Na úvod dva pojmy – třídění a řazení. Správnější je řazení. Ve skutečnosti netřídíme data do nějakých skupin (tříd), ale jde nám o stanovení pořadí. Oba pojmy jsou ovšem natolik zažité, že není velkou chybou mluvit o třídících algoritmech a setřídění. Co pořadí definuje? U čísel je to samozřejmě jejich velikost, tj. umístění na číselné ose. A co určuje pořadí textových řetězců? Textový řetězec sestává z jednotlivých znaků, takže by se dalo čekat, že Excel bere znak po znaku a porovnává jejich pořadí v tabulce ASCII (ANSI). Nicméně jazykové sady obsahují i některé výjimky. V naší abecedě k nim patří spojení znaků „c“ a „h“ coby písmeno „ch“. Ale od začátku…

Nápověda Excelu k tématu řazení stojí za starou bačkoru. Velmi strohé informace jsem našel na stránce SharePointu a Excel Services.

Ani odtud se toho bohužel moc nedozvíme ve vztahu k české lokalizaci Excelu. Lepším zdrojem informací je web Charlese Williamse.

Vzal jsem znaky české kódové stránky Windows 1250 a vložil je do Excelu, následně přihodil do kotle chybové a pravdivostní hodnoty, prázdnou buňku, prázdný řetězec, apostrof, datumy, časy a další data, a začal experimentovat.

První na ráně je karta Data a nástroje skupiny Seřadit a filtrovat. Výsledek řazení A-Z ukazuje obrázek (přeuspořádáno z důvodu úspory místa, čtěte zleva doprava a dolů jako v knize).

Vzestupné řazení na listu – Data / Seřadit

Z obrázku můžeme vysledovat následující:
První se řadí čísla, mezi něž patří i zmíněný datum a čas (interně vyjádřené celým číslem či zlomkem). Za nimi se objevují na oko prázdné buňky – prázdný řetězec ("") a apostrof (', jako formátovací znak viditelný pouze v řádku vzorců). Následuje viditelný apostrof, pomlčky, oba typy mezery a řada symbolů. Až poté se objevují znaky abecedy. A tady se na chvilku zastavíme. Nástroje ve skupině Seřadit a filtrovat běžně nerozlišují velikost písmen (A = a, což odpovídá textovému porovnávání) a pracují v duchu stabilního řazení, tj. nemění původní vzájemné pořadí položek se stejným klíčem. Toto chování lze ovlivnit pod tlačítkem Seřadit / Možnosti / Rozlišovat malá a velká (A-Z: a < A, e < ë < é < ě, ch < Ch < CH, u < ü < ű < ú < ů).

Seřadit – rozlišování velikosti písmen

Písmeno „ch“ se začlení správně v souladu s českou abecedou (řazení se řídí tzv. „collating sequence“), nicméně posloupnost dvojice velké-malé písmeno není dodržena. Na konci seznamu najdeme pravdivostní hodnoty (NEPRAVDA < PRAVDA, v logice 0 < 1, nikoli N < P), chybové hodnoty (neřadí se podle názvu, jsou si rovnocenné, vztahuje se na ně pravidlo stabilního řazení) a jako úplně poslední se objeví skutečně prázdná buňka (bude vždy poslední bez ohledu na řazení A-Z, Z-A).

Co prostý test rovnosti obsahu dvou buněk?
Nyní porovnáme dříve testované hodnoty vůči sobě v prostém vzorci. V tomto případě na rozdíl od předchozího Excel nerozlišuje velikost písmen (bez ohledu na užití funkce STEJNÉ), a zrovna tak nevidí rozdíl u prázdných buněk (prázdná, prázdný řetězec, formátovací apostrof). Písmeno „ch“ respektuje českou abecedu.

Jak je tomu ve VBA?
Pro porovnávání řetězců slouží vestavěná funkce StrComp, která nabízí porovnání binární a textové. Nechová se bohužel ve všech případech stejně jako řazení na listu.

```vba
'Option Compare Text

Sub TestStrCompare()

    Dim strA As String
    Dim strB As String
    Dim intSrovnani As Integer

    'StrComp
    'vbUseCompareOption ... nastavení se řídí Option Compare v úvodu modulu
    'vbBinaryCompare ... binární porovnávání (a < á, A < a)
    'vbTextCompare ... textové porovnávání (a < á, a = A, tj. nerozlišuje velikost písmen)
    'vbDatabaseCompare ... pouze pro Microsoft Access, řídí se nastavením databáze)

    'výsledek intSrovnani
    '-1 ... strA < strB
    ' 0 ... strA = strB
    ' 1 ... strA > strB

    strA = "a"
    strB = "A"
    '1
    intSrovnani = StrComp(strA, strB, vbBinaryCompare)
    '0
    intSrovnani = StrComp(strA, strB, vbTextCompare)

    strA = "a"
    strB = "á"
    '-1
    intSrovnani = StrComp(strA, strB, vbBinaryCompare)
    '-1
    intSrovnani = StrComp(strA, strB, vbTextCompare)

    strA = "ci"
    strB = "ch"
    '1, nerespektuje českou abecedu ("collating sequence")
    'tj. neodpovídá řazení na listu
    intSrovnani = StrComp(strA, strB, vbBinaryCompare)
    '-1
    intSrovnani = StrComp(strA, strB, vbTextCompare)

    strA = "ch"
    strB = "Ch"
    '1, neodpovídá řazení na listu pro tento znak
    'ale je v souladu s ostatním řazením na listu (A < a)
    intSrovnani = StrComp(strA, strB, vbBinaryCompare)
    '0
    intSrovnani = StrComp(strA, strB, vbTextCompare)

End Sub
```

