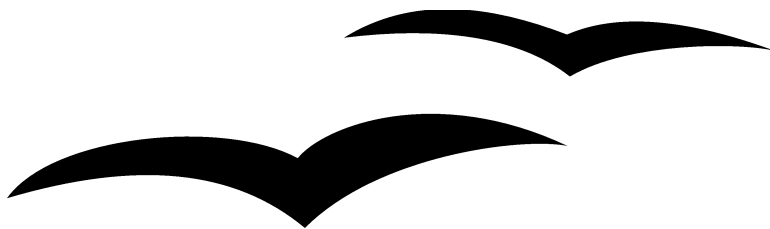


Makrá v OpenOffice.org

Interaktívne vstupy



Obsah

Správy.....	3
Nezalomitelné medzery.....	5
Práca so súbormi.....	9
Textové súbory.....	10
Inicializačné súbory.....	11
Vstupné funkcie.....	14
Spočítavame textové reťazce.....	14
Prevod čísla na text.....	16
Vstupy priamo z dokumentu.....	25
Verzia pre modul Calc.....	27

Začínáme tretiu veľkú časť programovania makier v OpenOffice.org, ktorú zameriame na interaktívne vstupy, t. j. na možnosť riadenia a zadávania rôznych parametrov a informácií priamo v makrách.

Doteraz sme písali makrá, ktoré bežali priamo bez akéhokoľvek interaktívneho zásahu používateľa. Ale napr. už pri nezaľomiteľných medzerách sme spomínali rôzne výklady typografických pravidiel v oblasti spojky „a“. Túto problematiku sme doteraz riešili tak, že sme nechali na každého používateľa, nech si upraví makro podľa seba. Takéto riešenie však nemusí byť postačujúce, pretože ak chceme napr. vo väčších podnikoch toto makro nasadiť každému používateľovi, pričom každý má svoje predstavy o tomto probléme, museli by sme mať dve rozdielne verzie.

Preto musíme do našich makier pridať možnosť interaktívneho zadania niektorých údajov. Pochopiteľne, teraz vzniká otázka, aké prostriedky nám v tomto smere ponúka OpenOffice.org. Je ich pomerne dosť a môžeme ich v zásade rozdeliť do troch skupín – správy, vstupné funkcie a dialógy. V nasledujúcich dieloch si ich postupne preberieme a ukážeme ich možnosti.

Správy

Už pri prvých makrách sme spomínali, že systémová funkcia „msgbox“ má tri parametre, pričom prvý je vypisovaný reťazec, druhý určuje aké tlačidlá sa majú zobrazit' a tretí je názov tohto okna. Teraz sa pristavíme práve pri druhom parametre – definícii tlačidiel, pretože sme zatiaľ používali iba hodnotu 0, pri ktorej sa zobrazovalo jedno jediné tlačidlo „OK“. funkcia „msgbox“ však v tomto smere ponúka omnoho viac možností.

Pochopiteľne, vzniká otázka, aký význam má možnosť zobrazenia viacerých tlačidiel. Táto otázka má jednoduchú odpoveď – „msgbox“ je vlastne funkcia, ktorá vracia ako návratovú hodnotu kódované číslo stlačeného tlačidla. Samozrejme, pokiaľ sme zobrazovali jedno jediné tlačidlo „OK“, nemali sme inú možnosť, ako ho na koniec stlačiť a preto nás návratová hodnota ani nemohla zaujímať.

Pozrime sa však na to, aké hodnoty môže tento parameter nadobúdať a aké tlačidlá sa pri tom vlastne zobrazujú. Aby sme sa vyhli prípadným rozdielom medzi jazykovými verziami a prekladmi, budeme uvádzať ich originálny anglický názov s tým, že v zátvorke uvedieme aj slovenský preklad. Základné možnosti sú nasledovné:

- 0 – zobrazuje sa tlačidlo „OK“
- 1 – zobrazujú sa tlačidlá „OK“ a „Cancel“ („OK“ a „Zrušiť“)
- 2 – zobrazujú sa tlačidlá „Abort“, „Retry“ a „Ignore“ („Prerušit'", „Skúsiť znovu“ a „Ignorovať“)
- 3 – zobrazujú sa tlačidlá „Yes“, „No“ a „Cancel“ („Áno“, „Nie“ a „Zrušiť“)
- 4 – zobrazujú sa tlačidlá „Yes“ a „No“ („Áno“ a „Nie“)
- 5 – zobrazujú sa tlačidlá „Retry“ a „Cancel“ („Skúsiť znovu“ a „Zrušiť“)

Ďalej pomocou tohto istého parametra môžeme určiť, ktoré tlačidlo bude definovať štandardnú návratovú hodnotu (t. j. ak po zobrazení správy hneď stlačíme klávesu „Enter“, ktorú hodnotu vlastne zadáme):

- 0 – štandardnú hodnotu definuje prvé tlačidlo



Obrázok 1: Možnosti tlačidiel vo funkcii "Msgbox"

256 – štandardnú hodnotu definuje druhé tlačidlo
 512 – štandardnú hodnotu definuje tretie tlačidlo

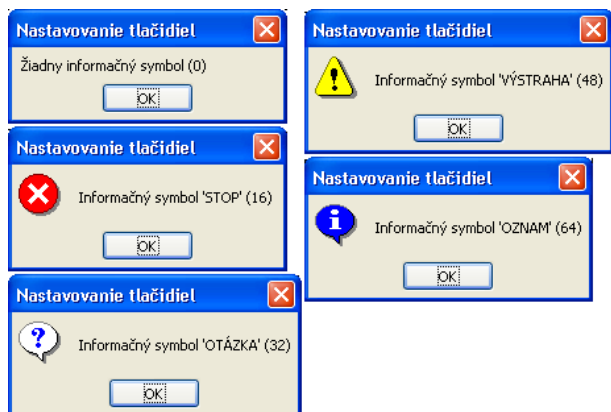
Uvedené hodnoty musíme pripočítať k číslu, pomocou ktorých zobrazujeme požadované tlačidlá, takže napr. pomocou hodnoty 258 (2+256) zobrazíme trojicu tlačidiel „Abort“, „Retry“ a „Ignore“ s tým, že druhé tlačidlo je prednastavené ako štandardné:

`msgbox("Štandardné je druhé tlačidlo (256)", 258, "Nastavovanie tlačidiel")`

Ako posledné, čo môžeme pomocou tohto parametra nastaviť, je obrázkový informačný symbol, ktorý sa zobrazí v ľavej hornej časti okna:



Obrázok 2: Štandardné tlačidlo



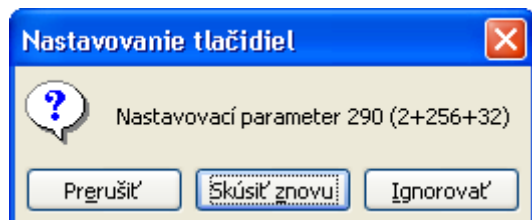
Obrázok 3: Informačné symboly

0 – nezobrazí sa žiadny informačný symbol
 16 – zobrazí sa informačný symbol „Stop“
 32 – zobrazí sa informačný symbol „Otázka“
 48 – zobrazí sa informačný symbol „Výstraha“
 64 – zobrazí sa informačný symbol „Oznam“

Uvedené hodnoty znovu musíme pripočítať k už pripravenému číslu, pomocou ktorého nastavujeme tlačidlá, takže napr. pomocou hodnoty 290 (2+256+32) zobrazíme trojicu

tlačidiel „Abort“, „Retry“ a „Ignore“ s tým, že druhé tlačidlo je prednastavené ako štandardné a ako informačný symbol budeme mať otáznik:

`msgbox("Nastavovací parameter 290 (2+256+32)", 290, "Nastavovanie tlačidiel")`



Obrázok 4: Príklad nastavenia parametra vo funkcii "Msgbox"

Na tomto mieste ešte musíme upozorniť na zvláštny charakter symbolu „Oznam“, pretože pokiaľ ho použijeme, ostatné nastavenia tohto parametra sú ignorované a vždy sa zobrazí iba tlačidlo „OK“, t. j. je úplne jedno, či zadáme hodnotu 64, 65, 66...

