Makrá v

Július Pastierik

Makrá v OpenOffice.org

pastierik@inet.sk

Vytvorené pre internetový denník Inet.sk za podpory Liberix, o. p. s.

Text neprešiel jazykovou úpravou.

Elektronická verzia vychádza vlastným nákladom autora.
ISBN 978-80-970479-0-0

Táto kniha vznikla na základe článkov o programovaní makier, ktoré boli zverejnené v rokoch 2005 – 2009 na portáli Inet.sk. Je rozširovaná zadarmo v nádeji, že bude užitočná, avšak bez akejkoľvek záruky. Kniha je zverejnená pod licenciou Creative Commons BY-ND (uveďte autora, nezasahujte do diela).

Zdrojové texty makier sú zverejnené podľa ustanovení Všeobecnej verejnej licencie GNU GPL, vydávanej nadáciou Free Software Foundation a to buď verzie 2 tejto licencie, alebo, podľa vášho uváženia, ktorejkoľvek neskoršej verzie.

Z uvedeného vyplýva, že všetky zverejnené makrá sú voľne šíriteľné a môžete ich šíriť a modifikovať podľa ustanovení spomenutej Všeobecnej verejnej licencie GNU GPL. Na zverejnené makrá sa preto nevzťahuje žiadna záruka. Makrá sa poskytujú „tak, ako sú", bez záruky akéhokoľvek druhu, či už výslovnej, alebo vyplývajúcej, vrátane (ale nielen) záruk o vhodnosti pre určitý účel. Pokiaľ ide o kvalitu a výkonnosť makier, leží všetko riziko na vás. Ak by sa v makrách prejavili chyby, spadajú náklady za všetku potrebnú údržbu, opravy či nápravu na váš vrub.

V žiadnom prípade, s výnimkou toho, keď to vyžaduje platný zákon, vám nebude žiadny z držiteľov autorských práv ani žiadna iná strana zodpovedať za škody, vrátane všetkých všeobecných, špeciálnych, náhodných, alebo následných škôd vyplývajúcich z používania, alebo neschopnosti používať makro (vrátane, ale nie iba, straty alebo skreslenia dát, alebo trvalých škôd spôsobených vám alebo tretím stranám, alebo zlyhaním funkcie programu v súčinnosti s inými programami), a to aj v prípade, že držiteľ autorských práv alebo iná strana, boli upozornené na možnosť takýchto škôd.

Obsah

... začíname...

1. Začíname...

Mnohí používatelia kancelárskych balíkov poznajú pojem makro, iní však ani nevedia, čo to makro vôbec je. Mnohí z tých, ktorí sa s týmto pojmom už stretli by si chceli makro naprogramovať sami, ale nevedia, ako na to. Preto vznikol tento seriál, aby dal niektoré odpovede na tieto otázky.

Veľmi veľa používateľov kancelárskych balíkov skôr či neskôr narazí na problém, ktorý sa dá vyriešiť pomocou jednoduchého makra. Ale nie každý vôbec vie, že sa to dá takto riešiť a z tých, ktorí makrá poznajú aspoň ako pojem, si málokto trúfa také makro aj naprogramovať. V nasledujúcom seriáli sa rozhodne nebudeme snažiť urobiť zo všetkých používateľov programátorov, ale budeme sa ich snažiť naučiť aspoň základy, aby si niektoré jednoduché veci dokázali naprogramovať sami. Zároveň v ňom budeme uvádzať jednoduché príklady makier, ktoré, samozrejme, mnohí dokážu priamo využiť vo svojej každodennej práci s kancelárskym balíkom OpenOffice.org. Vhodné inšpirácie tu však určite nájdu aj používatelia iných kancelárskych balíkov, akým je napr. Microsoft Office.

1.1. „Nudné“ ale potrebné informácie na úvod

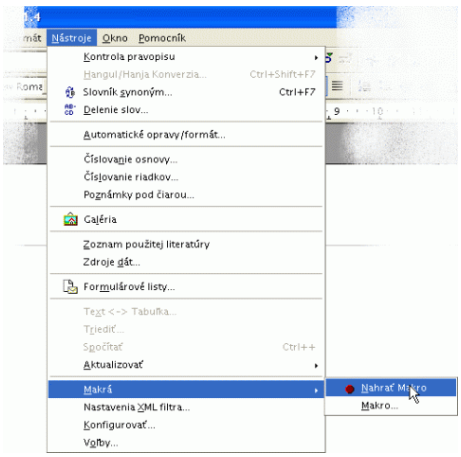
Priznám sa, že nemám rád zbytočné definície, a preto si dovolím predstaviť makro v kancelárskom balíku OpenOffice.org ako program, pomocou ktorého si dokážeme uľahčiť našu prácu s týmto programom alebo program, ktorý nám rozširuje jeho možnosti. Tieto programy môžu byť nielen maličké a jednoduché, ako je napr. vloženie záhlavia listu, ale aj pomerne rozsiahle, ako je napr. francúzsky makromodul Dmaths (www.dmaths.com), pomocou ktorého dokážeme o.i. priamo do textového dokumentu vkladať grafy funkcií. Vývojom makier pre OpenOffice.org sa zaoberá pomerne dosť programátorov, ktorí výsledky svojej práce poskytujú zadarmo, ako sme u Open Source programov vlastne zvyknutí, vrátane zdrojových textov aj iným. Z mnohých zdrojov takýchto makier spomeňme aspoň www.ooomacros.org/index.php. Mnohí tu určite nájdu zaujímavé makrá, ktoré priamo dokážu využiť pre svoju potrebu, iní aspoň inšpiráciu pre vlastnú tvorbu.

Skôr než pristúpime k vlastnej práci, ešte si musíme spomenúť, že makrá môžeme rozdeliť na dve základné skupiny. Prvá skupina makier (ako je napr. už aj spomínaný makromodul Dmaths) rozširuje vlastnosti OpenOffice.org a preto sa stávajú jeho súčasťou bez ohľadu na spracovávaný dokument. Takýmito makrami sa budeme zaoberať aj v našom seriáli. Okrem toho existuje aj iná skupina makier, ktorá je potrebná iba pre konkrétny dokument a preto takéto makrá bývajú súčasťou iba tohto dokumentu. Pri ich otváraní bývame upozornení (ak sme si to, samozrejme, nezakázali), že dokument obsahuje makro a teda že môže byť potenciálnym zdrojom vírusov. I keď som sa osobne s vírusmi v mne známych makrách pre kancelársky balík OpenOffice.org zatiaľ ešte nestretol, musím na tomto mieste na túto skutočnosť upozorniť. Táto skutočnosť, samozrejme, neznamená, že by sa takéto makrá nemali používať. Práve naopak, ich miesto je naozaj dôležité a mnohokrát nezastupiteľné. V konečnom dôsledku sa takýmto spôsobom distribuujú skoro všetky makrá, vrátane už viackrát spomínaného

makromodulu Dmaths.

1.2. Konečne „programujeme“

Aby sme v tomto seriáli oslovili čo najviac používateľov, budeme sa prioritne venovať modulu Writer, pretože prevažná väčšina používateľov pracuje práve s ním. Začneme tým najjednoduchším „programovaním“ – nahrávaním toho, čo píšeme. Týmto si môžeme vytvoriť napr. makro, pomocou ktorého automaticky vložíme záhlavie listu, alebo pozdrav s podpisom.

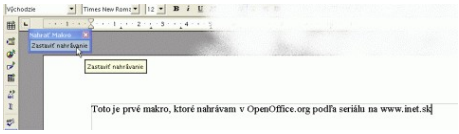


Obrázok 1.1: Spustíme nahrávanie makra

V hlavnom menu kancelárskeho balíka OpenOffice.org si vyberieme položku „Nástroje-Makrá-Nahrať makro“, ktorú si prípadne môžeme nechať zobrazovať aj na paneli funkcií. Po jej výbere sa nám zobrazí malé plávajúce okno „Nahrať makro“, kde nájdeme jediné tlačidlo „Zastaviť nahrávanie“, ktoré je, samozrejme, určené na ukončenie vlastného nahrávania. Toto okno si zatiaľ nebudeme všímať a začneme vkladať text, ktorý chceme nahrať, napr.:

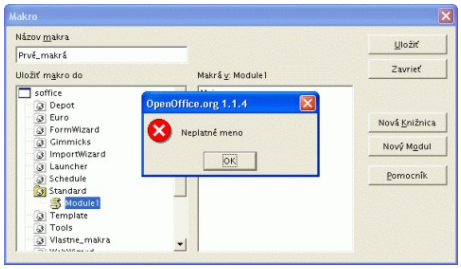
```
Toto je prvé makro, ktoré nahrávam
v OpenOffice.org podľa seriálu na
www.inet.sk
```

Teraz sa vrátime k už spomínanému oknu „Nahrať makro“ a zastavíme nahrávanie. Následne sa nám zobrazí nové dialógové okno „Makro“. V ňom sme vlastne vyzvaní, aby sme uložili práve „naprogramované“ makro pre vloženie napísaného textu. V zozname „Uložiť makro do“ nájdeme položku „soffice-Standard-Module1“ a v okienku „Názov makra“ zadáme názov, pod akým chceme toto makro používať.



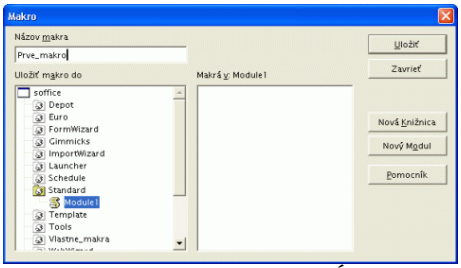
Obrázok 1.2: Zastavíme nahrávanie

Pri programovaní si musíme zvyknúť, že názvy (a to nielen makier, ale obecne akékoľvek názvy v programoch) nemôžu mať ľubovoľný tvar, t.j. musia začínať písmenom, nesmú obsahovať pomlčky, medzery, či iné netlačiteľné znaky. Zároveň nesmieme používať znaky s diakritikou. Pokiaľ sa pokúsime o zapísanie nesprávneho názvu, OpenOffice.org nás na to upozorní.



Obrázok 1.3: Meno nesmie obsahovať dĺžne

Z vlastnej praxe vám môžem odporučiť, že ak chcete používať prehľadné dlhé názvy s oddeľujúcou medzerou, namiesto nej je najvhodnejšie po-



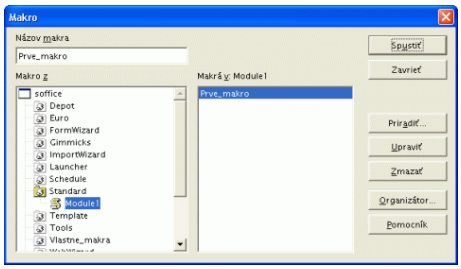
Obrázok 1.4: Meno bez dĺžňov je v poriadku

užiť ako náhradu znak podčiarknutia. Po zadaní správneho mena (napr. Prve_makro) pomocou tlačidla „Uložiť“ uložíme naše makro. Ak sme sa náhodou pomýlili, a makro nechceme uložiť, ukončíme prácu pomocou tlačidla „Zavrieť“.

1.3. Ako spustíme makro?

Napísali sme svoje prvé makro. Robili sme ho však preto, aby sme ho mohli používať aj inokedy a preto musíme vedieť, ako makro spustíme. Cez voľbu „Nástroje-Makrá-Makro“ znovu otvoríme dialógové okno „Makro“.

Znovu prejdeme na položku „soffice-Standard-Module1“ a vo vedľajšom okienku sa nám zobrazí zoznam našich makier, kde teraz vidíme aj naše „Prve_makro“. Prejdeme naň a stlačíme tlačidlo „Spustiť“ Do aktuálneho textového dokumentu sa nám automaticky vloží nami zadefinovaný (teda nahraný) text:



Obrázok 1.5: Spúšťame naše makro

```
Toto je prvé makro, ktoré
nahrávam v OpenOffice.org podľa
seriálu na www.inet.sk
```

Ako vidíme, naše makro funguje bezchybne a stalo sa súčasťou kancelárskeho balíka OpenOffice.org. Určite si teraz mnohí z vás vedia predstaviť, načo všetko sa takéto jednoduché nahrávanie dá využiť. Pripomeňme preto iba, že pri takomto nahrávaní sa zapamätajú, samozrejme, aj formátovacie nastavenia, ako je typ a rez písma, veľkosť, farba a pod., takže si môžeme veľmi jednoducho vytvoriť naozaj bohatú skupinu rôznych zaujímavých makier.

2. Konfigurujeme OpenOffice.org

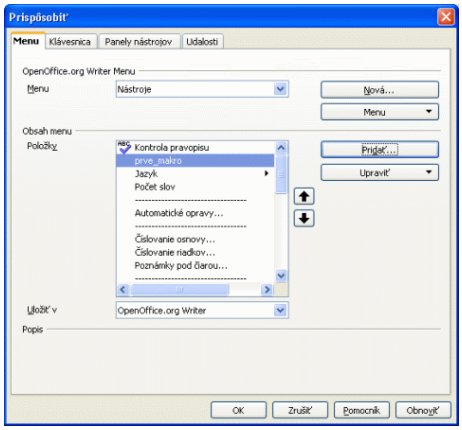
Ukázali sme si ako sa dá vytvoriť jednoduché makro bez toho, aby sme vôbec vedeli programovať. Zároveň sme si ukázali jednu z možností, ako ho spustiť. Samozrejme, táto možnosť nie je jediná a, priznajme si, ani tá najpohodlnejšia.

Okrem už uvedeného spôsobu môžeme makrá sprístupniť prakticky do všetkých častí OpenOffice.org. Tomuto účelu (a, samozrejme, nielen tomuto) je určená voľba v menu „Nástroje-Prispôsobiť“. V starších verziách OpenOffice.org to bola voľba „Nástroje-Konfigurovať“. Zároveň sa tam trochu líšil spôsob vlastnej konfigurácie. Vzhľadom na aktuálnosť verzie 2.0[1] sa však už nebudeme viacej zaoberať staršími verziami.

Po zvolení spomínanej voľby sa otvorí dialógové okno „Prispôsobiť“. Okno je tvorené štyrmi záložkami, na ktorých si môžeme vlastne zmeniť vzhľad a prístupné funkcie celého OpenOffice.org. Nebudeme sa zaoberať podrobným popisom všetkých funkcií, obmedzíme sa iba na priradenie makra k jednotlivým možnostiam a, samozrejme, na opačný krok – zrušenie tohto priradenia.

1 V čase písania tejto knihy je aktuálna verzia 3.2.1

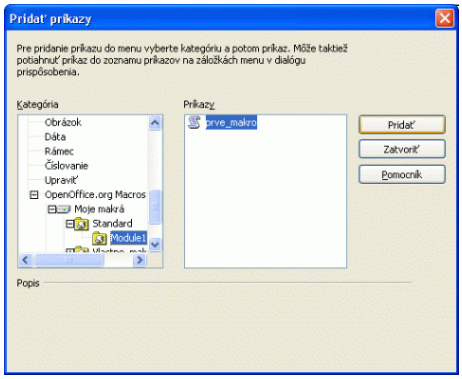
2.1. Záložka „Menu“



Obrázok 2.1: Záložka "Menu"

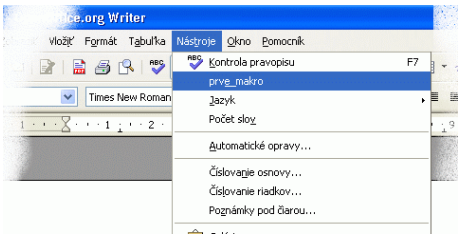
Na tejto záložke môžeme zmeniť menu OpenOffice.org. Vzhľadom na to, že tu vlastne môžeme zmeniť celé menu OpenOffice.org, musíme byť veľmi pozorní, aby sme si, náhodou, nezrušili prístup k dôležitým funkciám. Venujme sa však vlastnému zaradeniu nášho makra do menu. Záložka sa skladá z dvoch zoznamov.

V prvom si vyberieme menu, do ktorého chceme pridať naše makro (napr. „Nástroje“). V druhom zozname sa nám následne zobrazí vlastný obsah menu. Prejdeme na miesto za ktoré chceme vložiť naše makro (napr. na položku „Kontrola pravopisu“) a stlačíme tlačidlo „Pridať“. Zobrazí sa nám okno „Pridať príkazy“, kde si v zozname „Kategória“ nájdeme položku, kde sme si uložili naše makro („OpenOffice.org Macros – Moje makrá – Standard – Module1“) a v okienku „Príkazy“ si označíme požadované makro „Prve_makro“. Následne stlačíme tlačidlo „Pridať“.



Obrázok 2.2: Vyhľadáme naše makro

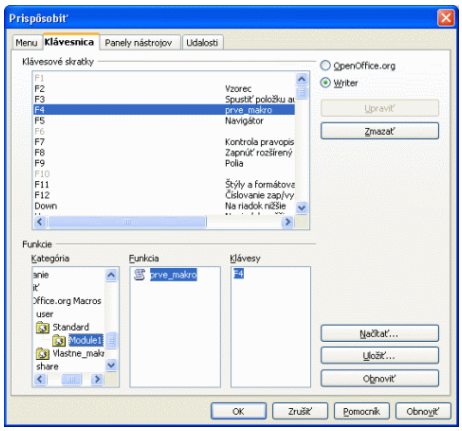
Makro sa nám zobrazí v zozname položiek vybraného menu. V prípade potreby ho môžeme ešte popresúvať (napr. ak ho chceme mať úplne na začiatku menu) a nakoniec celú prácu potvrdíme tlačidlom „OK“.



Obrázok 2.3: Naše makro ako súčasť menu

Ak sme postupovali správne, makro máme teraz priamo prístupné v menu „Nástroje-Prve_makro“. Pokiaľ chceme naše makro z menu premenovať alebo natrvalo odstrániť, prejdeme znovu do tejto záložky dialógového okna „Prispôsobiť“, prejdeme na našu položku v menu a stlačíme tlačidlo „Upraviť“, kde vyberieme voľbu „Premenovať“ alebo „Zmazať“. Potom zmenu znovu potvrdíme tlačidlom „OK“.

2.2. Záložka „Klávesnica“



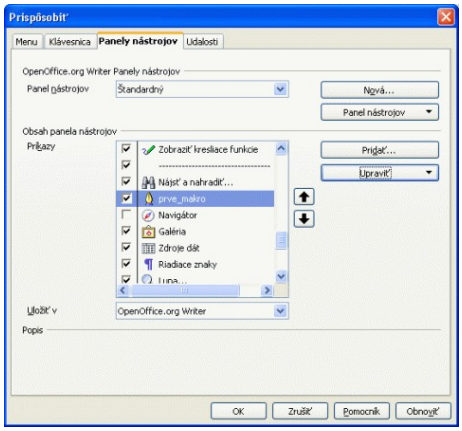
Obrázok 2.4: Záložka "Klávesnica"

Na tejto záložke môžeme naše makro priradiť k nejakej klávesovej skratke. V okienku „Klávesové skratky“ si nájdeme tú skratku, ktorú chceme používať pre spúšťanie nášho makra. Pozor na označenie prepínacích hodnôt „OpenOffice.org“ a „Writer“ (nachádzajú sa v pravo hore vedľa okienka „Klávesové skratky“). Podľa ich nastavenia

totiž priradíme makro k vybranej klávesovej skratke buď pre celý OpenOffice.org, alebo iba pre modul Writer. Pokiaľ má jedna klávesová skratka nastavené rôzne makrá (iné pre Writer a iné pre celý OpenOffice.org), prednosť má makro priradené ku klávesovej skratke pre modul Writer.

Zoznam makier máme v tomto prípade priamo prístupný v spodnej časti tejto záložky. Naše makro nájdeme, samozrejme, podobne ako v predchádzajúcom príklade v okienkach „Kategória“ a „Funkcia“. Potom stlačíme tlačidlo „Upraviť“, čím makro priradíme ku vybranej klávesovej skratke. Naopak, ak chceme makro od klávesovej skratky odpojiť, po jeho označení príslušnej skratky stlačíme tlačidlo „Zmazať“. Celú prácu potvrdíme tlačidlom „OK“.

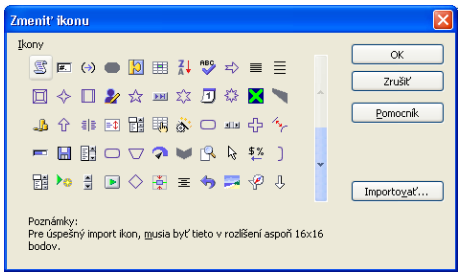
2.3. Záložka „Panely nástrojov“



Obrázok 2.5: Záložka "Panely nástrojov"

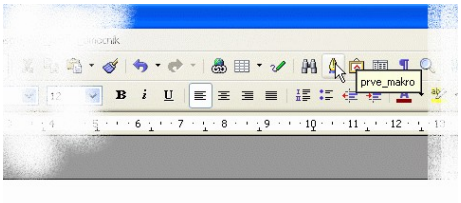
Prešli sme na záložku, kde si makro môžeme priradiť na niektorý panel z funkciami. Práca na tejto záložke je veľmi podobná práci na záložke „Menu“ a preto iba stručne spomeňme, že najprv si v zozname panelov vyberieme panel, na ktorý chceme priradiť naše makro, napríklad „“Štandardný“ a následne sa v zozname zobrazených príkazov presunieme na miesto, kde chceme vložiť naše makro.

Následne cez voľbu „Pridať“ vyhľadáme a pridáme do zoznamu príkazov naše makro. Pretože robíme s panelmi nástrojov, pridáme k nášmu makru aj vhodnú ikonu. Ich zoznam sa nám otvorí po výbere voľby „Upraviť – Zmeniť ikonu...“. Vtedy sa nám otvorí ďalšie okno „Zmeniť ikonu“. Tu si nájdeme takú, ktorá sa nám najviac páči alebo vyhovuje našej predstave a výber potvrdíme tlačidlom „OK“.



Obrázok 2.6: Vyberáme si ikonu

Ak nám zoznam ikoniek nevyhovuje alebo nepostačuje, môžeme ho ľahko rozšíriť o vlastné tlačidlá. Pre tento účel si ich musíme, samozrejme, najprv vytvoriť. Majú rozmery 16x16 alebo 26x26 bodov a môžu byť v jednom z vyše dvadsiatich podporovaných grafických formátov. Samozrejme, stačí ak si vytvoríme tlačidlá iba toho rozmeru, aký máme nastavený vo voľbe „Nástroje-Možnosti-Zobraziť-Veľkosť ikony“ (malé – 16x16 alebo veľké – 26x26). Vlastné súbory s ikonami potom pridáme do zoznamu cez voľbu „Importovať“.

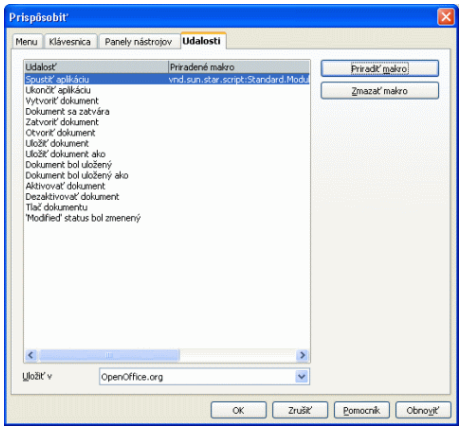


Obrázok 2.7: Naše makro v paneli nástrojov

Podobne ako pri zmene menu, aj tu môžeme nášmu vloženému príkazu (cez voľbu „Upraviť – Premenovať“ alebo „Upraviť – Zmazať“) zmeniť názov alebo ho úplne zmazať. Názov príkazu (pokiaľ to, samozrejme, nezakážeme) sa po prejdení na túto položku zobrazuje

v malom žltom nápovednom okienku.

2.4. Záložka „Udalosti“

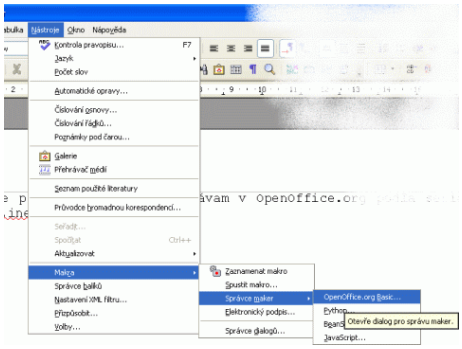


Obrázok 2.8: Záložka "Udalosti"

Na tejto záložke môžeme priradiť makro k nejakej udalosti. Najprv si musíme nastaviť, či chceme priradiť makro k udalosti, ktorá bude platná pre celý OpenOffice.org, alebo iba pre aktuálny dokument tým, že si vyberieme miesto uloženia. Následne si vyberieme udalosť v okienku „Udalosť“ a stlačíme tlačidlo „Priradiť makro“. Otvorí sa nám okno „Vyberač makier“, kde už známym spôsobom nájdeme naše makro. Pokiaľ chceme naopak makro od udalosti odobrať, musíme po výbere tejto udalosti stlačiť tlačidlo „Zmazať makro“. Celú prácu, samozrejme, potvrdzujeme na konci tlačidlom „OK“.

Ako vidíme, možnosti OpenOffice.org v oblasti (nielen) rýchleho a prehľadného spúšťania makier sú naozaj bohaté a nám nezostáva nič iné, ako si vybrať ten najvhodnejší systém, ktorý nám bude najlepšie vyhovovať. Zároveň sme si vlastne dnes ukázali, ako si môžeme prispôsobiť celé pracovné rozhranie tohto balíka našim požiadavkám, takže sa nemusíme prispôsobovať my pracovnému rozhraniu, ale ono sa dokáže prispôsobiť nám.

3. Čo sme to naprogramovali?

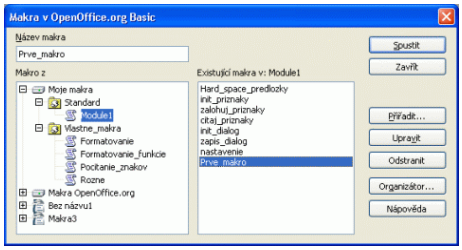


Obrázok 3.1: Otvoríme si okno "Makrá"

Ako prvý krok sme „programovali“ makro jeho nahrávaním. Pozrieme sa na to, aký program sme takto vlastne vytvorili a popíšeme si všetky príkazy.

Hoci sme už uviedli, že nahrávanie makier nie je priame programovanie, vytvorili sme takýmto postupom procedúru v jazyku StarOffice Basic. Je to tak preto, že potrebné inštrukcie za nás vložil priamo OpenOffice.org. Pozrime sa teda na to, čo napísal za nás. Otvoríme si dialógové okno „Makrá“. Na rozdiel od verzie 1.1.4 je prístup k tomuto oknu trochu zložitejší, pretože verzia 2.0 podporuje makrá až v štyroch programovacích jazykoch. V našom prípade musíme prejsť cez voľby v menu „Nástroje-Makrá-Usporiadať makrá...-OpenOffice.org Basic...“. V ľavej časti vidíme zoznam našich makier v rozbaľovacom poli „Moje makrá“.

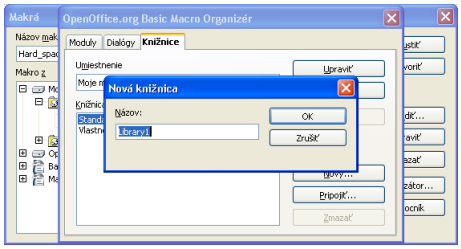
3.1. A znovu niečo konfigurujeme



Obrázok 3.2: Určite budeme mať viac makier

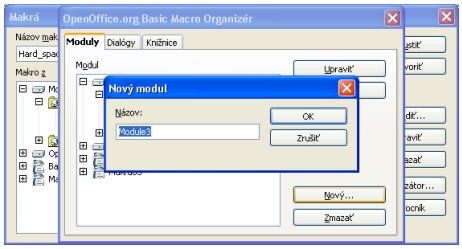
Samozrejme, nemusíme sa uspokojiť iba so štandardným umiestnením. Pokiaľ budeme mať makier viac (a budeme ich mať určite viac), je vhodné si ich usporiadať podľa určitého systému. Za týmto účelom si môžeme vytvoriť vlastné knižnice,

v ktorých budeme definovať vlastné moduly s makrami. K správe knižníc a modulov sa dostaneme cez tlačidlo „Organizátor“. Ako vidíme, otvorilo sa nám nové okno „OpenOffice.org Basci Makro Organizér“, kde sa nachádzajú tri záložky – „Moduly“, „Dialógy“ a „Knižnice“.



Obrázok 3.3: Vkladáme nová knižnicu

V záložke „Knižnice“ si cez tlačidlo „Nový“ vytvoríme našu knižnicu (napr. „Vlastne_makra“). Následne si v záložke „Moduly“ nájdeme túto knižnicu a tak isto cez tlačidlo „Nový“ si v nej vytvoríme vlastné moduly (napr. „Rozne“, „Formatovanie“ a pod.). Pretože sa nám pri vytváraní knižnice automaticky vytvoril aj modul „Module1“, tento vymažeme (tlačidlo „Zmazať“).



Obrázok 3.4: Vkladáme nový modul

Záložku „Dialógy“ si zatiaľ nebudeme všímať. Určite sa však k nej vrátime v niektorom z budúcich pokračovaní tohto seriálu, keď si budeme hovoriť o tvorbe vlastných dialógových okien. Teraz by sme však ešte predbiehali.

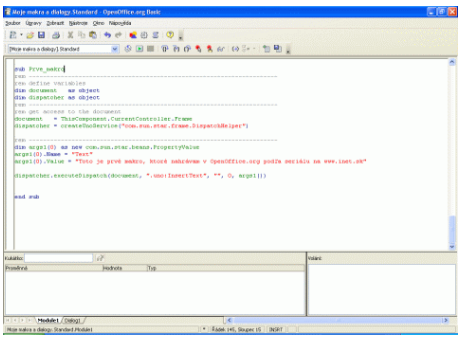
3.2. Dešifrujeme „tajomné“ programové zápisy

Vráťme sa však k nášmu makru „Prve_makro“. Po jeho nájdení stlačíme tlačidlo „Upraviť“, čím sa dostaneme do modulu OpenOffice.org Basic, ktorý je určený pre programovanie makier. Vidíme, že OpenOffice.org za nás naprogramoval takýto program:

```
sub Prve_makro
rem
----------------------------------------------------------------
--
rem define variables
dim document as object
dim dispatcher as object
rem
----------------------------------------------------------------
--
rem get access to the document
document = ThisComponent.CurrentController.Frame
dispatcher =
createUnoService("com.sun.star.frame.DispatchHelper")

rem
----------------------------------------------------------------
--
dim args1(0) as new com.sun.star.beans.PropertyValue
args1(0).Name = "Text"
args1(0).Value = "Toto je prvé makro, ktoré nahrávam v
OpenOffice.org podľa seriálu na www.inet.sk"
```

Čo sme to naprogramovali?

```
dispatcher.executeDispatch(document, ".uno:InsertText", "",
0, args1())

end sub
```



Obrázok 3.5: Modul Basic

Rozlúštime teraz tento tajomný zápis. Je to vlastne podprogram (t.j. časť programu) v jazyku StarOffice Basic, ktorý si podrobne popíšeme. Hneď na začiatku si však musíme uvedomiť, že žiaden automat nedokáže „programovať“ úplne optimálne a preto je možné, že takýto program sami dokážeme naprogramovať lepšie a jednoduchšie. Nič viac, pre mnohých bude zo začiatku práve takáto forma programovania úplne postačujúca, najmä keď si sa mi dokážu prispôsobiť takto vytvorené programy podľa svojich predstáv a potrieb. Na druhej strane, aj z takéhoto jednoduchého príkladu môže byť zrejmé, že analýza cudzích programov nemusí byť (a mnohokrát naozaj ani nie je) vôbec jednoduchá a teda, že ani otvorený zdrojový kód neznamená, že aj profesionálny programátor sa v ňom musí vyznať.

```
sub Prve_makro
```

Tento príkaz určuje, že všetko to, čo je za ním je podprogram (SUBroutine) StarOffice Basic, ktorý sme nazvali Prve_makro. Názov makra (podprogramu) musí byť, samozrejme, jednoznačné. V tejto súvislosti ešte musíme uviesť, že pri akýchkoľvek názvoch programov, premenných či funkcií sa v jazyku StarOffice Basic nerozoznáva veľkosť písmen, t.j. napr. názov „prve_makro“ je úplne totožný s názvom „Prve_makro“. Prípadné používanie veľkých a malých písmen nám však môže značne pomôcť v udržaní potrebného prehľadu v programe.

```
Rem ------
```

Tento príkaz označuje tzv. poznámku (REMark). Je to vlastne iba kozmetický prvok, pomocou ktorého si udržujeme potrebný prehľad najmä vo väčších programoch a tento príkaz nemá žiadny vplyv na ostatné vykonávané funkcie a príkazy. Okrem takejto poznámky môžeme poznámku označovať aj znakom apostrof:

```
' Aj toto je poznámka
```

Poznámku, ktorá začína apostrofom môžeme pridávať aj na koniec akéhokoľvek iného príkazu, kým poznámka REM musí byť vždy na osobitnom riadku:

```
vysledok=sin(x) ' Výpočet sínusu
REM Výpočet sínusu
vysledok=sin(x)
```

Poznámku však využijeme nielen ako kozmetický prvok, ale aj pri ladení. Vtedy si ako poznámku označíme napr. tie časti programu, ktoré nás zbytočne zdržujú, nie sú ešte odladené a pod., alebo máme pripravených viacero variánt a postupným „zapoznámkovaním“ a „odpoznámkovaním“ zisťujeme, ktorá je tá najsprávnejšia.

```
dim document as object
dim dispatcher as object
```

Príkaz DIM (DIMension) je určený na definovanie tzv. premenných. Premenná je akoby nejaká krabička, do ktorej ukladáme rôzne údaje s tým, že ich neskôr, keď sa nám to bude hodiť, primerane použijeme. Každá takáto krabička je špeciálna a je určená iba na jeden typ údajov, t.j. napr. na ukladanie textu, čísiel a pod. O jednotlivých druhoch premenných si budeme hovoriť podrobnejšie až v siedmom pokračovaní nášho seriálu, takže teraz iba skonštatujme, že sme si týmito príkazmi vytvorili dve objektové premenné, ktoré sme pomenovali „document“ a „dispatcher“.

```
  document = ThisComponent.CurrentController.Frame
```

Znak „rovná sa“ = je jeden z najdôležitejších príkazov vôbec, pretože je to tzv. priraďovací príkaz. Pomocou neho priraďujeme do premenných ich hodnoty – či už priamo, alebo ako výsledky operácií a funkcií. Pomocou tohto konkrétneho príkladu sme do premennej „Document“ priradili aktuálny dokument.

```
  dispatcher =
createUnoService("com.sun.star.frame.DispatchHelper")
```

Ďalším priraďovacím príkazom sme do premennej „dispatcher“ priradili tzv. UNO (Universal Network Objects) objekt. Je to vlastne univerzálne objektovo orientované programátorské rozhranie, ktoré sa používa pre riadenie kancelárskeho balíka OpenOffice.org.

```
  dim args1(0) as new com.sun.star.beans.PropertyValue
  args1(0).Name = "Text"
  args1(0).Value = "Toto je prvé makro, ktoré nahrávam v
OpenOffice.org podľa seriálu na www.inet.sk"
```

Definovali sme si ďalšiu premennú „args1“, pomocou ktorej budeme vkladať vlastný text. Je to štruktúrovaná premenná, kde do časti „Name“ priraďujeme typ a do časti „Value“ hodnotu, z ktorou budeme pracovať. Pretože v našom príklade vkladáme textový reťazec, ako typ musíme použiť „Text“ a ako hodnotu vlastný vkladaný reťazec.

```
  dispatcher.executeDispatch(document, ".uno: InsertText",
"", 0, args1())
```

Hoci na to tento riadok na prvý pohľad vôbec nevyzerá, je to vlastne volanie iného podprogramu, či lepšie povedané v tomto prípade štandardnej UNO metódy. Prístup k UNO rozhraniu sme získali jeho priradením do premennej „dispatcher“ a preto príslušnú metódu „executeDispatch“ máme prístupnú priamo cez ňu. V tomto konkrétnom prípade nám jej volanie zabezpečí vloženie textu (parameter „InsertText“) do aktuálneho dokumentu (premenná „document“), pričom vkladaný text máme uložený v premennej „args1“.

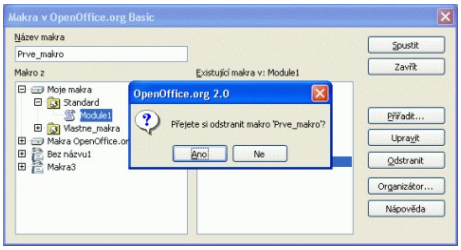
Pokiaľ si budete nahrávať rôzne iné makrá, zistíte, že táto istá metóda, samozrejme, s rôznymi parametrami sa používa skoro pre všetky činnosti. Ich jednoducho analýzou môžete sami zistiť, aký význam majú jednotlivé parametre. My ju však pri vlastnom programovaní nebudeme skoro vôbec používať.

```
  end sub
```

Na koniec nasleduje už iba bodka za vetou, t.j. označenie konca procedúry (END SUBroutine).

Rozobrali sme podrobne príkazy nášho prvého makra. Na koniec práce nezabudne-

me zavrieť modul Basic a pokiaľ budeme chcieť, vymažeme naše prvé makro (tlačidlo „Zmazať“). Každý si môže teraz sám odskúšať nahrávať a analyzovať rôzne makrá, čím získa určitú predstavu o rôznych UNO metódach a ich parametroch.



Obrázok 3.6: Mažeme makro

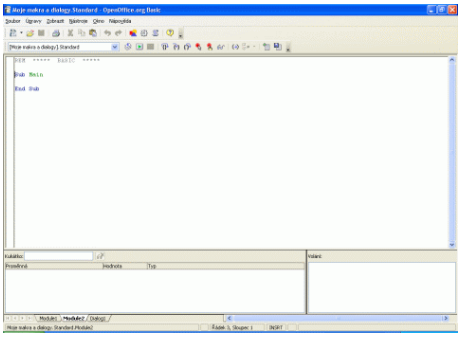
4. Vymazávanie medzier

Teraz si naprogramujeme prvú verziu makra pre vymazávanie viacnásobných medzier. Toto makro bude základom pre veľa ďalších, pomocou ktorých budeme formátovať dokument.

Určite sa aj vám stalo, že pri písaní zadáte omylom viac medzier (a vzhľadom na to, že osobne píšem veľmi veľa, môžem iba potvrdiť, že je to naozaj častý preklep). Tento problém sa síce dá riešiť aj pomocou funkcie „Nájsť a nahradiť“, ale ak píšete veľa, je to pomerne nepraktické. Preto si naprogramujeme makro, ktoré to bude robiť automaticky za nás. Zároveň si na tomto makre postupne ukážeme rôzne funkcie programovacieho jazyka StarOffice Basic.

4.1. Programujeme v StarOffice Basic

Nové makrá môžeme do OpenOffice.org vkladať priamo tak, že si už známym postupom, o ktorom sme písali minule (menu „Nástroje-Makrá-Usporiadať makrá...-OpenOffice.org Basic...“) otvoríme dialógové okno pre správu makier. Tu prejdeme na modul, kde chceme vložiť naše makro a stlačíme tlačidlo „Upraviť“.


Obrázok 4.1: Makro "Main"

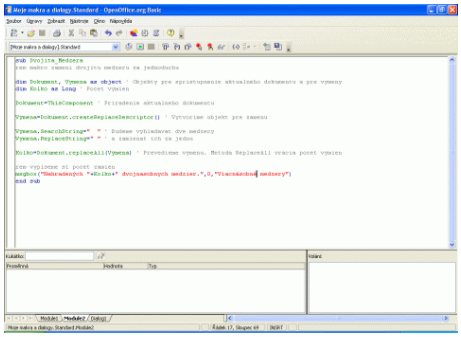
Štandardne v každom module vytvára na začiatku OpenOffice.org prázdne makro s názvom „Main“:

```
REM ***** BASIC *****
Sub Main
End Sub
```

S kľudným svedomím môžeme toto makro vymazať, pretože ho nebudeme používať. Teraz napíšeme nasledovné makro s tým, že poznámky a prázdne riadky, samozrejme, nemusíme písať, sú uvedené iba pre lepšiu zrozumiteľnosť a prehľadnosť:

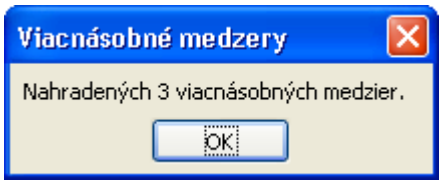
```
sub Dvojita_Medzera
rem makro zameni dvojitu medzeru za jednoduchu

dim Dokument, Vymena as object ' Objekty pre spristupnenie
aktualneho dokumentu a pre vymeny
dim Kolko as Long ' Pocet vymien

Dokument=ThisComponent ' Priradenie aktualneho dokumentu

Vymena=Dokument.createReplaceDescriptor() ' Vytvorime
```

```
objekt pre zamenu

Vymena.SearchString="  " ' Budeme vyhladavat dve medzery
Vymena.ReplaceString=" " ' a zamienat ich za jednu
Kolko=Dokument.replaceAll(Vymena) ' Prevedieme vymenu.
Metoda ReplaceAll vracia pocet vymien

rem vypiseme si pocet zamien
msgbox("Nahradených "+Kolko+" dvojnasobnych
medzier.",0,"Viacnásobné medzery")
end sub
```



Obrázok 4.2: Naše makro počas písania

Prejdeme do pôvodného dokumentu (nemusíme ukončovať modul „Basic“, stačí, ak prepneme iba okno), kde máme na odskúšanie napísaný text s viacnásobnými medzerami a spustíme vytvorené makro „Dvojita_Medzera“. Ak sme všetko naprogramovali správne, na konci sa dozvieme počet prevedených zámen dvojitých medzier.



Obrázok 4.3: Makro funguje

V prípade bezchybného chodu zatvoríme následne modul „Basic“, inak musíme hľadať a opravovať chyby.

Teraz sa pozrime na naše makro podrobne a rozoberme si príkaz za príkazom (teda okrem poznámok).

```
sub Dvojita_Medzera
```

Naše makro sme nazvali „Dvojita_Medzera“. Názvy sa budeme snažiť dávať také, aby sme podľa nich ľahko rozpoznali, o čo sa vlastne jedná.

```
dim Dokument, Vymena as object
```

Definovali sme dve premenné typu object. Niektorí autori používajú také názvy premenných, kde prvé písmeno vyjadruje aj jej typ (napr. v našom prípade oDokument, oVymena). Je to dobrá pomôcka, ktorú však osobne veľmi nepoužívam.

Premenná „Dokument“ je určená pre prístup k aktuálnemu dokumentu. Druhá premenná „Vymena“ je určená pre prevedenie vlastnej zámeny dvojitých medzier za jednoduché.

```
dim Kolko as Long
```

Do číselnej premennej „Kolko“ si uložíme počet výmien.

```
Dokument=ThisComponent
```

Priradenie aktuálneho dokumentu do premennej „Dokument“. Týmto sme získali k tomuto dokumentu prístup.

```
Vymena=Dokument.createReplaceDescriptor()
```

Pre výmenu v aktuálnom dokumente (premenná „Dokument“) musíme vytvoriť ob-

jekt pre nahrádzanie, ktorý priradíme do premennej „Vymena“.

```
Vymena.SearchString="  "
```

V objekte „Vymena“ nastavíme do jeho vyhľadávacej časti reťazec, ktorý chceme nahradiť. V našom prípade sú to dve medzery. Reťazce uzatvárame medzi dve úvodzovky. Pozor na častú chybu začiatočníkov, pokiaľ si pripravujú zdrojový text v externom editore – ako si môžete všimnúť, programové úvodzovky nie sú totožné s bežnými slovenskými a českými textovými úvodzovkami a preto je potrebné ich skontrolovať.

```
Vymena.SearchString="  " ' Budeme vyhladavat dve medzery
Vymena.SearchString="  " ' Toto su nespravne uvodzovky pre ohranicenie textu
' A toto je nespravny apostrof pre poznamku
```

Obrázok 4.4: Správne a nesprávne úvodzovky

Na priloženej zosnímanej obrazovke si môžete všimnúť tento rozdiel. To isté platí aj pre apostrof, ktorý označuje poznámku.

```
Vymena.ReplaceString=" "
```

V objekte „Vymena“ nastavíme do jeho nahrádzajúcej časti reťazec, za ktorý chceme vymeniť hľadaný text. V našom prípade je to jedna medzera uzatvorená, ako inak, do úvodzoviek.

```
Kolko=Dokument.replaceAll(Vymena)
```

Pretože sme si v objekte „Vymena“ pripravili všetko, čo potrebujeme pre našu zámenu, môžeme túto previesť zavolaním metódy „ReplaceAll“, ktorá je prístupná v každom aktuálnom dokumente (my ho máme sprístupnený pomocou premennej „Dokument“). Táto metóda zároveň vracia počet zámen, ktorý si uložíme do premennej „Kolko“.

```
msgbox("Nahradených "+kolko+" dvojnasobnych medzier.", 0,
"Viacnásobné medzery")
```

Určite sme zvedaví, koľko takýchto preklepov robíme a preto si zobrazíme ich počet. Na tento účel použijeme systémovú procedúru „msgbox“. V zátvorke sa nachádzajú tzv. parametre, ktoré sú oddelené čiarkou. Prvý z nich je reťazec: "Nahradených "+kolko+" dvojnasobnych medzier.". Je to vlastný vypisovaný text. Druhý parameter je číslo nula: 0. Ním určujeme aké tlačidlá sa majú zobraziť. Nebudme sa teraz zaoberať ostatnými hodnotami, pretože tie majú význam pri vstupe a preto iba pripomeňme, že hodnota 0 znamená zobrazenie tlačidla „OK“. Posledný parameter je znovu reťazec: "Viacnásobné medzery". Použije sa ako názov tohto okna.

```
end sub
```

Týmto príkazom sme ukončili makro „Dvojita_Medzera“.

Ako vidíte, postavili sme týmto makrom základ pre naozaj veľa makier, kde je potrebná zámena jedného reťazca za iný. Priamo v tomto tvare sa dá použiť napr. pre makro, pomocou ktorého budeme zamieňať tri bodky „...“ za „trojbodku“ „…“, t.j. príslušné riadky (parametre procedúry „msgbox“ si zmeňte sami) budú vyzerať takto:

```
Vymena.SearchString="..."
Vymena.ReplaceString="…"
```

Hoci sa nám makro pre výmenu dvojitých medzier zdá dobré, nehodí sa nám pre také opravy, keď ako preklep nebudeme mať iba dve medzery, ale viac (vtedy ho totiž musíme spúšťať opakovane dovtedy, kým sa vyskytujú dvojité medzery). Tieto (a ešte aj iné nedostatky) však budeme odstraňovať až v niektorých z budúcich dielov tohto seriálu, keď sa zoznámime s príkazmi jazyka StarOffice Basic.

5. Trinásta komnata sa otvára...

Pozrime sa na tajomstvá programovania a otvorme trinástu komnatu. Zistíme, že programovanie je vlastne veľmi jednoduché – a práve pre túto jednoduchosť nezvládne programovanie každý. Je to vlastne ako so známou Rubikovou kockou – mnohí ju vedia skladať, ale iba nemnohí na to prišli sami bez cudzej pomoci. Mnohí pritom používajú desiatky príkazov, ale v skutočnosti na jej zloženie stačia iba dva jednoduché príkazy, ktoré opakujeme vo vhodnej postupnosti...

Vráťme sa však k predchádzajúcemu dielu nášho seriálu a pripomeňme si, že makro pre zámenu dvojnásobných medzier má ešte nedostatky. Ten hlavný je v tom, že nedokáže odstrániť napr. trojnásobné medzery na jeden krát. Ako by sme to riešili v prípade ručnej zámeny? Najprv by sme asi zamenili dve medzery za jednu a potom by sme tento krok opakovali dovtedy, kým by sa nejaká zámena vyskytovala.

Tento postup sa nazýva algoritmus. Samozrejme, nie je to jediný postup, pomocou ktorého dokážeme vyriešiť náš problém ale je to postup, ktorý je správny – a o to nám predovšetkým ide. Algoritmus je totiž prvý – základný krok programovania. Zatiaľ ho máme popísaný iba slovne a preto ho teraz musíme v druhom kroku prepísať do nejakého programovacieho jazyka, teda v našom prípade do jazyka StarOffice Basic. A napokon v poslednom – treťom kroku musíme program odladiť, t.j. odstrániť prípadné chyby. Tento krok býva mnohokrát ten najdlhší a najnáročnejší.

6. Príkazy StarOffice Basic

Toto je celé tajomstvo programovania. Vráťme sa teraz k druhému kroku – prepisu algoritmu do programovacieho jazyka StarOffice Basic. Na to, aby sme to mohli robiť, potrebujeme poznať jeho príkazy. A tak isto, ako pri Rubikovej kocke, v ľubovoľnom programovacom jazyku nám stačia iba dva základné príkazy – vetvenie podľa podmienky a opakovací cyklus, ktoré opakujeme vo vhodnej postupnosti. Zdá sa, že to je málo, ale je to naozaj tak (a existujú na to dokonca aj matematické dôkazy podľa teórie grafov).

Pri praktickom programovaní sa však stretneme s trochu viac príkazmi, ako sú dva – sú to však iba rôzne varianty, ktoré primerane zjednodušujú niektoré často sa opakujúce postupnosti. Pri popise týchto príkazov si potom uvedieme, ako sa dajú naprogramovať aj pomocou základných príkazov, takže uvidíme toto zjednodušenie.

6.1. Vetvenie

Vetvenie je vlastne rozdelenie programu na dve časti – jedna sa vykoná vtedy, ak podmienka vetvenia je splnená, druhá vtedy, ak nie je splnená. S vetvením sa stretávame úplne stále a všade aj v každodennom živote – veď napokon obyčajné rozhodovanie sa je vlastne vetvenie...

6.1.1. Príkaz IF

Vetvenie má v jazyku StarOffice Basic nasledujúci formát:

```
IF podmienka THEN
 ' … príkazy pri splnenej podmienke…
ELSE
 ' … príkazy pri nesplnenej podmienke…
END IF
```

V prípade, že nechceme nič vykonať, ak podmienka nie je splnená, tak tento príkaz nemusí obsahovať vetvu ELSE:

```
IF podmienka THEN
 ' … príkazy pri splnenej podmienke…
END IF
```

Chceme zistiť, či nejaké číslo je párne. To, či je číslo párne sa najlepšie zistí tak, že zvyšok po delení dvomi musí byť nula (na zistenie zvyšku po delení nám slúži matematický operátor MOD). Test potom môžeme naprogramovať napríklad takto:

```
IF cislo MOD 2 = 0 THEN
 ' cislo je párne
ELSE
 ' cislo je nepárne
END IF
```

V mnohých prípadoch nám však nestačí iba takéto jednoduché vetvenie. Napríklad, ak potrebujeme zistiť, či nejaký rok je alebo nie je prestupný. Prestupný rok je totiž rok, ktorý je deliteľný 4 alebo 400 (napr. roky 1600, 2004 boli prestupné) ale nie je deliteľný 100 (napr. rok 1900 nebol prestupný). Naprogramujeme si teda funkciu, ktorá nám vráti logickú pravdu (TRUE) alebo nepravdu (FALSE) podľa toho, či rok je alebo nie je prestupný:

```
function prestupny_rok(rok as long) as boolean
dim pom_prestupny as boolean
IF rok MOD 4 = 0 THEN
 ' rok môže, ale nemusí byť prestupný
 IF rok MOD 400 = 0 THEN
  pom_prestupny=true ' rok je prestupný, napr. 1600
 ELSE
  IF rok MOD 100 = 0 THEN
   pom_prestupny=false ' rok je neprestupný, napr. 1900
  ELSE
   pom_prestupny=true ' rok je prestupný, napr. 2004
  END IF
 END IF
ELSE
 pom_prestupny=false ' rok je neprestupný
END IF
prestupny_rok=pom_prestupny
end function
```

Pri programovaní takýchto postupností príkazov musíme byť opatrní, aby sme presne vedeli, ku ktorým príkazom IF patria vetvy ELSE. Ako vidíme, v jednej vetve ELSE sa nachádza ďalší vnorený príkaz IF. Pre trochu zjednodušené zapisovanie tohto prípadu nám ponúka jazyk StarOffice Basic špeciálny zápis ELSEIF (bez medzery), ktorým akoby sme spojili dva príkazy IF do jedného a teda na konci budeme mať iba jeden príkaz END IF:

```
function prestupny_rok(rok as long) as boolean
dim pom_prestupny as boolean
IF rok MOD 4 = 0 THEN
 ' rok môže, ale nemusí byť prestupný
 IF rok MOD 400 = 0 THEN
  pom_prestupny=true ' rok je prestupný, napr. 1600
  ELSEIF rok MOD 100 = 0 THEN
   pom_prestupny=false ' rok je neprestupný, napr. 1900
  ELSE
   pom_prestupny=true ' rok je prestupný, napr. 2004
 END IF
ELSE
 pom_prestupny=false ' rok je neprestupný
END IF
prestupny_rok=pom_prestupny
end function
```

Alebo, ak chceme, tak sa to dá naprogramovať ešte kratšie a jednoduchšie napr. takto:

```
function prestupny_rok(rok as long) as boolean
dim pom_prestupny as boolean
pom_prestupny=false ' predpokladáme, že rok je neprestupný
IF rok MOD 4 = 0 THEN
 ' rok môže, ale nemusí byť prestupný
 IF rok MOD 400 = 0 THEN
  pom_prestupny=true ' rok je prestupný, napr. 1600
 ELSEIF rok MOD 100 <> 0 THEN
  pom_prestupny=true ' rok je prestupný, napr. 2004
 END IF
END IF
prestupny_rok=pom_prestupny
end function
```

V týchto príkladoch sme si zároveň ukázali ďalší príkaz – FUNCTION a END FUNCTION. Funkcie poznáte všetci, napríklad funkcia sínus (SIN). Takéto funkcie si môžeme definovať aj sami s tým, že výsledok priraďujeme do jej názvu – v našom príklade je to predposledný príkaz: prestupny_rok=pom_prestupny

Funkcie následne voláme tak, že ich priradíme do nejakej premennej, napríklad:

```
dim je_prestupny as boolean
je_prestupny=prestupny_rok(2004)
```

To by na úvod k funkciám stačilo. Nebudeme si teraz uvádzať obecný tvar ich defi-

nície, pretože sme sa ešte nezoznámili s typmi premenných.

6.1.2. Príkaz SELECT

Niekedy sa pri programovaní stretneme s tým problémom, že potrebujeme naprogramovať celú kaskádu príkazov IF pre určitý rozsah jednej a tej istej premennej, napríklad vtedy, ak potrebujeme vetvenie podľa jednotlivých mesiacov:

```
IF mesiac="Január" THEN
 ' … príkazy, ak je mesiac január
ELSEIF mesiac="Február" THEN
 ' … príkazy, ak je mesiac február
ELSEIF mesiac="Marec" THEN
 ' … príkazy, ak je mesiac marec
ELSEIF mesiac="Apríl" THEN
 ' … príkazy, ak je mesiac apríl
ELSEIF mesiac="Máj" THEN
 ' … príkazy, ak je mesiac máj
ELSEIF mesiac="Jún" THEN
 ' … príkazy, ak je mesiac jún
ELSEIF mesiac="Júl" THEN
 ' … príkazy, ak je mesiac júl
ELSEIF mesiac="August" THEN
 ' … príkazy, ak je mesiac august
ELSEIF mesiac="September" THEN
 ' … príkazy, ak je mesiac september
ELSEIF mesiac="Október" THEN
 ' … príkazy, ak je mesiac október
ELSEIF mesiac="November" THEN
 ' … príkazy, ak je mesiac november
ELSEIF mesiac="December" THEN
 ' … príkazy, ak je mesiac december
ELSE
 ' … príkazy, ak je nesprávna hodnota premennej mesiac
END IF
```

Ako vidíme, takáto postupnosť je už dosť neprehľadná. V takýchto prípadoch môžeme použiť príkaz SELECT, ktorý vlastne nahradí uvedenú postupnosť príkazov a má nasledovný formát (počet vetiev CASE nie je obmedzený):

```
SELECT CASE premenna
 CASE hodnota_1:
  ' … príkazy, ak sa premenná rovná tejto hodnote
 CASE hodnota_2:
  ' … príkazy, ak sa premenná rovná tejto hodnote
 CASE rozsah_hodnot_1
  ' … príkazy, ak sa premenná rovná niektorej z uvedených
hodnôt
 CASE rozsah_hodnot_2
  ' … príkazy, ak sa premenná rovná niektorej z uvedených
hodnôt
```

```
 CASE ELSE
  ' … príkazy pri ostatných prípadoch
END SELECT
```

Vetva CASE ELSE môže, pravdaže, podobne ako vetva ELSE v príkaze IF úplne chýbať. Náš príklad s mesiacmi by sme mohli naprogramovať takto:

```
SELECT CASE mesiac
 CASE "Január":
  ' … príkazy, ak je mesiac január
 CASE "Február"
  ' … príkazy, ak je mesiac február
 CASE "Marec"
  ' … príkazy, ak je mesiac marec
 CASE "Apríl"
  ' … príkazy, ak je mesiac apríl
 CASE "Máj"
  ' … príkazy, ak je mesiac máj
 CASE "Jún"
  ' … príkazy, ak je mesiac jún
 CASE "Júl"
  ' … príkazy, ak je mesiac júl
 CASE "August"
  ' … príkazy, ak je mesiac august
 CASE "September"
  ' … príkazy, ak je mesiac september
 CASE "Október"
  ' … príkazy, ak je mesiac október
 CASE "November"
  ' … príkazy, ak je mesiac november
 CASE "December"
  ' … príkazy, ak je mesiac december
 CASE ELSE
  ' … príkazy, ak je nesprávna hodnota premennej mesiac
END SELECT
```

Ako ste si mohli všimnúť, v popise príkazu SELECT sme spomínali aj rozsah hodnôt. Pre predstavu uveďme vetvenie po štvrťrokoch:

```
SELECT CASE cislo_mesiaca
 CASE 1, 2, 3
  ' … príkazy pre prvý štvrťrok
 CASE 4 TO 6
  ' … príkazy pre druhý štvrťrok
 CASE 7, 8, 9
  ' … príkazy pre tretí štvrťrok
 CASE 10 TO 12
  ' … príkazy pre štvrtý štvrťrok
END SELECT
```

6.2. Cyklus

Cyklus je druhý (a posledný) typ príkazov programovacieho jazyka StarOffice Basic – ale nielen jeho, ale aj napr. jazyka Pascal, C a pod. Je to vlastne opakované vykonávanie jednej a tej istej časti programu. S cyklom sa stretávame úplne stále – napokon úplne každý deň robíme opakovane to isté – vstaneme, umyjeme sa, naraňajkujeme sa...

6.2.1. Príkaz DO

Príkaz cyklu DO môžeme v jazyku StarOffice Basic naprogramovať v štyroch tvaroch:

Variant 1

```
DO WHILE podmienka
  '… opakované príkazy
LOOP
```

Cyklus sa opakuje dovtedy, pokiaľ je splnená zadaná podmienka. V krajnom prípade – ak podmienka nie je nikdy splnená – sa príkazy vo vnútri cyklu nevykonajú ani raz. Tento príkaz môžeme použiť na náš príklad s vymazávaním viacnásobných medzier, kde ich potrebujeme opakovať dovtedy, pokiaľ vôbec nejaká zámena bola vykonaná. Na indikáciu stavu, že bola nejaká výmena bola vôbec vykonaná využijeme ich počet, ktorý v prípade vykonanej zámeny je nenulový.

```
SUB dvojita_medzera
 DIM Dokument, Vymena AS object
 DIM kolko, teraz AS Long
REM premenná kolko obsahuje celkový počet zámen, premenná
teraz počet zámen v jednom cykle
 Dokument=ThisComponent
 Vymena=Dokument.createReplaceDescriptor()
 Vymena.SearchString="  "
 Vymena.ReplaceString=" "
 teraz=Dokument.replaceAll(Vymena) ' prevedieme základnú
výmenu
 kolko=teraz
 DO WHILE teraz<>0 ' Opakujeme pokiaľ bola urobená nejaká
výmena
  ' aktuálny počet zámien uložíme do premennej teraz
  teraz=Dokument.replaceAll(Vymena) ' prevedieme opakovanú
výmenu
  REM spočítame úplne všetky zámeny
  kolko=kolko+teraz
 LOOP
 msgbox("Nahradených "+kolko+" dvojnasobnych
medzier.",0,"Viacnasobne medzery")
END SUB
```

Variant 2

```
DO UNTIL podmienka
  '… opakované príkazy…
LOOP
```

Cyklus sa ukončí vtedy, ak bude splnená zadaná podmienka. V krajnom prípade – ak podmienka je splnená úplne na začiatku – príkazy vo vnútri cyklu sa nevykonajú ani raz. Opakované vymazávanie medzier môžeme pomocou tohto príkazu naprogramovať napríklad takto:

```
SUB dvojita_medzera
 DIM Dokument, Vymena AS object
 DIM kolko, teraz AS Long
REM premenná kolko obsahuje celkový počet zámen, premenná
teraz počet zámen v jednom cykle
 Dokument=ThisComponent
 Vymena=Dokument.createReplaceDescriptor()
 Vymena.SearchString="  "
 Vymena.ReplaceString=" "
 teraz=Dokument.replaceAll(Vymena) ' prevedieme základnú
výmenu
 kolko=teraz
 DO UNTIL teraz=0 ' Skončíme ak nebola urobená žiadna
výmena
  ' aktuálny počet zámien uložíme do premennej teraz
  teraz=Dokument.replaceAll(Vymena) ' prevedieme opakovanú
výmenu
  REM spočítame úplne všetky zámeny
  kolko=kolko+teraz
 LOOP
 msgbox("Nahradených "+kolko+" dvojnasobnych
medzier.",0,"Viacnasobne medzery")
END SUB
```

Variant 3

```
DO
  ' … opakované príkazy
LOOP WHILE podmienka
```

Cyklus sa opakuje dovtedy, kým je splnená zadaná podmienka. Na rozdiel od prvých dvoch variant sa príkazy vo vnútri cyklu vykonajú minimálne jeden krát. Táto vlastnosť sa nám veľmi hodí, pretože aj vymazávanie viacnásobných medzier musíme urobiť aspoň raz.

```
SUB dvojita_medzera
 DIM Dokument, Vymena AS object
 DIM kolko, teraz AS Long
 Dokument=ThisComponent
 Vymena=Dokument.createReplaceDescriptor()
 Vymena.SearchString="  "
 Vymena.ReplaceString=" "
```

```
 kolko=0
 DO
  REM aktuálny počet zámien uložíme do premennej teraz
  teraz=Dokument.replaceAll(Vymena)
  REM spočítame úplne všetky zámeny
  kolko=kolko+teraz
  REM opakujeme kým aktuálny počet zámien je nenulový
 LOOP WHILE teraz<>0
 msgbox("Nahradených "+kolko+" dvojnasobnych
medzier.",0,"Viacnasobne medzery")
END SUB
```

Variant 4

```
  DO
   '… opakované príkazy
  LOOP UNTIL podmienka
```

Posledná verzia cyklusu DO sa ukončí vtedy, keď bude splnená zadaná podmienka. Tak isto ako v predchádzajúcej variante, aj teraz sa príkazy vo vnútri cyklu vykonajú minimálne jeden krát. Naše makro pre výmeny viacnásobných medzier môžeme teraz naprogramovať takto:

```
SUB dvojita_medzera
 DIM Dokument, Vymena AS object
 DIM kolko, teraz AS Long
 Dokument=ThisComponent
 Vymena=Dokument.createReplaceDescriptor()
 Vymena.SearchString="  "
 Vymena.ReplaceString=" "
 kolko=0
 DO
  REM počet zámien uložíme do premennej teraz
  teraz=Dokument.replaceAll(Vymena)
  REM spočítame všetky zámeny
  kolko=kolko+teraz
  REM opakujeme kým aktuálny počet zámien je nenulový
 LOOP UNTIL teraz=0
 msgbox("Nahradených "+kolko+" dvojnasobnych
medzier.",0,"Viacnasobne medzery")
END SUB
```

Na príklade jedného a toho istého algoritmu pre odstraňovanie viacnásobných medzier sme si ukázali štyri varianty príkazu DO. Pokiaľ si dobre všimnete drobné rozdiely v jednotlivých verziách týchto makier, ľahko pochopíte ich rozdielne správanie.

6.2.2. Príkaz FOR

Niekedy potrebujeme urobiť cyklus pre určitý rad hodnôt, ktoré sa pravidelne menia o určitý krok, pričom dopredu poznáme ich počiatočnú a koncovú hodnotu. Vtedy úspešne využijeme cyklus FOR:

```
  FOR premenna=pociatocna_hodnota TO koncova_hodnota STEP
```

```
krok
   ' … opakované príkazy
  NEXT premenna
```

Cyklus FOR je vlastne iba špeciálnym zjednodušeným zápisom cyklu DO, ktorý sa v krajnom prípade nevykoná ani raz:

```
  premenna=pociatocna_hodnota
  DO WHILE premenna<= koncova_hodnota
   ' … opakované príkazy
   premenna=premenna+krok
  LOOP
```

Tento cyklus môžeme napríklad použiť na výpočet n-faktoriálu a potom môžeme funkciu pre jeho výpočet naprogramovať napríklad takto:

```
function faktorial(n as long) as long
 dim pom_faktorial as long
 pom_faktorial=1
 FOR i = 2 TO n STEP 1
  pom_faktorial=pom_faktorial*i
 NEXT i
 faktorial=pom_faktorial
end function
```

V uvedenom príklade sa premenná i zväčšovala o 1. V takýchto prípadoch nemusí krok vôbec zadávať a celý príkaz môže vyzerať takto:

```
  pom_faktorial=1
  FOR i = 2 TO n
   pom_faktorial=pom_faktorial*i
  NEXT i
```

Samozrejme, krok nemusí byť len kladný, ale aj záporný:

```
  pom_faktorial=1
  FOR i = n TO 2 STEP -1
   pom_faktorial=pom_faktorial*i
  NEXT i
```

Ako premenné tohto cyklu však nemusia byť iba čísla, ale aj písmená. Napríklad cyklus pre všetky malé písmená abecedy by mohol vyzerať takto:

```
  FOR znak = "a" TO "z"
   ' … opakované príkazy
  NEXT i
```

Možnosti cyklu FOR však nekončia ani písmenami, pretože môžeme ako premennú použiť dokonca poradové indexy zo zoznamu hodnôt. Je to vlastne špeciálny prípad cyklusu číselného, kde počiatočná a koncová hodnota odkazuje na prvý a posledný prvok zo zoznamu. Napríklad cyklus pre všetky jednoznakové predložky a spojky by mohol aj z definíciou príslušného zoznamu vyzerať takto:

```
  DIM predlozky() AS STRING
  DIM i AS LONG
  predlozky() = ARRAY("a", "i", "k", "o", "s", "u", "v",
```

```
"z")
  FOR i = LBOUND(predlozky()) TO UBOUND(predlozky())
   ' … opakované príkazy
  NEXT i
```

6.2.3. Príkaz EXIT

Niekedy sa môže stať, že potrebujeme cyklus DO alebo FOR ukončiť predčasne – niekde uprostred jeho vykonávania. Presne na tento účel slúži príkaz EXIT, ktorý okamžite ukončí príslušný cyklus.

```
  FOR premenná = od TO po STEP krok
   '… príkazy
   EXIT FOR
   '… príkazy
  NEXT premenná
  DO
   '… príkazy
   EXIT DO
   '… príkazy
  LOOP
```

Príkaz EXIT využijeme napríklad vtedy, ak chceme zistiť, ktorý akademický titul sa vyskytuje ako prvý v aktuálnom dokumente. Takéto hľadanie môžeme naprogramovať napríklad takto:

```
SUB hladaj_titul
 DIM tituly() AS STRING
 DIM i AS INTEGER
 DIM ktory AS string
 DIM dokument, hladaj AS OBJECT
 tituly() = ARRAY("akad.", "Bc.", "Csc.", "doc.", "Dr.",
"DrSc.", "h.c.", "Ing.", "arch.", "JUDr.", "Mgr.", "MUDr.",
"MVDr.", "PaedDr.", "PharmDr.", "PhDr.", "PhMr.", "prof.",
"RNDR", "ThDr.")
 dokument=ThisComponent
 hladaj=Dokument.createReplaceDescriptor()
 ktory=""
 FOR i = LBOUND(tituly()) TO UBOUND(tituly())
  REM Na zistenie či sa hľadaný reťazec v dokumente
nachádza alebo nie použijeme „fintu"
  REM jeho zámeny za ten istý reťazec s tým, že zistíme
počet týchto „zámen“.
  hladaj.SearchString=tituly(i)
  hladaj.ReplaceString=tituly(i)
  if Dokument.replaceAll(Vymena)<>0 then
   ktory=tituly(i)
   EXIT FOR
  END IF
 NEXT i
 msgbox("Prvy titul je: "+ktory,0,"Hladanie akademickych
```

```
titulov")
END SUB
```

7. Typy premenných

V predchádzajúcom dieli tohto seriálu (Čo sme to naprogramovali?) sme naznačili, že premenná je akoby nejaká krabička, do ktorej ukladáme rôzne údaje,pričom každá krabička je určená iba na jeden typ údajov. A práve o týchto typoch budeme dnes hovoriť. Pre jednoduchosť budeme v nasledovnom (skôr technickom) popise týchto typov používať názov premennej „premenna“, pričom v prípade potreby uvedieme aj niekoľko príkladov alebo poznámok. Nebudeme sa však zaoberať ich praktickým použitím, napokon z príkladov makier, ktoré sme už programovali a ktoré ešte budeme programovať je to dosť zrejmé, ale obmedzíme sa naozaj iba na stručné zoznámenie.

7.1. Reťazcové premenné

Reťazec je vlastne textový údaj. StarOffice Basic ich ukladá v Unikóde,pričom dĺžka jedného reťazca je obmedzená na 65535 znakov. Pokiaľ chceme zadať reťazec priamo, musíme ho uzatvoriť do úvodzoviek. Niekedy však potrebujeme znak úvodzoviek mať priradený aj ako súčasť reťazca. Vtedy to musíme urobiť pomocou jeho ASCII kódu (34).

```
DIM premenna AS STRING
DIM premenna$ ' Skrátený formát definície
```

Príklady reťazcov:

```
premenna = "Príklad reťazca"
premenna = chr$(34)' priradenie úvodzovky do reťazca
```

7.2. Číselné premenné

StarOffice Basic rozoznáva päť rôznych typov číselných premenných, ktoré sa líšia najmä svojou veľkosťou a tým aj rozsahom čísiel, pre ktoré sú použiteľné.

Celé čísla v rozsahu od -32 768 do 32 767 (dva bajty)

```
DIM premenna AS INTEGER
DIM premenna% ' Skrátený formát definície
```

Celé čísla v rozsahu od -2 147 483 648 do 2 147 483 647 (štyri bajty)

```
DIM premenna AS LONG
DIM premenna& ' Skrátený formát definície
```

Reálne čísla v rozsahu od ± 1,401 298 x 10-45 do ± 3,402 823 x 1038 (štyri bajty)

```
DIM premenna AS SINGLE
DIM premenna! ' Skrátený formát definície
```

Reálne čísla v rozsahu od ± 4,940 656 458 412 47 x 10-324 do ± 1,797 693 134 862 32 x 10308 (osem bajtov)

```
DIM premenna AS DOUBLE
DIM premenna# ' Skrátený formát definície
```

Finančné čísla v rozsahu od -922 337 203 685 477,580 8 do 922 337 203 685 477,580 7 (osem bajtov)

```
DIM premenna AS CURRENCY
DIM premenna@ ' Skrátený formát definície
```

Finančné čísla majú špeciálny formát s pevným počtom štyroch desatinných miest a používajú sa pri presných finančných výpočtoch.