A jak ve VBA vlastně provádět řazení?

Nejčastěji ho potřebujeme uskutečnit na poli hodnot. Jenže samotné VBA pro tyto účely nenabízí zhola nic. Možná i vy jste se prokousávali teorií, studovali Bubble Sort, Merge Sort, Counting Sort, Selection Sort a nakonec našli algoritmického vítěze – rekurzivní verzi Quick Sortu. Nejedná se o ovšem o stabilní řazení (položky se stejným klíčem se mohou prohodit), a pokud u textových řetězců používá prostý test (< , >, =), pak jeho aplikování na češtině nedopadne dobře (C < Ch < c < ch < ci < č).

```
Sub TrideniQuickSortTest()

  Dim Pole1()
  Dim Pole2()

  Pole1 = Array(10, 2, 15, 125, 26, 18, 33, 35, 26, 51, 106)
  Pole2 = Array("Gates", "gates", "Windows", "Office", "windows")

  QuickSort Pole1, LBound(Pole1), UBound(Pole1)
  QuickSort Pole2, LBound(Pole2), UBound(Pole2)

End Sub

Public Sub QuickSort(vArray As Variant, inLow As Long, inHi As Long)

' velká číselná pole, jejichž počet položek je značně menší
' než největší hodnota
' textová pole

  Dim pivot As Variant
  Dim tmpSwap As Variant
  Dim tmpLow As Long
  Dim tmpHi As Long

  tmpLow = inLow
  tmpHi = inHi

  pivot = vArray((inLow + inHi) \ 2)

  While (tmpLow < = tmpHi)

    While (vArray(tmpLow) < pivot And tmpLow < inHi)
      tmpLow = tmpLow + 1
    Wend

    While (pivot < vArray(tmpHi) And tmpHi > inLow)
      tmpHi = tmpHi - 1
    Wend

    If (tmpLow < = tmpHi) Then
      tmpSwap = vArray(tmpLow)
      vArray(tmpLow) = vArray(tmpHi)
      vArray(tmpHi) = tmpSwap
      tmpLow = tmpLow + 1
      tmpHi = tmpHi - 1
    End If

  Wend

  If (inLow < tmpHi) Then QuickSort vArray, inLow, tmpHi
  If (tmpLow < inHi) Then QuickSort vArray, tmpLow, inHi

End Sub
```

Pozn. Existují i dvourozměrné (2D) varianty pro Quick Sort, kdy můžete třídit celé záznamy.

Tip

Až donedávna jsem neměl tušení, že je možné s pomocí funkce CreateObject využívat některé objekty a struktury z .NET Frameworku pod VBA. Pokud tedy nechcete na pole nasadit Quick Sort, můžete si dopomoci jinak.

```
1   Sub SerazeniPoleNET()
2
3       Dim myList As Object
4
5       'vytvoření pole (ArrayList)
6       Set myList = CreateObject("System.Collections.ArrayList")
7
8       'přidání položek do pole
9       myList.Add ("slon")
10      myList.Add ("kočka")
11      myList.Add ("prase")
12      myList.Add ("kůň")
13      myList.Add ("antilopa")
14
15      'setřídění
16      myList.Sort
17
18  End Sub
```

