim strFindTxt      As String   ' Suchbegriff
 8  Dim strReplTxt      As String   ' Ersatztext
 9  On Error GoTo ErrorHandler
10  lngHighLight = Options.DefaultHighlightColorIndex
11  Options.DefaultHighlightColorIndex = wdTurquoise
12  ' Kein Leerzeichen vor Satzzeichen!
13  strFindTxt = "[ ][,;.:\!\?]"
14  strReplTxt = vbNullString
15  Set rngDoc = docTgt.Range
16  With rngDoc.Find
```

```
17          .ClearFormatting
18          .Replacement.ClearFormatting
19          .Style = wdStyleNormal              ' nur bei Formatvorlage "Standard"
20          .Replacement.Highlight = True       ' Ersetzung hervorheben
21          .Font.Underline = wdUnderlineNone   ' Hyperlinks (unterstrichene Textstellen) ausnehmen
22          Do While .Execute( _
23              FindText:=strFindTxt, _
24              MatchCase:=True, _
25              MatchWholeWord:=True, _
26              MatchWildcards:=True, _
27              Forward:=True, _
28              Wrap:=wdFindStop) = True
29              rngDoc.Select
30              lngAsk = MsgBox("Leerzeichen vor Satzzeichen gefunden! Entfernen?", vbYesNo,
31      strCaption)
32              If lngAsk = vbYes Then
33                  With rngDoc
34                      .Text = LTrim(.Text) ' Leerzeichen davor entfernen.
35                  End With
36                  intUpdCnt = intUpdCnt + 1
37              End If
38              rngDoc.Collapse direction:=wdCollapseEnd
39          Loop
40      End With
41      ' Hinter Satzzeichen ist 1 Leerzeichen erforderlich!
42      strFindTxt = "[,;.:\!\?][! " & ChrW(34) & ChrW(13) & "]"
43      strReplTxt = Space(1)
44      Set rngDoc = ActiveDocument.Range
45      With rngDoc.Find
46          .ClearFormatting
47          .Replacement.ClearFormatting
48          .Style = wdStyleNormal              ' nur bei Formatvorlage "Standard"
49          .Replacement.Highlight = True       ' Ersetzung hervorheben
50          .Font.Underline = wdUnderlineNone   ' Hyperlinks (unterstrichene Textstellen) ausnehmen
51          Do While .Execute( _
52              FindText:=strFindTxt, _
53              MatchCase:=True, _
54              MatchWholeWord:=True, _
55              MatchWildcards:=True, _
56              Forward:=True, _
57              Wrap:=wdFindStop) = True
58              rngDoc.Select
59              lngAsk = MsgBox("Kein Leerzeichen hinter einem Satzzeichen gefunden! Einfügen?",
60      vbYesNo, strCaption)
61              If lngAsk = vbYes Then
62                  With Selection
63                      .Collapse direction:=wdCollapseStart
64                      .MoveRight Unit:=wdCharacter, Count:=1
65                      .InsertAfter Text:=strReplTxt ' Leerzeichen dahinter einfügen.
66                  End With
67              End If
68              intUpdCnt = intUpdCnt + 1
69              rngDoc.Collapse direction:=wdCollapseEnd
70          Loop
71      End With
72      MsgBox CStr(intUpdCnt) & " Ersetzung(en) vor/hinter Satzzeichen durchgeführt.", _
73          vbInformation, strCaption
74      ExitPoint:
75          On Error Resume Next
76          Set rngDoc = Nothing
77          Options.DefaultHighlightColorIndex = lngHighLight
78          Exit Sub
79      ErrorHandler:
80          MsgBox "Laufzeitfehler: " & Err.Number & ", " & vbCrLf & _
81          Err.Description & ", " & Err.Source, vbCritical, "ProperPunctuation"
82          Resume ExitPoint
83      End Sub
```