7.3. Logické premenné

Logické premenné môžu nadobúdať iba dve hodnoty – logickú nepravdu (nulu) – FALSE alebo logickú pravdu (jednotku) – TRUE.

```
DIM premenna AS BOOLEAN
```

7.4. Dátumové premenné

Dátumové premenné obsahujú hodnotu dátumu a času.

```
DIM premenna AS DATE
```

7.5. Dátové polia

Niekedy potrebujeme, aby premenná neobsahovala iba jednu hodnotu, ale celý rad (zoznam) hodnôt rovnakého typu. Na to slúžia dátove polia.

```
DIM premenna(rozsah_indexu, …) AS zakladny_typ
```

Príklad:

```
DIM mesiace(12) AS STRING
mesiace(1)="Január"
mesiace(2)="Február"
mesiace(3)="Marec"
mesiace(4)="Apríl"
mesiace(5)="Máj"
mesiace(6)="Jún"
mesiace(7)="Júl"
mesiace(8)="August"
mesiace(9)="September"
mesiace(10)="Október"
mesiace(11)="November"
mesiace(12)="December"
```

Polia nemusia mať iba jeden rozmer, napríklad pomocou tohto príkazu:

```
DIM matica (5, 4) AS DOUBLE
```

sme zadefinovali dvojrozmernú maticu čísiel. Pri praktickom programovaní mnohokrát potrebujeme, aby indexy nezačínali od jednotky, ale napríklad od nuly a pod. Na takéto účely môžeme použiť presnú špecifikáciu spodnej a hornej hranice indexu:

```
DIM pole (-10 TO 25) AS INTEGER
```

Niekedy však nemusíme vopred poznať počet prvkov poľa, ktoré špecifikujeme. Vtedy nemusíme rozsah indexu zadať vôbec. Takýto príklad sme mohli vidieť v minulom dieli nášho seriálu:

```
  DIM tituly() AS STRING
  tituly() = ARRAY("akad.", "Bc.", "Csc.", "doc.", "Dr.",
"DrSc.", "h.c.", "Ing.", "arch.", "JUDr.", "Mgr.", "MUDr.",
"MVDr.", "PaedDr.", "PharmDr.", "PhDr.", "PhMr.", "prof.",
"RNDR", "ThDr.")
```

Pre zistenie skutočného rozsahu indexov nám pri takýchto poliach slúžia systémové funkcie – LBOUND, ktorá vracia hodnotu (ako celé číslo) dolnej hranice indexu a UBOUND, ktorá vracia hodnotu (ako celé číslo) hornej hranice indexu:

```
  FOR i = LBOUND(tituly()) TO UBOUND(tituly())
```

7.6. Objektové premenné

Tieto premenné používame v našich makrách na riadenie prístupu k spracovávanému dokumentu. Obsahujú vlastne celú množinu údajov najrôznejších typov, ako je napríklad text, farba textu, farba pozadia, dátum, číslo strany a pod. Okrem toho je ich súčasťou aj množina operácií a funkcií, ktoré môžeme s týmito údajmi vykonávať, ako napríklad hľadanie, výmena a pod.

```
  DIM premenna AS OBJECT
```

7.7. Operátory

Operátory nám umožňujú prevádzať základné operácie s premennými. Delíme ich na tri skupiny.

7.7.1. Matematické operátory

`+` Sčítanie čísiel a dátumových údajov, spájanie reťazcov.

`-` Odčítanie čísiel a dátumových údajov.

`*` Násobenie čísiel.

`/` Delenie čísiel.

`\` Celočíselné delenie čísiel, t.j. výsledok je zaokrúhlený na celé číslo.

`^` Mocnina.

`MOD` Zvyšok po delení.

7.7.2. Logické operátory

`AND` Logický súčin.

`OR` Logický súčet.

`XOR` Exkluzívny logický súčet.

`NOT` Negácia.

`EQV` Ekvivalencia.

`IMP` Implikácia.

7.7.3. Porovnávacie operátory

= Rovnosť čísiel, dátumových hodnôt alebo reťazcov.

<> Nerovnosť čísiel, dátumových hodnôt alebo reťazcov.

> Väčší ako druhé číslo, dátumová hodnota alebo reťazec.

>= Väčší alebo rovný ako druhé číslo, dátumová hodnota alebo reťazec.

< Menší ako druhé číslo, dátumová hodnota alebo reťazec.

<= Menší alebo rovný ako druhé číslo, dátumová hodnota alebo reťazec.

8. (Skoro) naposledy mažeme medzery

Po predstavení príkazov a typov premenných sa znovu vrátime k makru pre mazanie viacnásobných medzier, ktoré sa budeme snažiť ešte zlepšiť a zjednodušiť.

Pri popise príkazu cyklu DO sme si uviedli ďalšie varianty makra pre odstraňovanie viacnásobných medzier. Na prvý pohľad by sme už mohli byť s výsledok spokojní – uvedené príklady makier naozaj vyriešia naše požiadavky. Na druhej strane si však musíme uvedomiť, že vo všetkých príkladoch makro opakovane prehliada celý dokument, čo môže byť časovo náročné.

Aby sme mohli zlepšiť tento stav, musíme zmeniť spôsob vyhľadávania. Našťastie OpenOffice.org podporuje vyhľadávanie pomocou regulárnych výrazov. Teraz sa snáď mnohí opýtajú, čo to vlastne regulárne výrazu sú. V jednoduchosti môžeme povedať, že je to vyhľadávanie pomocou zástupných znakov. Ako príklad môžeme použiť napr. príkaz DIR, ktorý pochádza ešte z operačného systému DOS, kde sme mohli vyhľadávať súbory pomocou tzv. hviezdičkovej konvencie. Hviezdičky vlastne nahrádzali ľubovoľný textový reťazec. Okrem toho sa mohol používať aj znak otáznika, ktorý nahrádzal presne jeden znak.

Samozrejme, možnosti príkazu DIR nie sú skutočné vyhľadávanie pomocou regulárnych výrazov, týmto príkladom sme chceli iba naznačiť problematiku.

```
SUB dvojita_medzera
 DIM Dokument, Vymena AS object
 DIM kolko AS Long
 Dokument=ThisComponent
 Vymena=Dokument.createReplaceDescriptor()
 ' Budeme hľadať popomocou regulárnych výrazov
 Vymena.SearchRegularExpression=True
 'Hľadáme medzeru, za ktorou sa nachádza ešte jedna a viac
medzier
 Vymena.SearchString="  +"
 Vymena.ReplaceString=" "
 kolko=Dokument.replaceAll(Vymena)
 msgbox("Nahradených "+kolko+" viacnásobných
medzier.",0,"Viacnásobne medzery")
END SUB
```

Ako vidíme, naše makro sa veľmi zjednodušilo. Pomocou príkazu:

```
Vymena.SearchRegularExpression=True
```

sme zabezpečili vyhľadávanie pomocou regulárnych príkazov, pričom pomocou reťazca " +" sme vyhľadali všetky viacnásobné medzery.

Zdá sa, že naše makro je už naozaj dobré a že na ňom už nemáme čo zlepšovať. Nie je to tak, pretože ešte má svoje nedostatky – nedokáže odstrániť napríklad nadbytočné medzery na začiatku alebo na konci odseku a pod. Pretože však používame regulárne výrazy, nebude nám teraz robiť problém naprogramovať aj tieto možnosti:

```
SUB dvojita_medzera
 DIM Dokument, Vymena AS object
 DIM kolko AS Long
 Dokument=ThisComponent
 Vymena=Dokument.createReplaceDescriptor()
 ' Budeme hľadať popomocou regulárnych výrazov
 Vymena.SearchRegularExpression=True
 'Hľadáme medzeru, za ktorou sa nachádza ešte jedna a viac
medzier
 Vymena.SearchString="  +"
 Vymena.ReplaceString=" "
 kolko=Dokument.replaceAll(Vymena)
 Vymena.SearchRegularExpression=True
 'Hľadáme zbytočné medzery na začiatku odseku
 Vymena.SearchString="^ *"
 ' ktoré vymažeme
 Vymena.ReplaceString=""
 kolko=kolko+Dokument.replaceAll(Vymena)
 Vymena.SearchRegularExpression=True

 'Hľadáme zbytočné medzery na konci odseku
 Vymena.SearchString=" *$"
 ' ktoré vymažeme
 Vymena.ReplaceString=""
 kolko=kolko+Dokument.replaceAll(Vymena)
 msgbox("Nahradených "+kolko+" viacnásobných
medzier.",0,"Viacnásobne medzery")
END SUB
```

9. Regulárne výrazy

Pri programovaní makra pre vymazávanie viacnásobných medzier sme postupne prešli viacerými fázami – od jednoduchej výmeny, ktorú sme však museli manuálne viackrát opakovať cez výmeny pomocou cyklov až po výmenu pomocou regulárnych výrazov. A práve regulárne výrazy môžu našu prácu mnohokrát veľmi uľahčiť a časovo skrátiť (a to, pravdaže, nielen pri programovaní). Už teraz môžeme naznačiť, že napríklad pri programovaní známeho makra „Vlnka“ pre vkladanie nezalomiteľných medzier za predložky bude mať použitie regulárnych výrazov naozaj veľmi zaujímavý časový efekt, pretože aj to dokážeme nakoniec naprogramovať na jeden prechod cez dokument – k tomu sa však dostaneme (podobne ako pri vymazávaní viacnásobných medzier) až neskôr v niektorých z ďalších dielov tohto seriálu.

Ako sme už spomínali, regulárne výrazy umožňujú vyhľadávanie pomocou zástupných znakov. Tieto výrazy obsahujú, pravdaže, aj bežné znaky tak, ako pri obyčajnom hľadaní, ibaže niektoré znaky (či celé postupnosti znakov) majú špecifický – zástupný význam. Pozrime sa teda bližšie práve na tieto špecifiká.

`.` – bodka predstavuje akýkoľvek jeden znak okrem konca riadku (odriadkovanie) a konca odseku. Napríklad pomocou regulárneho výrazu " . " vyhľadáme všetky jednoznakové reťazce (predložky, pomlčky a pod.).

`^` – pomocou striešky nájdeme hľadaný reťazec (ktorý je uvedený za ňou) iba vtedy, ak sa nachádza na začiatku odseku. Napríklad pomocou regulárneho výrazu "^ " dokážeme nájsť všetky jednoduché medzery na začiatku odseku.

`$` – pomocou doláru nájde hľadaný reťazec (ktorý je uvedený pred ním) iba vtedy, ak sa nachádza na konci odseku. Napríklad pomocou regulárneho výrazu " $" dokážeme nájsť všetky jednoduché medzery na konci odseku.

`^$` – touto postupnosťou hľadáme „prázdny“ reťazec, ktorý je zároveň na začiatku aj na konci odseku – teda prázdny odsek.

`^.` – touto postupnosťou vyhľadáme prvý znak odseku.

`*` – pomocou hviezdičky nájdeme žiadny alebo viacnásobný výskyt znaku, ktorý je uvedený pred ňou. Napríklad pomocou regulárneho výrazu "^ *" (použili sme ho v minulom dieli nášho seriálu) dokážeme nájsť všetky 0 až n-násobné medzery na začiatku odseku.

`+` – pomocou znaku plus nájdeme jednonásobný alebo viacnásobný výskyt znaku, ktorý je uvedený pred týmto znamienkom. Napríklad pomocou regulárneho výrazu "^ +" dokážeme nájsť všetky jednoduché a viacnásobné medzery na začiatku odseku.

`?` – pomocou otáznika nájdeme žiadny alebo práve jeden výskyt znaku, , ktorý je uvedený pred ním. Napríklad pomocou regulárneho výrazu " ?" dokážeme nájsť naraz všetky jednoduché a dvojnásobné medzery.

`\` – znak, ktorý sa nachádza za opačnou lomkou sa nepovažuje za špeciálny (zástupný) znak regulárneho výrazu, ale za obyčajný. Takto môžeme v regulárnych výrazoch vyhľadať napríklad znak bodky "\.", doláru "\$", opačnej lomky "\\" a pod. Výnimku tvoria postupnosti "\n", "\t", "\<", "\>", "\x" a referenčné odkazy "\1", "\2", ... ktoré majú špecifický význam.

`\n` – touto postupnosťou hľadáme zalomenie riadku (klávesová skratka Shift+Enter). Pokiaľ chceme nahradiť zalomenie riadku za zalomenie odseku, použijeme presne tento výraz aj v poli nahradiť.

`\t` – touto postupnosťou hľadáme znak tabelátora a môžeme ho použiť aj v poli nahradiť.

`\<` – pomocou tejto postupnosti nájdeme hľadaný reťazec (ktorý je uvedený za ňou) iba vtedy, ak sa nachádza na začiatku slova. Napríklad pomocou regulárneho výrazu "\<. " nájdeme všetky jednoznakové reťazce.

`\>` – pomocou tejto postupnosti nájdeme hľadaný reťazec (ktorý je uvedený pred ňou) iba vtedy, ak sa nachádza na konci slova. Napríklad pomocou regulárneho výrazu " .\>" nájdeme všetky jednoznakové reťazce.

`\xXXXX` – táto postupnosť predstavuje znak určený štvormiestnym hexadecimál-

nym kódom XXXX. Napríklad pomocou regulárneho výrazu "\x00A0" nájdeme všetky nezalomiteľné medzery.

() – zátvorky slúžia k zoskupovaniu znakov, ktoré sú potom považované akoby za jeden znak. Napríklad pomocou regulárneho výrazu "(text)+" nájdeme všetky jedno a viacnásobné výskyty reťazca „text“. Zároveň na výrazy v zátvorkách môžeme použiť aj referenčné odkazy, t.j. nemusíme ich opakovane písať celé. Napríklad pomocou regulárneho výrazu "(a)(b)c\1\2" nájdeme reťazec „abcab“.

| – zvislá čiara predstavuje logický výraz „alebo“, t.j. pomocou nej vyhľadáme reťazec, ktorý sa nachádza pred alebo za ňou. Napríklad pomocou regulárneho výrazu " (a|i|k|o|s|u|v|z) " vyhľadáme všetky jednoznakové predložky.

& – znak and má význam v poli nahradiť, kde pridá nájdený reťazec. Napríklad ak hľadáme reťazec "OpenOffice" a do pola nahradiť zadáme výraz "&.org", potom nahradíme všetky výskyty reťazca „OpenOffice“ za reťazec „OpenOffice.org“.

{koľko} – číslo v zložených zátvorkách určuje, koľkokrát sa má vyskytnúť znak, ktorý sa nachádza pred nimi. Napríklad pomocou regulárneho výrazu " {2}" nájdeme všetky dvojnásobné medzery.

{od, po} – rozsah dvoch čísiel v zložených zátvorkách určuje, koľkokrát (od až po) sa má vyskytnúť znak, ktorý sa nachádza pred nimi. Napríklad pomocou regulárneho výrazu " {2,5}" nájdeme všetky dvojnásobné až päťnásobné medzery.

{od,} – neohraničený rozsah čísiel v zložených zátvorkách určuje, minimálne koľkokrát (od) sa má vyskytnúť znak, ktorý sa nachádza pred nimi. Napríklad pomocou regulárneho výrazu " {2,}" nájdeme všetky viacnásobné medzery.

[zoznam znakov] – takýto zápis znakov uzatvorený v hranatých zátvorkách predstavuje jeden zo znakov, ktorý je uvedený v zozname. Napríklad pomocou regulárneho výrazu " [aikosuvz] " nájdeme všetky jednoznakové predložky. Ako zoznam môžeme použiť aj rozsah znakov, napríklad pomocou regulárneho výrazu " [a-z] " nájdeme všetky jednoznakové reťazce v ktorých je abecedný znak bez diakritiky. Zoznam môže byť aj kombinovaný, napríklad regulárny výraz "[a-kmo-z]" nájde všetky znaky medzi „a“ až „k“, znak „m“ a znaky medzi „o“ až „z“.

[^zoznam znakov] – takýto zápis znakov uzatvorený v hranatých zátvorkách predstavuje jeden zo znakov, ktorý nie je uvedený v zozname. Napríklad pomocou regulárneho výrazu " [^aikosuvz] " nájdeme všetky jednoznakové reťazce okrem predložiek. Ako zoznam môžeme použiť aj rozsah znakov a kombinácie, napríklad pomocou regulárneho výrazu "[^a-kmo-z]" vynecháme z hľadania znaky medzi „a“ až „k“, znak „m“ a znaky medzi „o“ až „z“.

Nasledujúcich osem postupností sa nemôže priamo použiť ako samostatné regulárne výrazy, ale tieto vždy musia obsahovať ešte aspoň jeden znak, inak OpenOffice.org nenájde ani jeden výskyt.

[:digit:] – táto postupnosť predstavuje desiatkové číslice.

[:alpha:] – táto postupnosť predstavuje abecedné znaky. Napríklad pomocou regulárneho výrazu " [:alpha:][:alpha:]? " nájdeme všetky jednoznakové a dvojznakové slová.

[:alnum:] – táto postupnosť predstavuje alfanumerické znaky, t.j. číslice a abecedné znaky spolu.

`[:space:]` – táto postupnosť predstavuje znaky tabelátora, zalomiteľných a nezalomiteľných medzier. Napríklad pomocou regulárneho výrazu "[:space:]?" nájdeme všetky viacnásobné medzery vrátane najrôznejších kombinácií s tabelátormi a nezalomiteľnými medzerami.

`[:print:]` – táto postupnosť predstavuje tlačiteľné znaky.

`[:cntrl:]` – táto postupnosť predstavuje netlačiteľné znaky.

`[:lower:]` – pokiaľ pri hľadaní rozoznávame veľkosť písmen, tak táto postupnosť predstavuje malé znaky, inak všetky abecedné znaky.

`[:upper:]` – pokiaľ pri hľadaní rozoznávame veľkosť písmen, tak táto postupnosť predstavuje veľké znaky, inak všetky abecedné znaky.

Skočili sme prvú veľkú časť seriálu o programovaní makier v OpenOffice.org. V novom roku budeme pokračovať druhou veľkou časťou, kde sa zameriame na formátovanie dokumentu (nezalomiteľné medzery za predložkami, akademickými titulmi a pod.).

Formátujeme dokumenty

10. Formátujeme dokumenty

Pri praktickom písaní sa mnohokrát stretávame s problémom potreby neskoršej automatickej úpravy textu podľa našich požiadavok. Pokiaľ píšete pomerne veľa (a to nielen ako ja recenzie či návody, ale aj romány, listy, dokumentáciu a pod.), postupom času zistíte, že OpenOffice.org v niektorých smeroch neposkytuje taký komfort ako iné, konkurenčné nástroje. Ako príklad spomeňme automatické vkladanie nezalomiteľných medzier za predložky. K tomu sa pridávajú ďalšie potreby, ktoré, samozrejme, už nemusia záležať od kancelárskeho balíka, pretože sú zamerané napríklad na automatické opravy preklepov, z ktorých znovu spomeňme opravu viacnásobných medzier. Všetky takéto úpravy môžeme súhrne označiť ako jednoduché formátovanie či upravovanie (opravovanie) dokumentu.

Prvé makro z tejto série sme už naznačili v predchádzajúcich dieloch nášho seriálu – makro pre odstraňovanie viacnásobných medzier. Aby sme „nevyšli z cviku“, začneme znovu týmto makrom, ktoré znovu trochu vylepšíme a pozmeníme.

10.1. Znovu odstraňujeme medzery

V poslednej verzii makra pre odstraňovanie viacnásobných medzier sme s využitím regulárnych výrazov dokázali v jednom kroku odstrániť všetky viacnásobné medzery, k čomu sme ešte pridali odstránenie medzier na začiatku a konci odseku. Pravdaže, ani toto riešenie ešte nezahŕňa všetky možnosti takýchto preklepov, pretože môžeme mať rôzne kombinácie nezalomiteľných a zalomiteľných medzier, tabelátorov s medzerami, nezalomiteľné medzery alebo tabelátory na začiatku alebo konci odseku a pod. Okrem toho sme medzery na začiatku alebo na konci odseku hľadali v dvoch krokoch. Pokiaľ zohľadníme tieto pripomienky, môže makro pre vymazávanie viacnásobných medzier vyzerať takto:

```
SUB Viacnasobna_medzera
 DIM Dokument, Vymena AS object
 DIM kolko AS Long
 DIM NM AS string

 Dokument=ThisComponent
 Vymena=Dokument.createReplaceDescriptor()

 NM=chr$(&HA0) ' Nezalomiteľná medzera (ctrl+space)
 ' Budeme hľadať popomocou regulárnych výrazov
 Vymena.SearchRegularExpression=True
 'Hľadáme viacnásobné zalomiteľné medzery
 Vymena.SearchString="  +"
 'A nahradíme ich jednoduchou medzerou
 Vymena.ReplaceString=" "
 kolko=Dokument.replaceAll(Vymena)
```

```
 Vymena.SearchRegularExpression=True
 'Hľadáme kombinácie zalomiteľná+nezalomiteľná+zalomiteľná
medzera,
 'nezalomiteľná+zalomiteľná a zalomiteľná+nezalomiteľná
medzera
 Vymena.SearchString=" "+NM+" |"+NM+" | "+NM
 ' ktoré nahradíme nezalomiteľnou medzerou
 Vymena.ReplaceString=NM
 kolko=kolko+Dokument.replaceAll(Vymena)

 'Hľadáme kombinácie medzera+tabelátor+medzera,
 'tabelátor+medzera a medzera+tabelátor
 Vymena.SearchString=" \t |\t | \t"
 ' ktoré nahradíme tabelátorom
 Vymena.ReplaceString="\t"
 kolko=kolko+Dokument.replaceAll(Vymena)

 Vymena.SearchRegularExpression=True
 'Hľadáme zbytočné medzery na začiatku alebo na konci
odseku
 Vymena.SearchString="^[:space:]*|[:space:]*$"
 ' ktoré vymažeme
 Vymena.ReplaceString=""
 kolko=kolko+Dokument.replaceAll(Vymena)

 Vymena.SearchRegularExpression=True
 'Hľadáme zbytočné tabelátory na začiatku alebo na konci
odseku
 Vymena.SearchString="^\t*|\t*$"
 ' ktoré vymažeme
 Vymena.ReplaceString=""
 kolko=kolko+Dokument.replaceAll(Vymena)

 msgbox("Nahradených "+kolko+" viacnásobných
medzier.",0,"Viacnásobne medzery")
END SUB
```

11. Funkcie a procedúry

V minulom dieli sme si ukázali makro pre vymazávanie viacnásobných medzier, kde sme pridali aj vymazávanie nepotrebných znakov na začiatku či konci odseku a pod. Pokiaľ si dobre všimnete toto makro, zistíte, že sa tam viac krát opakuje vlastne tá istá postupnosť týchto príkazov:

```
  Vymena.SearchRegularExpression=True
  Vymena.SearchString= … ' hľadaný reťazec
  Vymena.ReplaceString= … ' reťazec, ktorým nahrádzame
  kolko=Dokument.replaceAll(Vymena)
```

V prípade ďalších zámen (ktoré budeme ešte programovať) sa takýto program zby-

točne predlžuje a stráca prehľadnosť. Preto je vhodné, aby sme často opakované postupnosti rovnakých príkazov zadefinovali osobitne ako funkciu alebo procedúru, ktoré spoločným názvom nazývame aj podprogramy. Procedúru už vlastne používame, pretože aj už naprogramované makrá pre výmenu viacnásobných medzier sú vlastne procedúry. Je to vlastne akoby definícia nového príkazu. Samotná definícia má takýto obecný tvar:

```
SUB názov(zoznam parametrov)
 '…
END SUB
```

Funkcie nám tiež nie sú neznáme, pretože poznáme napr. matematické funkcie a pod. V našom makre pre výmenu viacnásobných medzier sme používali napr. systémovú funkciu Dokument.replaceAll. Samotná definícia má takýto obecný tvar:

```
FUNCTION nazov(zoznam_parametrov) AS typ_vysledku
 '…
 nazov=vysledok
END FUNCTION
```

Ako vidíme, vo funkcii musíme mať aspoň jeden špeciálny priraďovací príkaz, ktorým priradíme jej výsledok do názvu funkcie. Dosť však bolo teórie, pozrime sa na naše makro pre výmenu viacnásobných medzier. Uvedené štyri riadky by sme mohli naprogramovať potom takto:

```
FUNCTION Vymen(Co, Za as string, Regularne as Boolean) as
long

 rem Funkcia vymení reťazec Co za reťazec Za a vráti počet
výmien
 rem Regularne - určuje, či vyhľadávaný výraz je (TRUE)
alebo nie je (FALSE) regulárny výraz
 rem Ako výsledok bude počet zámen

 Vymena.SearchString=Co
 Vymena.ReplaceString=Za
 Vymena.SearchRegularExpression=Regularne
 Vymen=Dokument.replaceAll(Vymena) ' Do názvu funkcie
priraďujeme počet výmen

END FUNCTION
```

Vlastnú procedúru pre zámenu môžeme teraz naprogramovať takto:

```
SUB Viacnasobna_medzera

 DIM kolko AS Long
 DIM NM AS string
 NM=chr$(&HA0) ' Nezalomiteľná medzera (ctrl+space)

 ' Výmena viacnásobných medzier za jednoduchú
 kolko=Vymen("  +"," ",TRUE)
 ' Kombinácie nezalomiteľných a zalomiteľných medzier
```

```
 kolko=kolko+Vymen(" "+NM+" |"+NM+" | "+NM,NM,TRUE)
 ' Kombinácie tabelátorov a medzier
 kolko=kolko+Vymen(" \t |\t | \t","\t",TRUE)
 ' Medzery na začiatku a konci odseku
 kolko=kolko+Vymen("^[:space:]*|[:space:]*$","",TRUE)
 ' Tabelátory na začiatku a konci odseku
 kolko=kolko+Vymen("^\t*|\t*$","",TRUE)

 msgbox("Nahradených "+kolko+" viacnásobných
medzier.",0,"Viacnásobne medzery")

END SUB
```

Ako vidíme, makro sa naozaj zjednodušilo a obsahuje okrem poznámok presne deväť riadkov. Pravdaže, dá sa zapísať ešte kratšie – na päť riadkov s použitím jedného jediného priraďovacieho príkazu, ale to je už trochu na úkor čitateľnosti a prehľadnosti:

```
SUB Viacnasobna_medzera

 DIM kolko AS Long
 DIM NM AS string
 NM=chr$(&HA0) ' Nezalomiteľná medzera (ctrl+space)

 kolko=Vymen("  +"," ",TRUE)+Vymen(" "+NM+" |"+NM+" |
"+NM,NM,TRUE)+Vymen(" \t |\t | \t","\t",TRUE)
+Vymen("^[:space:]*|[:space:]*$","",TRUE)
+Vymen("^\t*|\t*$","",TRUE)

 msgbox("Nahradených "+kolko+" viacnásobných
medzier.",0,"Viacnásobne medzery")

END SUB
```

Alebo pre absolutistov ešte kratšie na tri riadky:

```
SUB Viacnasobna_medzera

 DIM NM AS string
 NM=chr$(&HA0) ' Nezalomiteľná medzera (ctrl+space)

 msgbox("Nahradených "+Vymen("  +"," ",TRUE)+Vymen(" "+NM+"
|"+NM+" | "+NM,NM,TRUE)+Vymen(" \t |\t | \t","\t",TRUE)
+Vymen("^[:space:]*|[:space:]*$","",TRUE)
+Vymen("^\t*|\t*$","",TRUE)+" viacnásobných
medzier.",0,"Viacnásobne medzery")
END SUB
```

Alebo, ak chcete, tak absolútne na jeden riadok:

```
SUB Viacnasobna_medzera
 msgbox("Nahradených "+Vymen("  +"," ",TRUE)+Vymen(" "+chr$
```

```
(&HA0)+" |"+chr$(&HA0)+" | "+chr$(&HA0),chr$(&HA0),TRUE)
+Vymen(" \t |\t | \t","\t",TRUE)+Vymen("^[:space:]*|
[:space:]*$","",TRUE)+Vymen("^\t*|\t*$","",TRUE)+"
viacnásobných medzier.",0,"Viacnásobne medzery")
END SUB
```

Posledné úpravy však nevedú (ak vôbec vedú) k zrýchleniu podprogramu a sú preto určené iba ako ukážka možností volania funkcií s prípadným šetrením pamäte, ktoré sme získali definíciou menšieho (alebo nulového) počtu premenných.

Ak sa pozrieme na zadefinované podprogramy zistíme, že sme kdesi „stratili“ definíciu premennej Vymena a vlastne nikde nepriraďujeme výmenu k aktuálnemu dokumentu. Niečo nám tam teda ešte chýba. Najprv si však povedzme, ako je to vôbec s premennými – ktoré poznáme iba vo vnútri podprogramu a ktoré aj mimo? Je to jednoduché – všetko záleží od miesta, kde ich definujeme. Ak zadefinujeme premenné iba vo vnútri podprogramu, potom ich poznáme iba tam – môžeme ich nazvať lokálne premenné. Ak ich zadefinujeme „vonku“, poznáme ich všade – môžeme ich nazvať ako globálne, alebo spoločné premenné.

Takže v tomto zmysle teraz zadefinujeme chýbajúce premenné a následne priradíme aktuálny dokument. Celé makro zo všetkými funkciami a procedúrami potom môžeme naprogramovať napríklad takto:

```
REM Definície spoločných premenných sú úplne pred
definíciou podprogramov

dim Dokument, Vymena as object
' Nezalomiteľnú medzeru budeme používať aj v iných makrách,
preto presunieme definíciu sem
dim NM as string

' V osobitnej procedúre inicializujeme spoločné premenné.
' Túto procedúru musíme potom volať ako prvú, inak
neurobíme potrebné priradenia.
sub Init

 rem Inicializácia spoločných premenných
 Dokument=ThisComponent
 Vymena=Dokument.createReplaceDescriptor()
 NM=chr$(&HA0) ' Nezalomiteľná medzera - hexadecimálny kód
A0

end sub

FUNCTION Vymen(Co, Za as string, Regularne as Boolean) as
long
 rem Funkcia vymení reťazec Co za reťazec Za a vráti počet
výmien
 rem Regularne - určuje, či vyhľadávaný výraz je (TRUE)
alebo nie je (FALSE) regulárny výraz
 rem Ako výsledok bude počet zámen
```

```
 Vymena.SearchString=Co
 Vymena.ReplaceString=Za
 Vymena.SearchRegularExpression=Regularne
 Vymen=Dokument.replaceAll(Vymena) ' Do názvu funkcie
priraďujeme počet výmen
END FUNCTION
SUB Viacnasobna_medzera

 DIM kolko AS Long

' Inicializácia spoločných premenných
 Init

 ' Výmena viacnásobných medzier za jednoduchú
 kolko=Vymen("  +"," ",TRUE)
 ' Kombinácie nezalomiteľných a zalomiteľných medzier
 kolko=kolko+Vymen(" "+NM+" |"+NM+" | "+NM,NM,TRUE)
 ' Kombinácie tabelátorov a medzier
 kolko=kolko+Vymen(" \t |\t | \t","\t",TRUE)
 ' Medzery na začiatku a konci odseku
 kolko=kolko+Vymen("^[:space:]*|[:space:]*$","",TRUE)
 ' Tabelátory na začiatku a konci odseku
 kolko=kolko+Vymen("^\t*|\t*$","",TRUE)

 msgbox("Nahradených "+kolko+" viacnásobných
medzier.",0,"Viacnásobne medzery")
END SUB
```

Toto je minule sľúbené makro pre výmenu viacnásobných medzier na osem riadkov. A hoci to tak nevyzerá, ešte stále to nie je jeho posledná verzia. Napokon, stále platia staré programátorské zákony – v každom programe je aspoň jedna chyba a v každom programe je aspoň jeden riadok zbytočný (len to neberte absolútne, lebo potom by sme nemali čo programovať :-).

12. Nezalomiteľná medzera

V dnešnom dieli nášho seriálu začneme rozširovať funkcie, ktorými budeme formátovať dokument. A začneme hneď jedným z najviac žiadaných makier, pomocou ktorého budeme podľa typografických pravidiel zamieňať zalomiteľné medzery za nezalomiteľné.

V minulom dieli sme si naprogramovali funkciu „Vymen“ a procedúru „Init“, pomocou ktorých sme následne značne zjednodušili naše makro pre opravu viacnásobných medzier. V nasledujúcich dieloch nášho seriálu tieto podprogramy už budeme štandardne využívať, takže sa k nim budeme vracať iba v nevyhnutných prípadoch, keď ich budeme meniť – teda napríklad aj v dnešnom dieli.

12.1. Nezalomiteľné medzery podľa typografických pravidiel

Pozrime sa teraz bližšie na ďalší problém, z ktorým sa pri praktickej práci stretávame. Je to ten fakt, že OpenOffice.org nevkladá automaticky nezalomiteľné medzery napr. za jednoznakové predložky. Toto však odporuje zásadám typografie, v ktorých sa o.i. hovorí, že k zalomeniu riadku nemá dôjsť pred jednociferným číslom bez bernej jednotky (napríklad „1 človek“), medzi rádovými skupinami cifier vo vnútri čísla alebo pri telefónnych číslach (napr. „1 587 235“), medzi číslom a značkou mernej jednotky (napr. „123 km“), medzi skratkami akademických titulov, vojenskými a vedeckými hodnosťami a menom (napr. „Ing. Pastierik“), medzi skratkou krstného mena a priezviskom (napr. „J. Pastierik“), za číslicami v dátumoch (napr. „16. 1. 2006“) a ani za jednoznakovou predložkou alebo spojkou.

V prípade predložiek a spojok je otázka nezalomiteľných medzier trochu nejednoznačná. Ide o to, že niektoré zdroje (napr. http://www.typo.cz) uvádzajú, že spojka „a“ (ale nie verzálka „A“) je výnimka a ďalej, že v prípade úzkej sadzby (do 25 znakov) sú možné výnimky aj za ostatnými jednoznakovými predložkami a spojkami. Iné pramene popisujú ako výnimku iba spojku „a“ a aj to iba v prípade úzkej sadzby. Pretože rozhodnutie leží na každom používateľovi, budeme sa snažiť vyriešiť otázku okolo spojky „a“ tak, aby si každý mohol urobiť jednoduchú úpravu podľa svojich požiadavok.

12.2. Nezalomiteľné medzery za spojkami

Ako vidíme, problematika nezalomiteľných medzier je pomerne široká a preto sa ňou budeme zaoberať postupne. Ako prvé, čo si budeme v spomínanej oblasti programovať, bude makro pre zámenu zalomiteľných medzier za jednoznakovými predložkami a spojkami, pretože toto je najčastejšie požadované od používateľov OpenOffice.org. Jednoduchá verzia tohto makra by mohla vyzerať takto (predložky rozpoznáme tak, že pred nimi a za nimi je medzera):

```
sub nezalomitelne_Spojky

 dim predlozky() as string
 dim i, kolko as long

 predlozky()=array("a","i","k","o","s","u","v","z")

 kolko=0

 REM pre všetky predložky
 for i=lbound(predlozky()) to ubound(predlozky())
  kolko=kolko+vymen(" "+predlozky(i)+" "," "+predlozky(i)
+NM,false)
 next i

 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné medzery")
end sub
```

Podobne, ako v prípade prvej verzie makra pre mazanie viacnásobných medzier, aj

toto makro má veľa nedostatkov. Ten najhlavnejší je však ten, že ak máme spojku napísanú veľkými písmenami, tak ju zamení za malé písmeno. Je jasné, že pri zámene musíme rozpoznávať veľké a malé písmena. Na tento účel si preto najprv musíme upraviť funkciu „Vymen“. Aby sme jej úpravu nemuseli robiť tak často, hneď si uvedieme aj ďalší parameter, pomocou ktorého budeme rozpoznávať, či hľadáme celé slová, alebo iba ich časti:

```
function Vymen(Co, Za as string, Regularne, Cele_slova,
Velke_pismena as Boolean) as long
 rem Funkcia vymení reťazec Co za reťazec Za a vráti počet
výmien
 rem Regularne - určuje, či vyhľadávaný výraz je (true)
alebo nie je (false) regulárny výraz
 rem Cele_slova - určuje, či vyhľadávaný výraz je (true)
alebo nie je (false) celé slovo
 rem Velke_pismena - určuje, či sa majú (true) alebo nemajú
(false) rozoznávať veľké a malé písmená

 Vymena.SearchString=Co
 Vymena.ReplaceString=Za
 Vymena.SearchRegularExpression=Regularne
 Vymena.SearchWords=Cele_slova
 Vymena.SearchCaseSensitive=Velke_pismena
 Vymen=Dokument.replaceAll(Vymena)
end function
```

Pravdaže, teraz si musíme upraviť aj makro pre zámenu viacnásobných medzier, pretože sme tam nepočítali s novými parametrami:

```
SUB Viacnasobna_medzera

 DIM kolko AS Long

' Inicializácia spoločných premenných
 Init

 ' Výmena viacnásobných medzier za jednoduchú
 kolko=Vymen("  +"," ",TRUE,false,false)
 ' Kombinácie nezalomiteľných a zalomiteľných medzier
 kolko=kolko+Vymen(" "+NM+" |"+NM+" |
"+NM,NM,TRUE,false,false)
 ' Kombinácie tabelátorov a medzier
 kolko=kolko+Vymen(" \t |\t | \t","\t",TRUE,false,false)
 ' Medzery na začiatku a konci odseku
 kolko=kolko+Vymen("^[:space:]*|
[:space:]*$","",TRUE,false,false)
 ' Tabelátory na začiatku a konci odseku
 kolko=kolko+Vymen("^\t*|\t*$","",TRUE,false,false)

 msgbox("Nahradených "+kolko+" viacnásobných
```

```
medzier.",0,"Viacnásobne medzery")

END SUB
```

Makro pre zámenu zalomiteľných medzier za predložkami a spojkami by teraz mohlo vyzerať takto:

```
sub nezalomitelne_Spojky

 dim predlozky() as string
 dim i, kolko as long

 predlozky()=array("a","i","k","o","s","u","v","z")

 kolko=0

 REM pre všetky predložky
 for i=lbound(predlozky()) to ubound(predlozky())
  rem malé predložky
  kolko=kolko+vymen(" "+predlozky(i)+" "," "+predlozky(i)
+NM,false,false,TRUE)
  rem veľké predložky - funkcia uCase prevedie malé znaky
na veľké
  kolko=kolko+vymen(" "+ucase(predlozky(i))+" ","
"+ucase(predlozky(i))+NM,false,false,TRUE)
 next i

 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné medzery")
end sub
```

V tejto verzii makra sa však vkladá nezalomiteľná medzera aj za spojku „a“, čo nie každému vyhovuje. Tento stav sa dá ošetriť napríklad tak, že budeme túto výnimku testovať:

```
sub nezalomitelne_Spojky
 dim predlozky() as string
 dim i, kolko as long

 predlozky()=array("a","i","k","o","s","u","v","z")

 kolko=0

 REM pre všetky predložky
 for i=lbound(predlozky()) to ubound(predlozky())
  rem malé predložky
  if predlozky(i)<>"a" then
   kolko=kolko+vymen(" "+predlozky(i)+" "," "+predlozky(i)
+NM,false,false,TRUE)
  end if
  rem veľké predložky - funkcia uCase prevedie malé znaky
```

```
na veľké
  kolko=kolko+vymen(" "+ucase(predlozky(i))+" ","
"+ucase(predlozky(i))+NM,false,false,TRUE)
 next i

 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné medzery")
end sub
```

Alebo, ešte jednoduchšie, vymenovaním všetkých malých aj veľkých spojok a predložiek osobitne:

```
sub nezalomitelne_Spojky

 dim predlozky() as string
 dim i, kolko as long

 predlozky()=array("A","i","I","k","K","o","O","s","S","u",
"U","v","V","z","Z")

 kolko=0

 REM pre všetky predložky
 for i=lbound(predlozky()) to ubound(predlozky())
  kolko=kolko+vymen(" "+predlozky(i)+" "," "+predlozky(i)
+NM,false,false,TRUE)
 next i

 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné medzery")
end sub
```

Ten, kto chce vkladať aj za spojku „a“ nezalomiteľnú medzeru, si ju môže do zoznamu jednoducho doplniť. Ani toto riešenie nie je však úplne dobré, pretože niekto môže chcieť, aby sa za touto spojku nevkladala nezalomiteľná medzera iba v prípade úzkej sadzby. Toto však budeme riešiť až niekedy inokedy.

12.3. Znovu používame regulárne výrazy

Makro pre vkladanie nezalomiteľných medzier za predložky, ktoré sme si naprogramovali naposledy, má niekoľko nedostatkov. Prvý, ktorý si môžeme všimnúť, je ten, že nevkladá medzery za tie predložky, ktoré sa nachádzajú na začiatku odseku. Pravdaže, tento nedostatok nemusí byť až taký závažný, pretože na tomto mieste nedochádza k nevhodnému zalomeniu. Na druhej strane, ak zlúčime dva odseky do jedného, tento problém sa už vyskytnúť môže. Môžeme to však jednoducho vyriešiť pomocou regulárnych výrazov, napríklad takto:

```
sub nezalomitelne_Spojky

 dim predlozky() as string
 dim i, kolko as long
```

```
 predlozky()=array("A","i","I","k","K","o","O","s","S","u",
"U","v","V","z","Z")

 kolko=0

 REM pre všetky predložky
 for i=lbound(predlozky()) to ubound(predlozky())
  kolko=kolko+vymen(" "+predlozky(i)+" "," "+predlozky(i)
+NM,false,false,TRUE)
  REM Vyhľadáme predložky aj na začiatku odseku
  kolko=kolko+vymen("^"+predlozky(i)+" ",predlozky(i)
+NM,TRUE,false,TRUE)
 next i

 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné medzery")
end sub
```

12.4. Ale spojky a predložky nebývajú vždy osamotené

Pozrime sa však na druhý problém, ktorý nám pri používaní tohto makra vznikol – v prípade, že máme v texte viacej jednoznakových spojok a predložiek tesne za sebou, nezamení všetky potrebné medzery. Takéto prípady sú pomerne časté a nie je problém sa stretnúť aj s tromi predložkami a spojkami za sebou, napr. v texte „... a i s tým sa musíme vyrovnať ...“. Je to vlastne preto, lebo konkrétne v uvedenom príklade sa za spojku „a“ vloží nezalomiteľná medzera a preto v ďalšom kroku pri teste spojky „i“ túto už nenájde. V makre totiž hľadáme reťazec zalomiteľná medzera + spojka + zalomiteľná medzera a my tam v skutočnosti máme reťazec nezalomiteľná medzera + spojka + zalomiteľná medzera. Naše makro preto rozšírime o ďalší test:

```
sub nezalomitelne_Spojky

 dim predlozky() as string
 dim i, kolko as long

 predlozky()=array("A","i","I","k","K","o","O","s","S","u",
"U","v","V","z","Z")

 kolko=0

 REM pre všetky predložky
 for i=lbound(predlozky()) to ubound(predlozky())
  kolko=kolko+vymen(" "+predlozky(i)+" "," "+predlozky(i)
+NM,false,false,TRUE)
  REM pred predložkou môže byť aj nezalomiteľná medzera
  kolko=kolko+vymen(NM+predlozky(i)+" ",NM+predlozky(i)
+NM,false,false,TRUE)
  REM Vyhľadáme predložky aj na začiatku odseku
  kolko=kolko+vymen("^"+predlozky(i)+" ",predlozky(i)
```

```
+NM,TRUE,false,TRUE)
 next i

 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné medzery")
end sub
```

Toto makro už funguje správne. Ako však vidíme, každú jednoznakovú spojku alebo predložku testujeme až tri krát. Je to tak preto, lebo sme si definovali jednoznakové spojky a predložky ako reťazec medzera + písmeno + medzera. Predložky však môžeme definovať aj inak – sú to vlastne jednoznakové slová, t.j. hľadané písmeno sa musí zároveň nachádzať na začiatku slova. Na rad teda znovu prichádzajú regulárne výrazy:

```
sub nezalomitelne_Spojky

 dim predlozky() as string
 dim i, kolko as long

 predlozky()=array("A","i","I","k","K","o","O","s","S","u",
"U","v","V","z","Z")

 kolko=0

 REM pre všetky predložky
 for i=lbound(predlozky()) to ubound(predlozky())
  REM hľadáme jednoznakové spojky, ktoré sú na začiatku
slova
  kolko=kolko+vymen("\<"+predlozky(i)+" "," "+predlozky(i)
+NM,TRUE,false,TRUE)
 next i

 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné medzery")
end sub
```

Toto makro rieši v jednom teste všetky problémy, ktoré sme dnes načrtli, pretože jednoznakové predložky a spojky, ktoré sa nachádzajú na začiatku odseku sa zároveň nachádzajú aj na začiatku slova. Zároveň sa týmto makrom rieši aj ten problém, keď pred jednoznakovou spojku a predložkou nie je medzera, ale tabelátor.

13. Ďalšie formátovacie funkcie

V doterajších dieloch nášho seriálu sme sa venovali dvom základným problémom – vymazávaniu viacnásobných medzier a vkladaniu nezalomiteľných medzier za jednoznakové predložky a spojky. Pravdaže, týmito príkladmi sme nevyriešili všetky problémy, ktoré sú spojené s touto problematikou. Napríklad, pod odstraňovanie viacnásobných medzier by sme mohli zaradiť aj odstraňovanie zbytočných medzier, resp. takých medzier, ktoré nezodpovedajú typografickým pravidlám. Tu patria napr. nesprávne vložené medzery medzi úvodzovky a text, zátvorky a text a pod. Podobne sme ešte nevyriešili celú problematiku okolo nezalomiteľných medzier, pretože tie sa

vkladajú (ako sme už spomínali v tretej časti) aj medzi skratkami akademických titulov, za číslicami v dátumoch a pod. A práve tejto problematike sa teraz budeme postupne venovať.

13.1. „Nezalomiteľné“ akademické tituly

Ako prvej sa budeme teraz venovať problematike vkladania nezalomiteľných medzier medzi akademické tituly a meno. Hoci je problematika podobná jednoznakovým predložkám, má svoje osobitnosti, ktoré si musíme rozobrať. Na rozdiel od predložiek má väčšina titulov pevný tvar veľkých a malých písmen (napr. magister farmácie – „PhMr.“). Nemusíme sa však zaoberať tým, ako boli napísané, ale ich vždy prepíšeme správnym tvarom, čím zároveň dosiahneme aj to, že automaticky opravíme ich prípadný nesprávny zápis. Toto však neplatí úplne pre všetky tituly, pretože skratky titulov akademik – „akad.“, architekt – „arch.“, docent – „doc.“ a profesor – „prof.“ sa píšu s malými písmenami, pravdaže, okrem prípadu, keď sa nachádzajú na začiatku vety (vtedy je prvé písmeno veľké).

Okrem titulov, ktoré sa uvádzajú pred menom existujú aj tituly, ktoré sa uvádzajú za menom, t.j. pri ktorých musíme vkladať nezalomiteľnú medzeru pred ne (napr. kandidát vied – „CSc.“). Okrem toho, pri tituloch za menom existuje možný dvojtvar pre skratku titulu čestného doktorátu (honoris causa) – h.c. (bez medzery)[2] alebo h. c. (s nezalomiteľnou medzerou).

Pri programovaní už nebudeme uvádzať riešenia, ktoré nehľadajú potrebné reťazce na začiatku alebo konci odseku, pretože sme si túto problematiku už dosť objasnili v predchádzajúcich dieloch nášho seriálu. Pri tituloch sa však stretávame s novým problémom – vždy končia bodkou, ktorú musíme zahrnúť do vyhľadávania. Vzhľadom na to, že budeme hľadať pomocou regulárnych výrazov, kde má bodka špecifické postavenie (nahrádza akýkoľvek znak), musíme ju zadávať kombináciu znakov „\.“ a z tohto dôvodu ju nebudeme v definícii titulov uvádzať.

V tejto súvislosti nám ešte spôsobia problémy skratky „h. c.“ a „h.c.“, pretože tu sa bodka nachádza aj v ich vnútri, a preto ich musíme riešiť úplne osobitne. Makro, kde sú zahrnuté všetky uvedené problémy, by mohlo vyzerať potom takto:

```
sub Nezalomitelne_tituly
 REM program prevedie zámenu zalomiteľnej medzery medzi
menom a titulom
 REM zároveň sa opravia nesprávne napísané tituly okrem
titulov
 REM akad., arch., doc., prof., ktoré sa píšu malým
písmenom,
 REM ale na začiatku vety majú prvé písmeno veľké, t.j.
Akad., Arch., Doc., Prof.

 dim Tituly_pred(), Tituly_pred_spec(), Tituly_za() as
string
 dim i, kolko as long

 REM Akademické tituly pred menom
```

2 Tvar bez medzery „h.c.“ nie je správny, ale o tom píšeme neskôr.

Ďalšie formátovacie funkcie

```
 Tituly_pred()=array("Bc", "Dr", "Ing", "JUDr", "Mgr",
"MUDr", "MVDr", "PaedDr", "PharmDr", "PhDr", "PhMr",
"RNDr", "ThDr")

 REM Akademické tituly pred menom, kde sú rôzne prvé
písmená
 Tituly_pred_spec()=array("akad", "Akad", "arch", "Arch",
"doc", "Doc", "prof", "Prof")

 REM Akademické tituly za menom
 Tituly_za()=array("CSc","DrSc")

 REM Mimo ostatných skratiek je skratka h.c.
 REM Najprv vložíme nezalomiteľnú medzeru do skratiek h. c.
 Kolko=Vymen("h. c.","h."+NM+"c.",false,false,false)
 REM nezalomiteľná medzera pred skratku h.c. bez medzier
 Kolko=Vymen(" h.c.",NM+"h.c.",false,false,false)
 REM nezalomiteľná medzera pred skratku h. c. s medzerou
 Kolko=Vymen("
h."+NM+"c.",NM+"h."+NM+"c.",false,false,false)

 REM Pre všetky tituly pre menom, ktoré sa zároveň opravujú
na správny tvar
 for i=lbound(Tituly_pred()) to ubound(Tituly_pred())
  kolko=kolko+Vymen("\<"+Tituly_pred(i)+"\. ",Tituly_pred(i)
+"."+NM,true,false,false)
 next i

 REM Pre všetky tituly pred menom, ktoré sa neopravujú
 for i=lbound(Tituly_pred_spec()) to
ubound(Tituly_pred_spec())
  kolko=kolko+Vymen("\<"+Tituly_pred_spec(i)+"\.
",Tituly_pred_spec(i)+"."+NM,true,false,true)
 next i

REM Pre všetky tituly, ktoré sú za menom
 for i=lbound(Tituly_za()) to ubound(Tituly_za())
  kolko=kolko+Vymen(" "+Tituly_za(i)
+".",NM+Tituly_za(i),false,false,false)
 next i

 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné tituly")
end sub
```

Ako vidíme, toto makro je na rozdiel od nezalomiteľných medzier za predložkami pomerne zložité, no napriek tomu dúfame, že ešte stále pochopiteľné a prehľadné. Vieme, že v ňom nie sú uvedené všetky možné tituly (napr. chýbajú vojenské hodnosti), predpokladáme, že tí, ktorí s nimi v texte pracujú si ich určite ľahko doplnia.

Okrem toho sa môže do makra doplniť aj výmena dvoch rôznych tvarov titulov čestného doktorátu za jeden – ten, ktorý sa komu viac páči. Toto však necháme prípadným záujemcom ako domácu úlohu.

14. Vlastná funkcia pre výmeny

V minulom dieli sme uverejnili makro pre vkladanie nezalomiteľných medzier za akademické tituly. Dnes tieto možnosti rozšírime o ďalšie typografické pravidlá, pričom si naprogramujeme vlastnú funkciu pre zamieňanie nájdeného textu.

14.1. „Nezalomiteľné“ dátumy

Ďalším problémom, s ktorým sa pri dodržiavaní typografických pravidiel môžeme stretnúť. Je vkladanie nezalomiteľných medzier za číslice v dátumoch. Najjednoduchšie je tento problému vyriešiť tak, že automaticky vložíme nezalomiteľnú medzeru za každé číslo, ktoré končí znakom bodky:

```
sub Nezalomitelne_datumy

 dim Cislovky() as string
 dim i, kolko as long

 Cislovky()=array("1.","2.","3.","4.","5.","6.","7.","8.","9
.","0.")

 kolko=0
 for i=lbound(Cislovky()) to ubound (Cislovky())
  kolko=kolko+Vymen(Cislovky(i)+" ",Cislovky(i)
+NM,false,false,false)
 next i
 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné dátumy")
end sub
```

Pravdaže, toto riešenie nie je optimálne, pretože vloží nezalomiteľnú medzeru za úplne každé radové číslo bez ohľadu na to, kde sa vyskytuje. Pokiaľ ho teda budeme mať napr. na konci vety (napr. „ … a tento pretekár skončil 123.“, úplne zbytočne nám tu vloží nezalomiteľnú medzeru. Tento problém by sme mohli vyriešiť tak, že budeme postupne testovať všetky dvojice číslic „1. 1“, „1. 2“, … „0. 0“. napr. takto:

```
sub Nezalomitelne_datumy

 dim Cislovky() as string
 dim i, j, kolko as long

 Cislovky()=array("1","2","3","4","5","6","7","8","9","0")

 kolko=0
 for i=lbound(Cislovky()) to ubound (Cislovky())
  for j=lbound(Cislovky()) to ubound (Cislovky())
   kolko=kolko+Vymen(Cislovky(i)+".
```

Vlastná funkcia pre výmeny

```
"+Cislovky(j),Cislovky(i)
+"."+NM+Cislovky(j),false,false,false)
  next j
 next i
 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné dátumy")
end sub
```

Toto riešenie je však veľmi pomalé, pretože celý dokument prehľadáme presne sto krát, kým neotestujeme a neopravíme všetky kombinácie a preto je v praxi viac-menej nepoužiteľné. Pravdaže, pri vyhľadávaní by sme mohli použiť regulárne výrazy, napr. „[0-9]\. [0-9]“, pomocou ktorých dokážeme vyhľadať všetky potrebné kombinácie na jeden krok. Problém je však v tom, ako zabezpečiť výmenu medzery, ktorá sa nachádza niekde uprostred hľadaného výrazu, pričom ani nevieme, ktorú konkrétnu dvojicu číslic takto nájdeme.

Toto sa dá riešiť tak, že keď nájdeme niektorú dvojicu dátumových číslic, následne v nej pomocou funkcií pre prácu s reťazcami opravíme potrebnú medzeru a až potom takto pripraveným reťazcom nahradíme pôvodný. Na to nám však nebude stačiť štandardná metóda pre zámenu, ktorú sme doteraz používali:

```
Vymena=Dokument.createReplaceDescriptor()
```

ale najprv musíme vyhľadať potrebný reťazec metódou pre hľadanie:

```
Hladaj=Dokument.createSearchDescriptor()
```

Preto sa teraz musíme vrátiť k definícii premenných a procedúre „Init“, aby sme tam doplnili požadované údaje:

```
REM Definície spoločných premenných sú úplne pred
definíciou podprogramov
dim Dokument, Vymena, Hladaj as object
dim NM as string

sub Init

 Dokument=ThisComponent
 Vymena=Dokument.createReplaceDescriptor()
 Hladaj=Dokument.createSearchDescriptor()
 NM=chr$(&HA0) ' Nezalomiteľná medzera - hexadecimálny kód
A0

end sub
```

Po nájdení budeme vo vnútri dátumu vymieňať zalomiteľnú medzeru za nezalomiteľnú. Na takéto zámeny si zadefinujeme obecnú funkciu, pomocou ktorej dokážeme zameniť ľubovoľný vnútorný reťazec za iný, pretože v takomto tvare sa nám bude hodiť aj pre iné účely:

```
function Zamen_Vo_Vnutri (V_Com, Co, Za as string) as
string
REM Funkcia vyhľadá v reťazci V_Com reťazec Co a nahradí ho
reťazcom Za.
```

```
REM Ak tento reťazec nenájde, ponechá vo výsledku pôvodný
reťazec V_Com.

 dim pom as string
 dim i, j as long

 REM Pre prípad, že sa reťazec nenájde, necháme vo výsledku
pôvodný reťazec
 pom=V_Com

 REM pozícia hľadaného reťazca
 i=instr(V_Com,Co)

 REM veľkosť zvyšku pravej strany pôvodného reťazca, ktorá
je za hľadaným reťazcom
 j=len(V_Com)-i-len(Co)+1

 REM ak sa reťazec Co našiel, je jeho pozícia väčšia ako 0
 if i>0 then
  REM do výsledku priradíme ľavú časť pôvodného reťazca
plus nahrádzaný reťazec
  pom=left(V_Com,i-1)+Za

  REM ak je za hľadaným reťazcom ešte nejaký zvyšok
pôvodného reťazca
  if j>0 then
   REM do výsledku priradíme aj tento zvyšok
   pom=pom+right(V_Com,j)
  end if
 end if

  Zamen_Vo_Vnutri=pom
end function
```

Teraz už môžeme zadefinovať vlastnú funkciu na výmenu, kde najprv nájdeme hľadaný výraz, potom v ňom zameníme vnútorný reťazec za iný a nakoniec upraveným reťazcom nahradíme pôvodný. Pravdaže, pri hľadaní využijeme možnosť nastavenia hľadania pomocou regulárnych výrazov, celých slov a rozlišovania veľkosti písmen. Ako výsledok tejto funkcie bude počet prevedených zámen.

```
function Vymen_hladanim(V_Com, Co, Za as string, Regularne,
Cele_slova, Velke_pismena as Boolean) as long

REM procedúra vyhľadá reťazec V_Com, pričom až v skutočne
nájdenom reťazci zamení vnútorný reťazec Co za reťazec Za.

 dim nasiel as object
 dim kolko as long

 Hladaj.searchString=V_Com
```

Vlastná funkcia pre výmeny

```
 Hladaj.SearchRegularExpression=Regularne
 Hladaj.SearchWords=Cele_slova
 Hladaj.SearchCaseSensitive=Velke_pismena

 kolko=0

 REM objekt s nájdeným reťazcom priradíme do premennej
Nasiel
 nasiel = Dokument.findFirst(Hladaj)

 rem opakuj, pokiaľ tento objekt obsahuje nejaké údaje
 Do While NOT isNull(nasiel)

  rem ak sa našiel reťazec
  if len(nasiel.String)>0 then
   rem ak sa našiel reťazec, tak urobíme zámenu a výsledkom
nahradíme pôvodný reťazec
   nasiel.String=Zamen_Vo_Vnutri(nasiel.String,Co,Za)
   kolko=kolko+1
  end if
  rem vyhľadáme ďalší reťazec
  nasiel = Dokument.findNext( nasiel.End, Hladaj)
 Loop

 Vymen_hladanim=kolko

end function
```

Procedúra pre výmenu nezalomiteľných medzier vo vnútri dátumov potom môže vyzerať takto:

```
sub Nezalomitelne_datumy

 dim kolko as long

 kolko=Vymen_hladanim("[0-9]\. [0-9]","
",NM,true,false,false)
 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné dátumy")

end sub
```

V prípade, že máme v texte uvedený dátum s údajom dňa, mesiaca a roku nedokáže toto makro vložiť nezalomiteľnú medzeru medzi číselný údaj mesiaca a roku. Toto však môžeme riešiť opakovaným vyhľadávaním:

```
sub Nezalomitelne_datumy

 dim kolko as long

 kolko=Vymen_hladanim("[0-9]\. [0-9]","
```

```
",NM,true,false,false)
 kolko=kolko+Vymen_hladanim("[0-9]\. [0-9]","
",NM,true,false,false)
 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné dátumy")

end sub
```

15. Končíme s nezalomiteľnými medzerami

V dnešnom dieli dokončíme množinu funkcií pre vkladanie nezalomiteľných medzier, čím, pochopiteľne, ešte stále nevyčerpáme celú túto problematiku. Rôzne iné možnosti však necháme ako domácu úlohu.

15.1. „Nezalomiteľné“ čísla

Ako posledné funkcie, ktoré si v oblasti nezalomiteľných medzier naprogramujeme budú procedúry pre vkladanie týchto medzier za čísla. Pripomeňme si, že nezalomiteľné medzery by mali byť za jednociferným číslom bez bernej jednotky (napr. „9 ľudí“), medzi rádovými skupinami cifier vo vnútri čísla alebo pri telefónnych číslach (napr. „5 784 452“) a medzi číslom a značkou mernej jednotky (napr. „15 kg“). Úplne najjednoduchšie riešenie je také, že nezalomiteľnú medzeru vložíme za úplne každé číslo, napríklad takto:

```
sub Nezalomitelne_cisla

 dim kolko as long

 kolko=Vymen_hladanim("[0-9] "," ",NM,true,false,false)
 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné čísla")

end sub
```

Toto riešenie bude určite mnohým postačovať, no na druhej strane vkladá nezalomiteľné medzery za čísla aj vtedy, keď to nie je potrebné, napr. v texte „12 stromov“. Preto musíme jednotlivé možnosti naprogramovať osobitne.

15.2. „Nezalomiteľné“ jednociferné čísla

Riešenie problematiky jednociferných čísiel je veľmi jednoduché, pretože za každým jednociferným číslom by mala nasledovať nezalomiteľná medzera. Jednociferné číslo zistíme podobne, ako sme zisťovali predložky – t.j. číslo za ktorým je medzera sa musí nachádzať na začiatku slova:

```
sub Nezalomitelne_jednociferne_cisla

 dim kolko as long

 kolko=Vymen_hladanim("\<[0-9] "," ",NM,true,false,false)
 msgbox("Nahradených "+kolko+" zalomiteľných
```

```
medzier.",0,"Nezalomiteľné jednociferné čísla")

end sub
```

15.3. „Nezalomiteľné“ telefónne čísla

V prípade rádových skupín cifier a telefónnych čísiel musíme zameniť medzeru, ktorá sa nachádza medzi dvomi číslami:

```
sub Nezalomitelne_telefonne_cisla

 dim kolko as long

 kolko=Vymen_hladanim("[0-9] [0-9]","
",NM,true,false,false)
 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné telefónne čísla")

end sub
```

15.4. „Nezalomiteľné“ merné jednotky

Na koniec sme si nechali problematiku nezalomiteľných medzier, ktoré sa nachádzajú medzi číslom a značkou mernej jednotky. Aby sme mohli vyriešiť túto problematiku, musíme, podobne ako pri akademických tituloch vymenovať všetky merné jednotky, ktoré sa za číslom môžu vyskytovať. Už v tomto však objavíme problém, pretože celkový počet rôznych merných jednotiek (vrátane ich kombinácií s násobkami „kilo“, „mega“, „piko“ ...) dosahuje rádovo stovky možností. Pravdaže, málokto v praxi používa skutočne všetky a preto, ak ich počet bude ešte únosný, môže použiť nasledovné makro, kde sme ako príklad použili jednotky dĺžky:

```
sub Nezalomitelne_merne_jednotky

 dim Merne_jednotky() as string
 dim i, kolko as long

 Merne_jednotky()=array("m", "km", "mm", "cm")

 for i=lbound(Merne_jednotky()) to ubound(Merne_jednotky())
  kolko=kolko+Vymen_hladanim("[0-9] "+Merne_jednotky(i),"
",NM,true,false,false)
 next i

 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné merné jednotky")

end sub
```

Počet merných jednotiek,pri ktorých bude uverejnené makro ešte dostatočne rýchle závisí najmä od veľkosti spracovávaných dokumentov a preto sa nedá jednoznačne ur-

čiť ich počet. Vzniká otázka, či by sa to predsa len nedalo vyriešiť inak, pretože testovanie rôznych možností jednotiek v rámci jedného cyklu je určite rýchlejšie, ako opakované prehľadávanie celého dokumentu. Našťastie, aj v tomto smere nám riešenie poskytujú priamo regulárne výrazy, kde máme možnosť zadať logickú podmienku „alebo“:

```
sub Nezalomitelne_merne_jednotky

 dim kolko as long

 kolko=kolko+Vymen_hladanim("[0-9] (m|km|mm|cm)","
",NM,true,false,false)

 msgbox("Nahradených "+kolko+" zalomiteľných
medzier.",0,"Nezalomiteľné merné jednotky")

end sub
```

Toto riešenie môžeme použiť aj na iné funkcie v oblasti formátovania nezalomiteľných medzier. O tom si však budeme hovoriť až niekedy inokedy.

16. Nadbytočné medzery

Pokiaľ si pamätáte, v tretej časti sme uviedli zatiaľ poslednú verziu makra pre odstraňovanie viacnásobných medzier, v ktorej sme počítali aj s prázdnymi medzerami na začiatkoch a koncoch odsekov. Týmto sme, pochopiteľne, neopravili všetky možné výskyty nadbytočných medzier, pretože sa s nimi môžeme stretnúť aj v iných častiach dokumentu, napr. v textoch v zátvorkách, medzi úvodzovkami a pod.

16.1. Čo nám hovoria normy

Aby sme vedeli, ako správne písať medzery, pozrime sa na to, čo nám v tomto smere hovorí norma pre úpravu písomností (STN 01 6910, ktorá je prakticky zhodná s normou ČSN 01 6910). Tam sa o. i. dozvieme, že interpunkčné znamienka bodka, čiarka, dvojbodka, bodkočiarka, výkričník a otáznik sa pripájajú k textu bez medzery (napr. „A preto, hoci nerád, to urobil.“). Za tieto znaky patrí medzera okrem tých prípadov, keď ich nasleduje viac za sebou (napr. „bolo to vo firme FIRMA s r. o., takže ...“). Medzera sa nedáva ani za bodku v peňažných číslach (napr. „12.000 Sk“), za bodku pri číselnom označovaní časti textu (napr. „kapitola 2.7“), za čiarku pri oddeľovaní desatinných miest (napr. „3,14“) a za dvojbodku pri písaní časových údajov (napr. „8:48 h“). Zátvorky a úvodzovky sa na začiatku a konci textu, ktorý dávame do zátvoriek alebo do úvodzoviek, neoddeľujú od tohto textu medzerami (ako môžeme vidieť vo všetkých príkladoch, ktoré sme v tomto odseku pred chvíľou uviedli). Medzery sa nepoužívajú ani pri spojovníku (napr. „slovensko-nemecký slovník“).

Ako vidíme, aj táto problematika – podobne ako problematika nezalomiteľných medzier – je pomerne široká. V tejto súvislosti sa ešte musíme vrátiť k piatej časti tohto seriálu, kde som spomínal dva možné tvary akademického titulu čestného doktorátu „honoris causa“ – „h.c.“ (bez medzery) alebo „h. c.“ (s nezalomiteľnou medzerou). Tieto dva tvary sa uvádzali v materiáloch, z ktorých som čerpal podklady pre zoznam aka-

demických titulov. Tvar bez medzery je však v rozpore so spomínanými normami pre úpravu písomnosti, čo som si vtedy neuvedomil. Týmto sa všetkým za uvedený omyl ospravedlňujem. Predmetný článok však už nebudem opravovať, pretože všetky uvedené makrá sú zároveň príkladmi riešenia určitého typu problematiky – v tomto prípade viacerých výnimiek z nejakého pravidla.

16.2. Nadbytočné medzery pri interpunkčných znamienkach

Ako prvé, čo si z uvedenej problematiky budeme programovať bude odstraňovanie nadbytočných medzier pred a za interpunkčnými znamienkami. Nebudeme sa už venovať pomalým náhradám cez cykly, ale využijeme rýchlejšie riešenie pomocou regulárnych výrazov.

```
sub Zmaz_interpunkcia_medzera

 dim kolko as long

 REM Vymazanie medzier za ľavými znakmi – úvodzovkami a
zátvorkami
 REM Tu patrí týchto šesť znakov: „ « " ( [ {
 REM Úvodzovky " musíme zadať pomocou ich kódu, zátvorky
( a [ musíme zadať pomocou uvádzajúceho znaku \

 kolko=Vymen_hladanim("(„|«|"+chr$(&H22)+"|\(|\[|{) "," 
","",true,false,false)

 REM Vymazanie medzier pred pravými zátvorkami a
úvodzovkami a pred interpunkciou
 REM Tu patrí týchto šesť znakov: “ » " ) ] }
 REM a týchto šesť interpunkčných znamienok: , . ! ? ; :
 REM Úvodzovky " musíme zadať pomocou ich kódu, znaky , . ?
) ] musíme zadať pomocou uvádzajúceho znaku \

 kolko=Vymen_hladanim(" (“|»|"+chr$
(&H22)+"|\)|\]|}|\,|\.|!|\?|;|:)"," ","",true,false,false)

 msgbox("Vymazaných "+kolko+" nadbytočných
medzier.",0,"Nadbytočné medzery")

end sub
```

16.3. Chýbajúce medzery pri interpunkčných znamienkach

V minulom dieli sme si spomínali, že v normách pre úpravu písomností sa spomína, že okrem niekoľkých výnimiek sa za interpunkčné znamienka vkladá medzera. A práve na tento problém sa teraz zameriame. Aby sme ho mohli vyriešiť, znovu si pripomeňme, že za interpunkčné znamienka sa vkladá medzera okrem tých prípadov, keď ich nasleduje viac za sebou, medzera sa nevkladá ani za bodku oddeľujúcu rády v číslach, desatinnú čiarku a za dvojbodku v časových údajoch. Z hľadiska programovania to

znamená, že medzeru budeme vkladať za diakritické znamienka vtedy, ak za nimi bezprostredne nasleduje písmeno a v prípade diakritických znamienok otáznika, výkričníka a bodkočiarky aj vtedy, ak za nimi bezprostredne nasleduje aj číslo.