Zobrazenie príslušných tlačidiel však nemá praktický význam, ak sa nedozvieme, ktoré tlačidlo bolo stlačené. Ako sme však už naznačili, funkcia „msgbox“ vracia kódované číslo stlačeného tlačidla podľa nasledujúceho zoznamu:

- 1 – bolo stlačené tlačidlo „OK“
- 2 – bolo stlačené tlačidlo „Cancel“
- 3 – bolo stlačené tlačidlo „Abort“
- 4 – bolo stlačené tlačidlo „Retry“
- 5 – bolo stlačené tlačidlo „Ignore“
- 6 – bolo stlačené tlačidlo „Yes“
- 7 – bolo stlačené tlačidlo „No“

```

dim tlacidlo as integer
tlacidlo=msgbox("Čo mám urobiť?", 290,"Otázka na používateľa")
if tlacidlo=3 then
    ' prerušiť
elseif tlacidlo=4 then
    ' skúsiť znovu
else
    ' ignorovať
end if

```

Ako vidíme, pomocou správ môžeme veľmi jednoducho riadiť niektoré makrá, napríklad už spomínané nezalomiteľné medzery za spojkami „a“. O tom si však budeme hovoriť až nabadúce.

Nezalomiteľné medzery

V minulom dieli sme si podrobne rozobrali možnosti funkcie „msgbox“, pomocou ktorej vypisujeme správy. Táto funkcia je úplne postačujúca na to, aby sme ju použili aj pri rozhodovaní, či budeme alebo nebudeme za spojku „a“ vkladat' nezalomiteľnú medzeru. Pretože za verzálku „A“ sa vkladá nezalomiteľná medzera vždy, musíme zmeniť spôsob vyhľadávania spojok tak, že budeme rozoznávať veľké a malé písmená. Najjednoduchšie riešenie môže vyzerat' potom napríklad takto:

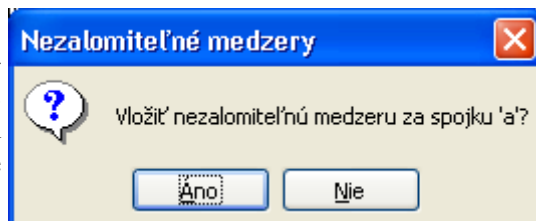
```

REM Makro pre vkladanie nezalomiteľných medzier za predložky a spojky
function f_Nezalomitelne_Spojky as long
    dim spojky as string
    dim tlacidlo as integer

    tlacidlo=msgbox("Vložit' nezalomiteľnú medzeru za spojku 'a'?", 36,"Nezalomiteľné medzery")
    if tlacidlo=6 then ' Yes
        spojky="\<(A|a|I|i|K|k|O|o|S|s|U|u|V|v|Z|z)"
    else ' No
        spojky="\<(A|I|i|K|k|O|o|S|s|U|u|V|v|Z|z)"
    endif
    f_Nezalomitelne_Spojky=Vymen_hladanim(spojky,"", NM, true, false, true)
end function

```

Toto riešenie má však svoje nedostatky – otázka je položená priamo vo funkcii pre vyhľadávanie, takže musíme na ňu odpovedať vždy. Toto nám však nemusí vždy vyhovovať, takže riadenie nezalomiteľnej medzery za spojkou „a“ budeme robiť radšej pomocou parametra:



Obrázok 5: Nastavenie parametra pre vkladanie nezalomiteľnej medzery

```
REM Makro pre vkladanie nezalomiteľných medzier za predložky a spojky
function f_Nezalomitelne_Spojky(spojka_a as boolean) as long
REM Parameter spojka_a určuje, či sa bude (TRUE) alebo nebude (FALSE)
REM vkladat' nezalomiteľná medzera za spojku „a“
dim spojky as string
```

```
if spojka_a then ' Yes
    spojky="\<(A|a|I|i|K|k|O|o|S|s|U|u|V|v|Z|z)"
else ' No
    spojky="\<(A|I|i|K|k|O|o|S|s|U|u|V|v|Z|z)"
endif
f_Nezalomitelne_Spojky=Vymen_hladanim(spojky,"", NM, true, false, true)
end function
```

A vlastnú otázku budeme klásť v procedúrach pre formátovanie textu:

```
REM Makro pre vkladanie nezalomiteľných medzier za predložky a spojky
sub Nezalomitelne_Spojky
dim tlacidlo as integer
Formatovanie_funkcie.Init
tlacidlo=msgbox("Vložit' nezalomiteľnú medzeru za spojku 'a'?", 36,"Nezalomiteľné
medzery")
    msgbox("Vložených"+Formatovanie_funkcie.f_Nezalomitelne_Spojky(tlacidlo=6)+"nezal
omiteľných medzier.", 0,"Nezalomiteľné spojky a predložky")
end sub
```

```
REM Makro pre celkové formátovanie dokumentu
sub Formatuj_Dokument
dim kolko as long
dim tlacidlo as integer
Formatovanie_funkcie.Init
tlacidlo=msgbox("Vložit' nezalomiteľnú medzeru za spojku 'a'?", 36,"Nezalomiteľné
medzery")
    kolko=Formatovanie_funkcie.f_Rozne_Zameny()
    kolko=kolko+Formatovanie_funkcie.f_P_Viacnasobna_Medzera()
    kolko=kolko+Formatovanie_funkcie.f_Zmaz_Interpunkcia_Medzera()+Formatovanie_fun
kcie.f_Vloz_Interpunkcia_Medzera()
    kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Datумы()+Formatovanie_funkcie.f
_Nezalomitelne_Jednociferne_Cisla()+Formatovanie_funkcie.f_Nezalomitelne_Telefonne_Ci
sla()+Formatovanie_funkcie.f_Nezalomitelne_Merne_Jednotky()
```

```

kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Tituly()
kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Spojky(tlacidlo=6)
msgbox("Prevedných"+kolko+"náhrad.", 0,"Formátovanie textu")
end sub

```

Pravdaže, môžeme si naprogramovať aj procedúry pre vkladanie medzier za spojky a predložky bez kladenia otázky, ktoré podľa potrieb môžeme priamo volať:

REM Makro pre vkladanie nezalomiteľných medzier za predložky a spojky vrátane spojky „a“

```

sub Nezalomitelne_Spojky_s_a
dim tlacidlo as integer
Formatovanie_funkcie.Init
msgbox("Vložených"+Formatovanie_funkcie.f_Nezalomitelne_Spojky(TRUE)+"nezalomiteľných medzier.", 0,"Nezalomiteľné spojky a predložky")
end sub

