

Zpracování dat

95 Úvod

V této kapitole naleznete řešení, které vám ukáže, jak můžete použít VBA pro zpracování dat. Dozvíte se, jak ukládat odkazy, jak porovnávat obsahy listů a jak se dají získat vzorce. Budeme se rovněž zabývat automatickým filtrováním, tříděním a kontingenčními tabulkami.

96 Odstranění duplicitních záznamů v Excelu 2007

Tabulkový procesor Excel se v praxi často používá i jako nástroj pro práci s databázemi. Pro tyto účely se však hodí pouze v tom případě, že se smíříte s řadou omezení. Jde o to, že v Excelu chybí takové základní funkce, které jsou k dispozici prakticky v každém programu pro práci s databázemi, a pokud je chcete použít i v Excelu, musíte je složitě dodělat.

Excel jako program pro práci s databázemi omezuje také fakt, že v databázi musí mít každá buňka v jednom sloupci naprosto stejný formát. Ve skutečné databázi se tohoto dosáhne velmi jednoduše nastavením formátu pole, zatímco v Excelu můžete mít v jednom sloupci v jedné buňce údaj ve formátu text a v jiné buňce téhož sloupce údaj ve formátu datum. I když pak obsah buňky na první pohled vypadá stejně, jejich obsah nesouhlasí.

Jednou z nejčastěji používaných databázových funkcí v každé databázi je mimo jiné ochrana před vznikem duplicitních záznamů, pro kterou se v programech pro práci s databázemi používají různé mechanismy. Excel naproti tomu vůbec nezajímá, zda se v některých buňkách nachází stejné údaje, a také tedy na tuto skutečnost vůbec neupozorňuje.

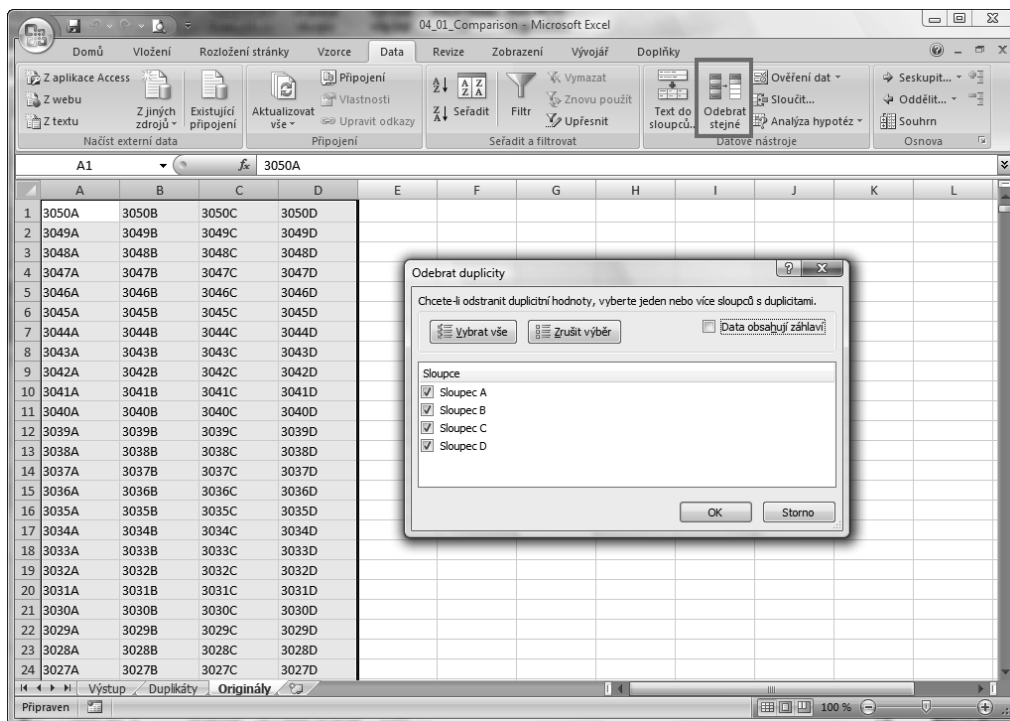
Pro odstranění nezáměrně vložených duplicitních záznamů by se ale v Excelu dal použít speciální filtr – záznamy je možné odfiltrovat jinam a přitom použít položku Bez duplicitních záznamů.

Excel 2007 obsahuje jako novinku vlastní funkci (záložka DATA, sekce DATOVÉ NÁSTROJE a tlačítko ODEBRAT STEJNÉ), pomocí níž se odstranění duplicitních záznamů provádí daleko pohodlněji (viz obrázek 80).

Chcete-li tu stejnou operaci provést v Excelu 2007 pomocí VBA, vyberte příslušnou oblast, z níž chcete odstranit duplicitní položky, a spusťte následující proceduru.

```
'=====
' Na CD          Data\Priklady\04_Zpracovani_dat\
' Název souboru  04_01_Comparison.xls
' List          Duplikáty
' Modul         md1_04_02_RemoveDuplicates
'=====
Public Sub RemoveMyDuplicates()
    Selection.CurrentRegion.RemoveDuplicates Header:=xlNo
End Sub
```

Pokud obsahuje vybraná oblast záhlaví, pak použijte namísto `Header:=xlNo` výraz `Header:=xlYes`.



Obrázek 80: Odstranění duplicitních položek

97 Odstranění duplicitních záznamů ve starších verzích Excelu

Ve starších verzích Excelu není pro odstranění duplicitních záznamů kromě speciálního filtru žádný jednodušší nástroj. Speciální filtr při operacích tohoto druhu ukládá výsledek filtrování na jiné místo, což ve většině případů není žádoucí.

V tomto řešení budeme duplicitní záznamy odstraňovat z označené oblasti přímo. Testovaná oblast se předá jako argument funkci `KillDoubleRecords`. Tato funkce pak jako výstup vrátí počet skutečně odstraněných záznamů.

Aby se při vyhledávání duplicitních položek nemusel každý řádek porovnávat s ostatními, což by bylo u rozsáhlejších databází časově velmi náročné, vydáme se poněkud jinou cestou – použijeme kolekci (`Collection`).

Prvky kolekce mají jeden jednoznačný klíč, který se sám nastaví při přidání prvku do kolekce. Tento klíč je víceméně libovolný řetězec, který se ale smí vyskytovat v kolekci pouze jednou. Pokud bychom se pokusili přidat do kolekce prvek, který by použil již existující klíč, pak by se vyvolala chyba číslo 457.

Každý záznam neboli obsah všech příslušných buněk v řádku se pro tento účel převede na řetězec, poté se spojí a výsledek se použije jako klíč. Pokud se při přidávání nového prvku do kolekce objeví chyba 457, pak tento záznam již v kolekci existuje a může se smazat.

