LEKCE 3

Uživatelské funkce (UDF)

V této lekci najdete:

Procedury a funkce

Většina příkladů kódu, který jste zatím v knize viděli, měla podobu tzv. procedur. Procedura je program ve VBA, který nevrací žádnou hodnou. Například následující procedura skryje všechny listy aktivního sešitu kromě listu s názvem Report (alespoň jeden viditelný list musí v každém případě v sešitu zůstat). Všimněte si deklarace obecné objektové proměnné `sh As Object`. Pokud bychom použili striktní typ Worksheet, kód by nepracoval pro listy s grafem, které jsou členy kolekce `Sheets`, avšak nejsou členy kolekce `Worksheets`. Protože typ objektu Sheet neexistuje, musíte použít proměnnou obecnou.

03-01 Funkce UDF.xlsm
modul MSubs

```
Sub HideSheets()

    Const SHEET_NAME As String = "Report"
    Dim x As Worksheet
    Dim sh As Object

    ' pokud list neexistuje, kód ukončit
    On Error Resume Next
    Set x = Sheets(SHEET_NAME)
    If x Is Nothing Then
        Exit Sub
    Else
        x.Visible = xlVisible
    End If
    On Error GoTo 0

    ' skrýt všechny listy kromě daného
    For Each sh In ActiveWorkbook.Sheets
        If StrComp(sh.Name, SHEET_NAME, vbTextCompare) <> 0 Then
            sh.Visible = xlSheetHidden
        End If
    Next sh

End Sub
```

Tato procedura, která může být deklarována jako `Public` (je spustitelná i z jiných modulů nebo přes uživatelské rozhraní) nebo `Private` (je spustitelná pouze z jiných procedur a funkcí v tomtéž modulu), nepřijímá žádné parametry. Procedury obecně nepředávají svým jménem žádnou hodnotu vnějšímu kódu. Samozřejmě že je v nich možno měnit hodnoty modulových i globálních proměnných, které uchovávají svou hodnotu i po skončení této procedury, pokud je sešit otevřen. Tato procedura obecně příliš užitečná není, protože pracuje pouze s aktivním sešitem a pro zadané jméno listu, vepsané v kódu.

Můžete napsat proceduru obecnější, která bude přijímat dva parametry. Prvním bude povinné jméno listu a druhým bude nepovinný parametr, přijímající objekt typu Workbook. Pokud tento parametr chybí, kód použije aktivní sešit.

03-01 Funkce UDF.xlsm
modul MSubs

```
Sub HideSheets2(SheetName As String, Optional ObjectWB As Workbook)

    Dim wb As Workbook
    Dim x As Worksheet
    Dim sh As Object

    ' pokud list neexistuje, kód ukončit
    On Error Resume Next
    Set x = Sheets(SheetName)
    If x Is Nothing Then
        Exit Sub
    Else
        x.Visible = xlVisible
    End If
    On Error GoTo 0

    ' pokud parametr chybí, použij aktivní sešit
    If ObjectWB Is Nothing Then
        Set wb = Activewrokbook
    Else
        Set wb = ObjectWB
    End If


    ' skrýt všechny listy kromě daného
    For Each sh In wb.Sheets
        If StrComp(sh.Name, SheetName, vbTextCompare) <> 0 Then
            sh.Visible = xlSheetHidden
        End If
    Next sh

End Sub
```

Tuto proceduru již není možné spustit přes uživatelské prostředí. Musíte ji spustit z jiné procedury nebo z okna Immediate jedním z následujících způsobů:

```
Sub Test_HideSheets()

HideSheets2 "Report", "02-08 UDFS.xlsm"

Call HideSheets2("Report", "02-08 UDFS.xlsm")

End Sub
```

Pokud použijete výraz `Call`, seznam argumentů *musí* být uzavřen do závorek. Pokud výraz `Call` nepoužijete, seznam argumentů do závorek uzavřen být *nesmí* (ačkoli můžete samozřejmě do závorek uzavřít *jednotlivé* argumenty!).

Uživatelské funkce se od procedur liší tím, že vrací hodnotu, která v kódu dále může a nemusí být využita. Proceduru můžete změnit na funkci prostou změnou slova `Sub` na `Function`. Bude praco-

vat stejně, ale už ji nebude možno spustit na kartě Vývojář, skupina Kód. Můžete ji využít pouze v další proceduře nebo funkci, případně spustit v okně Immediate. Hlavní výhodou je ale možnost vracet hodnotu. V této knize jste se již setkali s funkcí, která vrací `True`, pokud je list v daném sešitu přítomen, a `False`, pokud chybí:

```
Public Function bWorksheetExists(SheetName As String) As Boolean

On Error Resume Next
    bWorksheetExists = Len(Worksheets(SheetName).Name) > 0
    Err.clear
On Error Goto 0

End Function
```

Tento kód ovšem pracuje pouze pro aktivní sešit, neboť neumožňuje specifikovat sešit konkrétní. Obecnější funkce, která bude pracovat s aktivním sešitem, pokud druhý argument chybí, může vypadat například takto:

03-01 Funkce UDF.xlsm
modul MFunctions

```
Public Function bWorksheetExists(SheetName As String, _
                        Optional ObjectWB As Workbook) _
                                                    As Boolean
Dim wb As Workbook
If ObjectWB Is Nothing Then
    Set wb = ActiveWorkbook
Else
    Set wb = ObjectWB
End If


On Error Resume Next
    bWorksheetExists = Len(wb.Worksheets(SheetName).Name) > 0
    Err.Clear
On Error GoTo 0

End Function
```