```
Sub Satzzeichen_LZ(docTgt As Document, strCaption As String)
' 9.2 Überflüssige Leerzeichen vor und hinter Satzzeichen entfernen.
' Hinter Satzzeichen in der Markierung stets Leerzeichen, außer bei Gänsefüßchen.
' Unterstrichene Textstellen (Hyperlinks) sind ausgenommen.
' Ausgenommen ist auch, wenn hinter dem Satzzeichen eine Zahl folgt.
    Dim CurrentColor As WdColorIndex    ' Farbindex
    Dim intChoice    As Integer         ' Auswahl d. Benutzer
    Dim intUpdCnt    As Integer         ' Zähler f. Änderungen
    Dim objRng       As Range           ' Suchbereich
    Dim strFindTxt   As String          ' Suchbegriff
    Dim strReplTxt   As String          ' Ersatzbegriff
    Dim StyleNormal  As Style           ' Schreibstil
    On Error GoTo ErrorHandler
    Set objRng = ActiveDocument.Content
    If Len(objRng.Text) = 0 Then
        MsgBox "Nichts markiert!", vbExclamation, strCaption
        GoTo ExitPoint
    End If
    objRng.Collapse direction:=wdCollapseStart
    With Application
        .ScreenUpdating = True
        ' Aktuelle Hervorhebungsfarbe zwischenspeichern u. setzen.
        CurrentColor = .Options.DefaultHighlightColorIndex
        .Options.DefaultHighlightColorIndex = wdYellow
    End With
    ' Normalen Schreibstil vorgeben.
    Set StyleNormal = ActiveDocument.Styles(wdStyleNormal)
    ' Fehlendes Leerzeichen hinter Satzzeichen einfügen, OK!
    strFindTxt = "([,;.:\!\?])([A-Za-z])"       ' finden
    strReplTxt = "\1" & Space(1) & "\2"         ' ersetzen
    With objRng
        With .Find
            .ClearFormatting
            .Text = strFindTxt
            With .Replacement
                .ClearFormatting
                .Text = strReplTxt
                .Highlight = True
            End With
            .Style = StyleNormal
            .Forward = True
            .Wrap = wdFindStop ' wdFindContinue ' wdFindAsk
            .Format = True
            .Font.Underline = wdUnderlineNone ' keine Hyperlinks
            .MatchCase = False
            .MatchWholeWord = False
            .MatchAllWordForms = False
            .MatchSoundsLike = False
            .MatchWildcards = True
            .Execute
        End With ' Find
        Do While .Find.Found
            .Duplicate.Select
            Select Case MsgBox("Fehlendes Leerzeichen hinter Satzzeichen einfügen?", _
                        vbYesNoCancel, strCaption)
            Case vbCancel
                MsgBox "Abbruch durch den Benutzer!", vbExclamation, _
                    "Fehlendes Leerzeichen hinter Satzzeichen einfügen!"
                GoTo ExitPoint
            Case vbYes
                intUpdCnt = intUpdCnt + 1
                With Selection
                    .MoveLeft Unit:=wdCharacter, Count:=1, Extend:=wdExtend
                    .Text = .Text & Space(1)
                End With
            End Select
```