Při použití této metody je ale jeden problém. Budeme-li totiž pomocí počítadla (`For i=1 To 6 ... Next`) procházet postupně každý řádek oblasti a odstraníme-li následně aktuální řádek, pak pokud nijak neupravíme hodnotu počítadla, se řádek přeskočí. Přesněji řečeno se jedná o ten řádek, který po odstranění řádku převzal jeho místo. Abychom se tomuto nežádoucímu jevu vyhnuli, procházíme celou oblast odzadu dopředu. Odstranění aktuálního řádku pak totiž nebude vyžadovat žádné úpravy počítadla.

Drobnou nevýhodou kolekce je skutečnost, že se v klíči nerozlišují malá a velká písmena – nicméně ve většině případů nám to určitě vadit nebude.

```
'=====
' Na CD           Data\Priklady\04_Zpracovani_dat\
' Název souboru   04_01_Comparison.xls
' List           Duplikáty
' Modul          mdl_04_03_KillDoubleRecords
'=====
```

Option Explicit

```
Public Sub TestKillDoubleRecords()
```

```
    MsgBox KillDoubleRecords( _
        Worksheets("List1").Range("A1:D10000") _
    ), , "Odstraněné záznamy"
```

```
End Sub
```

```
Public Function KillDoubleRecords(Range1 As Range) As Long
```

```
    Dim rngField      As Range
    Dim lngCount       As Long
    Dim strKey         As String
    Dim myCol          As New Collection
    Dim i              As Long
```

```
On Error Resume Next
```

```
For i = Range1.Rows.Count To 1 Step -1
```

```
    ' Procházení všech řad postupně
    ' zesponu nahoru
```

```
    strKey = ""
```

```
For Each rngField In Range1.Rows(i).Cells
```

```
    ' Postupně se prochází všechny buňky v této řadě
```

```
    With rngField
```

```
        If .Value <> "" Then
            ' Vytvoření jednoznačného klíče
            ' z informací uvedených v řádku
            strKey = strKey & CStr(.Value)
        End If
    End With
```

```
Next rngField
```

```
If strKey <> "" Then
```

```
    Err.Clear ' Odstranění chyby paměti
```

```
    ' Přidání prvku do kolekce
    myCol.Add strKey, "X" & strKey
```

```
If Err.Number = 457 Then
```

```

' Pokud se objeví chyba 457, pak již záznam existuje.
' Vzhledem k tomu, že porovnávání funguje pouze jako LIKE
' (nerozlišují se malá a velká písmena),
' provádí se kontrola ještě jednou.
' Nyní se použije položka Compare fungující na úrovni modulu
If strKey = myCol("X" & strKey) Then
    ' Odstranění aktuálního řádku oblasti
    Range1.Rows(i).Delete Shift:=xlUp
    ' Zjištění počtu odstraněných řádků
    lngCount = lngCount + 1
End If
ElseIf Err.Number <> 0 Then
    MsgBox Err.Description
End If
End If

Next i
KillDoubleRecords = lngCount ' Vrácení počtu odstraněných řádků
End Function

```

98 Porovnávání listů (nalezené odlišnosti se umístí do zvláštního listu)

Pokud s nějakým souborem dat pracuje více osob na různých počítačích a pokud je zdroj dat, v tomto případě sešit Excelu, na každém z těchto počítačů jako kopie originálního sešitu, pak po úpravě tyto sešity z různých počítačů obsahují různé záznamy. Pokud se následně listy z těchto sešitů mají použít v jedné databázi, pak je nutno rozhodnout, které záznamy odstranit a které ponechat.

Abyste se mohli zodpovědně rozhodnout, které záznamy ponechat a které odstranit, musíte nejprve zjistit, které záznamy se vůbec změnily. Možná vás napadne provést jednoduché porovnávání 1:1 každého řádku listu s řádky na listech jiných sešitů. Nicméně tento způsob vám nedoporučujeme, protože stačí jeden odstraněný nebo vložený řádek a už se při porovnávání doslova ztratíte.

V tomto řešení se bude umístění záznamu s odlišným obsahem vypisovat do zvláštního listu (VÝSTUP), přičemž odlišná umístění (v různých řádcích) v tomto případě nebudou hrát roli. Rozpoznají se tak bez problému všechny stejné záznamy, a to ať se vyskytují v jakékoliv buňce.

Adresy buněk ve výstupní tabulce navíc budou mít podobu hypertextových odkazů, takže po klepnutí na ně se pohodlně přemístíte na řádek, v němž se původně vyskytovaly.

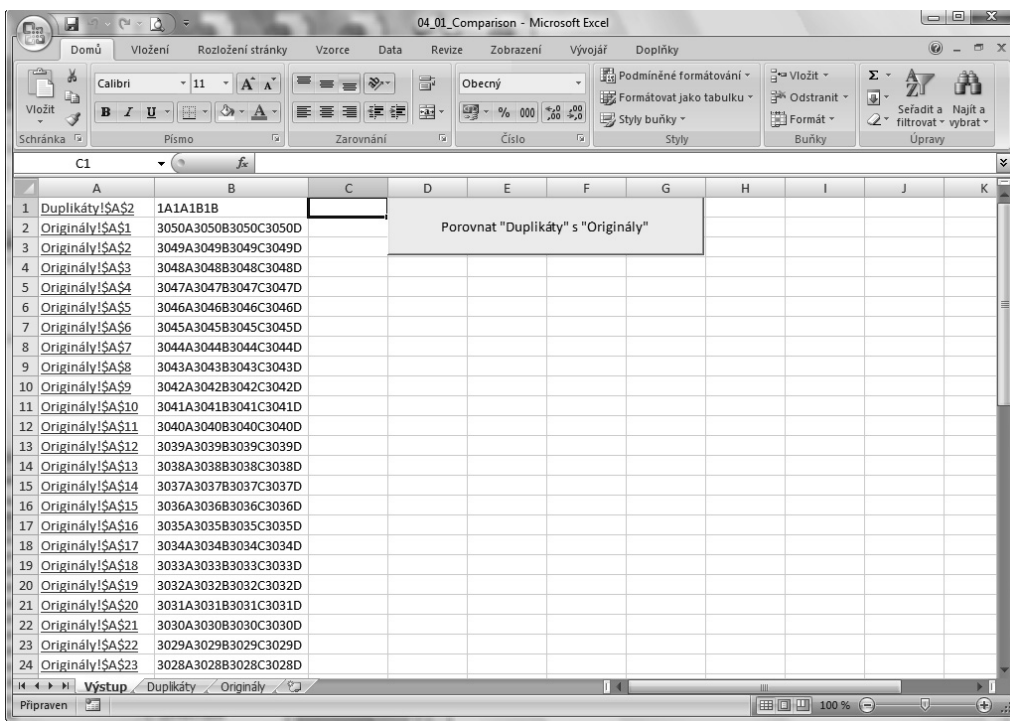
V proceduře `TestCompare` se volá funkce `FindUnique`, která jako výsledek vrací kolekci s jednou se vyskytujícími záznamy. Jako argument funkce se předávají porovnávané oblasti a list, kam se mají vypsat výsledky.