Pokud použijete tento kód v původní proceduře, můžete ji přepsat následujícím způsobem:

03-01 Funkce UDF.xlsm
modul MSubs

```
Sub HideSheets3(SheetName As String, Optional ObjectWB As Workbook)

    Dim wb As Workbook
    Dim sh As Object

    ' pokud parametr chybí, použij aktivní sešit
    If ObjectWB Is Nothing Then
        Set wb = ActiveWorkbook
    Else
```

```
        Set wb = ObjectWB
    End If

    ' pokud list neexistuje, kód ukončit
    If bWorksheetExists(SheetName, wb) Then
       Set sh = Sheets(SheetName)
    Else
        Exit Sub
    End If

    ' skrýt všechny listy kromě daného
    For Each sh In wb.Sheets
        If StrComp(sh.Name, SheetName, vbTextCompare) <> 0 Then
            sh.Visible = xlSheetHidden
        End If
    Next sh

End Sub
```

Pokud jsou předávanými argumenty proměnné, můžete je funkci nebo proceduře předat pomocí klíčových slov `ByVal` (předání hodnotou) a `ByRef` (předání odkazem). Výchozí metodou předání při vynechání těchto slov je `ByRef`. Pokud je argument předán odkazem na původní proměnnou, vnořená funkce tuto původní proměnnou změní. Při předání hodnotou je vytvořena kopie původní proměnné a ta zůstává beze změny. Klíčové slovo `ByVal` se používá zejména při předávání argumentů tzv. API funkcím Windows, kdy by změna původní systémové hodnoty mohla způsobit i pád systému.

Vše ozřejmí následující příklad. Mějme jednoduchou proceduru, která zvyšuje hodnoty tří proměnných o jedničku:

```
Sub Increment(X As Long, _
                ByRef Y As Long, _
                ByVal Z As Long)

    X = X + 1
    Y = Y + 1
    Z = Z + 1

End Sub
```

Proceduru vnoříme do testovací procedury, která jí předá tři proměnné:

```
Sub Test_Increment()
Dim lX As Long, lY As Long, lZ As Long
lX = 1: lY = 1: lZ = 1

Call Increment(lX, lY, lZ)

Debug.Print lX, lY, lZ

End Sub
```

Výsledkem je výpis, který ukáže, že první dvě původní proměnné, předané odkazem, byly zvýšeny o jedničku. Třetí proměnná zůstala beze změny, neboť subprocedura pracovala s její *kopií*:

```
2             2             1
```

Funkci můžete jako argument předat i pole. Rovněž tak může funkce pole i vrátit. Následující funkce přijímá pole hodnot, z nichž vybere pouze jedinečné hodnoty a ty setřídí.

03-01 Funkce UDF.xlsm
modul MFunctions

```
Public Function SortUniques(Values As Variant)
    Dim Uniques(), y, i As Long
    Dim col As New Collection

    ' pro případ, že je funkce volána z listu
    If TypeName(Values) = "Range" Then
        y = WorksheetFunction.Transpose(Values)
    Else
        y = Values
    End If

    'pokud  argument tvoří jen jedna hodnota
    If Not IsArray(y) Then
        SortUniques = y
        Exit Function
    End If

    'pokud je pole vícerozměrné, vrať chybu
    On Error Resume Next
        Err.Clear
        If UBound(y, 2) > 0 Then
            If Not CBool(Err.Number) Then
                SortUniques = CVErr(xlErrNum)
                Err.Clear
                Exit Function
            End If
        End If
    On Error GoTo 0

    'kolekce jedinečných hodnot
    On Error Resume Next
        For i = LBound(y) To UBound(y)
            col.Add y(i), CStr(y(i))
        Next i
    On Error GoTo 0

    'kopírování kolekce do pole
    ReDim Uniques(1 To col.Count)
    For i = 1 To col.Count
        Uniques(i) = col.Item(i)
    Next i
```

```
    'volání třídicího algoritmu
    BubbleSort Uniques

    'přiřazení setříděného pole do proměnné
    SortUniques = Uniques

End Function
```

Funkce je mírně složitější, než by mohla být, protože je její začátek přizpůsoben tak, aby ji bylo možno volat i z listu. Pokud je do argumentu `Values` typu `Variant` předán objekt `Range`, znamená to, že funkce byla zavolána z listu. Do pomocného pole je v takovém případě přiřazená transformace této oblasti tak, aby bylo výsledkem vodorovné jednorozměrné pole. Při přímém přiřazení by vzniklo dvojrozměrné pole svislé.

Dalším krokem je kontrola na jednu hodnotu. V takovém případě samozřejmě není co třídit, hodnota je předána do funkce a funkce končí.