```

REM Makro pre vkladanie nezalomiteľných medzier za predložky a spojky s výnimkou spojky „a“

```

sub Nezalomitelne_Spojky_bez_a
dim tlacidlo as integer
Formatovanie_funkcie.Init
msgbox("Vložených"+Formatovanie_funkcie.f_Nezalomitelne_Spojky(FALSE)+"nezalomiteľných medzier.", 0,"Nezalomiteľné spojky a predložky")
end sub

```

Možnosti programovacieho jazyka StarOffice Basic nám však ponúkajú aj iné – zaujímavejšie možnosti. Jednou z nich je tá, že môžeme zadať tzv. voliteľné parametre, t. j. také, ktoré môžeme, ale nemusíme zadať. V definícii parametrov ich zadávame tak, že pred nimi uvedieme kľúčové slovo „Optional“:

```
function meno (parameter_1 as typ1,..., Optional paramater_x as typ_x)
```

V takomto prípade musíme, pochopiteľne, vo funkcii vyriešiť prípad, že tento parameter nebol zadáný a v takom prípade mu priradíme nami požadovanú štandardnú hodnotu. Na zistenie, či parameter je alebo nie je zadný nám slúži logická systémová funkcia „IsMissing“, ktorá vráti hodnotu TRUE, ak parameter nie je zadáný. Makro pre vkladanie nezalomiteľných medzier za spojky potom môže vyzeráť napríklad takto (predpokladáme, že štandardne chceme za spojku „a“ vkladať nezalomiteľnú medzeru):

```

REM Makro pre vkladanie nezalomiteľných medzier za predložky a spojky
function f_Nezalomitelne_Spojky(optional spojka_a as boolean) as long
REM Parameter spojka_a určuje, či sa bude (TRUE) alebo nebude (FALSE)
REM vkladať nezalomiteľná medzera za spojku „a“
REM Ak tento parameter nie je zadáný, potom štandardná hodnota je TRUE
dim spojky as string

```

```
REM ak parameter spojka_a nie je zadaný, niečím ho musíme nahradiť
dim pom_spojka_a as boolean
```

```
if not IsMissing(spojka_a) then ' Parameter je zadaný
    pom_spojka_a=spojka_a
else ' Parameter nie je zadaný
    pom_spojka_a=TRUE
endif
```

```
if pom_spojka_a then ' Yes
    spojky="\<(A|a|I|i|K|k|O|o|S|s|U|u|V|v|Z|z)"
else ' No
    spojky="\<(A|I|i|K|k|O|o|S|s|U|u|V|v|Z|z)"
endif
f_Nezalomitelne_Spojky=Vymen_hladanim(spojky,"", NM, true, false, true)
end function
```

Kto požaduje, aby mal štandardnú hodnotu opačnú, nech si svoje makro v rámci domácej úlohy príslušne upraví. Následne môžeme naprogramovať aj príslušné procedúry pre volanie tohto makra napríklad takto:

```
REM Makro pre vkladanie nezalomiteľných medzier za predložky a spojky
REM Toto makro použijeme vtedy, ak si chceme vybrať, či za spojku „a“ budeme
REM alebo nebudeme vkladať nezalomiteľné medzery
sub Nezalomitelne_Spojky
    dim tlacidlo as integer
    Formatovanie_funkcie.Init
    tlacidlo=msgbox("Vložiť nezalomiteľnú medzeru za spojku 'a'?", 36,"Nezalomiteľné medzery")
    msgbox("Vložených"+Formatovanie_funkcie.f_Nezalomitelne_Spojky(tlacidlo=6)+"nezalomitelnych medzier.", 0,"Nezalomiteľné spojky a predložky")
end sub
```

```
REM Makro pre celkové formátovanie dokumentu
REM V tomto makre budeme používať štandardnú hodnotu pre riadenie vkladania
REM nezalomiteľnej medzery za spojku „a“
sub Formatuj_Dokument
    dim kolko as long
    Formatovanie_funkcie.Init
```

```

kolko=Formatovanie_funkcie.f_Rozne_Zameny()
kolko=kolko+Formatovanie_funkcie.f_P_Viacnashna_Medzera()
kolko=kolko+Formatovanie_funkcie.f_Zmaz_Interpunkcia_Medzera()+Formatovanie_funkcie.f_Vloz_Interpunkcia_Medzera()
kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Datумы()+Formatovanie_funkcie.f_Nezalomitelne_Jednociferne_Cisla()+Formatovanie_funkcie.f_Nezalomitelne_Telefonne_Cisla()+Formatovanie_funkcie.f_Nezalomitelne_Merne_Jednotky()
kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Tituly()
kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Spojky()
msgbox("Prevedných"+kolko+"náhrad.", 0,"Formátovanie textu")
end sub

```

Ako vidíme, procedúra Formatuj_Dokument sa vrátila k podobe, v akej sme ju používali doteraz.

Práca so súbormi

V minulom dieli sme si ukázali niekoľko možností, ako môžeme využiť správy pre riadenie programu. Doterajšie riešenia však mali jednu nevýhodu – buď sme naprogramovali dve makrá, alebo sme pri každom spustení zadávali čo chceme vlastne urobiť. Omnoho zaujímavejšie je však urobiť niektoré makrá tak, aby si každý používateľ mohol v inom (nazvime ho inicializačnom) podprograme nastaviť také parametre, aké chce a potrebuje. Ostatné makrá sa potom, pochopiteľne, správajú podľa týchto nastavení. Pre praktickú stránku veci je však potrebné, aby sa tieto nastavenia uložili, t.j., aby sme ich nemuseli znovu zadávať pri novom spustení OpenOffice.org.

Na to, aby sme si nastavenia pamätali aj po skončení nejakého programu ich musíme uložiť do nejakého pracovného súboru. Pretože v OpenOffice.org je v tejto oblasti najjednoduchšie použitie textových súborov, zameriame sa na prácu s nimi.

Pre prístup k súborom musíme použiť niekoľko štandardných príkazov a funkcií. Samozrejme, ako prvá nás bude zaujímať funkcia „Open“, pomocou ktorej dokážeme súbor nielen otvoriť, ale aj vytvoriť:

```

Open Meno_súboru [For Mód] [Access Typ_prístupu] [Ochrana] As #Číslo_súboru
[Len=Dĺžka_záznamu]

```

Jednotlivé parametre majú nasledovný význam:

Meno_súboru je reťazec, ktorý obsahuje názov súboru vrátane prístupovej cesty. Pri názvoch súborov sa používa URL formát, t.j. „file:///...“. Pre prípadný prevod medzi systémovým a URL formátom môžeme použiť funkciu ConvertToUrl(systémový_názov), alebo naopak pre opačný prevod funkciu ConvertFromURL(URL_názov).

Za kľúčovým slovom **For** môžeme určiť režim práce so súborom, pričom môžeme zadať jednu z nasledujúcich hodnôt:

Append – rozšírenie existujúceho súboru.

Binary – k dátam budeme pristupovať binárne pomocou funkcií Put a Get.

Input – otvorenie existujúceho súboru pre čítanie. Pokus o otvorenie neexistujúceho súboru vedie ku chybe.

Output – vytvorenie nového súboru (alebo prepísanie existujúceho súboru) pre zápis.

Random – úprava relatívnych súborov.