```
            .Collapse wdCollapseEnd
            .Find.Execute
        Loop
    End With
    ' Vor Satzzeichen oder schließender Klammer kein Leerzeichen.
    objRng.Collapse direction:=wdCollapseStart
    strFindTxt = "[ ]{1;}([.,:;\!\?\)\}\]])"    ' finden
    strReplTxt = "\1"                           ' ersetzen
    With objRng
        .Collapse direction:=wdCollapseStart
        With .Find
            .ClearFormatting
            .Text = strFindTxt
            With .Replacement
                .ClearFormatting
                .Text = strReplTxt
                .Highlight = True
            End With
            .Style = StyleNormal
            .Forward = True
            .Wrap = wdFindContinue
            .Font.Underline = wdUnderlineNone
            .Format = False
            .MatchCase = False
            .MatchWholeWord = False
            .MatchAllWordForms = False
            .MatchSoundsLike = False
            .MatchWildcards = True
            .Execute
        End With
        Do While .Find.Found
           .Duplicate.Select
            Select Case MsgBox("Leerzeichen vor Satzzeichen/schließender Klammer
entfernen?", vbYesNoCancel + vbDefaultButton1, "Ersetzen")
            Case vbCancel
                MsgBox "Abbruch durch den Benutzer!", vbExclamation, "Vor
Satzzeichen/schließender Klammer keine Leerzeichen!"
                GoTo ExitPoint
            Case vbYes
                intUpdCnt = intUpdCnt + 1
                .Text = Trim(Selection.Text)
                Selection.MoveRight Unit:=wdCharacter, Count:=2
            End Select
            .Collapse direction:=wdCollapseEnd
            .Find.Execute
        Loop
    End With
    MsgBox "Fertig! " & CStr(intUpdCnt) & " überflüssige Leerzeichen entfernt!",
vbInformation, strCaption
ExitPoint:
    On Error Resume Next
    With Application
        .ScreenUpdating = True
        .Options.DefaultHighlightColorIndex = CurrentColor  ' <--- Farbe zurücksetzen
    End With
    Set objRng = Nothing
    Exit Sub
ErrorHandler:
    MsgBox "Laufzeitfehler: " & Err.Number & ", " & Err.Description & ", Prozedur:
Satzzeichen_LZ"
    Resume ExitPoint
End Sub

Sub ChangelsToOnes(docTgt As Document, strCaption As String)
    ' 10. l-Zeichen, die keine 'eins' darstellen, in '1' umwandeln.
    Dim intChanges  As Integer  ' Zähler f. Ersetzungen
    Dim intHits     As Integer  ' Trefferzähler
```

```
    Dim objRng      As Range    ' Suchbereich
    Set objRng = docTgt.Range(0, 0)
    With objRng.Find
        .ClearFormatting
        .Text = "[0-9]{1;}l[0-9]{1;}"
        With .Replacement
            .Text = ""
            .Highlight = True
        End With
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = True
        .MatchCase = False
        .MatchWholeWord = False
        .MatchAllWordForms = False
        .MatchSoundsLike = False
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            objRng.Duplicate.Select
            If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "l", "1", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
            Selection.Collapse direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    Set objRng = ActiveDocument.Range(0, 0)
    With objRng.Find
        .Text = "l[0-9]{1;}"
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = False
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            objRng.Duplicate.Select
            If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "l", "1", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
            Selection.Collapse direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    Set objRng = ActiveDocument.Range(0, 0)
    With objRng.Find
        .Text = "[0-9]{1;}l"
        With .Replacement
            .Text = ""
            .Highlight = True
        End With
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = True
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            objRng.Duplicate.Select
           If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "l", "1", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
```

```
            Selection.Collapse direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    MsgBox CStr(intHits) & " Fundstelle(n), davon " & CStr(intChanges) & " ersetzt!",
vbExclamation, strCaption
End Sub

Sub ChangeOsToZeros(docTgt As Document, strCaption As String)
    ' 11. O-Zeichen, die keine null darstellen, in 0 umwandeln.
    Dim intChanges  As Integer  ' Zähler f. Ersetzungen
    Dim intHits     As Integer  ' Trefferzähler
    Dim objRng      As Range    ' Suchbereich
    Set objRng = docTgt.Range(0, 0)
    With objRng.Find
        .ClearFormatting
        .Text = "[0-9]{1;}O[0-9]{1;}"
        With .Replacement
            .Text = ""
            .Highlight = True
        End With
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = True
        .MatchCase = False
        .MatchWholeWord = False
        .MatchAllWordForms = False
        .MatchSoundsLike = False
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            objRng.Duplicate.Select
            If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "O", "0", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
            Selection.Collapse direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    Set objRng = ActiveDocument.Range(0, 0)
    With objRng.Find
        .Text = "O[0-9]{1;}"
        With .Replacement
            .Text = ""
            .Highlight = True
        End With
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = True
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            objRng.Duplicate.Select
            If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "O", "0", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
            Selection.Collapse direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    Set objRng = ActiveDocument.Range(0, 0)
    With objRng.Find
        .Text = "[0-9]{1;}O"
        With .Replacement
```