Poté se odstraní obsahy buněk v listu pro výstup a postupně se prochází každý prvek vrácené kolekce. Každý prvek kolekce obsahuje pole se dvěma prvky, které opět obsahují adresu buňky a záznam, a to v podobě řetězce znaků. Obojí se nakonec zobrazí v listu určeném pro výstup, který navíc použije adresy k vytvoření hypertextových odkazů.

```

' =====
' Na CD          Data\Priklady\04_Zpracovani_dat\
' Název souboru  04_01_Comparison.xls
' List          Výstup
' Modul         mdl_04_04_FindUni

```



Obrázek 81: Zobrazení jednoznačných záznamů

```

' =====
Public Sub TestCompare()
    Dim rngSource1 As Range
    Dim rngSource2 As Range
    Dim wsDestSheet As Worksheet
    Dim colResult As Collection
    Dim varItem As Variant
    Dim i As Long

    ' Oblast 1, která se bude porovnávat s oblastí 2
    Set rngSource1 = Worksheets("Duplikáty").Range("A1:D10000")

    ' Oblast 2, která se bude porovnávat s oblastí 1
    Set rngSource2 = Worksheets("Originály").Range("A1:D10000")

    ' Cílová oblast (Výstup)
    Set wsDestSheet = Worksheets("Výstup")

    ' Zjištění odlišných záznamů
    Set colResult = FindUnique(rngSource1, rngSource2)

    ' Výstup do zadaného listu
    With wsDestSheet

        ' Vyčištění listu pro výstup
        .Cells.Clear

        For Each varItem In colResult

```

```

        i = i + 1
        .Cells(i, 1) = varItem(1) ' Výstup adresy
        .Cells(i, 2) = varItem(2) ' Výstup obsahu záznamu

        ' Vytvoření hypertextového odkazu na záznam
        .Hyperlinks.Add _
            Anchor:=.Cells(i, 1), _
            Address:="", _
            SubAddress:=varItem(1), _
            TextToDisplay:=varItem(1)
    Next
End With
End Sub

```

Ve funkci `FindUnique` používáme kolekci k poněkud jinému účelu, než tomu bylo v předcházejícím řešení, kde jsme ji použili k porovnání záznamů. V tomto případě se při výskytu chyby 457 nebude odstraňovat žádný záznam, ale z kolekce se odstraní prvek s příslušným klíčem. Na konci tedy v kolekci zůstávají pouze jedinou se vyskytující záznamy.

Důležité je podotknout, že se duplicitní záznamy ve stejné tabulce odstraní předem (viz řešení 96 a 97). Pokud by se tak nestalo, pak se prvek z kolekce odstraní při druhém výskytu v listu, a tak se stejný záznam v jiné tabulce rozpozná jako jedinečný.

Při přidání prvku do kolekce se obsah prvku uloží jako pole obsahující adresu a obsah záznamu.

```

' =====
' Na CD          Data\Priklady\04_Zpracovani_dat\
' Název souboru  04_01_Comparison.xls
' List          Výstup
' Modul         mdl_04_04_FindUni
' =====
Private Function FindUnique(Range1 As Range, Range2 As Range) As Collection
    Dim rngRow      As Range
    Dim rngField    As Range
    Dim rngSource    As Range
    Dim strKey       As String
    Dim strAddress   As String
    Dim myCol        As New Collection
    Dim x(1 To 2)    As String
    Dim i            As Long

    On Error Resume Next

    For i = 1 To 2 ' Procházení obou oblastí

        If i = 1 Then
            Set rngSource = Range1
        Else
            Set rngSource = Range2
        End If

        For Each rngRow In rngSource.Rows
            ' Všechny řádky oblasti za sebou

            strKey = ""
            strAddress = ""

            With rngRow.Cells(1)
                ' Získání adresy první buňky řádku

```

```

        strAddress = .Worksheet.Name & "!" & .Address
    End With

    For Each rngField In rngRow.Cells
        ' Postupné procházení všech buněk řádku
        With rngField
            If .Value <> "" Then
                ' Vytvoření jednoznačného klíče ze všech údajů
                ' na řádku
                strKey = strKey & CStr(.Value)
            End If
        End With
    Next rngField

    If strKey <> "" Then
        Err.Clear ' Odstranění chyby paměti

        x(1) = strAddress
        x(2) = strKey
        ' Přidání prvku do kolekce
        myCol.Add x, "X" & strKey
        If Err.Number = 457 Then
            ' Pokud se objeví chyba 457, pak již záznam existuje.
            ' Vzhledem k tomu, že porovnávání funguje pouze jako LIKE
            ' (nerozlišují se malá a velká písmena),
            ' provádí se kontrola ještě jednou.
            ' Nyní se použije položka Compare fungující na úrovni modulu
            If strKey = myCol("X" & strKey)(2) Then
                myCol.Remove "X" & strKey
            End If
        ElseIf Err.Number <> 0 Then
            MsgBox Err.Description
        End If
    End If

    Next rngRow
Next i

Set FindUnique = myCol

End Function

```

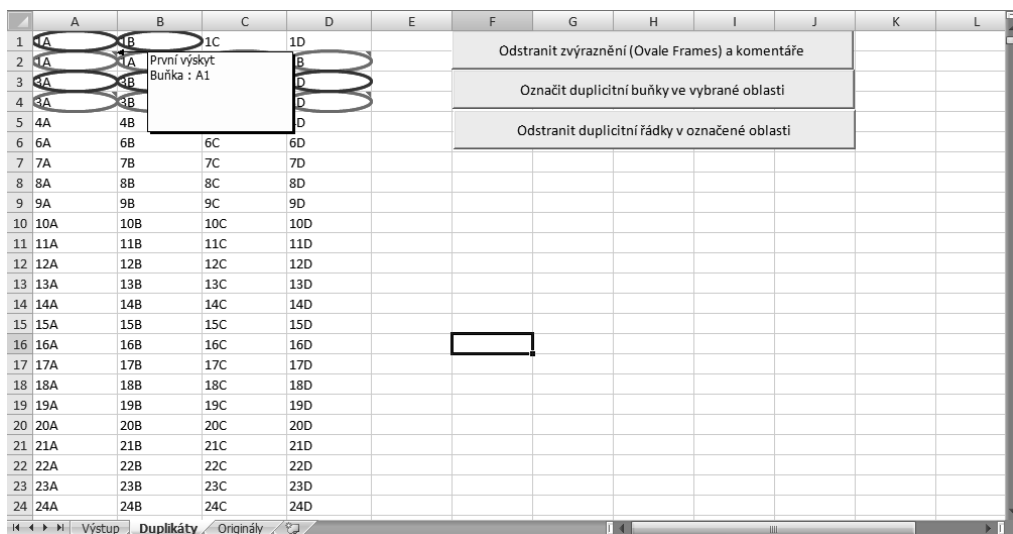
99 Označení buněk se stejným obsahem

V tomto řešení vám ukážeme, jak ve vybrané oblasti najít buňky se stejným obsahem a jak je zvýraznit pomocí oválu. Současně se do těchto buněk vloží komentář, který bude obsahovat údaj o adrese buňky, kde se tento obsah vyskytl poprvé (viz obrázek 82).

