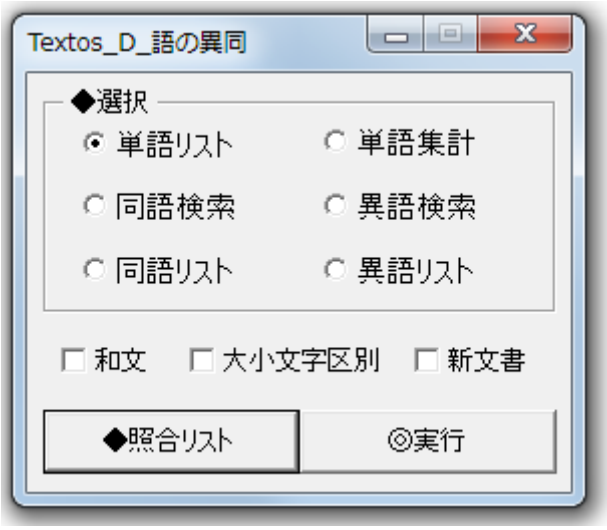
単語の比較とスペルチェック COMPARA

Word のテキスト内にある単語を抽出し、異なり語のリストを作成するプログラムです。また、Word のテキスト内の単語（または単語リスト）を、先に作成した単語リストと照合して、その異同を確かめます。

フォーム



単語リスト	テキストまたはリストにある単語（異なり語）のリストを作成します。
単語集計	単語の頻度を計算します。
同語検索	現文書の単語が照合リストに存在すれば、それを青色でマークします。あらかじめ、文書の中でよく使われる語のリストを作成すれば、繰り返された語をチェックすることができます。また、注目すべき語をマークすることができます。
異語検索	現文書の単語が照合する単語リストに存在しなければ、それを緑色でマークします。スペルチェックに利用します。
同語リスト	照合リストに存在する現文書の単語リストを出力します。
異語リスト	照合リストに存在しない現文書の単語リストを出力します。
和文	日本語（欧文を含む）の単語を対象にします。欧文だけを対象として高速化するときはチェックをしないでください。
大小文字区別	単語リストを作成するときに大小文字を区別します。
新文書	新文書に出力します。
照合リスト	次の同語検索と異語検索に使う照合文書を入力します。
実行	選択したオプションを実行します。

データ例(1)

単語リスト：テキストの異なる単語リストを作成します。

単語リスト

単語
リスト
：
テキスト
の
異なる
を
作成
します
。

単語集計

単語　2
リスト　2
：　1
テキスト　1
の　1
異なる　1
を　1
作成　1
します　1
。　1
　2

同語検索

上の単語リストの出力を単語.docx として保存し、これを照合リストとします。「照合リスト」ボタンをクリックして、この単語.docx を選択してください。これは変更しない限り一度だけです。

データ例(2)

異なる単語の頻度を計算します。

同語リスト

異なる
単語
の
を
します
。

異語検索

異なる単語の頻度を計算します。

異語リスト

頻度
計算

コントロール

オブジェクト名	キャプション
Textos_D_語の異同フォーム	TEXTOS-D: 語の異同
frm 選択	◆選択
cmd 照合リスト	◆照合リスト
cmd 実行	◎実行
lbl 経過	◎経過

コード

```
Option Explicit '変数を明示
  Dim obj 連想配列  As Object, obj 連想配列 2 As Object
  Dim obj 正規表現  As Object
  Dim  カウンタ&, 語数&, 出力文字数&, 出力文字列$, 作業用文字列$
  Dim var 単語配列, var 単語
```

バリアント型の変数は変数型を指定しなくてもよいのですが、ここではわかりやすくするために、var という接頭辞をつけています。

```
Private Sub UserForm_Activate() 'ユーザーフォームを表示
  Set obj 連想配列  = CreateObject("Scripting.Dictionary")
  Set obj 連想配列 2 = CreateObject("Scripting.Dictionary")
```

```
  '連想配列オブジェクトを生成
  Set obj 正規表現 = CreateObject("VBScript.RegExp")
  '正規表現オブジェクトを生成
  obj 正規表現.Global = True '全体検索
End Sub
```