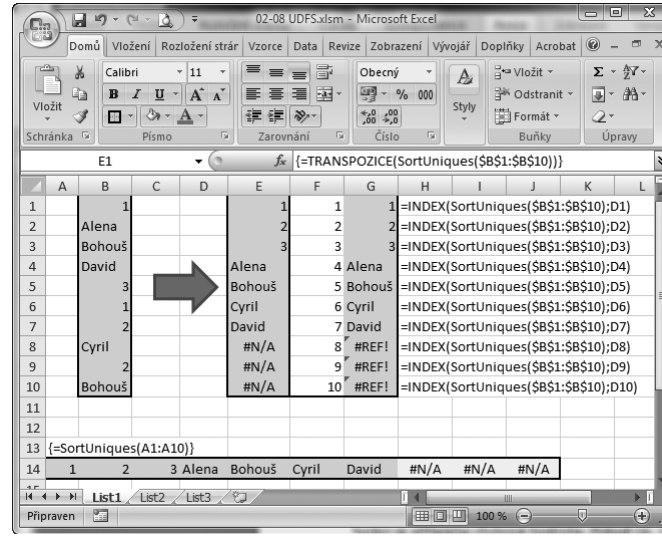
Ačkoli funkci můžete modifikovat tak, aby pracovala i pro vícerozměrná pole, v tomto případě kontrolujeme, jestli je horní mez druhého rozměru větší než nula. Pokud ano, pole je vícerozměrné a funkci je přiřazena chybová hodnota. Pokud ne, výraz `UBound(y,2)` vyvolá výjimku a výraz `Not Cbool(Err.Number)` bude vyhodnocen jako `False`.

Další část kódu přidá jednotlivé prvky pole do kolekce, a protože je používán jednoznačný klíč shodný s hodnotou, žádná hodnota nebude do kolekce přidána dvakrát.

Obsah kolekce poté nahrajeme do pole a pole předáme proceduře `BubbleSort` ke třídění. Výsledek je nakonec zpětně předán funkci `SortUniques`.

Pro vývojáře v Excelu má význam, že funkci je možno použít v listu několika způsoby, jak je vidět na obrázku.

Do oblastí E1:E10 a A14:J14 jsme zadali jediný maticový vzorec tak, že jsme vybrali celou oblast, zapsali vzorec a stiskli Ctrl+Shift+Enter. V oblasti G1:G10 jsou obyčejné vzorce.



Obrázek 3.1: Použití vlastní funkce v listu

Použití vlastních funkcí jako vzorců v listu

Pokud chcete použít funkci jako vzorec v listu, musí být zapsána v normálním modulu. Pokud bude v modulu některé třídy, ať již se jedná o list, sešit, formulář nebo vlastní třídu, bude pro uživatelské rozhraní neviditelná. Rovněž nesmí být deklarována klíčovým slovem `Private`.

Další důležité rozhodnutí, které musíte učinit, je, ve kterém sešitu bude funkce uložena. Máte na výběr z několika možností.

Pokud je funkce pouze pro potřebu na vašem počítači a chcete, aby byla dostupná pro jakýkoli otevřený sešit, můžete ji uložit do normálního modulu ve skrytém sešitu **Personal.xlsb**. Tento sešit je vytvořen automaticky při nahrání prvního makra s volbou Uložit makro do osobního sešitu a ve Windows Vista je uložen v osobní složce %AppData%\Roaming\Microsoft\Excel\XLSTART.

Pokud funkce souvisí se sešitem, který je distribuován mezi více uživatelů, můžete funkci umístit do modulu tohoto sešitu.

Má-li být funkce k dispozici více uživatelům pro různé sešity, můžete ji distribuovat pomocí doplňku (Add-in). Kód v nainstalovaném doplňku poběží i při vysokém stupni zabezpečení před makry.

Jestliže jsou soubory vytvářeny na základě šablony, uložte funkci do modulu této šablony.

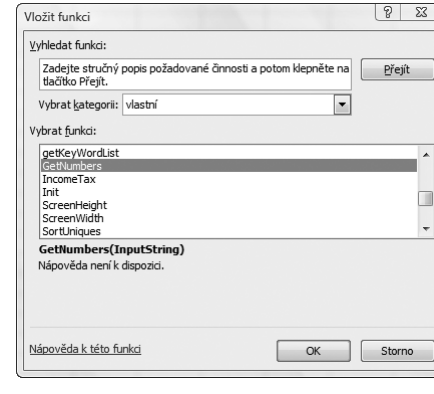
Když znáte nazpaměť jméno funkce a její parametry, můžete vzorec zapsat do buňky přímo. Pokud si nejste jisti, můžete použít standardní postup vložení funkce přes průvodce.

Jako příklad jsme vzali funkci, která z libovolného textu extrahuje číslice, přičemž převodem na typ `Long` nevýznamné vedoucí nuly zleva „zahodí". Toto je samozřejmě možno změnit vypuštěním závěrečné konverze:

03-01 Funkce UDF.xlsm
modul MFunctions

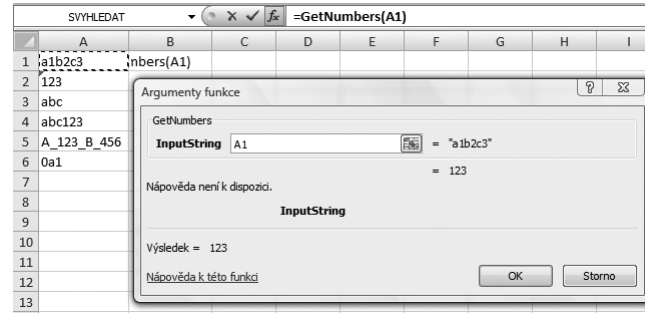
```
Function GetNumbers(InputString As String)
    Dim tmp As String, char As String
    Dim i As Long
    For i = 1 To Len(InputString)
        char = Mid$(InputString, i, 1)
        If char Like "[0-9]" Then
            tmp = tmp & char
        End If
    Next i
    If Len(tmp) > 0 Then
        GetNumbers = CLng(tmp)
    Else
        GetNumbers = vbNullString
    End If
End Function
```

Vyberte buňku, do které chcete vzorec vložit, a na kartě Vzorce, sada Knihovna funkcí, klepněte na položku Vložit funkci. Vyberte kategorii Vlastní a vyberte funkci GetNumbers.