Na rozdiel od predchádzajúcich makier nebudeme teraz nič mazať, ale iba vkladať. Preto si najprv musíme naprogramovať funkciu pre vloženie reťazca do nájdeného textu:

```
function Vloz_hladanim(V_Com, Co as string, Kde as long,
Regularne, Cele_slova, Velke_pismena as Boolean)
REM funkcia vyhľadá reťazec V_Com a na pozíciu Kde do neho
vloží reťazec Co

 dim nasiel as object
 dim kolko, i, j as long
 dim pom as string

 Hladaj.searchString=V_Com
 Hladaj.SearchRegularExpression=Regularne
 Hladaj.SearchWords=Cele_slova
 Hladaj.SearchCaseSensitive=Velke_pismena

 kolko=0

 REM úprava parametra Kde (pozícia 0 - vloženie pred
reťazec sa musí upraviť na 1)
 i=kde+1

 REM objekt s nájdeným reťazcom priradíme do premennej
Nasiel
 nasiel = Dokument.findFirst(Hladaj)

 rem opakuj, pokiaľ tento objekt obsahuje nejaké údaje
 Do While NOT isNull(nasiel)

  rem ak sa našiel reťazec
  if len(nasiel.String)>0 then
   REM veľkosť zvyšku pravej strany pôvodného reťazca
   j=len(nasiel.String)-i+1
   REM do výsledku priradíme ľavú časť pôvodného reťazca
plus vkladaný reťazec
   pom=left(nasiel.String,i-1)+Co
   REM ak je za hľadaným reťazcom ešte nejaký zvyšok
pôvodného reťazca
   if j>0 then
    REM do výsledku priradíme aj tento zvyšok
    pom=pom+right(nasiel.String,j)
   end if
   nasiel.String=pom
   kolko=kolko+1
```

Nadbytočné medzery

```
  end if

  rem vyhľadáme ďalší reťazec
  nasiel = Dokument.findNext( nasiel.End, Hladaj)
 Loop

 Vloz_hladanim=kolko

end function
```

S využitím tejto funkcie môžeme potom naprogramovať makro pre vkladanie medzier napríklad takto:

```
sub Vloz_interpunkcia_medzera

 dim kolko as long

 REM a týchto šesť interpunkčných znamienok: , . ! ? ; :
 REM Znaky , . ? musíme zadať pomocou uvádzajúceho znaku \

 REM Za znakmi ? ! ; hľadáme ľubovoľný alfanumerický znak
 kolko=Vloz_hladanim("(\?|!|;)[:alnum:]{1}","
",1,true,false,false)

 REM Za znakmi , . : hľadáme ľubovoľné písmeno
 kolko=kolko+Vloz_hladanim("(\,|\.|:)[:alpha:]{1}","
",1,true,false,false)

 msgbox("Vložených "+kolko+" medzier.",0,"Vložené medzery")

end sub
```

Toto makro však nevkladá medzeru medzi bodku (čiarku, dvojbodku) a číslo v tom prípade, ak sa pred ním nachádza iný znak ako číslo (napr. „... sa zúčastnili všetci.12 ľudí potom ...“). Komu to nevyhovuje, musí si makro príslušne upraviť:

```
sub Vloz_interpunkcia_medzera

 dim kolko as long

 REM a týchto šesť interpunkčných znamienok: , . ! ? ; :
 REM Znaky , . ? musíme zadať pomocou uvádzajúceho znaku \

 REM Za znakmi ? ! ; hľadáme ľubovoľný alfanumerický znak
 kolko=Vloz_hladanim("(\?|!|;)[:alnum:]{1}","
",1,true,false,false)

 REM Za znakmi , . : hľadáme ľubovoľné písmeno
 kolko=kolko+Vloz_hladanim("(\,|\.|:)[:alpha:]{1}","
",1,true,false,false)
```

```
 REM Ak pred znakmi , . : nie je číslica a za nimi je
číslica, vložíme medzeru
 kolko=kolko+Vloz_hladanim("[^0-9](\,|\.|:)[:digit:]{1}","
",2,true,false,false)

 msgbox("Vložených "+kolko+" medzier.",0,"Vložené medzery")

end sub
```

Teraz však prehľadávame dokument v prípade znakov bodky, čiarky a dvojbodky dvakrát. Toto sa dá vyriešiť napríklad takto:

```
sub Vloz_interpunkcia_medzera

 dim kolko as long

 REM a týchto šesť interpunkčných znamienok: , . ! ? ; :
 REM Znaky , . ? musíme zadať pomocou uvádzajúceho znaku \

 REM Za znakmi ? ! ; hľadáme ľubovoľný alfanumerický znak
 kolko=Vloz_hladanim("(\?|!|;)[:alnum:]{1}","
",1,true,false,false)

 REM Ak pred znakmi , . : nie je číslica a za nimi je
ľubovoľný alfanumerický znak
 kolko=kolko+Vloz_hladanim("[^0-9](\,|\.|:)[:alnum:]{1}","
",2,true,false,false)

 msgbox("Vložených "+kolko+" medzier.",0,"Vložené medzery")

end sub
```

17. Malá rekapitulácia

V dnešnom dieli seriálu o programovaní makier v OpenOffice.org si zrekapitulujeme všetky makrá pre formátovanie dokumentov, ktoré sme doteraz vytvorili a trochu (teda dosť podstatne) ich upravíme.

Na úvod si urobme krátky prehľad doteraz naprogramovaných makier pre formátovanie dokumentov:

– makro pre odstraňovanie viacnásobných medzier „Viacnasobna_Medzera“

– makro pre vkladanie nezalomiteľných medzier za predložky a spojky „ Nezalomitelne_Spojky“

– makro pre vkladanie nezalomiteľných medzier pred a za akademické tituly „ Nezalomitelne_tituly“

– makro pre vkladanie nezalomiteľných medzier do dátumov „ Nezalomitelne_datumy“

– makro pre vkladanie nezalomiteľných medzier medzi jednociferné čísla a text „Nezalomitelne_Jednociferne_Cisla“

– makro pre vkladanie nezalomiteľných medzier v telefónnych číslach „Nezalomitelne_Telefonne_Cisla“

– makro pre vkladanie nezalomiteľných medzier medzi čísla a merné jednotky „Nezalomitelne_Merne_Jednotky“

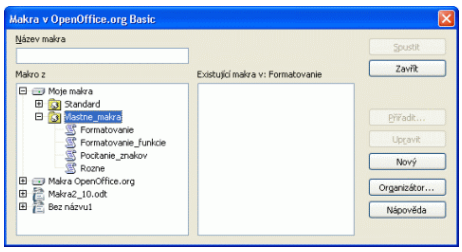
– makro pre mazanie nadbytočných medzier pri interpunkčných znamienkach „Zmaz_Interpunkcia_Medzera“

– makro pre vkladanie chýbajúcich medzier pri interpunkčných znamienkach „ Vloz_Interpunkcia_Medzera“

K tomu ešte musíme pridať makro pre zámenu troch bodiek za „trojbodku“, ktoré sme naznačili ešte v štvrtom pokračovaní prvej časti tohto seriálu a ktoré môžeme nazvať „Rozne_Zameny“.

Pri praktickej úprave dokumentov je veľmi nepohodlné, aby sme každé makro spúšťali osobitne a tak isto nechceme, aby sme museli pri ich spoločnom spustení osobitne potvrdzovať zistené počty jednotlivých prevedených zámen. Na druhej strane však niekedy potrebujeme práve osobitné, resp. selektívne spúšťanie. Aby sme toto všetko mohli vyriešiť, musíme si uvedené makrá prispôsobiť.

Každé makro rozdelíme na dve časti. Prvá časť bude funkcia, ktorá prevedie príslušné formátovanie a ako výsledok nám vráti počet výmien (opráv) bez toho, aby ho vypisovala na obrazovku. Druhá časť bude malý podprogram, ktorý bude predmetnú funkciu volať a následne aj vypisovať počet prevedených opráv.



Obrázok 17.1: Pre makrá vytvoríme osobitné moduly

Aby sme si udržali potrebný prehľad, rozdelíme si tieto dve časti do osobitných modulov v rámci skupín makier. Cez menu „Nástroje – Makrá – Správca makier – OpenOffice.org Basic...“ si otvoríme dialógové okno „Makrá v OpenOffice.org Basic“. V skupine makier „Vlastne_makra“ si vytvoríme dva moduly, ktoré nazveme „Formatovanie“ a „Formatovanie_funkcie“. V module „Formatovanie“ budeme mať malé podprogramy, ktoré budú volať vlastné formátovacie funkcie, ktoré uložíme do modulu „Formatovanie_funkcie“.

Aby sme rozlíšili funkciu od podprogramu, vložíme na jej začiatok písmeno „f“. Celý modul „Formatovanie_funkcie“ bude vyzerať nasledovne:

```
REM Definície spoločných premenných sú úplne pred
definíciou podprogramov

dim Dokument, Vymena, Hladaj as object
dim NM as string

REM Inicializácia spoločných premenných
sub Init
 Dokument=ThisComponent
 Vymena=Dokument.createReplaceDescriptor()
 Hladaj=Dokument.createSearchDescriptor()
 NM=chr$(&HA0) ' Nezalomiteľná medzera – hexadecimálny kód
```

```
A0
end sub

REM Výmena reťazca Co za reťazec Za
function Vymen(Co, Za as string, Regularne, Cele_slova,
Velke_pismena as Boolean) as long
 Vymena.SearchString=Co
 Vymena.ReplaceString=Za
 Vymena.SearchRegularExpression=Regularne
 Vymena.SearchWords=Cele_slova
 Vymena.SearchCaseSensitive=Velke_pismena
 Vymen=Dokument.replaceAll(Vymena)
end function

REM Vlastná výmena reťazca Co za reťazec Za vo vnútri
reťazca V_Com
function Zamen_Vo_Vnutri (V_Com, Co, Za as string) as
string
 dim pom as string
 dim i, j as long
 pom=V_Com
 i=instr(V_Com,Co)
 j=len(V_Com)-i-len(Co)+1
 if i>0 then
  pom=left(V_Com,i-1)+Za
  if j>0 then
   pom=pom+right(V_Com,j)
  end if
 end if
 Zamen_Vo_Vnutri=pom
end function

REM Výmena vnútorného raťazca Co za reťazec Za hľadaním
reťazca V_Com
function Vymen_hladanim(V_Com, Co, Za as string, Regularne,
Cele_slova, Velke_pismena as Boolean) as long
 dim nasiel as object
 dim kolko as long
 Hladaj.searchString=V_Com
 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
   nasiel.String=Zamen_Vo_Vnutri(nasiel.String,Co,Za)
   kolko=kolko+1
  end if
```

```
  nasiel = Dokument.findNext( nasiel.End, Hladaj)
 Loop
 Vymen_hladanim=kolko
end function
REM Vyhľadanie reťazca V_Com a na pozíciu Kde do neho vloží
reťazec Co
function Vloz_hladanim(V_Com, Co as string, Kde as long,
Regularne, Cele_slova, Velke_pismena as Boolean) as long
 dim nasiel as object
 dim kolko, i, j as long
 dim pom as string
 Hladaj.searchString=V_Com
 Hladaj.SearchRegularExpression=Regularne
 Hladaj.SearchWords=Cele_slova
 Hladaj.SearchCaseSensitive=Velke_pismena
 kolko=0
 i=kde+1
 nasiel = Dokument.findFirst(Hladaj)
 Do While NOT isNull(nasiel)
  if len(nasiel.String)>0 then
   j=len(nasiel.String)-i+1
   pom=left(nasiel.String,i-1)+Co
   if j>0 then
    pom=pom+right(nasiel.String,j)
   end if
   nasiel.String=pom
   kolko=kolko+1
  end if
  nasiel = Dokument.findNext( nasiel.End, Hladaj)
 Loop
 Vloz_hladanim=kolko
end function

REM Makro pre odstraňovanie viacnásobných medzier
function f_Viacnasobna_Medzera as long
 dim kolko AS Long
 kolko=Vymen(" +"," ",TRUE,false,false)
 kolko=kolko+Vymen(" "+NM+" |"+NM+" |
"+NM,NM,TRUE,false,false)
 kolko=kolko+Vymen(" \t |\t | \t","\t",TRUE,false,false)
 kolko=kolko+Vymen("^[:space:]*|
[:space:]*$","",TRUE,false,false)
 kolko=kolko+Vymen("^\t*|\t*$","",TRUE,false,false)
 f_Viacnasobna_Medzera=kolko
end function

REM Makro pre vkladanie nezalomiteľných medzier za
predložky a spojky
function f_Nezalomitelne_Spojky as long
```

```
 f_Nezalomitelne_Spojky=Vymen_hladanim("\<(a|i|k|o|s|u|v|z)
"," ",NM,true,false,false)
end function

REM Makro pre vkladanie nezalomiteľných medzier pred a za
akademické tituly
function f_Nezalomitelne_Tituly as long
 dim Tituly_pred(), Tituly_pred_spec(), Tituly_za() as
string
 dim i, kolko as long
 Tituly_pred()=array("Bc", "Dr", "Ing", "JUDr", "Mgr",
"MUDr", "MVDr", "PaedDr", "PharmDr", "PhDr", "PhMr",
"RNDr", "ThDr")
 Tituly_pred_spec()=array("akad", "Akad", "arch", "Arch",
"doc", "Doc", "prof", "Prof")
 Tituly_za()=array("CSc","DrSc")
 kolko=Vymen("h. c.","h."+NM+"c.",false,false,false)
 kolko=kolko+Vymen(" h.c.",NM+"h.c.",false,false,false)
 kolko=kolko+Vymen("
h."+NM+"c.",NM+"h."+NM+"c.",false,false,false)
 for i=lbound(Tituly_pred()) to ubound(Tituly_pred())
  kolko=kolko+Vymen("\<"+Tituly_pred(i)+"\. ",Tituly_pred(i)
+"."+NM,true,false,false)
 next i
 for i=lbound(Tituly_pred_spec()) to
ubound(Tituly_pred_spec())
  kolko=kolko+Vymen("\<"+Tituly_pred_spec(i)+"\.
",Tituly_pred_spec(i)+"."+NM,true,false,true)
 next i
 for i=lbound(Tituly_za()) to ubound(Tituly_za())
  kolko=kolko+Vymen(" "+Tituly_za(i)
+".",NM+Tituly_za(i),false,false,false)
 next i
 f_Nezalomitelne_Tituly=kolko
end function

REM Makro pre vkladanie nezalomiteľných medzier do dátumov
function f_Nezalomitelne_Datumy as long
 dim kolko as long
 kolko=Vymen_hladanim("[0-9]\. [0-9]","
",NM,true,false,false)
 kolko=kolko+Vymen_hladanim("[0-9]\. [0-9]","
",NM,true,false,false)
 f_Nezalomitelne_Datumy =kolko
end function

REM Makro pre vkladanie nezalomiteľných medzier medzi
jednociferné čísla a text
function f_Nezalomitelne_Jednociferne_Cisla as long
```

```
 f_Nezalomitelne_Jednociferne_Cisla =Vymen_hladanim("\<[0-
9] "," ",NM,true,false,false)
end function

REM Makro pre vkladanie nezalomiteľných medzier v
telefónnych číslach
function f_Nezalomitelne_Telefonne_Cisla as long
 f_Nezalomitelne_Telefonne_Cisla =Vymen_hladanim("[0-9] [0-
9]"," ",NM,true,false,false)
end function

REM Makro pre vkladanie nezalomiteľných medzier medzi čísla
a merné jednotky
function f_Nezalomitelne_Merne_Jednotky as long
 f_Nezalomitelne_Merne_Jednotky=Vymen_hladanim("[0-9] (m|
km|mm|cm)"," ",NM,true,false,false)
end function

REM Makro pre mazanie nadbytočných medzier pri
interpunkčných znamienkach
function f_Zmaz_Interpunkcia_Medzera as long
 dim kolko as long
 kolko=Vymen_hladanim("(„|«|"+chr$(&H22)+"|\(|\[|{) ","
","",true,false,false)
 kolko=kolko+Vymen_hladanim(" (“|»|"+chr$
(&H22)+"|\)|\]|}|\,|\.|!|\?|;|:)"," ","",true,false,false)
 f_Zmaz_Interpunkcia_Medzera=kolko
end function

REM Makro pre vkladanie chýbajúcich medzier pri
interpunkčných znamienkach
function f_Vloz_Interpunkcia_Medzera as long
 dim kolko as long
 kolko=Vloz_hladanim("(\?|!|;)[:alnum:]{1}","
",1,true,false,false)
 kolko=kolko+Vloz_hladanim("[^0-9](\,|\.|:)[:alnum:]{1}","
",2,true,false,false)
 f_Vloz_Interpunkcia_Medzera=kolko
end function

REM Makro pre rôzne zámeny (napr. troch bodiek za
„trojbodku“)
function f_Rozne_zameny as long
 dim Co(), Za() as string
 dim n as long
 dim kolko as long
 Co()=array("...")
 Za()=array("…")
 kolko=0
```

```
 for n=lbound(Co()) to ubound(Za())
  kolko=kolko+Vymen(Co(n),Za(n),false,false,false)
 next n
 f_Rozne_zameny=kolko
end function
```

Všimnite si zaujímavú zmenu makra pre vkladanie nezalomiteľných medzier za jednoznakové predložky a spojky, pretože sme aj pri ňom využili našu vlastnú funkciu pre výmenu.

18. Záverečné úpravy

V poslednom dieli druhej časti seriálu o programovaní makier v OpenOffice. org dokončíme prvú (ale nie poslednú) komplexnú verziu makier pre formátovanie dokumentov.

Naposledy sme si povedali, že makrá pre formátovanie dokumentov rozdelíme do dvoch modulov, pričom sme si uviedli zdrojové texty modulu „Formatovanie_funkcie“. Dnes si ukážeme, ako budeme tieto funkcie volať v procedúrach, ktoré uložíme do modulu „Formatovanie“. Prístup k podprogramom, ktoré sa nachádzajú v iných moduloch nám umožňuje zápis, ktorý je podobný zápisu pri volaní metód v rámci objektových premenných:

```
meno_modulu.meno_podprogramu
```

Ako sme si spomínali minule, chceme zároveň naprogramovať procedúru, ktorá bude volať všetky formátovacie funkcie. Pri ich volaní však nemôžeme použiť hocijaké poradie, pretože potom by sme nedosiahli požadovaný výsledok. Ako prvé musia byť prevedené rôzne zámeny, pretože napr. pred „trojbodkou“ nechávame medzeru, ale pred bodkou nie. Následne je potrebné vymazať viacnásobné medzery, lebo potom by v prípade ich výskytu napr. za spojkou zostala kombinácia nezalomiteľná + zalomiteľná medzera. Ešte pred opravou zalomiteľných medzier za spojkami a predložkami sa najprv musí vložiť medzera za interpunkčné znamienka, lebo potom by mohla zostať zalomiteľná medzera za spojkou, ak predtým pred ňou chýbala. Takisto pre opravou predložiek a spojok musíme zmazať nadbytočné medzery pri interpunkčných znamienkach, pretože v prípade, že napr. na konci zátvorky je iba spojka, zbytočne sa najprv zamení medzera za nezalomiteľnú, aby sa za chvíľu úplne vymazala. Pretože sme vo funkcii pre mazanie nadbytočných medzier pri pri interpunkčných znamienkach neuvažovali s možnosťou nezalomiteľnej medzery a tabelátora, musíme si najprv v module „Formatovanie_funkcie“ opraviť túto funkciu:

```
function f_Zmaz_Interpunkcia_Medzera as long
 dim TAB as string
 dim kolko as long
 TAB=chr$(&H09) ' Tabelátor
 kolko=Vymen_hladanim("(„|«|"+chr$(&H22)+"|\(|\[|{) ","
","", true, false, false)
 kolko=kolko+Vymen_hladanim("(„|«|"+chr$(&H22)+"|\(|\[|
{)"+NM, NM,"", true, false, false)
 kolko=kolko+Vymen_hladanim("(„|«|"+chr$(&H22)+"|\(|\[|
{)"+TAB, TAB,"", true, false,false)
```

```
 kolko=kolko+Vymen_hladanim(" (“|»|"+chr$
(&H22)+"|\)|\]|}|\,|\.|!|\?|;|:)","","", true,false,false)
 kolko=kolko+Vymen_hladanim(NM+"(“|»|"+chr$
(&H22)+"|\)|\]|}|\,|\.|!|\?|;|:)", NM,"", true, false,
false)
 kolko=kolko+Vymen_hladanim(TAB+"(“|»|"+chr$
(&H22)+"|\)|\]|}|\,|\.|!|\?|;|:)", TAB,"", true, false,
false)
 f_Zmaz_Interpunkcia_Medzera=kolko
end function
```

V prípade, že budete potrebovať znak tabelátora aj v iných funkciách, je vhodné zaradiť jeho definíciu medzi spoločné premenné a následne jeho inicializáciu zaradiť do podprogramu Init. To však ponecháme prípadným záujemcom ako domácu úlohu. Modul „Formatovanie“, kde sú definované podprogramy pre formátovanie dokumentu môže pri dodržaní vyššie uvedených podmienok vyzerať napr. takto:

```
REM Makro pre odstraňovanie viacnásobných medzier
sub Viacnasobna_Medzera
 Formatovanie_funkcie.Init
 msgbox("Vymazanie"+Formatovanie_funkcie.f_Viacnasobna_Medz
era()+"viacnásobných medzier.", 0,"Viacnásobné medzery")
end sub

REM Makro pre vkladanie nezalomiteľných medzier za
predložky a spojky
sub Nezalomitelne_Spojky
 Formatovanie_funkcie.Init
 msgbox("Vložených"+Formatovanie_funkcie.f_Nezalomitelne_Sp
ojky()+"nezalomiteľných medzier.", 0,"Nezalomiteľné spojky
a predložky")
end sub

REM Makro pre vkladanie nezalomiteľných medzier pred a za
akademické tituly
sub Nezalomitelne_Tituly
 Formatovanie_funkcie.Init
 msgbox("Vložených"+Formatovanie_funkcie.f_Nezalomitelne_Ti
tuly()+"nezalomiteľných medzier.", 0,"Nezalomiteľné tituly")
end sub

REM Makro pre vkladanie nezalomiteľných medzier do dátumov
sub Nezalomitelne_Datumy
 Formatovanie_funkcie.Init
 msgbox("Vložených"+Formatovanie_funkcie.f_Nezalomitelne_Da
tumy()+"nezalomiteľných medzier.", 0,"Nezalomiteľné
dátumy")
end sub

REM Makro pre vkladanie nezalomiteľných medzier medzi
```

```
jednociferné čísla a text
sub Nezalomitelne_Jednociferne_Cisla
 Formatovanie_funkcie.Init
 msgbox("Vložených"+Formatovanie_funkcie.f_Nezalomitelne_Je
dnociferne_Cisla()+"nezalomiteľných medzier.",
0,"Nezalomiteľné jednociferné čísla")
end sub

REM Makro pre vkladanie nezalomiteľných medzier v
telefónnych číslach
sub Nezalomitelne_Telefonne_Cisla
 Formatovanie_funkcie.Init
 msgbox("Vložených"+Formatovanie_funkcie.f_Nezalomitelne_Te
lefonne_Cisla()+"nezalomiteľných medzier.",
0,"Nezalomiteľné telefónne čísla")
end sub

REM Makro pre vkladanie nezalomiteľných medzier medzi čísla
a merné jednotky
sub Nezalomitelne_Merne_Jednotky
 Formatovanie_funkcie.Init
 msgbox("Vložených"+Formatovanie_funkcie.f_Nezalomitelne_Me
rne_Jednotky()+"nezalomiteľných medzier.", 0,"Nezalomiteľné
merné jednotky")
end sub

REM Makro pre vloženie nezalomiteľných medzier za čísla ako
také
sub Nezalomitelne_Cisla
 dim kolko as long
 Formatovanie_funkcie.Init
 kolko=Formatovanie_funkcie.f_Nezalomitelne_Datumy()
 kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Jednocifer
ne_Cisla()
 kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Telefonne_
Cisla()
 kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Merne_Jed
notky()
 msgbox("Vložených"+kolko+"nezalomiteľných medzier.",
0,"Nezalomiteľné čísla")
end sub

REM Makro pre mazanie nadbytočných medzier pri
interpunkčných znamienkach
sub Zmaz_Interpunkcia_Medzera
 Formatovanie_funkcie.Init
 msgbox("Vymazaných"+Formatovanie_funkcie.f_Zmaz_Interpunkc
ia_Medzera()+"nadbytočných medzier.", 0,"Nadbytočné
interpunkčné medzery")
```

```
end sub

REM Makro pre vkladanie chýbajúcich medzier pri
interpunkčných znamienkach
sub Vloz_Interpunkcia_Medzera
 Formatovanie_funkcie.Init
 msgbox("Vložených"+Formatovanie_funkcie.f_Vloz_Interpunkci
a_Medzera()+"chýbajúcich medzier.", 0,"Chýbajúce
interpunkčné medzery")
end sub

REM Makro pre opravu (vloženie/vymazanie) medzier pri
interpunkčných znamienkach
sub Oprav_Interpunkcia_Medzera
 Formatovanie_funkcie.Init
 msgbox("Opravených"+Formatovanie_funkcie.f_Zmaz_Interpunkc
ia_Medzera()
+Formatovanie_funkcie.f_Vloz_Interpunkcia_Medzera()
+"medzier.", 0,"Nesprávne interpunkčné medzery")
end sub

REM Makro pre rôzne zámeny (napr.troch bodiek za
„trojbodku“)
sub Rozne_zameny
 Formatovanie_funkcie.Init
 msgbox("Vykonaných"+Formatovanie_funkcie.f_Rozne_zameny()
+"výmien.", 0,"Rôzne zámeny")
end sub

REM Makro pre celkové formátovanie dokumentu
sub Formatuj_Dokument
 dim kolko as long
 Formatovanie_funkcie.Init
 ' Prvé musia byť rôzne zámeny
 kolko=Formatovanie_funkcie.f_Rozne_Zameny()
 ' Potom odstránime viacnásobné medzery
 kolko=kolko+Formatovanie_funkcie.f_Viacnasobna_Medzera()
 ' Opravíme medzery pri interpunkčných znamienkach
 kolko=kolko+Formatovanie_funkcie.f_Zmaz_Interpunkcia_Medze
ra()+Formatovanie_funkcie.f_Vloz_Interpunkcia_Medzera()
 ' Vložíme nezalomiteľné medzery k číslam
 kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Datumy()
+Formatovanie_funkcie.f_Nezalomitelne_Jednociferne_Cisla()
+Formatovanie_funkcie.f_Nezalomitelne_Telefonne_Cisla()
+Formatovanie_funkcie.f_Nezalomitelne_Merne_Jednotky()
 ' Opravíme nezalomiteľné medzery pri akademických tituloch
 kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Tituly()
 ' A nakoniec opravíme nezalomiteľné medzery pri spojkách a
predložkách
```

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

Skončili sme druhú veľkú časť seriálu o programovaní makier v OpenOffice. org. Ako ste si mohli všimnúť aj v dnešnom dieli, uvedené funkcie určite neposkytujú všetky možnosti, ktoré pri práci potrebujeme. Na základe doterajších príkladov si však už určite dokážete naprogramovať aj prípadné úpravy, pomocou ktorých vhodne upravíte alebo rozšírite ich možnosti. V našom seriáli budeme teraz pokračovať treťou veľkou časťou, v ktorej si naznačíme možnosti programovania makier pre OpenOffice.org v oblasti vstupov a výstupov, t.j. budeme programovať rôzne vstupné formuláre (dialógové okná) a pod. Pri tomto programovaní sa stále budeme zaoberať funkciami pre formátovanie dokumentov, takže tieto znovu prekonajú niekoľko zmien.

Interaktívne vstupy

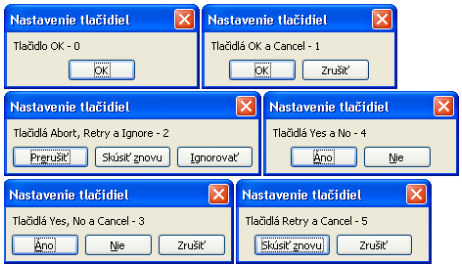
Začíname tretiu veľkú časť programovania makier v OpenOffice.org, ktorú zameriame na interaktívne vstupy, t. j. na možnosť riadenia a zadávania rôznych parametrov a informácií priamo v makrách.

Doteraz sme písali makrá, ktoré bežali priamo bez akéhokoľvek interaktívneho zásahu používateľa. Ale napr. už pri nezalomiteľných 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.

19. 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 zobraziť a tretí je názov tohto okna. Teraz sa pristavíme práve pri druhom parametri – 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.



Obrázok 19.1: Možnosti tlačidiel vo funkcii "Msgbox"

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“ („Prerušiť“, „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):

Obrázok 19.2: Štandardné tlačidlo

0 – štandardnú hodnotu definuje prvé tlačidlo

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:

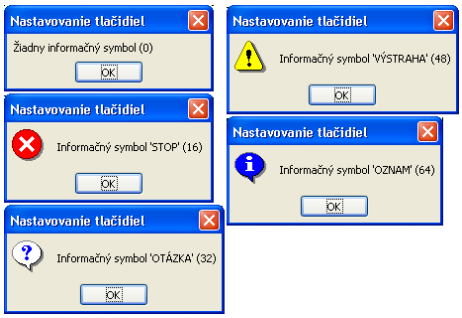
0 – nezobrazí sa žiadny informačný symbol

16 – zobrazí sa informačný symbol „Stop“

32 – zobrazí sa informačný symbol „Otázka“

48 – zobrazí sa informačný symbol „Výstraha“

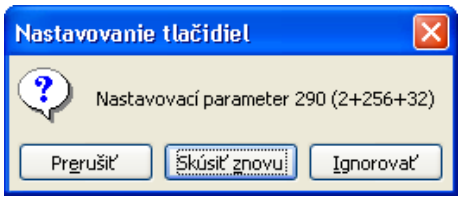
64 – zobrazí sa informačný symbol „Oznam“



Obrázok 19.3: Informačné symboly

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 19.4: 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...

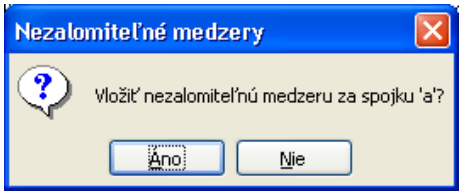
Zobrazenie príslušných tlačidiel však nemá praktický význam, ak sa nedozvieme, ktoré tlačidlo bolo stlačené. Ako sme však už naznačili, funkcia „msgbox“ vracia kódované číslo stlačeného tlačidla podľa nasledujúceho zoznamu:

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 nabudúce.

19.1. Nezalomiteľné medzery



Obrázok 19.5: Nastavenie parametra pre vkladanie nezalomiteľnej 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“ vkladať 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 vyzerať 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("Vložiť 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)"
 end if
 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:

```
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 vkladať 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)"
 end if
 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("Vložiť nezalomiteľnú medzeru za spojku
'a'?", 36,"Nezalomiteľné medzery")
 msgbox("Vložených"+Formatovanie_funkcie.f_Nezalomitelne_Sp
ojky(tlacidlo=6)+"nezalomiteľných 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("Vložiť 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_Medze
ra()+Formatovanie_funkcie.f_Vloz_Interpunkcia_Medzera()
```

```
 kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Datumy()
+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(tl
acidlo=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_Sp
ojky(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_Sp
ojky(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, parameter_2 as typ2,
…, Optional paramater_x as typ_x)
```

V takomto prípade musíme, pochopiteľne, vo funkcii vyriešiť prípad, že tento parameter nebol zadaný 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 zadaný. Makro pre vkladanie nezalomiteľných medzier za spojky potom môže vyzerať 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 zadaný, potom štandardná
hodnota je TRUE
 dim spojky as string
 REM ak parameter spojka_a nie je zadaný, niečim 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
 end if

 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)"
 end if
 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_Sp
ojky(tlacidlo=6)+"nezalomiteľných 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_Viacnasobna_Medzera()
 kolko=kolko+Formatovanie_funkcie.f_Zmaz_Interpunkcia_Medze
ra()+Formatovanie_funkcie.f_Vloz_Interpunkcia_Medzera()
 kolko=kolko+Formatovanie_funkcie.f_Nezalomitelne_Datumy()
+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.

20. 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_suboru [For Mod] [Access Typ_pristupu]
[Ochrana] As #Cislo_suboru [Len=Dlzka_zaznamu]
```

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.

20.1. 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 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 #Cislo_suboru, data
```

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

```
Line Input #Cislo_suboru, retazcova_premenna
```

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

```
Close #Cislo_suboru
```

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

20.2. 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 veci 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 vyzerať 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
 ' 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)
 end if

 ' 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
 end if
 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
 end if
 Citaj_nastavenie_predlozky=male_a
end function
```

Vlastné makro pre vkladanie nezalomiteľných medzier za predložky môže teraz vyzerať 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) "
 end if
 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)
 end if
end sub
```

21. 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.

21.1. 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 „obľúbené“ výrazy (napr. u mňa je to slovo „samozrejme“ vrátane jeho synoným), 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 nahrádzame nájdený reťazec, a výsledok môže vyzerať 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
```

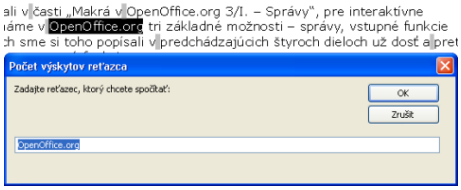
Vlastná procedúra pre zadanie hľadaného reťazca a následné spočítanie jeho výskytu potom môže vyzerať napríklad takto (predpokladajme, že funkciu „Spocitaj_hladanim“ sme uložili do modulu „Formatovanie_funkcie“):

```
sub Spocitaj_slova
 dim slovo as string
```

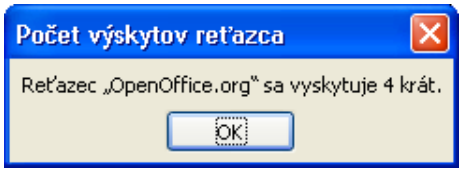
```
 dim kolko as long

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

 Formatovanie_funkcie.Init
 kolko=Formatovanie_funkcie.Spocitaj_hladanim(slovo,False,Tr
ue,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 vyzerať 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())
 end if

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

 Formatovanie_funkcie.Init
 kolko=Formatovanie_funkcie.Spocitaj_hladanim(slovo,False,Tr
ue,False)

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



Obrázok 21.1: Označený text je ponúknutý ako štandardná hodnota ...



Obrázok 21.2: ... a následne spočítaný

21.2. 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 číslovky rozdeliť do nasledujúcich skupín:

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

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

Číslovky od 20 do 50 (prípona „dsať") – dvadsať, tridsať, štyridsať.

Číslovky 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 číslovkách do desať sa musíme osobitne zamerať na číslovky jedna a dva. Obidve sa skloňujú podľa toho, pred čím sú uvedené – jedenmilión, jednamiliarda, dvamilióny, dvemiliardy. Číslovka jedna a dva sa však neskloňuje v zložených číslach (t.j. dvadsaťjedenmiliónov, dvadsaťjedenmiliárd, tridsaťdvamiliárd) a okrem toho sa číslovka jedna zvyčajne neuvádza pred stovkami a vyššími rádmi, 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 sfalšovať, t.j. napr. pri peňažných poukážkach. 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 „dsať" (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, pěttisíc) 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 číslicach, pretože tak sa menia rádové prípony (tisíc, milión, ...). Aby sme nestratili potrebný prehľad, naprogramujeme si najprv pre jednotlivé skupiny (číslovky do 10, číslovky 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
   ' Číslovka „jeden“ sa pri rádoch tisíc, milión,...
nepíše
   if rad=0 then
    pom_nazov="jeden"
   end if
  case 2
   ' Dvojka sa skloňuje – dva, dvesto, dvetisíc, dvamilióny,
dvemiliardy, …
   if (rad mod 2)=0 then
    pom_nazov="dva"
   else
    pom_nazov="dve"
   end if
  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="sedemnásť"
    case 8
     pom_nazov="osemnásť"
    case 9
     pom_nazov="devätnásť"
   end select
  case 2 to 4
   ' "dsať" - dvadsať až štyridsať
   pom_nazov=slovne_jedno_cislo(desiatky,0)+"dsať"+slovne_j
edno_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
neskloňuje (základný tvar je pre rád 0)
   pom_slovne= slovne_jedno_cislo(jednotky,0)
  end if
 else
  ' je to číslo od 10 do 99
  pom_slovne=slovne_desiatky(jednotky, desiatky)
 end if

 ' teraz k prevedenému dvojcifernému číslu pridáme text o
stovkách
 select case stovky
  case 1
   ' Pri stovkách sa jednosto 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.
'
dvemiliardystoštyridsaťsedemmiliónovštyristoosemdesiattriti
sícšesťstoštyridsaťsedem
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 ' „lióny“ - milión, bilióny,
triliónov
   select case cislo
    case 0
     pom_kolko=""
    case 1
     pom_kolko=predpona+"lión"
    case 2 to 4
     pom_kolko=predpona+"lióny"
    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
  end if
 end if
 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 ' rády: 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
   end if
  loop
 end if

 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 vyzerať 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
```

```
 end if

 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
   end if
  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...
   end if
   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"
   end if
   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
   end if
  end if
 end if
```

```
 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.DispatchHel
per")
  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())
 end if
end sub
```

21.3. 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.DispatchHe
lper")
   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())
  end if
 end if
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 oddeľovač 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 oddeľujúci znak z nájdeného slova
vylúčiť, t.j.
  ' vrátime sa o krok späť
  kurzor.goright(1,true)
  poms=kurzor.getstring
 end if
 posledne_slovo=poms
end function
```

Vlastné makro pre vloženie slovného znenia môže potom vyzerať 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
 end if

 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“ volať ihneď po napísaní príslušného čísla bez toho, aby sme za ním zadali čo i len medzeru.

21.4. 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
   end if
  end if
 end if

 ' 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
   end if
  end if
 end if

 ' 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)
 end if
 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 zabudujeme možnosť vkladania „pružnej“ nezalomiteľnej medzery, čo v súčasnosti nedokážu ani najznámejšie platené kancelárske programy.

Dialógy

Začíname štvrtú časť seriálu o programovaní makier v OpenOffice.org, ktorú budeme venovať dialógom. Najskôr však trochu odbočíme a znovu sa vrátime k makru pre vkladanie nezalomiteľných medzier za jednoznakové predložky a spojky.

22. Nezalomiteľné medzery počas písania

Vkladanie nezalomiteľných medzier už počas písania jednoznakových predložiek a spojok je požiadavka, ktorú sme doteraz márne od OpenOffice.org očakávali. Pretože toto makro sa nám zároveň bude vynikajúco hodiť ako východiskový materiál pre dialógy, začneme najprv s ním.

22.1. Obsluha klávesnice

V prípade takejto obsluhy si musíme uvedomiť, že potrebujeme priamo obsluhovať stlačené klávesy. Na tento účel môžeme do systému priradiť vlastné funkcie, ktoré však musia spĺňať určité (UNO) špecifiká. Musia to byť vlastne dve logické funkcie „KeyPressed“ a „KeyReleased“, pričom pomocou prvej z nich obsluhujeme „stlačenie“ a pomocou druhej z nich „pustenie“ klávesy. Obidve tieto funkcie však musia mať rovnaký prefix (t.j. prvú časť názvu), napr. „Nazov_“, pretože práve pomocou tohto prefixu ich zväzujeme so systémom:

```
function Nazov_KeyPressed(StlacenaKlavesa) as boolean
function Nazov_KeyReleased(StlacenaKlavesa) as boolean
createUnoListener("Nazov_","com.sun.star.awt.XKeyHandler")
```

OpenOffice.org predpokladá, že takéto funkcie sú naozaj iba pomocné a väčšinou budeme chcieť využiť aj originálne funkcie na obsluhu klávesnice. Toto zaisťuje automaticky podľa toho, akú výstupnú logickú hodnotu týmto funkciám priradíme. Ak im priradíme hodnotu „TRUE“, potom pôvodné funkcie nevolá (predpokladá, že celú obsluhu sme vykonali sami). Ak im priradíme hodnotu „FALSE“, potom akoby sme žiadnu obsluhu klávesnice nevykonali, resp. naše funkcie sú pokladané naozaj za iba pomocné a volajú sa ešte aj pôvodné (originálne) obslužné funkcie.

22.2. Posledne vložené slovo

Pre vlastné sledovanie predložiek si musíme uvedomiť, že ich nebudeme spätne vyhľadávať, ale po zadaní zalomiteľnej medzery musíme hneď vedieť, či predchádzajúce slovo bola alebo nebola jednoznaková predložka (spojka) a podľa toho sa zariadime. Preto musíme zo stlačených kláves sledovať zadávané slovo (ale iba posledné slovo). V rámci tohto sledovania si musíme ešte uvedomiť, že pri písaní sa aj mýlime a používame aj opravné klávesy (DEL, BACKSPACE, ...). Ako vidíme, táto problematika nie je nijako jednoduchá a to ešte nehovoríme o tom, že sa môžeme kedykoľvek myšou presunúť na úplne iné miesto v texte, tam opraviť nejaký znak a znovu sa vrátiť na terajšie miesto a pod. Aby sme sa dopracovali k pokiaľ možno čo najjednoduchšiemu riešeniu, obmedzíme sa iba na niektoré zo spomenutých problémov a prípadné rozšírenia už necháme na jednotlivých používateľov.

Skôr ako uvedieme zdrojové texty, ešte pripomeňme, že premenné, cez ktoré bu-

deme sprístupňovať dokument a príslušnú obsluhu klávesnice musia byť viditeľné počas celého písania a preto musíme namiesto deklarácie DIM použiť deklaráciu GLOBAL, pretože takto zaistíme, že sa ich definícia a nadobudnuté hodnoty nestratia. Vlastné makrá môžu teraz vyzerať napríklad takto:

```
global ObsluhaDokumentu, ObsluhaKlavesnice as object
global PosledneSlovo, OddelovaceSlov, JednoznakovePredlozky
as string

sub SpustiObsluhuKlavesnice
 on error goto next

 ' Inicializácia premenných
 PosledneSlovo = ""
 JednoznakovePredlozky="aAiIkKoOsSuUvVzZ"
 ' Znaky, ktoré ukončujú slová - medzera, nezalomiteľná
medzera, tabelátor, CR, LF
 OddelovaceSlov=" "+chr(&HA0)+chr(&H09)+chr(&H0A)+chr(&H0D)
 ' pomlčka, lomka, bodka, čiarka, výkričník, otáznik,
bodkočiarka, trojbodka
 OddelovaceSlov=OddelovaceSlov+"-/.,!?;…"
 ' zátvorky
 OddelovaceSlov=OddelovaceSlov+"()[]{}"

 ' Aktuálny dokument
 ObsluhaDokumentu = ThisComponent.getCurrentController
 ' Vlastná obsluha klávesnice
 ObsluhaKlavesnice =
createUnoListener("Klavesnica_","com.sun.star.awt.XKeyHandle
r")
 ' Priradenie tejto obsluhy k aktuálnemu dokumentu
 ObsluhaDokumentu.addKeyHandler(ObsluhaKlavesnice)

end sub

sub UkonciObsluhuKlavesnice
 on error goto next
 ObsluhaDokumentu.removeKeyHandler(ObsluhaKlavesnice)
 PosledneSlovo = ""
end sub

sub VlozZnak(Retazec as string)
' Na aktuálnu pozíciu v dokumente vložíme reťazec
dim ViditelnyKurzor, AktualnyText, AktualnyKurzor as object
 ViditelnyKurzor =
ThisComponent.getCurrentController().getViewCursor()
 AktualnyText = ThisComponent.getText()
 AktualnyKurzor =
AktualnyText.createTextCursorByRange(ViditelnyKurzor.getStart
```

```
())
 AktualnyText.insertString(AktualnyKurzor.getStart(),
Retazec, true)

end sub

function Klavesnica_KeyPressed(StlacenaKlavesa) as boolean
' Obsluha stlačenej klávesy
dim Obsluzena as boolean

 'Predpokláme, že neobslúžime klávesnicu
 Obsluzena=False

 if instr(OddelovaceSlov,StlacenaKlavesa.KeyChar)<>0 then
  ' Došli sme na koniec slova
  if ZistiPredlozku and (StlacenaKlavesa.KeyChar=" ") then
   ' Je to predložka, ktorá končí zalomiteľnou medzerou -
túto nahradíme
   VlozZnak(chr$(&HA0)) ' No Break Space
   ' Sami sme vložili nezalomiteľnú medzeru - obslúžili sme
stlačenú klávesu
   Obsluzena=true
  end if
  ' Boli sme na konci slova, už nás nebude zaujímať
  PosledneSlovo=""
 else
  ' Neboli sme na konci slova
  ' Ešte musíme otestovať riadiace znaky, ktoré nebudeme
sledovať v zadávanom slove
  select case StlacenaKlavesa.KeyChar
   case chr$(&H00) to chr$(&H07)
    PosledneSlovo=""
   case chr$(&H08) ' BackSpace
    if len(PosledneSlovo)>0 then
     PosledneSlovo=left(PosledneSlovo,len(PosledneSlovo)-1)
    else
     PosledneSlovo=""
    end if
   case chr$(&H09) to chr$(&H1F)
    PosledneSlovo=""
   case chr$(&H20) to chr$(&H7E)
    PosledneSlovo=PosledneSlovo+StlacenaKlavesa.KeyChar
   case chr$(&H7F) ' CTRL BS
    PosledneSlovo=""
   case else
    PosledneSlovo=PosledneSlovo+StlacenaKlavesa.KeyChar
  end select
 end if
 Klavesnica_KeyPressed=Obsluzena
```

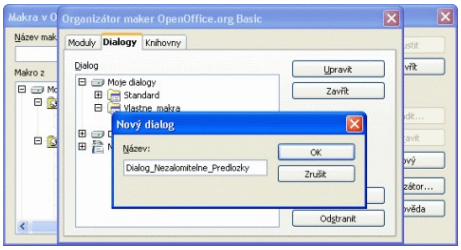
```
end function

function Klavesnica_KeyReleased(StlacenaKlavesa) as boolean
' Obsluha pustenej klávesy
 Klavesnica_KeyReleased = False
end function

function ZistiPredlozku as boolean
dim JePredlozka as boolean

 JePredlozka=false
 if len(PosledneSlovo)=1 then
  ' Ak je to jednoznakové slovo, tak to môže byť predložka
  JePredlozka=instr(1,JednoznakovePredlozky,PosledneSlovo,0)
 end if
 ZistiPredlozku=JePredlozka

end function
```

23. Prvý dialóg

Už sme si ukázali viacero spôsobov, ako nastaviť parametre pre interaktívne riadenie makier v OpenOffice.org. Posledná možnosť, ktorej budeme venovať celú túto časť seriálu sú dialógy, alebo dialógové okná. V dnešnom dieli si ukážeme postup, ako do makier vôbec dokážeme pridať vlastné dialógové okno.

Aby sme dialógy mohli navrhovať, musíme si najprv otvoriť dialógové okno „Makrá v OpenOffice.org Basic“, ku ktorému sa dostaneme cez voľby v menu „Nástroje/Makrá/Správca makier/OpenOffice.org Basic...“. Stlačením tlačidla „Organizátor...“ si sprístupnime dialógové okno „Organizátor makier v OpenOffice.org Basic“, kde vidíme tri záložky: „Moduly“, „Dialógy“ a „Knižnice“. O záložkách „Moduly“ a „Knižnice“ sme si už hovorili na začiatku celého tohto seriálu a preto sa teraz nimi už nebudeme zaoberať, ale pozornosť zameriame na strednú záložku „Dialógy“.

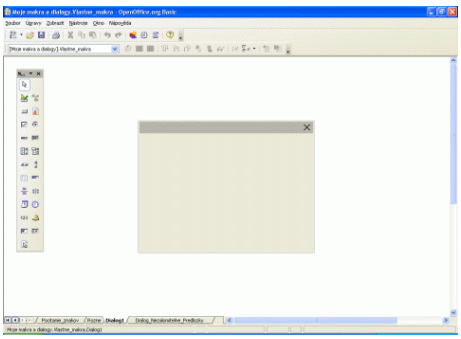
23.1. Organizácia dialógov



Obrázok 23.1: Zadanie názvu nového dialógu

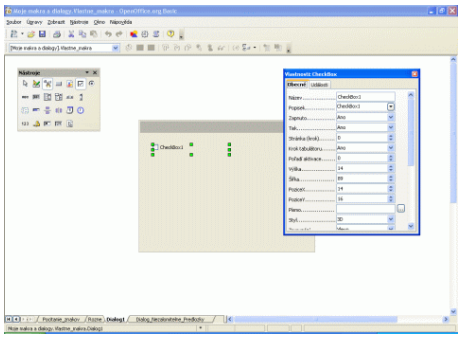
Aj na tejto záložke nájdeme stromovú štruktúru jednotlivých knižníc, po rozbalení ktorých sa nám zobrazí zoznam dialógov, ktoré v nich máme definované. V organizátorovi máme teraz možnosť pridať názvy nových dialógov, ktoré chceme v budúcnosti definovať, takže napr. do knižnice „Vlastne_makra“ si cez tlačidlo „Nový...“ pridáme nový dialóg, ktorý nazveme napr. „Dialog_Nezalomitelne_Predlozky_“. Pozn.: vzhľadom na to, že mám definovaný iný dialóg s takýmto názvom (kde mám definované napr. aj výber medzi vkladaním pružnej a nepružnej nezalomiteľnej medzery), pridal som teraz pracovne znak podčiarknika aj na koniec názvu, aby som to rozoznal. Celý dialóg by však bolo ešte predčasne zverejňovať.

23.2. Vlastný návrh dialógového okna



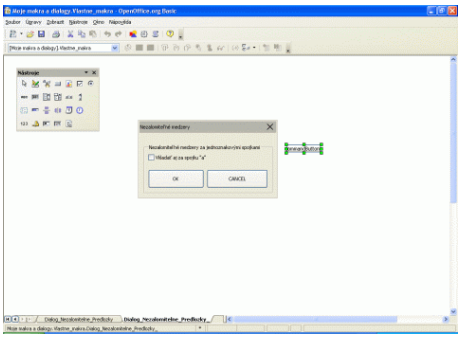
Obrázok 23.2: Pracovné rozhranie pre návrh dialógu

Vlastný dialóg môžeme začať navrhovať pomocou tlačidla „Upraviť“. Týmto sa nám otvorí modul Basic, kde vkladáme aj vlastné makrá,pochopiteľne, na záložke, kde máme pripravené základné okno budúceho dialógu. Ak si dobre všimnete modul Basic zistíte, že v spodnej časti sú záložky, pomocou ktorých sa môžete ľahko prepínať medzi jednotlivými časťami príslušnej knižnice, t.j. ako medzi dialógmi, tak aj príslušnými modulmi s podprogramami. Zároveň v ľavej hornej časti nájdete rozbaľovacie okno, kde môžete rýchlo prepínať medzi jednotlivými knižnicami, takže vám stačí modul Basic otvoriť raz a veľmi jednoducho sa dostanete na ktorékoľvek makro či dialóg v celom vašom systéme makier.



Obrázok 23.3: Vložíme prvok a pozrieme si jeho vlastnosti

Venujme sa však vlastnému návrhu dialógu. V pracovnom okne vidíme dve časti – zoznam nástrojov a prázdne dialógové okno s jediným prvkom – krížikom „X“, ktorý naznačuje budúce miesto skutočného krížika „X“ na jeho zatváranie. Všetko ostatné si teraz musíme zadefinovať. Vlastné definovanie je veľmi jednoduché - v zozname nástrojov si označíme prvok dialógového okna, ktorý tam chceme vložiť, presunieme sa do nášho navrhovaného okna, stlačíme ľavé tlačidlo myši a potiahnutím vyznačíme oblasť, kde sa má prvok umiestniť. Pravdaže, nikto nepredpokladá, že sa na prvý krát presne trafíme na miesto, kde prvok chceme mať a že zároveň presne nastavíme aj jeho veľkosť. Okrem toho budeme následne aj tak nastavovať rôzne texty, ktoré si môžu vynútiť zmenu veľkosti nielen tohto prvku, ale aj celého navrhovaného dialógového okna.



Obrázok 23.4: Prvok môžeme umiestniť aj mimo dialógového okna – návrh sa ešte upraví

Preto si teraz v zozname nástrojov vyberieme výberovú šípku (je to prvý prvok úplne vľavo hore) a označíme si ten prvok, ktorá chceme upravovať. Takýmto istým spôsobom sa dá označiť aj celé dialógové okno, ale nesmiete kliknúť na jeho ľubovoľnú časť, ale iba na okraj. Po označení jednoduchým presúvaním myši môžeme teraz meniť veľkosť tohto okna alebo ho presúvať. Po stlačení pravého tlačidla sa zobrazí kontextové menu, kde môžeme (a aj musíme) nastavovať ďalšie parametre tohto prvku. Hoci sa tomu budeme venovať až nabudúce, už teraz odporúčame vyskúšať, čo urobí zmena parametra „Titulok“ (nebojte sa použiť aj mäkčene a dĺžne). Jednotlivé prvky sa nemusíte báť umiestňovať ani mimo vlastného navrhovaného okna, pretože je to časť

kde robíte návrh a OpenOffice.org predpokladá, že sa s veľkosťou ešte „pohráte“, t.j. nakoniec všetko nastavíte tak, aby sa to zmestilo do vytváraného dialógu.

24. Dialóg pre nezalomiteľnú medzeru

V minulom dieli sme si poukázali na základné možnosti, ktoré nám pri návrhu vlastného dialógového okna ponúka OpenOffice.org. Pokiaľ ste si za domácu úlohu odskúšali, ako sa s týmto modulom pracuje, nebudete mať dnes žiadne problémy.

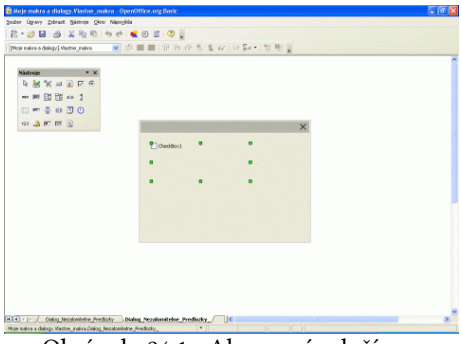
24.1. Zoznam položiek

Ako prvý krok, ktorý si musíme premyslieť, je vlastne zoznam položiek, ktoré v dialógovom okne budeme mať. V prípade, že ideme nastavovať parameter pre vkladanie nezalomiteľnej medzery za spojku „a“ to bude najlepšie zaškrtávacie políčko, pomocou ktorého nastavíme, či sa za túto spojku má alebo nemá vkladať nezalomiteľná medzera. Ďalej si musíme určiť, ako sa má príslušný parameter nastaviť, t.j. či pri zaškrtnutom políčku chceme alebo naopak nechceme vkladať túto medzeru, aby sme podľa toho vhodne formulovali príslušné sprievodné texty.

Tým by sa zdalo, že máme všetky políčka vyčerpané, ale nie je to tak. Pokiaľ totiž chceme aj získať zadané hodnoty, musíme dialógové okno korektne zatvoriť pomocou tlačidla „OK“. Toto tlačidlo si však musíme do neho tak isto zadať. Okrem toho je vhodné, aby sme si zadefinovali aj tlačidlo „Cancel“ pre prípad, že nechceme previesť žiadnu zmenu a nevieme, čo všetko sme vlastne pomenili. Toto tlačidlo sa zatiaľ síce môže zdať nadbytočné, v budúcich verziách však nadobudne svoj význam a preto sa ním budeme zaoberať už teraz.

24.2. Optické členenie a nápovedné texty

Ako posledné, čo si navrhneme, bude rôzne optické členenie pomocou rámikov s nápovedným textom. Týmto spôsobom dokážeme veľmi jednoducho opticky označiť určité skupiny tlačidiel, čo sa nám tak isto zíde najmä v budúcnosti pri komplexnejších návrhoch. Pochopiteľne, úplne na záver nesmieme zabudnúť ani na názov nami navrhovaného dialógu.

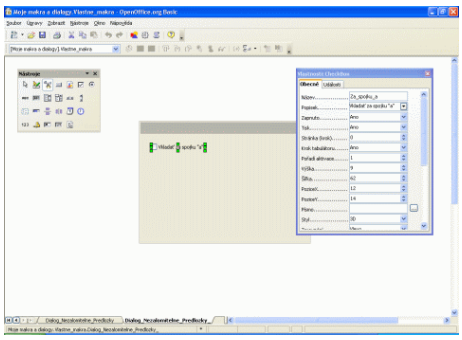


Obrázok 24.1: Ako prvé vložíme zaškrtávacie políčko

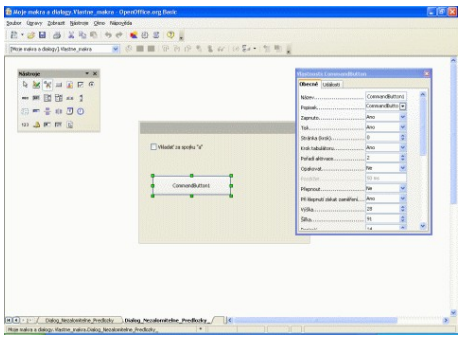
24.3. Zaškrtávacie políčka

Teraz sa však už venujme vlastnému návrhu. Predpokladajme, že sme si otvorili prázdny dialóg, do ktorého teraz postupne budeme vkladať jednotlivé prvky. Aby sme nestratili potrebný prehľad, budeme postupovať presne podľa bodov, ktoré sme si uviedli pred chvíľou a preto ako prvé vložíme zaškrtávacie políčko. Po jeho výbere v poli nástrojov nakreslíme vo vlastnom dialógovom okne radšej väčší obdĺžnik, aby sme mali istotu, že sa tam všetky údaje zmestia. Jeho veľkosť potom aj tak upravíme.

Po jeho nakreslení si ho označíme a cez kontextové menu (pravé tlačidlo myši) zmeníme jeho vlastnosti. Položka „Názov“ je veľmi dôležitá, pretože pomocou nej budeme v budúcnosti získavať nastavené údaje a musí zodpovedať názvosloviu, ktoré používame pri menách premenných v jazyku StarOffice Basic. Pretože sa mi štandardne ponúkané názvy nepáčia, zmeníme ho napr. na „Za_spojku_a“. Ďalej zmeníme položku „Popis“, pretože tento text sa vypisuje za vlastným zaškrtávacím políčkom priamo v dialógovom okne. Povedzme, že chceme, aby zaškrtnuté políčko znamenalo, že sa má za spojku „a“ vkladať nezalomiteľná medzera a preto si sem vložíme napr. tento text: „Vkladať za spojku "a"“. Tým máme všetky potrebné parametre nastavené. Teraz podľa skutočne zobrazeného textu vhodne zmeníme veľkosť zadaného políčka, aby sme ho nemali zbytočne veľký, ale zase aby nebol ani príliš malý. Pokiaľ nám to nevadí, zobrazené údaje „Vlastnosti“ zatvárať nemusíme, aspoň ich budeme mať automaticky prístupné pre všetky ďalšie zadávané políčka, takže ich nebudeme musieť pracne označovať a následne toto okienko otvárať.



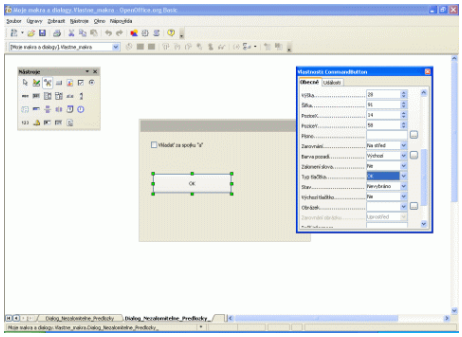
Obrázok 24.2: Zobrazíme si vlastnosti vloženého objektu



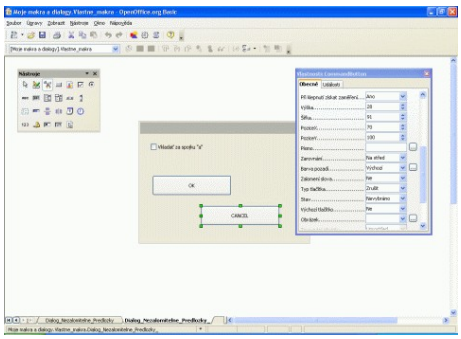
Obrázok 24.3: Ako druhé vložíme tlačidlo ...

24.4. Tlačidlá „OK“ a „Cancel“

Teraz vložíme do dialógového okna tlačidla „OK“ a „Cancel“. V zozname nástrojov si vyberieme symbol tlačidla a následne nakreslíme primerane veľké tlačidlo do nášho návrhu. Vo vlastnostiach zmeníme znovu názov a popis tohto tlačidla. Ako prvé si zadáme tlačidlo „OK“ a tento názov použijeme aj ako názov, tak aj ako popis. Teraz ešte musíme OpenOffice.org oznámiť, že práve toto tlačidlo bude vykonávať túto funkciu. Posunieme sa vo vlastnostiach nadol a v položke „Typ tlačidla“ vyberieme hodnotu „OK“. Podobným spôsobom zadáme teraz tlačidlo „Cancel“ s tým, že ako typ tlačidla vyberieme hodnotu „Zrušiť“.

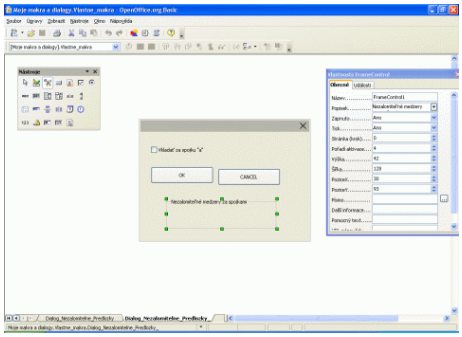


Obrázok 24.4: ... a nastavíme ho na "OK"



Obrázok 24.5: Nesmieme zabudnúť ani na tlačidlo "Cancel"

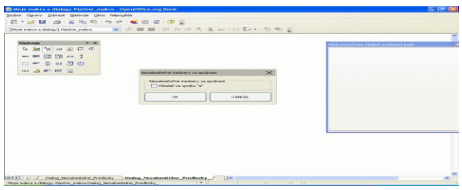
24.5. Okrasné a oddeľujúce prvky



Obrázok 24.6: Ako posledný si vložíme ozdobný rámik

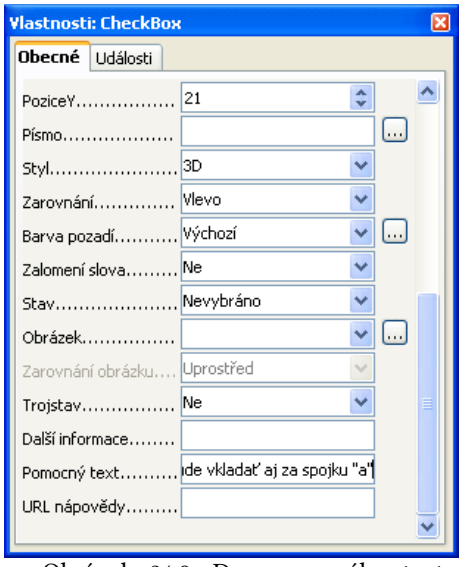
Ako posledný typ sme si povedali, že zadáme rámik, pomocou ktorého budeme oddeľovať jednotlivé skupiny prvkov dialógového okna. V prípade tohto typu obvykle nemeníme jeho názov, pretože sa naň prakticky nikde neodvoláva a nastavujeme iba popis, ktorý v stručnosti vystihuje orámované prvky dialógového okna. Nakoniec si označíme hlavné dialógové okno a zadáme jeho popis.

24.6. Celkové „kozmetické“ úpravy



Obrázok 24.7: Dialógové okno nakoniec upravíme podľa nášho vkusu

Pretože máme zadané už všetky prvky a texty, prevedieme si teraz celkové rozmiestnenie tak, aby sme dosiahli nami požadovanú estetickú úroveň, t.j. jednotlivé prvky teraz budeme postupne označovať, premiestňovať a meniť ich veľkosť. Pokiaľ chceme, môžeme veľkostné údaje a umiestnenie zadávať aj číselne priamo v parametroch tlačidiel, takže návrh môžeme previesť naozaj veľmi detailne a podrobne. Okrem toho môžeme pri každom prvku v jeho vlastnostiach zadať do položky „Pomocný text“ aj ľubovoľný nápovedný text, ktorý sa bude zobrazovať v žltom okienku, keď sa myšou premiestnime nad tento prvok, napr. k zaštrávaciemu okienku napíšeme takýto nápovedný text: „Nezalomiteľná medzera sa bude vkladať aj za spojku "a"“. Pravdaže, tieto nápovedné texty môžeme pridať k ľubovoľnému prvku (teda aj k textovému rámiku, tlačidlu „OK“, ...), čím môžeme pripraviť dialógové okno aj pre menej zdatných používateľov.



Obrázok 24.8: Do pomocného textu môžeme zadať stručnú nápovedu

25. Základné funkcie pre prácu s dialógmi

V minulom dieli nášho seriálu sme si navrhli prvé dialógové okno, pomocou ktorého chceme nastavovať parameter pre riadenie vkladania nezalomiteľnej medzery za spojku „a“. Tento návrh vlastne spočíval z výberu vhodných položiek (políčok, prvkov), ktoré sme pre náš dialóg požadovali a ich následnej grafickej úpravy vrátane vhodného textového sprievodu.

Pochopiteľne, takýto návrh musíme vždy urobiť ako prvý, pretože až na jeho základe môžeme s takto pripraveným dialógovým oknom pracovať. A práve na túto

prácu – t.j. otvorenie dialógového okna, nastavovanie a čítanie údajov z jednotlivých políčok až po jeho zatvorenie sa zameriame práve dnes.

25.1. Dialóg ako objekt v jazyku StarOffice Basic

Z hľadiska programovacieho jazyka StarOffice Basic je dialóg špecifický objekt s množinou preddefinovaných funkcií. Teraz sa musíme vrátiť k druhému dielu, v ktorom sme spomínali, že všetky dialógy máme vlastne uložené v nami zadefinovaných knižniciach, citujem: „... Aj na tejto záložke nájdeme stromovú štruktúru jednotlivých knižníc, po rozbalení ktorých sa nám zobrazí zoznam dialógov, ktoré v nich máme definované. V organizátorovi máme teraz možnosť pridať názvy nových dialógov, ktoré chceme v budúcnosti definovať, takže napr. do knižnice "Vlastne_makra" si cez tlačidlo "Nový..." pridáme nový dialóg, ktorý nazveme napr. "Dialog_Nezalomitelne_Predlozky_"...“

Predpokladajme, že sme dodržali túto štruktúru. K nášmu dialógu sa potom dostaneme pomocou nasledujúcej postupnosti príkazov:

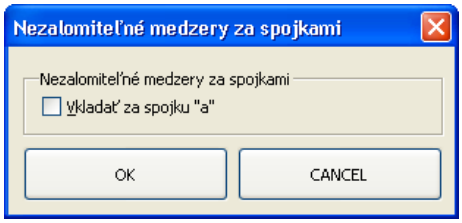
```
  REM Definícia premennej, pomocou ktorej budeme
pristupovať k nášmu dialógu
  dim dlg as object

  REM Sprístupnenie knižnice Vlastne_makra
  DialogLibraries.LoadLibrary("Vlastne_makra")

  REM Priradenie dialógu Dialog_Nezalomitelne_Predlozky_ do
premennej dlg
  dlg=CreateUnoDialog(DialogLibraries.Vlastne_makra.Dialog_
Nezalomitelne_Predlozky_)
```

Keď už máme dialógové okno sprístupnené, môžeme ho otvoriť a zatvoriť. Zatvorenie je jednoduché, pretože sa môže urobiť kliknutím na štandardné zatváracie tlačidlo okien (de facto sa správa ako tlačidlo „Cancel“), alebo, ak sme definovali tlačidlá „OK“ a „Cancel“, tak kliknutím na jedno z nich. Pravdaže, v programe nás bude následne zaujímať, ako sme dialóg vlastne ukončili. Všetky tieto činnosti (t.j. otvorenie a oznámenie, ako sme dialógové okno zatvorili) nám zabezpečuje jedna jediná funkcia „Execute“, ktorá je súčasťou dialógu:

```
  if dlg.Execute()=1 then
   REM Dialógové okno sme ukončili klávesou „OK“
  else ' nadobúda hodnotu 0
   REM Dialógové okno sme ukončili klávesou „CANCEL“
  end if
```

Pokiaľ už s dialógovým oknom nebudeme pracovať, musíme ho nakoniec uvoľniť z pamäte príkazom:

```
  dlg.dispose()
```



Obrázok 25.1: Takto vyzerá naše dialógové okno po jeho zavolaní

To by z funkcií pre prácu s celým dialógovým oknom mohlo zatiaľ stačiť. Ešte stále sme sa však nedostali k jednotlivým položkám (okrem tlačidiel „OK“ a „Cancel“), ktoré máme v dialógovom okne definované a ktoré nás, pochopiteľne, zaujímajú najviac. Všetky položky, ktoré sme do dialógového okna vkladali sú prístupné v časti „model“, za ktorým uvádzame ich názov. Ak máme dialógové okno prístupné (predpokladajme že v premennej dlg), k nami zadefinovanému zaškrtávaciemu okienku „Vkladať za spojku a“, ktoré sme minule vo vlastnostiach nazvali „Za_spojku_a“ sa dostaneme nasledovne:

```
dlg.model.Za_spojku_a
```

Týmto spôsobom sa sprístupňuje celý rad parametroch a hodnôt, ktoré pre túto položku môžeme nastavovať. Zatiaľ sa však budeme zaoberať iba jednou z nich – jej hodnotou. Táto je následne prístupná cez položku „State“:

```
dlg.model.Za_spojku_a.State
```

Pochopiteľne, programovací jazyk StarOffice Basic má štandardne nastavené počiatočné hodnoty všetkých typov položiek, ktoré do dialógového okna môžeme vložiť. V prípade zaškrtávacieho okienka je to hodnota 0, t.j. okienko nie je zaškrtnuté. Vlastnému nastavovaniu a jeho následnému odpamätávaniu do pracovného súboru na disku sa však budeme venovať až nabudúce.

26. Nastavovanie parametrov pomocou dialógu

V poslednom dieli nášho seriálu sme si stručne načrtli funkcie pre prácu s dialógmi. Na základe toho môžeme teraz pristúpiť k ďalšiemu kroku, v ktorom dialóg využijeme pre vlastné nastavenie parametra v našich podprogramoch ohľadne predložky „a“. Pretože všetky potrebné funkcie sme v nižších verziách už uviedli, môžeme priamo pristúpiť k vlastnému programovaniu bez toho, aby sme obzvlášť museli komentovať jednotlivé podprogramy.