V proceduře `MarcDoubleCells` se nejprve vyhledá případný výskyt všech tvarů (Shapes), konkrétně se hledají objekty typu `msoShapeOval`. Všechny případně nalezené objekty tohoto typu se následně odstraní. Poté se z listu odstraní všechny komentáře, v jejichž textu tvoří prvních 12 znaků řetězec „První výskyt“. K tomu účelu se prochází všechny buňky, které obsahují komentáře. Kód

```
Selection.SpecialCells(xlCellTypeComments)
```

vrací rozsah `Range` vybrané oblasti, přičemž se jedná o oblast, která obsahuje pouze buňky určitého typu, v tomto případě pouze buňky s komentářem.



Obrázek 82: Označení buněk se stejným obsahem

Poté se taktéž odstraní případně se vyskytující ohraničení a komentáře z předchozího vyhledávání duplicitních položek. K tomu účelu se postupně prochází všechny buňky vybrané oblasti. Obsah každé buňky se proto převádí do formátu řetězce a výsledek se použije jako klíč pro vložení do kolekce. Jako obsah prvku se použije adresa aktuální buňky. Pokud se při vkládání vyskytne chyba 457, pak již tento klíč v kolekci existuje. V tomto případě se obsah buňky označí, a to červeným oválem, přičemž se upraví jeho velikost a umístí se dále tak, aby přesně odpovídal velikosti buňky.

Poté se zjišťuje, zda se před adresou uloženou v tomto prvku náhodou nenachází znak -. Pokud ne, pak buňka, v níž se tato hodnota poprvé objevila, ještě nebyla označena. Nyní se pro toto v kolekci uložené umístění buňky použije modrý ovál.

Aby se při dalším výskytu první buňka znovu neoznačovala, odstraní se z kolekce a přidá se nová se stejným klíčem. U kolekci se bohužel hodnota nedá dodatečně měnit. Před vlastní adresu se však pro odlišení jakožto obsahu prvku tentokrát použije znak -. Pokud se při kontrole adresy zjistí na prvním místě znak -, pak se musí při určení skutečné adresy z řetězce odstranit. To provede řádek kódu

```
strAddress = Mid(strAddress, 2)
```

Jinými slovy proměnná `strAddress` pak bude obsahovat adresu buňky, kde se tato hodnota objevila poprvé. Do aktuální buňky se nakonec přidá komentář obsahující adresu buňky, kde se poprvé vyskytl její obsah.

```
'=====
' Na CD          Data\Priklady\04_Zpracovani_dat\
' Název souboru  04_01_Comparison.xls
' List           Duplikáty
' Modul          mdl_04_05_Marc
'=====

Public Sub MarcDoubleCells()
    Dim colX      As New Collection
    Dim rngX      As Range
```

```

Dim wsSheet As Worksheet
Dim objShape As Shape
Dim strAddress As String
On Error Resume Next

' Vytvoření objektové proměnné pro vybranou oblast
Set wsSheet = Selection.Cells(1).Worksheet

For Each objShape In wsSheet.Shapes
    ' Procházení všech tvarů na listu sešitu
    If objShape.AutoShapeType = msoShapeOval Then
        ' Odstranění tvarů typu msoShapeOval
        objShape.Delete
    End If
Next objShape

For Each rngX In Selection.SpecialCells(xlCellTypeComments)
    ' Procházení všech buněk s komentářem
    If Left(rngX.Comment.Text, 12) = "První výskyt" Then
        ' Odstranění komentáře
        rngX.Comment.Delete
    End If
Next

For Each rngX In Selection

    If rngX.Value <> "" Then

        Err.Clear ' Odstranění chyby paměti

        ' Otestování, zda se obsah buňky může
        ' vložit do kolekce jako klíč
        colX.Add rngX.Address(0, 0), "X" & rngX.Value

        If Err.Number = 457 Then
            ' Pokud se objeví chyba 457, pak již záznam existuje.
            ' Vzhledem k tomu, že porovnávání funguje pouze jako LIKE,
            ' nerozlišují se malá a velká písmena.

            With wsSheet

                ' Přidání oválu
                With .Shapes.AddShape(msoShapeOval, _
                    rngX.Left, _
                    rngX.Top, _
                    rngX.Width, _
                    rngX.Height)
                    .Fill.Visible = msoFalse
                    .Line.ForeColor.SchemeColor = 10
                End With

                ' Načtení adresy
                strAddress = colX("X" & rngX.Value)

                If Left(strAddress, 1) <> "-" Then
                    ' Přidání oválu u buňky s prvním výskytem

                    With .Shapes.AddShape(msoShapeOval, _
                        .Range(strAddress).Left, _
                        .Range(strAddress).Top, _

```

```

        .Range(strAddress).Width, _
        .Range(strAddress).Height)
        .Fill.Visible = msoFalse
        .Line.ForeColor.SchemeColor = 12
    End With

    ' Odstranění položky s adresou
    colX.Remove "X" & rngX.Value

    ' Znak "-" před adresou, aby se při dalším
    ' výskytu dalo rozpoznat, že byl již tvar vložen
    ' do první buňky.
    colX.Add "-" & strAddress, "X" & rngX.Value

Else

    ' Odstranění znaku "-"
    strAddress = Mid(strAddress, 2)

End If

    ' Komentář s upozorněním na první výskyt
    rngX.AddComment "První výskyt" & vbLf & _
        "Buňka : " & strAddress

End With ' wsSheet

End If ' Err.Number = 457

End If ' rngX.Value <> ""

Next rngX

End Sub

```

100 Vypsání vzorců do listu sešitu

V listu sešitu se dají vzorce zobrazit a opět skrýt klávesovou zkratkou **Ctrl+9**. Stejného výsledku dosáhnete, když na záložce **VZORCE** klepnete v sekci **ZÁVISLOSTI VZORCŮ** na tlačítko **ZOBRAZIT VZORCE** (viz obrázek 83).