連想配列オブジェクトと正規表現オブジェクトについては→「準備」

```
Private Sub cmd 実行_Click() '実行ボタンクリック
    If Len(Selection) < 2 Then MsgBox "範囲を選択してください。": Exit Sub

  語数& = Selection.Words.Count '選択範囲の語数
  出力文字列$ = "" '初期化
  出力文字数& = 0
  カウンタ& = 0
    If Len(Selection) < 2 Then
    If MsgBox("全範囲を選択しますか？", vbYesNoCancel, "範囲を選択していません。") = vbYes Then
      Selection.WholeStory '全範囲選択
    Else
      Exit Sub '実行終了
    End If
  End If

  出力文字列$ = "" '初期化
  出力文字数& = 0
  カウンタ& = 0

  If chk 新文書 Then Call 新文書

  If opt 単語リスト And chk 和文 Then Call 和文単語リスト
  If opt 単語リスト And Not chk 和文 Then Call 欧文単語リスト

  If opt 単語集計 And chk 和文 Then Call 和文単語集計
  If opt 単語集計 And Not chk 和文 Then Call 欧文単語集計

  If opt 同語検索 Or opt 異語検索 Then Call 同語異語検索
```

```
  If (opt 同語リスト  Or opt 異語リスト) And chk 和文 _
    Then Call 和文同語異語リスト
  If (opt 同語リスト  Or opt 異語リスト) And Not chk 和文 _
    Then Call 欧文同語異語リスト
End Sub
```

実行ボタンクリック時に、範囲の指定を確認し、出力文字に関する変数を初期化します。新文書出力であれば「新文書」サブルーチンを呼び込みます。次に、それぞれのオプションボタンの選択に従って対応するサブルーチンを呼び込みます。同語検索と異語検索の手順はほとんど同じですが、異なる部分（連想配列との照合の仕方）についてはサブルーチンの中で区別します。同語リストと異語リストについても同様です。

```
Private Sub  新文書()
  Selection.Copy '選択範囲コピー
  Documents.Add '新文書追加
  Selection.Paste 'ペースト
  Selection.WholeStory '全範囲選択
End Sub
```

選択範囲を新文書にコピー&ペーストし、全範囲を選択します。

```
Private Sub cmd 照合リスト_Click()
  Dim FW&, SW& '最小化ボタン用変数

  FW& = FindWindow("ThunderDFrame", Textos_D_語の異同フォーム.Caption)
  'ハンドル
  SW& = SetWindowPos(FW&, 1, 150, 150, 0, 0, &H1) 'フォームを後面に

  With Dialogs(wdDialogFileOpen) 'ダイアログボックス
    .Name = "*.docx": .Display 'ダイアログボックス表示

    If InStr(.Name, "*") Or .Name = "" Then Exit Sub '文書名なし→終了

    Documents.Open FileName:=.Name '選択した文書を開く
    cmd 照合リスト.Caption = .Name '文書名をキャプションに代入
  End With
```

```
  If MsgBox("OK、またはキャンセル？", vbOKCancel) = vbCancel Then '確認
    ActiveDocument.Close '現文書を閉じる
    Exit Sub
  End If

  var 単語配列 = Split(ActiveDocument.Range.Text, vbCr) '改行でスプリット
  ActiveDocument.Close '現文書を閉じる
  obj 連想配列.RemoveAll '連想配列のキーとアイテムを削除

  For Each var 単語 In var 単語配列 '単語リストファイル
    作業用文字列$ = 特殊文字除去$(var 単語)
    obj 連想配列(作業用文字列$) = "" 'アイテムに空白を代入
  Next

  SW& = SetWindowPos(FW&, -1, 150, 150, 0, 0, &H1) 'フォームを前面に
End Sub
```

ダイアログボックスで改行で区切られた単語リストの文書を開き、文書の内容を開業でスプリットして配列を作成し、その配列のキーを連想配列のキーとしてアイテムに空白ををを代入します。

```
Private Sub UserForm_QueryClose(Cancel%, CloseMode%)
'×印終了ボタン
  Set obj 連想配列 = Nothing '連想配列オブジェクトを解放
  Set obj 正規表現 = Nothing '正規表現オブジェクトを解放
  End '終了
End Sub
```

×印終了ボタンを押したときに連想配列を開放し終了します。

```
Private Sub 和文単語リスト()
  obj 連想配列.RemoveAll '連想配列のキーとアイテムを削除

  For Each var 単語 In Selection.Words '選択範囲の語について
    作業用文字列$ = 和文文字列$(var 単語)

    If Not obj 連想配列.Exists(作業用文字列$) Then '連想配列にキーがなければ…
```

```
        obj 連想配列(作業用文字列$) = 1 'アイテムに 1 を代入
        作業用文字列$ = 作業用文字列$ & vbCr '単語＆改行
        Call 高速文字列作成
      End If

      Call 経過表示
    Next

    Call 文字列出力
End Sub
```