Za kľúčovým slovom **Access** určujeme prístup k súborom, pričom môžeme zadať jednu z nasledujúcich hodnôt:

Read – iba pre čítanie.

Write – iba pre zápis.

Read Write – pre čítanie aj zápis.

Ako ďalší parameter môže byť **spôsob ochrany** zdieľania súboru inými aplikáciami, pričom tu môžeme zadať jednu z nasledujúcich hodnôt:

Shared – súbor môžu otvoriť aj iné aplikácie.

Lock Read – súbor je chránený proti čítaniu.

Lock Write – súbor je chránený proti zápisu.

Lock Read Write – zabránenie prístupu k súboru.

Za znakom „**mreža**“ nasleduje číslo súboru, t.j. dátového kanálu, na ktorom budeme k nemu pristupovať. Môže nadobudnúť hodnotu od 0 do 511 a toto číslo získame tesne pred otvorením súboru pomocou funkcie FreeFile. Toto číslo je dôležité, pretože odteraz budeme k súboru pristupovať už iba pomocou neho.

V poslednom parametri „**Len**“ môže byť určená dĺžka záznamu, ktorú používame v relatívnych súboroch.

Textové súbory

V našich makrách sa budeme teraz zaoberať iba jednoduchými textovými súbormi a preto si ukážeme prácu iba s nimi. Ostatné typy (napr. binárne súbory) a možnosti (napr. zdieľanie) sú určené pre skúsenejších programátorov. Textové súbory môžeme vytvoriť napríklad takto:

```
dim subor_nastaveni as string
```

```
dim subor as integer
```

```
subor_nastaveni="file:///c:/Program files/OpenOffice.org 2.0/Formatuj.dat"
```

```
subor = Freefile
```

```
open subor_priznakov for Output as #subor
```

A následne ich môžeme otvoriť pre načítanie zapamätaných údajov takto:

```
dim subor_nastaveni as string
```

```
dim subor as integer
```

```
subor_nastaveni="file:///c:/Program files/OpenOffice.org 2.0/Formatuj.dat"
```

```
subor = Freefile
```

```
open subor_priznakov for Input as #subor
```

Pre zapisovanie do textových súborov sa používa funkcia „**Print**“ v nasledovnom tvare:

Print #Číslo_súboru, dáta

Pre čítanie už uložených údajov budeme používať funkciu „**Line input**“:

Line Input #Číslo_súboru, reťazcová_premenná

Nakoniec práce je potrebné otvorený súbor zatvoriť, čo urobíme pomocou príkazu „**Close**“:

Close #Číslo_súboru

Na základe tých poznatkov by sme teraz mohli naprogramovať procedúry pre načítavanie a ukladanie našich nastavení, ale to si ukážeme až nabudúce.

Inicializačné súbory

V dnešnom pokračovaní seriálu o programovaní makier v OpenOffice.org si ukážeme, ako vyriešime ukladanie nastavení do „inicializačného“ súboru s tým, že automaticky dokážeme vyriešiť prípad aj jeho prípadného poškodenia či úplnej absencie.

Minule sme si uviedli krátky prehľad základných funkcií, ktoré potrebujeme pri práci so súbormi. Ešte pred tým, ako si naprogramujeme vlastné procedúry pre prácu s nimi si musíme uviesť niekoľko požiadavok a podmienok, ktoré budeme na ne klásť.

Pri práci so súbormi musíme mať na pamäti, či súbor, s ktorým chceme pracovať vôbec existuje a ak neexistuje, tak kto a ako ho vytvorí. Pretože inicializačné súbory v našom prípade budeme vytvárať interne (t.j. iba pomocou nami naprogramovaných podprogramov), musíme predpokladať, že tieto súbory ešte vôbec neexistujú.

Pre praktickú stránku vecí je navyše vhodné, aby sme testovali nielen ich existenciu, ale údaje (počet a typ), ktoré sú v nich uložené. Vďaka tomuto riešeniu totiž môžeme tieto súbory v ďalších verziách makier modifikovať (t.j. meniť počet údajov, ktoré si v nich pamätáme) bez toho, aby sme ich museli pred tým manuálne vymazať. Pravdaže, týmto riešením sa vyhneme aj prípadných iným chybám, ku ktorým by mohlo dôjsť v prípade čo aj nechceného poškodenia týchto súborov.

Pre zistenie, či súbor existuje použijeme štandardnú funkciu „FileExists(meno_súboru)“, ktorá vracia logickú hodnotu True, ak súbor so zadaným menom existuje alebo logickú hodnotu False, ak neexistuje. Pokiaľ zohľadníme predchádzajúce pripomienky, základná verzia príslušných programov pre prácu so súbormi môže vyzeráť napríklad takto:

REM Pretože meno súboru budeme potrebovať na viacerých miestach, nastavíme si ho iba raz napr. vo forme funkcie

function Subor_nezalomitelne_predlozky as string

REM Funkcia vracia názov inicializačného súboru

Subor_nezalomitelne_predlozky="file:///c:/Program files/OpenOffice.org 2.0/Nezalomitelne_Predlozky.dat"

end function

REM Procedúra pre vytvorenie/uloženie nastavenia

sub Uloz_nastavenie_predlozky(male_a as boolean)

REM Procedúra vytvorí inicializačný súbor, kde uloží príslušné nastavenia

dim subor as integer

```

subor = Freefile
open Subor_nezalomitelne_predlozky for Output as #subor
' REM Funkcia PRINT zapíše do súboru slovo „True“ alebo „False“
print #subor, male_a
close #subor
end sub

function Citaj_nastavenie_predlozky as boolean
dim i, subor as integer
dim riadok as string
dim male_a as boolean

    REM Tu si nastavíme štandardnú hodnotu pre prípad poškodenia alebo absencie
    inicializačného súboru
    male_a=True

    ' Ak nastavovací súbor neexistuje, vytvoríme ho a uložíme do neho štandardné hodnoty
    if not fileexists(Subor_nezalomitelne_predlozky) then
        Uloz_nastavenie_predlozky(male_a)
    endif

    ' Teraz súbor určite existuje - načítame ho
    subor = Freefile
    open Subor_nezalomitelne_predlozky for Input as #subor
    i=0 ' Počítadlo načítaných riadkov
    ' Ak nie sme na konci súboru, tak môžeme načítať uloženú informáciu
    if not eof(subor) then
        i=i+1
        Line Input #subor, riadok
    endif
    close #subor

    if i<1 then
        ' V súbore nebol správny počet údajov – prepíšeme ho na štandardné hodnoty
        Uloz_nastavenie_predlozky(male_a)
    else

```

```

' Počet údajov je správny, ešte otestujeme ich správnosť
select case riadok
case "True"
male_a=True
case "False"
male_a=False
case else
' Údaj je poškodený – nie je to logická hodnota a preto súbor opravíme
Uloz_nastavenie_predlozky(male_a)
end select
endif
Citaj_nastavenie_predlozky=male_a
end function

```

Vlastné makro pre vkladanie nezalomiteľných medzier za predložky môže teraz vyzerat' takto:

```

function f_Nezalomitelne_Spojky as long
dim spojky as string

if Citaj_nastavenie_predlozky then
spojky="\<(A|a|I|i|K|k|O|o|S|s|U|u|V|v|Z|z) "
else
spojky="\<(A|I|i|K|k|O|o|S|s|U|u|V|v|Z|z) "
endif
f_Nezalomitelne_Spojky=Vymen_hladanim(spojky," ", NM, true, false, true)
end function

```

A nakoniec si ešte musíme napísať procedúru pre vlastné zadanie potrebného nastavenia:

```

sub Nastav_parameter_pre_medzery
dim tlacidlo as integer
tlacidlo=msgbox("Vložiť nezalomiteľnú medzeru za spojku 'a'?", 36,"Nezalomiteľné medzery")

if tlacidlo=6 then ' Yes
Uloz_nastavenie_predlozky(True)
else ' No
Uloz_nastavenie_predlozky(False)
endif
end sub

```

Vstupné funkcie

Ako sme už spomínali, pre interaktívne zadávanie údajov máme v OpenOffice.org tri základné možnosti – správy, vstupné funkcie a dialógy. O správach sme si toho popísali v predchádzajúcich štyroch dieloch už dosť a preto sa teraz zameriame na vstupné funkcie.