Když list obsahuje příliš mnoho vzorců, je více či méně nepřehledný. Další přítěžující okolností je skutečnost, že nelze současně zobrazit vzorce a hodnoty. Pokud byste navíc potřebovali ještě zobrazit přímé předchůdce, pak kvůli zobrazení šipce toho uvidíme ještě méně.

Proto je velmi šikovné, když si vzorce v takovém listu zobrazíme do zvláštního listu. V tomto řešení si představíme způsob, jak načíst vzorce obsažené v listu a jak zjistit přímé předchůdce.

Výsledky pak zobrazíme do listu **ZOBRAZENÍ VZORCŮ** (viz obrázek 84).

Procedura `FindFormulas` slouží k postupnému procházení buněk na listu, které obsahují vzorce. Pro zjištění buněk představujících přímé předchůdce se používá vlastní funkce `GetDirectPrecedents`. Výsledek se zapisuje do listu **ZOBRAZENÍ VZORCŮ**.

Metoda `SpecialCells` vrací společně s konstantou `xlCellTypeFormulas` předanou jako první parametr oblast, která obsahuje pouze buňky se vzorci, a tato oblast se pak přiřadí do proměnné `rn-`

B29						
	A	B	C	D	E	F
	Zobrazení vzorců z listu List1					
1	Adresa buňky	Text vzorce	Hodnota vzorce	Buňky předchůdců		
2	List1!A1	=List2!A1&" "&A13&A14	Leden ab	List2!A1	List1!A13	List1!A14
3	List1!A2	=List2!A2*1	#HODNOTA!	List2!A2		
4	List1!A3	=List2!A3	Březen	List2!A3		
5	List1!A4	=List2!A4	Duben	List2!A4		
6	List1!A5	=List2!A5	Květen	List2!A5		
7	List1!A6	=List2!A6	Červen	List2!A6		
8	List1!A7	=List2!A7	2	List2!A7		
9	List1!A8	=List2!A8	Srpen	List2!A8		
10	List1!A9	=List2!A9	Září	List2!A9		
11	List1!A10	=List2!A10	Říjen	List2!A10		
12	List1!A11	=List2!A11	Listopad	List2!A11		
13	List1!A12	=List2!A12	Prosinec	List2!A12		
14						
15						
16						
17						

Obrázek 83: Zobrazení/skrytí vzorců

G16						
	A	B	C	D	E	F
	Zobrazení vzorců z listu List1					
1	Adresa buňky	Text vzorce	Hodnota vzorce	Buňky předchůdců		
2	List1!A1	=List2!A1&" " &A13&A14	Leden ab	List2!A1	List1!A13	List1!A14
3	List1!A2	=List2!A2*1	#HODNOTA!	List2!A2		
4	List1!A3	=List2!A3	Březen	List2!A3		
5	List1!A4	=List2!A4	Duben	List2!A4		
6	List1!A5	=List2!A5	Květen	List2!A5		
7	List1!A6	=List2!A6	Červen	List2!A6		
8	List1!A7	=List2!A7		2 List2!A7		
9	List1!A8	=List2!A8	Srpen	List2!A8		
10	List1!A9	=List2!A9	Září	List2!A9		
11	List1!A10	=List2!A10	Říjen	List2!A10		
12	List1!A11	=List2!A11	Listopad	List2!A11		
13	List1!A12	=List2!A12	Prosinec	List2!A12		
14						

Obrázek 84: Zobrazení vzorců na zvláštním listu

gFormulas. Poté se vymaže obsah cílového listu a vloží se do něj nadpisy sloupců. Pokud zpracováváný list žádné vzorce neobsahuje, pak se procedura na tomto místě ukončí.

V dalším kroku se budou v cyklu For Each ... Next postupně zpracovávat všechny buňky oblasti. Adresa aktuální buňky se vzorcem, která se skrývá v proměnné rngCell jako Range, se zapíše včetně hypertextového odkazu do sloupce A. Do sloupce B se pak umístí vzorec v podobě textu a do sloupce C pak přijde hodnota vypočtená pomocí tohoto vzorce.

```

' =====
' Na CD          Data\Priklady\04_Zpracovani_dat\
' Název souboru  04_02_GetFormulas.xlsm
' List           Zobrazení vzorců
' Modul         mdl_04_06_Formulas
' =====

```

```

Public Sub FindFormulars()
    Dim rngFormulars As Range
    Dim rngCell As Range
    Dim varErg As Variant
    Dim strHyperlink As String
    Dim i As Long
    Dim k As Long

    On Error Resume Next

    Set rngFormulars = Worksheets("List1").Cells.SpecialCells( _
        xlCellTypeFormulas)

    With Worksheets("Zobrazení vzorců") ' List pro výstup

        ' Zobrazení nadpisů buněk
        .Cells.Clear
        i = 1
        .Rows(i).Font.Bold = True
        .Cells(i, 1).Value = "Adresa buňky"
        .Cells(i, 2).Value = "Text vzorce"
        .Cells(i, 3).Value = "Hodnota vzorce"
        .Cells(i, 4).Value = "Buňky předchůdců"

        ' Pokud nejsou žádné vzorce, pak konec
        If rngFormulars.Count = 0 Then Exit Sub

        For Each rngCell In rngFormulars
            ' Procházení všech buněk se vzorci

            i = i + 1
            strHyperlink = rngCell.Worksheet.Name & "!" & _
                rngCell.Address(0, 0)

            ' Vložení adresy obsahující vzorec
            .Cells(i, 1).Value = strHyperlink

            ' Vytvoření hypertextového odkazu ukazujícího na buňku se vzorcem
            .Hyperlinks.Add _
                Anchor:=.Cells(i, 1), _
                Address:="", _
                SubAddress:=strHyperlink

            ' Zobrazení vzorce
            .Cells(i, 2).Value = "" & rngCell.FormulaLocal

            ' Zobrazení hodnoty vzorce
            .Cells(i, 3).Value = rngCell.Value

            ' Zjištění předchůdců
            varErg = GetDirectPrecedents(rngCell)

            If IsArray(varErg) Then
                ' Předchůdci existují

                For k = LBound(varErg) To UBound(varErg)
                    ' Procházení všech předchůdců

                    ' Zjištění adres všech předchůdců
                    .Cells(i, k + 3).Value = varErg(k)
                
```

```

' Vytvoření hypertextových odkazů ukazujících na předchůdce
.Hyperlinks.Add _
    Anchor:=Cells(i, k + 3), _
    Address:="", _
    SubAddress:=varErg(k), _
    TextToDisplay:=varErg(k)

Next k

End If

Next rngCell

End With

End Sub

```

Funkce `GetDirectPrecedents` pak v podobě pole vrací adresy buněk předchůdců. Této funkci se jako argument předává aktuální buňka se vzorcem. Dojde k načtení prvků tohoto vráceného pole a k jeho vložení do sloupce D cílového listu. Kromě toho se navíc ještě jednotlivé položky rozšíří o hypertextový odkaz. Všechny hodnoty v řádku přitom patří ke vzorci, jehož adresa se vyskytuje ve sloupci A.