Přirozené řazení

V rámci řazení převažuje u textových řetězců postup znak po znaku. V tomto duchu je ve vzestupné posloupnosti A100 < A11, což zpravidla nevyhovuje našim potřebám. Chceme-li, aby A11 < A100, C2 < C10 atd., tak mluvíme o tzv. "přirozeném" řazení. Excel toto nenabízí a musíme si pomoci kódem VBA. Teoreticky postupujeme tak, že číselnou část řetězce dorovnáváme na stejný počet číslic (doplňujeme úvodní nulu), tj. A11 bude A011, C2 pak C02. Jinou cestou je užití API funkce StrCmpLogicalW z knihovny shlwapi.dll.

```
'-1 ... lpString1 < lpString2
' 0 ... lpString1 = lpString2
' 1 ... lpString1 > lpString2

Public Declare Function StrCmpLogicalW Lib "shlwapi" (ByVal lpString1 As Long, _
   ByVal lpString2 As Long) As Long

Sub TestPrirozeneTrideniAPI()

   'přirozené třídění
   'StrPtr nutné (jedná se o Unicode funkci)
   Debug.Print StrCmpLogicalW(StrPtr("a"), StrPtr("A"))    ' 0
   Debug.Print StrCmpLogicalW(StrPtr("a"), StrPtr("á"))    '-1
   Debug.Print StrCmpLogicalW(StrPtr("14"), StrPtr("100")) '-1
   Debug.Print StrCmpLogicalW(StrPtr("a1"), StrPtr("a2"))  '-1
   Debug.Print StrCmpLogicalW(StrPtr("a10"), StrPtr("a2")) ' 1
   Debug.Print StrCmpLogicalW(StrPtr("c"), StrPtr("č"))    '-1
   Debug.Print StrCmpLogicalW(StrPtr("č"), StrPtr("Č"))    ' 0
   Debug.Print StrCmpLogicalW(StrPtr("c"), StrPtr("ch"))   '-1
   Debug.Print StrCmpLogicalW(StrPtr("ci"), StrPtr("ch"))  '-1

End Sub
```

Dovolím si ještě jednu ukázku API funkce – CompareStringW z knihovny kernel32.dll.

```
'Michael S. Kaplan
'http://www.siao2.com/2005/10/28/486019.aspx

Public Enum CmpFlags
   NORM_IGNORECASE = &H1&
   NORM_IGNORENONSPACE = &H2&
   NORM_IGNORESYMBOLS = &H4&
   NORM_IGNOREKANATYPE = &H10000
   NORM_IGNOREWIDTH = &H20000
   LINGUISTIC_IGNORECASE = &H10&
   LINGUISTIC_IGNOREDIACRITIC = &H20&
   SORT_DIGITSASNUMBERS = &H8& 'Windows 7
   SORT_STRINGSORT = &H1000&
End Enum

Public Enum CSTR_
   LESS_THAN = 1      'string 1 less than string 2
   EQUAL = 2          'string 1 equal to string 2
   GREATER_THAN = 3   'string 1 greater than string 2
End Enum

Public Declare Function CompareStringW Lib "kernel32" (ByVal Locale As Long, _
   ByVal dwCmpFlags As Long, ByVal lpString1 As Long, ByVal cchCount1 As Long, _
   ByVal lpString1 As Long, ByVal cchCount1 As Long) As Long

Public Function CompareString(ByVal lcid As Long, ByVal flags As CmpFlags, _
   ByVal st1 As String, ByVal st2 As String) As CSTR_
   CompareString = CompareStringW(lcid, flags, StrPtr(st1), Len(st1), _
      StrPtr(st2), Len(st2))
End Function

Sub TestCompareString()

   Debug.Print CompareString(1029, NORM_IGNORECASE, "a", "A")        '2
   Debug.Print CompareString(1029, NORM_IGNORECASE, "a", "á")        '1
   Debug.Print CompareString(1029, NORM_IGNORECASE, "14", "100")     '3
   Debug.Print CompareString(1029, NORM_IGNORECASE, "c", "č")        '1
   Debug.Print CompareString(1029, NORM_IGNORECASE, "č", "Č")        '2
   Debug.Print CompareString(1029, NORM_IGNORECASE, "c", "ch")       '1
   Debug.Print CompareString(1029, NORM_IGNORECASE, "ci", "ch")      '1

   Debug.Print CompareString(1029, SORT_STRINGSORT, "a", "A")        '1
   Debug.Print CompareString(1029, SORT_STRINGSORT, "a", "á")        '1
   Debug.Print CompareString(1029, SORT_STRINGSORT, "14", "100")     '3
   Debug.Print CompareString(1029, SORT_STRINGSORT, "c", "č")        '1
   Debug.Print CompareString(1029, SORT_STRINGSORT, "č", "Č")        '1
   Debug.Print CompareString(1029, SORT_STRINGSORT, "c", "ch")       '1
   Debug.Print CompareString(1029, SORT_STRINGSORT, "ci", "ch")      '1

   'přirozené třídění
   Debug.Print CompareString(1029, SORT_DIGITSASNUMBERS, "14", "100")  '1
   Debug.Print CompareString(1029, SORT_DIGITSASNUMBERS, "a1", "a2")   '1
   Debug.Print CompareString(1029, SORT_DIGITSASNUMBERS, "a10", "a2")  '3

End Sub
```

Téma rozhodně není vyčerpáno. Museli bychom se zabývat vyhledávacími funkcemi SVYHLEDAT a POZVYHLEDAT, zobecnit úvahu řazení ve vztahu k operačnímu systému a jeho lokalizaci, pobavit se o tématu porovnávání v databázích, kódových stránkách, práci s textem v operační paměti, vyzkoušet chování Excelu ve webovém prohlížeči a také Office 365. A nakonec bychom vinili z našich útrap Cyrila s Metodějem, Husa, Komenského a další osobnosti naší historie a jazyka. Proto je na čase říci dost, končíme.

Ke stažení
excel-razeni.zip