Obrázek 3.2: Vložení funkce ze seznamu dostupných vlastních funkcí

Po klepnutí na tlačítko OK se objeví standardní dialog, ve kterém můžete vybrat vstupní buňku.



Obrázek 3.3: Dialogové okno vložení vlastní funkce

Pro soukromé potřeby to většinou postačí, avšak pokud budete funkci distribuovat jiným uživatelům, pravděpodobně budete chtít přidat alespoň stručnou nápovědu a také funkci zařadit do příslušné kategorie. K tomu slouží metoda `MacroOptions` objektu `Application`. Pokud se podíváte do nápovědy VBA, zjistíte, že textové funkce jsou v kategorii číslo 7.

Procedura, která funkci zaregistruje, je jednoduchá a přijímá tři parametry:

03-01 Funkce UDF.xlsm
modul MSubs

```
Sub RegisterFunction(fName As String, fDescription, fCategory)
    Application.MacroOptions Macro:=fName, _
                             Description:=fDescription, _
                             Category:=fCategory
End Sub
```

Toto proceduru můžete volat pro zaregistrování funkce například takto:

```
Sub Register_GetNumbers()
    Dim fnName As String
    Dim fnDesc As String
    Dim fnCat As Long

    fnName = "GetNumbers"
    fnDesc = "Extrahuje čísla z libovolného řetězce." & vbLf
    fnDesc = fnDesc & "Nevýznamné nuly nejsou brány v úvahu."
    fnCat = 7

    Call RegisterFunction(fnName, fnDesc, fnCat)

End Sub
```

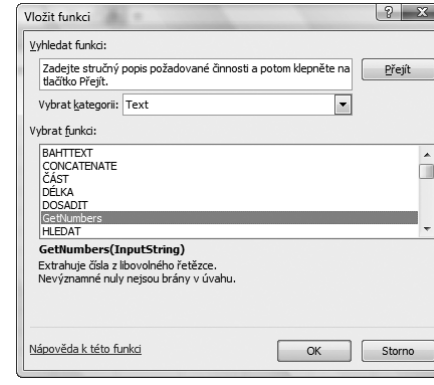
Pokud nyní znovu spustíte průvodce vložením vzorce, funkce již nebude v kategorii vlastní, nýbrž v kategorii textové funkce.

Pokud chcete vzorec odregistrovat, zavolejte proceduru tímto způsobem:

```
Sub UnRegister_GetNumbers()
    Dim fnName As String

    fnName = "GetNumbers"
```

```
    Call RegisterFunction(fnName, Empty, Empty)

End Sub
```

Pokud toto provedete, funkce již nebude viditelná v průvodci, a to v žádné kategorii. Na listu ji však můžete dále použít. Jestliže ji chcete vrátit do kategorie vlastní, zaregistrujte ji pod číslem 14. Pokud místo čísla předáte jméno vlastní kategorie, bude tato kategorie vytvořena.



Obrázek 3.4: Vlastní funkce zaregistrovaná v kategorii textových funkcí s vlastním komentářem

Předešlé verze Excelu po odregistrování funkci ponechaly v kategorii Vlastní. Zbavit se jí nadobro nebyl snadný úkol a jediná možnost byla využití jazyka XLM. Tato metoda přesahuje rámec této knihy.

V kategorii Vlastní se objevují všechny funkce ve VBA, které jsou umístěny v normálních modulech a nejsou deklarovány klíčovým slovem `Private`. Pokud tyto funkce nechcete v dialogu Vložit funkci vidět, použijte v záhlaví modulu výraz `Option Private Module`. Funkce bude teoreticky stále možné v listu jako vzorce použít, ale nebudou se objevovat ani v rozbalovacím seznamu funkcí, ani v dialogu. Musíte jejich jméno a argumenty znát nazpaměť.

Uživatelská funkce, použitá v listu (UDF), nemůže měnit žádné vlastnosti buněk v listu ani hodnoty jiných buněk, ale pouze hodnoty buňky, ve které je zapsána. To se samozřejmě týká i formátu. Pro dynamické přizpůsobení formátu hodnotě používejte podmíněný formát. Funkce nevrátí chybu, ale všechny příkazy pokoušející se měnit cokoli jiného, než hodnotu buňky se vzorcem, budou ignorovány. Funkce nemůže odstraňovat ani přidávat listy.

UDF nemůže měnit ani většinu nastavení aplikace Excel. Například tyto příkazy budou ignorovány:

```
ActiveWindow.DisplayGridlines = False
Application.DisplayFormulaBar = False
Application.StatusBar = "Kalkuluji..."
Application.Wait Now + TimeValue("0:00:01")
```

V Excelu 2007 je však na rozdíl od předešlých verzí možno měnit typ kurzoru:

```
Application.Cursor = xlWait
Application.Cursor = xlDefault
```

Výše uvedené se týká pouze funkce, použité ve vzorci v listu. Tatáž funkce, volaná z jiné procedury nebo funkce, vykoná všechny příkazy.

UDF musí být umístěna v normálním modulu. Nesmí být umístěna v modulu jakékoli třídy, ať již se jedná o třídu `ThisWorkbook`, `Worksheet`, `Chart`, formulář, nebo třídu vlastní.