Uvnitř funkce `GetDirectPrecedents` se nejprve zjišťuje, zda vzorec neobsahuje uvozovky. Pokud uvozovky obsahuje, vyvstává možnost, že se přímá buňka s předchůdcem nachází na jiném listu. Bohužel vlastnost `DirectPrecedents` libovolné buňky adresu buňky přímého předchůdce nevrací. Tato vlastnost je funkcí pouze u stejného listu, a nikoliv u odkazů typu `REMOTE`.

Chcete-li tedy adresu předchůdce přesto nějakým způsobem zjistit, musíte vynaložit o něco více úsilí. Ze všeho nejdříve se musí všechny operátory v textu vzorce nahradit co možná nejednoznačnejším řetězcem. V našem případě použijeme řetězec obsahující pět znaků plus, za normálních okolností by se tato posloupnost znaků neměla ve vzorci nikde vyskytovat (kromě řetězců).

Poté se řetězec pomocí funkce `Split` převede na jednorozměrné pole, přičemž se jako `DELIMITER` (oddělovač = posloupnost znaků, která slouží pro identifikaci místa, kde se má provést rozdělení) použije právě řetězec pěti znaků plus. Prvek tohoto pole, který nebude obsahovat uvozovky a který bude obsahovat znak vykřičníku, se pak interpretuje jako odkaz na buňku a uloží se jako prvek do pole `varErg`.

Zmíněné dvě podmínky týkající se uvozovek a vykřičníku potřebujeme k tomu, abychom zabránili, že se vykřičník v nějakém řetězci chybně interpretuje jako odkaz na buňku.

Dále se použije na buňku vlastnost `DirectPrecedents`. Ta slouží k získání buněk přímých předchůdců na stejném listu. Adresy těchto buněk se spojí s názvem listu, přičemž vznikne plnohodnotný odkaz obsahující název listu, vykřičník a adresu. Tento odkaz se pak znovu uloží jako prvek pole `varErg`.

Nakonec se jako výsledek funkce vrátí pole `varErg`.

```

' =====
' Na CD          Data\Priklady\04_Zpracovani_dat\
' Název souboru  04_02_GetFormulas.xlsm
' List           Zobrazení vzorců
' Modul          mdl_04_06_Formulas
' =====

```

```

Function GetDirectPrecedents(rngFormula As Range) As Variant
    Dim varDummy      As Variant
    Dim avarReplace    As Variant
    Dim varErg         As Variant
    Dim strErg         As String
    Dim strWSName      As String
    Dim i              As Long
    Dim k              As Long
    Dim rngX           As Range
    Dim rngY           As Range

    On Error Resume Next

    With rngFormula

        strErg = .Formula
        strWSName = .Worksheet.Name

        If InStr(strErg, "!") Then
            ' Je možný i odkaz na buňku na jiném listu

            ' Nahrazení všech operátorů řetězcem "++++"
            avarReplace = Array("+", "-", "*", "/", "%", "&, _
                "<>", ">=", "<=", ">", "<", "=")
            For i = 0 To UBound(avarReplace)
                strErg = Replace(strErg, avarReplace(i), "++++")
            Next i

            ' Převod na pole, dělicím znakem (Delimiter) je
            ' řetězec "++++"
            varDummy = Split(strErg, "++++")

            ReDim varErg(1 To UBound(varDummy) + 1)

            ' Procházení všech prvků pole
            For i = 0 To UBound(varDummy)

                If InStr(varDummy(i), "") = 0 Then
                    ' V odkazech na buňky se
                    ' nenachází žádné uvozovky

                    If InStr(varDummy(i), "!") <> 0 Then
                        ' Odkaz na jiný list

                        k = k + 1
                        ' Uložení odkazu
                        varErg(k) = varDummy(i)

                    End If

                End If

            End For

            Next i

            ' Upravení velikosti výsledného pole
            ReDim Preserve varErg(1 To k)

        End If

    End With

    Err.Clear

```

```

' Pokus o přímé získání buněk předchůdců
' Nefunguje u odkazů na buňky na jiných listech
Set rngX = .DirectPrecedents ' .Precedents

If Err.Number = 0 Then
    ' Existuje alespoň jedna buňka předchůdce

    For Each rngY In rngX
        ' Postupné procházení buněk všech předchůdců

        If IsArray(varErg) Then
            i = UBound(varErg) + 1
        Else
            ReDim varErg(1 To 1)
            i = 1
        End If

        ' úprava velikosti cílového pole
        ReDim Preserve varErg(1 To i)

        ' Uložení adresy buňky předchůdce včetně názvu listu
        ' do pole
        varErg(i) = strWSName & "!" & rngY.Address(0, 0)

    Next rngY

End If

End With

' Výsledkem funkce je pole
GetDirectPrecedents = varErg

End Function