26.1. Funkcie pre odpamätanie nastavení do súboru

```
global ObsluhaDokumentu, ObsluhaKlavesnice as object
global PosledneSlovo, OddelovaceSlov, JednoznakovePredlozky
as string
global Priznak_nezalomitelna_a as integer

REM Funkcie pre prácu so súborom, kde sú nastavené príznaky

function Subor_nezalomitelne_predlozky as string
 Subor_nezalomitelne_predlozky="file:///c:/Program
files/OpenOffice.org 2.0/Nezalomitelna_Predlozka_a.dat"
end function

sub Init_zoznam_predloziek
 JednoznakovePredlozky="AiIkKoOsSuUvVzZ"
```

Nastavovanie parametrov pomocou dialógu

```
 if Priznak_nezalomitelna_a=1 then
  JednoznakovePredlozky="a"+JednoznakovePredlozky
 end if
end sub

sub Uloz_nastavenie_predlozky
 dim i, subor as integer

 subor = Freefile
 open Subor_nezalomitelne_predlozky for Output as #subor
 print #subor,str(Priznak_nezalomitelna_a)
 close #subor
end sub

sub Citaj_nastavenie_predlozky
 dim i, subor as integer
 dim riadok as string
 ' Inicializácia na štandardné štandardné hodnoty
 Priznak_nezalomitelna_a=1
 next i

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

 ' Teraz súbor určite existuje - načítame ho
 subor = Freefile
  open Subor_nezalomitelne_predlozky for Input as #subor
 i=0
 if not eof(subor)
  i=i+1
  Line Input #subor, riadok
  Priznak_nezalomitelna_a=val(riadok)
 end if
 close #subor

 ' Ak v súbore chýbal nejaký údaj, tak ho opravíme
 if i<1 then
  Uloz_nastavenie_predlozky
 end if
 Init_zoznam_predloziek

end sub
```

26.2. Funkcie pre vlastnú prácu s dialógovým oknom

Teraz si napíšeme funkcie pre vlastnú prácu s dialógovým oknom „Dialog_nezalomi-

telne_predlozky_“. Pretože sme sa tomuto ešte nevenovali podrobne, budeme príslušné funkcie trochu viac komentovať.

```
REM Procedúra, pomocou ktorej nastavíme počiatočné hodnoty
príslušných položiek dialógového okna
sub init_dialog_predlozky(oDialog as object)

 REM Načítame ich zo súboru
 Citaj_nastavenie_predlozky

 REM Nastavenie počiatočnej hodnoty podľa toho, ako ju máme
odpamätanú v inicializačnom súbore
 oDialog.model.Za_spojku_a.State=Priznak_nezalomitelna_a

end sub

REM Procedúra, pomocou ktorej zapíšeme získané hodnoty z
príslušných položiek dialógového okna do inicializačného
súboru
sub zapis_dialog_predlozky(oDialog as object)
 Priznak_nezalomitelna_a=oDialog.model.Za_spojku_a.State

 Uloz_nastavenie_predlozky

 REM Po uložení do súboru je vhodné, aby sme si zmenené
nastavenie aj zmenili v globálnych premenných
 Init_zoznam_predloziek

end sub

REM Vlastné volanie dialógu s príslušnými nastaveniami
sub Nastav_Priznak_Nezalomitelne_a
 dim dlg as object

 REM Sprístupnenie knižnice
 DialogLibraries.LoadLibrary("Vlastne_makra")

 REM Sprístupnenie dialógového okna
 dlg=CreateUnoDialog(DialogLibraries.Vlastne_makra.Dialog_N
ezalomitelne_Predlozky_)

 REM Inicializácia počiatočných hodnôt
 init_dialog_predlozky(dlg)

 REM Vlastné volanie dialógového okna
 if dlg.Execute()=1 then
  REM Ak sme ukončili tlačidlom „OK“, tak zmenené hodnoty
zapíšeme
  zapis_dialog_predlozky(dlg)
```

```
 end if

 REM Uvoľnenie dialógového okna z pamäte
 dlg.dispose()

end sub
```

26.3. Vlastné vkladanie nezalomiteľných medzier počas ich písania

Celé nastavenie vykonáme zavolaním poslednej procedúry „Nastav_Priznak_Nezalomitelne_a“. Teraz už môžeme pridať procedúry pre vkladanie nezalomiteľných medzier za jednoznakové predložky počas ich písania. Na rozdiel od už uverejnenej verzie budeme teraz nastavovať zoznam predložiek podľa inicializačného súboru (volanie funkcie „Citaj_nastavenie_predlozky„).

```
sub SpustiObsluhuKlavesnice
 on error goto next
 ' Inicializácia premenných
 Citaj_nastavenie_predlozky
 PosledneSlovo = ""
 ' Znaky, ktoré ukončujú slová - medzera, nezalomiteľná
medzera, tabelátor, CR, LF
 OddelovaceSlov=" "+chr(&HA0)+chr(&H09)+chr(&H0A)+chr(&H0D)
 ' pomlčka, lomka, bodka, čiarka, výkričník, otáznik,
bodkočiarka, trojbodka
 OddelovaceSlov=OddelovaceSlov+"-/.,!?;…"
 ' zátvorky
 OddelovaceSlov=OddelovaceSlov+"()[]{}"

 ' Aktuálny dokument
 ObsluhaDokumentu = ThisComponent.getCurrentController
 ' Vlastná obsluha klávesnice
 ObsluhaKlavesnice =
createUnoListener("Klavesnica_","com.sun.star.awt.XKeyHandle
r")
 ' Priradenie tejto obsluhy k aktuálnemu dokumentu
 ObsluhaDokumentu.addKeyHandler(ObsluhaKlavesnice)

end sub

sub UkonciObsluhuKlavesnice
 on error goto next
 ObsluhaDokumentu.removeKeyHandler(ObsluhaKlavesnice)
 PosledneSlovo = ""
end sub
sub VlozZnak(Retazec as string)
' Na aktuálnu pozíciu v dokumente vložíme reťazec
dim ViditelnyKurzor, AktualnyText, AktualnyKurzor as object
```

```
 ViditelnyKurzor =
ThisComponent.getCurrentController().getViewCursor()
 AktualnyText = ThisComponent.getText()
 AktualnyKurzor =
AktualnyText.createTextCursorByRange(ViditelnyKurzor.getStart
())
 AktualnyText.insertString(AktualnyKurzor.getStart(),
Retazec, true)

end sub

function Klavesnica_KeyPressed(StlacenaKlavesa) as boolean
' Obsluha stlačenej klávesy
dim Obsluzena as boolean

 'Predpokláme, že neobslúžime klávesnicu
 Obsluzena=False

 if instr(OddelovaceSlov,StlacenaKlavesa.KeyChar)<>0 then
  ' Došli sme na koniec slova
  if ZistiPredlozku and (StlacenaKlavesa.KeyChar=" ") then
   ' Je to predložka, ktorá končí zalomiteľnou medzerou -
túto nahradíme
   VlozZnak(chr$(&HA0)) ' No Break Space
   ' Sami sme vložili nezalomiteľnú medzeru - obslúžili sme
stlačenú klávesu
   Obsluzena=true
  end if
  ' Boli sme na konci slova, už nás nebude zaujímať
  PosledneSlovo=""
 else
  ' Neboli sme na konci slova
  ' Ešte musíme otestovať riadiace znaky, ktoré nebudeme
sledovať v zadávanom slove
  select case StlacenaKlavesa.KeyChar
   case chr$(&H00) to chr$(&H07)
    PosledneSlovo=""
   case chr$(&H08) ' BackSpace
    if len(PosledneSlovo)>0 then
     PosledneSlovo=left(PosledneSlovo,len(PosledneSlovo)-1)
    else
     PosledneSlovo=""
    end if
   case chr$(&H09) to chr$(&H1F)
    PosledneSlovo=""
   case chr$(&H20) to chr$(&H7E)
    PosledneSlovo=PosledneSlovo+StlacenaKlavesa.KeyChar
   case chr$(&H7F) ' CTRL BS
    PosledneSlovo=""
```

```
   case else
    PosledneSlovo=PosledneSlovo+StlacenaKlavesa.KeyChar
  end select
 end if
 Klavesnica_KeyPressed=Obsluzena
end function

function Klavesnica_KeyReleased(StlacenaKlavesa) as boolean
' Obsluha pustenej klávesy
 Klavesnica_KeyReleased = False
end function

function ZistiPredlozku as boolean
dim JePredlozka as boolean

 JePredlozka=false
 if len(PosledneSlovo)=1 then
  ' Ak je to jednoznakové slovo, tak to môže byť predložka
  JePredlozka=instr(1,JednoznakovePredlozky,PosledneSlovo,0)
 end if
 ZistiPredlozku=JePredlozka

end function
```

27. Pružná nezalomiteľná medzera

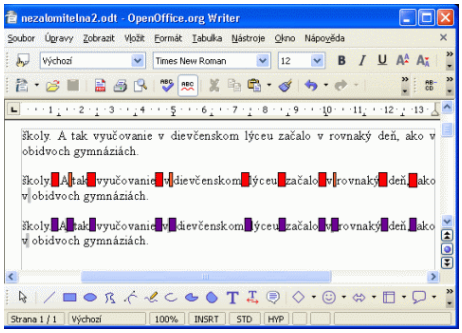
Hneď na úvod si zopakujme, čo to vlastne nezalomiteľná medzera je. Podľa slovenských a českých typografických pravidiel sa na konci riadku nemajú vyskytovať jednoznakové predložky a spojky (a, i, k, o, s, u, v, z). Z hľadiska práce na počítači je preto potrebné za ne vložiť vhodný znak (nezalomiteľnú medzeru), ktorá zabezpečí, že ak sa vyskytnú ma konci riadku, tak sa automaticky zaradia na riadok nový.

Pokiaľ používame zarovnanie textu do bloku (a toto zarovnanie sa používa veľmi často), textový editor musí dosiahnuť, že slová budú spolu a zároveň roztiahnuté na celý riadok. To sa dá dosiahnuť iba jedným spôsobom – „roztiahnutím“ medzier. Pravdaže, každý textový editor, ktorý umožňuje zarovnávať text do bloku to aj robí. Jedná sa vlastne o veľmi jednoduchú „matematiku“ stačí zistiť, koľko voľného priestoru medzi slovami máme a vydeliť ho počtom medzier. Pravdaže, v praxi sa stane, že toto číslo nebude celé, tak program potom upraví jednotlivé medzery tak, že niektoré urobí o jeden bod väčšie.

27.1. Čo na to hovorí matematika

Matematicky vynikajúce riešenie... pokiaľ však nepoužijeme nezalomiteľné medzery. Pokiaľ si dobre všimnete, žiadny z textových editorov tieto medzery nezväčšuje, t. j. ponecháva im konštantnú veľkosť a zväčšuje iba „obyčajné“ medzery. Týmto spôsobom sa však text značne deformuje a mnohokrát pôsobí opticky veľmi disharmonicky. Pravdaže, mnohým to nevadí, ale ak budete čítať desiatky strán textu, oči sa omnoho skôr unavia pri nerovnomernom rozloženom texte. Na nasledujúcom obrázku je použi-

tý príklad, kde v prvom riadku vidíte obyčajné medzery, v druhom nezalomiteľné a v treťom pružne roztiahnuté nezalomiteľné medzery. Na tomto mieste upozorňujem, že sa nejedná o vymyslený text, ale o príklad z rukopisu románu s presne nastavenými okrajmi, ktorý spolu s manželkou vydávame ku koncu tohto roku. Myslím si, že farebné rozdiely vyjadrujúce veľkosť príslušných medzier sú dostatočne výrazné.



Obrázok 27.1: Rozdiely medzi rôznymi typmi medzier

27.2. A čo na to hovoria kódy znakov

Dosť však bolo teórie, ale pozrime sa na to, ako sa dá dosiahnuť „pružná nezalomiteľná“ medzera. Po dlhom hľadaní som v zozname unikódov (verzia 4.1) našiel znak, ktorý umožnil do textu vkladať takéto medzery. Prísne vzaté, nejedná sa o žiadnu medzeru, ale o znak „Word Joiner“ (hexadecimálny kód 2060 – v prípade kódov budeme väčšinou používať iba hexadecimálne kódy), ktorý vlastne do textu vkladá nezalomiteľnú medzeru nulovej veľkosti, čím dokáže spojiť dve za sebou idúce slová. Žiaľ, mnoho programov nedokáže dobre zobrazovať tieto unikódy. Medzi tie, ktoré ho zobrazujú správne patrí okrem OpenOffice.org napr. Adobe Reader, Foxit Reader, Tomahawk, Mozilla Firefox, GSView a pod. Medzi tie, ktoré ho správne nezobrazujú patria programy MS Office, Opera a pod. Aby ste si mohli odskúšať, ako sa zobrazuje tento znak vo vami obľúbených (či neobľúbených) programoch vrátane internetového prehliadača, prikladám teraz tri riadky textu. V prvom sú použité iba obyčajné medzery, v druhom iba nezalomiteľné a v treťom znak „Word Joiner“ spolu s obyčajnou medzerou za ním.

V tomto riadku sú všetky medzery obyčajné.

V tomto riadku sú všetky medzery nedeliteľné.

V tomto riadku je pred každou medzerou znak „Word Joiner“.

28. Aké medzery vlastne používať

Naposledy sme si uviedli tri rôzne možnosti vkladania medzier za jednoznakové predložky a spojky – obyčajné medzery (vtedy vlastne netreba nič robiť, pretože to je štandardne nastavená hodnota v OpenOffice.org), nahrádzanie tejto medzery znakom nezalomiteľnej medzery („Non Break Space“ – hexadecimálny kód A0) alebo vkladanie znaku spájania slov („Word Joiner“ – hexadecimálny kód 2060) pred obyčajnú medzeru.

28.1. Obyčajne potrebujeme všetky tri typy

Osobne môžem potvrdiť, že skoro každodenne potrebujem používať všetky tri uvedené „typy“ medzier, takže sa to bude týkať aj ostatných používateľov. Aj preto som na konci minulého dielu uviedol tieto tri vety:

V tomto riadku sú všetky medzery obyčajné.

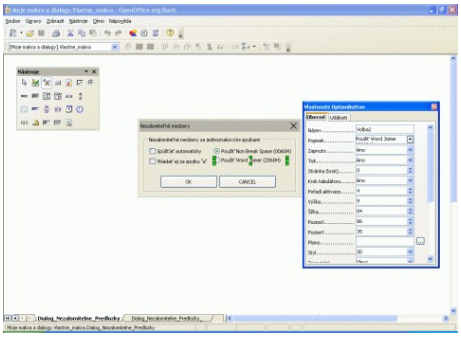
V tomto riadku sú všetky medzery nedeliteľné.

V tomto riadku je pred každou medzerou znak „Word Joiner“.

Aby si každý vedel predstaviť, že nie vždy sa nám môže hodiť to najlepšie riešenie. Pokiaľ potrebujem uverejniť napríklad tento článok na stránkach www.inet.sk, používam bežné nezalomiteľné medzery, ktoré sa správne zobrazujú vo všetkých prehliadačoch, pretože by si ich používatelia Internet Explorera či Opery (iné som neoveroval) nemohli dobre prečítať. To isté chcem dosiahnuť pri zasielaní správ rôznym externým organizáciám (v RTF či DOC formáte) a tam používam ako nezalomiteľné, tak aj obyčajné medzery.

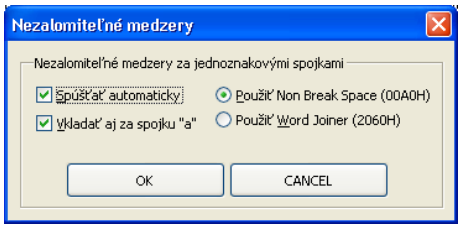
28.2. Zadefinujeme si nové dialógové okno s prepínacími položkami

Ako vidíme, problematika týchto znakov nie je práve jednoduchá a preto je dobré, ak si nastavíme možnosť rôzneho prepínania medzi nimi. Za týmto účelom si zadefinujeme dialóg „Dialog_Nezalomitelne_Predlozky“ (teraz som už vynechal znak podčiarnika na konci, pretože tento dialóg skutočne používam už dlhšiu dobu). Nebudeme sa už venovať položkám, ktoré sme už spomínali a priamo pristúpime k pridávaniu prepínacích položiek. Tieto nájdeme ukryté v známych symboloch „krúžkov“. Je dobré, ak ich zadávame za sebou. V zozname vlastností si musíme navyše všímať hodnotu „Poradie aktivácie“, pretože s týmto číslom budeme ešte zaobchádzať (v našom príklade to bude 3 a 4, pretože som tieto položky zadal v tomto poradí). Pravdaže, toto poradie sa dá meniť, ale treba si dať pozor, pretože OpenOffice.org automaticky prečísluje ostatné položky v prípade, že dôjde ku konfliktu.



Obrázok 28.1: Definícia dialógového okna s prepínacími položkami

Pretože niektorí používatelia budú chcieť, aby sa im makro pre vkladanie nezalomiteľných medzier za jednoznakové predložky a spojky automaticky spúšťalo pri otvorení dokumentu (ale iní – ako napr. ja zase nie), pridáme si do nášho dialógu ešte ďalšie zaškrtávacie okienko, ktorým budeme nastavovať aj tento parameter. Celkové dialógové okno môže potom po spustení vyzerať napríklad takto:



Obrázok 28.2: Záverečný vzhľad dialógového okna pre nastavenie parametrov nezalomiteľných medzier

Teraz nám už nič nebráni v tom, aby sme si naprogramovali jednotlivé funkcie pre vlastnú prácu s dialógovým oknom a vkladaním nezalomiteľných medzier. Pravdaže, vzhľadom na väčší počet nastavovaných parametrov je potrebné vymazať príslušný dátový súbor, v ktorom máme zapamätané doterajšie nastavenia (však sa hneď pri prvom spustení obnoví):

```
global ObsluhaDokumentu, ObsluhaKlavesnice as object
global PosledneSlovo, OddelovaceSlov, JednoznakovePredlozky
as string
global Priznaky_nezalomitelne_medzery(3) as integer
global Obsluhovat_Klavesnicu as boolean
```

```
function Subor_nezalomitelne_predlozky as string
 Subor_nezalomitelne_predlozky="file:///c:/Program
files/OpenOffice.org 2.0/Nezalomitelne_Predlozky.dat"
end function

sub Init_zoznam_predloziek
 JednoznakovePredlozky="AiIkKoOsSuUvVzZ"
 if Priznaky_nezalomitelne_medzery(2)=1 then
  JednoznakovePredlozky="a"+JednoznakovePredlozky
 end if
end sub

sub Uloz_nastavenie_predlozky
 dim i, subor as integer
 subor = Freefile
 open Subor_nezalomitelne_predlozky for Output as #subor
 for i=1 to 3
  print #subor,str(Priznaky_nezalomitelne_medzery(i))
 next i
 close #subor
end sub

sub Citaj_nastavenie_predlozky
 dim i, subor as integer
 dim riadok as string
 ' Inicializácia na štandardné štandardné hodnoty
 for i=1 to 3
  Priznaky_nezalomitelne_medzery(i)=1
 next i

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

 ' Teraz súbor určite existuje - načítame ho
 subor = Freefile
  open Subor_nezalomitelne_predlozky for Input as #subor
 i=0
 While not eof(subor)
  i=i+1
  Line Input #subor, riadok
  Priznaky_nezalomitelne_medzery(i)=val(riadok)
 wend
 close #subor

 ' Ak v súbore chýbal nejaký údaj, tak to opravíme
 if i<3 then
```

```
  Uloz_nastavenie_predlozky
 end if
 Init_zoznam_predloziek
end sub

sub init_dialog_predlozky(oDialog as object)
 Citaj_nastavenie_predlozky
 oDialog.model.Automaticky.State=Priznaky_nezalomitelne_medz
ery(1)
 oDialog.model.Za_spojku_a.State=Priznaky_nezalomitelne_med
zery(2)
 oDialog.model.TabIndex=3
 oDialog.model.TabIndex=4
 select case Priznaky_nezalomitelne_medzery(3)
  case 2
   oDialog.model.Volba1.State=0
   oDialog.model.Volba2.State=1
  case else
   oDialog.model.Volba1.State=1
   oDialog.model.Volba2.State=0
 end select
end sub

sub zapis_dialog_predlozky(oDialog as object)
 Priznaky_nezalomitelne_medzery(1)=oDialog.model.Automatick
y.State
 Priznaky_nezalomitelne_medzery(2)=oDialog.model.Za_spojku_
a.State
 if oDialog.model.Volba1.State=1 then
  Priznaky_nezalomitelne_medzery(3)=1
 else
  Priznaky_nezalomitelne_medzery(3)=2
 end if
 Uloz_nastavenie_predlozky
 ' A ešte podľa nového nastavenia zmeníme zoznam predložiek
 Init_zoznam_predloziek
end sub

sub NastavPriznakyNezalomitelnePredlozky
dim dlg as object
 DialogLibraries.LoadLibrary("Vlastne_makra")

 dlg=CreateUnoDialog(DialogLibraries.Vlastne_makra.Dialog_N
ezalomitelne_Predlozky)
 init_dialog_predlozky(dlg)

 if dlg.Execute()=1 then
  zapis_dialog_predlozky(dlg)
 end if
```

```
 dlg.dispose()
end sub

sub SpustiObsluhuKlavesnice
 on error goto next
 if
ThisComponent.supportsService("com.sun.star.text.TextDocume
nt") then
   ' Sme v textovom dokumente, kde chceme nahrádzať medzery
   Obsluhovat_Klavesnicu=True
   PosledneSlovo = ""
   ' medzera, nezalomiteľná medzera, tabelátor, CR, LF
   OddelovaceSlov=" "+chr(&HA0)+chr(&H09)+chr(&H0A)
+chr(&H0D)
   ' pomlčka, lomka, bodka, čiarka, výkričník, otáznik,
bodkočiarka, trojbodka
   OddelovaceSlov=OddelovaceSlov+"-/.,!?;…"
   ' zátvorky
   OddelovaceSlov=OddelovaceSlov+"()[]{}"
   ObsluhaDokumentu = ThisComponent.getCurrentController
   ObsluhaKlavesnice =
createUnoListener("Klavesnica_","com.sun.star.awt.XKeyHandle
r")
   ObsluhaDokumentu.addKeyHandler(ObsluhaKlavesnice)
 end if
end sub
sub UkonciObsluhuKlavesnice
 on error goto next
 if
ThisComponent.supportsService("com.sun.star.text.TextDocume
nt") then
   ' Sme v textovom dokumente a chceme ukončiť obsluhu
klávesnice
    ObsluhaDokumentu.removeKeyHandler(ObsluhaKlavesnice)
    PosledneSlovo = ""
 end if
end sub

function VlozZnak(Retazec as string)
' Na aktuálnu pozíciu v dokumente vložíme reťazec
dim ViditelnyKurzor, AktualnyText, AktualnyKurzor as object

 ViditelnyKurzor =
ThisComponent.getCurrentController().getViewCursor()
 AktualnyText = ThisComponent.getText()
 AktualnyKurzor =
AktualnyText.createTextCursorByRange(ViditelnyKurzor.getStart
() )
 AktualnyText.insertString(AktualnyKurzor.getStart(),
```

```
Retazec, true)

end function

function Klavesnica_KeyPressed(StlacenaKlavesa) as boolean
' Obsluha stlačenej klávesy
dim Obsluzena as boolean

 'Predpokláme, že nebudeme vkladať nezalomiteľnú medzeru
 Obsluzena=False
 if instr(OddelovaceSlov,StlacenaKlavesa.KeyChar)<>0 then
  ' Došli sme na koniec slova
  if Obsluhovat_Klavesnicu then
   if ZistiPredlozku and (StlacenaKlavesa.KeyChar=" ") then
    ' Je to predložka, ktorá končí zalomiteľnou medzerou -
túto nahradíme
    if Priznaky_nezalomitelne_medzery(3)=1 then
     VlozZnak(chr$(&HA0)) ' No Break Space
     ' Sami sme vložili medzeru - obslúžili sme stlačenú
klávesu
     Obsluzena=true
    else
     VlozZnak(chr$(&H2060)) ' Word Joiner
     ' V tomto prípade sa musí ešte vložiť aj zadaná medzera
a preto sme neobslúžili stlačenú klávesu
    end if
   end if
  end if
  PosledneSlovo=""
 else
  ' Ešte musíme otestovať riadiace znaky, ktoré nebudeme
sledovať v zadávanom slove
  select case StlacenaKlavesa.KeyChar
   case chr$(&H00) to chr$(&H07)
    PosledneSlovo=""
   case chr$(&H08) ' BS
    if len(PosledneSlovo)>0 then
     PosledneSlovo=left(PosledneSlovo,len(PosledneSlovo)-1)
    else
     PosledneSlovo=""
    end if
   case chr$(&H09) to chr$(&H1F)
    PosledneSlovo=""
   case chr$(&H20) to chr$(&H7E)
    PosledneSlovo=PosledneSlovo+StlacenaKlavesa.KeyChar
   case chr$(&H7F) ' CTRL BS
    PosledneSlovo=""
   case else
    PosledneSlovo=PosledneSlovo+StlacenaKlavesa.KeyChar
```

```
  end select
 end if
 Klavesnica_KeyPressed=Obsluzena
end function

function Klavesnica_KeyReleased(StlacenaKlavesa) as boolean
' Obsluha pustenej klávesy
 Klavesnica_KeyReleased = False
end function

function ZistiPredlozku as boolean
dim JePredlozka as boolean

 JePredlozka=false
 if len(PosledneSlovo)=1 then
  ' Ak je to jednoznakové slovo, tak to môže byť predložka
  JePredlozka=instr(1,JednoznakovePredlozky,PosledneSlovo,0)
 end if
 ZistiPredlozku=JePredlozka
end function
```

28.3. Inicializácia prepínačov

Teraz k vlastným prepínačom. Inicializovali sme ich týmito dvomi príkazmi:

```
  oDialog.model.TabIndex=3
  oDialog.model.TabIndex=4
```

OpenOffice.org sa správa tak, že ak uvedieme bezprostredne za sebou bez prerušenia iným príkazom zoznam prepínačov, považuje ich za jeden zoznam a bude prepínať automaticky medzi nimi, t.j. označením jednej položky zruší označenie inej. K uvedeným funkciám ešte musíme naprogramovať aj podporné obslužné programy, resp. ich priradiť k akciám ako je otvorenie či zatvorenie súboru, ale tomu všetkému sa budeme venovať až na budúce.

29. Ako si v tom udržať poriadok a prehľad

V minulom dieli sme si ukázali pomerne rozsiahle funkcie pre prácu s dialógovým oknom, v ktorom nastavujeme parametre pre vkladanie nezalomiteľných medzier za jednoznakové spojky a predložky. Pravdaže, priame volanie týchto funkcií nie je práve pohodlné a okrem toho sme si do dialógového okna zaradili nový parameter – „Spúšťať automaticky pri otvorení dokumentu“. Táto funkcia je určite mnohými požadovaná, pretože sa nechcú zaoberať tým, aby po otvorení dokumentu ešte manuálne museli spúšťať nejaké makro pre vkladanie (zámenu) medzier.

Preto bude vhodné, ak si všetky funkcie, ktoré sme si uviedli minule uložili do osobitného modulu „Nezalomitelne_predlozky_funkcie“ a dnes si vytvoríme nový modul „Nezalomitelne_predlozky“, do ktorého si zadáme iba jednoduché programy, ktoré budú už príslušné kompletné funkcie v ostatných moduloch volať za nás:

Ako si v tom udržať poriadok a prehľad

```
sub Nastav_priznaky_nezalomitelne_predlozky
 Nezalomitelne_predlozky_funkcie.NastavPriznakyNezalomiteln
ePredlozky
end sub

sub Automaticky_Obsluhuj_Klavesnicu
 Nezalomitelne_predlozky_funkcie.Citaj_nastavenie_predlozky
 if
Nezalomitelne_predlozky_funkcie.Priznaky_nezalomitelne_medz
ery(1)=1 then
  Nezalomitelne_predlozky_funkcie.SpustiObsluhuKlavesnice
 end if
end sub

sub Obsluhuj_Klavesnicu
 Nezalomitelne_predlozky_funkcie.Citaj_nastavenie_predlozky
 Nezalomitelne_predlozky_funkcie.SpustiObsluhuKlavesnice
end sub

sub Neobsluhuj_Klavesnicu
 Nezalomitelne_predlozky_funkcie.UkonciObsluhuKlavesnice
end sub

sub Povol_Klavesnicu
 Nezalomitelne_predlozky_funkcie.Obsluhovat_Klavesnicu=True
end sub

sub Pozastav_Klavesnicu
 Nezalomitelne_predlozky_funkcie.Obsluhovat_Klavesnicu=Fals
e
end sub
```

29.1. Automatické spúšťanie počas otvárania súboru

Pre tých, ktorí potrebujú spúšťať automatické nahrádzanie medzier už počas otvárania súborov je tu pripravená funkcia „Automaticky_Obsluhuj_Klavesnicu“ a funkcia „Neobsluhuj_Klavesnicu“. Práve pre nich je určená aj zaujímavá dvojica funkcií „Povol_Klavesnicu“ a „Pozastav_Klavesnicu“. Tieto funkcie sú určené na to, aby sa dalo občas (z ľubovoľného dôvodu) na určitý čas prerušiť (funkcia „Pozastav_Klavesnicu“) a znovu povoliť (funkcia „Povol_Klavesnicu“) automatické nahrádzanie. Všetci ostatní používatelia si tu napokon nájdu dvojicu funkcií „Obsluhuj_Klavesnicu“ a „Neobsluhuj_Klavesnicu“.

Pravdaže, ešte sme sa nezmienili o tom, ako vlastne zabezpečiť automatické spúšťanie obsluhy klávesnice. Za týmto účelom musíme v dialógovom okne „Nástroje-Prispôsobiť“ na záložke „Udalosti“ priradiť k položke „Vytvoriť dokument“ a k položke „Otvoriť dokument“ volanie makra „Automaticky_Obsluhuj_Klavesnicu“. Zároveň je nutné, aby sme k položke „Zatvoriť dokument“ priradili volanie makra „Neobsluhuj_Klavesnicu“.

29.2. Niekoľko nedostatkov a chýb

Na tomto mieste musíme upozorniť na niektoré chyby, ktoré uvedené makrá ešte stále majú. Používajú sa v nich globálne premenné a preto, ak spustíte nahrádzanie vo viacerých dokumentoch, bude to viesť ku chybe. Pri manuálnom spustení nahrádzania je ďalej potrebné toto pred uzatvorením dokumentu ukončiť, pretože to vedie ku chybe, že chceme uzatvoriť dokument, v ktorom máme spustené nejaké makro. Podobne, ak si otvoríte niekoľko dokumentov a v jednom povolíte nahrádzanie, je pred uzatvorením ktoréhokoľvek z nich v tomto dokumente nahrádzanie ukončiť a až potom prípadne znovu povoliť.

Viem, že tieto chyby môžu na prvý pohľad vyzerať hrôzostrašne. Žiaľ, zatiaľ[3] sa mi nepodarilo prísť na to, ako ich odstrániť. Napriek tomu však verím, že uvedené makrá mnohým pomôžu aj v takom tvare, v akom sú dané k dispozícii, pretože nie každý pracuje naraz s viacerými dokumentami.

30. Niekoľko slov na záver...

Týmto dielom sme ukončili základnú časť práce s dialógmi a tým vlastne aj zo základmi programovania makier v OpenOffice.org. Pred skoro trištvrte rokom sme začali tento seriál s trochu neskromným cieľom – naučiť sa programovať makrá. Ak si pamätáte, začali sme „nahrávaním“ makra pre automatické vkladanie textu a dnes končíme pomerne rozsiahlym komplexom programov, v ktorom nájdete aj automatické nahrádzanie medzier už počas ich písania. Viem, že cez internet sa ťažko učí programovať. Ak sa však v tomto „školskom“ roku našiel aspoň jeden spokojný používateľ, ktorý si na to „trúfol“, splnili sme cieľ, ktorý sme si na začiatku položili. Je 29.6.2006 a pozajtra začnú aj mojim deťom školské prázdniny. Oddýchnime si, načerpajme sily a začiatkom septembra sa znovu teším na stretnutie so všetkými „programovaniachtivými“ používateľmi OpenOffice.org.

3 V čase písania tejto knihy je už problém úspešne vyriešený

Dialógy pod drobnohľadom

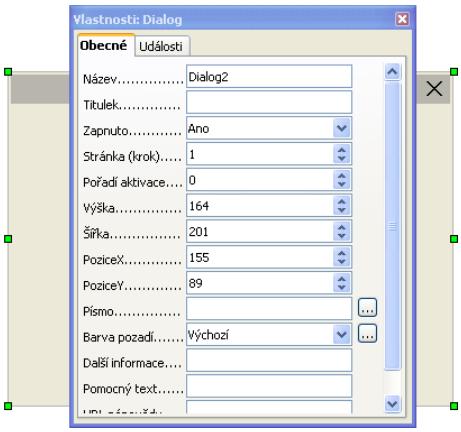
Predchádzajúcu – štvrtú časť seriálu o programovaní makier v OpenOffice.org sme venovali základom práce s dialógmi. Dialógy v OpenOffice.org však môžu byť omnoho väčšie a komplikovanejšie a preto sa nimi budeme naďalej podrobnejšie zaoberať.

V doterajších dieloch sme si ukázali jednoduché dialógy, kde sme zatiaľ použili tri možnosti zadávania vstupných údajov – zaškrtávacie políčka, prepínače a tlačidlá. Toto nám však mnohokrát nevystačuje a navyše, niekedy jedno dialógové okno svojimi rozmermi nemusí stačiť na zadanie všetkých parametrov, ktoré postačujeme a preto musíme naprogramovať celú postupnosť okien s tým, že sa budeme môcť medzi nimi presúvať.

Týmto všetkým možnostiam sa budeme venovať v nasledujúcej piatej časti seriálu o programovaní makier v OpenOffice.org. Pravdaže, nadviažeme na tie časti, ktoré sme si už predstavili s týmto sa nebudeme už venovať, ale predpokladáme, že cez prázdniny mal každý dosť času si nejaké jednoduché dialógy odskúšať.

31. Viacstránkové dialógy

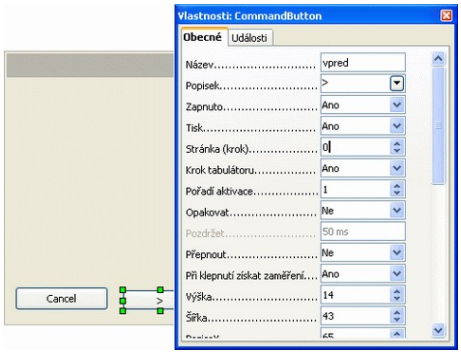
Ako prvé, čomu sa budeme venovať, budú dialógy, ktoré budú zaberať viacero listov – strán. Za týmto účelom je v zozname vlastností jednotlivých prvkov, ktoré vkladáme do vlastného dialógu položka „Stránka (krok)...“. Jej štandardná hodnota je nula, čo znamená, že práve zadávaný prvok sa bude nachádzať na úplne všetkých listoch navrhovaného dialógu – a naopak, na liste č. nula sa budú zobrazovať úplne všetky prvky bez ohľadu na to, na ktorej strane sme ich definovali.



Obrázok 31.1: Zobrazenie parametrov dialógu

31.1. Definícia dialógu

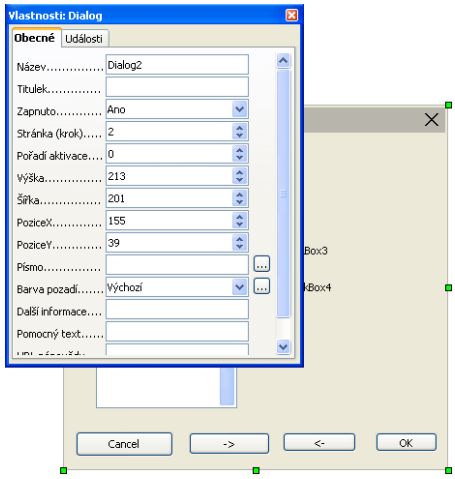
Poďme však po poriadku a pozrime sa, ako máme vlastne začať. Hneď pri začiatku definície dialógu je vhodné označiť prvé pracovné okno, do ktorého ešte len ideme vkladať prvky ako stránku č. 1. Teraz je vhodné vložiť tie prvky, ktoré budeme mať určite na všetkých stránkach – obvykle to budú tlačidlá pre prechod medzi nimi a prípadne tlačidlá „OK“ a „Cancel“. Pri vkladaní týchto tlačidiel nesmieme zabudnúť, že sa nám bude štandardne ponúkať stránka č. 1 a preto túto musíme opraviť na nulu.



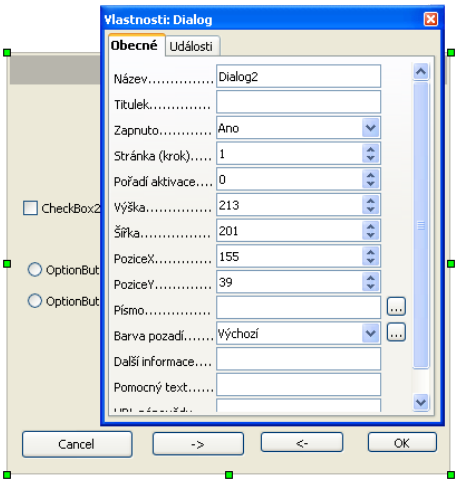
Obrázok 31.2: Zmena stránky na „0“

Vkladanie ďalších prvkov je jednoduché. Podobne, ako sme si to už ukázali postupne vkladáme do pracovného rozhrania všetky potrebné prvky. Pokiaľ chceme vložiť novú stránku, klepneme na hlavné okno navrhovaného dialógu a zmeníme číslo strany. Tým sa nám zobrazí príslušná strana, kde môžeme pokračovať v definícii.

Ku predchádzajúcim stranám sa dostaneme podobne – stačí, ak po označení celého dialógového okna napíšeme príslušné číslo strany. Pozor, ak toto urobíme na niektorom prvku, tento sa automaticky presunie na zadanú stranu a z práve definovanej „zmizne“. Na tomto mieste ešte musíme upozorniť, že OpenOffice.org rozoznáva aj záporné čísla stránok.



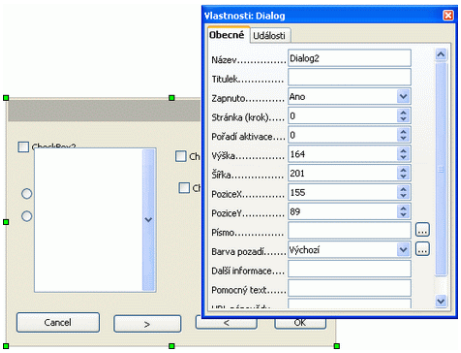
Obrázok 31.3: Definícia prvkov na stránke „2“



Obrázok 31.4: Definícia prvkov na stránke „1“

Pretože sa pri takejto definícii jedná o jeden a ten istý dialóg, ktorý je akurát rozdelený do viacerých listov, každá zmena jeho celkovej veľkosti sa automaticky prejaví na každej strane a preto nie je možné, aby sme navrhli dialóg, ktorý bude mať rôzne veľkosti strán. Na toto musíme pamätať najmä vtedy, ak sa nám nejaká stránka zdá príliš prázdna a radi by sme ju zmenšili.

Pravdaže, teraz nás zaujíma, ako sa medzi jednotlivými stránkami dokážeme presúvať. O tom si však budeme hovoriť až nabudúce, pričom, ako inak, sa znovu vrátime k problematike nezalomiteľných medzier za jednoznakové predložky a spojky – veď prečo by sme nemali zmeniť práve tento dialóg na viacstránkový.



Obrázok 31.5: Zobrazenie všetkých prvkov naraz je možné pomocou stránky „0“

31.2. Rozdelenie dialógu na viac stránok

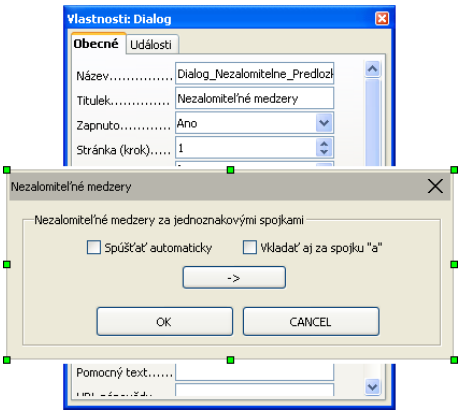
V dnešnom dieli seriálu o programovaní makier v OpenOffice.org si ukážeme praktickú ukážku dvojstránkového dialógu pre nastavovanie parametrov pri vkladaní nezalomiteľných medzier za jednoznakové predložky a spojky.

Ako sme si už naznačili v minulom dieli, dnes si ukážeme, ako sa prakticky programujú podprogramy pre prepínanie medzi jednotlivými oknami viacstránkových dialógov. Aby sme nemuseli vymýšľať nový príklad, rozdelíme si dialóg, ktorý sme si vytvorili pre nastavovanie parametrov pri vkladaní nezalomiteľných medzier za jednoznakové predložky a spojky.

Ako si určite dobre pamätáte, tento dialóg sa skladal z dvoch zaškrtávacích okienok (či sa má vkladanie nezalomiteľných medzier spúšťať automaticky pri spustení OpenOffice.org a či sa má nezalomiteľná medzera vkladať aj za spojku „a“) a dvoch

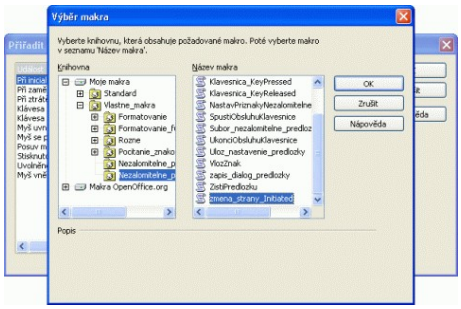
prepínacích políčok, pomocou ktorých sme určovali, aký znak sa má za spojky vkladať („Non Break Space“ alebo „Word Joiner“).

Teraz si ukážeme postup, ako tento dialóg rozdelíme do dvoch stránok. Ako prvé, čo urobíme, bude to, že si v návrhovej časti dialógov označíme celý dialóg a zmeníme jeho číslo stránky (krok) na 1. Teraz vo vnútri dialógu označíme postupne zaškrtávacie okienka a tak isto zmeníme túto položku u nich na číslo stránky na hodnotu jedna.



Obrázok 31.6: Presunutie prvkov na prvú stránku

V druhom kroku znovu označíme celý dialóg a zmeníme číslo stránky na 2. Teraz vo vnútri dialógu označíme postupne prepínacie políčka a zmeníme aj ich čísla stránok na hodnotu dva. Keď teraz budeme prepínať medzi jednotlivými stránkami uvidíme, že nám tam zostali iba príslušné políčka a tlačidlá „OK“ a „Cancel“. Aby sme však mohli medzi nimi prepínať aj po spustení dialógu, musíme si pridať aspoň jedno tlačidlo, pomocou ktorého to budeme realizovať. Nazveme ho „zmena_strany“, musíme mu priradiť číslo strany 0 a ako zobrazovaný text zadáme šípku „->“. Nakoniec si ešte upravíme vzhľad jednotlivých stránok tak, aby nám viac vyhovoval – napr. tým, že namiesto toho, aby sme mali políčka pod sebou, umiestnime ich vedľa seba.



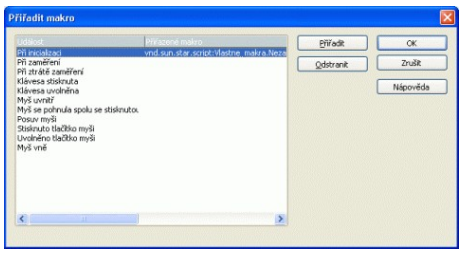
Obrázok 31.7: Výber makra pre tlačidlo

V tomto kroku sme dospeli k najdôležitejšej časti – tlačidlu „zmena_strany“ musíme priradiť makro, pomocou ktorého vlastne budeme jednotlivé stránky prepínať. Pretože máme iba dve stránky, bude nám stačiť pomerne jednoduchá procedúra, kde stačí tieto stránky vymieňať. Zároveň však bude vhodné, ak budeme meniť aj zobrazovanú šípku (text) v tlačidle „zmena_strany“ z „->“ na „<-“ a naopak. Príslušná procedúra môže potom vyzerať napr. takto:

```
rem Prepínanie medzi dvoma stranami dialógu
sub zmena_strany_Initiated
 if dlg.model.step=1 then
  ' Ak sme na strane 1, prepneme sa na stranu 2
  dlg.model.step=2 ' týmto priradením sa automaticky
zobrazila 2 strana dialógu
  dlg.model.zmena_strany.enabled=true ' povolíme platnosť
tlačidla „zmena_strany“
  dlg.model.zmena_strany.label="<-" ' a zmeníme jeho popis
 else
  ' Ak sme na strane 2, prepneme sa na stranu 1
  dlg.model.step=1
  dlg.model.zmena_strany.enabled=true
  dlg.model.zmena_strany.label="->"
```

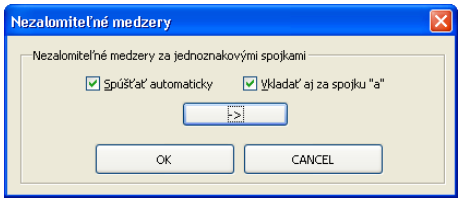
```
 end if
end sub
```

31.3. Priradenie makier k tlačidlám



Obrázok 31.8: Vo vlastnostiach tlačidla vidíme priradené makro

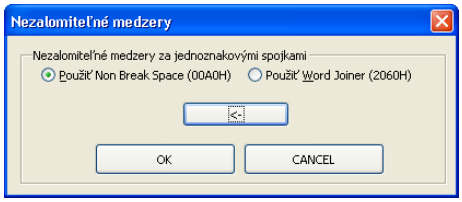
Teraz túto procedúru priradíme ku tlačidlu „zmena_strany“. V návrhu dialógu sa v zobrazených vlastnostiach tohto tlačidla prepneme do druhej záložky „Udalosti“, kde stlačením prvého tlačidla „...“ vyberieme položku „Pri inicializácii“. Tam následne pomocou tlačidla „Priradiť“ vyhľadáme a priradíme túto procedúru k označenému tlačidlu.



Obrázok 31.9: Prvky dialógu na prvej stránke

Teraz nám ešte zostáva trochu upraviť ostatné funkcie, ktoré sme používali pri práci s týmto dialógom. Ako prvé je nutné, aby sme premennú dlg definovali ako globálnu, pretože inak ju nebude pred chvíľou definovaná procedúra poznať a pri priraďovaní makier k tlačidlám nemáme možnosť definovať parametre funkcie.

Okrem toho je potrebné, aby sme pri inicializácii dialógu nastavili aj stranu dialógu, ktorá sa má zobraziť ako prvá a pretože nevieme, aký sme mali posledne zobrazovaný text tlačidla „zmena_strany“, tak aj túto položku. Ostatné časti sa nemusia meniť, pretože umiestnením nejakej položky na inú stranu dialógu sa nemenia jej ostatné vlastnosti. Vlastné makrá potom môžu vyzerať nasledovne (uvádzame iba zmenené procedúry):



Obrázok 31.10: Prvky dialógu na druhej stránke

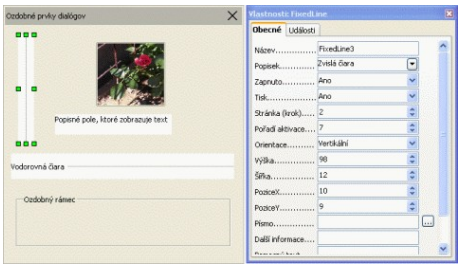
```
global dlg as object

sub init_dialog_predlozky(oDialog as object)
 Citaj_nastavenie_predlozky
 oDialog.model.Automaticky.State=Priznaky_nezalomitelne_medz
ery(1)
 oDialog.model.Za_spojku_a.State=Priznaky_nezalomitelne_med
zery(2)
 oDialog.model.TabIndex=3
 oDialog.model.TabIndex=4
 select case Priznaky_nezalomitelne_medzery(3)
  case 2
   oDialog.model.Volba1.State=0
   oDialog.model.Volba2.State=1
  case else
   oDialog.model.Volba1.State=1
   oDialog.model.Volba2.State=0
 end select
 oDialog.model.step=1 ' Ako prvú zobrazíme stranu 1
```

```
 oDialog.model.zmena_strany.enabled=true ' Povolíme
prepínanie na stranu 2
 oDialog.model.zmena_strany.label="->" ' A zobrazíme text –
šípku – naznačujúcu smer dopredu na ďalšiu stranu
end sub

sub NastavPriznakyNezalomitelnePredlozky

 DialogLibraries.LoadLibrary("Vlastne_makra")

 dlg=CreateUnoDialog(DialogLibraries.Vlastne_makra.Dialog_N
ezalomitelne_Predlozky)
 init_dialog_predlozky(dlg)

 if dlg.Execute()=1 then
  zapis_dialog_predlozky(dlg)
 end if
 dlg.dispose()
end sub
```

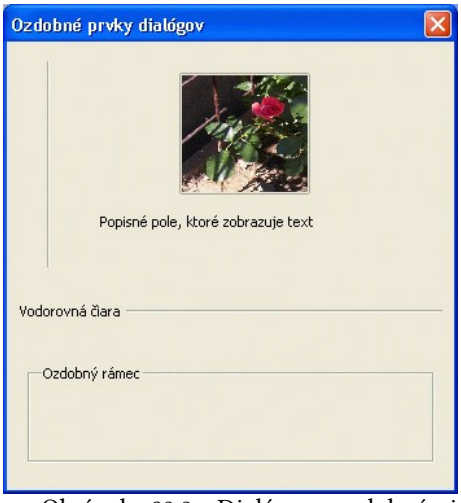
32. Ďalšie možnosti dialógov

Dnes sa zoznámime s ďalšími – tentoraz ozdobnými – prvkami, ktoré môžeme vkladať do našich dialógov, veď doteraz sme sa zoznámili iba so štyrmi druhmi z vyše dvadsiatich.

V doterajších dieloch o dialógoch sme si ukázali, ako sa pracuje s tlačidlami, zaškrtávacími okienkami, prepínacími políčkami a rámcami, pomocou ktorých môžeme rozdeľovať okno dialógu na rôzne prehľadné skupiny. Pretože už vieme pracovať aj s viacstránkovými dialógmi (hoci sme si ukázali iba prácu s dvojstránkovým dialógom, určite si dokážete predstaviť prechod aj medzi viacerými stránkami – to ponechávam prípadným záujemcom ako domácu úlohu), nezostáva nám nič iné, ako si postupne ukázať ostatné prvky, ktoré v dialógoch môžeme používať.



Obrázok 32.1: Vkladanie ozdobných prvkov



Obrázok 32.2: Dialóg s ozdobnými prvkami

32.1. Ozdobné prvky v dialógoch

Ako prvé si ukážeme „ozdobné“ prvky, t.j. také, ktoré sa svojim charakterom podobajú už predstaveným rámcom. Obvykle ich nepoužívame na žiadne operácie (aj keď i to sa dá priradením makra k nejakej akcii, napr. pri posune myši dovnútra vloženého obrázku), ale pomocou nich skrášľujeme a sprehľadňujeme naše dialógy, najmä

ak sú rozsiahlejšieho charakteru.

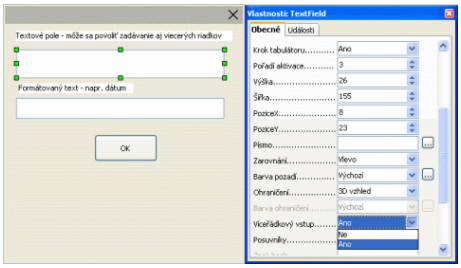
Pretože sme už spomínali a používali rámce, teraz si ako prvé pridáme oddeľujúce čiary. Môžu byť buď vodorovné alebo zvislé s tým, že ku každej z nich môžeme pridať aj textový popis. V prípade, že čiaru zadefinujeme ako vodorovnú, tento popis sa zobrazí na jej začiatku. Ako ďalší „ozdobný“ prvok si môžeme pridať textové popisné pole, t.j. ľubovoľný jednoriadkový text, ktorým napr. označíme či popíšeme niektorú časť dialógového okna. Je to podobný prvok ako rámec s tým rozdielom, že sa nezobrazujú žiadne čiary.

Posledný prvok, pomocou ktorého môžeme ozdobiť náš dialóg je obrázok. Na tomto mieste musíme upozorniť, že obrázok sa do vlastného dialógu nevkladá (pri jeho vkladaní nemáme možnosť zmeniť parameter „odkaz“) a preto pri jeho prípadnom premiestnení sa zobrazí iba prázdny rámik. Pokiaľ si vytvárame dialóg iba sami pre seba, nie je to až tak na závadu, ale v prípade, že chceme náš dialóg uložiť napr. do súboru, resp. nainštalovať ho aj iným používateľom, môže nám to spôsobiť určité komplikácie.

33. Vstupné prvky v dialógoch

V dnešnej, už štvrtej časti programovania makier v OpenOffice.org, ktorú zameriavame na prácu s vlastnými dialógovými oknami sa zameriame na jednoduché formátované a neformátované textové vstupy.

33.1. Textové vstupy

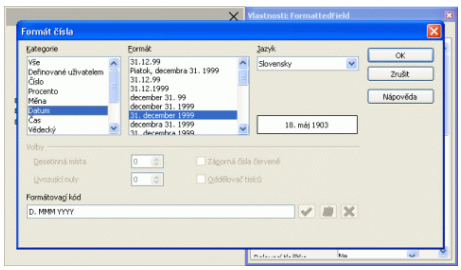


Obrázok 33.1: Textové vstupné pole

Z hľadiska jednoduchých textových vstupov máme k dispozícii dva rôzne dialógové prvky. Jedná sa o jednoduché neformátované textové pole a formátované textové pole. Teraz si v jednoduchosti popíšeme ich vlastnosti a možnosti.

Neformátované textové pole je určené na interaktívne zadanie ľubovoľného textu s tým, že môžeme preddefinovať jeho vstupnú hodnotu. Predpokladajme, že máme definovaný dialóg „dlg“, do ktorého vložíme textové pole „Textove_pole“. Vlastný text potom nájdeme v premennej:

```
dlg.model.Textove_pole.text
```



Obrázok 33.2: Formátované textové pole

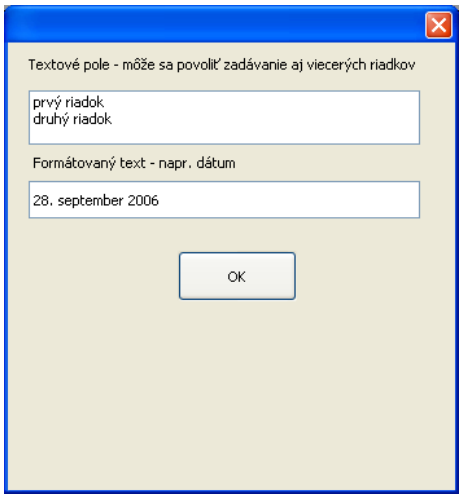
Pokiaľ chceme, môžeme pre textové pole v jeho vlastnostiach povoliť viacriadkový vstup. Jednotlivé riadky sú potom oddelené znakom „Carriage return“ (hexadecimálny kód 0D), takže inicializácia dvoch riadkov vstupnej hodnoty textu vyzerá napríklad takto:

```
dlg.model.Textove_pole.text="prvý riadok"+chr(13)+"druhý riadok"
```

Formátované textové pole sa veľmi podobá bunkám v tabuľkovom procesore Calc. Je to tak preto, že môžeme zadať jeho formát, pričom sa používajú presne tie isté formátovacie prvky, ako pri bunkách v tomto tabuľkovom procesore. Pretože vlastné zadávané hodnoty môžu mať rôzne vlastnosti (text, číslo, dátum, logická hodnota a pod.)

a preto sa inicializujú a získavajú ako text a je iba na nás, ako z nich získame prípadnú „netextovú“ hodnotu.

Pravdaže, OpenOffice.org nás v tomto smere nesklame a nech do takto zadefinovaného políčka zadáme čokoľvek, prevedie to do príslušného formátu a naozaj vám odporúčam zadať napr. do políčka zadefinovaného ako dátum napríklad svoje krstné meno – po kliknutí na iné políčko program automaticky túto hodnotu zmení napr. na minimálnu hodnotu, ktorú definujeme vo vlastnostiach tohto typu poľa.



Obrázok 33.3: Neformátované textové pole môže mať aj viacero riadkov

Prístup k formátovanému textovému poľu je presne ten istý, ako pri neformátovanom, t.j. napr. ak sme do dialógu „dlg“ vložili formátované textové pole „Formatovacie_pole“, inicializácia môže vyzerať takto:

```
dlg.model.Formatovacie_pole.text="28.september 2006"
```

Pravdaže, ako sme už spomínali, z premennej dlg.model.Textove_pole.text získame zadaný vstup po zavretí nášho dialógového okna vo formáte textu a v prípade potreby si ho musíme sami previesť na hodnotu iného typu.

33.2. Číselné vstupy

V dnešnom pokračovaní seriálu o programovaní makier v OpenOffice.org sa zameriame na číselné vstupy, ktoré môžeme použiť v našich dialógoch.

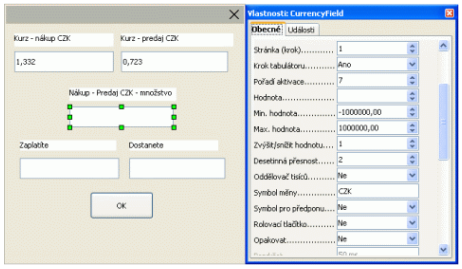
V minulom dieli sme si predstavili dva jednoduché textové vstupy, ktoré sme rozdelili na neformátované a formátované. Podobná situácia je aj v oblasti číselných vstupov, kde máme tak isto dve základné možnosti – obyčajný číselný vstup a číselný vstup pre zadanie korunovej (menovej) hodnoty.

Neformátovaný číselný vstup je určený na interaktívne zadanie ľubovoľného čísla. Podobne, ako pri ostatných vstupoch, aj tu môžeme preddefinovať jeho hodnotu. Špecialitou číselných vstupov je možnosť definovania minimálnej a maximálnej hodnoty spolu s počtom desatinných miest (t.j. desatinnou presnosťou). Pri vlastnom vstupe síce môžeme zadať aj viac desatinných miest, číslo však bude automaticky zaokrúhlené.

Ak budeme predpokladať, že máme definovaný dialóg „dlg“, do ktorého vložíme číselný vstup „Ciselny_vstup“, vlastnú hodnotu potom nájdeme v premennej:

```
dlg.model.Ciselny_vstup.value
```

a inicializáciu prevedieme napríklad takto (pozor, desatinné miesta sa oddeľujú bodkou – tu neplatia gramatické, ale programovacie pravidlá):



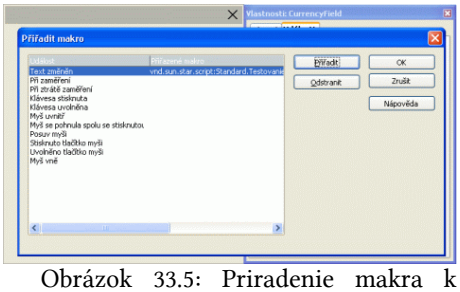
Obrázok 33.4: Nastavenie parametrov pre číselný vstup

```
dlg.model.Ciselny_vstup.value=123.57
```

Číselný vstup pre zadanie meny je určený na interaktívne zadanie čísla s tým, že vo vstupnom okienku sa automaticky vypíše aj názov menovej jednotky (Sk a pod.). Tento názov môžeme ľubovoľne meniť a podobne, ako pri neformátovanom číselnom vstupe môžeme zadať počet desatinných miest, čím si môžeme vytvoriť napr. kurzový lístok aj s prípadným prepočtom.

Ak budeme predpokladať, že máme definovaný dialóg „dlg“, do ktorého vložíme číselný menový vstup „Menovy_vstup“, vlastnú hodnotu potom nájdeme v premennej:

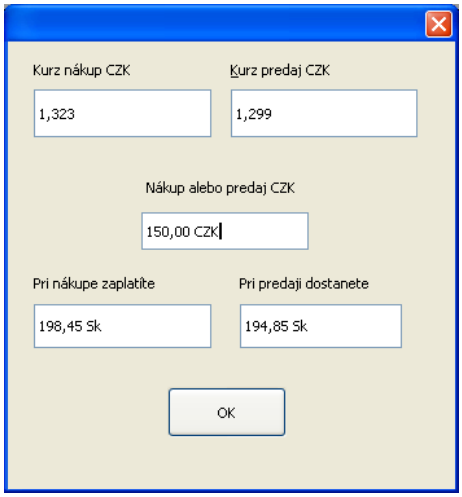
```
dlg.model.Menovy_vstup.value
```



Obrázok 33.5: Priradenie makra k vstupnému číselnému políčku

A aby sme nezostali iba v teoretickej rovine, urobme si jednoduchý príklad na práve spomínanú možnosť kurzového lístka spojeného aj s prepočtom nákupu alebo predaja povedzme medzi českou a slovenskou korunou.

Navrhovaný dialóg môže mať jednoduchý tvar – dve obyčajné okienka pre zadanie kurzov (nákup a predaj) a tri menové okienka – v prvom zadáme, koľko českých korún chceme nakúpiť alebo predať a vo zvyšných dvoch sa nám zobrazia príslušné údaje v slovenských korunách. Okrem toho nám do dialógu stačí pridať tlačidlo „OK“ pre jeho ukončenie a niekoľko popisných textov, aby sme sa v ňom jednoducho dokázali orientovať.



Obrázok 33.6: Prepočet kurzu v praxi

Okrem toho musíme k vstupnému menovému políčku priradiť funkciu, pomocou ktorej budeme pri zadávaní čísla prepočítavať príslušné výstupné hodnoty. Toto dosiahneme tým, že v časti „Udalosti“ priradíme príslušné makro (nazvime si ho „Prepocet_meny“) k udalosti „Zmenený text“. O priraďovaní funkcií k jednotlivým udalostiam sme sa už venovali v časti o viacokienkových dialógoch a preto to teraz spomíname už iba okrajovo.

Vlastné makrá môžu potom vyzerať nasledovne:

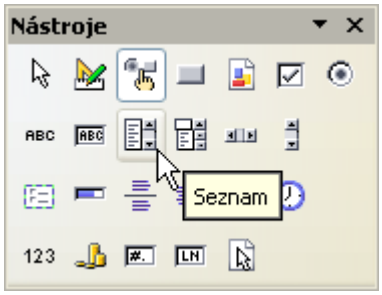
```
sub Prepocet_meny
 dlg.model.Menove_pole1v.value=dlg.model.Ciselne_pole1.value
*dlg.model.Menove_pole1.value
 dlg.model.Menove_pole2v.value=dlg.model.Ciselne_pole2.valu
e*dlg.model.Menove_pole1.value
end sub

sub Dialog_kurzovy_listok
 DialogLibraries.LoadLibrary("Standard")
 dlg=CreateUnoDialog(DialogLibraries.Standard.Dialog2)
```

```
 REM Kurzy sa dajú zmeniť priamo aj v dialógu
 dlg.model.Ciselne_pole1.value=1.323
 dlg.model.Ciselne_pole2.value=1.299

 dlg.model.Menove_pole1.value=0
 dlg.model.Menove_pole1v.value=0
 dlg.model.Menove_pole2v.value=0

 dlg.Execute()
 dlg.dispose()

end sub
```

33.3. Textový zoznam záznamov

Pri praktickej práci niekedy požadujeme, aby sme si mohli vyberať hodnoty z určitého preddefinovaného zoznamu. A práve na takéto typy makier sa zameriame v dnešnom pokračovaní nášho seriálu.

Určite sa už mnohí stretli s problematikou, keď si potrebovali vybrať určitú množinu údajov (napr. kalendárne mesiace), podľa ktorých chceli následne vykonať určité operácie. Preto si v dnešnom dieli ukážeme, ako si dokážeme vytvoriť jednoduchý dialóg so základnými funkciami pre presúvanie údajov z preddefinovaného zoznamu do druhého. Pravdaže, následné spracovanie vybraných údajov už záleží iba od od konkrétnych požiadavok toho-ktorého používateľa a nebudeme sa tým teraz zaoberať.

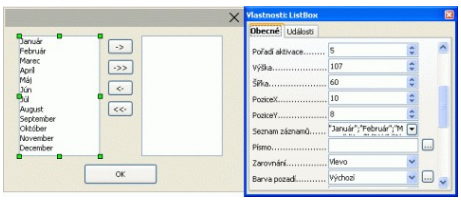


Obrázok 33.7: Výber zoznamu záznamov

Ako prvé, čo urobíme, je pochopiteľne vlastný návrh príslušného dialógu (povedzme si ho nazveme „Dialog2“). Ten sa bude skladať z dvoch zoznamov, pričom do prvého z nich si zároveň zadefinujeme vstupné hodnoty. Tieto sa zadávajú v položke „Zoznam záznamov“, pričom musíme upozorniť na to, že pri ich zadávaní sa musí na ich oddelenie používať kombinácia kláves „Shift“+“Enter“. Ukončenie zadávania následne vykonáme zadaním obyčajnej klávesy „Enter“.

Pravdaže, nemusíme používať iba tento spôsob inicializácie, a hodnoty môžeme zadať podobne, ako sme to robili pri číselných alebo textových vstupoch v príslušnom makre pred spustením vlastného dialógu.

Okrem definície záznamov si musíme do dialógu zadefinovať aj tlačidlá pre presun údajov medzi jednotlivými okienkami. Aby sme ukázali všetky možnosti presunu údajov, zadefinujeme si štyri tlačidlá, ktoré budú postupne reprezentovať možnosť presunu jedného vybraného prvku alebo všetkých prvkov zo zdrojového zoznamu do výberu a prípadné spätné vrátenie jedného prvku alebo všetkých prvkov nazad do pôvodného zdrojového


Obrázok 33.8: Definícia zoznamu záznamov v dialógu

zoznamu. Spätné vrátenie je určené vlastne na opravy a preto nebudeme v dialógu definovať klávesu „Cancel“.

Pri jednotlivých tlačidlách musíme, pochopiteľne, definovať aj inicializačné procedúry, ktoré sa vykonajú po ich stlačení.

```
sub PresunVybrany_Initiated
 REM Procedúra pre presun vybraného prvku zo zdrojového
zoznamu
 dim vstupne_pole, vystupne_pole as object

 ' Priradíme si riadenie zoznamov k dvom pomocným premenným
 vstupne_pole=dlg.getControl("Zoznam_1") ' Zdrojový zoznam
 vystupne_pole=dlg.getControl("Zoznam_2") ' Vytváraný
zoznam

 if vstupne_pole.SelectedItem>0 then
  ' Ak sme vybrali nejaký prvok v zdrojovom zozname, tak ho
pridáme na začiatok vytváraného zoznamu
  vystupne_pole.AddItem(vstupne_pole.SelectedItem, 0)
  ' Vybraný prvok následne vymažeme zo zdrojového zoznamu
  vstupne_pole.removeItems(vstupne_pole.SelectedItemPos, 1)
 end if
end sub

sub PresunVsetko_Initiated
 REM Procedúra pre presun všetkých prvkov zo zdrojového
zoznamu
 dim i as integer
 dim vstupne_pole as object

 vstupne_pole=dlg.getControl("Zoznam_1")

 ' Cyklus pre všetky prvky pôvodného zoznamu
 for i=lbound(dlg.model.Zoznam_1.StringItemList()) to
ubound(dlg.model.Zoznam_1.StringItemList())
  ' Vyberieme vždy prvý prvok (má index 0), pretože
následným vymazaním sa zoznam posúva
  vstupne_pole.SelectItem(dlg.model.Zoznam_1.StringItemList(
0), 1)
  ' Prevedieme vlastný presun pomocou už naprogramovanej
procedúry pre presun jedného vybraného prvku
  PresunVybrany_Initiated
 next i
end sub

sub VratVybrany_Initiated
 REM Procedúra pre vrátenie vybraného prvku do zdrojového
zoznamu
 dim vstupne_pole, vystupne_pole as object
```

```
 vstupne_pole=dlg.getControl("Zoznam_1")
 vystupne_pole=dlg.getControl("Zoznam_2")

 if vystupne_pole.SelectedItem>0 then
  vstupne_pole.AddItem(vystupne_pole.SelectedItem, 0)
  vystupne_pole.removeItems(vystupne_pole.SelectedItemPos,
1)
 end if
end sub

sub VratVsetko_Initiated
 REM Procedúra pre vrátenie všetkých prvkov do zdrojového
zoznamu
 dim i as integer
 dim vystupne_pole as object

 vystupne_pole=dlg.getControl("Zoznam_2")

 for i=lbound(dlg.model.Zoznam_2.StringItemList()) to
ubound(dlg.model.Zoznam_2.StringItemList())
  vystupne_pole.SelectItem(dlg.model.Zoznam_2.StringItemLis
t(0), 1)
  VratVybrany_Initiated
 next i
end sub
```

Teraz si trochu vysvetlíme použité metódy, aby sme si v prípade potreby mohli naprogramovať aj makrá s inými vlastnosťami.

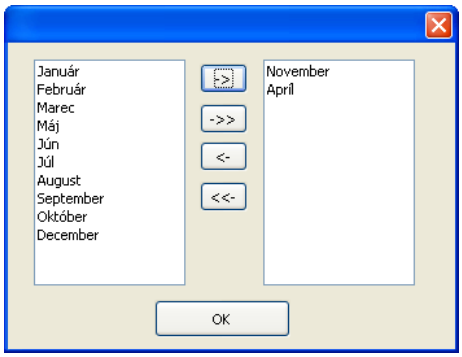
Metóda AddItem(vkladaný prvok, pozícia) je určená na vkladanie prvkov a má dva parametre – prvý je text, ktorý vkladáme do príslušného zoznamu a druhý je pozícia, na ktorú ho vkladáme. Pozor, pri textových poliach každá Indexácia začína od 0. Pomocou tejto metódy by sme mohli inicializovať zoznam mesiacov napr. takto:

```
sub Init_mesiace
 dim vstupne_pole as object

 vstupne_pole=dlg.getControl("Zoznam_1")

 ' Prvky budeme vkladať na začiatok, preto ich musíme zadať
v poradí od posledného ku prvému
 vstupne_pole.AddItem("December", 0)
 vstupne_pole.AddItem("November", 0)
 vstupne_pole.AddItem("Október", 0)
 vstupne_pole.AddItem("September", 0)
 vstupne_pole.AddItem("August", 0)
 vstupne_pole.AddItem("Júl", 0)
 vstupne_pole.AddItem("Jún", 0)
 vstupne_pole.AddItem("Máj", 0)
 vstupne_pole.AddItem("Apríl", 0)
```

```
 vstupne_pole.AddItem("Marec", 0)
 vstupne_pole.AddItem("Február", 0)
 vstupne_pole.AddItem("Január", 0)
end sub
```

Metóda RemoveItems(pozícia prvku, počet) je určená na mazanie prvkov a má dva parametre – prvý je pozícia prvého vymazávaného prvku a druhá počet vymazávaných prvkov.

Ako ste si mohli všimnúť, zoznam všetkých prvkov nájdeme v položke StringItemList, t.j. v našom prípade dlg.model.Zoznam_1.StringItemList alebo dlg.model.Zoznam_2.StringItemList.



Obrázok 33.9: Dialóg pre výber mesiacov v praxi

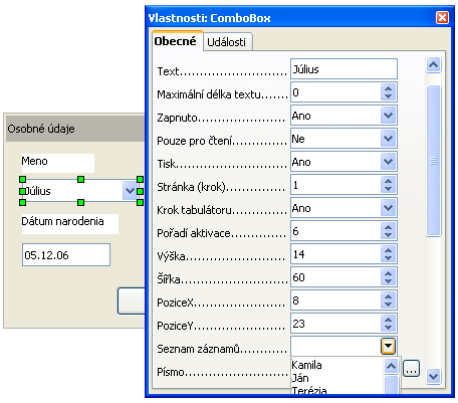
Nakoniec si ešte uvedieme príklad makra pre vlastnú inicializáciu tohto dialógu:

```
sub makro_zoznam
 DialogLibraries.LoadLibrary("Standard")

 dlg=CreateUnoDialog(DialogLibraries.Standard.Dialog2)

 dlg.Execute()
 REM tu si musí každý vložiť svoje príkazy pre spracovanie
vybraných prvkov zoznamu
 dlg.dispose()
end sub
```

33.4. Rozbaľovací zoznam záznamov

V minulom dieli nášho seriálu sme si ukázali, ako dokážeme vyberať prvky z jedného zoznamu a v prípade potreby ich prenášať do druhého a naspäť. Dnes budeme pokračovať v tejto problematike a ukážeme si výber hodnoty z rozbaľovacieho zoznamu.

Možnosti, kde sa dajú používať preddefinované zoznamy určite nemusíme pripomínať, pretože tí, ktorí sa so zoznamami stretávajú už určite vidia ich praktické použitie. Z praktického hľadiska sa nám niekedy namiesto obyčajného zoznamu hodí tzv. kombinovaný zoznam, alebo lepšie povedané kombinovaný rozbaľovací zoznam (známy aj pod názvom „Combo box“). Tento sa líši oproti bežného zoznamu nielen tým, že v zabalenom tvare je zobrazený iba jeden záznam, ale najmä tým, že na výstupe nám vráti vždy práve jednu hodnotu, a to aj vtedy, ak si žiadnu nevyberieme – vtedy vráti základnú hodnotu, ktorú máme definovanú vlastne mimo tento zoznam – preto hovoríme o tzv. kombinovanom zozname.