Nesnažte se měnit argumenty, předané pomocí klíčového slova `ByRef`. Jakmile se funkce pokusí změnit hodnotu v předané oblasti, je porušeno pravidlo uvedené výše a funkce vrátí chybu #HODNOTA!:


03-01 Funkce UDF.xlsm
modul MFunctions

```
Function FindRow(ByRef rg As Range, str As String)
    Dim clFound As Range
    Set clFound = rg.Find(str, LookIn:=xlValues, LookAt:=xlPart)
    If clFound Is Nothing Then
        FindRow = 0
    Else
        FindRow = clFound.Row
    End If

    'tento řádek zaviní chybu
    rg.Cells(1, 1) = "dummy"

End Function
```

Do UDF vždy předávejte jako argumenty všechny buňky nebo oblasti, na kterých je výsledek závislý. Pokud tak neučiníte, funkce bude přepočítána pouze tehdy, když se změní obsah oblastí, které jsou jako argumenty předány. Pomocí příkazu `Application.Volatile` můžete funkci přinutit přepočítávat se při změně jakékoli buňky, avšak toto vede ke znatelnému zpomalení vaší aplikace. Proto toto řešení používejte pouze v nouzi, kdy všechny argumenty jako hodnoty předat nelze.

Pokud se UDF může odkazovat na prázdné buňky, doporučujeme kontrolu pomocí funkce `IsEmpty()`.

Příklady uživatelských funkcí

Na závěr této lekce uvedeme několik funkcí, které mají skutečný praktický význam. Pokud je pečlivě prostudujete, jistě dokážete napsat funkce podobné, které budou sloužit vašemu konkrétnímu účelu.

Funkce vracející výši progresivní daně

Poměrně běžnou úlohou je výpočet částky, kdy je v určitých pásmech pevná procentní sazba, jako je tomu například u v politice často diskutované progresivní daně. Následující funkce přijímá tři parametry: vstupní částku, spodní hranice pásem a jejich sazby. Samozřejmě předpokládáme, že následující spodní hranice je zároveň horní hranicí pásma předcházejícího.


03-01 Funkce UDF.xlsm
modul MFunctions

```
Public Function IncomeTax(Income As Double, _
                          rgHurdles As Range, _
                          rgRates As Range)

    Dim cl As Range, i As Long
    Dim tmp As Double

    If rgHurdles.Cells.Count <> rgRates.Cells.Count Then
        IncomeTax = CVErr(xlErrNum)
        Exit Function
```

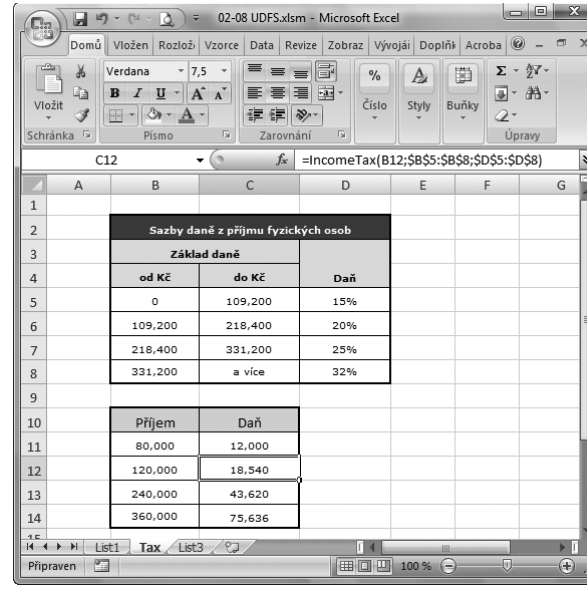
```
    End If

    For Each cl In rgHurdles
        i = i + 1
        If Income > cl.Value Then
            If Income > cl(2).Value And i <> rgHurdles.Cells.Count Then
                tmp = tmp + (cl(2).Value - cl.Value) * rgRates(i)
            Else
                tmp = tmp + (Income - cl.Value) * rgRates(i)
            End If
        Else
            GoTo ExitHere
        End If
    Next cl
ExitHere:
    IncomeTax = tmp
End Function
```

Funkci můžete použít například podle obrázku.

Například pro základ 120000 Kč bude daň vypočítána takto:

(120000–109200)*20%+109200*15% = 18540

Funkce samozřejmě funguje i pro snižující se sazby, jak tomu bývá například u degresivní provize z prodeje.



Obrázek 3.5: Použití vlastní funkce, vypočítávající progresivní daň, v listu

Funkce sčítající barvy

Ačkoli pro obarvení buněk podle hodnot je správné využít podmíněný formát, někdy máte obarvené buňky přímo přes uživatelské menu a chcete spočítat jejich počet pro danou barvu, nebo součet buněk dané barvy.

Pro různé typy kalkulací můžete samozřejmě napsat různé funkce, avšak následující funkce byla navržena tak, aby volbou třetího parametru bylo možno rozlišit mezi počtem („Count“) a součtem („Sum“):

03-01 Funkce UDF.xlsm
modul MFunctions

```
Public Function CalcColor(rg As Range, _
                          Optional rgColor As Range, _
                          Optional Operation As String)
    Dim cl As Range
    Dim lColor As Long
    Dim tmp

    If rgColor Is Nothing Then Set rgColor = Application.Caller
```