```

101 Podmíněné formátování

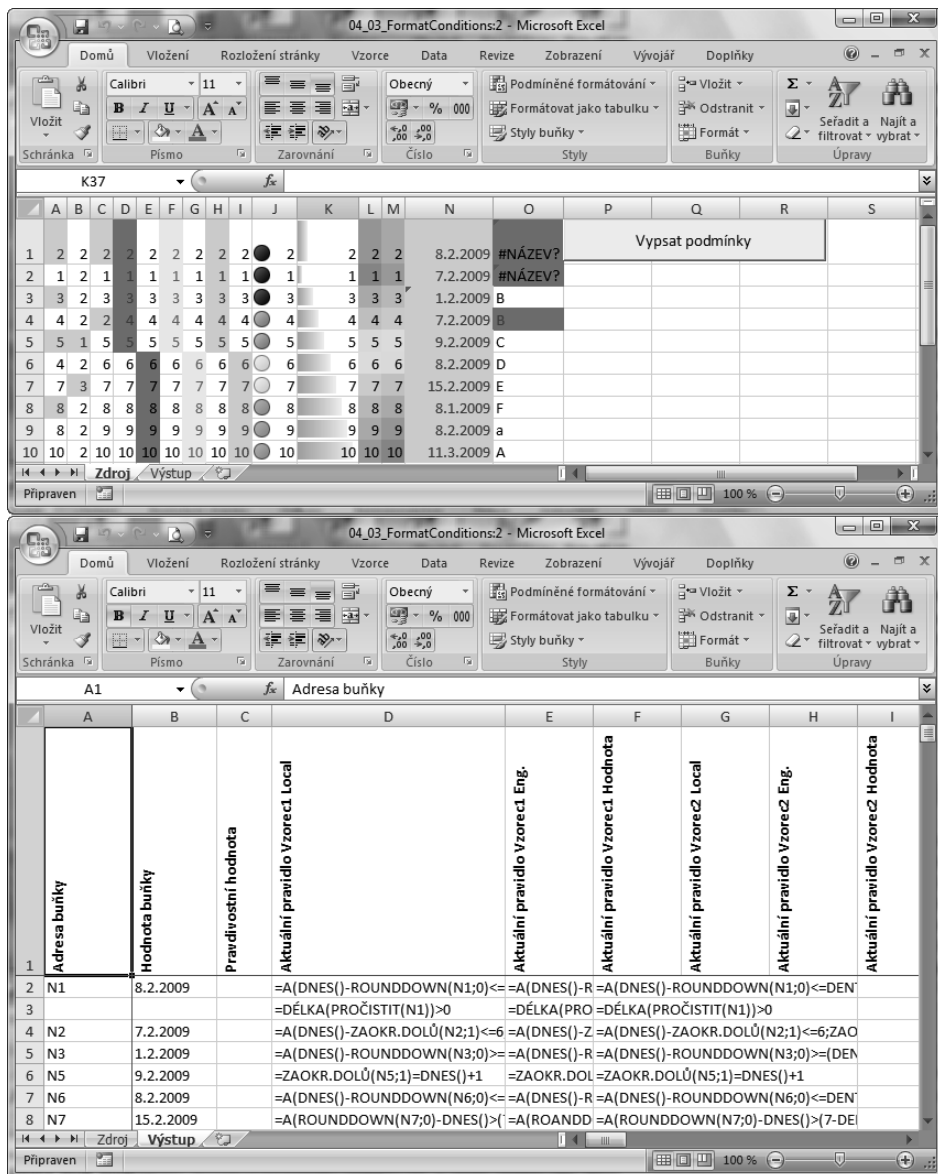
Vývojáři Excelu 2007 funkci podmíněného formátování od základů přepracovali. Kromě nových možností, jako je zobrazení různých hodnot v podobě barevné škály, ikoněk či sloupců, nyní můžeme pro buňky definovat libovolný počet pravidel.

Pro nezkušeného uživatele se zde otevírají zcela nové možnosti. Programátorům ve VBA se však situace nijak neusnadnila. Použití podmíněného formátování je stejně obtížné jako dříve i u tak zdánlivě banálních vlastností, jako je zjištění právě nastavené barvy pozadí buňky nebo spočítání množství buněk s formátováním podle skutečného pravidla.

V podstatě platí, že standardně nastavené vlastnosti formátování buňky lze přepsat vlastnostmi nastavenými podle splněného kritéria podmíněného formátování, které tak má větší prioritu. Pokud například načítáme pozadí buňky (`ActiveCell.Interior.Color`) pomocí splněného kritéria, pak je k dispozici pouze vlastnost formátování nastavená standardně.

Situaci ztěžuje i fakt, že neexistuje ani vlastnost objektu, která dokáže rozpoznat, zda je podmínka nastavená kritériem v daný okamžik splněna. Pokud však potřebujete tuto skutečnost zjistit, musíte zjistit stav pravdivosti u všech kritérií (pravidel), což za jistých okolností také znamená, že musíte vzorce spočítat pomocí VBA.

Toto řešení postupně načítá všechna nastavená pravidla v listu sešitu a výsledek zobrazuje ve zvláštním listu, přičemž zohledňuje ty nejdůležitější vlastnosti.



Obrázek 85: Zobrazení pravidel podmíněného formátování

Po zavolání procedury `PrintFormatConditions` se zpracují a do cílového listu zobrazí vlastnosti nastavených pravidel.

Nejprve se vytvoří oblast (Range), a to pomocí funkce `SpecialCells` a argumentu `xlCellTypeAllFormatConditions`. Tato oblast obsahuje pouze ty buňky ze zdrojového listu, které obsahují podmíněné formátování. Poté se v cílovém listu vytvoří a naformátuje řádek s nadpisy.

V dalším kroku se pomocí cyklu `For Each ... Next` prochází jednotlivé buňky. Při každém průchodu se načítají obecné informace o buňce, dále její obsah a počet nastavených pravidel (`FormatConditions.Count`). Tyto informace se pak dočasně ukládají do některé z vlastních proměnných (`varValue`, `lngCount`). Adresa a hodnota aktuální buňky se zapisuje přímo do cílového listu.

Objekt `FormatConditions` příslušné buňky obsahuje všechna nastavená pravidla, která se vyskytují jako prvky objektu typu `FormatCondition`. Poté se v cyklu `For Each ... Next` postupně každé jednotlivé pravidlo vrací jako objektová proměnná typu `objCondition`. Tento objekt nyní obsahuje podmínky a formátování aktuálního pravidla.

Vlastnost `AppliesTo` objektu `objConditions` obsahuje adresu, na kterou se pravidlo používá. Vlastní podmínky pak obsahují vlastnosti `Formula1` a `Formula2`. Kromě jednoznačných hodnot se zde mohou vyskytovat i odkazy nebo vzorce. Vlastnost `Formula2` Excel používá pouze tehdy, pokud je nutno použít dvě hodnoty, například u hodnot rozsahů `OD ... DO`.

Chcete-li obsah vzorce spočítat pomocí VBA, pak musíte použít funkci `Evaluate` – ta však bohužel neumožňuje používat názvy funkcí v české podobě. Proto je třeba funkce zadávat přeložené do angličtiny. Můžete k tomu účelu použít vlastnost `FormulaLocal` dané buňky a načíst vlastnost `Formula`.

Tento postup vám ale nedoporučujeme, protože za prvé je jeho zpracování poměrně časově náročné a také musíte najít nějakou buňku, kterou budete moci k tomuto účelu použít. Pokud tato buňka leží mimo používanou oblast, pak se používaná oblast zvětší a po jejím odstranění už ji nebudete moci obnovit.

Další nevýhodou je nepěkně vypadající blikání, které nastává, když použitá buňka leží v již viditelné oblasti. Kromě toho se může vyvolat přepočítání a spustit procedura pro obsluhu události. Jejím spuštěním můžete sice zabránit vypnutím zpracování událostí a automatického přepočítávání, daleko snazší ovšem je, když nemusíte měnit vůbec žádnou buňku.

V tomto řešení proto převádíme české názvy funkcí na anglické (pomocí funkce `TranslateFormulaLocal`), ty se však převedou jen zčásti. To ale nevadí, protože jde pouze o přeložení slov, takže zisk znalostí se tak drží na snesitelné úrovni.