Pri programovaní makier v OpenOffice.org máme k dispozícii iba jednu vstupnú funkciu „InputBox“, ktorá vracia zadaný reťazec. Táto funkcia má tri vstupné parametre:

InputBox("Nápovedný text", "Nadpis okna", "Štandardná hodnota")

Hoci táto funkcia dokáže vrátiť iba reťazec, je to úplne postačujúce, pretože prípadný prevod na iný typ premenných je v jazyku StarOffice Basic vyriešený veľmi jednoducho. Jednotlivé parametre nebudeme teraz teoreticky rozoberať, ale v niekoľkých dieloch si na konkrétnych príkladoch radšej ukážeme praktické použitie tejto funkcie. Myslím si, že z týchto príkladov pochopí ich význam každý bez osobitnej potreby vysvetľovania.

Spočítavame textové reťazce

Ako prvý príklad, v ktorom vynikajúco využijeme možnosti funkcie „InputBox“ je makro pre spočítanie počtu výskytov textových reťazcov. Každý z nás má svoje „oblúbené“ výrazy (napr. u mňa je to slovo „samozrejme“ vrátane jeho synonym), ktoré, chtiac-nechtiac, prenáša aj do písomného prejavu. Pravdaže (a znovu je tu moje „samozrejme“ – teraz v slove „pravdaže“), toto môže prekážať a preto ako dobrá pomôcka môže byť zistenie ich počtu. Pre vlastné spočítanie by sme mohli použiť „fintu“ náhrady toho istého reťazca za ten istý, pričom by sme zistili počet týchto „výmien“. Takéto riešenie je však veľmi nepraktické, pretože v prípade, že máme príslušný výraz napísaný raz veľkými písmenami, inokedy malými a inokedy zas iba s veľkým písmenom na začiatku, príslušná výmena by toto formátovanie zrušila a všetko by sa nahradilo rovnakým textom. Preto je vhodnejšie, ak si na zistenie vlastného počtu urobíme vlastnú funkciu „Spocitaj_hladanim“. Na to nám stačí veľmi jednoducho upraviť napr. funkciu „Vymen_hladanim“, kde vynecháme tie časti, v ktorých nahradzame nájdený reťazec, a výsledok môže vyzeráť napríklad takto:

REM Funkcia spočíta počet výskytov reťazca Co

function Spocitaj_hladanim(Co as string, Regularne, Cele_slova, Velke_pismena as Boolean) as long

dim nasiel as object

dim kolko as long

Hladaj.searchString=Co

Hladaj.SearchRegularExpression=Regularne

Hladaj.SearchWords=Cele_slova

Hladaj.SearchCaseSensitive=Velke_pismena

kolko=0

nasiel = Dokument.findFirst(Hladaj)

Do While NOT isNull(nasiel)

if len(nasiel.String)>0 then

kolko=kolko+1

end if

```
nasiel = Dokument.findNext( nasiel.End, Hladaj)
```

```
Loop
```

```
Spocitaj_hladanim=kolko
```

```
end function
```

Vlastná procedúra pre zadanie hľadaného reťazca a následné spočítanie jeho výskytu potom môže vyzeráť napríklad takto (predpokladajme, že funkciu „Spocitaj_hladanim“ sme uložili do modulu „Formatovanie_funkcie“):

```
sub Spocitaj_slova
```

```
dim slovo as string
```

```
dim kolko as long
```

```
slovo= inputbox("Zadajte reťazec, ktorý chcete spočítať:", "Počet výskytov reťazca", "")
```

```
Formatovanie_funkcie.Init
```

```
kolko=Formatovanie_funkcie.Spocitaj_hladanim(slovo,False,True,False)
```

```
msgbox("Reťazec „"+slovo+"“ sa vyskytuje "+kolko+" krát.",0,"Počet výskytov reťazca")
```

```
end sub
```

Pri takejto práci je však pohodlnejšie, ak si označíme hľadaný reťazec do bloku, a makro nám ho následne automaticky ponúkne ako štandardnú hodnotu, ktorú stačí iba „odklepnúť“. Na tento účel využijeme systémovú objektovú premennú „getCurrentSelection“, pomocou ktorej získame prístup k označenému textu. Pochopiteľne, v makre musíme pamätať aj na to, že žiadny text neoznačíme. Výsledné makro potom môže vyzeráť takto:

```
sub Spocitaj_slova
```

```
dim slovo, oznacene as string
```

```
dim kolko as long
```

```
dim dokument, vyber as object
```

```
dokument=thisComponent ' aktuálny dokument
```

```
vyber=dokument.getCurrentSelection() ' aktuálny vyber
```

```
oznacene="" ' V prípade, že nebude označený žiadny text, bude štandardný text prázdny
```

```
' Ak je počet označených častí nenulový
```

```
if vyber.getCount()>0 then
```

```
' Vyberieme označený reťazec a orežeme ho o okrajové medzery
```

```
oznacene=trim(vyber.getByIndex(0).getString())
```

endif

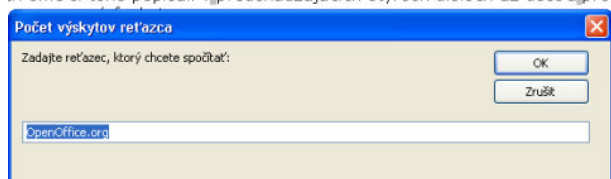
```
slovo= inputbox("Zadajte reťazec, ktorý chcete spočítať:", "Počet výskytov  
reťazca", oznacene)
```

Formatovanie_funkcie.Init

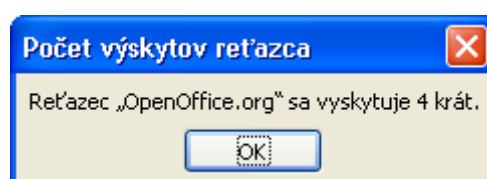
```
kolko=Formatovanie_funkcie.Spocitaj_hladanim(slovo, False, True, False)
```

```
msgbox("Reťazec „"+slovo+"“ sa vyskytuje "+kolko+" krát.", 0, "Počet výskytov reťazca")  
end sub
```

ali v časti „Makrá v OpenOffice.org 3/1. – Správy“, pre interaktívne
iáme v OpenOffice.org tri základné možnosti – správy, vstupné funkcie
ch sme si toho popisali v predchádzajúcich štyroch dieloch už dosť aľpret



Obrázok 6: Označený text je ponúknutý ako štandardná hodnota ...