For … Next で選択範囲の語を連想配列に格納します。アイテムは何でもかまいませんが、ここではキーによって格納の有無が判断されます。格納されていない語は新語（異語）となるので、これに改行コードを加えて、高速文字列作成を行います。

```
Private Sub 欧文単語リスト()
  作業用文字列$ = Replace(Selection.Text, vbCr, " ") '改行を空白に
  var 単語配列 = Split(作業用文字列$, " ") '空白でスプリット

  obj 連想配列.RemoveAll '連想配列のキーとアイテムを削除

  For Each var 単語 In var 単語配列 '選択範囲の語について
    作業用文字列$ = 欧文文字列$(var 単語)

    If 作業用文字列$ <> "" And Not obj 連想配列.Exists(作業用文字列$) Then
    '連想配列にキーがなければ…
      obj 連想配列(作業用文字列$) = "" 'アイテムに空白を代入
      作業用文字列$ = 作業用文字列$ & vbCr '単語＆改行
      Call 高速文字列作成
    End If
  Next

  Call 文字列出力
End Sub
```

前の和文単語リストとよく似ていますが、欧文ではスペースをセパレータ（区切り記号）として使えるので、Split 関数によって配列を作成します。欧文でも Word の Words コレクションを利用することは可能ですが、配列を使うほうが単語オブジェ

クトを参照するよりも高速になります。

```
Private Sub 和文単語集計()
  obj 連想配列.RemoveAll '連想配列のキーとアイテムを削除

  For Each var 単語 In Selection.Words '選択範囲の語について
    作業用文字列$ = 和文文字列$(var 単語)

    If Not obj 連想配列.Exists(作業用文字列$) Then
    '連想配列にキーがなければ…
      obj 連想配列(作業用文字列$) = 1
    Else
    '連想配列にキーがあれば…
      obj 連想配列(作業用文字列$) = obj 連想配列(作業用文字列$) + 1
    End If
  Next

  For Each var 単語 In obj 連想配列.Keys 'それぞれのキーについて
    作業用文字列$ = var 単語 & vbTab & obj 連想配列(var 単語) & vbCr
    'キーとアイテムの連続
    Call 高速文字列作成
    Call 経過表示
  Next

  Call 文字列出力
End Sub
```

はじめの For ... Next で選択範囲の語をキーとし、その頻度をアイテムとして格納します。次の For ... Next でキー（語）とアイテム（頻度）の連続を出力します。

```
Private Sub 欧文単語集計()
  作業用文字列$ = Replace(Selection.Text, vbCr, " ") '改行を空白に
  var 単語配列 = Split(作業用文字列$, " ") '空白でスプリット

  obj 連想配列.RemoveAll '連想配列のキーとアイテムを削除

  For Each var 単語 In var 単語配列 '選択範囲の語について
    作業用文字列$ = 欧文文字列$(var 単語)
```

```
    If 作業用文字列$ <> "" Then
      If Not obj 連想配列.Exists(作業用文字列$) Then
      '連想配列にキーがなければ…
        obj 連想配列(作業用文字列$) = 1
      Else
      '連想配列にキーがあれば…
        obj 連想配列(作業用文字列$) = obj 連想配列(作業用文字列$) + 1
      End If
    End If
  Next

  For Each var 単語 In obj 連想配列.Keys 'それぞれのキーについて
    作業用文字列$ = var 単語 & vbTab & obj 連想配列(var 単語) & vbCr
    'キーとアイテムの連続
    Call 高速文字列作成
  Next

  Call 文字列出力
End Sub
```

前の和文単語集計と似ていますが、欧文単語リストと同じ理由で文字型変数の配列を使用します。

```
Private Sub 同語異語検索()
  For Each var 単語 In Selection.Range.Words '選択範囲の語について
    If chk 和文 Then 作業用文字列$ = 和文文字列$(var 単語)
    If Not chk 和文 Then 作業用文字列$ = 欧文文字列$(var 単語)

    If 作業用文字列$ <> "" _
    And obj 連想配列.Exists(作業用文字列$) = opt 同語検索 Then
      If opt 同語検索 Then var 単語.HighlightColorIndex = wdTurquoise
      '同語検索→蛍光色＝水色
      If opt 異語検索 Then var 単語.HighlightColorIndex = wdBrightGreen
      '異語検索→蛍光色＝明るい緑色
    End If
```

```
    Call 経過表示
  Next
End Sub
```