Pokud se již zabýváme funkcí pro překlad, pak se při jejím používání nedivme, když se tu a tam vyskytnou zdánlivě chybně zapsaná anglická klíčová slova. Dochází k tomu, když se například používají funkce `ZAOKROUHLIT`, `ZAOKR.NAHORU` či `ZAOKR.DOLŮ` a zároveň klíčové slovo `A`. Pokud se v jednom vzorci nachází takových funkcí více, pak může snadno při překladu dojít k chybě.

```
' =====
' Na CD           Data\Priklady\04_Zpracovani_dat\
' Název souboru   04_03_FormatConditions.xlsm
' List           Zobrazení vzorců
' Modul          mdl_04_07_General
```

```

'=====
Public Function TranslateFormulaLocal( _
    ByVal strFormula As String _
    ) As String
strFormula = Replace(strFormula, _
    "RUNDEN", "ROOND", , , vbTextCompare)
strFormula = Replace(strFormula, _
    "UND", "AND", , , vbTextCompare)
strFormula = Replace(strFormula, _
    "ROOND", "ROUND", , , vbTextCompare)

```

V programu dále používáme šest funkcí pro překlad. Jedná se o funkce `GetTopBottomTypeString`, `GetIconSetTypeString`, `GetIconSetIDString`, `GetOperatorTypeString`, `GetAboveBelowTypeString` a `GetColorScaleTypeString`. Ty zde však nezobrazujeme. Jejich úkolem je pouze převádět předané hodnoty na příslušné názvy konstant. Převod se provádí podle následujícího vzoru:

```

'=====
' Na CD          Data\Priklady\04_Zpracovani_dat\
' Název souboru  04_03_FormatConditions.xlsm
' List          Zobrazení vzorců
' Modul         mdl_04_07_General
'=====
Public Function GetIconSetIDString( _
    IconSetID As Long _
    ) As String

    ' Z předané hodnoty vytvoří text,
    ' který obsahuje názvy konstant
    Select Case IconSetID
        Case xl3Flags
            GetIconSetIDString = "xl3Flags"

```

Vlastnost `Priority` objektu `objCondition` vrací prioritu pravidla – čím nižší je hodnota, tím vyšší je priorita aktuálního pravidla. Excel zobrazuje formátování pouze u pravidla s nejvyšší prioritou, u něhož je navíc podmínka `Pravda`. Pokud není u žádného pravidla hodnota `Pravda`, pak se přistupuje ke standardně nastavenému formátování buněk.

Vlastní formátování pravidla se skrývá ve vlastnostech `Interior`, `Borders`, `Font` a `NumberFormat` objektu `objCondition`. Hodnoty těchto vlastností se postupně načítají a ukládají do výstupního pole.

Obsah následující části procedury závisí na typu pravidla. Typ pravidla je přitom dán vlastností `Type` objektu `objCondition`.

- Pokud se jedná o hodnotu, kdy typ přijímá hodnotu konstanty `xlCellValue`, pak funkce `IsCellValueTrue` vrací informaci o tom, zda je podmínka splněna. Jako argument se jí předává objekt `objCondition`. Tato funkce se pak později zapisuje zvlášť.
- Pokud se jedná o typ `xlExpression`, pak se zjišťuje, zda se v oblasti hodnot nachází hodnota buňky, která obsahuje vlastnost `Formula1`. Oblastí hodnot může být vzorec nebo název oblasti. Nejprve se učiní pokus o výpočet vzorce pomocí metody `Evaluate`, a pokud se výpočet nezdaří, pak se vyzkouší získat oblast s názvem `Formula1`. Pokud se dostaneme k nějaké platné oblasti (`Range`), pak se použije funkce tabulky `CountIf`, kterou se zjistí, zda se v ní nachází hodnota buňky.

- U typu `xlUniqueValues` se při nastaveném pravidlu jedná o označení unikátů nebo duplikátů. Vlastnost `ApplyTo` udává oblast, na níž se bude testovat, zda je hodnota buňky jednoznačná, nebo zda se vyskytuje dvakrát. Tuto informaci vám poskytne funkce `CountIf`.
- Podmínka typu pravidla `xlTextString` pracuje se vzorcem, který překládá do angličtiny, pomocí metody `Evaluate` spočítá hodnotu vzorce a vrací pravdivostní hodnotu.
- Novinkou je typ `xlColorScale`, který zodpovídá za zobrazování barevných škál, které představují určitou hodnotu. Zde se načítají pouze vlastnosti barev, které se ukládají do pole.
- Podobně se chovají i datové sloupce, kterým odpovídá typ `xlDatabar`. V tomto případě se načítají a ukládají vlastnosti typu barva, maximální a minimální hodnota a také informace, zda se jedná o hodnoty typu procenta.
- U typu `xlTop10` se načítají vlastnosti typu pořadí, řada a informace o tom, zda se jedná o údaje v procentech.
- Další novinkou je typ `xlIconSets`, u nějž se různým hodnotám v určité oblasti přiřazují různé ikony.
- Funkce `GetIconSetIDString` přitom překládá hodnotu do názvu své konstanty, která označuje aktuálně používaný soubor ikon. Tato hodnota pochází od objektové vlastnosti `IconSet.ID`. Typ sady ikon se překládá do svého názvu konstanty pomocí funkce `GetIconSetTypeString` a operátor pravidla pak pomocí funkce `GetOperatorTypeString`.
- Mezi další typy, které se rovněž zpracovávají, patří `xlBlanksCondition`, `xlNoBlanksCondition`, `xlTimePeriod`, `xlErrorsCondition` a `xlNoErrorsCondition`. Všechna kritéria existují jako vzorce, které se překládají do angličtiny a které se počítají pomocí funkce `Evaluate`; také se testuje jejich pravdivostní hodnota.

Všechny zjištěné informace se okamžitě zapisují do cílového listu, přičemž pro každou informaci je vyhrazen jeden sloupec. O druhu informace vás informuje nadpis sloupce. Vzhledem k tomu, že pravidlo `Icon` může mít různý počet kritérií, může se stát, že v buňce pro nadpis (pro sloupec 72 a další) žádný nadpis nenajdete.

Po vypsaní všech informací se může pro právě aktivní buňku zpracovávat další nastavené pravidlo. Tyto informace se pak zapíší do dalšího řádku cílového listu, nicméně první dva sloupce se nevyplní. To znamená, že v řádcích, kde jsou dva první sloupce prázdné, se jedná o pravidla pro buňku nad tou aktuální, u níž se v prvních dvou sloupcích vyskytují hodnoty.

Po zpracování všech pravidel pro buňku se pokračuje zpracováním další buňky, která obsahuje podmíněné formátování.

```
'=====
' Na CD                Data\Priklady\04_Zpracovani_dat\
' Název souboru        04_03_FormatConditions.xlsm
' List                Zobrazení vzorců
' Modul               mdl_04_07_General
'=====
Public Sub PrintFormatCondition()
    Dim rngCell        As Range
    Dim rngList        As Range
    Dim lngCount       As Long
    Dim objCondition   As Object
    Dim varValue       As Variant
    Dim i              As Long
```