```
            .Text = ""
            .Highlight = True
        End With
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = True
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            objRng.Duplicate.Select
            If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "O", "0", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
            Selection.Collapse direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    MsgBox CStr(intHits) & " Fundstelle(n), davon " & CStr(intChanges) & " ersetzt!",
vbExclamation, strCaption
End Sub

Sub FindReplaceSearchTerm(docTgt As Document, strCaption As String)
    ' 12. Best. Zeichenfolge finden und ersetzen
    Dim intChanges  As Integer  ' Zähler f. Ersetzungen
    Dim intHits     As Integer  ' Trefferzähler
    Dim objRng      As Range    ' Suchbereich
    Dim strFindTxt  As String   ' Suchbegriff
    Dim strReplTxt  As String   ' Ersatzbegriff
    Set objRng = docTgt.Content
    If Len(objRng) < 1 Then
        MsgBox "Abbruch: Das aktuelle Dokument ist leer!", vbExclamation, strCaption
        GoTo ExitPoint
    End If
    Do While Len(strFindTxt) < 1
        strFindTxt = InputBox("Gesuchte Zeichenfolge", strCaption)
        If strFindTxt = vbNullString Then
            MsgBox "Abbruch: Eingabe des Suchbegriffs ist Pflicht!", vbExclamation, strCaption
            GoTo ExitPoint
        End If
    Loop

    Do While Len(strReplTxt) < 1
        strReplTxt = InputBox("Ersatzbegriff", strCaption)
        If strReplTxt = vbNullString Then
            MsgBox "Abbruch: Eingabe d. Ersatzbegriffs ist Pflicht!", vbExclamation, strCaption
            GoTo ExitPoint
        End If
    Loop
    With objRng
        With .Find
            .ClearFormatting
            .Text = strFindTxt
            .Style = ActiveDocument.Styles(wdStyleNormal)
            With .Replacement
                .ClearFormatting
                .Text = ""
                .Highlight = True
            End With
            .MatchWholeWord = True
            .Forward = True
            .Format = False
            .MatchCase = True
            .MatchWildcards = False
            .Wrap = wdFindStop
            .Execute
```

```
        End With
        Do While .Find.Found
            ' Die Auswahl auf den gesamten Absatz ausweiten: => objRng.Expand Unit:=wdParagraph
            ' Fensterinhalt d. Dokumentfensters verschieben, bis der angegebene Suchbereich darin angezeigt wird:
            ' ActiveWindow.ScrollIntoView obj:=Selection.Range, Start:=True
            .Duplicate.Select   ' Fundstelle markieren
            intHits = intHits + 1
            Select Case MsgBox("Ersetzen:" & _
               vbCr & vbTab & strFindTxt & vbCr & "durch:" & _
               vbCr & vbTab & strReplTxt, vbYesNoCancel, strCaption)
            Case vbCancel
                GoTo ExitPoint
            Case vbYes
                .Text = strReplTxt
                intChanges = intChanges + 1
            Case Else
                ' nichts tun
            End Select
            .Collapse direction:=wdCollapseEnd
            .Find.Execute
        Loop
    End With
    MsgBox CStr(intHits) & " Fundstelle(n), davon " & CStr(intChanges) & " ersetzt!",
vbExclamation, strCaption
ExitPoint:
   On Error Resume Next
   Exit Sub
ErrorHandler:
    MsgBox Err.Number & ", " & Err.Description & " in Prozedur 'FindReplaceSearchTerm'"
    Resume ExitPoint
End Sub