Obrázok 7: ... a následne spočítaný

Prevod čísla na text

Ako sme si sľúbili minule, dnes si naprogramujeme makro pre prevod čísla na text. Skôr, ako sa pustíme do jeho písania, musíme previesť podrobnejšiu analýzu tejto problematiky, pričom sa budeme opierať o pravidlá slovenského pravopisu (vzhľadom na podobnosť jazyka si čitatelia z Českej republiky následne určite dokážu prispôsobiť tieto podmienky na český pravopis).

Z hľadiska programovania môžeme číselky rozdeliť do nasledujúcich skupín:

Číselky do 10 – nula, jedna, ..., deväť.

Číselky od 10 do 20 – desať, jedenásť, ..., devätnásť.

Číselky od 20 do 50 (prípona „dsať“) – dvadsať, tridsať, štyridsať.

Číselky od 50 do 100 (prípona „desiat“) – päťdesiat, šesťdesiat, ..., deväťdesiat.

Stovky – sto, dvesto, ..., deväťsto.

Vyššie rády – tisíc, dvamilióny, päťmimiliárd, ...

Pozrime sa teraz na jednotlivé skupiny trochu podrobnejšie. Pri číselkách do desať sa musíme osobitne zamerať na číselky jedna a dva. Obidve sa skloňujú podľa toho, pred čím sú uvedené – jedenmilión, jednamiliarda, dvamilióny, dvemiliardy. Číselka jedna a dva sa však neskloňuje v zložených číslach (t.j. dvadsaťjedenmiliónov, dvadsaťjedenmiliárd, tridsaťdvamiliárd) a okrem toho sa číselka jedna zvyčajne neuvádza pred stovkami a vyššími rádmami, t.j. píšeme obvykle sto, tisíc a nie jednosto, jedentisíc. Druhý tvar (jednosto, jedenmilión) sa však požaduje v tých prípadoch, keď je dôležité, aby sa nemohlo napísané číslo sfaľšovať, t.j. napr. pri peňažných poukázkach. V rámci tejto skupiny ešte musíme spomenúť číslo nula, ktoré sa vypisuje iba vtedy, ak je celé prevádzané číslo 0.

Ďalej sa musíme zamerať na vyššie rádové prípony. Tieto môžeme rozdeliť na dve základné skupiny – „lióny“ (milión, bilión, ...) a „liardy“ (miliarda, biliarda, ...). Tieto sa skloňujú podľa toho, koľko ich je – jedenmilión, dvamilióny, päťmiliónov a jednamiliarda, dvemiliardy, päťmiliárd. Ak si všimneme, tieto prípony sa skladajú z dvoch častí. Prvá časť „mi“, „bi“, „tri“, ... vyjadruje vlastný rád a druhá „lión“, „lónov“, „liárd“, ... sa skloňuje podľa množstva (ospravedlňujem sa jazykovedcom za prípadné nesprávne termíny, ale takto sa mi najlepšie vyjadruje problematika, ktorú ideme programovať).

Hoci nie som odborník na jazyky, musím na tomto mieste upozorniť na niektoré väčšie rozdiely medzi slovenčinou a češtinou, aby si aj českí čitatelia mohli naprogramovať svoje verzie makier. Pri číslach od 20 do 100 stačí v slovenčine pridať prípony „dsat“ (dvadsať) a „desiat“ (päťdesiat), kým v češtine to platí iba pre príponu „cet“ (dvacet) a ostatné desiatkové číslice (s príponou „desát“) sa musia vymenovať osobitne (padesát, šedesát, ...). Pri stovkách sú rozdiely medzi slovenčinou a češtinou ešte väčšie, pretože v slovenčine existuje iba prípona „sto“ (jednosto, päťsto) kým v češtine sú až štyri – „sto“ (jednosto), „stě“ (dvěstě), „sta“ (třista), „set“ (pětset). Podobný rozdiel je aj v tisícoch, pretože v slovenčine sa používa iba prípona „tisíc“ (dvetisíc, päťtisíc), kým v češtine dve – „tisíc“ (jedentisíc, pětisíc) a „tisíce“ (dvatisíce).

Ako vidíme, problematika prevodu čísla na text je pomerne zložitá. Pri vlastnom prevode čísla je najvhodnejší ten postup, že sa číslo analyzuje po troch čísliciach, pretože tak sa menia rádové prípony (tisíc, milión, ...). Aby sme nestratili potrebný prehľad, naprogramujeme si najprv pre jednotlivé skupiny (číslvky do 10, číslvky od 10 do 100, ...) osobitné podprogramy:

```
function slovne_jedno_cislo(cislo, rad as integer) as string
dim pom_nazov as string
pom_nazov=""
select case cislo
case 1
' Číslvka „jeden“ sa pri rádoch tisíc, milión,... nepíše
if rad=0 then
pom_nazov="jeden"
endif
case 2
' Dvojka sa skloňuje - dva, dvesto, dvetisíc, dvamilióny, dvemiliardy, ...
if (rad mod 2)=0 then
pom_nazov="dva"
else
pom_nazov="dve"
endif
case 3
pom_nazov="tri"
case 4
pom_nazov="štyri"
```

```

case 5
    pom_nazov="päť"
case 6
    pom_nazov="šesť"
case 7
    pom_nazov="sedem"
case 8
    pom_nazov="osem"
case 9
    pom_nazov="deväť"
end select
slovne_jedno_cislo=pom_nazov
end function

```

```

function slovne_desiatky(jednotky, desiatky as integer) as string
dim pom_nazov as string
pom_nazov=""
select case desiatky
case 1
    ' desať – devätnásť
select case jednotky
case 0
    pom_nazov="desať"
case 1
    pom_nazov="jedenásť"
case 2
    pom_nazov="dvanásť"
case 3
    pom_nazov="trinásť"
case 4
    pom_nazov="štrnásť"
case 5
    pom_nazov="pätnásť"
case 6
    pom_nazov="šestnásť"
case 7

```

```

    pom_nazov="sedemnást"
case 8
    pom_nazov="osemnášt"
case 9
    pom_nazov="devätnášt"
end select
case 2 to 4
    ' "dsat" – dvadsať až štyridsať
    pom_nazov=slovne_jedno_cislo(desiatky,0)+"dsat"+slovne_jedno_cislo(jednotky,0)
case else
    ' "desiat" – päťdesiat až deväťdesiat
    pom_nazov=slovne_jedno_cislo(desiatky,0)+"desiat"+slovne_jedno_cislo(jednotky,0)
end select
slovne_desiatky=pom_nazov
end function

```

```

function slovne_stovky(cislo, rad as integer) as string
dim stovky, desiatky, jednotky as integer
dim pom_slovne as string
' z čísla vypreparujeme číslice na mieste jednotiek, desiatok a stoviek
jednotky=cislo mod 10
desiatky=int(cislo/10) mod 10
stovky=int(cislo/100)

pom_slovne=""

' Najprv prevedieme na text jednotky a desiatky (číslo 0 až 99)
if desiatky=0 then
    ' iba jednotky (čísla 0 až 9)
    if stovky=0 then
        ' celá trojica je číslo 0 až 9 a vtedy sa číslovka 1 a 2 skloňuje podľa rádu
        pom_slovne= slovne_jedno_cislo(jednotky,rad)
    else
        ' je to zložené číslo a vtedy sa číslovka 1 a 2 nesklonuje (základný tvar je pre rád 0)
        pom_slovne= slovne_jedno_cislo(jednotky,0)
    endif
endif