Obrázok 33.10: Definícia rozbaľovacieho zoznamu

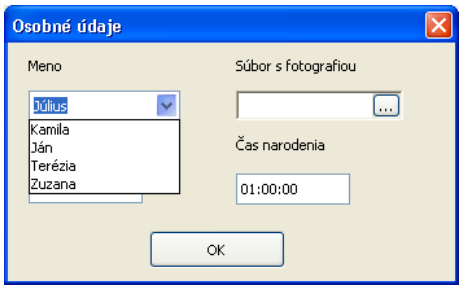
Pozrime sa však na to, ako sa takýto zoznam definuje. Je to veľmi podobné, ako keď sme minule

definovali obyčajný zoznam – jednotlivé záznamy zadávame v jeho vlastnostiach do položky „Zoznam záznamov“ s tým, že ich musíme oddeľovať kombináciou kláves „Shift“+“Enter“, pretože zadávanie sa ukončuje klávesou „Enter“. Okrem toho, pravdaže, môžeme tento záznam definovať aj programovo pomocou metódy „AddItem“, ktorú sme si popisovali už minule, napr.:

```
  dim vstupne_pole as object

  REM Predpokladajme, že máme definovaný rozbaľovací zoznam
s názvom „Zoznam“
  vstupne_pole=dlg.getControl("Zoznam")
  vstupne_pole.AddItem("náš text", 0)
```

Ako sme spomínali, v rozbaľovacom zozname máme aj jednu štandardnú hodnotu, ktorú sa nám vráti v prípade, že nevyberieme žiadnu inú položku zo zoznamu. Túto hodnotu definujeme vo vlastnostiach zoznamu hneď za jeho názvom v položke „Text“. Výstupnú hodnotu získame nakoniec takisto cez hodnotu text:

```
  dlg.model.Zoznam.Text
```



Obrázok 33.11: Rozbaľovací zoznam v praxi

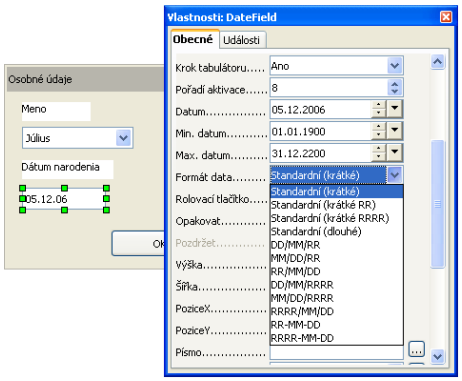
Vlastnú inicializáciu dialógu vrátane následného získania vybranej hodnoty pre ďalšie spracovanie môžeme nakoniec urobiť napr. pomocou takejto jednoduchej funkcie:

```
function Daj_polozku_zoznamu
 DialogLibraries.LoadLibrary("Standard")
 dim co as string

 dlg=CreateUnoDialog(DialogLibraries.Standard.Dialog2)

 dlg.Execute()
 co=dlg.model.Zoznam.Text
 dlg.dispose()
 Daj_polozku_zoznamu=co
end function
```

33.5. Dátum a čas

Doteraz sme si už ukázali veľa možností vstupov v dialógoch – od zaškrtávacích či prepínacích políčok, cez rôzne číselné a textové vstupy až po zoznamy. Nakoniec nám ešte zostali možnosti vstupu dátumu, času alebo výber mena súboru.

Pravdaže, v doterajších možnostiach sme sa už stretli aj možnosťou zadávania dátumov alebo časov pomocou formátovaného textového poľa. To nám však niekedy nemusí vyhovovať, pretože chceme dostať dátum alebo čas priamo v číselnej



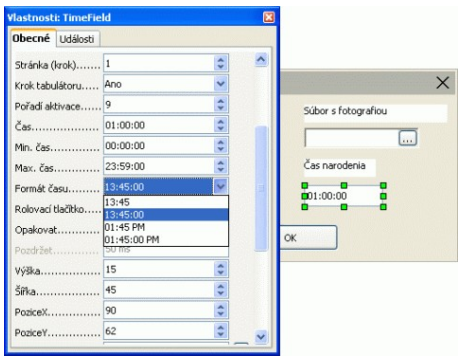
Obrázok 33.12: Definovanie dátumového vstupu

podobe. A na to nám slúžia iné – špeciálne vstupné položky.

Prvá z nich je dátum. Pri jeho definícii nájdeme viacero parametrov, ktoré nás musia zaujímať. Medzi takéto patria ohraničenie rozsahu dátumu spodná horná hranica a formát, v akom chceme zadávať dátum. Okrem toho môžeme preddefinovať jeho štandardnú hodnotu. Výstupnú hodnotu následne získame pomocou metódy „Date“, ktorá vracia číslo typu Long. Ak teda máme napr. definovanú vstupnú položku dátum s názvom „Datum“, tak pomocou štruktúry:

```
dlg.model.Datum.Date
```

získame dátumové číslo, ktoré je vždy v tvare RRRRMMDD, kde štyri cifry RRRR označujú rok, za nimi nasledujú dve cifry MM, ktoré označujú mesiac a nakoniec sú dve cifry DD, ktoré označujú deň.



Obrázok 33.13: Definovanie časového vstupu

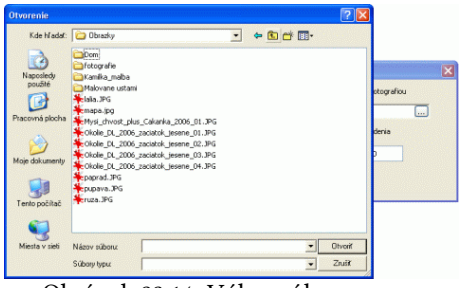
Dátumovému vstupu je veľmi podobný aj časový vstup. Aj tu nájdeme parametre, pomocou ktorých môžeme obmedziť jeho rozsah. Kým pri dátumoch toto obmedzenie nemusí mať až taký význam, pri čase to môžeme veľmi prakticky využiť napr. iba na pracovný čas a pod. Pravdaže, aj tu môžeme zadať preddefinovanú hodnotu a môžeme si vybrať formát, v akom chceme čas zadávať. Výstupnú hodnotu získame pomocou metódy „Time“, ktorá vracia číslo typu Long. Pri predpoklade, že máme definovanú vstupnú položku čas s názvom „Cas“, výstupnú hodnotu získame pomocou štruktúry:

```
dlg.model.Cas.Time
```

Výstupné číslo je vždy vo formáte HHMMSSTT, kde dve cifry HH označujú hodinu (od 00 do 23, t.j. v 24 hodinovom formáte), za nimi dve cifry MM označujú minútu, nasledné dve cifry SS označujú sekundy a posledné dve cifry TT označujú stotiny sekúnd.

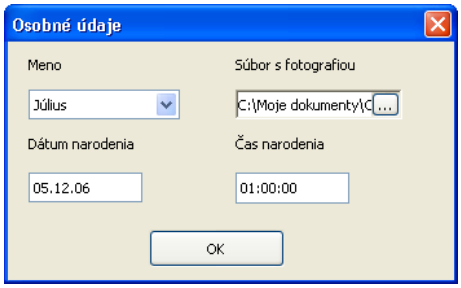
33.6. Výber mena súboru

Posledný typ vstupu, ktorý si preberieme v rámci dialógov je zadanie mena súboru pomocou jeho výberu z už existujúcich súborov. Už v minulých dieloch sme ukazovali, ako si môžeme uložiť nastavené parametre do nejakého súboru na disku. Vtedy sme však meno súboru zadávali priamo v programe. Mnohokrát však potrebujeme vybrať na spracovanie už existujúci súbor na disku. A práve na toto je určený posledný vstup, ktorý si preberáme.



Obrázok 33.14: Výber súboru

Po jeho výbere nemusíme nastavovať okrem názvu (ako ho vôbec chceme zmeniť) vôbec nič, pokiaľ nepožadujeme, aby sa nám vracala nejakú preddefinovaná hodnota. Pokiaľ ju chceme zadať, musíme ju zadefinovať, podobne ako pri iných textových vstupoch, do položky „Text“. Vlastný výber súboru prevádzame stlačením tlačidla



Obrázok 33.15: Príklad vstupu dátumu, času a mena súboru

s tromi bodkami, ktoré sa nachádza v pravej časti tohto vstupného prvku. Toto tlačidlo vyvolá systémovú funkciu pre výber názvu ľubovoľného súboru. Výstupnú hodnotu získame pomocou metódy „Text“. Pokiaľ budeme predpokladať, že máme definovanú vstupnú súborovú položku „Subor“, tak výstupnú hodnotu získame pomocou nasledujúcej štruktúry:

```
dlg.model.Subor.Text
```

Príslušný dialóg môžeme následne inicializovať a výstupné hodnoty získať napríklad takouto procedúrou:

```
dim meno, fotografia as string
dim datum_narodenia, cas_narodenia as long

sub Informacie_o_osobe
 DialogLibraries.LoadLibrary("Standard")

 dlg=CreateUnoDialog(DialogLibraries.Standard.Dialog2)
 dlg.Execute()
 meno=dlg.model.Zoznam.Text
 fotografia=dlg.model.Subor.Text
 datum_narodenia=dlg.model.Datum.Date
 cas_narodenia=dlg.model.Cas.Time
 dlg.dispose()
end sub
```

34. Dialóg pre formátovanie dokumentu

Po dlhšom čase sa znovu vrátime k formátovaniu dokumentu s tým, že si všetky podmienky pre jeho formátovanie nastavíme v jednom dialógovom okne. Pravdaže, podľa toho následne upravíme aj príslušné funkcie a procedúry.

V minulom dieli nášho seriálu sme ukončili možnosti vstupov v dialógoch. Teraz si na niekoľkých praktických príkladoch ukážeme, ako tieto dialógy využijeme v praxi. Ako prvý príklad si zoberieme formátovanie dokumentov, ktoré sme tu už mali veľmi často, aby sme uzavreli aj túto časť nášho seriálu.

34.1. Čo vlastne chceme formátovať?

Najprv si znovu spomeňme, čo všetko sme vo formátovaní dokázali zatiaľ naprogramovať. V prvom rade to bolo odstraňovanie viacnásobných medzier (a to nielen v texte, ale aj na začiatku či na konci odsekov). Ďalej sme si ukázali odstraňovanie nadbytočných medzier pri interpunkčných znamienkach (a naopak aj ich vkladaniu, ak chýbali) a zátvorkách. Nie malú časť formátovania dokumentu sme venovali nahrádzaniu zalomiteľných medzier za nezalomiteľné medzery za jednoznakovými predložkami a spojkami (so špecifickým problémom okolo spojky „a“), za a pred akademickými titulmi, v číslach a pod.

Venovali sme sa aj problematike automatického nahrádzania zalomiteľných medzier

za nezalomiteľné priamo počas písania jednoznakových predložiek a spojok a vkladaniu tzv. pružnej nezalomiteľnej medzery, čo sa nám podarilo pomocou znaku „Word Joiner“. Nakoniec sme si ukázali jednoduchú formu ukladania nášho nastavenia do konfiguračných súborov (tejto časti sa budeme venovať ešte podrobnejšie).

Ako vidíme, problematika formátovania dokumentov, ktorej sme sa doteraz venovali je naozaj široká a preto si aj záverečné práce rozdelíme do viacerých etáp. Dnes sa budeme venovať návrhu dialógu „Formátovanie dokumentov“. Pravdaže, nebudeme sa teraz venovať vlastnému vkladaniu jednotlivých prvkov, ale si ich v krátkosti iba vymenujeme, aby sme sa nabudúce mohli na ne odvolávať.

34.2. Nezalomiteľné medzery

V rámci nezalomiteľných medzier si tieto rozdelíme na tri skupiny. Prvá sa bude venovať jednoznakovým spojkám a predložkám. Tu si zadefinujeme dve zaškrtávacie políčka „Za_predlozkami“ a „Za_spojkou_a“ a dve dvojice prepínacích políčok. Prvou dvojicou „NBSP1“ a „WJ1“ budeme určovať typ nezalomiteľnej medzery (NBSP – „Non Break Space“, WJ – „Word Joiner“) a druhou dvojicou „On_line“ a „Off_line“ budeme určovať, či sa bude nezalomiteľná medzera vkladať priamo počas písania textu, alebo iba vtedy, keď spustíme celé formátovanie dokumentu.

Ďalej si v rámci skupiny nezalomiteľných medzier zadefinujeme zaškrtávacie políčko „Za_titulmi“, ktorým budeme určovať, či sa majú zameniť zalomiteľné medzery za a pred akademickými titulmi. K tomuto tlačidlu ďalej zadefinujeme dvojicu prepínacích políčok „NBSP2“ a „WJ2“, ktorej význam je taký istý, ako pri spojkách a predložkách, akurát sa týka akademických titulov.

Posledné – tretie zaškrtávacie políčko v rámci skupiny nezalomiteľných medzier bude „Za_cislami“ bude určovať, či budeme vkladať nezalomiteľné medzery za jednociferné čísla, medzi číslice v rámci telefónnych čísliel, medzi čísla a merné jednotky a medzi čísla v dátumoch. K tomu zadefinujeme poslednú dvojicu prepínacích políčok „NBSP3“ a „WJ3“ pre určenie typu nezalomiteľnej predložky. Pravdaže, kto chce, môže si túto časť rozdeliť podrobnejšie – podľa jednotlivých typov (jednociferné čísla, čísla a merné jednotky…), ale to už necháme na každého používateľa podľa toho, ak to potrebuje osobitne riešiť.

34.3. Odstraňovanie nadbytočných medzier a pod.

Druhá veľká skupina, ktorú si nastavíme bude sada zaškrtávacích políčok, ktorými určíme, čo chceme v dokumente opraviť pri jeho formátovaní. Budú to tieto tlačidlá:

„Zrus_medzery“ – pomocou tohto zaškrtávacieho okienka určíme, že sa budú vymazávať viacnásobné medzery vo vnútri textu.

„Zrus_okraje“ – pomocou tohto zaškrtávacieho okienka určíme, že sa budú vymazávať viacnásobné medzery na začiatku a na konci odsekov. Možno sa teraz čudujete, prečo túto možnosť uvádzam ako osobitnú – veď na začiatku odseku by sa medzery nemali používať. Je to síce pravda, ale nie vždy ich vymazanie je vyhovujúce… napríklad ak si skopírujete zdrojový text programu, ktorý máte napísaný prehľadne tak, že príkazy, ktoré sú vnorené do vnútra iného príkazu máte posunuté vpravo, tak máte na začiatku odseku práve medzery – a vtedy ich tam aj potrebujete ponechať (osobne túto

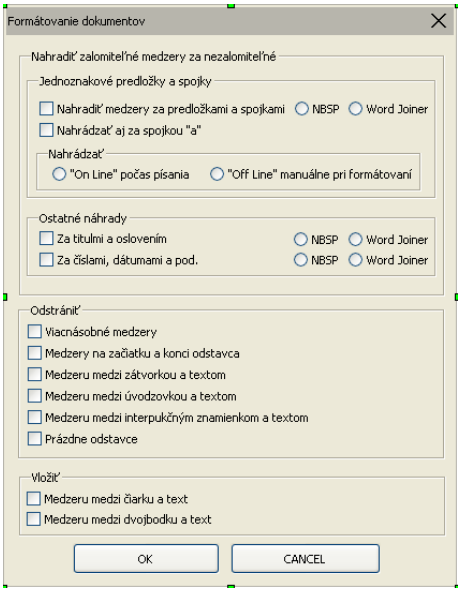
možnosť musím používať aj pri písaní článkov o programovaní makier v OpenOffice.org). Z podobných dôvodov sa tu nachádzajú aj ďalšie nastavenia, ktoré môžu využiť iní používatelia.

„Zrus_zatvorky“ – pomocou tohto zaškrtávacieho okienka určíme, že sa budú vymazávať medzery za ľavými a pred pravými zátvorkami.

„Zrus_uvodzovky“ – pomocou tohto zaškrtávacieho okienka určíme, že sa budú vymazávať medzery za dolnými a pred hornými úvodzovkami.

„Zrus_interpunkcie“ – pomocou tohto zaškrtávacieho okienka určíme, že sa budú vymazávať medzery medzi textom a interpunkčným znamienkom (čiarka, bodka,...).

„Zrus_odseky“ – posledným zaškrtávacím okienkom z tejto skupiny nastavíme odstránenie všetkých prázdnych odsekov. Tejto voľbe sme sa venovali iba v rámci popisu regulárnych výrazov, teraz si ju však aj naprogramujeme.



Obrázok 34.1: Dialóg pre nastavenie parametrov formátovania dokumentu

34.4. Vkladanie medzier

Posledná skupina sa venuje vkladaniu chýbajúcich medzier za interpunkčné znamienka. Na toto miesto si vložíme dve zaškrtávacie tlačidlá:

„Vloz_bodku“ – pomocou tohto zaškrtávacieho okienka určíme, či sa budú vkladať chýbajúce medzery za znak bodky. Aj tento prípad je špecifický a to preto, lebo niekedy nám toto nevyhovuje – konkrétne v prípade, že v texte používame názov „OpenOffice.org“, kde za znakom bodky medzera nie je. Pretože sa jedná o oficiálny názov, nemôžeme ho meniť podľa pravidiel typografie. Pravdaže, dalo by sa to vyriešiť aj tak, že medzery vložíme všade a potom pri tomto texte ich nazad vymažeme – je to však nepohodlné a navyše nesystémové riešenie, pretože takýchto výnimiek môže byť viac. Zatiaľ preto necháme riešenie iba v tomto tvare a prípadné vhodnejšie riešenie si necháme niekedy nabudúce, resp. na domácu úlohu.

„Vloz_interpunkciu“ – pomocou tohto zaškrtávacieho okienka určíme, či sa budú vkladať chýbajúce medzery za ostatné interpunkčné znamienka.

34.5. Tlačidlá

Na koniec nezabudneme vložiť aj dve tlačidlá – tlačidlo pre potvrdenie zadaných údajov „OK“ a tlačidlo pre zrušenie zadaných zmien „CANCEL“.

35. Systémové cesty závislé od používateľa

V dnešnom dieli seriálu si ukážeme, ako môžeme automaticky ukladať rôzne inicializačné nastavenia do súborov v adresároch, ktoré sú závislé od používateľa počítača.

Už v štvrtom dieli tretej časti tohto seriálu o programovaní makier

v OpenOffice.org sme si ukázali, ako si môžeme ukladať nastavenia do súboru. Vtedy sme, pre jednoduchosť, použili iba priamu cestu do konkrétneho adresára:

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

Toto riešenie má, pravdaže, niekoľko nedostatkov. Konkrétne uvedený príklad je hádam ten najhorší, aký sa dá pri programovaní zvoliť (a práve preto je dobré, že sa zvolil, aby sa ukázali jeho všetky nedostatky).

Ako základ je to, že pokiaľ si nainštalujeme novú verziu OpenOffice.org, tak sa zmení aj jeho kmeňový adresár „OpenOffice.org 2.0“ - v súčasnosti je to pre najnovšiu verziu „OpenOffice.org 2.1“. Po odinštalovaní pôvodnej verzie, pravdaže, môžeme pôvodný adresár s príslušnými súbormi ponechať, vedie to však k zníženiu prehľadnosti.

Iná možnosť je zmeniť vlastné makro a presunúť inicializačný definičný súbor na nové miesto. Toto by sme však museli pracne robiť pri každej novej inštalácii, takže to tiež nie je práve vyhovujúce riešenie.

Pravdaže, ako miesto pre ukladanie inicializačných definičných súborov si môžeme zvoliť aj iný adresár, než je práve adresár, kde sa nachádza program OpenOffice.org, napríklad:

```
Subor_nezalomitelne_predlozky="file:///c:/Program
files/OpenOffice.org_init_subory/Nezalomitelne_Predlozky.dat
"
```

Toto riešenie je nezávislé od inštalácie OpenOffice.org. Má však ešte stále jednu nevýhodu – pokiaľ sa pri počítači strieda viacej používateľov, nemôže si každý nastaviť svoje vlastné podmienky, čo môže byť niekedy veľmi nevyhovujúce. Preto je najlepšie také riešenie, kde si každý používateľ uloží svoje nastavenia do osobitného adresára, ktorý mu poskytuje OpenOffice.org v rámci príslušného operačného systému.

Na tento účel máme v rámci používateľského rozhrania jazyka StarOffice Basic k dispozícii objekt „com.sun.star.util.PathSettings“, ktorý nám poskytne všetky potrebné cesty a ktorý môžeme inicializovať napríklad nasledovným spôsobom:

```
dim cesty as object
cesty=CreateUnoService("com.sun.star.util.PathSettings")
```

Objektová premenná „cesty“ obsahuje teraz nasledovné údaje:

Backup – Cesta ku kópiám dokumentov, ktoré boli automaticky zálohované.

Basic – Cesta k makrám a dialógom, ktoré sú napísané v jazyku Basic. Táto hodnota môže obsahovať viacej ciest, ktoré sú oddelené bodkočiarkou.

Favorite – Cesta k obľúbeným položkám.

Gallery – Cesta ku galériovej databáze a multimediálnym súborom. Táto hodnota môže obsahovať viacej ciest, ktoré sú oddelené bodkočiarkou.

Graphic – Táto cesta sa zobrazuje v dialógu pre otvorenie alebo ukladanie grafických objektov.

Help – Cesta k nápovedným súborom OpenOffice.org.

Module – Cesta k jednotlivým modulom OpenOffice.org.

Storage – Cesta k informáciám o pošte, súborom správ, FTP serveri a pod.

Temp – Cesta k dočasným súborom.

Template – Cesta k šablónam, ktoré máme zaradené do OpenOffice.org. Táto hodnota môže obsahovať viacej ciest, ktoré sú oddelené bodkočiarkou.

UserConfig – Cesta k adresáru pre používateľské nastavenia.

Work – Cesta k používateľskej pracovnej zložke. Používa sa v dialógu pri otváraní a zatváraní súborov a môže byť používateľom aj zmenená.

Obrázok 35.1: Zoznam všetkých ciest

Pre zistenie všetkých ciest si môžeme napísať takéto jednoduché makro:

```
sub zoznam_ciest
 dim cesty as object
 dim zoznam, CR as string
 CR=chr$(10)
 cesty=CreateUnoService("com.sun.star.util.PathSettings")
 zoznam="Backup: "+CR+cesty.Backup+CR+CR
 zoznam=zoznam+"Basic: "+CR+cesty.Basic+CR+CR
 zoznam=zoznam+"Favorite: "+CR+cesty.Favorite+CR+CR
 zoznam=zoznam+"Gallery: "+CR+cesty.Gallery+CR+CR
 zoznam=zoznam+"Graphic: "+CR+cesty.Graphic+CR+CR
 zoznam=zoznam+"Help: "+CR+cesty.Help+CR+CR
 zoznam=zoznam+"Module: "+CR+cesty.Module+CR+CR
 zoznam=zoznam+"Storage: "+CR+cesty.Storage+CR+CR
 zoznam=zoznam+"Temp: "+CR+cesty.Temp+CR+CR
 zoznam=zoznam+"Template: "+CR+cesty.Template+CR+CR
 zoznam=zoznam+"UserConfig: "+CR+cesty.UserConfig+CR+CR
 zoznam=zoznam+"Work: "+CR+cesty.Work
 msgbox zoznam
end sub
```

Pokiaľ chcete zmeniť adresár, ktorý nájdeme v adresári „Work“, môžeme si na to naprogramovať takúto jednoduchú procedúru:

```
sub zmen_work (novy_work as string)
 REM Predpokladáme, že cesta nie je zadaná vo všeobecnom
URL formáte
 dim cesty as object
 cesty=CreateUnoService("com.sun.star.util.PathSettings")
 cesty.work=ConvertToURL(novy_work)
end sub
```

Alebo takúto procedúru:

```
sub zmen_work (novy_work as string)
 REM Predpokladáme, že cesta je uložená vo všeobecnom URL
formáte
 dim cesty as object
 cesty=CreateUnoService("com.sun.star.util.PathSettings")
 cesty.work=novy_work
end sub
```

Pre ukladanie našich nastavení sa javí, že bude najvhodnejšie používať adresár, kde sú uložené aj ostatné používateľské nastavenia (UserConfig). Funkciu pre zistenie názvu inicializačného súboru môžeme potom naprogramovať napríklad takto:

```
function inicializacny_subor (nazov_suboru as string) as
string
 REM Funkcia vracia názov inicializačného súboru v
„UserConfig“ adresári aktuálneho používateľa
 dim cesty as object
 cesty=CreateUnoService("com.sun.star.util.PathSettings")
 inicializacny_subor=cesty.UserConfig+"/"+nazov_suboru
end function
```

36. Nastavenie používateľského profilu

V minulom dieli sme si ukázali, ako je možné ukladať súbory do adresárov, ktoré sú závislé od používateľa počítača. Dnes tieto služby využijeme naplno a rozšírime ich ešte o možnosť vytvárania vlastných používateľských profilov.

36.1. Dialóg pre používateľské profily

Na tento účel si zadefinujeme ďalší dialóg, ktorý nazveme povedzme „Dialog_Profily“. Vložíme do neho nasledovné prvky:

kombinované rozbalovacie textové pole „**Zoznam_Profilov**“, v ktorom budeme mať následne zobrazené všetky profily;

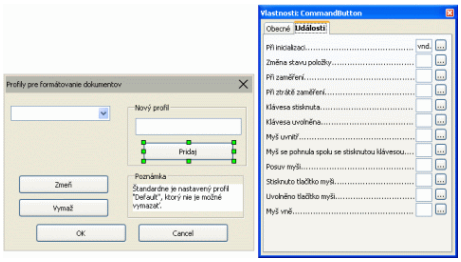
vstupné textové pole „**Novy_Profil**“ pre zadávanie názvov nových profilov;

tlačidlo „**Pridaj**“ pre potvrdenie nového profilu;

tlačidlo „**Zmeň**“ pre editovanie vybraného profilu;

tlačidlo „**Vymaž**“ pre vymazanie vybraného profilu;

tlačidlá „**OK**“ a „**Cancel**“ pre ukončenie činnosti.



Obrázok 36.1: Dialógové okno pre nastavovanie používateľských profilov

Okrem toho si môžeme pridať informačný text, že „Štandardne je nastavený profil "Default", ktorý nie je možné vymazať“. Pokiaľ chceme, aby sa tento text zobrazoval na viacero riadkov, je potrebné na príslušných miestach vložiť kombináciu kláves „Shift“+“Enter“, ktorá to umožňuje.

Teraz sa pozrime na vlastné makrá, ktoré budeme pre prácu potrebovať. Vzhľadom na to, že sú úzko previazané aj s dialógom pre nastavovanie parametrov formátovania

dokumentu (ktorý sme si už uvádzali) a s vlastnými funkciami pre toto formátovanie, je ich značné množstvo a preto ich neuvedieme naraz, ale postupne v niekoľkých dieloch tohto seriálu. Dnes sa budeme venovať iba tým, ktoré priamo súvisia s dialógom „Dialog_Profily“.

36.2. Makrá pre prácu so zoznamom používateľských profilov

Vytvorme si v zozname makier nový modul „Profily_funkcie“, kde vložíme (zatiaľ) nasledovné definície premenných a procedúry (budeme ich priebežne komentovať):

```
dim dlg as object
dim zoznam_profilov(50) as string
dim pocet_profilov as integer
global priznaky_formatovania(16) as integer
global aktualny_profil as string
dim oVstupy as object
```

Objektová premenná „dlg“ je určená na vlastné priradenie dialógu „Dialog_Proflily“. Musí byť definovaná externe preto, lebo sa na ňu odvolávame vo funkciách, ktoré sú priradené k tlačidlám „Pridaj“, „Zmeň“ a „Vymaž“.

V premennej „zoznam_profilov“ budeme mať uložené názvy profilov. Jej rozsah sme si dali 50 profilov, čo je určite viac než dostatočný počet pre jedného používateľa. Pokiaľ nie, stačí jednoduchá zmena tohto parametra. Táto premenná je určená na testovanie názvov profilov pri ich pridávaní, aby sme nemali dva rovnaké profily. Počet profilov máme uložený v nasledujúcej premennej „pocet_profilov“.

S globálnou premennou „priznaky_formatovania“ sa dnes ešte nestretneme, lebo je určená pre dialóg „Dialog_formatovanie“, ktorý sme si už navrhovali, ale jeho makrá budeme popisovať až nabudúce. Táto premenná je preto globálna, lebo sa podľa nej riadi celá množina funkcií pre formátovanie dokumentu, ktoré si tak isto budeme uvádzať až nabudúce.

Do globálnej premennej „aktualny_profil“ uložíme nakoniec vybraný profil. Podobne, ako premenná „priznaky_formatovania“ je určená pre neustálu informáciu o tom, aký profil práve chceme používať pre formátovanie dokumentu.

Pomocná premenná „oVstupy“ je určená pre pridávanie a odoberanie položiek do zoznamu profilov vo vlastnom dialógu.

A teraz sa pozrime na vlastné funkcie. Pretože makrá sú pomerne dobre okomentované, nebudeme bližšie rozoberať to, ako sú naprogramované.

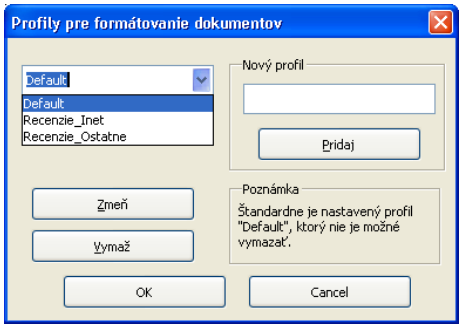
```
function inicializacny_subor (nazov_suboru as string) as
string
 ' Funkcia vracia názov inicializačného súboru v
„UserConfig“ adresári aktuálneho používateľa
 ' K názvu súboru pridávame automaticky príponu „.dat“
 dim cesty as object
 cesty=CreateUnoService("com.sun.star.util.PathSettings")
 inicializacny_subor=cesty.UserConfig+"/"+nazov_suboru+".da
t"
end function
```

```
sub zapis_nazvy_profilov
 ' Procedúra je určená na zápis názov profilov do
definičného súboru
 dim i, subor as integer

 subor = Freefile
 open
inicializacny_subor("Zoznam_Profilov_Formatovania_Dokumento
v") for Output as #subor
 for i=0 to pocet_profilov
  print #subor, zoznam_profilov(i)
 next i
 close #subor
end sub

sub init_dialog_profily
 ' Inicializácia zoznamu profilov z definičného súboru
 ' Vždy máme minimálne profil „Default“
 dim i, subor as integer
 dim riadok as string

 zoznam_profilov(0)="Default"
 pocet_profilov=0
 oVstupy=dlg.getControl("Zoznam_Profilov")
 oVstupy.AddItem("Default", 0)

 ' Ak nastavovací súbor neexistuje, vytvoríme ho
 if not
fileexists(inicializacny_subor("Zoznam_Profilov_Formatovani
a_Dokumentov")) then
   zapis_nazvy_profilov
 end if
 ' Teraz súbor určite existuje - načítame ho
 subor = Freefile
 open
inicializacny_subor("Zoznam_Profilov_Formatovania_Dokumento
v") for Input as #subor
 ' Inicializácia pomocného počítadla počtu profilov
 i=0
 While not eof(subor)
  Line Input #subor, riadok
  if i>0 then
   ' Profil Default už máme zapísaný v zozname, zaujímajú
nás iba tie ostatné
   pocet_profilov=pocet_profilov+1 ' aktuálny počat profilov
   oVstupy.AddItem(riadok, pocet_profilov) ' pridanie
profilu do dialógu
   zoznam_profilov(pocet_profilov)=riadok ' pridanie
profilu do zoznamu pre testovanie
```

```
  end if
  i=i+1
 wend
 close #subor

 dlg.model.Zoznam_Profilov.Text="Default" ' Aby sa nám
zobrazil v zozname prvý profil

end sub

sub Vloz_profil_Initiated
 ' Túto procedúru musíme priradiť k tlačidlu „Pridaj“!

 pom_text=dlg.model.Novy_profil.Text
 if pom_text<>"" then
  ' Názov musí byť zadaný
  existuje=false
  for i=0 to pocet_profilov
   ' Otestujeme či profil s daným názvom existuje bez
ohľadu na veľkosť písmen
   existuje=existuje or
(ucase(pom_text)=ucase(zoznam_profilov(i)))
  next i
  if not existuje then
   ' Ak profil neexistuje, pridáme ho do zoznamu
   oVstupy=dlg.getControl("Zoznam_Profilov")
   pocet_profilov=pocet_profilov+1
   zoznam_profilov(pocet_profilov)=pom_text
   oVstupy.AddItem(pom_text, pocet_profilov)
  else
   ' Oznam o tom, že nie je možné pridať už existujúci
profil
   msgbox "Profil "+pom_text+" už existuje."
  end if
 else
  'Oznam o tom, že názov musí byť zadaný
  msgbox "Názov musí byť zadaný"
 end if
end sub

sub Vymaz_profil_Initiated
 ' Túto procedúru musíme priradiť k tlačidlu „Vymaž“!

 oVstupy=dlg.getControl("Zoznam_Profilov")
 ' Priradenie názvu vybraného profilu do pomocnej premennej
 pom_text=dlg.model.Zoznam_Profilov.Text
 if pom_text<>"Default" then
  ' Mažeme iba iné profily ako je „Default“
  for i=1 to pocet_profilov
```

```
    ' Vyhľadáme pozíciu vybraného profilu
    if pom_text=zoznam_profilov(i) then
     ' Vymažeme profil z dialógu
     oVstupy.removeItems(i, 1)
     for j=i to pocet_profilov-1
      ' Vymažeme profil z pomocného zoznamu a posunieme
položky v zozname
      zoznam_profilov(j)=zoznam_profilov(j+1)
     next j
    end if
   next i
   ' Upravíme počet záznamov
   pocet_profilov=pocet_profilov-1
   ' Nastavíme zobrazovaný text v rozbaľovacom poli, lebo
predtým zobrazený sme práve vymazali
   dlg.model.Zoznam_Profilov.Text="Default"
  else
   ' Oznam o tom, že nemôžeme vymazať profil „Default“
   msgbox "Profil Deafault nie je možné vymazať!"
  end if

end sub

sub Zmen_profil_Initiated
 ' Túto procedúru musíme priradiť k tlačidlu „Zmeň“!

 ' Uloženie názvu vybraného profilu do pomocnej premennej
 pom_text=dlg.model.Zoznam_Profilov.Text
 ' Nastavenie parametrov pre formátovanie
 Spusti_dialog_formatovania(pom_text)

end sub

sub Nastav_Profil_Formatovania
 ' Vlastná funkcia pre spustenie dialógu „Dialog_profily“

 DialogLibraries.LoadLibrary("Vlastne_makra")
 dlg=CreateUnoDialog(DialogLibraries.Vlastne_makra.Dialog_P
rofily)
 init_dialog_profily
 Citaj_definiciu_profilu("Default")

 if dlg.Execute()=1 then
  zapis_nazvy_profilov
  Aktualny_profil=dlg.model.Zoznam_profilov.text
 end if

 ' Ak nie je náhodou vybraný žiadny profil, nastaví sa
profil „Default“
```

```
 if Aktualny_profil="" then
  Aktualny_profil="Default"
 end if
 Formatovanie.Nastav_formatovanie

 dlg.dispose()
end sub
```



Obrázok 36.2: Dialóg pre nastavenie používateľského profilu

Procedúry „Spusti_dialog_formatovanie“ (bola volaná v procedúre „Zmen_profil_Initiated“) a „Formatovanie.Nastav_formatovanie“ (bola volaná v procedúre „Nastav_Profil_Formatovania“) budú uvedené v ďalších dieloch seriálu.

36.3. Makrá pre prácu s vlastnými používateľskými profilmi

V dnešnom pokračovaní seriálu o makrách v OpenOffice.org si uvedieme procedúry a funkcie pre nastavovanie parametrov formátovania dokumentov osobitne podľa zvoleného používateľského profilu.

V predchádzajúcom dieli seriálu sme si ukázali prácu s používateľskými profilmi s tým, že vzhľadom na značný rozsah procedúr a funkcií, ktoré s tým následne súvisia, tieto rozdelíme do viacerých skupín a uvedieme ich postupne. Dnes sa budeme venovať skupine procedúr pre nastavovanie parametrov formátovania dokumentov (dialóg „ Dialog_Formatovanie“).

Podobne, ako minule, nebudeme sa vlastným procedúram venovať príliš osobitne, pretože sú pomerne dobre okomentované a uvedieme iba ich zdrojové texty. Premenné dialógu „Dialog_Formatovanie“ sme podrobne rozoberali pri jeho definícií.

Vlastné volanie dialógu „Dialog_Formatovanie“ je zabezpečené cez dialóg „Dialog_Profily“ (tlačidlo „Zmeň“) a preto nie je potrebné, aby sa volalo osobitne. Ešte spomeňme, že nasledujúce procedúry budeme ukladať do rovnakého modulu pre makrá ako minule, t.j. do modulu „Profily_funkcie“.

```
sub Zapis_definiciu_profilu(profil as string)
 ' Procedúra zapíše príznaky pre formátovanie do súboru pre
daný profil.
 ' Názov súboru je totožný s názvom profilu.
 dim i, subor as integer

 subor = Freefile
 open inicializacny_subor(profil) for Output as #subor
 for i=1 to 16
  print #subor, str(priznaky_formatovania(i))
 next i
 close #subor
end sub

sub Citaj_definiciu_profilu(profil as string)
 ' Procedúra načíta príznaky zo súboru pre daný profil.
 ' Ak súbor neexistuje, alebo je poškodený (má menej údajov
```

```
ako je potrebné), bude opravený.

 dim i, subor as integer
 dim riadok as string
 ' Inicializácia na štandardné štandardné hodnoty - všetky
políčka sú zaškrtnuté.
 ' Ako pevné medzery sa vkladajú NBSP.
 ' Za jednoznakovými predložkami vrátane "a" sa vkladajú
NBSP „off-line“.

 for i=1 to 16
  Priznaky_formatovania(i)=1
 next i
 ' Nastavenie „off-line“ vkladania medzier za jednoznakové
predložky
 Priznaky_formatovania(4)=2

 ' Ak nastavovací súbor neexistuje, vytvoríme ho a uložíme
do neho štandardné hodnoty
 if not fileexists(inicializacny_subor(profil)) then
   Zapis_definiciu_profilu(profil)
 end if

 ' Teraz súbor určite existuje - načítame ho
 subor = Freefile
  open inicializacny_subor(profil) for Input as #subor
 i=0
 While not eof(subor)
  i=i+1
  Line Input #subor, riadok
  Priznaky_formatovania(i)=val(riadok)
 wend
 close #subor

 ' Ak v súbore chýbal nejaký údaj, tak to teraz opravíme
 if i<16 then
  Zapis_definiciu_profilu(profil)
 end if

end sub

sub Init_dialog_formatovania(oDialog as object, profil as
string)
 ' Inicializácia dialógu „Dialog_Formatovanie“ podľa
definičného súboru vybraného profilu.

 Citaj_definiciu_profilu(profil)

 oDialog.model.Za_predlozkami.State=Priznaky_formatovania(1
```

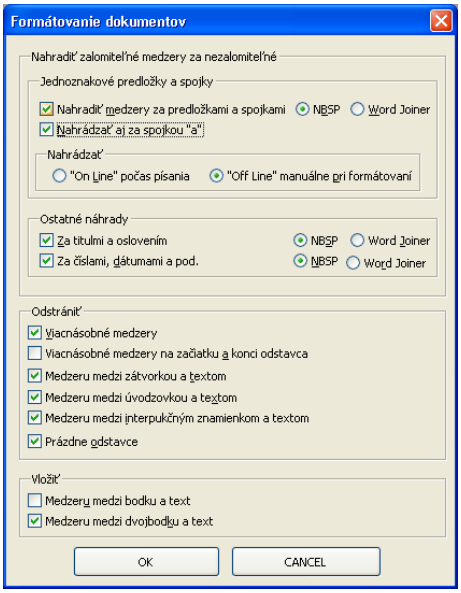
```
)

 oDialog.model.TabIndex=4
 oDialog.model.TabIndex=5
 select case Priznaky_formatovania(2)
  case 2
   oDialog.model.NBSP1.State=0
   oDialog.model.WJ1.State=1
  case else
   oDialog.model.NBSP1.State=1
   oDialog.model.WJ1.State=0
 end select

 oDialog.model.Za_spojkou_a.State=Priznaky_formatovania(3)

 oDialog.model.TabIndex=8
 oDialog.model.TabIndex=9
 select case Priznaky_formatovania(4)
  case 2
   oDialog.model.On_line.State=0
   oDialog.model.Off_line.State=1
  case else
   oDialog.model.On_line.State=1
   oDialog.model.Off_line.State=0
 end select

 oDialog.model.Za_titulmi.State=Priznaky_formatovania(5)

 oDialog.model.TabIndex=13
 oDialog.model.TabIndex=14
 select case Priznaky_formatovania(6)
  case 2
   oDialog.model.NBSP2.State=0
   oDialog.model.WJ2.State=1
  case else
   oDialog.model.NBSP2.State=1
   oDialog.model.WJ2.State=0
 end select

 oDialog.model.Za_cislami.State=Priznaky_formatovania(7)

 oDialog.model.TabIndex=16
 oDialog.model.TabIndex=17
 select case Priznaky_formatovania(8)
  case 2
   oDialog.model.NBSP3.State=0
   oDialog.model.WJ3.State=1
  case else
   oDialog.model.NBSP3.State=1
```

```
    oDialog.model.WJ3.State=0
  end select

  oDialog.model.Zrus_medzery.State=Priznaky_formatovania(9)
  oDialog.model.Zrus_okraje.State=Priznaky_formatovania(10)
  oDialog.model.Zrus_zatvorky.State=Priznaky_formatovania(11)
  oDialog.model.Zrus_uvodzovky.State=Priznaky_formatovania(12
)
  oDialog.model.Zrus_interpunkcie.State=Priznaky_formatovani
a(13)
  oDialog.model.Zrus_odseky.State=Priznaky_formatovania(14)

  oDialog.model.Vloz_bodku.State=Priznaky_formatovania(15)
  oDialog.model.Vloz_interpunkciu.State=Priznaky_formatovani
a(16)

end sub

sub Spusti_dialog_formatovania (profil as string)
  ' Vlastná práca s dialógom „Dialog_Formatovanie“.
  ' Volanie tohto dialógu je zabezpečené v dialógu
„Dialog_Profily“ cez tlačidlo „Zmeň“.

  dim oDialog as object
  ' Ak nie je zadaný názov profilu, budeme robiť s profilom
„Default“.
  if profil="" then profil="Default"

  oDialog=CreateUnoDialog(DialogLibraries.Vlastne_makra.Dial
og_Formatovanie)
  Init_dialog_formatovania(oDialog, profil)

  if oDialog.Execute()=1 then
    ' Nastavenie premenných pre formátovanie podľa príznakov
v dialógu.

    Priznaky_formatovania(1)=oDialog.model.Za_predlozkami.Sta
te

    if oDialog.model.NBSP1.State=1 then
      Priznaky_formatovania(2)=1
    else
      Priznaky_formatovania(2)=2
    end if

    Priznaky_formatovania(3)=oDialog.model.Za_spojkou_a.State
    if oDialog.model.On_line.State=1 then
      Priznaky_formatovania(4)=1
    else
```

```
   Priznaky_formatovania(4)=2
  end if

  Priznaky_formatovania(5)=oDialog.model.Za_titulmi.State
  if oDialog.model.NBSP2.State=1 then
   Priznaky_formatovania(6)=1
  else
   Priznaky_formatovania(6)=2
  end if

  Priznaky_formatovania(7)=oDialog.model.Za_cislami.State

  if oDialog.model.NBSP3.State=1 then
   Priznaky_formatovania(8)=1
  else
   Priznaky_formatovania(8)=2
  end if

  Priznaky_formatovania(9)=oDialog.model.Zrus_medzery.State
  Priznaky_formatovania(10)=oDialog.model.Zrus_okraje.State
  Priznaky_formatovania(11)=oDialog.model.Zrus_zatvorky.Stat
e
  Priznaky_formatovania(12)=oDialog.model.Zrus_uvodzovky.Sta
te
  Priznaky_formatovania(13)=oDialog.model.Zrus_interpunkcie
.State
  Priznaky_formatovania(14)=oDialog.model.Zrus_odseky.State

  Priznaky_formatovania(15)=oDialog
.model.Vloz_bodku.State
  Priznaky_formatovania(16)=oDialog
.model.Vloz_interpunkciu.State

  Zapis_definiciu_profilu(profil)
 end if

 oDialog.dispose()

end sub
```

Pokiaľ sa predsa len nájdu používatelia, ktorí nechcú využívať výhody viacerých profilov, môžu si upraviť začiatok procedúry „Spusti_dialog_formatovania“ (ktorú v tomto prípade musia volať osobitne) napríklad takto:

```
  sub Spusti_dialog_formatovania
   ' Vlastná práca s dialógom
„Dialog_Formatovanie“.
```



Obrázok 36.3: Dialóg pre nastavenie parametrov formátovania dokumentu

```
dim oDialog as object
dim profil as string

profil="Default"
```

Pravdaže, ako názov súboru pre uloženie parametrov (ktorý je totožný s názvom profilu a v tomto prípade je to „Default“) si môžu zvoliť ľubovoľný iný názov.

37. Makrá pre formátovanie dokumentu

V nasledujúcej časti si postupne uvedieme všetky potrebné makrá pre formátovanie dokumentu s využitím nastaviteľných parametrov v rôznych používateľských profiloch.

37.1. Nezalomiteľné medzery počas písania textu

V rámci formátovania dokumentu má osobitné postavenie vkladanie nezalomiteľných medzier za jednoznakové predložky a spojky, pretože pri týchto máme možnosť ich vkladania priamo počas písania textu, a, pravdaže, aj počas celkového formátovania dokumentu.

Čiastočne sa teda musíme vrátiť k štvrtej časti tohto seriálu, kde sme sa zmieňovanou problematikou zaoberali. Pravdaže, nemá zmysel, aby sme znovu rozoberali všetky podrobnosti, ktoré sme vtedy uviedli a preto sa hneď budeme venovať príslušným makrám.

V zozname modulov si zadefinujeme modul – „Nezalomitelna_predlozky_funkcie“, kde si naprogramujeme príslušné funkcie pre vkladanie nezalomiteľných medzier priamo počas písania textu. Pravdaže, v príslušných funkciách už budeme využívať nastavenia, ktoré sme si nastavili a vybrali podľa nami definovaných používateľských profilov.

Aby sme sa v nových nastaveniach (príznakoch) lepšie orientovali, ešte raz si v krátkosti zopakujme ich význam aj s príslušnými hodnotami. Príznaky sú definované v globálnej premennej „Priznaky_formatovania“, ktorú sme zadefinovali v module „Profily_funkcie“:

Priznaky_formatovania(1) – príznak (=1) vkladania nezalomiteľných medzier za jednoznakovými predložkami a spojkami.

Priznaky_formatovania(2) – znak, ktorý sa vkladá za jednoznakové predložky a spojky (NBSP=1, WJ=2).

Priznaky_formatovania(3) – príznak, či sa má vkladať nezalomiteľná medzera aj za predložku „a“ (=1).

Priznaky_formatovania(4) – príznak, či sa majú nezalomiteľné medzery vkladať za jednoznakové spojky (ak je to povolené) priamo (ON-line) počas písania textu (=1).

Priznaky_formatovania(5) – príznak (=1) vkladania nezalomiteľných medzier za akademické tituly.

Priznaky_formatovania(6) – znak, ktorý sa vkladá za akademické tituly (NBSP=1, WJ=2).

Priznaky_formatovania(7) – príznak (=1) vkladania nezalomiteľných medzier za

číslice a dátumy.

Priznaky_formatovania(8) – znak, ktorý sa vkladá za číslice a dátumy (NBSP=1, WJ=2).

Priznaky_formatovania(9) – príznak (=1), či sa majú odstrániť viacnásobné medzery.

Priznaky_formatovania(10) – príznak (=1), či sa majú odstrániť viacnásobné medzery na začiatku a konci odsekov.

Priznaky_formatovania(11) – príznak (=1), či sa majú odstrániť medzery medzi zátvorkami a textom.

Priznaky_formatovania(12) – príznak (=1), či sa majú odstrániť medzery medzi úvodzovkami a textom.

Priznaky_formatovania(13) – príznak (=1), či sa majú odstrániť medzery medzi interpunkčnými znamienkami a textom.

Priznaky_formatovania(14) – príznak (=1), či sa majú odstrániť prázdne odseky.

Priznaky_formatovania(15) – príznak (=1), či sa má vložiť medzera medzi bodku a text.

Priznaky_formatovania(16) – príznak (=1), či sa má vložiť medzera medzi dvojbodku a text.

Ale teraz sa už venujme vlastným makrám pre vkladanie nezalomiteľných medzier za jednoznakové spojky a predložky priamo počas ich písania:

```
global ObsluhaDokumentu, ObsluhaKlavesnice as object
global PosledneSlovo, OddelovaceSlov as string
global Obsluhovat_Klavesnicu as boolean

function JednoznakovePredlozky as string
 ' Pretože počas písania môžeme zmeniť nastavenie profilu,
 ' definícia zoznamu predložiek je definovaná ako funkcia
 JednoznakovePredlozky="AiIkKoOsSuUvVzZ"
 if Profily_funkcie.Priznaky_formatovania(3)=1 then
  JednoznakovePredlozky="a"+JednoznakovePredlozky
 end if
end function

sub SpustiObsluhuKlavesnice
 on error goto next
 ' Obsluhu klávesnice spustíme iba vtedy, ak vôbec povolili
zámenu predložiek
 ' a zároveň chceme túto zámenu robiť ON-line
 Obsluhovat_Klavesnicu=(Profily_funkcie.Priznaky_formatovani
a(1)=1) and (Profily_funkcie.Priznaky_formatovania(4)=1)
 PosledneSlovo = ""
 ' medzera, nezalomiteľná medzera, tabelátor, CR, LF
 OddelovaceSlov=" "+chr(&HA0)+chr(&H09)+chr(&H0A)+chr(&H0D)
 ' pomlčka, lomka, bodka, čiarka, výkričník, otáznik,
bodkočiarka, trojbodka
```

```
 OddelovaceSlov=OddelovaceSlov+"-/.,!?;…"
 ' zátvorky
 OddelovaceSlov=OddelovaceSlov+"()[]{}"
 if
ThisComponent.supportsService("com.sun.star.text.TextDocume
nt") then
   ' Sme v textovom dokumente, kde chceme nahrádzať medzery
   ObsluhaDokumentu = ThisComponent.getCurrentController
   ObsluhaKlavesnice =
createUnoListener("Klavesnica_","com.sun.star.awt.XKeyHandle
r")
   ObsluhaDokumentu.addKeyHandler(ObsluhaKlavesnice)
 end if
end sub

sub UkonciObsluhuKlavesnice
 on error goto next
 if
ThisComponent.supportsService("com.sun.star.text.TextDocume
nt") then
   ' Sme v textovom dokumente a chceme ukončiť obsluhu
klávesnice
    ObsluhaDokumentu.removeKeyHandler(ObsluhaKlavesnice)
    PosledneSlovo = ""
 end if
end sub

function VlozZnak(Retazec as string)
 ' Na aktuálnu pozíciu v dokumente vložíme reťazec
 dim ViditelnyKurzor, AktualnyText, AktualnyKurzor as object

 ViditelnyKurzor =
ThisComponent.getCurrentController().getViewCursor()
 AktualnyText = ThisComponent.getText()
 AktualnyKurzor =
AktualnyText.createTextCursorByRange(ViditelnyKurzor.getStart
())
 AktualnyText.insertString(AktualnyKurzor.getStart(),
Retazec, true)
end function

function Klavesnica_KeyPressed(StlacenaKlavesa) as boolean
' Obsluha stlačenej klávesy
 dim Obsluzena as boolean
 'Predpokláme, že nebudeme vkladať nezalomiteľnú medzeru
 Obsluzena=False
 if instr(OddelovaceSlov, StlacenaKlavesa.KeyChar)<>0 then
   ' Došli sme na koniec slova
   if Obsluhovat_Klavesnicu then
```

```
    ' Ak máme vôbec meniť medzery za nezalomiteľné
    if ZistiPredlozku and (StlacenaKlavesa.KeyChar=" ") then
     ' Je to predložka, ktorá končí zalomiteľnou medzerou -
túto nahradíme
     if Profily_funkcie.Priznaky_formatovania(2)=1 then
      VlozZnak(chr$(&HA0)) ' No Break Space (NBSP)
      ' Sami sme vložili medzeru - obslúžili sme stlačenú
klávesu
      Obsluzena=true
     else
      VlozZnak(chr$(&H2060)) ' Word Joiner (WJ)
      ' V tomto prípade sa musí ešte vložiť aj zadaná medzera
a preto sme neobslúžili stlačenú klávesu
     end if
    end if
   end if
   PosledneSlovo=""
 else
   ' Ešte musíme otestovať riadiace znaky, ktoré nebudeme
sledovať v zadávanom slove
   select case StlacenaKlavesa.KeyChar
    case chr$(&H00) to chr$(&H07)
     PosledneSlovo=""
    case chr$(&H08) ' BS
     if len(PosledneSlovo)>0 then
      PosledneSlovo=left(PosledneSlovo, len(PosledneSlovo)-
1)
     else
      PosledneSlovo=""
     end if
    case chr$(&H09) to chr$(&H1F)
     PosledneSlovo=""
    case chr$(&H20) to chr$(&H7E)
     PosledneSlovo=PosledneSlovo+StlacenaKlavesa.KeyChar
    case chr$(&H7F) ' CTRL BS
     PosledneSlovo=""
    case else
     PosledneSlovo=PosledneSlovo+StlacenaKlavesa.KeyChar
   end select
 end if
 Klavesnica_KeyPressed=Obsluzena
end function

function Klavesnica_KeyReleased(StlacenaKlavesa) as boolean
' Obsluha pustenej klávesy
 Klavesnica_KeyReleased = False
end function

function ZistiPredlozku as boolean
```

```
dim JePredlozka as boolean
 JePredlozka=false
 if len(PosledneSlovo)=1 then
  ' Ak je to jednoznakové slovo, tak to môže byť predložka
  JePredlozka=instr(1,JednoznakovePredlozky, PosledneSlovo,
0)
 end if
 ZistiPredlozku=JePredlozka
end function
```

Pravdaže, nesmieme zabudnúť ani na funkcie, ktoré nám zjednodušujú prístup k uvedeným makrám. Na tieto funkcie si zadefinujeme nový modul „Nezalomitelne_predlozky“:

```
sub Automaticky_Obsluhuj_Klavesnicu
 ' Procedúra pre automatické spustenie vkladania
nezalomiteľných medzier
 ' ktorú môžeme volať počas otvárania súboru. Spustenie
závisí od nastavených
 ' parametrov v používateľskom profile.
 Formatovanie.Nastav_formatovanie
 ' Podľa aktuálneho profilu sa zistia príznaky
 if (Profily_funkcie.Priznaky_formatovania(1)=1) and
(Profily_funkcie.Priznaky_formatovania(4)=1) then
  Nezalomitelne_predlozky_funkcie.SpustiObsluhuKlavesnice
 end if
end sub

sub Obsluhuj_Klavesnicu
 ' Spustenie automatického vkladania bez ohľadu na
nastavené parametre v používateľskom profile.
 Nezalomitelne_predlozky_funkcie.SpustiObsluhuKlavesnice
 Nezalomitelne_predlozky_funkcie.Obsluhovat_Klavesnicu=True
end sub

sub Neobsluhuj_Klavesnicu
 ' Zastavenie automatického vkladania bez ohľadu na
nastavené parametre v používateľskom profile.
 Nezalomitelne_predlozky_funkcie.Obsluhovat_Klavesnicu=Fals
e
 Nezalomitelne_predlozky_funkcie.UkonciObsluhuKlavesnice
end sub

sub Povol_Klavesnicu
 ' Povolenie pozastaveného automatického vkladania bez
ohľadu na nastavené parametre v používateľskom profile.
 Nezalomitelne_predlozky_funkcie.Obsluhovat_Klavesnicu=True
end sub

sub Pozastav_Klavesnicu
```

```
 ' Pozastavenie automatického vkladania bez ohľadu na
nastavené parametre v používateľskom profile.
 Nezalomitelne_predlozky_funkcie.Obsluhovat_Klavesnicu=Fals
e
end sub
```

37.2. Funkcie pre výmenu a hľadanie

V minulom dieli sme si uviedli funkcie pre automatické vkladanie nezalomiteľnej medzery za jednoznakové predložky a spojky priamo počas písania textu. Teraz nám už nezostáva nič iné, ako si uviesť makrá pre ostatné formátovanie dokumentu, ktoré sme si v našom seriáli už viackrát spomínali. Teraz ich upravíme na nastavenia podľa parametrov, ktoré si môžeme definovať v používateľských profiloch a nebudeme sa už venovať iným verziám.

Vlastné formátovacie funkcie si uložíme do modulu „Formatovanie_funkcie“ a ponecháme ich v obecnom tvare, aby sa dali použiť aj bez obmedzení, ktoré sú nastavené v používateľských príznakoch pre formátovanie dokumentu. Tieto príznaky budeme následne akceptovať až vo funkciách celkového formátovania dokumentu (tieto si uložíme do modulu „Formatovanie“), čím dosiahneme obecnejšie možnosti ich použitia.

Hoci sa makrá oproti naposledy uverejneným verziám dosť pomenili, sú dostatočne okomentované na to, aby sme sa im osobitne venovali a preto znovu uvedieme iba ich zdrojové texty. Vzhľadom na veľký rozsah týchto zdrojových textov dnes si znovu uvedieme iba makrá pre vyhľadávanie a nahrádzanie textu.

```
REM Definície spoločných premenných sú úplne pred
definíciou podprogramov
dim Dokument, Vymena, Hladaj as object
dim NM, TAB, WJ as string

REM Inicializácia spoločných premenných
sub Init
 Dokument=ThisComponent
 Vymena=Dokument.createReplaceDescriptor()
 Hladaj=Dokument.createSearchDescriptor()
 NM=chr$(&HA0) ' Nezalomiteľná medzera - hexadecimálny kód
A0
 TAB=chr$(&H09) ' Tabelátor - hexadecimálny kód 09
 WJ=chr$(&H2060) ' Word Joiner
end sub

REM Výmena reťazca Co za reťazec Za
function Vymen(Co, Za as string, Regularne, Cele_slova,
Velke_pismena as Boolean) as long
 Vymena.SearchString=Co
 Vymena.ReplaceString=Za
 Vymena.SearchRegularExpression=Regularne
 Vymena.SearchWords=Cele_slova
```

```
 Vymena.SearchCaseSensitive=Velke_pismena
 Vymen=Dokument.replaceAll(Vymena)
end function

REM Vlastná výmena reťazca Co za reťazec Za vo vnútri
reťazca V_Com
function Zamen_Vo_Vnutri (V_Com, Co, Za as string) as
string
 dim pom as string
 dim i, j as long
 pom=V_Com
 i=instr(V_Com, Co)
 j=len(V_Com)-i-len(Co)+1
 if i>0 then
  pom=left(V_Com, i-1)+Za
  if j>0 then
   pom=pom+right(V_Com, j)
  end if
 end if
 Zamen_Vo_Vnutri=pom
end function

REM Výmena vnútorného reťazca Co za reťazec Za hľadaním
reťazca V_Com
function Vymen_hladanim(V_Com, Co, Za as string, Regularne,
Cele_slova, Velke_pismena as Boolean) as long
 dim nasiel as object
 dim kolko as long
 Hladaj.searchString=V_Com
 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
   nasiel.String=Zamen_Vo_Vnutri(nasiel.String, Co, Za)
   kolko=kolko+1
  end if
  nasiel = Dokument.findNext(nasiel.End, Hladaj)
 Loop
 Vymen_hladanim=kolko
end function
REM Vyhľadanie reťazca V_Com a na pozíciu Kde do neho vloží
reťazec Co
function Vloz_hladanim(V_Com, Co as string, Kde as long,
Regularne, Cele_slova, Velke_pismena as Boolean) as long
 dim nasiel as object
 dim kolko, i, j as long
```

```
 dim pom as string
 Hladaj.searchString=V_Com
 Hladaj.SearchRegularExpression=Regularne
 Hladaj.SearchWords=Cele_slova
 Hladaj.SearchCaseSensitive=Velke_pismena
 kolko=0
 i=kde+1
 nasiel = Dokument.findFirst(Hladaj)
 Do While NOT isNull(nasiel)
  if len(nasiel.String)>0 then
   j=len(nasiel.String)-i+1
   pom=left(nasiel.String, i-1)+Co
   if j>0 then
    pom=pom+right(nasiel.String, j)
   end if
   nasiel.String=pom
   kolko=kolko+1
  end if
  nasiel = Dokument.findNext(nasiel.End, Hladaj)
 Loop
 Vloz_hladanim=kolko

end function
```

37.3. Vymazávanie viacnásobných medzier

Naposledy sme si uviedli všeobecné makrá pre vyhľadávanie a nahrádzanie textu. Dnes budeme pokračovať v už konkrétnych formátovacích makrách a začneme makrami pre mazanie viacnásobných medzier. Myslíme si, že vzhľadom na rozsah iba makra pre odstraňovanie medzier je toho viac než dosť, pretože toto makro prekonalo veľmi veľké zmeny a vlastne spojilo do seba viacero drobných podprogramov.

```
REM Makro pre odstraňovanie viacnásobných medzier
function f_Viacnasobna_Medzera (viacnasobne, odseky,
uvodzovky, zatvorky, znamienka as integer) as long
 ' Funkcia odstráni medzery podľa parametrov (ak sa =1, tak
platia):
 ' viacnasobne - odtsránenie viacnásobných medzier v strede
dokumentu
 ' odseky - odstránanie viacnásobných medzier na začiatku a
konci odseku
 ' zatvorky - odstránenie medzery medzi zátvorkou a textom
 ' uvodzovky - odstránenie medzery medzi úvodzovkou a
textom
 ' znamienka - odstránenie medzery medzi znamienkami a
textom

 dim kolko as Long
 dim hladany_retazec as string
```

```
 kolko=0

 if viacnasobne=1 then
  kolko=kolko+Vymen("  +"," ", TRUE, false, false)
  kolko=kolko+Vymen(" "+NM+" |"+NM+" | "+NM, NM, TRUE,
false, false)
  kolko=kolko+Vymen(" \t |\t | \t","\t", TRUE, false,
false)
 end if

 if odseky=1 then
  kolko=kolko+Vymen("^[: space:]*|[: space:]*$","", TRUE,
false, false)
  kolko=kolko+Vymen("^\t*|\t*$","", TRUE, false, false)
 end if

 ' Makro naprogramujeme tak, aby čo nejmenejkrát
prechádzalo dokumentom
 ' T.j. najprv si pripravíme kompletný regulárny výraz a až
potom budeme prevádzať výmeny
 hladany_retazec=""
 if uvodzovky=1 then
  ' Nastavíme si regulárny výraz pre ľavé úvodzovky
  hladany_retaz="(„|«|"+chr$(&H22)
 end if
 if zatvorky=1 then
  ' Nastavíme si regulárny výraz pre ľavé zátvorky
  if hladany_retazec<>"" then
   ' Ak už máme úvodzovky, musíme pridať znak "alebo"
   hladany_retazec=hladany_retazec+"|"
  else
   ' Ešte nemáme nič, musíme zadať začiatok regulárneho
výrazu
   hladany_retazec="("
  end if
  hladany_retazec=hladany_retazec+"\(|\[|{"
 end if
 if hladany_retazec<>"" then
  ' Ešte ukončíme regulárny výraz
  hladany_retazec=hladany_retazec+")"
  ' Ak máme nastavený výraz pre zátvorky alebo úvodzovky,
tak vymažeme nadbytočné medzery
  ' Za nadbytočné medzery považujeme obyčajné medzery, NBSP
a tabelátory
  kolko=kolko+Vymen_hladanim(hladany_retazec+" "," ","",
true, false, false)
  kolko=kolko+Vymen_hladanim(hladany_retazec+NM, NM,"",
true, false, false)
  kolko=kolko+Vymen_hladanim(hladany_retazec+TAB, TAB,"",
```

```
true, false, false)
 end if

 hladany_retazec=""
 if uvodzovky=1 then
  ' Nastavíme si regulárny výraz pre pravé úvodzovky
  hladany_retaz="(“|»|"+chr$(&H22)
 end if
 if zatvorky=1 then
  ' Nastavíme si regulárny výraz pre pravé zátvorky
  if hladany_retazec<>"" then
   ' Ak už máme úvodzovky, musíme pridať znak "alebo"
   hladany_retazec=hladany_retazec+"|"
  else
   ' Ešte nemáme nič, musíme zadať začiatok regulárneho
výrazu
   hladany_retazec="("
  end if
  hladany_retazec=hladany_retazec+"\)|\]|}"
 end if
 if znamienka=1 then
  ' Nastavíme si regulárny výraz pre interpunkčné znamienka
  if hladany_retazec<>"" then
   ' Ak už máme úvodzovky alebo zátvorky, musíme pridať
znak "alebo"
   hladany_retazec=hladany_retazec+"|"
  else
   ' Ešte nemáme nič, musíme zadať začiatok regulárneho
výrazu
   hladany_retazec="("
  end if
  hladany_retazec=hladany_retazec+"\,|\.|!|\?|;|:|…"
 end if
 if hladany_retazec<>"" then
  ' Ešte ukončíme regulárny výraz
  hladany_retazec=hladany_retazec+")"
  ' Ak máme nastavený výraz pre zátvorky alebo úvodzovky,
tak vymažeme nadbytočné medzery
  ' Za nadbytočné medzery považujeme obyčajné medzery, NBSP
a tabelátory
  kolko=kolko+Vymen_hladanim(" "+hladany_retazec," ","",
true, false, false)
  kolko=kolko+Vymen_hladanim(NM+hladany_retazec, NM,"",
true, false, false)
  kolko=kolko+Vymen_hladanim(TAB+hladany_retazec, TAB,"",
true, false, false)
 end if
 f_Viacnasobna_Medzera=kolko
end function
```

37.4. Ostatné formátovacie funkcie

V ďalšom dieli seriálu o programovaní makier si doplníme všetky ostatné funkcie pre vlastné formátovanie dokumentu, ktoré sme doteraz v seriáli spomínali.

Minule sme si ukázali veľmi dlhé makro pre vymazávanie viacnásobných medzier. Dnes k tomu pridáme zvyšné formátovacie funkcie, s ktorými sme sa v tomto seriáli už postupne stretali – odstraňovanie prázdnych odsekov, nahrádzanie nezalomiteľných medzier za akademickými titulmi, číslami a pod. Pravdaže, nezabudneme ani vložiť chýbajúce medzery medzi interpunkčné znamienka a text.

```
REM Makro pre odstraňovanie prázdnych odsekov
function f_Prazdny_odsek as long
 kolko=Vymen("^$","", TRUE, false, false)
end function

REM Makro pre vkladanie nezalomiteľných medzier za
predložky a spojky
function f_Nezalomitelne_Spojky(spojka_a, NBSP as integer)
as long
 dim spojky as string
 dim znak as string

 ' Definícia znaku, za ktorý budeme meniť zalomiteľnú
medzeru
 if NBSP=1 then
  znak=NM
 else
  znak=WJ+" "
 end if

 ' Definícia regulárneho výrazu pre spojky
 if spojka_a=1 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) "
 end if
 f_Nezalomitelne_Spojky=Vymen_hladanim(spojky," ", znak,
true, false, true)

end function

REM Makro pre vkladanie nezalomiteľných medzier pred a za
akademické tituly
function f_Nezalomitelne_Tituly (NBSP as integer) as long
 dim Tituly_pred(), Tituly_pred_spec(), Tituly_za() as
string
 dim i, kolko as long
 dim znak as string
```

```
 ' Definícia znaku, za ktorý budeme meniť zalomiteľnú
medzeru
 if NBSP=1 then
  znak=NM
 else
  znak=WJ+" "
 end if

 Tituly_pred()=array("Bc", "Dr", "Ing", "JUDr", "Mgr",
"MUDr", "MVDr", "PaedDr", "PharmDr", "PhDr", "PhMr",
"RNDr", "ThDr")
 Tituly_pred_spec()=array("akad", "Akad", "arch", "Arch",
"doc", "Doc", "prof", "Prof")
 Tituly_za()=array("CSc.","DrSc.")

 ' Špeciálne vloženie nezalomiteľnej medzery medzi skratku
h. c. a následne pred ňu.
 kolko=Vymen("h. c.","h."+znak+"c.", false, false, false)
 kolko=kolko+Vymen(" h."+znak+"c.", znak+"h."+znak+"c.",
false, false, false)

 ' Pre všetky tituly pred menom, kde sa opravuje aj veľkosť
znakov
 for i=lbound(Tituly_pred()) to ubound(Tituly_pred())
  kolko=kolko+Vymen("\<"+Tituly_pred(i)+"\. ",
Tituly_pred(i)+"."+znak, true, false, false)
 next i

 ' Pre všetky tituly pred menom, kde sa neopravuje aj
veľkosť znakov
 for i=lbound(Tituly_pred_spec()) to
ubound(Tituly_pred_spec())
  kolko=kolko+Vymen("\<"+Tituly_pred_spec(i)+"\. ",
Tituly_pred_spec(i)+"."+znak, true, false, true)
 next i

 ' Pre všetky tituly za menom
 for i=lbound(Tituly_za()) to ubound(Tituly_za())
  kolko=kolko+Vymen(" "+Tituly_za(i), znak+Tituly_za(i),
false, false, false)
 next i

 f_Nezalomitelne_Tituly=kolko
end function

REM Makro pre vkladanie nezalomiteľných medzier medzi čísla
a text
function f_Nezalomitelne_Cisla (NBSP as integer) as long
 dim merne_jednotky as string
```

```
 dim kolko as long
 dim znak as string

 ' Definícia znaku, za ktorý budeme meniť zalomiteľnú
medzeru
 if NBSP=1 then
  znak=NM
 else
  znak=WJ+" "
 end if

 ' Definícia regulárneho výrazu pre merné jednotky
 merne_jednotky="(m|km|mm|cm)"

 ' Nezalomiteľné medzery do dátumov
 kolko=Vymen_hladanim("[0-9]\. [0-9]"," ", znak, true,
false, false)
 kolko=kolko+Vymen_hladanim("[0-9]\. [0-9]"," ", znak,
true, false, false)

 ' Jednociferné čísla
 kolko=kolko+Vymen_hladanim("\<[0-9] "," ", znak, true,
false, false)
 ' Telefónne čísla
 kolko=kolko+Vymen_hladanim("[0-9] [0-9]"," ", znak, true,
false, false)

 ' Merné jednotky
 kolko=kolko+Vymen_hladanim("[0-9] "+merne_jednotky," ",
znak, true, false, false)

 f_Nezalomitelne_Cisla=kolko

end function

REM Makro pre vkladanie chýbajúcich medzier pri
interpunkčných znamienkach
function f_Vloz_Interpunkcia_Medzera (bodka, ostatne as
integer) as long
 dim kolko as long

 kolko=0
 if bodka=1 then
  ' Vloženie chýbajúcej medzery medzi znak bodky a text
  kolko=Vloz_hladanim("[^0-9]\.[: alnum:]{1}"," ", 2,true,
false, false)
 end if
 if ostatne=1 then
  ' Vloženie chýbajúcej medzery medzi ostatné znamienka a
```

```
text
  kolko=kolko+Vloz_hladanim("(\?|!|;|…)[: alnum:]{1}"," ",
1,true, false, false)
  kolko=kolko+Vloz_hladanim("[^0-9](\,|:)[: alnum:]{1}","
", 2,true, false, false)
 end if
 f_Vloz_Interpunkcia_Medzera=kolko

end function
```

37.5. Záverečné funkcie

V (takmer) poslednom dieli o formátovaní dokumentu si ukážeme, ako vlastne dokážeme všetky naprogramované funkcie a dialógy využiť pomocou niekoľkých jednoduchých funkcií a procedúr.