選択範囲のそれぞれの語について特殊文字を除去し、その語の連想配列内の有無と同語・異語検索のオプションが一致すれば、それぞれの色（水色または明るい緑色）の蛍光ペンで塗ります。

```
Private Sub 和文同語異語リスト()
  For Each var 単語 In Selection.Range.Words '選択範囲の語について
    var 単語 = 和文文字列$(var 単語)

    If var 単語 <> "" And obj 連想配列.Exists(var 単語) = opt 同語リスト _
    And Not obj 連想配列 2.Exists(var 単語) Then
      作業用文字列$ = var 単語 & vbCr '単語と改行の連続
      Call 高速文字列作成
      obj 連想配列 2(var 単語) = "" '存在の標識
    End If

    Call 経過表示
  Next

  Call 文字列出力
End Sub
```

選択範囲のそれぞれの語について特殊文字を除去し、その語の連想配列内の有無と同語・異語検索のオプションが一致し、、出力文字列に未だ存在しない語を高速文字列作成によって出力文字列に追加し、すべての単語の検索を修了したら、文字列を出力します。

```
Private Sub 欧文同語異語リスト()
  作業用文字列$ = Replace(Selection.Text, vbCr, " ") '改行を空白に
  var 単語配列 = Split(作業用文字列$, " ") '空白でスプリット

  For Each var 単語 In var 単語配列 '選択範囲の語について
    var 単語 = 欧文文字列$(var 単語)

    If var 単語 <> "" And obj 連想配列.Exists(var 単語) = opt 同語リスト _
```

```
    And Not obj 連想配列 2.Exists(var 単語) Then
      作業用文字列$ = var 単語 & vbCr '単語と改行の連続
      Call 高速文字列作成
      obj 連想配列 2(var 単語) = "" '存在の標識
    End If
  Next

  Call 文字列出力
End Sub
```

前の和文同語異語リストと似ていますが、欧文単語リストと同じ理由で文字型変数の配列を使用します。

```
Private Sub 高速文字列作成()
  If 出力文字数& + Len(作業用文字列$) > Len(出力文字列$) Then
    出力文字列$ = 出力文字列$ & Space(99999) '領域超過→領域拡張
  End If

  Mid$(出力文字列$, 出力文字数& + 1) = 作業用文字列$ '追加文字列挿入
  出力文字数& = 出力文字数& + Len(作業用文字列$) '全文字列の長さ
End Sub
```

文字列を次々に追加していくと、その度にメモリーの確保を行うので時間がかかります。ここでは、文字列の作成を高速化するために一定の長さを超えた時、大きなスペースの領域を拡張し、その拡張した文字列の中に、追加文字列を挿入していきます。

```
Private Sub 文字列出力()
  Selection.Delete '選択範囲を削除（文字装飾を除去）
  Selection = Left$(出力文字列$, 出力文字数&) '文字列の該当部分を出力
End Sub
```

選択範囲にあるデータを削除し、そこに文字列の該当部分（長さを m&とする左部分）を代入します。

```
Private Function 欧文文字列$(対象)

  obj 正規表現.IgnoreCase = Not chk 大小文字区別 '大小文字区別を判定
  obj 正規表現.Pattern = "[^a-zA-Z¥xC0-¥xD6¥xD8-¥xF6¥xF8-¥xFF]+"
```

```
  '検索パターン
  欧文文字列$ = obj 正規表現.Replace(対象, "") '置換・代入

  If Not chk 大小文字区別 Then 欧文文字列$ = LCase(欧文文字列$)
  '大小文字区別ない→小文字に変換
End Function
```

欧文文字列だけを抽出します。

```
Private Function 和文文字列$(対象)
  obj 正規表現.IgnoreCase = Not chk 大小文字区別 '大小文字区別を判定
  obj 正規表現.Pattern = "[^a-zA-Z¥xC0-¥xD6¥xD8-¥xF6¥xF8-¥xFF ｧ-ﾝぁ-ゞ
一-鶴]+"
  '検索パターン
  和文文字列$ = obj 正規表現.Replace(対象, "") '置換・代入

  If Not chk 大小文字区別 Then 和文文字列$ = LCase(和文文字列$)
  '大小文字区別ない→小文字に変換
End Function
```

和文文字列だけを抽出します。

2012/5/23 textos-d.docm H. Ueda