Sub AddPeriodToFootnotes(docTgt As Document, strCaption As String)
    ' 13. Fehlendes Satzendezeichen in Fußnote einfügen
    Const conLast As String = ".?!" ' Zeichen für Satzende
    Dim ftIdx     As Integer        ' Schleifenzähler f. Fußnoten
    Dim ftRng     As Range          ' Fußnotenbereich
    Dim intHits   As Integer        ' Trefferzählen (f. fehlendes Satzendezeichen)
    Dim strLast   As String * 1     ' Letztes Zeichen einer Fußnote
    On Error GoTo ErrHandler
    intHits = 0
    For ftIdx = 1 To ActiveDocument.Footnotes.Count
        Set ftRng = ActiveDocument.Footnotes(ftIdx).Range
        With ftRng
            strLast = Right(.Text, 1)
            If InStr(1, conLast, strLast, vbTextCompare) = 0 Then
               ' Satzendezeichen
               .Collapse direction:=wdCollapseEnd
               .Text = .Text & "."
               intHits = intHits + 1
            End If
        End With
    Next ftIdx
    If intHits = 1 Then
        MsgBox "1 fehlendes Zeichen am Ende einer Fußnote eingefügt.", vbInformation,
strCaption
    Else
        MsgBox CStr(intHits) & " fehlende Zeichen am Ende von Fußnoten eingefügt.",
vbInformation, strCaption
    End If
ExitPoint:
    On Error Resume Next
    Exit Sub
ErrHandler:
    If Err.Number = 5941 Then
        MsgBox "Keine Fußnoten im aktuellen Dokument gefunden", vbExclamation, strCaption
    End If
    Resume ExitPoint
```

```
37 End Sub
```

5.2 Modul *Setup*

```
Option Base 1
Option Explicit
Dim strTgtFileNm As String
```

```
Private Sub UserForm_Initialize()
    With frmGetFile                  ' Anfangswerte setzen
      .Left = 10                     ' Position links
      .Top = 10                      ' Position oben
      .StartUpPosition = 0           ' manuelle Positionierung
      .lblFileNm.Caption = ""        ' Bezeichnungsfeld leeren
      .cmdCancel.Enabled = True      ' Schaltfläche aktivieren
      .cmdStart.Enabled = False      ' Schaltfläche deaktivieren
      .Show                          ' Benutzerformular anzeigen
    End With
End Sub
```

```
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
    If CloseMode = vbFormControlMenu Then
        Cancel = True
        Call cmdCancel_Click
    End If
End Sub
```

```
Private Sub cmdCancel_MouseMove(ByVal Button As Integer, _
        ByVal Shift As Integer, ByVal x As Single, ByVal y As Single)
          frmGetFile.cmdCancel.SetFocus
End Sub
```

```
Private Sub cmdStart_MouseMove(ByVal Button As Integer, _
        ByVal Shift As Integer, ByVal x As Single, ByVal y As Single)
           frmGetFile.cmdStart.SetFocus
End Sub
```

```
Private Sub cmdCancel_Click() ' Schaltfläche "Abbruch"
    Dim intRetVal As Integer, strMsg As String
    strMsg = "Sind Sie sicher, dass Sie abbrechen und beenden möchten?"
    intRetVal = MsgBox(strMsg, vbQuestion + vbYesNo, "Abbruch?")
    Select Case intRetVal
        Case vbYes
            frmGetFile.Hide
            Unload frmGetFile
        Case Else
            Exit Sub
    End Select
End Sub
```