Doteraz sme si uviedli všetky funkcie, ktoré chceme používať pre formátovanie nášho dokumentu. Zároveň sme si naprogramovali aj dialógový modul, pomocou ktorého dokážeme nastaviť viacero používateľských profilov. Ako to však všetko zjednotiť, aby sme v tom mali prehľad? Najlepšie tak, že si vytvoríme (ako sme už spomínali v minulých dieloch) modul „Formatovanie“, kde si naprogramujeme tieto makrá:

```
REM Makro pre načítanie nastavených formátovacích parametrov
aktuálneho profilu
sub Nastav_formatovanie
 ' ON-line/Off-line nahrádzanie sa riadi manuálne alebo
automaticky pomocou funkcií,
 ' ktoré sú definované v module "Nezalomitelne_predlozky" a
tu sa tym nezaoberame
 if Profily_funkcie.aktualny_profil="" then
  ' Ak nie je vybraný iný profil, nastaví sa defaultný
profil
  Profily_funkcie.aktualny_profil="Default"
 end if
 ' Načítame parametre
 Profily_funkcie.Citaj_definiciu_profilu(Profily_funkcie.akt
ualny_profil)
end sub

REM Makro pre celkové formátovanie dokumentu
sub Formatuj_dokument
 dim kolko as long
 Nastav_formatovanie
 Formatovanie_funkcie.Init
 kolko=Formatovanie_funkcie.f_Viacnasobna_Medzera(_
  Profily_funkcie.Priznaky_formatovania(9),_
  Profily_funkcie.Priznaky_formatovania(10),_
  Profily_funkcie.Priznaky_formatovania(12),_
  Profily_funkcie.Priznaky_formatovania(11),_
  Profily_funkcie.Priznaky_formatovania(13))
```

```
 if Profily_funkcie.Priznaky_formatovania(14)=1 then
  kolko=kolko+f_Prazdny_odsek
 end if
 if Profily_funkcie.Priznaky_formatovania(1)=1 then
  kolko=kolko+f_Nezalomitelne_Spojky(Profily_funkcie.Priznak
y_formatovania(3),_
   Profily_funkcie.Priznaky_formatovania(2))
 end if
 if Profily_funkcie.Priznaky_formatovania(5)=1 then
  kolko=kolko+f_Nezalomitelne_Tituly(Profily_funkcie.Priznak
y_formatovania(6))
 end if
 if Profily_funkcie.Priznaky_formatovania(7)=1 then
  kolko=kolko+f_Nezalomitelne_Cisla(Profily_funkcie.Priznaky
_formatovania(8))
 end if
 kolko=kolko+f_Vloz_Interpunkcia_Medzera(Profily_funkcie.Pri
znaky_formatovania(15),_
  Profily_funkcie.Priznaky_formatovania(16))
 msgbox("Prevedných "+kolko+" náhrad.", 0,"Formátovanie
textu")
end sub

REM Niekoľko makier ako ukážky aj iných volaní, než pomocou
celkového formátovania dokumentu

REM Makro pre vymazanie prázdnych odsekov
sub Prazdny_odsek
 Formatovanie_funkcie.Init
 msgbox("Vymazanie "+Formatovanie_funkcie.f_Prazdny_odsek()
+" prázdnych odsekov.", 0,"Prázdne odseky")
end sub

REM Makro pre vkladanie NBSP za predložky a spojky
sub Nezalomitelne_Spojky
 Formatovanie_funkcie.Init
 msgbox("Vložených
"+Formatovanie_funkcie.f_Nezalomitelne_Spojky(1,1)+"
nezalomiteľných medzier.", 0,"Nezalomiteľné spojky a
predložky")
end sub
```

38. Niekoľko slov na záver

Pre vlastnú prácu nám teraz stačia dve makrá – prvé je makro „Nastav_Profil_Formatovania“, ktoré nájdeme v module „Profily_funkcie“ a druhé je makro „Formatuj_dokument“, ktoré sme si práve naprogramovali do modulu „Formatovanie“. Ako ste si mohli všimnúť, v makre „Formatuj dokument“ sa nachádza na konci riadkov

znak podčiarknutia. Je to regulárny zápis v jazyku StarOffice Basic v prípade, že riadok je príliš dlhý a chceme ho takto rozdeliť na viacero riadkov, aby sme nestratili potrebný prehľad.

Pravdaže, ako sme si uviedli v predchádzajúcich zdrojových textoch, môžeme si vytvoriť viacero pomocných funkcií, ktoré budú volať iba niektoré z naprogramovaných funkcií so špecifickým nastavením parametrov. To všetko však už záleží od každého používateľa a jeho požiadavok.

Končíme piatu, veľmi veľkú časť o programovaní makier v OpenOffice.org. V šiestej časti budeme ešte stále pokračovať v oblasti práce s textovým procesorom Writer a uvedieme si niekoľko zaujímavých a praktických makier, ako je napríklad vzájomná zámena znakov Word Joiner za Non Break Space, zámena troch bodiek za znak „trojbodky“ a pod.

Makromoduly a drobnosti

V šiestej časti seriálu o programovaní makier v OpenOffice.org sa budeme venovať rôznym makrám, ktoré spríjemnia našu každodennú prácu. Nie všetky budeme programovať, ale predstavíme si aj niekoľko zaujímavých makier či celých makromodulov od iných autorov.

V doterajších častiach o programovaní makier pre OpenOffice.org sme sa venovali najmä formátovaniu dokumentov s tým, že sme si na týchto príkladoch postupne predstavili jazyk StarOffice Basic, typy premenných, prácu s dialógmi, možnosti nastavovania parametrov s ich pamätaním v rôznych profiloch a pod. Teraz zúročíme tieto poznatky a uvedieme si rôzne (nielen) drobné makrá, aby sme videli, že nie všetko sa musí venovať formátovaniu.

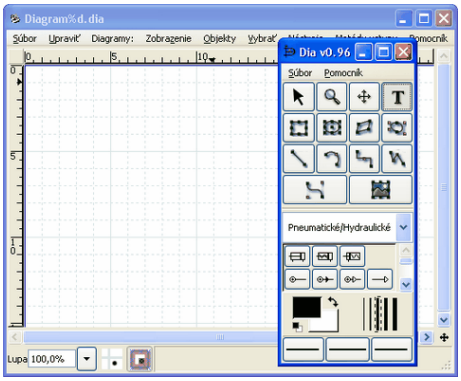
39. Makromodul (nielen) pre vzorce – Dmaths

Aby sme začali trochu netradične, dnes nebudeme programovať, ale predstavíme si francúzsky makromodul Dmaths. Tento makromodul je určený najmä pre tých, ktorí často pracujú so vzorcami. Určite mnohí z nich už neraz „nadávali“, že pre vloženie vzorca musia pracne vchádzať do modulu Math a prečo OpenOffice.org neumožňuje ich zadanie jednoduchšie, najmä ak dobre ovládajú skriptovací jazyk pre ich zadávanie.

A práve na riešenie takýchto problémov je ako stvorený predstavovaný makromodul Dmaths. Pretože jeho inštalácia je trochu špecifická, popíšeme si ju. Na tomto mieste sa ospravedlňujem majiteľom počítačov Apple, ale pretože ich nevlastním, nepopíšem inštaláciu pre ich operačný systém.

Po rozbalení distribučného ZIP balíka otvoríme súbor install.odt (pravdaže, musíme povoliť spúšťanie makier z tohto súboru). Následne klikneme na jazyk, v ktorom chceme, aby sa makromodul nainštaloval. Tu nás poteší, že medzi nimi nájdeme aj slovenčinu[4]. Na záver súbor zavrieme. V adresári, kde sme súbor rozbalili sa nám medzitým vytvoril súbor „instalmono.bat“ (pre operačný systém Windows) alebo súbor „instalmono.sh“ (pre operačný systém Linux).

Po tomto kroku musíme ukončiť OpenOffice.org vrátane spusteného rýchleho štartu a spustiť príslušný súbor „instalmono“. Tým spustíme vlastnú inštaláciu. Počas nej môžeme zadať cestu k trom rôznym programom, ktoré potom môžeme volať stlačením tlačidla na novovytvorenom paneli makromodulu Dmaths priamo z prostredia OpenOffice.org.



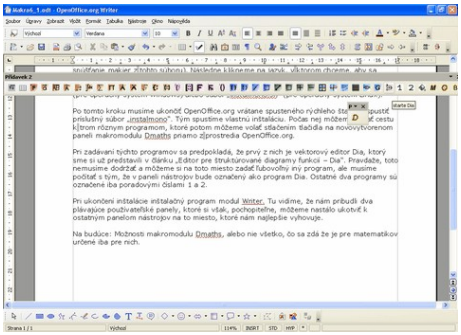
Obrázok 39.1: Vektorový editor DIA

Pri zadávaní týchto programov sa predpokladá, že prvý z nich je vektorový editor Dia, ktorý sme si už predstavili v článku „Editor pre štruktúrované diagramy – Dia“. Pravdaže, toto nemusíme dodržať a môžeme si na toto miesto zadať ľubovoľný iný program, ale musíme počítať s tým, že v paneli nástrojov bude označený ako program Dia. Os-

4 Žiaľ, toto pre najnovšie verzie už neplatí

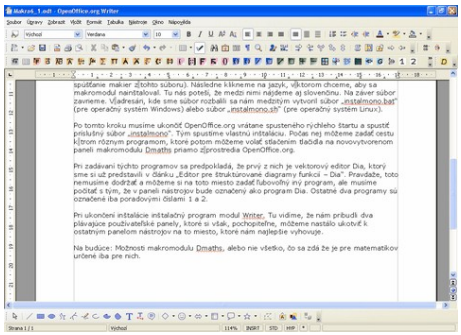
tatné dva programy sú označené iba poradovými číslami 1 a 2.

Pri ukončení inštalácie inštalačný program modul Writer. Tu vidíme, že nám pribudli dva plávajúce používateľské panely, ktoré si však, pochopiteľne, môžeme nastálo ukotviť k ostatným panelom nástrojov na to miesto, ktoré nám najlepšie vyhovuje.



Obrázok 39.2: Používateľské panely makromodulu Dmaths

Teraz si ukážeme možnosti, ktoré nám tento makromodul ponúka. Pri tomto popise sa budeme odvolávať na tlačidlá, ktoré sa nachádzajú v jeho pracovnom paneli. Pravdaže, ku mnohým z nich existujú aj klávesové skratky, ich používanie necháme už na láskavého čitateľa.

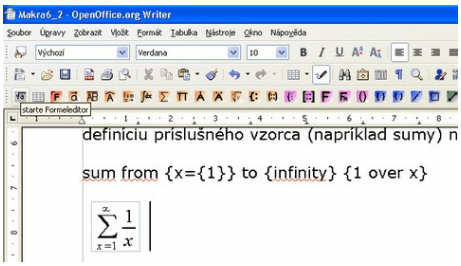


Obrázok 39.3: Ukotvený pracovný panel Dmaths

Pre tých, ktorí dobre ovládajú skriptovací jazyk modulu Math bude najzaujímavejšia prvá možnosť – vloženie vzorca priamo z prostredia textového procesora Writer. Stačí, ak napíšeme definíciu príslušného vzorca (napríklad sumy) na to miesto, kde ho chceme vložiť:

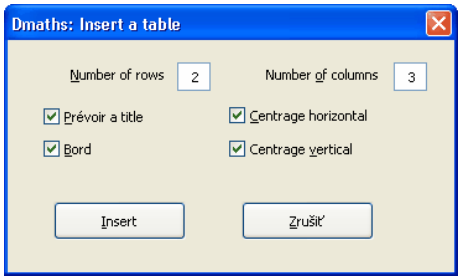
sum from {x={1}} to {infinity} {1 over x}

Definíciu potom označíme a stlačíme tlačidlo „starte Formeleditor“. Následne sa nám do textu automaticky vloží vzorec tak isto, ako keby sme ho zadali cez modul Math. Pokiaľ toto tlačidlo stlačíme bez označenia definície vzorca, otvorí sa nám modul Math.



Obrázok 39.4: Priame vloženie vzorca v textovom procesore

Ďalšia možnosť, ktorú nám ponúka makromodul Dmaths je vkladanie tabuliek. Žiaľ, už v tomto dialógu vidíme, že preklad do slovenčiny nie je urobený na všetkých miestach dokonale, ale nájdeme tu konkrétne anglicko-francúzsky text. Preto musíme v niektorých prípadoch skúšať, čo vlastne znamenajú príslušné kolonky.



Obrázok 39.5: Vloženie tabuľky

Ďalšie možnosti tohto modulu budeme rozoberať podľa farieb tlačidiel. Prvé je červené tlačidlo „F“. Jeho význam je podobný ako pri vkladaní vzorcov s tým rozdielom, že k označenej definícii vzorca pridá ešte aj zmenu veľkosti na 10:

size 10 {sum from {x={1}} to {infinity} {1 over x}}

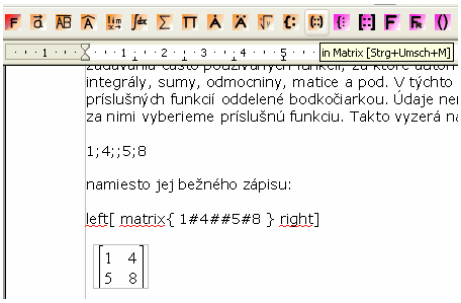
Teraz sa budeme venovať oranžovým tlačidlám. Tieto funkcie sú určené pre zrýchlené zadávania často používaných funkcií, za ktoré autori tohto modulu považujú vek-

tory, limity, integrály, sumy, odmocniny, matice a pod. V týchto prípadoch stačí, ak zadáme parametre príslušných funkcií oddelené bodkočiarkou. Údaje nemusíme ani označovať, ale stačí, ak tesne za nimi vyberieme príslušnú funkciu. Takto vyzerá napríklad zjednodušená definícia matice:

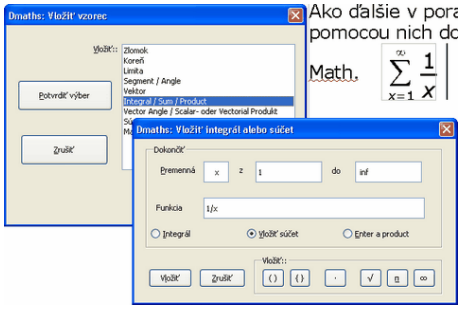
1; 4;; 5; 8

namiesto jej bežného zápisu:

left[matrix{1#4##5#8} right]

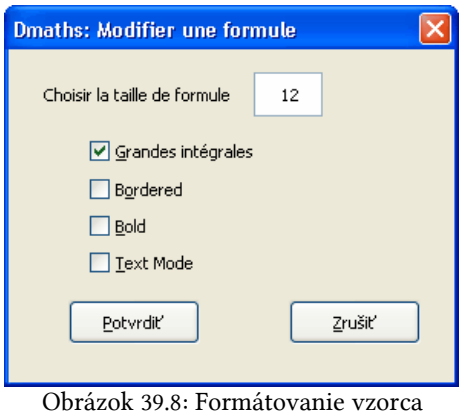


Obrázok 39.6: Jednoduché vloženie matice



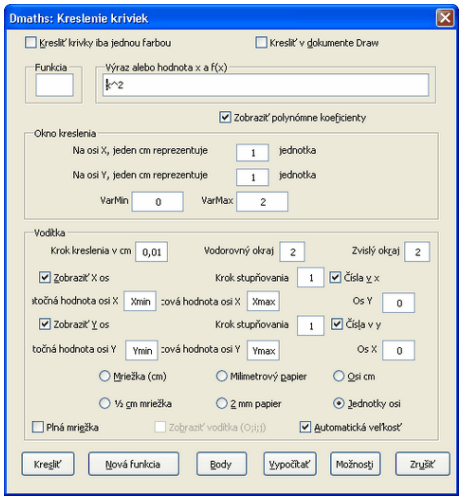
Obrázok 39.7: Vloženie vzorca bez potreby jeho definície

Ako ďalšie v poradí nájdeme fialové tlačidlá. Tieto sú určené najmä pre začiatočníkov, pretože pomocou nich dokážeme zadávať vzorce bez akejkoľvek znalosti (resp. iba so základnou znalosťou používania zloženej zátvorky pre zoskupovanie) skriptovacieho jazyka modulu Math.



Obrázok 39.8: Formátovanie vzorca

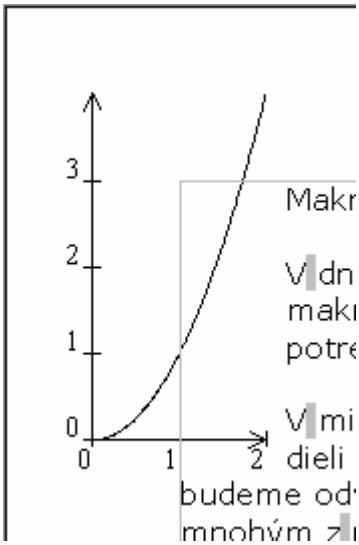
Tmavomodré tlačidlá sú určené pre zmenu vybraného textu (nie vzorcov). Po označení textu ho ho môžeme vložiť do obyčajných alebo zložených zátvoriek, zmeniť ho na kurzívu alebo uložiť do rámca.



Obrázok 39.9: Definovanie grafu funkcie

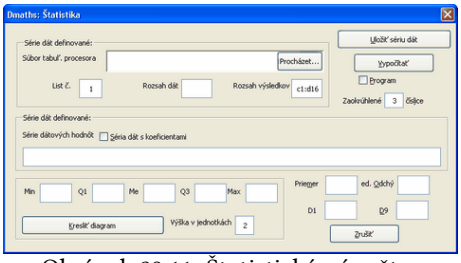
Zelené tlačidlá sú určené pre formátovanie už vloženého vzorca. Môžeme zmeniť jeho veľkosť, rez, resp. ho orámovať.

Svetlomodré tlačidlá ukrývajú možnosti, ktoré nám bežne OpenOffice.org (a nielen on) neposkytuje. Pomocou nich totiž môžeme kresliť grafy funkcií, vkladať milimetrový papier, prevádzať štatistické výpočty atď. Upozorňujem, že graf sa vždy nakreslí do ľavého horného rohu strany a preto ho musíme následne presunúť na to miesto, kde ho potrebujeme umiestniť.



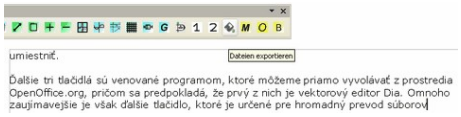
Obrázok 39.10: Vložený graf funkcie

Ďalšie tri tlačidlá sú venované programom, ktoré môžeme priamo vyvolávať z prostredia OpenOffice.org, pričom sa predpokladá, že prvý z nich je vektorový editor Dia.



Obrázok 39.11: Štatistické výpočty

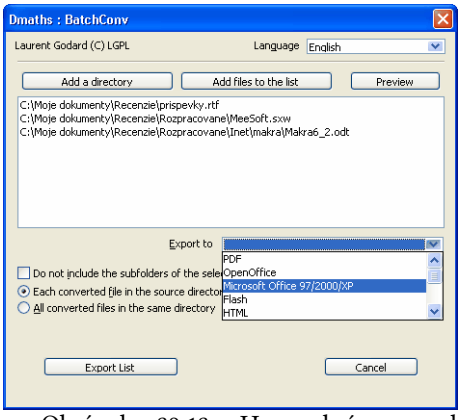
Omnoho zaujímavejšie je však ďalšie tlačidlo, ktoré je určené pre hromadný prevod súborov. Program dokáže naraz konvertovať súbory z rôznych vstupných formátov do jednotného výstupného formátu. Pravdaže, vstupné formáty musia zodpovedať súborom rovnakého druhu, t.j. textové do textových a pod. Modul podporuje prevod do týchto formátov: PDF, OpenOffice.org, Microsoft Office 97/2000/XP, Flash, HTML, Fichier Texte, RTF, Microsoft Word 95, Microsoft Winword 6.0, Microsoft Excel 95, Microsoft Excel 5.0, JPG, PNG, TIFF, SVG, EPS.



Obrázok 39.12: Tlačidlo pre hromadnú konverziu súborov

Nakoniec sa v paneli nástrojov nachádzajú tlačidlá pre nastavenie pracovných parametrov a zálohovanie používateľských adresárov OpenOffice.org. Týmto funkciám sa už nebudeme špeciálne venovať.

Na záver už iba uveďme, že makromodul Dmaths je šírený pod licenciou GNU General Public Licence. Úplné znenie licencie si môžeme prečítať tak, že si otvoríme nový súbor, napíšeme text „copie“ a stlačíme tlačidlo F3.



Obrázok 39.13: Hromadný prevod súborov

Internet: www.dmaths.com

40. Počítanie znakov

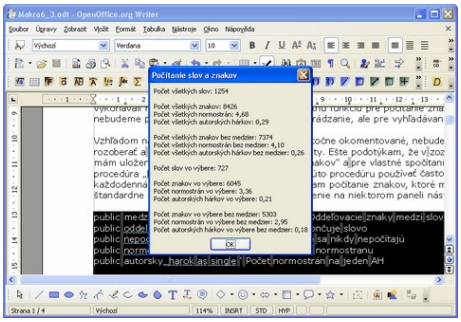
Ani vám nepostačujú štandardné možnosti, ktoré ponúka OpenOffice.org pri počítaní znakov? Dnes si ukážeme makro, pomocou ktorého dokážeme vypočítať o.i. napr. aj počet normostrán.

Tí, ktorí prispievajú do rôznych časopisov (papierových či internetových), alebo píšu romány sa ešte stále stretávajú s tým, že sú odmeňovaní podľa počtu normostrán, napísaných znakov ale bez medzier a pod. Podobne, pri rôznych súťažiach je požadované, aby napr. román mal minimálne toľko a toľko normostrán. Žiaľ, OpenOffice.org nám ponúka v bežnej štatistike iba základné údaje o počte znakov a vyššie uvedené možnosti nemá.

Okrem toho niekedy (napr. pri rozdeľovaní dlhého textu na viacero kratších pokračovaní) potrebujeme spočítať uvedené veci iba pre vybranú časť textu, resp. naraz aj pre celý text aj pre označený text. Preto si dnes uvedieme makro, ktoré nám dokáže spočítať počet slov, počet všetkých znakov, počet všetkých znakov bez medzier, počet normostrán zo všetkých znakov a počet autorských hárkov z normostrán a to naraz ako pre celý dokument, tak aj pre vybranú časť textu (pokiaľ nejakú označíme).

Hoci by sme pre vlastné počítanie niektorých znakov (v našom prípade to budú medzery) mohli teoreticky použiť aj náhradu znaku za ten istý znak, je to nepraktické (zbytočne by sme vykonávali náhradu). Preto si zadefinujeme vlastnú funkciu pre počítanie znakov, kde nebudeme používať štandardnú metódu pre nahrádzanie, ale pre vyhľadávanie.

Vzhľadom na to, že makrá sú, ako obvykle, čiastočne okomentované, nebudeme ich už viac rozoberať a uvedieme si priamo ich zdrojové texty. Ešte podotýkam, že v zozname makier ich mám uložené v osobitnom module „Pocitanie_znakov“ a pre vlastné spočítanie sa volá procedúra „Pocitanie_znakov“. Tím, ktorí budú túto procedúru používať často (u mňa je to každodenná náležitosť a vlastne vôbec nepoužívam počítanie znakov, ktoré má OpenOffice.org štandardne k dispozícii), odporúčam jej umiestnenie na niektorom paneli nástrojov.



Obrázok 40.1: Počítanie znakov

```
public medzery, medzery_regularne as string ' Oddeľovacie
znaky medzi slovami
public oddelovac_slov as string ' Znak, ktorý ukončuje
slovo
public nepocitane_znaky as string ' Znaky, ktoré sa nikdy
nepočítajú
public normo_strana as single ' Počet znakov na normostranu
public autorsky_harok as single ' Počet normostrán na jeden
AH

dim Dokument, Hladaj, vyber as object

REM Inicializácia premenných
sub init_znaky

 medzery=chr(&H20)+chr(&HA0)+chr(&H09) ' Space, Hard space,
Tab
 ' Definícia medzier pre vyhľadávanie pomocou regulárnych
výrazov
 medzery_regularne="("+chr(&H20)+"|"+chr(&HA0)+"|"+chr(&H09
)+")"

 oddelovac_slov=medzery+chr(&H0A)+chr(&H0D) ' Medzery, CR,
LF
 ' Napr. Frídek-Místek sú dve slova, preto tu je aj pomlčka
' alebo lomka a bodka napr. pre http odkazy
 oddelovac_slov=oddelovac_slov+"-"+"/"+"."
 ' alebo lomka a bodka napr. pre http odkazy

 nepocitane_znaky=chr(&H0A)+chr(&H0D) ' CR, LF
```

```
' Ak sa zmení norma, alebo niekto potrebuje tieto údaje
počítať inak, môže si zmeniť tieto čísla
 normo_strana=1800 ' Počet znakov na normostranu je 1800
 autorsky_harok=16 ' Počet normostrán na autorský hárok je
16

 Dokument=ThisComponent
 Hladaj=Dokument.createSearchDescriptor()
 vyber=dokument.getCurrentSelection() ' aktuálny vyber

end sub

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
' Vyhľadanie prvého výskytu hľadaného reťazca
 nasiel = Dokument.findFirst(Hladaj)
' Pokiaľ sa reťazec našiel, tak
 Do While NOT isNull(nasiel)
  if len(nasiel.String)>0 then
   kolko=kolko+1
  end if
  ' Vyhľadanie ďalšieho výskytu hľadaného rečazca
  nasiel = Dokument.findNext(nasiel.End, Hladaj)
 Loop
 Spocitaj_hladanim=kolko
end function

 rem Procedúra pre spočítanie znakov a slov v celom
dokumente a vo výberoch
Sub Pocitanie_znakov

 init_znaky

 ' Štatistické údaje o celom dokumente
 dim vsetky_znaky, vsetky_slova, vsetky_znaky_bez as long
 dim vsetky_normo, vsetky_normo_bez, vsetky_AH,
vsetky_AH_bez as single
 ' Štatistické údaje o výbere
 dim vyber_pocet, vyber_znaky, vyber_znaky_bez, vyber_slova
as long
 dim vyber_normo, vyber_normo_bez, vyber_AH, vyber_AH_bez
```

```
as single

 dim riadok as string
 dim vyber_ktory, dlzka_riadku, i as long
 dim slovo as boolean
 dim znak, CR, real_format as string

 CR=chr(&H0D)
 ' Formát pre zobrazenie počtu nájdených a vypočítaných
údajov (na dve desatinné miesta)
 real_format= "##########0.00"
 ' Údaje pre celý dokument
 ' Celkový počet slov
 vsetky_slova=dokument.WordCount

 ' Celkový počet znakov
 vsetky_znaky=dokument.CharacterCount
 ' Celkový počet znakov bez medzier
 vsetky_znaky_bez=vsetky_znaky-
Spocitaj_Hladanim(medzery_regularne, true, false, false)

 ' Výpočet normostrán, AH a to s medzerami aj bez nich pre
celý dokument
 vsetky_normo=vsetky_znaky/normo_strana ' počet normostrán
 vsetky_normo_bez=vsetky_znaky_bez/normo_strana ' počet
normostrán bez medzier
 vsetky_AH=vsetky_normo/autorsky_harok ' počet autorských
hárkov
 vsetky_AH_bez=vsetky_normo_bez/autorsky_harok ' počet AH
bez medzier

 ' Údaje pre vybrané časti textu

 ' Počet výberov (naraz môžeme označiť viacero nezávislých
častí textu)
 vyber_pocet=vyber.getCount()

 vyber_slova=0 ' počet slov
 vyber_znaky=0 ' počet znakov s medzerami
 vyber_znaky_bez=0 ' počet znakov bez medzier

' Pre všetky výbery
 for vyber_ktory=0 to vyber_pocet-1
  ' Text príslušného výberu
  riadok = vyber.getByIndex(vyber_ktory).getString()
  dlzka_riadku=len(riadok)

  ' Tvorba slova pre počítanie ich počtu
  slovo=false ' slovo neobsahuje ešte žiadny znak
```

```
  ' Pre všetky znaky z príslušného výberu
  for i=1 to dlzka_riadku
   znak=mid(riadok, i,1) ' spracovávame i-ty znak z textu
   if instr(oddelovac_slov, znak)=0 then
    ' znak nie je oddeľovač slov, t.j. je časťou slova
    slovo=true ' slovo určite obsahuje nejaký znak
   Elseif slovo then
     ' znak je oddeľovač slova a slovo obsahovalo nejaký
znak - zvyšuje sa počet slov
    vyber_slova=vyber_slova+1
    ' Slovo bolo ukončené - znovu začíname tvoriť ďalšie
    slovo=false
   end if

   if instr(nepocitane_znaky, znak)=0 then
    ' znak sa nenachádza v zozname nepočítaných znakov,
t.j. počítame ho
    vyber_znaky=vyber_znaky+1
    if instr(medzery, znak)=0 then
     ' znak nie je medzera, počítame ho teda aj medzi
všetky ostatné znaky
     vyber_znaky_bez=vyber_znaky_bez+1
    end if
   end if
  next i
 next vyber_ktory

 ' ak sme na konci posledného slova nemali oddeľovač,
 ' tak toto slovo ešte nemáme započítané v celkovom počte
slov
 if slovo then
  vyber_slova=vyber_slova+1
 end if

 ' Výpočet normostrán, AH a to s medzerami aj bez nich pre
vybranú časť dokumentu
 vyber_normo=vyber_znaky/normo_strana
 vyber_normo_bez=vyber_znaky_bez/normo_strana
 vyber_AH=vyber_normo/autorsky_harok
 vyber_AH_bez=vyber_normo_bez/autorsky_harok

' Využijeme premennú "znak" na vytvorenie výstupného
formulára

 znak="Počet všetkých slov: "+vsetky_slova+CR+CR
 znak=znak+"Počet všetkých znakov: "+vsetky_znaky+CR
 znak=znak+"Počet všetkých normostrán:
"+format(vsetky_normo, real_format)+CR
 znak=znak+"Počet všetkých autorských hárkov:
```

```
"+format(vsetky_AH, real_format)+CR+CR

 znak=znak+"Počet všetkých znakov bez medzier:
"+vsetky_znaky_bez+CR
 znak=znak+"Počet všetkých normostrán bez medzier:
"+format(vsetky_normo_bez, real_format)+CR
 znak=znak+"Počet všetkých autorských hárkov bez medzier:
"+format(vsetky_AH_bez, real_format)+CR+CR+CR

 znak=znak+"Počet slov vo výbere: "+vyber_slova+CR+CR

 znak=znak+"Počet znakov vo výbere: "+vyber_znaky+CR
 znak=znak+"Počet normostrán vo výbere:
"+format(vyber_normo, real_format)+CR
 znak=znak+"Počet autorských hárkov vo výbere:
"+format(vyber_AH, real_format)+CR+CR

 znak=znak+"Počet znakov vo výbere bez medzier:
"+vyber_znaky_bez+CR
 znak=znak+"Počet normostrán vo výbere bez medzier:
"+format(vyber_normo_bez, real_format)+CR
 znak=znak+"Počet autorských hárkov vo výbere bez medzier:
"+format(vyber_AH_bez, real_format)

 msgbox(znak, 0,"Počítanie slov a znakov")

End Sub
```

41. Drobnosti, ktoré potešia

V dnešnom pokračovaní seriálu o programovaní makier v OpenOffice.org si uvedieme rôzne naozaj drobné makrá, ktoré nám pomôžu pri formátovaní dokumentu.

V doterajších dieloch sme si o. i. uviedli makrá, pomocou ktorých sme vkladali nezalomiteľné medzery za jednoznakové predložky, akademické tituly, do vnútra napr. telefónnych čísiel a pod. V bežnej praxi sa však mnohokrát stretávame aj s ďalšími potrebami v tejto oblasti. Napríklad, pokiaľ chceme použiť úzku sadzbu, potom nám nezalomiteľné medzery naopak prekážajú a potrebujeme ich nahradiť zalomiteľnými.

S týmto istým (alebo podobným) problémom sa môžeme stretnúť napr. aj pri potrebe zaslania rukopisu, keď si vydavateľ žiada, aby sme v texte nezalomiteľné medzery nepoužívali, pretože ich prípadné vloženie (t.j. zalomenie textu) si robí sám pomocou iných nástrojov.

Podobná problematika vzniká napr. aj vtedy, keď potrebujeme spracovať text, ktorý bol pripravený napr. v textovom procesore MS Word. Tento program vkladá nezalomiteľné medzery automaticky, čo nám však nemusí vyhovovať, pretože mi ich chceme mať napr. iba tam, kde sme si to nastavili v našom pracovnom profile. To je aj prípad, keď musíme upravovať text, ktorý pripravil niekto iný.

Osobne sa stretávam aj s ďalším problémom – potrebujem nahradiť úplne všetky

medzery sa nezalomiteľné. Jedná sa napr. aj o príspevky do tohto seriálu o programovaní makier – konkrétne o zdrojové texty. Problém je vlastne v tom, že potrebujem dosiahnuť, aby zdrojové texty boli odsadené podľa štruktúry príkazov. Pokiaľ tam nechám obyčajné medzery (ktoré sa v zdrojovom texte automaticky nachádzajú po jeho skopírovaní z modulu Basic), tieto mi redakčný systém na začiatku riadku „odreže“. Preto ich musím nahradiť medzerami nezalomiteľnými.

Pravdaže, zároveň chcem, aby jeden programový riadok bol aj zobrazený ako jeden riadok a preto musím nahradiť zalomiteľné medzery za nezalomiteľné aj vo vnútri vlastných príkazov. Samozrejme, tí používatelia, ktorí si prípadne skopírujú takto uvedený zdrojový text musia počítať s tým, že im modul Basic následne vyhlási chybu (t.j. nebudú priamo fungovať) a musia si ho opraviť. Je to však seriál o programovaní a nie o kopírovaní hotových zdrojových textov, takže si každý môže dať tú námahu, aby si ich aspoň prepísal...

Ako posledná problematika, ktorej sa budeme dnes venovať je problematika rôznych nezalomiteľných medzier. V predchádzajúcich dieloch sme si uviedli dva typy – bežnú nezalomiteľnú medzeru (znak „Non Break Space“) a tzv. pružnú nezalomiteľnú medzeru (znak „Word Joiner“). Pri bežnej práci sa nám môžu hodiť makrá, ktoré dokážu automaticky zamedziť vzájomne tieto znaky, t.j. podľa potreby môžeme previesť jeden typ na druhý a opačne. Pravdaže, keď sme už spomínali zámenu všetkých medzier za nezalomiteľné medzery, aj v tých makrách uvedieme obidve tieto možnosti.

Tak, ako v predchádzajúcich dieloch, vlastné makrá nebudeme osobitne popisovať, pretože sú (podľa môjho názoru) dostatočne okomentované. Iba pripomeňme, že v nich používame funkcie, ktoré sme už uviedli pri formátovaní dokumentu (konkrétne funkcie „Init“ a „Vymen“), pri ktorých budeme predpokladať, že sú uložené v module „Formatovanie_funkcie“ – t.j. voláme ich Formatovanie_funkcie.init a Formatovanie_funkcie.Vymen. Dnes uvedené funkcie si uložíme do modulu „Rozne“.

```
REM Makro pre nahradenie nezalomiteľných medzier
zalomiteľnými
REM Musíme nahradiť ako znak NBSP - hexadecimálny kód A0,
REM tak aj znak WJ - hexadecimálny kód 2060
function f_Zalomitelna_Medzera as long
 Formatovanie_funkcie.init
 f_Zalomitelna_Medzera=Formatovanie_funkcie.Vymen(chr$
(&HA0)," ", false, false, false)+_
  Formatovanie_funkcie.Vymen(chr$(&H2060),"", false, false,
false)
end function

REM Makro pre nahradenie zalomiteľných medzier
nezalomiteľnými medzerami NBSP
function f_Nezalomitelna_Medzera_NBSP as long
 Formatovanie_funkcie.init
 f_Nezalomitelna_Medzera_NBSP=Formatovanie_funkcie.Vymen("
", chr$(&HA0), false, false, false)
end function
```

```
REM Makro pre nahradenie zalomiteľných medzier
nezalomiteľnými medzerami WJ
function f_Nezalomitelna_Medzera_WJ as long
 Formatovanie_funkcie.init
 f_Nezalomitelna_Medzera_WJ=Formatovanie_funkcie.Vymen(" ",
chr$(&H2060)+" ", false, false, false)
end function

REM Makro pre zámenu NBSP za WJ
function f_NBSP_za_WJ as long
 Formatovanie_funkcie.init
 f_NBSP_za_WJ=Formatovanie_funkcie.Vymen(chr$(&HA0), chr$
(&H2060)+" ", false, false, false)
end function

REM Makro pre zámenu WJ za NBSP
function f_WJ_za_NBSP as long
 Formatovanie_funkcie.init
 f_WJ_za_NBSP=Formatovanie_funkcie.Vymen(chr$(&H2060)+" ",
chr$(&HA0), false, false, false)
end function

REM Funkcie môžeme volať aj priamo, ale vtedy nám nevypíše
počet zámen.
REM Ak chceme vidieť aj tento počet, musíme si na to
naprogramovať procedúry.

sub Zalomitelna_Medzera
 ' Nahradenie nezalomiteľných medzier zalomiteľnými
 msgbox("Zmenených "+f_Zalomitelna_Medzera+"
nezalomiteľných medzier.", 0,"Zalomiteľné medzery")
end sub

sub Nezalomitelna_Medzera_NBSP
 ' Nahradenie zalomiteľných medzier nezalomiteľnými
medzerami NBSP
 msgbox("Zmenených "+f_Nezalomitelna_Medzera_NBSP+"
zalomiteľných medzier.", 0,"Nezalomiteľné medzery NBSP")
end sub

sub Nezalomitelna_Medzera_WJ
 ' Nahradenie zalomiteľných medzier nezalomiteľnými
medzerami WJ
 msgbox("Zmenených "+f_Nezalomitelna_Medzera_WJ+"
zalomiteľných medzier.", 0,"Nezalomiteľné medzery WJ")
end sub

sub NBSP_za_WJ
 ' Zámena NBSP za WJ
```

```
 msgbox("Zmenených "+f_NBSP_za_WJ+" NBSP za WJ.", 0,"Zámena
NBSP za WJ")
end sub

sub WJ_za_NBSP
 ' Zámena WJ za NBSP
 msgbox("Zmenených "+f_WJ_za_NBSP+" WJ za NBSP.", 0,"Zámena
WJ za NBSP")
end sub
```

42. Čarujeme s koncami odsekov

V ďalšom pokračovaní seriálu o programovaní makier v OpenOffice.org si naprogramujeme dve podobné makrá, pomocou ktorých budeme upravovať konce odsekov.

Pri písaní textu bežne používame viacero odsekov, t.j. odsek ukončujeme klávesou „Enter“. V niektorých prípadoch to však nie je žiadúce, pretože potrebujeme, aby sme v dokumente vlastne nemali klasické odseky, ale aby sa celý dokument správal v skutočnosti ako jediný odsek (t.j. ako keby sme namiesto klávesy „Enter“ stláčali kombináciu „Shift + Enter“, ktorá predstavuje zalomenie riadku).

Samozrejme, môžete sa opýtať, v akých prípadoch to niekto vôbec môže potrebovať. Odpoveď je veľmi jednoduchá – napr. aj ja pri písaní príspevkov na Inet.sk. Je to tak preto, lebo redakčný systém, pomocou ktorého sa príspevky vkladajú, pri klasickom ukončení odseku vkladá iné odsadenie, ako pri zalomení riadku. Pri obyčajnom texte by to ešte nebolo na prekážku, ale pri vkladaní zdrojových textov programov by to dopadlo priam katastrofálne, pretože by sa tento nedal prakticky nastaviť ako osobitný štýl (ten totiž aplikuje redakčný systém na označený text, pričom však rešpektuje aj zadané odseky, takže každý riadok by bol uvedený v osobitnom rámiku).

V makrách pre formátovanie dokumentu sme si o. i. uviedli makro pre odstraňovanie prázdnych odsekov. A tu sa znovu dostávame k opačnému problému – niekedy potrebujeme mať prázdne odseky medzi ostatnými odsekmi. Pre príklad takejto potreby nemusíme chodiť ďaleko. Znovu sa vrátime k problematike uverejňovania textov na internete, kde ich jednoducho inak nedokážeme opticky oddeliť (t.j. príslušný redakčný systém musí medzi ne vložiť dvakrát ukončenie riadku –

). Pokiaľ však máme text pripravený tak, že odseky sme si odsadzovali iba nastavením vzdialenosti medzi nimi, je to pre takéto (ale nielen pre takéto) spracovanie naozaj nepostačujúce.

Dosť však bolo teórie, uveďme si teraz vlastné makrá pre upravovanie odsekov. Na tomto mieste musíme upozorniť, že vzhľadom na možnosti, ktoré nám ponúka aplikačné programovacie rozhranie OpenOffice.org sa tieto zmeny dajú previesť iba v označenom texte a preto, ak ich chceme aplikovať na celý dokument, musíme si ho najprv celý označiť (napr. klávesovou skratkou „Ctrl + A“).

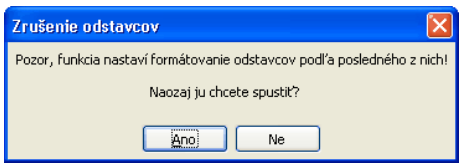
Ešte upozornime, že makrá pre úpravu odsekov menia aj formát celého označeného textu na formát posledného odseku. Táto vlastnosť je automatická a programovo by sa riešila naozaj veľmi obtiažne (pokiaľ by sa vôbec dala vyriešiť). Vzhľadom na to sa v makrách vôbec nerieši a preto s ňou musíme počítať ešte pred tým, ako makro spustíme. Z uvedeného dôvodu som do makier pridal aj upozornenie s možnosťou jeho ukončenia bez prevedenia týchto zmien.

Preto, pokiaľ chcete v texte používať rôzne formáty textu, najprv spustite tieto makrá a až následne meňte formáty textu. Pokiaľ vám to nevyhovuje, nezostáva vám nič iné, ako príslušné zmeny previesť manuálne, čo je však pri dlhých dokumentoch časovo naozaj náročná činnosť. Na druhej strane práve táto vlastnosť môže mnohým vyhovovať – jednoduchým spôsobom si práve pomocou tohto makra môžu zmeniť formát celého dokumentu, napr. v prípade, že spracovávali podklady od viacerých autorov resp. s rôznych zdrojov.

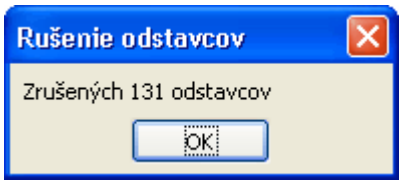
Ak náhodou nepotrebujú ich ďalšie vlastnosti (ako je vloženie prázdnych odsekov či zmenu koncov odsekov na zalomenie riadku), môžu v ďalších krokoch tieto zmeny vrátiť späť. Na to môže slúžiť napr. formátovanie dokumentu, kde nájdeme možnosť odstránenia prázdnych odsekov, alebo v prípade zalomenia riadku použiť náhradu reťazca „\n“ za reťazec „\n“ s využitím regulárnych výrazov (v budúcom dieli si na to uvedieme krátke makro).

```
REM Makro pre zámenu odsekov za line break (odstráni znaky
CR z textu)
REM Makro funguje iba v označenom (vybranom) texte
Sub Zrusenie_odsekov
 ' Na konci odsekov je kombinácia znakov CR+LF,
 ' na konci riadkov (ručné zalomenie - shift+enter) iba
znak LF.
 ' Preto zrušenie odsekov spočíva vlastne v odstránení
znakov CR.

 dim dokument, vyber as object
 dim vyber_ktory, dlzka_riadku, i as long
 dim pocet_CR as long
 dim riadok as string
 dim znak, CR, slovo as string
 dim zamena as boolean
 pocet_CR=0
 CR=chr(&H0D)
 ' Na začiatku upozorníme, že makro mení formát odsekov
 slovo="Pozor, funkcia nastaví formátovanie odsekov podľa
posledného z nich!"+CR+CR
 slovo=slovo+"
     Naozaj ju chcete spustiť?"+CR+CR
 ooo=msgbox(slovo,4,"Zrušenie odsekov") ' Dialógové okno s
tlačidlami „Áno/Nie“

 if ooo=6 then
  ' Stlačené tlačidlo "ÁNO"

  Dokument=ThisComponent
  vyber=dokument.getCurrentSelection() ' aktuálny výber
  vyber_pocet=vyber.getCount() ' Počet výberov

  for vyber_ktory=0 to vyber_pocet-1
   riadok = vyber.getByIndex(vyber_ktory).getString() '
```

```
reťazec z výberu
   dlzka_riadku=len(riadok)
   zamena=false

   rem pre všetky znaky z príslušného výberu
   for i=1 to dlzka_riadku
    znak=mid(riadok,i,1) ' spracovávame i-ty znak z výberu
    if instr(CR,znak)<>0 then
     ' znak je CR, odstránime ho a spočítame počet
odstránených odsekov
     mid(riadok,i,1,"") ' Nájdený znak CR nahradíme prázdnym
znakom
     dlzka_riadku=len(riadok)
     pocet_CR=pocet_CR+1 ' Pripočítame práve odstránený
znak CR, t.j. odstránený odsek
     zamena=true ' Indikácia, že sme odstránili znak CR
    end if
   next i
   if zamena then
    ' Ak sme našli znak CR (t.j. koniec odseku), zameníme
starý text za nový, čím odstránime odsek
    vyber.getByIndex(vyber_ktory).setString(riadok)
   end if
  next vyber_ktory
 end if
 ' Na konci si vypíšeme počet odstránených odsekov
 msgbox("Zrušených "+pocet_CR+" odsekov",0,"Rušenie
odsekov")

End Sub
```



Obrázok 42.1: Upozornenie, že dôjde k preformátovaniu odsekov

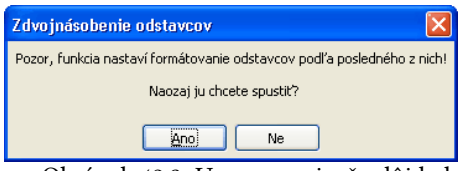


Obrázok 42.2: Oznámenie o počte zrušených odsekov

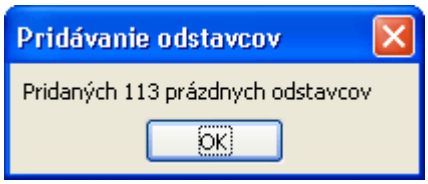
```
REM Makro pre pridanie prázdnych odsekov za každý iný odsek
REM Makro funguje iba v označenom (vybranom) texte
Sub Zdvojnasobenie_odsekov
 ' Na konci odsekov je kombinácia znakov CR+LF,
 ' preto za znakom LF pridáme ďalšiu kombináciu CR+LF.
 dim dokument, vyber as object
 dim vyber_ktory, dlzka_riadku, i as long

 dim pocet_CR as long
 dim riadok as string
```

```
 dim znak, CR, LF, slovo as string
 dim zamena as boolean

 pocet_CR=0
 CR=chr(&H0D)
 LF=chr(&H0A)

 ' Na začiatku upozorníme, že makro mení formát odsekov
 slovo="Pozor, funkcia nastaví formátovanie odsekov podľa
posledného z nich!"+CR+CR
 slovo=slovo+"
     Naozaj ju chcete spustiť?"+CR+CR
 ooo=msgbox(slovo,4,"Zdvojnásobenie odsekov") ' Dialógové
okno s tlačidlami „Áno/Nie“

 if ooo=6 then
  ' Stlačené tlačidlo "ÁNO"

  Dokument=ThisComponent
  vyber=dokument.getCurrentSelection() ' aktuálny vyber
  vyber_pocet=vyber.getCount() ' Počet výberov

  for vyber_ktory=0 to vyber_pocet-1
   riadok = vyber.getByIndex(vyber_ktory).getString() '
reťazec z výberu
   dlzka_riadku=len(riadok)
   zamena=false
   rem pre všetky znaky z výberu
   for i=1 to dlzka_riadku
    znak=mid(riadok,i,1) ' spracovávame i-ty znak z výberu
    if instr(LF,znak)<>0 then
     ' znak je LF, nahradíme ho LF+CR+LF čím pridáme odsek
a spočítame počet zámen
     mid(riadok,i,1,LF+CR+LF)
     ' Posunieme index o dva vložené znaky (CR+LF)
     i=i+2
     dlzka_riadku=len(riadok)
     pocet_CR=pocet_CR+1 ' Počet vložených prázdnych odsekov
     zamena=true ' Indikácia vloženia prázdneho odseku
    end if
   next i
   if zamena then
    ' Do textu sme vložili prázdny odsek, preto nahradíme
starý text za nový, čím vlastne prevedieme vlastnú úpravu
    vyber.getByIndex(vyber_ktory).setString(riadok)
   end if
  next vyber_ktory
 end if
 ' Na konci si vypíšeme počet vložených prázdnych odsekov
```

```
 msgbox("Pridaných "+pocet_CR+" prázdnych
odsekov",0,"Pridávanie odsekov")
End Sub
```



Obrázok 42.3: Upozornenie, že dôjde k preformátovaniu odsekov



Obrázok 42.4: Oznámenie o počte vložených odsekov

43. Rôzne zámeny

V dnešnom pokračovaní seriálu o programovaní makier v OpenOffice.org si znovu uvedieme makrá, pomocou ktorých dokážeme formátovať dokument a to konkrétne v oblasti zámen postupnosti znakov za typografické znaky.

Pri písaní textu bežne používame tri bodky pre označenie pokračovania textu (niekedy so skrytým podtextom). V typografii sa však pri sádzaní textu nepoužívajú tri bodky, ale špeciálny znak, ktorý nazvem „trojbodka“ (prosím, keby ma jazykovedci nekameňovali za tento názov, pre bežných používateľov je však dosť názorný). Hoci sa dá v OpenOffice.org nastaviť, aby sa tri bodky automaticky nahradzovali týmto znakom, nedôjde k tomu v tom prípade, ak nasledujú tesne za predchádzajúcim slovom (t.j. nie je medzi nimi a slovom medzera), čo je z gramatického hľadiska veľmi častý prípad. Tak isto k tejto zmene nedôjde vtedy, ak po ich napísaní nestlačíme medzeru či klávesu „Enter“ (napr. na konci textu).

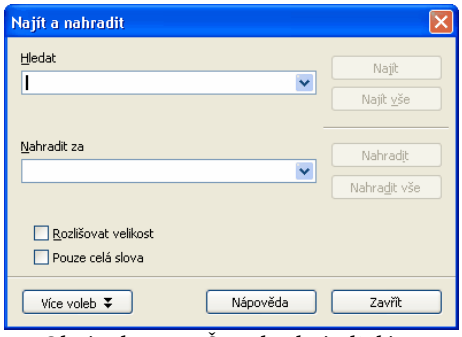
Podobná problematika sa týka aj spojovníkov, pomlčiek a rozdeľovníkov. Z typografického hľadiska majú všetky tieto znaky rôzne dĺžky – spojovník a rozdeľovník je v skutočnosti kratšia čiarka, ktorá nemá medzeru medzi sebou a slovom, kým pomlčka je dlhšia a pred a aj za ňou sa nachádza medzera. OpenOffice.org automaticky dokáže upraviť spojovník na pomlčku (pre ktorú vlastne nemáme priamu klávesu) v prípade, ak sa nachádza medzi medzerami. Pokiaľ však pomlčku píšeme za text, ktorý je ukončený bodkou (napr. nejaká skratka), potom túto zámenu automaticky neprevedie. Pravdaže, tento problém dokážeme riešiť aj tým, že si zadefinujeme do automatických opráv napr. náhradu dvoch spojovníkov za pomlčku. Toto riešenie však nemusí vyhovovať každému a preto je vhodné, aby sme spolu so zámenou troch bodiek za „trojbodku“ naprogramovali aj zámenu spojovníka za pomlčku, ak sa nachádza medzi medzerami.

A aby sme nezabudli na sľub, ktorý sme si dali minule, uvedieme si dnes aj makro, pomocou ktorého dokážeme zameniť ručné zalomenie riadku (klávesová skratka „Shift + Enter“) za zalomenie odseku (klávesa „Enter“). Podobne, ako sme to robili aj v minulosti, aj teraz použijeme makrá, ktoré sme si už doteraz naprogramovali a ktoré sa nachádzajú v module „Formatovanie_funkcie“.

```
REM Makro pre rôzne zámeny (napr. troch bodiek za
„trojbodku“)
function f_Rozne_zameny as long
```

```
 dim V_com(), Co(), Za() as string
 dim n as long
 dim kolko as long

 Formatovanie_funkcie.init
 ' Budeme zamieňať tri bodky za trojbodku
 ' a obyčajné pomlčky za dlhú pomlčku medzi dvomi
medzerami.
 ' Pri hľadaní použijeme regulárne výrazy, lebo spojovníky
môžu začínať a končiť
 ' ľubovoľnou medzerou - preto je definícia vyhľadávacích
reťazcov trochu komplikovanejšia

 V_com()=array("\.\.\.","[:space:]+-[:space:]+")
 Co()=array("...","-") ' tri bodky a spojovník
 Za()=array("…","-",) ' „trojbodka“ a pomlčka

 kolko=0
 for n=lbound(V_com()) to ubound(V_com())
  kolko=kolko+Formatovanie_funkcie.Vymen_hladanim(V_Com(n),
Co(n),Za(n),true,false,false)
 next n
 f_Rozne_zameny=kolko

end function

REM Pokiaľ si chceme vypísať počet zámen, môžeme volať
predchádzajúcu funkciu touto procedúrou
sub Rozne_zameny
 msgbox("Prevedených "+f_Rozne_zameny+" zámen.",0,"Rôzne
zámeny")
end sub

REM Makro pre zámenu zalomenia riadku za koniec odseku
function f_Koniec_odseku as long
 Formatovanie_funkcie.init
 f_Koniec_odseku
=Formatovanie_funkcie.Vymen("\n","\n",true,false,false)
end function

REM Pokiaľ si chceme vypísať počet zámen koncov riadkov za
odseky
sub Koniec_odseku
 msgbox("Prevedených "+f_Koniec_odseku+" zámen.",0,"Zámena
konca riadku za odsek")
end sub
```

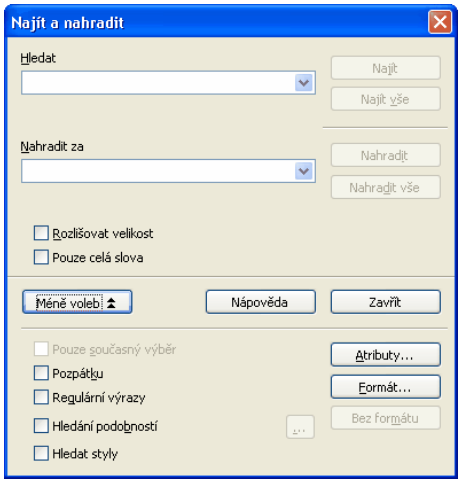
44. Vlastný dialóg pre zámeny



Obrázok 44.1: Štandardný dialóg pre zámeny

V ďalšom pokračovaní seriálu o programovaní makier v OpenOffice.org si vytvoríme vlastný dialóg pre hľadanie a nahrádzanie textu.

Pri praktickej práci s OpenOffice.org často používame dialógové okno „Nájsť a nahradiť“ (klávesová skratka „Ctrl + F“). Mnohým však nevyhovuje to, že napríklad zaškrtávacie políčko pre regulárne výrazy sa zobrazí až po stlačení tlačidla „Viac volieb“, pričom si OpenOffice.org nepamätá ani jeho predchádzajúce nastavenie. Toto platí pre každé otvorenie tohto dialógového okna, takže ak často používame práve regulárne výrazy, je táto práca veľmi spomalená.

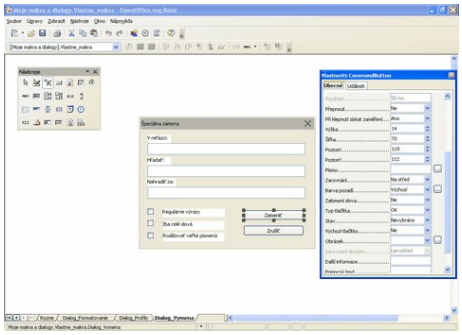


Obrázok 44.2: Rozšírené možnosti štandardného dialógu pre zámeny

Preto si teraz naprogramujeme vlastné dialógové okno pre nahrádzanie textu. V doterajších dieloch sme si naprogramovali dve vlastné funkcie, pomocou ktorých sme nahrádzali jeden reťazec druhým. Konkrétne ide o funkcie „Vymen“ a „Vymen_hladanim“, ktoré sme si uložili do modulu makier „Formatovanie_funkcie“.

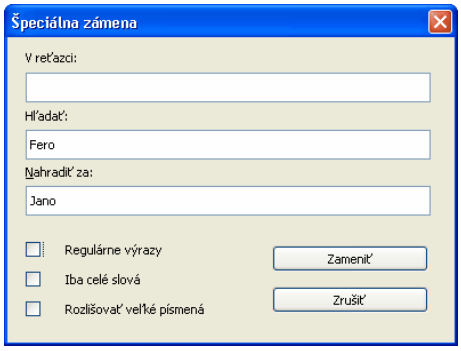
V stručnosti si pripomeňme, že funkcia „Vymen“ využíva štandardné služby pre zámenu reťazcov s tým, že môžeme nastaviť tri logické parametre pre vyhľadávanie – regulárne výrazy, celé slová a rozlišovanie veľkosti písmen. Funkcia „Vymen_hladanim“ je už zložitejšia, pretože ako parametre nemáme dva, ale tri reťazce. Je to tak preto, lebo v tejto funkcii zamieňame celý nájdený reťazec, ale iba jeho vnútornú časť. Pravdaže, aj tu máme možnosť použiť už spomenuté tri logické parametre pre vyhľadávanie.

Tieto funkcie sa nám preto priamo ponúkajú na to, aby sme si pripravili vlastný dialóg pre vyhľadávanie. Dialóg nazveme „Dialog_Zamena“ a musí obsahovať aspoň tieto položky: tri textové polia, ktoré nazveme „V_retazci“, „Hladat“ a „Nahradit_za“; tri zaškrtávacie políčka – „Regularne_vyrazy“, „Cele_slova“ a „Velke_pismena“; dve tlačidlá – „Zamenit“, ktorému priradíme typ tlačidla „OK“ a „Zrusit“, ktorému priradíme typ „Cancel“ („Zrušiť“). Okrem toho si jednotlivé položky doplníme vhodnými textovými popismi.



Obrázok 44.3: Návrh vlastného dialógu pre zámeny

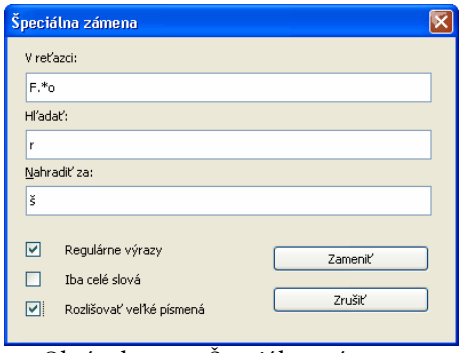
Vlastnej tvorbe dialógu sa už, samozrejme, nebudeme venovať, pretože na to boli určené predchádzajúce diely, kde sme sa tomu podrobne venovali. Teraz nám nezostáva už nič iné, iba naprogramovať vlastné makrá pre obsluhu tohto dialógu s následným volaním už spomínaných vyhľadávacích funkcií.



Obrázok 44.4: Obyčajná zámena pomocou vlastného dialógu

Aby sme nemuseli zadávať špeciálne zaškrtávacie políčko pre rozlíšenie situácie, kedy chceme využiť funkciu „Vymen“ a kedy funkciu „Vymen_hladanim“, rozlíšime to automaticky podľa toho, či zadáme text do dialógovej textovej premennej V“_retazci“. Ak áno, tak budeme používať funkciu „Vymen_hladanim“, ak nie, tak funkciu „Vymen“. Napokon, z vlastného makra je dosť jasné a prípadne si toto môžete uviesť ako doplňujúcu textovú informáciu aj do vytváraného dialógu.

Aby sa zadané hodnoty a parametre pamätali aj pri nových spusteniach tohto dialógu, príslušné premenné zadefinujeme ako typ GLOBAL. Pretože OpenOffice.org automaticky nastavuje premenné pri ich deklarácii na nulové hodnoty, nemusíme sa starať o to, aké hodnoty budú mať pred prvým spustením. Musíme však počítať s tým, že ak náhodou toto nastavenie nebude v niektorej verzii alebo na niektorom operačnom systéme fungovať, tak pri pri prvom spustení uvidíme náhodné hodnoty.



Obrázok 44.5: Špeciálna zámena pomocou vlastného dialógu

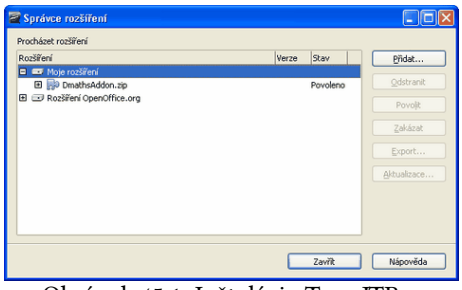
Makrá si pre lepšiu orientáciu uložíme do modulu „Specialna_zamena“. Pre vlastné vyvolanie s pustenie dialógu so zámenami voláme procedúru „Specialne_zamen“, pričom na konci si vypíšeme, koľko výmen bolo urobených.

```
REM Premenné si zadefinujeme ako global, aby sa pamätali aj
pri ďalšom spustení
REM Pozor! Pri prvom spustení môžu obsahovať náhodné
veličiny
global zamen_V_retazci, zamen_Hladat, zamen_Nahradit_za as
string
global zamen_Regularne_vyrazy, zamen_Cele_slova,
zamen_Velke_pismena as boolean

sub Init_dialog_zameny(oDialog as object)
 REM Inicializácia prvkov dialógu

 oDialog.model.V_retazci.Text = zamen_V_retazci
 oDialog.model.Hladat.Text = zamen_Hladat
 oDialog.model.Nahradit_za.Text = zamen_Nahradit_za
 if zamen_Regularne_vyrazy then
  oDialog.model.Regularne_vyrazy.State=1
 else
  oDialog.model.Regularne_vyrazy.State=0
 end if
```

```
  if zamen_Cele_slova then
  oDialog.model.Cele_slova.State=1
 else
  oDialog.model.Cele_slova.State=0
 end if
 if zamen_Velke_pismena then
  oDialog.model.Velke_pismena.State=1
 else
  oDialog.model.Velke_pismena.State=0
 end if

end sub

sub vykonaj_zamenu(oDialog as object)
 REM Vlastná zámena podľa zadaných hodnôt v dialógu

 dim kolko as long

 REM Príprava globálnych premenných
 zamen_V_retazci = oDialog.model.V_retazci.Text
 zamen_Hladat = oDialog.model.Hladat.Text
 zamen_Nahradit_za = oDialog.model.Nahradit_za.Text
 zamen_Regularne_vyrazy =
oDialog.model.Regularne_vyrazy.State=1
 zamen_Cele_slova = oDialog.model.Cele_slova.State=1
 zamen_Velke_pismena = oDialog.model.Velke_pismena.State=1

 REM Príprava dokumentu pre zámenu
 Formatovanie_funkcie.Init
 if zamen_V_retazci="" then
  REM Ak nezadáme reťazec "V_retazci", zamieňame priamo
  kolko=Formatovanie_funkcie.Vymen(zamen_Hladat,
zamen_Nahradit_za, zamen_Regularne_vyrazy,
zamen_Cele_slova, zamen_Velke_pismena)
 else
  REM Ak zadáme reťazec "V_retazci", zanieňame špeciálne
  kolko=Formatovanie_funkcie.Vymen_hladanim(zamen_V_retazci
, zamen_Hladat, zamen_Nahradit_za, zamen_Regularne_vyrazy,
zamen_Cele_slova, zamen_Velke_pismena)
 end if
 msgbox("Prevedných "+kolko+" zámien.", 0,"Špeciálne
zámeny")
end sub

sub Specialne_zamen

 dim oDialog as object

 oDialog=CreateUnoDialog(DialogLibraries.Vlastne_makra.Dial
```

```
og_Zamena)
 Init_dialog_zameny(oDialog)

 if oDialog.Execute()=1 then
  vykonaj_zamenu(oDialog)
 end if
 oDialog.dispose()

end sub
```

45. Rozšírenie pre typografické úpravy textu – TypoJTB

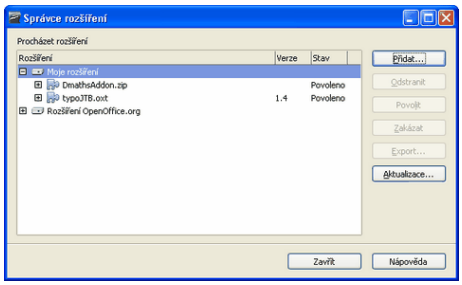


Obrázok 45.1: Inštalácia TypoJTB

V doterajších dieloch sme sa venovali predovšetkým úpravám textu podľa typografických pravidiel. Predstavme si preto makromodul TypoJTB z dielne Tomáša Bílka, ktorý je venovaný práve tejto problematike.

Ako sme povedali už v prvom dieli tejto časti seriálu o programovaní makier pre OpenOffice.org, predstavíme si aj niekoľko zaujímavých makier, makromodulov (či lepšie povedané rozšírení) od iných autorov. Prvou lastovičkou v tejto oblasti bol francúzsky makromodul Dmaths a dnes to bude české rozšírenie TypoJTB.

Toto rozšírenie je distribuované vo formáte „OXT“ a do prostredia sa inštaluje cez správcu rozšírení (postupnosť volieb menu „Nástroje – Správca rozšírení“). Tam si vyberieme časť, kde chceme rozšírenie nainštalovať (t.j. či pre seba, alebo pre celý OpenOffice.org), stlačíme tlačidlo „Pridať“ a po vyhľadaní otvoríme príslušný súbor (v tomto prípade „typoJTB.oxt“). Rozšírenie sa následne automaticky nainštaluje. Pre jeho spustenie je potrebné zavrieť OpenOffice.org a znovu ho spustiť, pretože údaje o rozšíreniach sa načítavajú pri spúšťaní OpenOffice.org.



Obrázok 45.2: Zoznam rozšírení po nainštalovaní TypoJTB

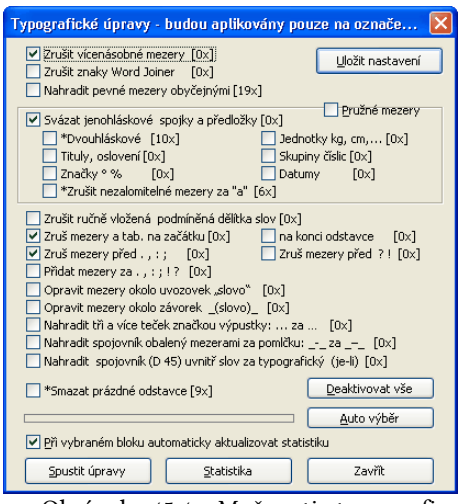
Hoci to sem priamo nepatrí, uveďme si ešte, že v prípade, ak chceme nejaké rozšírenie odinštalovať, urobíme to tak isto v správcovi rozšírení stlačením tlačidla „odstrániť“. Pokiaľ chceme, môžeme niektoré rozšírenia dočasne zakázať a znovu povoliť – príslušné funkcie sú dosť jasné z možností, ktoré vidíme v tomto správcovi. Pozor, všetky zmeny sa prejavia, podobne ako pri inštalácii, až po opätovnom spustení OpenOffice.org.

Obrázok 45.3: Lišta TypoJTB

Po nainštalovaní sa v prostredí OpenOffice.org zobrazí nová nástrojová lišta TypoJTB, kde sa nachádzajú ikony sprístupňujúce jeho všetky funkcie. Tieto môžeme rozdeliť do troch častí (takto sú aj usporiadané na nástrojovej lište) – úpravy textu,

informácie (štatistika) a pomocné funkcie. Pozrime sa teraz na ne podrobnejšie.

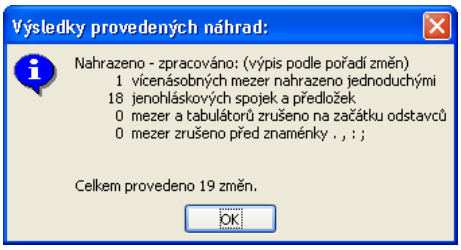
45.1. Typografické úpravy



Obrázok 45.4: Možnosti typografických úprav

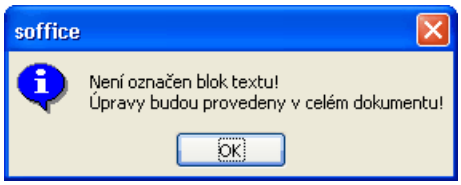
Táto funkcia určite patrí medzi tie najzaujímavejšie, pretože práve jej niektorým možnostiam sme sa venovali aj my pri programovaní makier. Umožňuje úpravy, ako je mazanie viacnásobných medzier, vloženie nezalomiteľných medzier za jednoznakové predložky a spojky, zámenu troch bodiek za znak výpustky (doteraz sme tento znak označovali ako „trojbodka“), opraviť medzery okolo interpunkčných znamienok (zrušiť medzery pred a pridať medzery za nimi) atď.

Makro umožňuje jednotlivé nastavenia uložiť, takže pri budúcom spustení ich nemusíme nastavovať znovu. Momentálne ešte makro neumožňuje vytvárať používateľské profily, ale autorovi som už navrhol túto možnosť, takže je možné, že táto funkcia bude do tejto časti implementovaná, čo by určite mnohí používatelia privítali.



Obrázok 45.5: Oznam o počte prevedených úprav

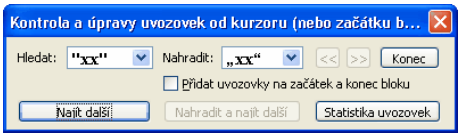
Veľmi zaujímavé a potrebné je aj to, že pri tejto funkcii sa automaticky rozoznáva, či máme označený nejaký text, alebo nie (o čom nás aj informuje). Pokiaľ ho máme označený, prevedú sa zmeny iba v tejto časti, pokiaľ nemáme vybranú žiadnu časť, prevedú sa automaticky v celom dokumente.



Obrázok 45.6: Oznámenie, že nebol označený text

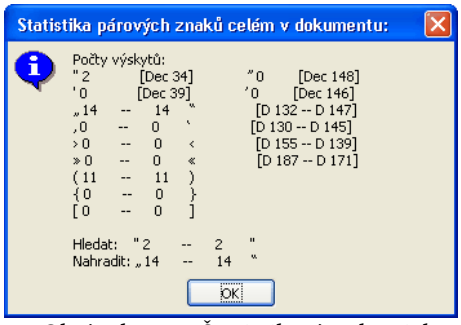
Pokiaľ po výbere funkcie „Typografické úpravy“ prevedieme tzv. štatistiku (pri označenom texte môžeme využiť aj možnosť jej automatickej prípravy), v hranatých zátvorkách, ktoré sa nachádzajú pri jednotlivých možnostiach uvidíme počet výskytov prípadných úprav. Teraz môžeme pomocou tlačidla „Auto výber“ automaticky označiť iba tie úpravy, kde máme nenulový výskyt príslušných úprav. Samozrejme, následným ručným nastavením môžeme zabrániť prevedeniu prípadných neželaných zmien. Pomocou tejto možnosti však môžeme urýchliť následnú úpravu, lebo sa zbytočne naprázdno nespúšťajú tie úpravy, ktoré de facto nie sú potrebné.

45.2. Kontrola úvodzoviek a zátvoriek



Obrázok 45.7: Kontrola úvodzoviek a zátvoriek

Druhá funkcia sa venuje úvodzovkám a zátvorkám, pričom umožňuje ich rôzne vzájomné nahrádzanie (dokonca aj úvodzovky za zátvorky či na-



Obrázok 45.8: Štatistika úvodzoviek a zátvoriek

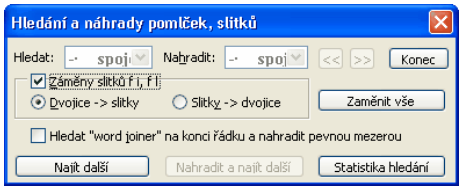
opak). Toto ocenia napr. tí, ktorí píšu zdrojové texty programov v prostredí OpenOffice.org, pričom práve napr. textové úvodzovky „“, ktorými OpenOffice.org automaticky ohraničuje text nie sú vhodné, pretože v programe potrebujú úvodzovky "". Počas programovania sa však nemusia zaoberať touto „drobnosťou“, pretože pomocou tejto funkcie na pár krokov dokážu urobiť všetky potrebné zmeny.

Táto funkcia funguje vlastne ako klasické vyhľadávanie s následnou možnosťou zámeny, pričom však vyhľadáva a nahrádza príslušné páry znakov bez ohľadu na text, ktorý sa nachádza vo vnútri. Ďalej tu nájdeme možnosť uzatvorenia označeného textu do úvodzoviek alebo zátvoriek. Výber znakov, ktorými sa ohraničí tento text prevádzame v časti „Nahradiť“ a vlastné ohraničenie vykonáme zaškrtnutím voľby „Pridať úvodzovky na začiatok a koniec bloku“ a následným potvrdením pomocou tlačidla „Pridať okolo bloku“.