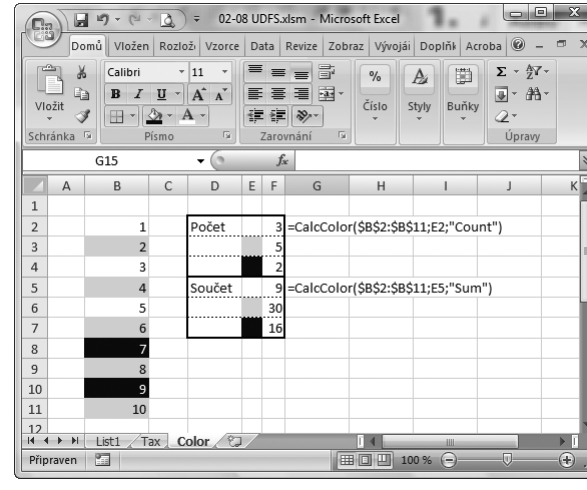
```
    If Len(Operation) = 0 Then Operation = "Sum"

    If IsError(Application.Match(Operation, Array("Sum", "Count", 0), 0)) Then
        CalcColor = CVErr(xlErrNum)
        Exit Function
    End If

    lColor = rgColor.Interior.ColorIndex

    For Each cl In rg
        If cl.Interior.ColorIndex = lColor Then
            Select Case Operation
            Case "Sum"
                tmp = tmp + Val(cl.Value)
            Case "Count"
                tmp = tmp + 1
            End Select
        End If
    Next cl

    CalcColor = tmp

End Function
```

Funkce kontroluje, zda byly nepovinné parametry zadány. Pokud tomu tak není, použije jako referenční buňku barvy buňku, ve které je vzorec zapsán. Výchozím typem kalkulace je sčítání.

Obrázek ukazuje možné použití.

Výsledek 30 dostanete i zapsáním následujícího vzorce do buňky E6:

=CalcColor(B2:B11)



Obrázek 3.6: Použití funkce, počítající s buňkami určité barvy, v listu

Funkce vracející údaje o souboru a uživateli

Funkci zkoumající, zda list v daném sešitu existuje, jsme již uvedli. Rovněž následující funkci, která prověří, zda je sešit otevřen, jste již v knize viděli:

```
Function bIsWorkbookOpen(FileName As String) As Boolean
    On Error Resume Next
    bIsWorkbookOpen = CBool(Len(Workbooks(FileName).Name))
End Function
```

Jméno souboru a jeho plné cesty můžete v listu zobrazit pomocí následujících funkcí:

03-01 Funkce UDF.xlsm
modul MFunctions

```
Function WBName()
    WBName = ThisWorkbook.Name
```

```
End Function

Function WBFullName()
    WBFullName = ThisWorkbook.FullName
End Function

Function WBPath()
    WBPath = ThisWorkbook.Path
End Function
```

Objekt `Application` má samozřejmě řadu dalších vlastností, které můžete tímto způsobem využít ve své funkci.

Dalším objektem, který můžete použít ve funkci, je objekt `FileSystemObject` z knihovny Microsoft Scripting Runtime (soubor *scrrun.dll*). Například následující funkce vrátí velikost souboru v bytech:

03-01 Funkce UDF.xlsm
modul MFunctions

```
Function FileSize(FullName As String)

    Dim fso As Object
    Dim f As Object
    Dim fl As File

    ' inicializace objektů
    Set fso = CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
        Set f = fso.GetFile(FullName)
    On Error GoTo 0

    ' pokud soubor neexistuje, vrať prázdný řetězec
    If f Is Nothing Then Then
        FileSize = vbNullString
    Else
        FileSize = f.Size
    End If

    ' uvolnění proměnných
    Set f = Nothing
    Set fso = Nothing

End Function
```

Funkci můžete použít v kombinaci s předešlými vzorci a můžete v listu zobrazit aktuální velikost sešitu. Do kterékoli buňky zapište tento vzorec.

=FILESIZE(WBFULLNAME())

Pomocí objektu `FileSystemObject` můžete získat mnoho dalších informací, jako například informace o souboru, jak je popsáno v tabulce 3.1.

Tabulka 3.1: Vlastnosti objektu File (soubor)

Vlastnost	Význam
f.DateCreated	Datum a čas vytvoření souboru
f.DateLastAccessed	Datum a čas posledního přístupu k souboru
f.DateLastModified	Datum a čas posledního uložení souboru
f.Drive	Písmeno diskové jednotky, na níž je soubor uložen
f.ParentFolder	Složka, ve které se soubor nachází
f.Path	Úplná cesta k souboru včetně jeho jména (!)
f.Type	Popis typu souboru tak, jak je vidět v průzkumníku

Po mírné modifikaci bude funkce pracovat pro složku:

03-01 Funkce UDF.xlsm
modul MFunctions

```
Function FolderSize(FolderPath As String)

    Dim fso As Object
    Dim fld As Object

    ' inicializace objektů
    Set fso = CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
        Set fld = fso.GetFolder(FolderPath)
    On Error GoTo 0

    ' pokud složka neexistuje, vrať prázdný řetězec
    If fld Is Nothing Then
        FolderSize = vbNullString
    Else
        FolderSize = fld.Size
    End If

    ' uvolnění proměnných
    Set fld = Nothing
    Set fso = Nothing

End Function
```

Objekt `Folder` má podobné vlastnosti jako objekt `File`. Má samozřejmě i dodatečné členy, jako například kolekci `Subfolders`. Využití této kolekce umožňuje procházet složku včetně všech jejích vnořených složek. Následující funkce prochází složku a hledá soubory, jejichž název obsahuje danou příponu. Pokud narazí na podsložky, zavolá sama sebe a znovu prochází všechny soubory dané podsložky. Tento algoritmus se nazývá *rekurzivní*. Jiné jeho použití je například u metody třídění *Quicksort*, která využívá binární třídění a je obecně rychlejší, než metoda *Bubble-sort*.