```
1
1   Private Sub cmdStart_Click()       ' Schaltfläche "Start"
2       Dim ctrl       As Control      ' Steuerelement
3       Dim docTgt     As Document     ' Zieldokument
4       Dim lngRtnCode As Long         ' Rückgabewert der MsgBox
5       Dim strFileNm  As String       ' Dateiname
6       Dim strMdlNm   As String       ' Modulname
7       Dim strProcNm  As String       ' Prozedurname
8       strTgtFileNm = frmGetFile.lblFileNm.Caption
9       For Each ctrl In frmGetFile.Controls
10         If TypeOf ctrl Is MSForms.CheckBox Then
11            If ctrl.Value = True Then
12               MsgBox ctrl.Name & " & ctrl.tag"
13            End If
14         End If
15      Next ctrl
16      On Error GoTo ErrorHandler
17      strMdlNm = "Cleaning"      ' Modulname
18      If ProcExists(ActiveDocument, strMdlNm, strProcNm) = True Then
19          strFileNm = frmGetFile.lblFileNm   ' Dateiname aus Benutzerformular
20          If Len(strFileNm) > 0 And Dir(strFileNm) <> vbNullString Then
21             lngRtnCode = MsgBox("'" & Dir(strFileNm) & "' öffnen und bereinigen?",
22  vbQuestion + vbYesNo, "Word-Datei öffnen und bereinigen?")
23             If lngRtnCode = vbYes Then
24                Set docTgt = Documents.Open(FileName:=strFileNm)         , Zieldokument
25                docTgt.Activate
26                Application.Run Macroname:=strProcNm, varg1:=docTgt   , Prozedur ausführen
27             Else
28                MsgBox „Es wurde ‚Abbruch‘ wurde gewählt!“, vbOKOnly, „Word-Datei öffnen“
29             End If
30          Else
31             MsgBox „Zieldokument konnte nicht geöffnet werden!“, vbExclamation,
32  „GetWordFile“
33          End If
34      Else
35          MsgBox „Die ausgewählte VBA-Prozedur existiert nicht!“, vbExclamation, strProcNm
36      End If
37  ExitPoint:
38  On Error Resume Next
39  Exit Sub
40  ErrorHandler:
41  MsgBox “Laufzeitfehler # “ & Err.Number & “, “ & Err.Description, vbCritical, “cmdStart”
42  Resume ExitPoint
43  End Sub
```

5.3 Modul *Tools*

```
Sub OpenWordFile()
   ' Aufgabe: Word-Datei auswählen, öffnen und interaktiv bereinigen.
   Dim strFullNm  As String    ' vollst. Dateiname d. ausgewählten Zieldokuments
   Dim lngRtnCode As Long      ' Rückgabewert d. MsgBox
   On Error GoTo ErrorHandler  ' Fehlerroutine
   With Application.FileDialog(msoFileDialogOpen)
      .Title = "Dateiauswahl"
      .ButtonName = "Auswählen"
      .AllowMultiSelect = False
      .InitialFileName = ThisDocument.Path
      With .Filters
         .Clear
         .Add "Word-Dateien (*.docx)", "*.docx", 1
         .Add "Word-Dateien (*.docm)", "*.docm", 2
         .Add "Alle Dateien (*.*)", "*.*", 3
      End With
      .FilterIndex = 1
      If .Show = True Then
         strFullNm = CStr(.SelectedItems(1)) ' Pfadname + Dateiname
         If Len(strFullNm) > 0 And Dir(strFullNm) <> vbNullString Then
            With frmGetFile
                .lblFileNm = strFullNm
                .cmdStart.Enabled = True
            End With
         Else
            MsgBox "Zieldokument kann nicht geöffnet werden", vbExclamation, "OpenWordFile"
         End If
      Else
         MsgBox "Es wurde nichts ausgewählt!", vbOKOnly, "Dateiauswahl"
      End If
   End With
ExitPoint:
   Exit Sub
ErrorHandler:
   MsgBox "Laufzeitfehler: " & Err.Description, vbCritical, "OpenWordFile"
   Resume ExitPoint
End Sub
```

```
1   Public Function ProcExists(docSrc As Document, strMdlNm As String, strProcNm As String) As
2   Boolean
3      ' Aufgabe: Existenzprüfung einer Prozedur.
4      ' Verweis erforderlich auf: 'Microsoft Visual Basic for Applications Extensibility 5.3'
5      Dim CodeMod As VBIDE.CodeModule, lngLine As Long
6      ProcExists = False
7      Err.Clear
8      On Error Resume Next
9      Set CodeMod = docSrc.VBProject.VBComponents(strMdlNm).CodeModule
10     If Err.Number = 0 Then
11        lngLine = CodeMod.ProcStartLine(strProcNm, vbext_pk_Proc)
12        If Err.Number = 0 Then ProcExists = True
13     End If
14  End Function
```