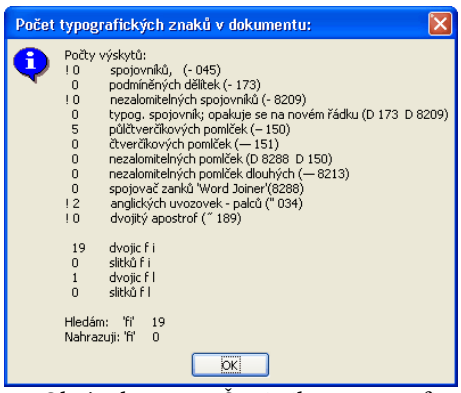
V rámci tejto funkcie je určite zaujímavá aj štatistika, kde nám makro vypíše všetky výskyty znakov, ktoré môžeme vyhľadávať a nahrádzať. Zároveň nám vypíše počet výskytov znakov „Hľadať“ a aj počet výskytov znakov „Nahradiť“, takže hneď vidíme, či musíme prípadnú zámenu vôbec spúšťať.

45.3. Hľadanie a nahradzovanie typografických znakov

Táto funkcia umožňuje vyhľadať dvojice znakov „fi“ a „fl“ a nahradiť ich príslušnými zliatkami, alebo naopak. Ďalej umožňuje nájsť a vzájomne nahradiť rôzne iné typografické značky (spojovník, typografický spojovník – t.j. taký, ktorý sa v prípade potreby opakuje na konci a začiatku riadku, rôzne pomlčky a pod.



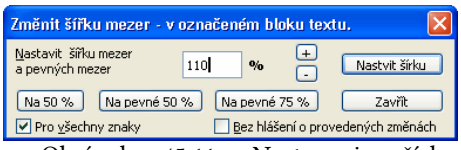
Obrázok 45.9: Hľadanie a nahradzovanie typografických znakov



Obrázok 45.10: Štatistika typografických znakov

Pokiaľ používate „pružnú“ nezalomiteľnú medzeru (t.j. znak „Word Joiner“), mohli ste sa už stretnúť s prípadom, kedy nefungovala úplne správne a spojenie, ktoré malo zostať na jednom riadku bolo rozdelené do dvoch riadkov. Táto funkcia dokáže nájsť tieto prípady a príslušné znaky „Word Joiner“ nahradí nezalomiteľnou medzerou, čím sa uvedené problémy odstránia. Podobne, ako pri predchádzajúcej funkcii, aj tu máme možnosť zobrazenia štatistiky, t.j. počtu výskytov znakov, ktoré môžeme v rámci tejto funkcie vyhľadávať a vzájomne zamieňať.

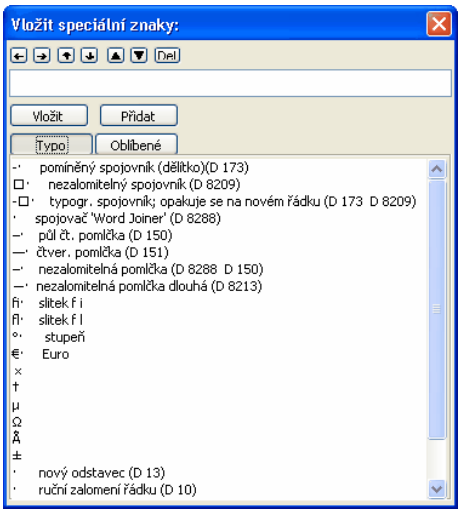
45.4. Nastavenie šírky medzier a znakov



Obrázok 45.11: Nastavenie šírky medzier a znakov

V poradí štvrtá funkcia, pomocou ktorej môžeme upravovať text, je zameraná na šírku písma. Umožňuje vo vybranej časti dokumentu nastavovať šírku medzier a prípadne aj ostatných znakov. Ak neoznačíme žiaden text, zmeny sa budú prevádzať v celom dokumente, ale iba pre medzery a aj to fungujú iba prednastavené hodnoty na troch tlačidlách – zmenšenie medzier na 50%, prevod obyčajných medzier na nedeliteľné (pevné) medzery a ich zmenšenie na 50% alebo 75%. Tieto tlačidlá však nemôžeme použiť v prípade, keď chceme zmeniť šírku všetkých znakov. Vtedy musíme príslušnú percentuálnu zmenu uviesť priamo zadaním jej číselnej hodnoty.

45.5. Vkladanie špeciálnych znakov



Obrázok 45.12: Vkladanie špeciálnych znakov

Hoci je vkladanie špeciálnych znakov prístupné aj cez menu („Vložiť – Špeciálny symbol“), táto možnosť je dosť nepraktická vtedy, ak často vkladáme iba pár znakov. A práve vtedy dokážeme výhodne použiť funkciu rozšírenia TypoJTB, kde nájdeme niekoľko týchto znakov preddefinovaných. Pracovné okno tejto funkcie je rozdelené do dvoch záložiek – „Typo“ a „Obľúbené“.

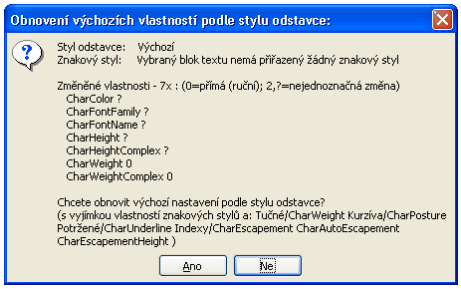
Ide v podstate o delenie čisto formálne, pretože do ktorejkoľvek z týchto záložiek si môžeme pridávať vlastné znaky či dokonca celé texty. Znaky môžeme zapisovať priamo, alebo zadaním ich hexadecimálneho alebo dekadického kódu. Ku každému znaku alebo textu môžeme pridať aj poznámku, ktorá sa bude zobrazovať pri ich popise, ale do vlastného textu sa nebude vkladať. Pravdaže, vložené (a aj preddefinované znaky) či texty, pokiaľ nám nevyhovujú alebo ich už nepoužívame, môžeme kedykoľvek vymazať. Vlastné vloženie potrebného znaku prevádzame nakoniec obyčajným dvojklikom.

45.6. Informácie o kódoch znakov

Obrázok 45.13: Informácie o kódoch znakov

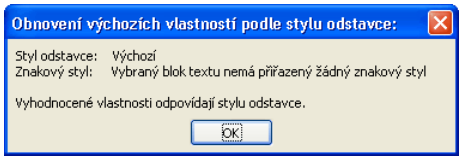
Pokiaľ potrebujete zistiť hexadecimálne a dekadické kódy znakov, stačí označiť príslušný text a spustiť túto funkciu. Následne sa vypíšu všetky vybrané znaky pod sebou s tým, že sú očíslované, výpis obsahuje dekadický a hexadecimálny kód a na koniec aj znak, ktorému tento kód prislúcha.

45.7. Rozdiely oproti štýlom



Obrázok 45.14: Obnovenie podľa pôvodného štýlu

Pokiaľ pri písaní zmeníme v texte nastavenia, ktoré nie sú zhodné s nastaveniami, ktoré sú definované v príslušnom použitom štýle, táto funkcia nám ich vypíše a zároveň nám ponúkne možnosť spätného nastavenia podľa príslušného štýlu. Túto funkciu ocenia najmä tí používatelia, ktorí musia dávať dohromady podklady z rôznych zdrojov, čím veľmi rýchlo a na jeden krok urobia všetky potrebné zmeny bez toho, aby museli pracne vyhľadávať, kde používateľ nesprávne manuálne zasiahol.

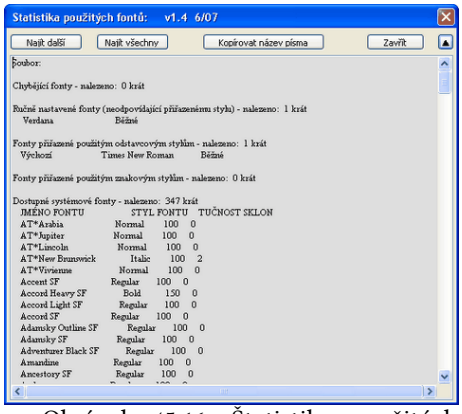


Obrázok 45.15: Oznámenie, že nemusíme robiť obnovenie

Pozor, funkcia spätného nastavenia funguje iba pre vybraný text (resp. znak, na ktorom sa práve nachádza kurzor), takže pokiaľ ju chceme aplikovať na celý dokument, musíme ho celý označiť. Pokiaľ sa žiadne zmeny nenájdu, program nám to oznámi jednoduchým výpisom a žiadne spätné úpravy (lebo nie je aké) nám neumožní vykonať.

45.8. Štatistika použitých fontov

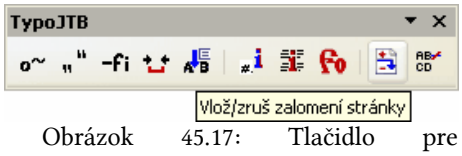
Posledná štatistická funkcia, ktorú nám rozšírenie TypoJTB ponúka, je štatistika použitých fontov. Táto funkcia je veľmi potrebná najmä pri prezeraní cudzích dokumentov, pretože nám povie nielen to, ktoré fonty sú v dokumente použité a aké fonty máme v systéme nainštalované, ale, čo je hádam najdôležitejšie, aj to, či sa v dokumente náhodou nepoužíva font, ktorý náš systém nepozná (t.j. ktorý by nám mohol spôsobovať problémy pri následnej tlači, exporte do PDF formátu a pod.).



Obrázok 45.16: Štatistika použitých fontov

45.9. Vloženie (zrušenie) zalomenia stránky

Táto funkcia vloží ručné zalomenie stránky za aktuálny odsek, resp. zruší toto zalomenie, ak sa pred ním nachádza. Pretože pri jej použití sa vôbec nemusíme premiestňovať na začiatok alebo koniec odseku, je toto vkladanie alebo mazanie naozaj veľmi pohodlné a značne uľahčuje prácu.



Obrázok 45.17: Tlačidlo pre vloženie/zrušenie zalomenia stránky

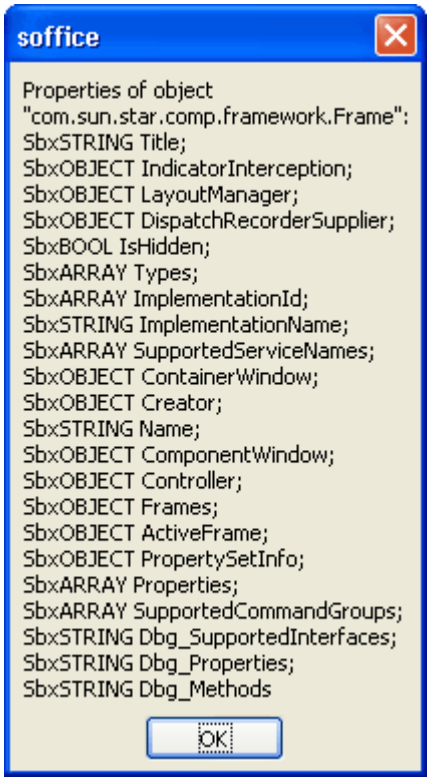
45.10. Nedeliť slovo

Obrázok 45.18: Tlačidlo pre nastavenie nedeliteľnosti slova

Posledná doplnková funkcia, ktorú nám rozšírenie TypoJTB poskytuje, je možnosť označenia slov, ktoré nechceme, aby sa delili. Makro túto situáciu rieši veľmi jednoducho – pre príslušné slovo, na ktorom sa nachádzame (alebo môžeme mať označených aj viacej slov) nastaví jazyk na hodnotu „žiadny“, takže pri následnom delení slov sa nemôže použiť príslušný slovník a slovo sa nerozdelí. Toto riešenie však prináša aj určitú nevýhodu – takéto slovo nepodlieha kontrole gramatiky, ale to je už „programátorská“ daň, ktorú musíme za túto možnosť „zaplatiť“.

Internet: www.volny.cz/macrojtb/typoJTB.html

46. Zdroje informácií

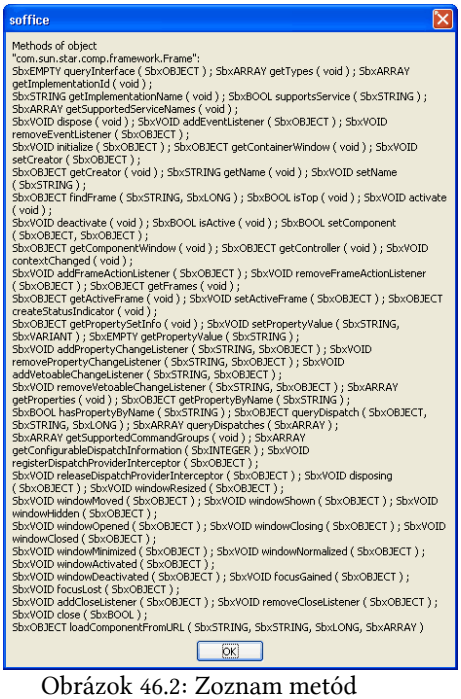


Obrázok 46.1: Zoznam vlastností

Chcete si naprogramovať nejaké makro, potrebujete zistiť premenné metódy, ktoré používa OpenOffice.org a neviete ako na to? Dnes si ukážeme niekoľko možností, ako to dokážete zistiť.

V predchádzajúcich 62 dieloch (áno, bolo ich presne 62) seriálu o programovaní makier sme si ukazovali rôzne makrá, pričom sme používali najrôznejšie premenné a metódy, ktoré nám ponúka OpenOffice.org. Samozrejme, mnohým už prišla na um otázka, ako a kde sa dajú získať podrobnejšie informácie.

46.1. Analýza premenných



Obrázok 46.2: Zoznam metód

Pokiaľ poznáme premennú, ktorú potrebujeme iba preskúmať, ponúka nám nástroj priamo jazyk StarOffice Basic. Objekty majú totiž v sebe zabudované aj dva reťazce, ktoré obsahujú základné informácie:

DBG_properties – reťazec, ktorý obsahuje všetky vlastnosti objektu

DBG_methods – reťazec, ktorý obsahuje všetky metódy objektu

Väčšina objektov má zabudovaný aj tretí reťazec DBG_supportedInterfaces, ktorý obsahuje in-

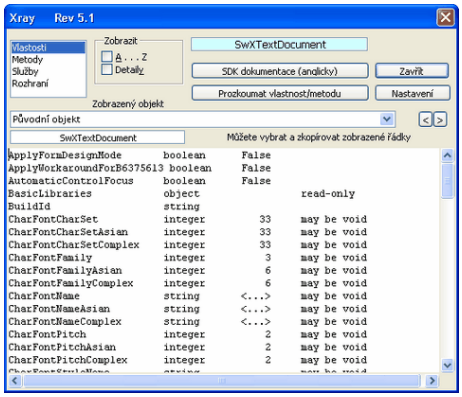
Obrázok 46.3: Zoznam rozhraní

formácie o rozhraniach, ktoré objekt používa.

Napríklad pomocou nasledujúceho makra dokážeme zistiť potrebné údaje pre objekt „ThisComponent.CurrentController.Frame“:

```
sub metody
 dim premenna as object
 premenna=ThisComponent.CurrentCon
troller.Frame

 msgbox premenna.DBG_properties
 msgbox premenna.DBG_methods
 msgbox
premenna.DBG_supportedInterfaces
end sub
```

Nevýhoda tohto riešenia je taká, že chýba akýkoľvek bližší popis a preto mnohokrát nezostáva nič iné, iba metódou pokusov a omylov zisťovať, na čo sa ktorá vlastnosť alebo metóda používa.

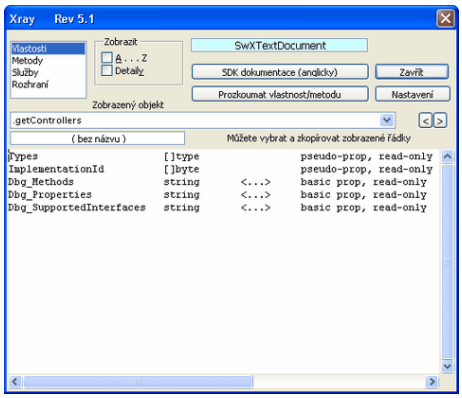


Obrázok 46.4: Makro XrayTool

Ako vhodnejší spôsob sa javí použiť už hotové makro XrayTool, ktoré nielenže dokáže vypisovať tieto údaje, ale ich vypisuje aj v omnoho prehľadnejšej forme. Makro je možné stiahnuť v češtine zo stránok Tomáša Bílka, autora minule predstaveného makra TypoJTB. Jeho inštaláciu si schválne nebudeme uvádzať, pretože už musíme predpokladať, že čitatelia tohto seriálu to dokážu vykonať sami.

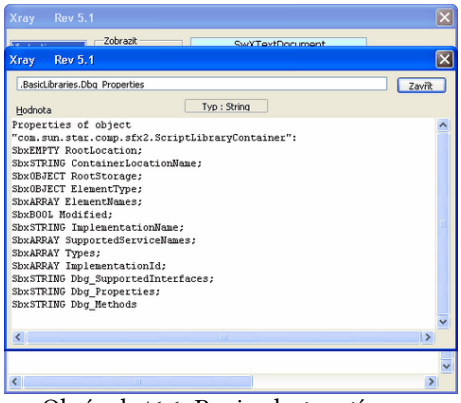
Po jeho nainštalovaní je potrebné ešte naprogramovať dve malé procedúry. V prvej budeme inicializovať makro XrayTool a v druhej zisťovať údaje o konkrétnej premennej. Osobne mám naprogramované tieto procedúry v nasledujúcej podobe (pre spúšťanie makra používam procedúru ZistiMetody):

```
sub LoadingLibrariesXrayTool
 BasicLibraries.LoadLibrary("XrayTo
ol")
End Sub

sub ZistiMetody
 LoadingLibrariesXrayTool
 dim premenna as object

 REM Nastav podľa potreby
```



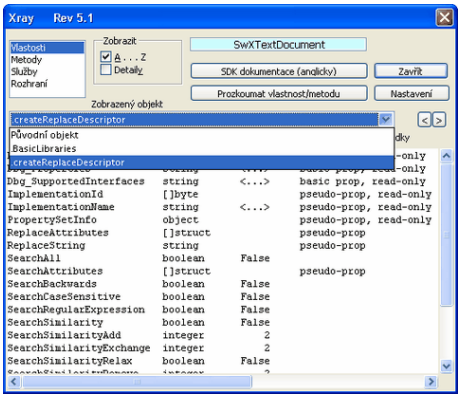
Obrázok 46.5: Prechádzanie po štruktúre objektovej premennej

```
premenna=ThisComponent

xray premenna
end sub
```



Obrázok 46.6: Popis vlastností

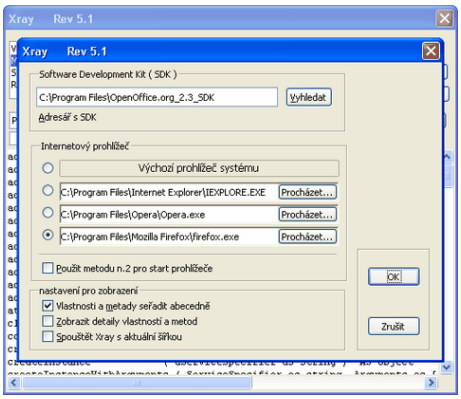
Pri takomto nastavení dokážeme zistiť údaje o všetkých premenných, pretože makro XrayTool umožňuje prechádzať po štruktúre. Pokiaľ sa presunieme na nejakú vlastnosť alebo metódu (stačí kliknúť na príslušnom riadku), pomocou tlačidla „Prozkoumat vlastnost/metodu“ sa nám zobrazia buď jej obsah, alebo vnútorná štruktúra, ktorú môžeme skúmať ďalej. Pravdaže, nájdeme tu aj už spomínané premenné DBG_properties, DBG_methods a DBG_supportedInterfaces.



Obrázok 46.7: Zoznam zobrazených objektov

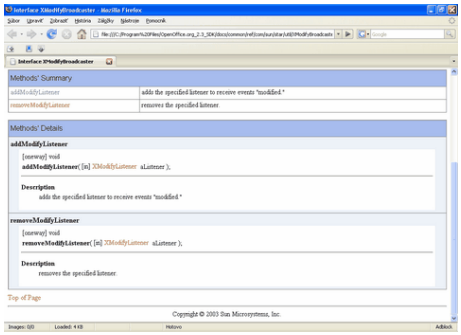
Pri vlastnom skúmaní máme možnosť prepínať medzi štyrmi údajmi – vlastnosťami, metódami, službami a rozhraním. Pokiaľ prechádzame po vnútornej štruktúre, príslušné premenné sa pamätajú v zozname zobrazených objektov, takže sa môžeme kedykoľvek medzi nimi rýchlo presúvať.

46.2. Dokumentácia



Obrázok 46.8: Nastavenie ciest a parametrov v XrayTool

To, čo je však na makre XrayTool najzaujímavejšie, je spolupráca s „Software Development Kit (SDK)“. Pokiaľ si stiahneme a nainštalujeme túto dokumentáciu napr. zo servera download.services.openoffice.org (žiaľ, je iba v angličtine), potom nám makro po stlačení tlačidla „SDK dokumentace (anglicky)“ dokáže zobraziť v nastavenom internetovom prehliadači informácie o príslušnej časti, pokiaľ sú k dispozícii (nie všetko sa v tejto dokumentácii nachádza). Nastavenie príslušných ciest (ako k inštalovanej dokumentácii, tak aj k internetovému prehliadaču) vykonáme cez tlačidlo „Nastavení“.



Obrázok 46.9: Zobrazené informácie z SDK

Okrem uvedených možností, môžeme pri programovaní použiť aj ďalšie dokumentácie, ktoré si môžeme stiahnuť z internetu. Okrem toho nám ako inšpirácia môžu poslúžiť aj iné makrá, kde nájdeme ukrytých nielen veľa dobrých myšlienok, ale

aj praktické použitie niektorých metód.

Na tomto mieste preto nesmieme zabudnúť na makrá od Andrewa Pitonyaka (www.pitonyak.org), ktoré sú naozaj na veľmi vysokej úrovni. Všetkým záujemcom odporúčam, aby si stiahli jeho dokumentáciu „Useful Macro Information For OpenOffice“, ktorá je viac než len malá programátorská príručka. Osobne som z tejto dokumentácie čerpal inšpiráciu pre on-line doplňanie nezalomiteľných medzier. Pravdaže, veľmi zaujímavá je aj kniha Andrewa Pitonyaka „OpenOffice.org Macros Explained“ či ostatné knihy o programovaní od tohto autora.

Ďalej nesmieme zabudnúť ani na základnú dokumentáciu k jazyku StarOffice Basic, ktorú na svojich stránkach docs.sun.com ponúka priamo firma SUN. Z osobných skúseností vám odporúčam stiahnuť si dokumentáciu pre StarOffice Basic od verzie 6 až po verziu 8. Prečo všetky tri? Preto, lebo počas vývoja OpenOffice.org už došlo k situácii, keď najprv platili veci, ktoré boli popísané v príručke pre „sedmičkovú“ verziu a potom sa vrátili k verzii „šestkovej“ (konkrétne sa jednalo o niektoré rozhrania pre dokumenty OpenOffice.org Calc).

Všetkým používateľom OpenOffice.org (t.j. nielen programátorom) odporúčam stránky wiki.services.openoffice.org/wiki/Documentation, kde nájdu podrobnú on-line dokumentáciu tohto produktu. Samozrejme, na svoje si prídu aj programátori, pretože časť „BASIC Guide“ je venovaná výhradne pre nich.

Týmto sme ukončili prvú veľkú časť seriálu o programovaní makier pre OpenOffice.org, v ktorej sme sa venovali predovšetkým modulu Writer. Nastal čas pre krátku dovolenku, aby si čitatelia mohli v pokoji stiahnuť a preštudovať odporučenú dokumentáciu a aby sme mohli načerpať nové inšpirácie pre pripravovanú druhú časť, ktorú budeme venovať ostatným modulom OpenOffice.org. Je pravda, že sme ani v rámci modulu Writer nespomenuli množstvo vecí (napr. makrá pre prácu s tabuľkami, makrá pre prácu so súbormi...), ale naozaj nie je možné uviesť všetko naraz.

Na úplný záver si však neodpustíme ešte jedno makro – a to makro, pomocou ktorého dokážeme spúšťať iné makro, ktorého názov mu odovzdáme ako parameter. Využitie tohto makra už ponecháme na láskavých čitateľov, ktorí si určite budú vedieť predstaviť napr. tabuľku, kde v jednej bunke zadajú názov a v druhej budú volať príslušné makro.

```
sub spusti_makro(nazov as string)
 dim document as object
 dim dispatcher as object

 document = ThisComponent.CurrentController.Frame
 dispatcher =
createUnoService("com.sun.star.frame.DispatchHelper")

 dispatcher.executeDispatch(document,
"vnd.sun.star.script:"+nazov+"?
language=Basic&location=application", "", 0, Array())

end sub
```

Ako konkrétny príklad použitia si uvedieme volanie makra pre počítanie znakov:

```
sub volanie_ineho_makra
 spusti_makro("Vlastne_makra.Pocitanie_znakov.Pocitanie_znak
ov")
end sub
```

46.3. Internet

XrayTool – www.volny.cz/macrojtb/index.html

Dokumentácia Adrewa Pitonyaka – www.pitonyak.org/AndrewMacro.odt

OpenOffice.org Macros Explained – www.pitonyak.org/OOME_3_0.odt

On-line dokumentácia OpenOffice.org – wiki.services.openoffice.org/wiki/Documentation

On-line dokumentácia OpenOffice.org Basic – wiki.services.openoffice.org/wiki/Documentation/BASIC_Guide

46.3.1. Dokumentácia SUN StarOffice:

StarOffice 6.0 Software Basic Programmer's Guide – 817-1826-10.pdf

StarOffice 7 Office Suite - Basic Programmer's Guide – 817-1826.pdf

StarOffice 8 Programming Guide for BASIC – 819-0439.pdf

On-line dokumentácia StarOffice 8 Programming Guide for BASIC – docs.sun.com/app/docs/doc/819-0439

Miš-maš makier

Aj naďalej budeme pokračovať v seriáli o makrách pre OpenOffice.org. Tento seriál však nebude vychádzať tak pravidelne, ako doteraz, ale jeho diely budú vznikať podľa potreby používateľov. V prvom dieli sa znovu vrátime k formátovaniu dokumentov, ktoré rozšírime o novú funkciu.

47. Oprava nesprávnych znakov WJ na konci riadku

V minulom roku sme ukončili pomerne rozsiahlu časť makier, ktoré sa venovali formátovaniu dokumentov. Okrem iného sme urobili makrá, ktoré dokázali vkladať do textu nezalomiteľné medzery za jednoznakové preložky a spojky, medzi akademické tituly a meno, medzi číslice a podobne.

Na základe požiadavok niektorých používateľov sme urobili aj verziu, pomocou ktorej sme dokázali vkladať tzv. pružné nezalomiteľné medzery. V podstate išlo o to, že sme pred obyčajnú medzeru vložili znak „Word Joiner“. Tento znak však nie vždy funguje tak, ako by mal – jednoducho niekedy oddelí predložku či spojku a následný text na dva riadky. Toto správanie je záhadou, ale je to tak.

Našťastie, aj na toto existuje riešenie, ktoré si dnes ukážeme. Spočíva v tom, že vyhľadáme, či nám na konci riadku „náhodou“ nezostal „trčať“ znak „Word Joiner“ aj s medzerou a ak áno, tak ho zameníme za obyčajnú nezalomiteľnú medzeru (znak NBSP).

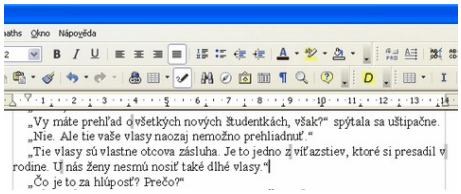
Pretože musíme hľadať konce riadkov, nezostáva nám nič iné, ako použiť viditeľný kurzor, kde je k dispozícii funkcia skočenia na koniec riadku. Pretože budeme prechádzať textom, bude pri vlastnej oprave pred našimi očami celý text rolovať. Pozor, pri dlhšom texte ma z toho trochu rozbolela hlava, preto odporúčam, aby ste priebeh sledovali iba občas.

Ako obvykle, vlastné makro je okomentované a preto nebudeme venovať osobitnú pozornosť príkazom, ktoré sú tam napísané.

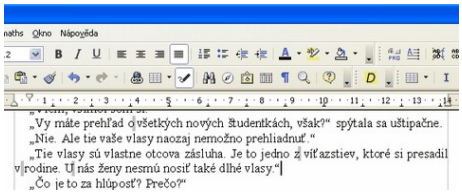
```
REM Procedúra hľadá nesprávne zalomenia WJ na konci riadku
REM a nahradí ich znakom NBSP
function WJ_za_NBSP_na_konci_riadku as long

 dim NBSP, WJ as string
 dim Kurzor, Povodny as object
 dim Koniec as boolean
 dim pom as string
 dim kolko as long

 kolko=0

 NBSP=chr$(&HA0)
 WJ=chr$(&H2060)

 kurzor= ThisComponent.currentcontroller.getViewCursor() '
viditeľný kurzor
```

Oprava nesprávnych znakov WJ na konci riadku

```
 Povodny=kurzor.getstart() ' Pôvodná poloha kurzoru – aby
sme sa mohli vrátiť po prerolovaní celého textu
 kurzor.collapsetostart() ' ak je niečo označené, tak
zrušiť výber, aby fungovala funkcia "skoč na koniec riadku"
 kurzor.gotoStart(false) ' Skočíme na začiatok dokumentu

 do ' Kým neprejdeme celý text, tak
  ' Označíme si tri znaky na konci riadku a ak posledné dva
sú WJ+medzera, tak je nesprávne zalomenie
  Kurzor.gotoEndOfLine(false) ' skočíme na koniec riadku
  Kurzor.goLeft(3, false)
  Kurzor.goRight(3, true) ' zvýraznenie posledných troch
znakov
  if right(Kurzor.string,2) = WJ+" " then
   ' Je to nesprávne zalomenie WJ+medzera
   pom=Kurzor.getstring() ' z posledných troch znakov
(písmeno+WJ+medzera)
   pom=left(pom,1)+NBSP ' písmeno+NBSP ' zoberieme iba prvé
písmeno a pridáme za to NBSP
   Kurzor.setstring(pom) ' Vlastná zámena
   kolko=kolko+1 ' počet zámen
  end if
  Kurzor.collapseToEnd() ' Zrušíme označený vymenený text
  Kurzor.gotoEndOfLine(false) ' Skočíme na koniec riadku
  Koniec=not(kurzor.goRight(1,false)) ' a posunieme sa o
jeden znak ďalej, ak to ide
 loop until Koniec ' Skončíme, ak sa už nemôžeme posunúť
 kurzor.gotoRange(Povodny,false) ' Vrátime sa na pôvodnú
polohu
 WJ_za_NBSP_na_konci_riadku=kolko
end function
```



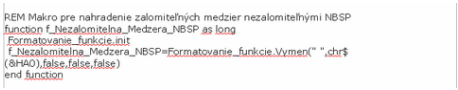
Obrázok 47.1: Nesprávne zalomený znak WJ na konci riadku



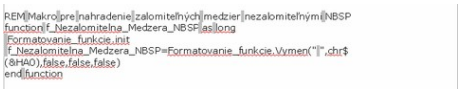
Obrázok 47.2: Vymenený znak WJ za NBSP

Pravdaže, v tejto súvislosti vzniká niekedy potreba drobných makier, pomocou ktorých dokážeme vzájomne konvertovať medzery medzi sebou. V nasledujúcom budeme predpokladať, že makrá, ktoré sme uviedli v minulom roku čiastočne používate – konkrétne funkciu „Vymen“ v module „ Formatovanie_funkcie“.

```
REM Makro pre nahradenie nezalomiteľných medzier
zalomiteľnými
function f_Zalomitelna_Medzera as long
 Formatovanie_funkcie.init
 f_Zalomitelna_Medzera=Formatovanie_funkcie.Vymen(chr$
(&HA0)," ",false,false,false)+_
  Formatovanie_funkcie.Vymen(chr$
(&H2060),"",false,false,false)
end function

REM Makro pre nahradenie zalomiteľných medzier
nezalomiteľnými NBSP
function f_Nezalomitelna_Medzera_NBSP as long
 Formatovanie_funkcie.init
 f_Nezalomitelna_Medzera_NBSP=Formatovanie_funkcie.Vymen("
",chr$(&HA0),false,false,false)
end function

REM Makro pre nahradenie zalomiteľných medzier
nezalomiteľnými WJ
function f_Nezalomitelna_Medzera_WJ as long
 Formatovanie_funkcie.init
 f_Nezalomitelna_Medzera_WJ=Formatovanie_funkcie.Vymen("
",chr$(&H2060)+" ",false,false,false)
end function

REM Makro pre zámenu NBSP za WJ
function f_NBSP_za_WJ as long
 Formatovanie_funkcie.init
 f_NBSP_za_WJ=Formatovanie_funkcie.Vymen(chr$(&HA0),chr$
(&H2060)+" ",false,false,false)
end function

REM Makro pre zámenu WJ za NBSP
function f_WJ_za_NBSP as long
 Formatovanie_funkcie.init
 f_WJ_za_NBSP=Formatovanie_funkcie.Vymen(chr$(&H2060)+"
",chr$(&HA0),false,false,false)
end function
```



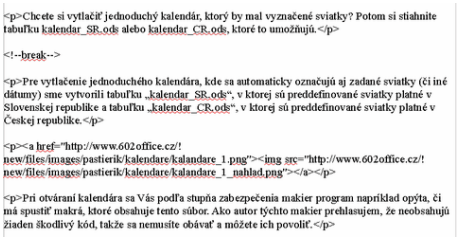
Obrázok 47.3: Text s obyčajnými medzerami



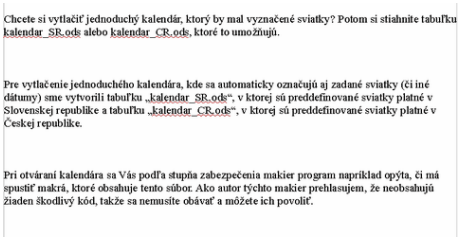
Obrázok 47.4: Zamenené medzery za NBSP

A úplne na záver si ako bonus uveďme jedno krátke makro, pomocou ktorého dokážeme rýchlo odstrániť HTML značky. Pozor, nejedná sa o prevod na text, ale iba o odstránenie príkazov, ktoré sú uzatvorené medzi znakmi „<“ a „>“. V mnohých prípadoch je však takýto prevod postačujúci a nie je problém, aby ste si, povedzme ako domácu úlohu, naprogramovali prevod kódov znakov na vlastné znaky.

```
REM Makro, ktoré vymaže príkazy HTML uzatvorené medzi <>
function f_Zrus_HTML as long
 Formatovanie_funkcie.init
 ' Pri hľadaní hľadá všetko, čo je uzatvorené v <>, ale vo
vnútri nesmú byť tieto znaky,
 ' lebo by označil aj text, ktorý je napr. pri internetových
odkazoch.
 f_Zrus_HTML=Formatovanie_funkcie.Vymen("<[^<^>]*>","",true
,false,false,false)
end function
```



Obrázok 47.5: Text v HTML kódovaní



Obrázok 47.6: Text po odstránení HTML značiek

48. Kopírovanie tabuľky Calc do tabuľky Writer

V dnešnom dieli seriálu o makrách si ukážeme, ako dokážeme vytvoriť tabuľku v textovom procesore Writer. Zároveň si ukážeme, ako do nej dokážeme skopírovať obsah zošitu z tabuľkového procesora Calc.

Pri tvorení dokumentov, v ktorých chceme prezentovať výsledky práce potrebujeme niekedy vkladať tabuľky. Hoci aj v textovom procesore Writer môžeme do nich vkladať niektoré vzorce, predsa len ich možnosti sú v tejto oblasti obmedzené. Vtedy prichádza do úvahy jediná možnosť – vytvoriť si tabuľku v zošite modulu Calc a nakoniec jej určitú časť preniesť do modulu Writer.

OpenOffice.org v tomto smere neposkytuje priame kopírovanie či dokonca vytvorenie textovej tabuľky, ktorej obsah by bol totožný s obsahom zošita v Calcu. Preto si naprogramujeme makro, ktoré to dokáže. Využijeme na to špeciálnu „schránku“, ktorú si zadefinujeme ako globálne premenné:

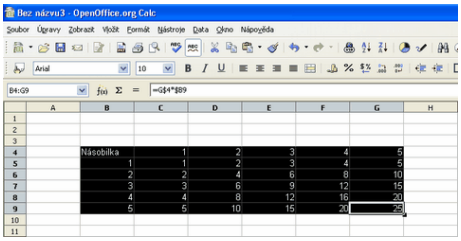
```
REM Makro pre kopírovanie tabuliek z Calcu do Writeru
REM Definícia globálnych premenných špeciálnej "schránky"
global bunky_z_calcu_do_writer(0,0)
global stlpce_z_calcu_do_writer, riadky_z_calcu_do_writer as
long
```

Pretože nikdy nechceme kopírovať celý zošit, vo funkcii pre skopírovanie obsahu z modulu Calc budeme pracovať iba s označenou časťou tabuľky:

```
REM Makro pre skopírovanie tabuľky z Calcu do špeciálnej
"schránky"
sub Kopiruj_tabulku_z_Calc_do_Writer
 dim dokument, bunka as object
 dim stlpec, riadok as long
 dim sprava as string
 REM Ak sme v tabuľkovom procesore, tak
 if
ThisComponent.supportsService("com.sun.star.sheet.Spreadshe
etDocument") then
  REM Nastavíme sa na označenú časť tabuľky
  Dokument=ThisComponent.getCurrentSelection()
  REM Zistíme rozsah označenej časti - počet stĺpcov a
riadkov
  stlpce_z_calcu_do_writer=dokument.getcolumns().Count
  riadky_z_calcu_do_writer=dokument.getrows().Count

  REM Ak je počet označených stĺpcov a riadkov nenulový,
tak
  if stlpce_z_calcu_do_writer<>0 and
riadky_z_calcu_do_writer<>0 then
   REM Zmeníme veľkosť "schránky" podľa aktuálneho výberu
   redim preserve
bunky_z_calcu_do_writer(riadky_z_calcu_do_writer,
stlpce_z_calcu_do_writer)

   REM Pre všetky stĺpce z výberu
   for stlpec=0 to stlpce_z_calcu_do_writer-1
    REM a pre všetky riadky z výberu
    for riadok=0 to riadky_z_calcu_do_writer-1
     REM Vyberieme bunku a skopírujeme jej obsah do
"schránky"
     bunka=dokument.getCellByPosition(stlpec,riadok)
     bunky_z_calcu_do_writer(riadok+1,stlpec+1)=bunka.getst
ring()
    next riadok
   next stlpec
  end if
  REM Vypíšeme, akú veľkú oblasť tabuľky sme skopírovali do
"schránky"
  sprava="Skopírovaných "+str(stlpce_z_calcu_do_writer)+"
stĺpcov a "+str(riadky_z_calcu_do_writer)+" riadkov"
  msgbox sprava
 end if
end sub
```

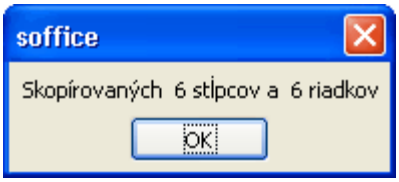
Kopírovanie tabuľky Calc do tabuľky Writer

Nakoniec si naprogramujeme funkciu pre vloženie obsahu špeciálnej „schránky“ do tabuľkového procesora. Tu musíme rozoznávať dve situácie. Prvá je, že sa nachádzame vo vnútri už existujúcej tabuľky – v tomto prípade prepíšeme jej obsah. Ak sa nenachádzame vo vnútri tabuľky, tak vytvoríme tabuľku novú s rozsahom, ktorý zodpovedá skopírovanej tabuľke z modulu Calc.



Obrázok 48.1: Výber časti tabuľky v module Calc pre kopírovanie

Ešte uveďme, že ak sa nachádzame v už existujúcej tabuľke, makro skopíruje tabuľku od ľavej hornej bunky bez ohľadu na to, kde sme sa predtým nachádzali:

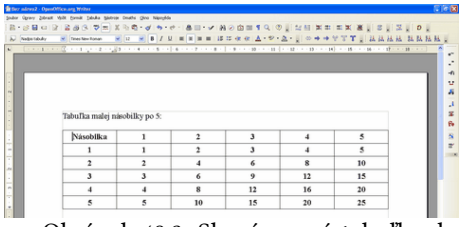


Obrázok 48.2: Výpis počtu odpamätaných buniek

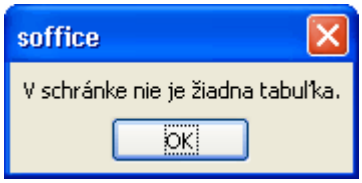
```
REM Makro pre skopírovanie tabuľky zo
špeciálnej "schránky" do Writeru
sub Vytvor_tabulku_z_Calc_do_Writer
 dim dokument, kurzor, tabulka, bunka
as object
 dim stlpec, riadok as long
 dim adresa as string

 REM Ak sme v textovom dokumente a veľkosť kopírovanej
tabuľky je nenulová, tak
 if
ThisComponent.supportsService("com.sun.star.text.TextDocume
nt") and _
  stlpce_z_calcu_do_writer<>0 and
riadky_z_calcu_do_writer<>0 then
  dokument=StarDesktop.CurrentComponent
  REM Nastavíme sa na aktuálne miesto v dokumente
  kurzor=ThisComponent.currentcontroller.getViewCursor()

  REM Ak nie sme v tabuľke, budeme vytvárať novú
  on error goto Nie_sme_v_tabulke

  REM Pokus o sprístupnenie tabuľky, v ktorej sa možno
nachádzame
  REM Ak nie sme v tabuľke, vedie to k chybe - uplatní sa
príkaz "on error goto Nie_sme_v_tabulke"
  tabulka=kurzor.TextTable

  REM Sme v už existujúcej tabuľke

  REM Nasledujúci test môžeme vynechať, ak nám nevadí, že
sa prepíše iba časť tabuľky
  REM Ak má existujúca tabuľka iné rozmery, ako
skopírovaná, tak to oznámime
  if tabulka.columns.count<>stlpce_z_calcu_do_writer or
tabulka.rows.count<>riadky_z_calcu_do_writer then
```

```
   msgbox "Zdrojová tabuľka má iné rozmery ako cieľová!"
   REM a ukončíme makro
   exit sub
  end if

  mame_tabulku:
  REM Už máme určite tabuľku, môžeme do nej vkladať obsah
"schránky"
  on error goto next

  REM Pre všetky riadky "schránky"
  for riadok=1 to riadky_z_calcu_do_writer
   REM a pre všetky stĺpce "schránky"
   for stlpec=1 to stlpce_z_calcu_do_writer
    REM Pripravíme si textovú adresu tabuľky
    adresa=chr(64+stlpec)&riadok
    REM Nastavíme sa do bunky textovej tabuľky
    bunka=tabulka.getCellByName(adresa)
    REM A zmeníme jej obsah podľa "schránky"
    bunka.String= bunky_z_calcu_do_writer(riadok,stlpec)
   next stlpec
  next riadok
 else
  REM Ak sme nič v Calcu neuložili, oznámime to
  msgbox "V schránke nie je žiadna tabuľka."
 end if

 REM Ukončíme makro, lebo ďalej je vytvorenie novej
tabuľky!
 exit sub

 Nie_sme_v_tabulke:
  REM Sme mimo tabuľky, pripravíme si novú
  tabulka=dokument.createInstance("com.sun.star.text.TextTab
le")
  REM Nastavíme jej rozmery
  tabulka.initialize(riadky_z_calcu_do_writer,
stlpce_z_calcu_do_writer)
  REM a vložíme ju do dokumentu na aktuálnu pozíciu
  dokument.Text.insertTextContent(kurzor, tabulka, false)
  REM Vrátime sa nazad - už sme v tabuľke
  goto mame_tabulku
end sub
```

V prípade, že sa nachádzame v už existujúcej tabuľke, testovali sme aj jej stávajúci rozmer, pričom sme požadovali, aby bol totožný so skopírovaným rozsahom.

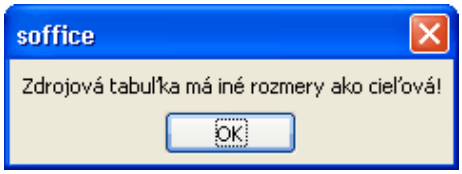


Obrázok 48.3: Skopírovaná tabuľka do dokumentu Writer



Obrázok 48.4: Oznam, že sme nevybrali žiadnu tabuľku pre kopírovanie

Toto však nemusí každému vyhovovať. Pokiaľ niekto nechce toto testovať, stačí, ak si príslušnú časť dá ako komentár (pre prípad, že by makro chcel niekedy v tomto smere predsa len zmeniť):



Obrázok 48.5: Oznam, že zdrojová a cieľová tabuľka nie sú zhodné

```
  REM Sme v už existujúcej tabuľke
  REM Ak má existujúca tabuľka iné
rozmery, ako skopírovaná, tak to oznámime
  '  if tabulka.columns.count<>stlpce_z_calcu_do_writer or
tabulka.rows.count<>riadky_z_calcu_do_writer then
  '   msgbox "Zdrojová tabuľka má iné rozmery ako cieľová!"
  '   REM a ukončíme makro
  '   exit sub
  '  end if
```

49. Práca so súbormi OpenOffice.org

49.1. Práca s textovými súbormi modulu Writer

Niekedy potrebujeme v makrách otvárať iné súbory – napríklad za účelom ich spojenia do jedného a podobne. Inokedy potrebujeme zistiť, či nejaký súbor vôbec existuje a ak nie, tak ho potrebujeme zároveň vytvoriť.

Tieto možnosti si ukážeme na príklade textového dokumentu, do ktorého budeme pridávať prvý odsek aktuálneho dokumentu. A aby sme nezostali iba pri tomto príklade, ukážeme si aj makro, pomocou ktorého pridáme do dokumentu tabuľku. Pravda-že, týmto nie sú zďaleka vyčerpané možnosti, ide iba o príklady, ako môžeme pracovať s dokumentom bez toho, aby sme ho manuálne otvárali, ukladali a zatvárali.

Ako obvykle, v makrách sa nachádzajú komentáre a preto ich bližšie nebudeme popisovať.

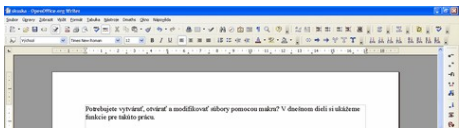
```
REM Funkcia otvorí existujúci súbor alebo vytvorí nový
súbor
REM Funkcia vráti hodnotu true, ak súbor otvorila
(vytvorila)
function otvor_subor (oSubor, nazov as string, novy) as
boolean
 dim url as string
  url=nazov
```

```
 novy=not FileExists(url) ' Zistíme, či súbor existuje
 if novy then
  ' Ak neexistuje, pripravíme si url pre nový súbor
  url="private:factory/swriter"
 end if
 ' Otvorenie existujúceho alebo vytvorenie nového súboru
 oSubor=StarDesktop.loadComponentFromURL(url, "_blank", 0,
Array())
 otvor_subor=not isNull(oSubor)
end function

REM Procedúra pre uloženie súboru
REM Podľa parametra "novy" sa uloží buď poď existujúcim
alebo novým názvom.
sub uloz_subor (oSubor, nazov as string, novy as boolean)
 if novy then
  ' Ak máme nový súbor, uložíme hod definovaným názvom
  oSubor.storeAsUrl(nazov, Array())
 else
  ' Ak máme otvorený existujúci súbor, uložíme ho priamo
  oSubor.store()
 end if

end sub

REM Procedúra pre zatvorenie súboru
sub zatvor_subor (oSubor)
 oSubor.dispose()
end sub
```

Makro pre pridanie prvého odseku aktuálneho dokumentu do súboru „ skuska.odt“:

```
sub pracuj_so_suborom_1
 dim dokument, kurzor, cesty as object
 dim aktual_dokument, aktual_kurzor as object
 dim nazov as string
 dim novy, otvoreny, koniec as boolean

 ' Budeme otvárať (vytvárať) súbor "skuska.odt" v
štandardnom pracovnom adresári ("Moje dokumenty" a pod.)
 cesty=CreateUnoService("com.sun.star.util.PathSettings")
 nazov=cesty.Work+"/skuska.odt"

 ' Nastavíme sa do aktuálneho dokumentu
 aktual_dokument=StarDesktop.CurrentComponent
 aktual_kurzor=aktual_dokument.text.createTextCursor
 otvoreny=otvor_subor (dokument, nazov, novy)
 if otvoreny then
  ' Ak sme otvorili/vytvorili súbor, tak
  ' Súbor sa správa ako "ThisComponent", preto môžeme
```

```
pracovať s viditeľným kurzorom
  kurzor=dokument.currentcontroller.getViewCursor()
  kurzor.gotoEnd(false) ' Skočíme na koniec súboru
  ' Pridáme odsek z aktuálneho dokumentu
  ' Skočíme na začiatok odseku
  aktual_kurzor.gotoStartOfParagraph(false)
  ' Označíme odsek
  aktual_kurzor.gotoEndOfParagraph(true)
  ' a vložíme ho do nového dokumentu
  kurzor.setstring(aktual_kurzor.String)
  kurzor.gotoEnd(false) ' Skočíme na koniec vloženého textu
  uloz_subor(dokument, nazov, novy) ' Uložíme
zmenený/vytvorený súbor
  zatvor_subor(dokument) ' Zatvoríme zmenený/vytvorený
súbor
 end if
end sub
```



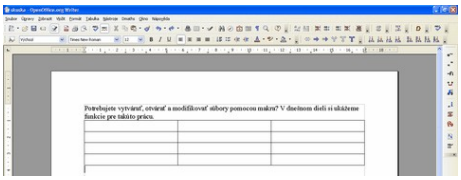
Obrázok 49.1: Vytvorený súbor "Skuska" s vloženým textom

Makro pre pridanie tabuľky na koniec súboru „skuska.odt“. V tomto makre pre zmenu na konci súbor „skuska.odt“ nezatvoríme, ale zostaneme v ňom pre ďalšiu prácu.

```
sub pracuj_so_suborom_2
 dim dokument, kurzor, tabulka, cesty as object
 dim nazov as string
 dim novy, otvoreny as boolean

 ' Budeme otvárať (vytvárať) súbor "skuska.odt" v
štandardnom pracovnom adresári ("Moje dokumenty" a pod.)
 cesty=CreateUnoService("com.sun.star.util.PathSettings")
 nazov=cesty.Work+"/skuska.odt"

 otvoreny=otvor_subor (dokument, nazov, novy)
 if otvoreny then
  ' Ak sme otvorili/vytvorili súbor, tak
  ' Súbor sa správa aj ako "StarDesktop.CurrentComponent",
preto môžeme využiť aj tieto metódy
  kurzor=dokument.currentcontroller.getViewCursor()
  kurzor.gotoEnd(false) ' Skočíme na koniec súboru
  ' A vložíme tam napríklad tabuľku - 4 riadky a 3 stĺpce
  tabulka=dokument.createInstance("com.sun.star.text.TextTab
le")
  tabulka.initialize(4,3)
  dokument.Text.insertTextContent(kurzor, tabulka, false)
  kurzor.gotoEnd(false) ' Skočíme na koniec vloženého textu
  ' Teraz môžeme vložiť údaje do tabuľky ... a nakoniec
uložiť a zatvoriť súbor
  ' Alebo ho jednoducho nezatvoríme a budeme pokračovať v
jeho úprave manuálne - príkazy stačí dať do komentára
```

```
  'uloz_subor(dokument, nazov, novy) ' Uložíme
zmenený/vytvorený súbor
  'zatvor_subor(dokument) ' Zatvoríme zmenený/vytvorený
súbor
 end if
end sub
```

49.2. Práca so zošitmi modulu Calc



Obrázok 49.2: Vložená tabuľka do súboru "Skuska"

Potom, ako sme si v predchádzajúcom dieli ukázali, ako dokážeme v makrách vytvárať (modifikovať) externé dokumenty si dnes ukážeme, ako dokážeme vytvárať nové listy v zošite tabuľkového procesora.

V diskusii k predchádzajúcej časti seriálu o makrách ste sa opýtali, ako je možné vytvárať listy v zošite tabuľkového procesora pomocou makra. Hoci makro nie je zložité, vzniká otázka, ako vlastne zadávať názov listu, pretože tieto sa musia nejako nazývať.

Mohli by sme, podobne ako keď sa zadáva nový list priamo v tabuľkovom procesore, tento názov zadávať cez dialóg alebo cez vstupné pole. Toto riešenie je však pomerne zbytočné, pretože ak už zadávame názov, tak to môžeme urobiť aj mimo prostredia makra – priamo pomocou nástrojov OpenOffice.org.

Z tohto hľadiska sa mi javí ako výhodnejšie, aby sme názov nového listu zadali priamo v makre. Pravdaže, v prípade, že list so zadaným názvom už existuje, nevytvoríme ho:

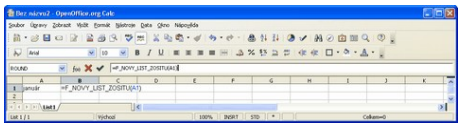
```
sub novy_list_zositu
 dim dokument, list as object
 dokument=StarDesktop.CurrentComponent

 if not dokument.Sheets.hasByName("Novy_list") then
  list=dokument.createInstance("com.sun.star.sheet.Spreadshe
et")
  dokument.Sheets.insertByName("Novy_list", list)
 end if
end sub
```

Pravdaže, toto riešenie vôbec nie je optimálne, pretože názov je konštantný a v príslušnom dokumente dokážeme takto pridať list iba raz. Preto je lepšie, ak si názov nového listu zadáme do niektorej bunky a pomocou jej obsahu nový list zadefinujeme.

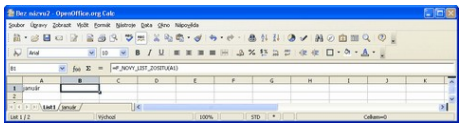
Týmto spôsobom dokážeme automaticky pripravovať povedzme nové listy pre faktúry, ak ich pomocou tabuľkového procesora vytvárame. Nie je problém, aby sme v niektorej bunke mali zapísané posledné číslo faktúry, z ktorej dokážeme odvodiť nové a priamo v makre ho môžeme aj upraviť. Preto si ukážeme príklad makra, kde názov nového listu je ako parameter funkcie, ktorú môžeme priamo zapísať do niektorej bunky v tabuľke.

```
function f_novy_list_zositu (nazov)
 dim dokument, list as object
 dokument=StarDesktop.CurrentComponent
```

```
 if not dokument.Sheets.hasByName(nazov) then
  list=dokument.createInstance("com.sun.star.sheet.Spreadshe
et")
  dokument.Sheets.insertByName(nazov, list)
 end if
 fnovy_list_zositu=""
end function
```

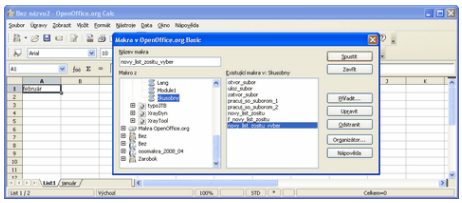


Obrázok 49.3: Volanie funkcie pre vytvorenie nového listu



Obrázok 49.4: Vytvorený nový list

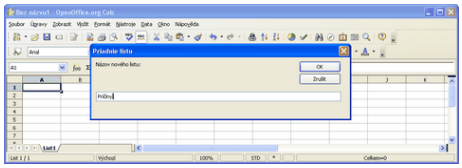
V prípade, že nám nevyhovuje ani takéto riešenie, môžeme pridanie nového listu riešiť napríklad tak, že si do niektorej bunky zapíšeme nový názov. Stačí, ak na tejto bunke zostaneme (čím je automaticky vybraná) a spustme makro, ktoré z aktuálneho výberu vyberie prvú bunku a podľa jej obsahu vytvorí nový list.



Obrázok 49.5: Volanie makra cez zoznam makier

```
sub novy_list_zositu_vyber
 dim dokument, list, bunka as object
 dim stlpce, riadky as long
 dim nazov as string

 dokument=ThisComponent.getCurrentSelection()
 stlpce=dokument.getcolumns().Count
 riadky=dokument.getrows().Count

 if stlpce<>0 and riadky<>0 then
  bunka=dokument.getCellByPosition(0,0)
  nazov=bunka.getstring()
  dokument=StarDesktop.CurrentComponent
  if not dokument.Sheets.hasByName(nazov) then
   list=dokument.createInstance("com.sun.star.sheet.Spreadsh
eet")
   dokument.Sheets.insertByName(nazov, list)
  end if
 end if
end sub
```

V poslednom prípade je vhodné, aby sme si makro pridali buď do panela nástrojov, alebo priradili k nemu klávesovú skratku, lebo jeho spúšťanie cez menu nie je práve najpohodlnejšie.



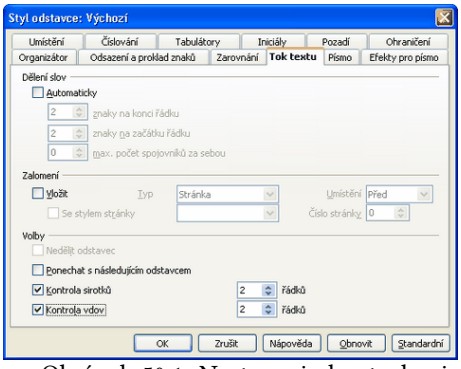
Obrázok 49.6: Zadanie názvu nového listu

Aby sme však neboli nespravodliví k tým, ktorí potrebujú zadávať meno listu priamo v makre, pridávame nakoniec aj túto verziu s tým, že budeme aj testovať, či názov nie je prázdny (v takom prípade by sa totiž vytvoril list s názvom „0“).

```
sub novy_list_zositu
 dim dokument, list as object
 dim nazov as string

 dokument=StarDesktop.CurrentComponent
 nazov=InputBox("Názov nového listu: ","Priadnie listu","")

 if not dokument.Sheets.hasByName(nazov) and nazov<>"" then
  list=dokument.createInstance("com.sun.star.sheet.Spreadshe
et")
  dokument.Sheets.insertByName(nazov, list)
 end if
end sub
```

50. Nastavenie rozostupu (kerningu) znakov

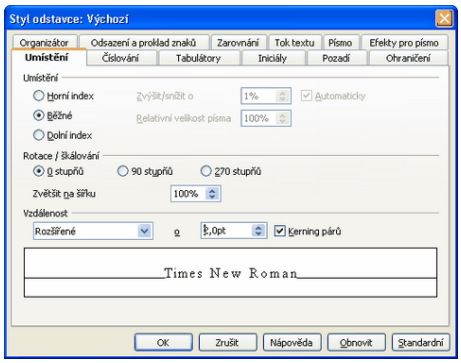
V dnešnom dieli seriálu o makrách sa znovu vrátime k textovým dokumentom a ukážeme si, ako dokážeme nastaviť rozostup medzi znakmi.



Obrázok 50.1: Nastavenie kontroly sirôt a vdov

Pokiaľ používame OpenOffice.org na písanie kníh či iných podobných publikácií, chceme, aby vyzeral dobre aj z hľadiska estetickej úpravy, ktoré sú overené generáciami našich predkov a ktoré do svojich pravidiel zahrnuli typografi. Medzi takéto náležitosti patrí napríklad aj to, že na konci strany sa nenecháva jeden riadok textu a na začiatku novej strany tiež nie (tzv. siroty a vdovy).

Samozrejme, v OpenOffice.org dokážeme nastaviť, že sa takéto riadky nebudú vyskytovať a to v nastaveniach štýlov odsekov (záložka „Tok textu“, voľby „Kontrola sirotkov“ a „Kontrola vdov“. Pokiaľ však toto nastavíme, častokrát sa nám stane, že na konci strany zostane prázdny riadok.



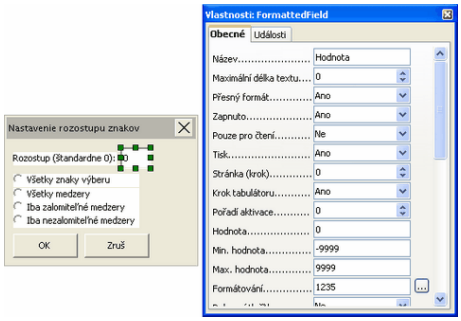
Obrázok 50.2: Nastavenie rozostupu znakov v štýle odseku

Pokiaľ tlačíme dokument na obidve strany papiera (ale nielen vtedy), vzniká nám nový problém – na dvoch susediacich stranách nemusí byť na konci strany text, čo pôsobí rušivo. A práve vtedy môžeme zasiahnuť podobne, ako to robili sadzači pri ručnej sadzbe – zmenením rozostupu medzi znakmi.

Samozrejme, rozostupy medzi znakmi sa dajú tiež nastaviť v štýle odseku (záložka „Umiestnenie“, nastavenia „Vzdialenosť“). To však platí pre všetky odseky a netýka sa to iba nastavenia konkrétnej strany. Pokiaľ chceme nastavovať

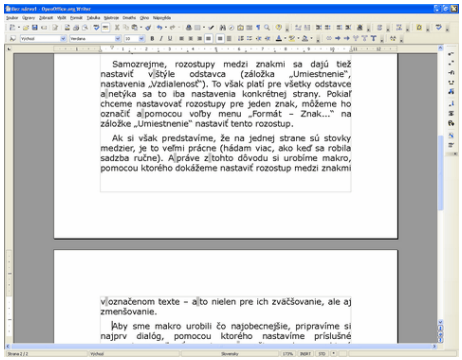
rozostupy pre jeden znak, môžeme ho označiť a pomocou voľby menu „Formát – Znak...“ na záložke „Umiestnenie“ nastaviť tento rozostup.

Ak si však predstavíme, že na jednej strane sú stovky medzier, je to veľmi prácne (hádam viac, ako keď sa robila sadzba ručne). A práve z tohto dôvodu si urobíme makro, pomocou ktorého dokážeme nastaviť rozostup medzi znakmi v označenom texte – a to nielen pre ich zväčšovanie, ale aj zmenšovanie.



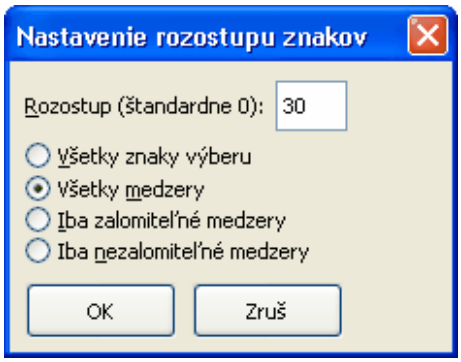
Obrázok 50.3: Príprava dialógu pre nastavenie rozostupov

Aby sme makro urobili čo najobecnejšie, pripravíme si najprv dialóg, pomocou ktorého nastavíme príslušné parametre – veľkosť rozostupu a množinu znakov, pre ktoré ho chceme použiť. Pretože obvykle nastavujeme rozostup buď medzi všetkými znakmi alebo iba v rámci medzier, do dialógu vložíme štyri prepínacie voľby: „Všetky znaky vo výbere“, „Všetky medzery“, „Iba zalomiteľné medzery“ a „Iba nezalomiteľné medzery“.



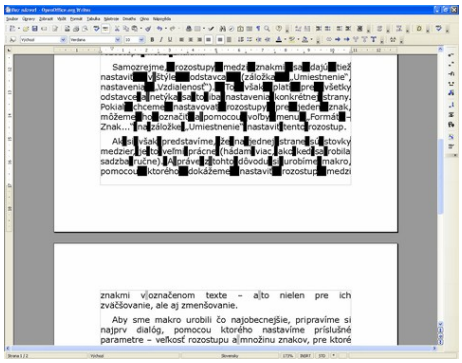
Obrázok 50.4: Text pred zmenou rozostupov medzier

Ďalej si v dialógu nastavíme pre vstupnú hodnotu formátovanie (celé čísla), minimálnu a maximálnu hodnotu (-9999 až 9999). Toto nastavenie urobíme preto, aby sme nemuseli zadanú hodnotu špeciálne testovať. Záporné hodnoty rozostupy zmenšujú, štandardný rozostup je hodnota 0 a kladné hodnoty rozostupy zväčšujú. Samozrejme, konkrétne čísla je potrebné vyskúšať osobitne pre každú nastavovanú stranu – v tomto smere sa nedá povedať, aká hodnota je najlepšia.



Obrázok 50.5: Nastavenie rozostupov medzier

Vlastné makro tvoria dve funkcie – otvorenie dialógu a nastavenie rozostupov. Makrá sú, ako obvykle, okomentované a preto sa nimi nemusíme hlbšie zaoberať.



Obrázok 50.6: Text po zmene rozostupov medzier

```
REM Makro pre úpravu rozostupov
medzi znakmi vo vybranom texte
sub uprav_kerning_znakov(medzera,
all, sp_nbsp, nbsp as long)
 ' Význam parametrov:
 ' Medzera – veľkosť rozostupu

 ' all – ak je 1, tak nastavujeme
```

```
všetky znaky
 ' sp_nbsp - ak je 1, tak nastavujeme všetky medzery
 ' nbsp - ak je 1, tak iba nezalomiteľné medzery
 ' Ak všetky predchádzajúce parametre sú 0, tak sa
nastavuje iba zalomiteľná medzera

 dim dokument, kurzor as object ' Aktuálny dokument,
viditeľný kurzor
 dim hladaj as string ' Hľadaný reťazec podľa parametrov

 dim args(18) as new com.sun.star.beans.PropertyValue '
Parametre pre hľadanie v označenom texte

 if all=0 then
  ' Ak nenastavujeme všetky znaky, tak
  hladaj=" " ' nastavenie hľadaného reťazca na SP
  if sp_nbsp=1 then
   ' Ak hľadáme všetky medzery, tak
   hladaj=" |"+chr$(&HA0) ' nastavenie hľadania SP alebo
NBSP
  else
   if nbsp=1 then
    ' Ak hľadáme iba NBSP, tak
    hladaj=chr$(&HA0) ' nastavenie hľadania NBSP
   end if
  end if

  ' Nastavenie nasledujúcich parametrov je z nahrávania
makra

  args(0).Name = "SearchItem.StyleFamily"
  args(0).Value = 2
  args(1).Name = "SearchItem.CellType"
  args(1).Value = 0
  args(2).Name = "SearchItem.RowDirection"
  args(2).Value = true
  args(3).Name = "SearchItem.AllTables"
  args(3).Value = false
  args(4).Name = "SearchItem.Backward"
  args(4).Value = false
  args(5).Name = "SearchItem.Pattern"
  args(5).Value = false
  args(6).Name = "SearchItem.Content"
  args(6).Value = false
  args(7).Name = "SearchItem.AsianOptions"
  args(7).Value = false
  args(8).Name = "SearchItem.AlgorithmType"
  args(8).Value = 1 ' Budeme hľadať regulárne výrazy
  args(9).Name = "SearchItem.SearchFlags"
```

```
  args(9).Value = 71680 ' Hľadá sa v označenom texte
(výbere). Pre hľadanie v celom dokumente by muselo byť
65536
  args(10).Name = "SearchItem.SearchString"
  args(10).Value = hladaj ' Hľadaný reťazec
  args(11).Name = "SearchItem.ReplaceString"
  args(11).Value = ""
  args(12).Name = "SearchItem.Locale"
  args(12).Value = 255
  args(13).Name = "SearchItem.ChangedChars"
  args(13).Value = 2
  args(14).Name = "SearchItem.DeletedChars"
  args(14).Value = 2
  args(15).Name = "SearchItem.InsertedChars"
  args(15).Value = 2
  args(16).Name = "SearchItem.TransliterateFlags"
  args(16).Value = 1280
  args(17).Name = "SearchItem.Command"
  args(17).Value = 1
  args(18).Name = "Quiet"
  args(18).Value = true

  dokument=ThisComponent.CurrentController.Frame
  dispatcher =
createUnoService("com.sun.star.frame.DispatchHelper")
  dispatcher.executeDispatch(dokument,
".uno:ExecuteSearch", "", 0, args()) ' Vlastné hľadanie -
označenie všetkých nájdených znakov
 end if
 kurzor= ThisComponent.currentcontroller.getViewCursor()
 ' Viditeľný kurzor
 ' ak sme nehľadali medzery, tak je to celý označený text,
inak iba nájdené medzery
 kurzor.CharKerning=medzera ' Zmena rozostupu označených
znakov
end sub

Rem Volanie dialógu a makra pre nastavenie rozostupov
sub Kerning_znakov

 dim medzera as long ' Pre získanie číselnej hodnoty z
dialógovej premennej "Hodnota"

 DialogLibraries.LoadLibrary("JP_Kerning_znakov") '
Otvorenie knižnice

 dlg=CreateUnoDialog(DialogLibraries.JP_Kerning_znakov.Dialo
g_Kerning_znakov) ' Sprístupnenie dialógu
```

```
 dlg.model.Vyber_all.State=0
 dlg.model.Vyber_SP_NBSP.State=1 ' V prepínačoch si
prednastavíme, že budeme meniť rozostupy všetkých medzier
 dlg.model.Vyber_SP.State=0
 dlg.model.Vyber_NBSP.State=0

 dlg.model.Hodnota.text="0" ' 0 znamená, že rozostupy majú
byť štandardné

 if dlg.Execute()=1 then
  ' Ak sme stlačili "OK", tak
  medzera=val(dlg.model.Hodnota.text) ' Hodnota rozostupov
  uprav_kerning_znakov(medzera,dlg.model.Vyber_all.State,dl
g.model.Vyber_SP_NBSP.State,dlg.model.Vyber_NBSP.State) '
Nastavenie rozostupov
 end if
 dlg.dispose() ' Zatvorenie dialógu
end sub
```

51. Rozšírenie pre slovník OpenOffice.org

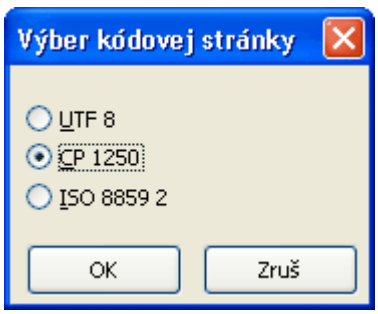
Chcete pomôcť pri tvorbe dobrého slovenského slovníka, ktorý používajú rôzne Open Source aplikácie (OpenOffice.org, Mozilla Thunderbird...)? Nainštalujte si do OpenOffice.org rozšírenie SK_SK_Slovnik.

Pokiaľ používate Open Source programy, kde sa kontroluje slovenská gramatika, ako sú OpenOffice.org či Mozilla Thunderbird, určite ste sa už stretli s tým, že nie všetky slová boli súčasťou slovníka. V spolupráci s pánom Zdenkom Podobným, hlavným autorom týchto slovníkov, sme pripravili rozšírenie pre OpenOffice.org, pomocou ktorého môžete spolupracovať na jeho zdokonalení.

Obrázok 51.1: Panel nástrojov "SK_SK_Slovnik"

Rozšírenie si nainštalujete jednoducho – stačí, ak poklepete na distribučný súbor „SK_SK_Slovnik.oxt“, alebo ho pridáte cez menu „Nástroje – Správca rozšírení“. Po reštarte OpenOffice.org sa vám pridá do programu panel nástrojov „SK_SK_Slovnik“, ktorý obsahuje jedno tlačidlo „Príprava slova pre slovník“.