```

```

else
' je to číslo od 10 do 99
pom_slovne=slovne_desiatky(jednotky, desiatky)
endif

' teraz k prevedenému dvojčifernému číslu pridáme text o stovkách
select case stovky
case 1
' Pri stovkách sa jednoto píše iba finančne
pom_slovne="sto"+pom_slovne
case 2 to 9
' pridávame príponu "sto" – dvesto až deväťsto
' číslovka 2 sa pri stovkách skloňuje ako pri tisícoch (dvesto – dvetisíc). t.j. tvar pre rád 1
pom_slovne=slovne_jedno_cislo(stovky,1)+"sto"+pom_slovne
end select
slovne_stovky=pom_slovne
end function

```

```

function slovne_rady(cislo, rad as integer) as string
' Texty o biliónoch, ... sú teoretické, pretože premenná typu LONG je maximálne
2.147.483.647, t.j.
' dvemiliardystoštyridsaťsedemmiliónovštyristoosemdesiatritisícšesťstoštyridsaťsedem
dim pom_kolko as string
dim predpona as string
predpona=""
pom_kolko=""
select case rad
case 1
predpona="tisíc"
case 2, 3
predpona="mi"
case 4, 5
predpona="bi"
case 6, 7
predpona="tri"
case 8, 9

```

```

    predpona="kvadri"
end select

if rad=1 then ' tisíce
select case cislo
case 0
    pom_kolko=""
case else
    pom_kolko=predpona
end select
else
if (rad mod 2)=0 then ' „líóny“ – milión, bilióny, triliónov
select case cislo
case 0
    pom_kolko=""
case 1
    pom_kolko=predpona+"lión"
case 2 to 4
    pom_kolko=predpona+"líóny"
case else
    pom_kolko=predpona+"liónov"
end select
else ' „liardy“ – miliarda, biliardy, triliárd
select case cislo
case 0
    pom_kolko=""
case 1
    pom_kolko=predpona+"liarda"
case 2 to 4
    pom_kolko=predpona+"liardy"
case else
    pom_kolko=predpona+"liárd"
end select
endif
endif
slovne_rady=pom_kolko

```

end function

Ako poslednú si teraz naprogramujeme funkciu pre vlastný prevod celého čísla. V prvej verzii nebudeme zatiaľ uvažovať o „finančných“ číslach, t.j. tvare „jedosto“, „jedentisíc a pod.

```
function daj_slovne(cislo as Long) as string
dim rad as integer ' rady: 1 – tisíc, 2 – milion, 3 – miliarda
dim slovne as string
dim analyza as long

slovne=""
if cislo=0 then
' Osobitne musíme ošetriť číslo „nula“
slovne="nula"
else
rad=0
do while cislo > 0
' z čísla vyberieme posledné tri číslice
analyza=cislo mod 1000
' ktoré prevedieme na text
slovne=slovne_stovky(analyza, rad)+slovne
' číslo zmenšíme o už spracovanú časť
cislo=int(cislo/1000)
' a pridáme text rádu, ktorý budeme spracovávať v ďalšom kroku cyklu
if cislo > 0 then
rad=rad+1
slovne=slovne_rady(cislo mod 1000, rad)+slovne
endif
loop
endif

daj_slovne=slovne
end function
```

Minule sme naprogramovali makro pre prevod čísla na text s tým, že sme zatiaľ neuvažovali o tzv. finančných číslach, t.j. tvare, keď sa číslovka jedna vypisuje aj pri stovkách či iných rádoch (jednosto, jedentisíc, ...). Toto však v mnohých prípadoch nevyhovuje a preto si uvedieme teraz makro, v ktorom je tento prípad ošetrený. Pochopiteľne, makro upravíme tak, aby sme mohli dostať obidva požadované tvary. Za týmto účelom pridáme do funkcie „daj_slovne“ druhý – nepovinný parameter, pomocou ktorého budeme určovať typ výsledku. Vlastné makro potom môže vyzeráť takto:

```

function daj_slovne(cislo as Long, optional financne as boolean) as string
dim rad as integer ' 1 – tisíc, 2 – milion, 3 – miliarda
dim slovne as string
dim pom_financne as boolean
dim analyza as long

if not IsMissing(financne) then ' Parameter typu výsledku je zadaný
    pom_financne=financne
else ' Parameter nie je zadaný
    pom_financne=FALSE ' štandardne nebudeme prevádzať na finančné číslo
endif

slovne=""

if cislo=0 then
    ' Osobitne musíme ošetriť číslo „nula“
    slovne="nula"
else
    rad=0
    do while cislo<>0
        ' z čísla vyberieme posledné tri číslice
        analyza=cislo mod 1000
        ' ktoré prevedieme na text
        slovne=slovne_stovky(analyza, rad)+slovne
        ' číslo zmenšíme o už spracovanú časť
        cislo=int(cislo/1000)
        ' a pridáme text rádu, ktorý budeme spracovávať v ďalšom kroku cyklu
        if cislo<>0 then
            rad=rad+1
            slovne=slovne_rady(cislo mod 1000, rad)+slovne
        endif
    loop

    if pom_financne then ' Úprava v prípade, že píšeme "finančné" číslovky
        ' Úprava výsledku, ak chceme finančné číslo
        if (analyza>=100) and (analyza<=199) then

```

```

' úprava výsledku pre stovky
slovne="jedno"+slovne ' jednosto...
endif
if (analyza=1) and (rad=0) then
' úprava výsledku, ak sme prevádzali na text číslo jedna, lebo
' číslovka "jeden" sa skloňuje, t.j. vo financiách sa píše "jedna koruna".
slovne="jedna"
endif
if (analyza=1) and (rad>0) then
' úprava výsledku pre ostatné rády
select case rad
case 1, 2, 4, 6, 8
slovne="jeden"+slovne ' jedentisíc, jedenmilión, ...
case 3, 5, 7, 9
slovne="jedna"+slovne ' jednamilarda, ...
end select
endif
endif
endif

daj_slovne=slovne
end function

```

Teraz si ukážeme, ako budeme toto makro volať tými najrôznejšími spôsobmi. Podobne, ako sme to urobili pri makrách pre formátovanie textu, aj teraz si rozdelíme podprogramy do rôznych modulov. Už naprogramované funkcie uložíme do modulu „Cislo_Slovom“ a nasledujúce podprogramy do modulu „Module1“ v zložke „Standart“. Robíme to z toho dôvodu, že pri volaní makier v Calcu je potrebné, aby sa tieto nachádzali v tejto zložke.