Následující příklad spočítá počet souborů určitého typu v dané složce.

03-01 Funkce UDF.xlsm
modul MFunctions

```
Function CountFiles(sFolder As String, _
                    sExtension As String, _
            Optional bSubDirs As Boolean)

    Dim fso As Object
    Dim fld As Object, sfld As Object
    Dim f As Object
    Dim lCounter As Long

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set fld = fso.GetFolder(sFolder)

    For Each f In fld.Files
        If StrComp(Right$(f.Name, 4), sExtension, vbTextCompare) = 0 Then
            lCounter = lCounter + 1
        End If
    Next f

    If bSubDirs Then
        For Each sfld In fld.SubFolders
            lCounter = lCounter + CountFiles(sfld.Name, sExtension, True)
        Next
    End If

    CountFiles = lCounter

    Set fso = Nothing

End Function
```

Funkci můžete použít například takto:

```
CountFiles(ThisWorkbook.Path, "xlsm", True)
```

V tomto případě vrátí počet všech sešitů s příponou „xlsm“ (sešit Excelu s povolenými makry) ve složce, ve které se nachází sešit, z něhož kód spouštíte, včetně všech jejích podsložek.

Jméno uživatele nainstalované sady MS Office vrátí tato jednoduchá funkce:

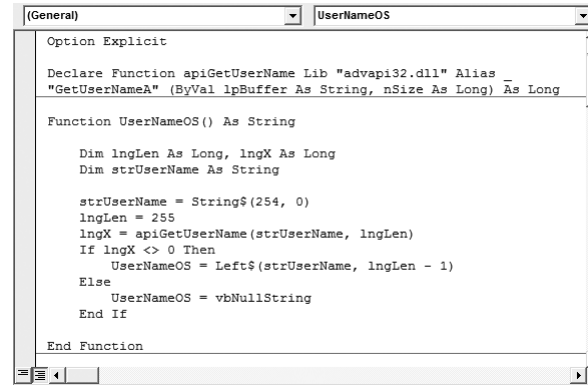
```
Function UserNameApp() As String
    UserNameApp = Application.UserName
End Function
```

Většinou však potřebujete zjistit jméno, pod kterým je uživatel přihlášen do systému. K tomuto účelu je třeba zavolat funkci API Windows. Její deklarace musí být na začátku modulu, před první funkcí nebo procedurou.

03-01 Funkce UDF.xlsm
modul MUser

Funkce vracející některá nastavení Windows

Některé základní informace můžete získat pomocí jednoduché funkce Environ(). Ta přijímá jediný argument, a to buď název informace, kterou chcete získat, nebo celočíselný index. Pokud použijete index, bude návratovou hodnotou řetězec včetně názvu informace, odděleného od informace rovnítkem. Pokud budete testovat v okně Immediate, vše bude jasnější:



Obrázek 3.7: Funkce API, která vrací název účtu přihlášeného uživatele

```
? Environ(5)
COMPUTERNAME=A100

? Environ("ComputerName")
A100
```

Pokud chcete vypsat všechny možnosti, dostupné na vašem počítači, do okna Immediate, použijte například tento kód:

03-01 Funkce UDF.xlsm
modul MSubs

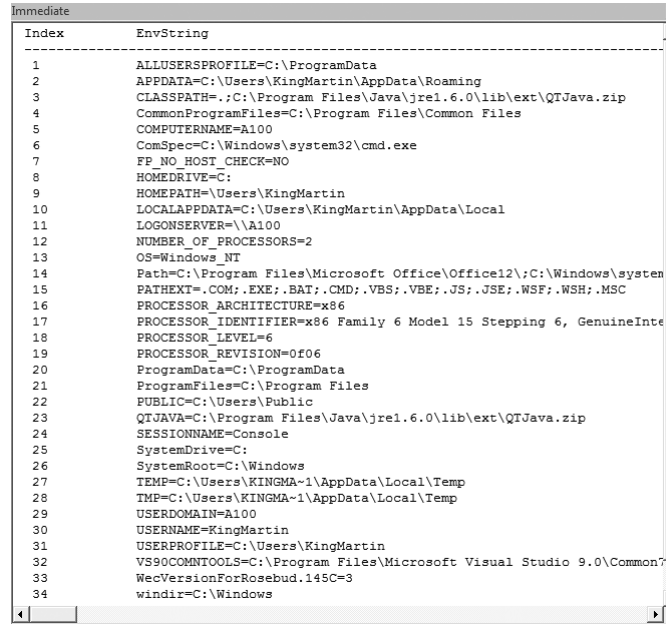
```
Sub ListEnviron()

Dim i As Long, Output As String

    i = 1
    Output = Environ(i)
    Debug.Print "Index", "EnvString"
    Debug.Print String(80, "-")
    Do While Len(Output) > 0
        Debug.Print i, Output
        i = i + 1
        Output = Environ(i)
    Loop

End Sub
```

Výsledkem bude výpis podobný výpisu na obrázku 3.8.

Je samozřejmé, že některé prvky nebudou na všech počítačích existovat. Například údaj, který má na našem systému index 32, existuje pouze tehdy, pokud má uživatel nainstalováno vývojové prostředí Visual Studio 2008.