Pretože si teraz predstavujeme vstupnú funkciu „InputBox“, ukážeme si túto možnosť ako prvú s tým, že prevedené číslo ihneď vložíme do písaného textu:

```

sub Vloz_cislo_slovom
dim cislo_s as string
dim cislo as long
dim dokument, dispecer as object
dim argumenty(0) as new com.sun.star.beans.PropertyValue

cislo_s= inputbox("Zadajte celé číslo, ktoré chcete vložiť slovom:", "Vloženie čísla slovom", "")
if cislo_s<>"" then

```

```

' ak vôbec zadáme nejaký údaj, tak ho prevedieme na text
cislo=val(cislo_s)
dokument=ThisComponent.CurrentController.Frame
dispecer=createUnoService("com.sun.star.frame.DispatchHelper")
argumenty(0).Name = "Text"
argumenty(0).Value = Cislo_Slovom.Daj_Slovne(cislo)
' a vložíme ho do textu na aktuálnu pozíciu
dispecer.executeDispatch(dokument, ".uno:InsertText", "", 0, argumenty())
endif
end sub

```

Vstupy priamo z dokumentu

Ešte praktickejšie riešenie je však také, že si napíšeme číslo priamo do dokumentu, označíme ho do bloku a napr. po stlačení niektorej klávesovej skratky, ku ktorej si priradíme nasledujúce makro ho ihneď prepíšeme jeho slovným znením:

```

sub Vloz_cislo_slovom
dim cislo_s as string
dim cislo as long
dim dokument, vyber, dispecer as object

dim argumenty(0) as new com.sun.star.beans.PropertyValue

dokument=thisComponent ' aktuálny dokument
vyber=dokument.getCurrentSelection() ' aktuálny vyber

' Ak je počet označených častí nenulový
if vyber.getCount()>0 then
' Vyberieme označený reťazec a orežeme ho o okrajové medzery
cislo_s=trim(vyber.getByIndex(0).getString())
if cislo_s<>"" then
' ak sme niečo označili, tak to prevedieme na text
cislo=val(cislo_s)
dokument=ThisComponent.CurrentController.Frame
dispecer=createUnoService("com.sun.star.frame.DispatchHelper")
argumenty(0).Name = "Text"
argumenty(0).Value = Cislo_Slovom.Daj_Slovne(cislo)
' a vložíme ho do textu na aktuálnu pozíciu

```

```

dispecer.executeDispatch(dokument, ".uno:InsertText", "", 0, argumenty())
endif
endif
end sub

```

Úplne najpohodlnejšie riešenie je však také, že napíšeme číslo do dokumentu a bez toho, aby sme ho označovali do bloku ho napr. po stlačení klávesovej skratky prepíšeme jeho slovným znením. Na tento účel však najprv musíme sami vyhľadať a označiť posledne vložené slovo pomocou takejto funkcie:

```

function posledne_slovo (kurzor as object) as string
' funkcia vráti posledne napísané slovo a zároveň ho označí do bloku
dim poms as string
dim slovo as boolean
dim OddelovaceSlov as string

OddelovaceSlov=" "+chr(&HA0)+chr(&H09)+chr(&H0A)+chr(&H0D)
OddelovaceSlov=OddelovaceSlov+"-./,!?;..."
OddelovaceSlov=OddelovaceSlov+"()[]{}"
poms=""
slovo=false

do while (not kurzor.isAtStartOfLine) and (not slovo)
' pokiaľ nie sme na začiatku riadku a nie sme ešte na začiatku slova
' tak sa presunieme o jeden znak doľava s tým, že označíme tento znak do bloku
kurzor.goleft(1,true)
' označené slovo si uložíme do pomocnej premennej
poms=kurzor.getstring
' ak sme našli oddelovač slov, tak sme na začiatku slova
slovo=instr(OddelovaceSlov,left(poms,1))
loop

if slovo and (len(poms)>1) then
' ak sme boli na začiatku slova (a ak sme vôbec niečo označili)
' tak musíme tento oddelujúci znak z nájdeného slova vylúčiť, t.j.
' vrátime sa o krok späť
kurzor.goright(1,true)
poms=kurzor.getstring
endif

```

```
posledne_slovo=poms  
end function
```

Vlastné makro pre vloženie slovného znenia môže potom vyzerat' takto:

```
sub Vloz_cislo_slovom  
dim kurzor, doc as object  
dim cislo as long  
dim scislo as string  
  
doc=ThisComponent.CurrentController  
kurzor=doc.getViewCursor() ' Aktuálna pozícia kurzora  
  
scislo=posledne_slovo(kurzor)  
cislo=val(scislo)  
  
if trim(str(cislo))=scislo then  
    scislo=Cislo_Slovom.Daj_Slovne(cislo)  
    kurzor.SetString(scislo) ' označené číslo prepíšeme jeho slovným znením  
endif  
  
kurzor.goright(len(scislo),false) ' nakoniec odznačíme blok (zostal označený aj po vložení  
textu)  
end sub
```

Ako ste si mohli všimnúť, funkcia pre hľadanie posledného slova ukončí jeho hľadanie, ak narazí na nejaký znak, ktorý považujeme za oddeľovač slov a preto musíme procedúru „Vloz_cislo_slovom“ volat' ihneď po napísaní príslušného čísla bez toho, aby sme za ním zadali čo i len medzeru.

Verzia pre modul Calc

Nakoniec si naprogramujeme funkciu pre vloženie slovného znenia čísla v module Calc. Pretože niekedy požadujeme, aby vložené slovo začínalo veľkým písmenom, vylepšíme ju aj o takýto nepovinný paramater:

```
function Slovom(Cislo as Long, optional Financne, optional Velke)  
dim nazov as string  
dim pom_fin, pom_velke as boolean  
  
' štandardne budeme predpokladať, že chceme finančné číslo  
pom_fin=TRUE  
if not IsMissing(Financne) then ' Parameter je zadaný
```

```

if TypeName(Financne)="String" then
' ak je vstupná hodnota logická, Calc ju odovzdá ako reťazec „true“ alebo „false“
if uCase(Financne)="TRUE" or uCase(Financne)="FALSE" then
    pom_fin=Financne
endif
endif
endif

' štandardne budeme predpokladať, že chceme začínať veľkým písmenom
pom_velke=TRUE
if not IsMissing(Velke) then ' Parameter je zadaný
if TypeName(Velke)="String" then
if uCase(Velke)="TRUE" or uCase(Velke)="FALSE" then
    pom_velke=Velke
endif
endif
endif

' číslo prevedieme na text
nazov=Cislo_Slovom.Daj_Slovne(cislo,pom_fin)

if pom_velke then
' a ak chceme prvé písmeno veľké, tak to zmeníme
nazov=uCase(left(nazov,1))+right(nazov,len(nazov)-1)
endif
Slovom=nazov
end function

```

Pri vlastnom volaní tejto funkcie si musíme uvedomiť, že nepovinné parametre sa „dopĺňajú“ zľava doprava, t.j. ak zadať tretí parameter „Velke“, musíme zadať aj druhý parameter „Financne“. Jednotlivé možnosti si však podrobnejšie preskúmajte ako domácu úlohu.

Skončili sme tretiu sériu nášho seriálu o programovaní makier v OpenOffice.org. Teraz môžete protestovať, že sme ešte neprebrali poslednú možnosť interaktívnych vstupov – dialógy. Vzhľadom na rozsah tejto problematiky im však budeme venovať celú nasledujúcu sériu článkov. Štvrtú sériu začneme, ako inak, makrami pre vkladanie nezalomiteľnej medzery za jednoznakové predložky s tým, že rozšírime možnosti OpenOffice.org o ich automatické vkladanie už počas písania (podobne, ako to majú konkurenčné produkty) a navyše do makra postupne budujeme možnosť vkladania „pružnej“ nezalomiteľnej medzery, čo v súčasnosti nedokážu ani najznámejšie platené kancelárske programy.