Proto doporučujeme při využití funkce `Environ()` opatrnost, zejména pokud vaše aplikace poběží na jiných počítačích. Na počítačích Macintosh funkce nefunguje vůbec. Přesto však můžete napsat jednoduché funkce, vracející například jméno uživatele Windows:

03-01 Funkce UDF.xlsm
modul MUser



Obrázek 3.8: Příklad výpisu všech informací dostupných přes funkci Environ

```
Function UserName() As String
    Dim OutPut As String
    OutPut = Environ("Username")
    If Len(OutPut) = 0 Then OutPut = "Nedefinováno"
    UserName = OutPut
End Function
```

nebo jméno počítače:

```
Function ComputerName() As String
    Dim OutPut As String
    OutPut = Environ("ComputerName")
    If Len(OutPut) = 0 Then OutPut = "Nedefinováno"
    ComputerName = OutPut
End Function
```

Bezpečnější je samozřejmě volat přímo systémové funkce API Windows. Následující funkce vrátí údaje o aktuálním rozlišení:

03-01 Funkce UDF.xlsm
modul MResolution

```
Private Const SM_CXSCREEN = 0
Private Const SM_CYSCREEN = 1

Private Declare Function GetSystemMetrics _
                Lib "user32" (ByVal nIndex As Long) As Long
```

```
Public Function ScreenWidth() As Long
    ScreenWidth = GetSystemMetrics(SM_CXSCREEN)
End Function

Public Function ScreenHeight() As Long
    ScreenHeight = GetSystemMetrics(SM_CYSCREEN)
End Function
```

Pokud je vaše aplikace navržena pro minimální dovolené rozlišení, můžete napsat testovací funkci a tu následně použít ve vašem kódu:

```
Public Function bTestResolution(ScrWidth As Long, ScrHeight As Long)
    bTestResolution = ScreenHeight >= ScrHeight And _
                      ScreenWidth >= ScrWidth
End Function

Sub TestResolution()

If Not bTestResolution(1200, 800) Then
    MsgBox "Zvyšte rozlišení na 1200 x 800"
    Exit Sub
Else
    ' kód pokračuje
End If

End Sub
```

Použití vlastností Application.Volatile a Application.Caller

O těchto vlastnostech jsme se již zmínili v lekci o objektu Application. Funkce, kterou používáte v listu, se bude standardně chovat jako nevolatilní funkce, neboli se bude přepočítávat pouze v případě, kdy se změní hodnota v některé z buněk, na které se odkazuje. Pokud chcete, aby se automaticky přepočítávala při změně kterékoli z buněk v listu, doplňte do jejího kódu řádek

```
Application.Volatile
```

Následující funkce vrátí číslo posledního zaplněného řádku sloupce, určeného písmenem nebo jejich kombinací.

03-01 Funkce UDF.xlsm
modul MFunctions

```
Function LastRow(strCol As String)
    Dim ws As Worksheet
    Dim clLast As Range

    Application.Volatile

    Set ws = Application.Caller.Parent

    With ws
        Set clLast = .Cells(.Rows.Count, strCol)
```

```
        If IsEmpty(clLast) Then
            With clLast.End(xlUp)
                If IsEmpty(.Cells(1, 1)) Then
                    LastRow = 0
                Else
                    LastRow = .Row
                End If
            End With
        Else
            LastRow = .Rows.Count
        End If
    End With

End Function
```

Funkce je do rozumné míry robustní, a pokud je poslední buňka sloupce neprázdná, nebo je celý sloupec prázdný, vrací jako výsledek správně plný počet řádků listu, respektive nulu.

Řádek

```
Application.Caller.Parent
```

vrací odkaz na objekt třídy `Worksheet`, v němž je funkce jako vzorec použita. Vlastnost `Caller` objektu Worksheet vrací v tomto případě odkaz na buňku (objekt `Range`), ve které je vzorec zapsán. Vlastnost `Parent` obecně vrací jméno objektu, formuláře nebo kolekce, která vlastní daný objekt, kolekci nebo ovládací prvek. Vlastnost `Caller` může vracet i jiné hodnoty. Podrobnosti najdete v nápovědě.

Otázky

Otázka 3.1:

Jak spustíte proceduru Sub, která přijímá parametr?

Otázka 3.2:

Jaký je rozdíl mezi předáním proměnné jako parametru s klíčovými slovy ByRef a ByVal? Která je výchozí? A kdy musíte použít klíčové slovo ByVal?

Otázka 3.3:

Kam umístíte kód funkce, pokud chcete, aby byla dostupná v kterémkoli sešitu na vašem počítači?

Otázka 3.4:

Napište funkci GetText, která z předaného řetězce vrátí pouze alfabetické znaky (písmena). Pro jednoduchost použijte operátor Like.

Otázka 3.5:

Funkci z předešlé otázky zaregistrujte mezi textové vzorce listu, včetně krátkého popisu. Uveďte i kód pro její odregistrování.

Otázka 3.6:

Uveďte některá základní omezení funkce, pokud je použita jako vzorec v listu. Které příkazy budou ignorovány?

Otázka 3.7:

Jak přinutíte funkci, která je použita jako vzorec v listu, k přepočítání, pokud je změněna jakákoli buňka v sešitu, byť se na ni daná funkce neodkazuje? Jaké je nebezpečí tohoto přístupu?

Otázka 3.8:

Co znamená pojem rekurzivní funkce? Napište příklad rekurzivní funkce vracející faktoriál čísla.

Otázka 3.9:

Proč je zapotřebí opatrnosti při použití funkce Environ?

Otázka 3.10:

Jak se v kódu funkce, použité v listu, odkážete na buňku, ve které je vzorec umístěn?