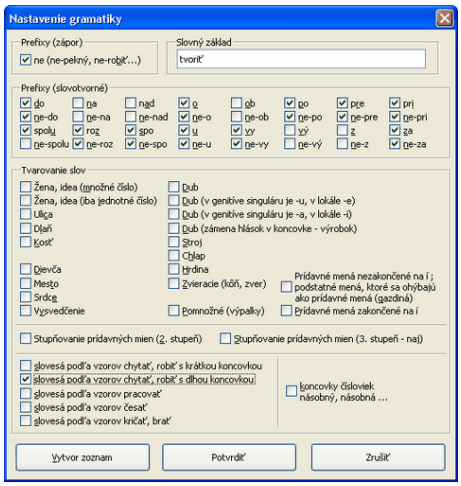
Po jeho stlačení ste pri prvom spustení rozšírenia (resp. reštarte OpenOffice.org) vyzvaní k zadaniu kódovej stránky, ktorú používa váš operačný systém. Pre Windows je potrebné vybrať voľbu „CP 1250“, v Mandrive voľbu „UTF 8“ (preventívne je pripravená aj možnosť „ISO 8859-2“). Následne sa po krátkom čase, v ktorom sa načíta definícia tvarovania slov (práve tá je závislá od kódovej stránky) zobrazí okno, v ktorom môžete zadať slovo do slovníka.



Obrázok 51.2: Výber kódovej stránky

Pokiaľ neviete, akú kódovú stránku máte použiť, dá sa to jednoducho vyskúšať. Vyberte si prvú, zadajte povedzme slovo „tvoriť“ a tvarovanie „slovesá podľa vzorov chytať, robiť s dlhou koncovkou“. Dajte vytvoriť kontrol-

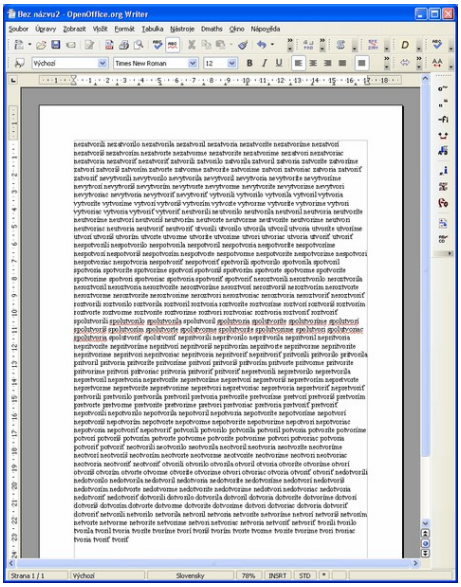
ný zoznam, ktorý v prípade správnej kódovej stránky bude obsahovať slová: tvorili tvorilo tvorila tvoril tvoria tvoríte tvoríme atď.



Obrázok 51.3: Nastavenie gramatiky

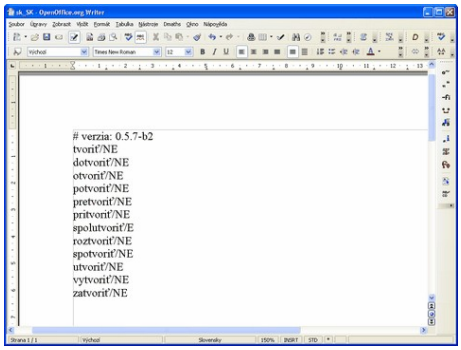
Dialógové okno je rozdelené na niekoľko logických častí. V úvodnej časti sú predpony, pomocou ktorých môžeme napísaním jedného základného slova zadať do slovníka slov niekoľko. Pri tomto musíme upozorniť, že predpony typu „nedo“, „nena“ a pod. sa pridajú iba vtedy, ak sú zároveň pridané aj predpony „do“, „na“ atď. Týmto spôsobom môžeme napríklad zadaním slova „robiť“ a predpôn „ne“, „do“, „nedo“, „na“, „nena“ ... pridať do slovníka nasledujúce slová: robiť, nerobiť, dorobiť, nedorobiť, narobiť, nenarobiť...

Pozor, tieto predpony sa k slovám pridávajú bez zmeny základného slova, takže napríklad pridaním predpony „po“ k slovu „otvoriť“ pridáte slovo „pootvoriť“.


Obrázok 51.4: Zoznam slov

Následne je potrebné zadať tvarovanie, ako sú vzory podstatných mien, vzory slovies a pod. Aby sme si boli istí, či sme zadali správne tvarovanie, pomocou tlačidla „Vytvor zoznam“ sa nám otvorí nový súbor, v ktorom sa následne vytvoria všetky slová (vrátane tvarovania), ktoré budú považované za správne. Samozrejme, v zozname sú akceptované zároveň všetky zadané predpony, takže slov môže byť veľmi veľa.

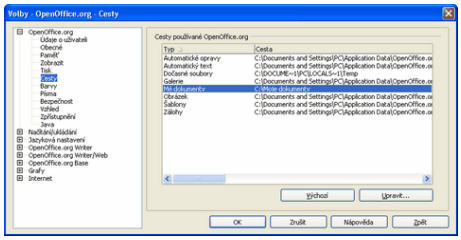
Ak sme zadali slovo správne, stlačením tlačidla „Potvrdiť“ sa pridá toto slovo do súboru „sk_SK.odt“. Tento súbor, pokiaľ neexistuje, sa automaticky vytvorí v systémovom adresári „Moje dokumenty“ a ako prvý riadok vloží verziu rozšírenia (aktuálne 0.5.7-b2). Konkrétny adresár záleží od toho, aký ho máme nastavený v systéme a môžeme ho zistiť cez voľbu menu „Nástroje – Voľby – OpenOffice.org – Cesty“. Pokiaľ súbor už existuje, budú sa do neho postupne pridávať slová.



Obrázok 51.5: Ukážka vytvorených definícií slov v súbore "sk_SK.odt"

Na tomto mieste zároveň poprosíme, pokiaľ objavíte, že napriek správne zadanému tvarovaniu sa v kontrolnom zozname nachádzajú aj slová, ktoré nie sú gramaticky správne, prosím napíšte to autorovi slovníka, aby to v budúcej verzii opravil. Následne podľa toho bude urobená aj nová verzia tohto rozšírenia.

Pretože automatické zatváranie tohto súboru viedlo k nestabilite rozšírenia (havárii OpenOffice.org), je potrebné zatvoriť súbor manuálne. Súbor sa však predtým automaticky uloží, takže o toto sa už nemusíme starať. Pozor! Pred vkladaním ďalšieho slova je nutné súbor zatvoriť, lebo inak dôjde k opätovnému otvoreniu tohto súboru (teraz však v režime „Read-Only“), čo vedie k havárii makra.



Obrázok 51.6: Zistenie pracovných ciest cez menu "Nástroje – Voľby – OpenOffice.org – Cesty"

Pre pridanie ďalšieho slova musíme znovu stlačiť tlačidlo na pracovnom paneli rozšírenia. Pokiaľ máme na riadku, kde sa nachádzame niečo napísané, program skopíruje tento riadok ako slovo, ktoré chceme vložiť do slovníka pričom kontroluje, či nie je zadané v tvare, aký sa používa v slovníku. Ak áno, nastaví aj príslušné tvarovanie.

Upozorňujeme, že týmto rozšírením sa automaticky negeneruje nový slovník a preto slovo, ktoré zadáme sa nestane ihneď súčasťou slovníka. Keď to budete považovať za potrebné, súbor „sk_SK.odt“ je potrebné poslať mailom p. Zdenkovi Podobnému (zdenop at gmail.com) alebo autorovi rozšírenia p. Júliusovi Pastierikovi (pastierik at inet.sk). Po zaslaní tento súbor vymažte, aby sa do neho pridávali už iba nové slová.

Ďakujeme týmto všetkým tým používateľom, ktorí si rozšírenie nainštalujú a pomôžu pri tvorbe dobrého slovníka pre tie Open Source programy, ktoré ho budú používať. O každej novej verzii budeme pravidelne informovať. Všetkých „makromilov“ zároveň upozorňujeme, že zdrojové texty rozšírenia zverejníme v ďalšom dieli seriálu o makrách.

Internet: SK_SK_Slovnik.oxt

51.1. Tvorba rozšírenia

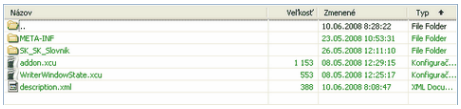
Pri popise rozšírenia pre prípravu slov do slovenského slovníka sme sľúbili, že prinesieme aj jeho zdrojové kódy. Skôr, ako sa dostaneme k tomuto makru, musíme sa ešte venovať problematike rozšírení, pretože toto makro je naň úzko previazané.

Pri praktickom programovaní makier potrebujeme z neho mnohokrát urobiť rozšírenie, pomocou ktorého sa makro automaticky vloží do OpenOffice.org vrátane možností jeho ľahkého volania či už pomocou menu alebo vložených panelov nástrojov. Pozrime sa preto na to, ako to dokážeme urobiť.

Hneď na začiatok uveďme, že tento návod rozhodne nie je úplný, ale iba popíše niektoré základné veci, ktoré musíme pri tvorbe rozšírení dodržať. Pretože rozšírenia sú vlastne ZIP súbory, ich štúdiom sa následne môžeme naučiť aj ďalšie „finty“. V neposlednom rade je potom na stránkach vývojárov OpenOffice.org k dispozícii dokumentácia pre tvorbu rozšírení.

Dosť však bolo teórie. Pri praktickej tvorbe rozšírenia je samozrejmé, že makro najprv odladíme v prostredí OpenOffice.org. Pokiaľ vieme, že z neho budeme rozbiť rozšírenie je najlepšie, ak ho uložíme do osobitnej knižnice. Táto je potom k dispozícii ako samostatný adresár v používateľskom adresári OpenOffice.org (v OS Windows ho nájdeme v adresári „C:\Documents and Settings\ používateľ \Application Data\Ope-

nOffice.org2\user\basic\“).



Obrázok 51.7: Obsah adresára "SK_SK_Slovnik"

Teraz si pripravíme na inom mieste adresár, ktorý bude tvoriť v konečnom dôsledku vlastné rozšírenie (napríklad „SK_SK_Slovnik“). Do neho skopírujeme 1:1 adresár knižnice s makrami, z ktorého ideme vytvárať rozšírenie (osobne používam úplne rovnaké názvy). Ďalej si musíme v tomto adresári vytvoriť adresár „META-INF“ – pozor, musí sa menovať presne takto vrátane dodržania veľkosti písmen.

Pre ďalšiu prácu potrebujeme textový editor, ktorý umožňuje pracovať so súbormi v kódovaní „UTF-8“. Osobne mi pre túto prácu najviac vyhovuje český editor PSPad, ktorý je možné používať zadarmo aj pre komerčné účely. O tom, čo všetko umožňuje si prinesieme informáciu vrámci recenzií programov.

Obrázok 51.8: Súbor "manifest.xml"

V adresári „META-INF“ teraz vytvoríme súbor „manifest.xml“ s nasledujúcim obsahom:

```
<?xml version="1.0" encoding="UTF-
8"?>
<manifest:manifest>
 <manifest:file-entry manifest:full-
path="WriterWindowState.xcu" manifest:media-
type="application/vnd.sun.star.configuration-data"/>
 <manifest:file-entry manifest:full-path="addon.xcu"
manifest:media-type="application/vnd.sun.star.configuration-
data"/>
 <manifest:file-entry manifest:full-path="SK_SK_Slovnik/"
manifest:media-type="application/vnd.sun.star.basic-
library"/>
</manifest:manifest>
```

Týmto zápisom oznamujeme OpenOffice.org, kde sa nachádzajú ostatné popisné časti rozšírenia.

Teraz sa vrátime do hlavného adresára, kde vytvoríme tri súbory: „addon.xcu“, „WriterWindowState.xcu“ a „description.xml“.

V súbore „description.xml“ definujeme základné parametre rozšírenia, ako je jeho názov (pozor, musí byť vrámci OpenOffice.org jedinečný), verzia, link, kde je uložený a pod. Jeho obsah môže byť napríklad takýto:

```
<?xml version="1.0" encoding="UTF-8"?>
<description
xmlns="http://openoffice.org/extensions/description/2006"
xmlns:xlink="http://www.w3.org/1999/xlink">
 <version value="0.5.7-b2" />
 <identifier value="SK_SK_Slovnik.addon" />
 <update-information>
  <src
xlink:href="http://www.inet.sk/images/user/pastierik/ooomak
ra/SK_SK_Slovnik.oxt"/>
```

```
 </update-information>
</description>
```

V súbore „WriterWindowState.xcu“ teraz budeme definovať, že do OpenOffice.org pridávame nový panel nástrojov, ktorý nazveme úplne rovnako, ako rozšírenie – „SK_SK_Slovnik“:

```
<?xml version='1.0' encoding='UTF-8'?>
<oor:component-data
xmlns:oor="http://openoffice.org/2001/registry"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
oor:name="WriterWindowState"
oor:package="org.openoffice.Office.UI">
 <node oor:name="UIElements">
  <node oor:name="States">
   <node
oor:name="private:resource/toolbar/addon_name.JP.addon.SK_SK
_Slovnik" oor:op="replace">
    <prop oor:name="UIName" oor:type="xs:string">
     <value xml:lang="en-US">SK_SK_Slovnik</value>
    </prop>
   </node>
  </node>
 </node>
</oor:component-data>
```

Samozrejme, nie sme obmedzení iba na jeden panel. Napríklad pomocou tejto definície zadefinujeme až štyri panely nástrojov („JP_formatuj_dokument“, „JP_formatuj_medzery“, „JP_specialna_zamena“ a „JP_medzery_on_off“):

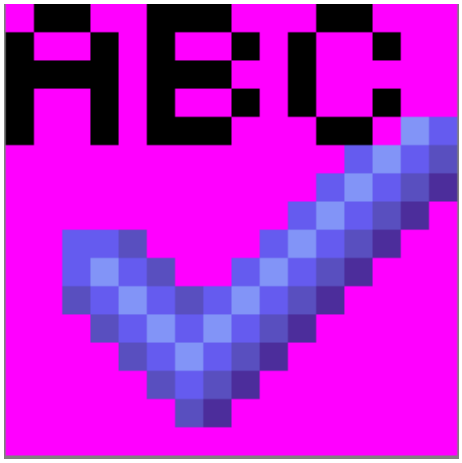
```
<?xml version='1.0' encoding='UTF-8'?>
<oor:component-data
xmlns:oor="http://openoffice.org/2001/registry"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
oor:name="WriterWindowState"
oor:package="org.openoffice.Office.UI">
 <node oor:name="UIElements">
  <node oor:name="States">

   <node
oor:name="private:resource/toolbar/addon_name.JP_format_zam
en.addon.JP_formatuj_dokument" oor:op="replace">
    <prop oor:name="UIName" oor:type="xs:string">
     <value xml:lang="en-US">JP_formatuj_dokument</value>
    </prop>
   </node>

   <node
oor:name="private:resource/toolbar/addon_name.JP_format_zam
en.addon.JP_formatuj_medzery" oor:op="replace">
    <prop oor:name="UIName" oor:type="xs:string">
```

```
      <value xml:lang="en-US">JP_formatuj_medzery</value>
     </prop>
    </node>

    <node
oor:name="private:resource/toolbar/addon_name.JP_format_zam
en.addon.JP_specialna_zamena" oor:op="replace">
     <prop oor:name="UIName" oor:type="xs:string">
      <value xml:lang="en-US">JP_specialna_zamena</value>
     </prop>
    </node>

    <node
oor:name="private:resource/toolbar/addon_name.JP_format_zam
en.addon.JP_medzery_on_off" oor:op="replace">
     <prop oor:name="UIName" oor:type="xs:string">
      <value xml:lang="en-US">JP_medzery_on_off</value>
     </prop>
    </node>

   </node>
  </node>
</oor:component-data>
```

Nakoniec musíme v súbore „addon.xcu“ zadefinovať vlastný obsah panela nástrojov. Nasledujúca definícia obsahuje popis iba jedného jediného tlačidla:

```
<?xml version='1.0' encoding='UTF-8'?>
<oor:node xmlns:oor="http://openoffice.org/2001/registry"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
oor:name="Addons" oor:package="org.openoffice.Office">
 <node oor:name="AddonUI">
  <node oor:name="OfficeToolBar">
   <node oor:name="name.JP.addon.SK_SK_Slovnik"
oor:op="replace">

    <node oor:name="btn01" oor:op="replace">
     <prop oor:name="Context" oor:type="xs:string">
      <value>com.sun.star.text.TextDocument</value>
     </prop>
     <prop oor:name="Title" oor:type="xs:string">
      <value xml:lang="sk">Príprava slova pre
slovník</value>
     </prop>
     <prop oor:name="URL" oor:type="xs:string">
      <value>macro:///SK_SK_Slovnik.Slovnik.Skontroluj_slovo
</value>
     </prop>
     <prop oor:name="Target" oor:type="xs:string">
      <value>_self</value>
```

```
      </prop>
      <prop
oor:name="ImageIdentifier"
oor:type="xs:string">
       <value>%origin
%/SK_SK_Slovnik/skontroluj</value>
      </prop>
     </node>

    </node>
   </node>
  </node>
</oor:node>
```



Obrázok 51.9: Tlačidlo "skontroluj_16.bmp"

Pri niektorých položkách sa teraz pristavíme trochu podrobnejšie.

Zápis „<node oor:name="btn01" oor:op="replace">“ určuje vnútorný názov tlačidla („btn01“). Na tomto názve záleží iba v rámci príslušného rozšírenia a dokonca iba v rámci definície jedného panela nástrojov.

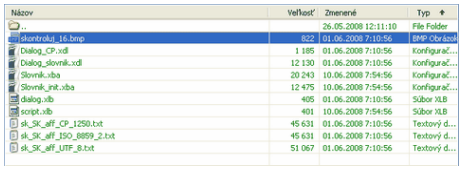
Zápis „<value>com.sun.star.text.TextDocument</value>“ hovorí o tom, že toto tlačidlo panela nástrojov bude prístupné v module Writer. Pre modul Calc by musel mať tvar „<value>com.sun.star.sheet.SpreadsheetDocument</value>“.

Zápis „<prop oor:name="Title" oor:type="xs:string"> <value xml:lang="sk">Príprava slova pre slovník</value>“ je nápomocný text, ktorý sa zobrazí, keď prejdeme nad príslušné tlačidlo.

Zápis „<prop oor:name="URL" oor:type="xs:string"> <value>macro:///SK_SK_Slovnik.Slovnik.Skontroluj_slovo</value>“ hovorí o tom, ktorá procedúra sa pri stlačení tlačidla bude volať.

Zápis <prop oor:name="ImageIdentifier" oor:type="xs:string"> <value>%origin %/SK_SK_Slovnik/skontroluj</value>“ hovorí o tom, ktoré tlačidlo sa bude zobrazovať. Tlačidlá musíme definovať špecificky v BMP formáte. Názov tlačidla je „skontroluj“ (musí byť jednoznačné v rámci príslušného rozšírenia), pričom názov súboru musí byť „skontroluj_16.bmp“ a, pokiaľ chceme definovať aj veľké tlačidlá, tak „skontroluj_26.bmp“. Ako naznačujú čísla pri názvoch, rozmer obrázku musí byť 16x16 (26x26) bodov. Ako priehľadná farba sa používa fialová farba, ako je vidno na zosnímanej definícii tlačidla. Vlastné súbory je potrebné uložiť do adresára, kde je definícia makra.

Keď máme všetky súbory pripravené, označíme ich a vytvoríme z nich ZIP súbor. Tomuto následne premenujeme príponu na OXT a rozšírenie máme hotové. Z makier odstránime knižnicu, z ktorej sme rozšírenie pripravovali a nič nám nebráni v jeho inštalácii.



Obrázok 51.10: Vložené súbory do adresára, kde je definícia makra

51.2. Makrá pre prípravu slov pre slovenský slovník

Dnešný diel seriálu makier bude trochu netradič-

ný, pretože si uvedieme makro pre prácu so slovníkmi. Preto netradičný, lebo sa jedná o veľmi špecifické makro, ktoré je viazané nielen na rozšírenie, pomocou ktorého je šírené, ale aj na konkrétnu verziu tvarovania slov v slovníku slovenského jazyka.

Pre pochopenie, ako je makro vytvorené si musíme aspoň v skratke uviesť, ako sú definované slovníky v programoch OpenOffice.org, Mozilla Thunderbird a Mozilla Firefox. Slovníky sa vlastne skladajú z dvoch súborov. Prvý má príponu „.dic“ a v ňom sú definované vlastné slová v tvare: slovo/tvarovanie (napríklad „tvar/B“, „nakázaný/NY“ ...).

Tvarovanie je vlastne definícia prefixov a sufixov, ktorými sa slovo môže upravovať. Táto definícia sa skladá z piatich častí. Prvá časť (PFX alebo SFX) určuje, či sa jedná o tvarovanie na začiatku (prefix) alebo konci (sufix) slova. Druhá časť je definičné písmeno príslušného tvarovania (to, ktoré je uvedené v slovníku za znakom lomky). Tretia časť je časť slova, ktorá sa nahrádza, štvrtá časť sú znaky, ktorými sa nahrádza a piata časť sú podmienky (definované pomocou regulárnych výrazov).

Napríklad definícia: „SFX z a e [^euo]a“ hovorí, že v slove, ktoré tvarujeme podľa vzoru žena (to hovorí časť „SFX z“) sa nahrádza posledné písmeno „a“ za „e“ ak slovo končí na písmeno „a“, pred ktorým však nesmú byť znaky „eou“. Vďaka tomuto zápisu stačí mať v slovníku slovo „žena/z“ a automaticky sa za správne považuje aj slovo „žene“.

Pri tvarovaní nájdeme aj špecifické tvarovania, kde sa nič nenahrádza, ale iba s pridáva. Vtedy sa namiesto znaku, ktorý sa má nahradiť píše číslo 0. Napríklad definícia „PFX N 0 ne.“ je určená pre zápory, teda definíciou slova „nakázaný/NY“ automaticky definujeme aj slovo „nenakázaný“.

Pred vlastným uvedením makra ešte musíme spomenúť, že ako prvý riadok definícii tvarovania je vždy uvedený ich počet, napríklad „SFX Z Y 157“. Pretože tento sa môže meniť, musíme pri čítaní definičného súboru tento počet zisťovať.

OpenOffice.org pri čítaní textových súborov používa kódovú stránku OpenOffice.org. Preto sme museli pripraviť definičné súbory špeciálne pre rôzne kódové stránky, pretože sa tam, prirodzene, nachádzajú aj znaky s diakritikou. Aby sme tieto súbory nemuseli hľadať, vložili sme ich priamo do rozšírenia.

Rozsah makier je naozaj veľký a preto ho rozdelíme do viacerých dielov tohto seriálu. Dnes začneme funkciami, ktoré sú uložené v module „Slovnik_init“ a sú určené pre počiatočnú prípravu rozšírenia pre spracovanie slova. Upozorňujeme, že makrá nie sú veľmi okomentované a preto sú určené pre vyspelejších programátorov.

Tak isto sa nebudeme bližšie zaoberať dialógmi, pretože to, ako sú definované (ich premenné) sa dá pozrieť v ich definícii.

```
sub slovnik_init(oDialog)
 dim Kurzor, Povodny as object
 dim riadok, slovo, parametre as string
 dim i, kolko as integer

 kurzor= ThisComponent.currentcontroller.getViewCursor() '
viditeľný kurzor
 Povodny=kurzor.getstart()
 kurzor.collapsetostart()
```

```
 Kurzor.gotoStartOfLine(false)
 Kurzor.gotoEndOfLine(true)
 riadok=trim(Kurzor.getstring())
 Kurzor.collapseToEnd()
 kurzor.gotoRange(Povodny,false)

 if riadok<>"" then
  slovo=riadok
  parametre=""
  i=instr(riadok,"/")
  if i<>0 then
   slovo=left(riadok,i-1)
   parametre=right(riadok,len(riadok)-i)
  end if
  oDialog_slovnik.model.Slovo_slovnika.Text=slovo

  if instr(1,parametre,"N",0)<>0 then
oDialog_slovnik.model.pfx_N.State=1
  if instr(1,parametre,"F",0)<>0 then
oDialog_slovnik.model.pfx_F.State=1
  if instr(1,parametre,"Z",0)<>0 then
oDialog_slovnik.model.sfx_Z.State=1
  if instr(1,parametre,"z",0)<>0 then
oDialog_slovnik.model.sfx_mz.State=1
  if instr(1,parametre,"U",0)<>0 then
oDialog_slovnik.model.sfx_U.State=1
  if instr(1,parametre,"D",0)<>0 then
oDialog_slovnik.model.sfx_D.State=1
  if instr(1,parametre,"K",0)<>0 then
oDialog_slovnik.model.sfx_K.State=1
  if instr(1,parametre,"M",0)<>0 then
oDialog_slovnik.model.sfx_M.State=1
  if instr(1,parametre,"S",0)<>0 then
oDialog_slovnik.model.sfx_S.State=1
  if instr(1,parametre,"V",0)<>0 then
oDialog_slovnik.model.sfx_V.State=1
  if instr(1,parametre,"A",0)<>0 then
oDialog_slovnik.model.sfx_A.State=1
  if instr(1,parametre,"C",0)<>0 then
oDialog_slovnik.model.sfx_C.State=1
  if instr(1,parametre,"H",0)<>0 then
oDialog_slovnik.model.sfx_H.State=1
  if instr(1,parametre,"B",0)<>0 then
oDialog_slovnik.model.sfx_B.State=1
  if instr(1,parametre,"b",0)<>0 then
oDialog_slovnik.model.sfx_mb.State=1
  if instr(1,parametre,"o",0)<>0 then
oDialog_slovnik.model.sfx_mo.State=1
  if instr(1,parametre,"O",0)<>0 then
```

```
oDialog_slovnik.model.sfx_O.State=1
  if instr(1,parametre,"J",0)<>0 then
oDialog_slovnik.model.sfx_J.State=1
  if instr(1,parametre,"L",0)<>0 then
oDialog_slovnik.model.sfx_L.State=1
  if instr(1,parametre,"Q",0)<>0 then
oDialog_slovnik.model.sfx_Q.State=1
  if instr(1,parametre,"Y",0)<>0 then
oDialog_slovnik.model.sfx_Y.State=1
  if instr(1,parametre,"I",0)<>0 then
oDialog_slovnik.model.sfx_I.State=1
  if instr(1,parametre,"P",0)<>0 then
oDialog_slovnik.model.sfx_P.State=1
  if instr(1,parametre,"X",0)<>0 then
oDialog_slovnik.model.sfx_X.State=1
  if instr(1,parametre,"E",0)<>0 then
oDialog_slovnik.model.sfx_E.State=1
  if instr(1,parametre,"W",0)<>0 then
oDialog_slovnik.model.sfx_W.State=1
  if instr(1,parametre,"T",0)<>0 then
oDialog_slovnik.model.sfx_T.State=1
  if instr(1,parametre,"R",0)<>0 then
oDialog_slovnik.model.sfx_R.State=1
  if instr(1,parametre,"č",0)<>0 then
oDialog_slovnik.model.sfx_c_c.State=1

 end if
end sub

function citaj_definiciu as boolean

 ' dim cesty as object
 dim subor as integer
 dim riadok as string
 dim prvy as long
 dim cesta as string
 dim nacital as boolean
 dim nazov as string

 'cesty=CreateUnoService("com.sun.star.util.PathSettings")

 ' DialogLibraries.LoadLibrary("SK_SK_Slovnik")

 oDialog_CP=CreateUnoDialog(DialogLibraries.SK_SK_Slovnik.D
ialog_CP)

 oDialog_CP.model.UTF_8.state=1
 oDialog_CP.model.CP_1250.state=0
 oDialog_CP.model.ISO_8859_2.state=0
```

```
 nacital=false

 if oDialog_CP.Execute()=1 then
  if oDialog_CP.model.UTF_8.state=1 then
nazov="sk_SK_aff_UTF_8.txt"
  if oDialog_CP.model.CP_1250.state=1 then
nazov="sk_SK_aff_CP_1250.txt"
  if oDialog_CP.model.ISO_8859_2.state=1 then
nazov="sk_SK_aff_ISO_8859_2.txt"
  REM Cesta, kde je nainštalované rozšírenie SK_SK_Slovnik
- treba si najprv nechať vypísať, aby sa zistilo, čo je
potrebné ponechať
  cesta=basiclibraries.getLibraryLinkURL("SK_SK_Slovnik")
  cesta=left(cesta,len(cesta)-10)+nazov
  subor = Freefile
  open cesta for Input as #subor
  'open cesty.Work+"/sk_SK_aff.txt" for Input as #subor
  prvy=0
  While not eof(subor)
   Line Input #subor, riadok
   prvy=spracuj_riadok(riadok,prvy)
  wend
  close #subor
  nacital=true
 end if
 oDialog_CP.dispose()
 citaj_definiciu=nacital

end function
```

V minulom dieli sme si uviedli prvú časť funkcií, ktoré sú uložené v module „Slovnik_init“ a sú určené pre počiatočnú prípravu rozšírenia pre spracovanie slova. Dnes budeme pokračovať týmto modulom a uvedieme si funkciu, ktorá číta definičný súbor tvarovania slov, podľa počtu definícii mení dimenziu polí a načítava príslušné časti tvarovania.

```
function spracuj_riadok(riadok as string, prvy as long) as
long

 dim prefix_sufix, co, za, podmienka as string
 dim poznamka as boolean
 dim znak, cast as string
 dim dlzka, i, ktora as integer
 dim kolko, pomprvy as long

 dlzka=len(riadok)
 pomprvy=prvy

 cast=""
```

```
 poznamka=false
 ktora=0
 for i=1 to dlzka
  if not poznamka then
   znak=mid(riadok,i,1)
   if znak="#" then
    poznamka=true
   else
    if znak=" " then
     ktora=ktora+1
     ' Ukončenie definície jednej časti
     select case ktora
      case 1 ' Či ide o prefix/sufix
       rem Toto nespracovávame
       cast=""
      case 2 ' Vlastný prefix/sufix
       prefix_sufix=cast
       cast=""
      case 3 ' Ak je prvý, tak Y/N, inak "co"
       if pomprvy<>0 then
        co=cast
       end if
       cast=""
      case 4 ' Ak je prvý, tak počet, inak "za"
       if pomprvy=0 then
        kolko=val(cast)
       else
        za=cast
       end if
       cast=""
      case 5 ' Podmienka
       podmienka=cast
       cast=""
     end select
    else
     cast=cast+znak
    end if
   end if
  end if
 next i
 if (pomprvy=0) and (ktora=3) then
  kolko=val(cast)
  cast=""
 end if
 if (pomprvy<>0) and (ktora=4) then
  ' Koniec definície bez poznámky alebo medzery na konci
riadku
  podmienka=cast
  cast=""
```

```
end if
if pomprvy=0 then
 def_pfx_sfx=0
 pomprvy=kolko
 select case prefix_sufix
  case "N"
   redim preserve pfx_N_d(3,pomprvy)
   pfx_N_p=pomprvy
  case "F"
   redim preserve pfx_F_d(3,pomprvy)
   pfx_F_p=pomprvy
  case "Z"
   redim preserve sfx_Z_d(3,pomprvy)
   sfx_Z_p=pomprvy
  case "z"
   redim preserve sfx_mz_d(3,pomprvy)
   sfx_mz_p=pomprvy
  case "U"
   redim preserve sfx_U_d(3,pomprvy)
   sfx_U_p=pomprvy
  case "D"
   redim preserve sfx_D_d(3,pomprvy)
   sfx_D_p=pomprvy
  case "K"
   redim preserve sfx_K_d(3,pomprvy)
   sfx_K_p=pomprvy
  case "M"
   redim preserve sfx_M_d(3,pomprvy)
   sfx_M_p=pomprvy
  case "S"
   redim preserve sfx_S_d(3,pomprvy)
   sfx_S_p=pomprvy
  case "V"
   redim preserve sfx_V_d(3,pomprvy)
   sfx_V_p=pomprvy
  case "A"
   redim preserve sfx_A_d(3,pomprvy)
   sfx_A_p=pomprvy
  case "C"
   redim preserve sfx_C_d(3,pomprvy)
   sfx_C_p=pomprvy
  case "H"
   redim preserve sfx_H_d(3,pomprvy)
   sfx_H_p=pomprvy
  case "B"
   redim preserve sfx_B_d(3,pomprvy)
   sfx_B_p=pomprvy
  case "b"
   redim preserve sfx_mb_d(3,pomprvy)
```

```
     sfx_mb_p=pomprvy
    case "o"
     redim preserve sfx_mo_d(3,pomprvy)
     sfx_mo_p=pomprvy
    case "O"
     redim preserve sfx_O_d(3,pomprvy)
     sfx_O_p=pomprvy
    case "J"
     redim preserve sfx_J_d(3,pomprvy)
     sfx_J_p=pomprvy
    case "L"
     redim preserve sfx_L_d(3,pomprvy)
     sfx_L_p=pomprvy
    case "Q"
     redim preserve sfx_Q_d(3,pomprvy)
     sfx_Q_p=pomprvy
    case "Y"
     redim preserve sfx_Y_d(3,pomprvy)
     sfx_Y_p=pomprvy
    case "I"
     redim preserve sfx_I_d(3,pomprvy)
     sfx_I_p=pomprvy
    case "P"
     redim preserve sfx_P_d(3,pomprvy)
     sfx_P_p=pomprvy
    case "X"
     redim preserve sfx_X_d(3,pomprvy)
     sfx_X_p=pomprvy
    case "E"
     redim preserve sfx_E_d(3,pomprvy)
     sfx_E_p=pomprvy
    case "W"
     redim preserve sfx_W_d(3,pomprvy)
     sfx_W_p=pomprvy
    case "T"
     redim preserve sfx_T_d(3,pomprvy)
     sfx_T_p=pomprvy
    case "R"
     redim preserve sfx_R_d(3,pomprvy)
     sfx_R_p=pomprvy
    case "č"
     redim preserve sfx_c_c_d(3,pomprvy)
     sfx_c_c_p=pomprvy
  end select
 else
  def_pfx_sfx=def_pfx_sfx+1
  if pomprvy=def_pfx_sfx then
   ' Spracovávame posledný riadok, bude nový
   pomprvy=0
```

```
end if
if co="0" then co=""
if za="0" then za=""
select case prefix_sufix
 case "N"
  pfx_N_d(1,def_pfx_sfx)=co
  pfx_N_d(2,def_pfx_sfx)=za
  pfx_N_d(3,def_pfx_sfx)=podmienka
 case "F"
  pfx_F_d(1,def_pfx_sfx)=co
  pfx_F_d(2,def_pfx_sfx)=za
  pfx_F_d(3,def_pfx_sfx)=podmienka
 case "Z"
  sfx_Z_d(1,def_pfx_sfx)=co
  sfx_Z_d(2,def_pfx_sfx)=za
  sfx_Z_d(3,def_pfx_sfx)=podmienka
 case "z"
  sfx_mz_d(1,def_pfx_sfx)=co
  sfx_mz_d(2,def_pfx_sfx)=za
  sfx_mz_d(3,def_pfx_sfx)=podmienka
 case "U"
  sfx_U_d(1,def_pfx_sfx)=co
  sfx_U_d(2,def_pfx_sfx)=za
  sfx_U_d(3,def_pfx_sfx)=podmienka
 case "D"
  sfx_D_d(1,def_pfx_sfx)=co
  sfx_D_d(2,def_pfx_sfx)=za
  sfx_D_d(3,def_pfx_sfx)=podmienka
 case "K"
  sfx_K_d(1,def_pfx_sfx)=co
  sfx_K_d(2,def_pfx_sfx)=za
  sfx_K_d(3,def_pfx_sfx)=podmienka
 case "M"
  sfx_M_d(1,def_pfx_sfx)=co
  sfx_M_d(2,def_pfx_sfx)=za
  sfx_M_d(3,def_pfx_sfx)=podmienka
 case "S"
  sfx_S_d(1,def_pfx_sfx)=co
  sfx_S_d(2,def_pfx_sfx)=za
  sfx_S_d(3,def_pfx_sfx)=podmienka
 case "V"
  sfx_V_d(1,def_pfx_sfx)=co
  sfx_V_d(2,def_pfx_sfx)=za
  sfx_V_d(3,def_pfx_sfx)=podmienka
 case "A"
  sfx_A_d(1,def_pfx_sfx)=co
  sfx_A_d(2,def_pfx_sfx)=za
  sfx_A_d(3,def_pfx_sfx)=podmienka
 case "C"
```

```
 sfx_C_d(1,def_pfx_sfx)=co
 sfx_C_d(2,def_pfx_sfx)=za
 sfx_C_d(3,def_pfx_sfx)=podmienka
case "H"
 sfx_H_d(1,def_pfx_sfx)=co
 sfx_H_d(2,def_pfx_sfx)=za
 sfx_H_d(3,def_pfx_sfx)=podmienka
case "B"
 sfx_B_d(1,def_pfx_sfx)=co
 sfx_B_d(2,def_pfx_sfx)=za
 sfx_B_d(3,def_pfx_sfx)=podmienka
case "b"
 sfx_mb_d(1,def_pfx_sfx)=co
 sfx_mb_d(2,def_pfx_sfx)=za
 sfx_mb_d(3,def_pfx_sfx)=podmienka
case "o"
 sfx_mo_d(1,def_pfx_sfx)=co
 sfx_mo_d(2,def_pfx_sfx)=za
 sfx_mo_d(3,def_pfx_sfx)=podmienka
case "O"
 sfx_O_d(1,def_pfx_sfx)=co
 sfx_O_d(2,def_pfx_sfx)=za
 sfx_O_d(3,def_pfx_sfx)=podmienka
case "J"
 sfx_J_d(1,def_pfx_sfx)=co
 sfx_J_d(2,def_pfx_sfx)=za
 sfx_J_d(3,def_pfx_sfx)=podmienka
case "L"
 sfx_L_d(1,def_pfx_sfx)=co
 sfx_L_d(2,def_pfx_sfx)=za
 sfx_L_d(3,def_pfx_sfx)=podmienka
case "Q"
 sfx_Q_d(1,def_pfx_sfx)=co
 sfx_Q_d(2,def_pfx_sfx)=za
 sfx_Q_d(3,def_pfx_sfx)=podmienka
case "Y"
 sfx_Y_d(1,def_pfx_sfx)=co
 sfx_Y_d(2,def_pfx_sfx)=za
 sfx_Y_d(3,def_pfx_sfx)=podmienka
case "I"
 sfx_I_d(1,def_pfx_sfx)=co
 sfx_I_d(2,def_pfx_sfx)=za
 sfx_I_d(3,def_pfx_sfx)=podmienka
case "P"
 sfx_P_d(1,def_pfx_sfx)=co
 sfx_P_d(2,def_pfx_sfx)=za
 sfx_P_d(3,def_pfx_sfx)=podmienka
case "X"
 sfx_X_d(1,def_pfx_sfx)=co
```

```
    sfx_X_d(2,def_pfx_sfx)=za
    sfx_X_d(3,def_pfx_sfx)=podmienka
   case "E"
    sfx_E_d(1,def_pfx_sfx)=co
    sfx_E_d(2,def_pfx_sfx)=za
    sfx_E_d(3,def_pfx_sfx)=podmienka
   case "W"
    sfx_W_d(1,def_pfx_sfx)=co
    sfx_W_d(2,def_pfx_sfx)=za
    sfx_W_d(3,def_pfx_sfx)=podmienka
   case "T"
    sfx_T_d(1,def_pfx_sfx)=co
    sfx_T_d(2,def_pfx_sfx)=za
    sfx_T_d(3,def_pfx_sfx)=podmienka
   case "R"
    sfx_R_d(1,def_pfx_sfx)=co
    sfx_R_d(2,def_pfx_sfx)=za
    sfx_R_d(3,def_pfx_sfx)=podmienka
   case "č"
    sfx_c_c_d(1,def_pfx_sfx)=co
    sfx_c_c_d(2,def_pfx_sfx)=za
    sfx_c_c_d(3,def_pfx_sfx)=podmienka
  end select
 end if
 spracuj_riadok=pomprvy
end function
```

Na úvod dnešného dielu seriálu o makrách v OpenOffice.org, v ktorom si predstavujeme rozšírenie pre prípravu slov do slovenského slovníka si ukážeme definíciu globálnych premenných (ktoré sme už používali) a funkcie, pomocou ktorých „tvarujeme“ zadané slovo. V podstate sa na to využíva funkcia hľadania a výmeny, pretože podmienky sú definované ako regulárne výrazy. Je to teda praktická ukážka, ako sa dajú využiť takéto funkcie aj na pomerne komplikované účely.

```
global pfx_N_d(3,0) as string
global pfx_F_d(3,0) as string

global sfx_mz_d(3,0) as string
global sfx_Z_d(3,0) as string
global sfx_U_d(3,0) as string
global sfx_D_d(3,0) as string
global sfx_K_d(3,0) as string
global sfx_M_d(3,0) as string
global sfx_S_d(3,0) as string
global sfx_V_d(3,0) as string
global sfx_A_d(3,0) as string
global sfx_C_d(3,0) as string
global sfx_H_d(3,0) as string
global sfx_B_d(3,0) as string
global sfx_J_d(3,0) as string
```

```
global sfx_L_d(3,0) as string
global sfx_O_d(3,0) as string
global sfx_Q_d(3,0) as string
global sfx_Y_d(3,0) as string
global sfx_I_d(3,0) as string
global sfx_P_d(3,0) as string
global sfx_X_d(3,0) as string
global sfx_E_d(3,0) as string
global sfx_W_d(3,0) as string
global sfx_T_d(3,0) as string
global sfx_R_d(3,0) as string
global sfx_mb_d(3,0) as string
global sfx_c_c_d(3,0) as string
global sfx_mo_d(3,0) as string

global pfx_N_p as long
global pfx_F_p as long

global sfx_mz_p as long
global sfx_Z_p as long
global sfx_U_p as long
global sfx_D_p as long
global sfx_K_p as long
global sfx_M_p as long
global sfx_S_p as long
global sfx_V_p as long
global sfx_A_p as long
global sfx_C_p as long
global sfx_H_p as long
global sfx_B_p as long
global sfx_J_p as long
global sfx_L_p as long
global sfx_O_p as long
global sfx_Q_p as long
global sfx_Y_p as long
global sfx_I_p as long
global sfx_P_p as long
global sfx_X_p as long
global sfx_E_p as long
global sfx_W_p as long
global sfx_T_p as long
global sfx_R_p as long
global sfx_mb_p as long
global sfx_c_c_p as long
global sfx_mo_p as long

global def_pfx_sfx as integer
global oDialog_slovnik as object
global nacitana_definicia as boolean
```

```
function oNovy_subor()
 dim url as string
 url="private:factory/swriter"
 oNovy_subor=StarDesktop.loadComponentFromURL(url,
"_blank", 0, Array())
end function

function fZamen_Vo_Vnutri (V_Com, Co, Za as string) as
string
 dim pom as string
 dim i, j as long
 pom=V_Com
 i=instr(V_Com,Co)
 j=len(V_Com)-i-len(Co)+1
 if i>0 then
  pom=left(V_Com,i-1)+Za
  if j>0 then
   pom=pom+right(V_Com,j)
  end if
 end if
 fZamen_Vo_Vnutri=pom
end function

sub kon_zoznam (odokument, slovo, sufix, kolko)
 dim kurzor, nasiel, oVymen as object
 dim pomslovo, co, za as string
 dim i as integer

 pomslovo=slovo+" "
 oVymen=odokument.createSearchDescriptor()
 oVymen.SearchRegularExpression=true
 oVymen.SearchWords=false
 oVymen.SearchCaseSensitive=false

 kurzor=odokument.currentcontroller.getViewCursor()
 ' Základné slovo

 kurzor.gotoStart(false)
 kurzor.setstring(pomslovo)
 kurzor.gotoStart(false)

 ' Sufixy
 for i=1 to kolko
  kurzor.gotoStart(false)
  kurzor.setstring(pomslovo)
  kurzor.gotoStart(true)
  oVymen.SearchString=sufix(3,i)+"\>"
  nasiel=odokument.findFirst(oVymen)
```

```
  if NOT isNull(nasiel) then
   co=sufix(1,i)
   za=sufix(2,i)
   if co="" then
    co=nasiel.String
    za=co+sufix(2,i)
   end if
   if right(trim(pomslovo),len(co))=co then
    nasiel.String=fZamen_Vo_Vnutri(nasiel.String,co,za)
   else
    kurzor.setstring("")
   end if
  else
   kurzor.setstring("")
  end if
 next i
end sub

sub spracuj_sufixy(odokument, slovo)
 kon_zoznam(odokument, slovo, array(), 0)
 if oDialog_slovnik.model.sfx_mz.State=1 then kon_zoznam
(odokument, slovo, sfx_mz_d, sfx_mz_p)
 if oDialog_slovnik.model.sfx_Z.State=1 then kon_zoznam
(odokument, slovo, sfx_Z_d, sfx_Z_p)
 if oDialog_slovnik.model.sfx_U.State=1 then kon_zoznam
(odokument, slovo, sfx_U_d, sfx_U_p)
 if oDialog_slovnik.model.sfx_D.State=1 then kon_zoznam
(odokument, slovo, sfx_D_d, sfx_D_p)
 if oDialog_slovnik.model.sfx_K.State=1 then kon_zoznam
(odokument, slovo, sfx_K_d, sfx_K_p)
 if oDialog_slovnik.model.sfx_M.State=1 then kon_zoznam
(odokument, slovo, sfx_M_d, sfx_M_p)
 if oDialog_slovnik.model.sfx_S.State=1 then kon_zoznam
(odokument, slovo, sfx_S_d, sfx_S_p)
 if oDialog_slovnik.model.sfx_V.State=1 then kon_zoznam
(odokument, slovo, sfx_V_d, sfx_V_p)
 if oDialog_slovnik.model.sfx_A.State=1 then kon_zoznam
(odokument, slovo, sfx_A_d, sfx_A_p)
 if oDialog_slovnik.model.sfx_C.State=1 then kon_zoznam
(odokument, slovo, sfx_C_d, sfx_C_p)
 if oDialog_slovnik.model.sfx_H.State=1 then kon_zoznam
(odokument, slovo, sfx_H_d, sfx_H_p)
 if oDialog_slovnik.model.sfx_B.State=1 then kon_zoznam
(odokument, slovo, sfx_B_d, sfx_B_p)
 if oDialog_slovnik.model.sfx_J.State=1 then kon_zoznam
(odokument, slovo, sfx_J_d, sfx_J_p)
 if oDialog_slovnik.model.sfx_L.State=1 then kon_zoznam
(odokument, slovo, sfx_L_d, sfx_L_p)
 if oDialog_slovnik.model.sfx_O.State=1 then kon_zoznam
```

```
(odokument, slovo, sfx_O_d, sfx_O_p)
 if oDialog_slovnik.model.sfx_Q.State=1 then kon_zoznam
(odokument, slovo, sfx_Q_d, sfx_Q_p)
 if oDialog_slovnik.model.sfx_Y.State=1 then kon_zoznam
(odokument, slovo, sfx_Y_d, sfx_Y_p)
 if oDialog_slovnik.model.sfx_I.State=1 then kon_zoznam
(odokument, slovo, sfx_I_d, sfx_I_p)
 if oDialog_slovnik.model.sfx_P.State=1 then kon_zoznam
(odokument, slovo, sfx_P_d, sfx_P_p)
 if oDialog_slovnik.model.sfx_X.State=1 then kon_zoznam
(odokument, slovo, sfx_X_d, sfx_X_p)
 if oDialog_slovnik.model.sfx_E.State=1 then kon_zoznam
(odokument, slovo, sfx_E_d, sfx_E_p)
 if oDialog_slovnik.model.sfx_W.State=1 then kon_zoznam
(odokument, slovo, sfx_W_d, sfx_W_p)
 if oDialog_slovnik.model.sfx_T.State=1 then kon_zoznam
(odokument, slovo, sfx_T_d, sfx_T_p)
 if oDialog_slovnik.model.sfx_R.State=1 then kon_zoznam
(odokument, slovo, sfx_R_d, sfx_R_p)
 if oDialog_slovnik.model.sfx_mb.State=1 then kon_zoznam
(odokument, slovo, sfx_mb_d, sfx_mb_p)
 if oDialog_slovnik.model.sfx_mo.State=1 then kon_zoznam
(odokument, slovo, sfx_mo_d, sfx_mo_p)
 if oDialog_slovnik.model.sfx_c_c.State=1 then kon_zoznam
(odokument, slovo, sfx_c_c_d, sfx_c_c_p)

end sub

sub vytvor_zoznam
 dim dokument, kurzor, nasiel, oVymen as object
 dim url, slovo, pomslovo as string
 dim i as integer

 slovo=trim(oDialog_slovnik.model.Slovo_slovnika.Text)
 if slovo<>"" then
  url="private:factory/swriter"
  dokument=StarDesktop.loadComponentFromURL(url, "_blank",
0, Array())
  ' Základné slovo
  spracuj_sufixy(dokument, slovo)
  ' Pre všetky prefixy
  if oDialog_slovnik.model.pfx_N.State=1 then
   for i=1 to pfx_N_p
    spracuj_sufixy(dokument, pfx_N_d(2,i)+slovo)
   next i
  end if
  if oDialog_slovnik.model.pfx_F.State=1 then
   for i=1 to pfx_F_p
    spracuj_sufixy(dokument, pfx_F_d(2,i)+slovo)
```

```
    next i
  end if
  if oDialog_slovnik.model.spfx_do.State=1 then
   spracuj_sufixy(dokument, "do"+slovo)
   if oDialog_slovnik.model.spfx_ne_do.State=1 then
    spracuj_sufixy(dokument, "nedo"+slovo)
   end if
  end if
  if oDialog_slovnik.model.spfx_na.State=1 then
   spracuj_sufixy(dokument, "na"+slovo)
   if oDialog_slovnik.model.spfx_ne_na.State=1 then
    spracuj_sufixy(dokument, "nena"+slovo)
   end if
  end if
  if oDialog_slovnik.model.spfx_nad.State=1 then
   spracuj_sufixy(dokument, "nad"+slovo)
   if oDialog_slovnik.model.spfx_ne_nad.State=1 then
    spracuj_sufixy(dokument, "nenad"+slovo)
   end if
  end if
  if oDialog_slovnik.model.spfx_o.State=1 then
   spracuj_sufixy(dokument, "o"+slovo)
   if oDialog_slovnik.model.spfx_ne_o.State=1 then
    spracuj_sufixy(dokument, "neo"+slovo)
   end if
  end if
  if oDialog_slovnik.model.spfx_ob.State=1 then
   spracuj_sufixy(dokument, "ob"+slovo)
   if oDialog_slovnik.model.spfx_ne_ob.State=1 then
    spracuj_sufixy(dokument, "neob"+slovo)
   end if
  end if
  if oDialog_slovnik.model.spfx_po.State=1 then
   spracuj_sufixy(dokument, "po"+slovo)
   if oDialog_slovnik.model.spfx_ne_po.State=1 then
    spracuj_sufixy(dokument, "nepo"+slovo)
   end if
  end if
  if oDialog_slovnik.model.spfx_pre.State=1 then
   spracuj_sufixy(dokument, "pre"+slovo)
   if oDialog_slovnik.model.spfx_ne_pre.State=1 then
    spracuj_sufixy(dokument, "nepre"+slovo)
   end if
  end if
  if oDialog_slovnik.model.spfx_pri.State=1 then
   spracuj_sufixy(dokument, "pri"+slovo)
   if oDialog_slovnik.model.spfx_ne_pri.State=1 then
    spracuj_sufixy(dokument, "nepri"+slovo)
   end if
```

```
end if
if oDialog_slovnik.model.spfx_spolu.State=1 then
 spracuj_sufixy(dokument, "spolu"+slovo)
 if oDialog_slovnik.model.spfx_ne_spolu.State=1 then
  spracuj_sufixy(dokument, "nespolu"+slovo)
 end if
end if
if oDialog_slovnik.model.spfx_roz.State=1 then
 spracuj_sufixy(dokument, "roz"+slovo)
 if oDialog_slovnik.model.spfx_ne_roz.State=1 then
  spracuj_sufixy(dokument, "neroz"+slovo)
 end if
end if
if oDialog_slovnik.model.spfx_spo.State=1 then
 spracuj_sufixy(dokument, "spo"+slovo)
 if oDialog_slovnik.model.spfx_ne_spo.State=1 then
  spracuj_sufixy(dokument, "nespo"+slovo)
 end if
end if
if oDialog_slovnik.model.spfx_u.State=1 then
 spracuj_sufixy(dokument, "u"+slovo)
 if oDialog_slovnik.model.spfx_ne_u.State=1 then
  spracuj_sufixy(dokument, "neu"+slovo)
 end if
end if
if oDialog_slovnik.model.spfx_vy.State=1 then
 spracuj_sufixy(dokument, "vy"+slovo)
 if oDialog_slovnik.model.spfx_ne_vy.State=1 then
  spracuj_sufixy(dokument, "nevy"+slovo)
 end if
end if
if oDialog_slovnik.model.spfx_vy_y.State=1 then
 spracuj_sufixy(dokument, "vý"+slovo)
 if oDialog_slovnik.model.spfx_ne_vy_y.State=1 then
  spracuj_sufixy(dokument, "nevý"+slovo)
 end if
end if
if oDialog_slovnik.model.spfx_z.State=1 then
 spracuj_sufixy(dokument, "z"+slovo)
 if oDialog_slovnik.model.spfx_ne_z.State=1 then
  spracuj_sufixy(dokument, "nez"+slovo)
 end if
end if
if oDialog_slovnik.model.spfx_za.State=1 then
 spracuj_sufixy(dokument, "za"+slovo)
 if oDialog_slovnik.model.spfx_ne_za.State=1 then
  spracuj_sufixy(dokument, "neza"+slovo)
 end if
end if
```

```
 end if
end sub
```

V poslednom pokračovaní makier pre prípravu slov do slovenského slovníka si ukážeme, ako pripravené slovo zapíšeme do výstupného súboru.

Na koniec seriálu makier, ktoré sú určené pre prípravu slov do slovenského slovníka už iba spomeňme, že v makre sme definovali aj také prefixy, ktoré nie sú súčasťou definičného súboru (do, na, nad...). Preto tieto prefixy musíme manuálne pridať k základnému slovu a tak dokážeme pripraviť jediným volaním viacero slov do slovníka.

```
function otvor_subor_sk_SK_odt (oSubor) as boolean
 dim cesty as object
 dim url as string
 dim novy as boolean

 cesty=CreateUnoService("com.sun.star.util.PathSettings")

 url=cesty.Work+"/sk_SK.odt"
 novy=not FileExists(url)
 if novy then
  url="private:factory/swriter"
 end if
 oSubor=StarDesktop.loadComponentFromURL(url, "_blank", 0,
Array())
 otvor_subor_sk_SK_odt=novy
end function

sub uloz_subor_sk_SK_odt (oSubor, novy)
 dim cesty, dispatcher as object
 dim url as string

 if novy then
  cesty=CreateUnoService("com.sun.star.util.PathSettings")
  url = cesty.Work+"/sk_SK.odt"
  oSubor.storeAsUrl(url, Array())
 else
  oSubor.store()
 end if
 'oSubor.dispose()
end sub

function daj_flagy as string
 dim pom as string
 pom=""
 if oDialog_slovnik.model.pfx_F.State=1 then pom=pom+"F"
 if oDialog_slovnik.model.sfx_mz.State=1 then pom=pom+"z"
 if oDialog_slovnik.model.sfx_Z.State=1 then pom=pom+"Z"
 if oDialog_slovnik.model.sfx_U.State=1 then pom=pom+"U"
 if oDialog_slovnik.model.sfx_D.State=1 then pom=pom+"D"
```

```
 if oDialog_slovnik.model.sfx_K.State=1 then pom=pom+"K"
 if oDialog_slovnik.model.sfx_M.State=1 then pom=pom+"M"
 if oDialog_slovnik.model.sfx_S.State=1 then pom=pom+"S"
 if oDialog_slovnik.model.sfx_V.State=1 then pom=pom+"V"
 if oDialog_slovnik.model.sfx_A.State=1 then pom=pom+"A"
 if oDialog_slovnik.model.sfx_C.State=1 then pom=pom+"C"
 if oDialog_slovnik.model.sfx_H.State=1 then pom=pom+"H"
 if oDialog_slovnik.model.sfx_B.State=1 then pom=pom+"B"
 if oDialog_slovnik.model.sfx_J.State=1 then pom=pom+"J"
 if oDialog_slovnik.model.sfx_L.State=1 then pom=pom+"L"
 if oDialog_slovnik.model.sfx_O.State=1 then pom=pom+"O"
 if oDialog_slovnik.model.sfx_mo.State=1 then pom=pom+"o"
 if oDialog_slovnik.model.sfx_Q.State=1 then pom=pom+"Q"
 if oDialog_slovnik.model.sfx_Y.State=1 then pom=pom+"Y"
 if oDialog_slovnik.model.sfx_I.State=1 then pom=pom+"I"
 if oDialog_slovnik.model.sfx_P.State=1 then pom=pom+"P"
 if oDialog_slovnik.model.sfx_X.State=1 then pom=pom+"X"
 if oDialog_slovnik.model.sfx_E.State=1 then pom=pom+"E"
 if oDialog_slovnik.model.sfx_W.State=1 then pom=pom+"W"
 if oDialog_slovnik.model.sfx_T.State=1 then pom=pom+"T"
 if oDialog_slovnik.model.sfx_R.State=1 then pom=pom+"R"
 if oDialog_slovnik.model.sfx_mb.State=1 then pom=pom+"b"
 if oDialog_slovnik.model.sfx_c_c.State=1 then pom=pom+"č"
 daj_flagy=pom
end function

sub zapis_definiciu
 dim vysledok, slovo, pomslovo, sufixy as string
 dim dokument, kurzor, oDoc as object
 dim novy as boolean

 slovo=trim(oDialog_slovnik.model.Slovo_slovnika.Text)
 vysledok = ""
 if slovo<>"" then
  sufixy=daj_flagy
  vysledok=sufixy
  if oDialog_slovnik.model.pfx_N.State=1 then
vysledok="N"+vysledok
  if vysledok<>"" then vysledok="/"+vysledok
  vysledok=slovo+vysledok
  novy=otvor_subor_sk_SK_odt(dokument)

  ' Základné slovo
  kurzor=dokument.currentcontroller.getViewCursor()
  kurzor.gotoEnd(false)
  if novy then
   kurzor.setstring("# verzia: 0.5.7-b2"+chr$(13))
   kurzor.gotoEnd(false)
  end if
```

```
  kurzor.setstring(vysledok+chr$(13))
  kurzor.gotoEnd(false)

  ' Pre všetky prefixy

  vysledok=sufixy
  if oDialog_slovnik.model.spfx_do.State=1 then
   pomslovo="do"+slovo
   if oDialog_slovnik.model.spfx_ne_do.State=1 then
    vysledok="N"+vysledok
   end if
   if vysledok<>"" then vysledok="/"+vysledok
   vysledok=pomslovo+vysledok
   kurzor.setstring(vysledok+chr$(13))
   kurzor.gotoEnd(false)
  end if
  vysledok=sufixy
  if oDialog_slovnik.model.spfx_na.State=1 then
   pomslovo="na"+slovo
   if oDialog_slovnik.model.spfx_ne_na.State=1 then
    vysledok="N"+vysledok
   end if
   if vysledok<>"" then vysledok="/"+vysledok
   vysledok=pomslovo+vysledok
   kurzor.setstring(vysledok+chr$(13))
   kurzor.gotoEnd(false)
  end if
  vysledok=sufixy
  if oDialog_slovnik.model.spfx_nad.State=1 then
   pomslovo="nad"+slovo
   if oDialog_slovnik.model.spfx_ne_nad.State=1 then
    vysledok="N"+vysledok
   end if
   if vysledok<>"" then vysledok="/"+vysledok
   vysledok=pomslovo+vysledok
   kurzor.setstring(vysledok+chr$(13))
   kurzor.gotoEnd(false)
  end if
  vysledok=sufixy
  if oDialog_slovnik.model.spfx_o.State=1 then
   pomslovo="o"+slovo
   if oDialog_slovnik.model.spfx_ne_o.State=1 then
    vysledok="N"+vysledok
   end if
   if vysledok<>"" then vysledok="/"+vysledok
   vysledok=pomslovo+vysledok
   kurzor.setstring(vysledok+chr$(13))
   kurzor.gotoEnd(false)
  end if
```

```
vysledok=sufixy
if oDialog_slovnik.model.spfx_ob.State=1 then
 pomslovo="ob"+slovo
 if oDialog_slovnik.model.spfx_ne_ob.State=1 then
  vysledok="N"+vysledok
 end if
 if vysledok<>"" then vysledok="/"+vysledok
 vysledok=pomslovo+vysledok
 kurzor.setstring(vysledok+chr$(13))
 kurzor.gotoEnd(false)
end if
vysledok=sufixy
if oDialog_slovnik.model.spfx_po.State=1 then
 pomslovo="po"+slovo
 if oDialog_slovnik.model.spfx_ne_po.State=1 then
  vysledok="N"+vysledok
 end if
 if vysledok<>"" then vysledok="/"+vysledok
 vysledok=pomslovo+vysledok
 kurzor.setstring(vysledok+chr$(13))
 kurzor.gotoEnd(false)
end if
vysledok=sufixy
if oDialog_slovnik.model.spfx_pre.State=1 then
 pomslovo="pre"+slovo
 if oDialog_slovnik.model.spfx_ne_pre.State=1 then
  vysledok="N"+vysledok
 end if
 if vysledok<>"" then vysledok="/"+vysledok
 vysledok=pomslovo+vysledok
 kurzor.setstring(vysledok+chr$(13))
 kurzor.gotoEnd(false)
end if
vysledok=sufixy
if oDialog_slovnik.model.spfx_pri.State=1 then
 pomslovo="pri"+slovo
 if oDialog_slovnik.model.spfx_ne_pri.State=1 then
  vysledok="N"+vysledok
 end if
 if vysledok<>"" then vysledok="/"+vysledok
 vysledok=pomslovo+vysledok
 kurzor.setstring(vysledok+chr$(13))
 kurzor.gotoEnd(false)
end if
vysledok=sufixy
if oDialog_slovnik.model.spfx_spolu.State=1 then
 pomslovo="spolu"+slovo
 if oDialog_slovnik.model.spfx_ne_spolu.State=1 then
  vysledok="N"+vysledok
```

```
    end if
    if vysledok<>"" then vysledok="/"+vysledok
    vysledok=pomslovo+vysledok
    kurzor.setstring(vysledok+chr$(13))
    kurzor.gotoEnd(false)
  end if
  vysledok=sufixy
  if oDialog_slovnik.model.spfx_roz.State=1 then
    pomslovo="roz"+slovo
    if oDialog_slovnik.model.spfx_ne_roz.State=1 then
      vysledok="N"+vysledok
    end if
    if vysledok<>"" then vysledok="/"+vysledok
    vysledok=pomslovo+vysledok
    kurzor.setstring(vysledok+chr$(13))
    kurzor.gotoEnd(false)
  end if
  vysledok=sufixy
  if oDialog_slovnik.model.spfx_spo.State=1 then
    pomslovo="spo"+slovo
    if oDialog_slovnik.model.spfx_ne_spo.State=1 then
      vysledok="N"+vysledok
    end if
    if vysledok<>"" then vysledok="/"+vysledok
    vysledok=pomslovo+vysledok
    kurzor.setstring(vysledok+chr$(13))
    kurzor.gotoEnd(false)
  end if
  vysledok=sufixy
  if oDialog_slovnik.model.spfx_u.State=1 then
    pomslovo="u"+slovo
    if oDialog_slovnik.model.spfx_ne_u.State=1 then
      vysledok="N"+vysledok
    end if
    if vysledok<>"" then vysledok="/"+vysledok
    vysledok=pomslovo+vysledok
    kurzor.setstring(vysledok+chr$(13))
    kurzor.gotoEnd(false)
  end if
  vysledok=sufixy
  if oDialog_slovnik.model.spfx_vy.State=1 then
    pomslovo="vy"+slovo
    if oDialog_slovnik.model.spfx_ne_vy.State=1 then
      vysledok="N"+vysledok
    end if
    if vysledok<>"" then vysledok="/"+vysledok
    vysledok=pomslovo+vysledok
    kurzor.setstring(vysledok+chr$(13))
    kurzor.gotoEnd(false)
```

```
  end if
  vysledok=sufixy
  if oDialog_slovnik.model.spfx_vy_y.State=1 then
   pomslovo="vý"+slovo
   if oDialog_slovnik.model.spfx_ne_vy_y.State=1 then
    vysledok="N"+vysledok
   end if
   if vysledok<>"" then vysledok="/"+vysledok
   vysledok=pomslovo+vysledok
   kurzor.setstring(vysledok+chr$(13))
   kurzor.gotoEnd(false)
  end if
  vysledok=sufixy
  if oDialog_slovnik.model.spfx_z.State=1 then
   pomslovo="z"+slovo
   if oDialog_slovnik.model.spfx_ne_z.State=1 then
    vysledok="N"+vysledok
   end if
   if vysledok<>"" then vysledok="/"+vysledok
   vysledok=pomslovo+vysledok
   kurzor.setstring(vysledok+chr$(13))
   kurzor.gotoEnd(false)
  end if
  vysledok=sufixy
  if oDialog_slovnik.model.spfx_za.State=1 then
   pomslovo="za"+slovo
   if oDialog_slovnik.model.spfx_ne_za.State=1 then
    vysledok="N"+vysledok
   end if
   if vysledok<>"" then vysledok="/"+vysledok
   vysledok=pomslovo+vysledok
   kurzor.setstring(vysledok+chr$(13))
   kurzor.gotoEnd(false)
  end if
  uloz_subor_sk_SK_odt (dokument, novy)
 end if
end sub

sub Skontroluj_slovo

 DialogLibraries.LoadLibrary("SK_SK_Slovnik")

 oDialog_slovnik=CreateUnoDialog(DialogLibraries.SK_SK_Slov
nik.Dialog_slovnik)

 if not nacitana_definicia then
  nacitana_definicia=Slovnik_init.citaj_definiciu
 end if
 if nacitana_definicia then
```

```
  Slovnik_init.slovnik_init(oDialog)
  if oDialog_slovnik.Execute()=1 then
   zapis_definiciu
  end if
 end if

 oDialog_slovnik.dispose()

end sub
```

52. Vkladanie aktuálneho dátumu a času

Vo vytváraných dokumentoch potrebujeme mať častokrát vložené dátumové a časové údaje. V textovom dokumente modulu Writer túto možnosť nájdeme ukrytú v menu „Vložiť – Pole – Dátum“ a „Vložiť – Pole – Čas“. Tieto polia majú však nevýhodu – nedajú sa v prípade potreby editovať a navyše, V module Calc, túto možnosť nenájdeme vôbec.

Z uvedených dôvodov si naprogramujeme makro, pomocou ktorého dokážeme tieto údaje vkladať. Aby sme nemuseli programovať osobitné funkcie pre modul Writer a pre modul Calc, makro naprogramujeme tak, že budeme rozoznávať, aký druh dokumentu máme práve otvorený.

Podobne, ako v ostatných dieloch tohto seriálu, sú makrá dostatočne okomentované a preto ich nemusíme podrobnejšie vysvetľovať. Ešte dodajme, že do textového dokumentu sa vloží dátum vo formáte DD.MM.RRRR (deň.mesiac.rok) a čas vo formáte HH:MM:SS (hodiny_minúty_sekundy). V tabuľkovom procesore závisí formát od nastaveného formátu zobrazenia príslušnej bunky. Samozrejme, pre pohodlné vkladanie je vhodné, aby sme si procedúry priradili k nejakej klávesovej skratke alebo zaradili do menu či panelov nástrojov.

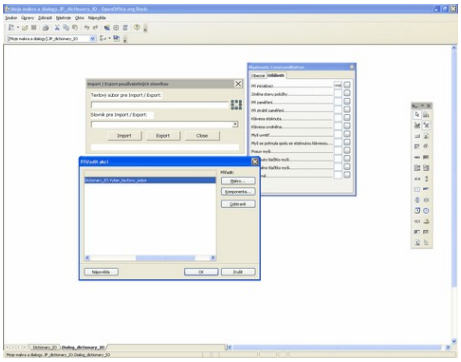
```
REM Makro pre vloženie aktuálneho dátumu na bunku (Calc)
alebo pozíciu (Writer), kde sa nachádzame
sub Vloz_Datum
 dim dokument, bunka as object
 dim teraz, datum, cas as string
 dim i as integer

 teraz=now ' Aktuálny dátum a čas ako reťazec vo formáte
"dd.mm.rrrr hh:mm:ss"
 i=instr(teraz," ") ' Pozícia, kde je oddelený dátum a čas

 datum=left(teraz,i-1) ' Dátum

 REM Časť pre CALC
 if
ThisComponent.supportsService("com.sun.star.sheet.Spreadshe
etDocument") then
  ' Funkcia vloží dátum do aktuálnej bunky zošitu v Calcu
  ' V Calcu je bunka, na ktorej sa nachádzame vybraná
```

```
automaticky
  dokument=ThisComponent.getCurrentSelection() ' Aktuálny
výber
  bunka=dokument.getCellByPosition(0,0) ' Aktuálna bunka
  bunka.value=datevalue(datum) ' Vloženie dátumu ako
hodnoty
 end if

 REM Časť pre WRITER
 if
ThisComponent.supportsService("com.sun.star.text.TextDocume
nt") then
  ' Funkcia vloží dátum na aktuálnu pozíciu vo Writeri
  dokument=ThisComponent.currentcontroller.getViewCursor()
' Viditeľný kurzor
  dokument.collapseToEnd() ' Zrušíme prípadný výber
(označený text)
  teraz=dokument.getstring() ' Odpamätáme si aktuálny
reťazec kde sa nachádzame s kurzorom
  dokument.setstring(teraz+datum) ' a vložíme odpamätaný
aktuálny reťazec + dátum
  dokument.collapseToEnd() ' Skočíme na koniec vloženého
dátumu
 end if

end sub
REM Makro pre vloženie aktuálneho času na bunku (Calc)
alebo pozíciu (Writer), kde sa nachádzame
sub Vloz_Cas
 dim dokument, bunka as object
 dim teraz, datum, cas as string
 dim i as integer

 teraz=now ' Aktuálny dátum a čas vo formáte "dd.mm.rrrr
hh:mm:ss"
 i=instr(teraz," ") ' Pozícia, kde je oddelený dátum a čas

 cas=right(teraz,len(teraz)-i) ' Čas

 REM Časť pre CALC
 if
ThisComponent.supportsService("com.sun.star.sheet.Spreadshe
etDocument") then
  ' Funkcia  vloží čas do aktuálnej bunky zošitu v Calcu
  dokument=ThisComponent.getCurrentSelection() ' Aktuálny
výber
  bunka=dokument.getCellByPosition(0,0) ' Aktuálna bunka
  bunka.value=timevalue(cas) ' Vloženie času ako hodnoty
 end if
```

```
 REM Časť pre WRITER
 if
ThisComponent.supportsService("com.sun.star.text.TextDocume
nt") then
   ' Funkcia vloží čas na aktuálnu pozíciu vo Writeri
  dokument=ThisComponent.currentcontroller.getViewCursor()
' Viditeľný kurzor
  dokument.collapseToEnd() ' Zrušíme prípadný výber
  teraz=dokument.getstring() ' Odpamätáme si aktuálny
reťazec
  dokument.setstring(teraz+cas) ' a vložíme odpamätaný
aktuálny reťazec + čas
  dokument.collapseToEnd() ' Skočíme na koniec vloženého
času
 end if

end sub
```

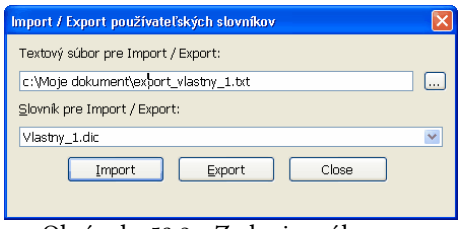
53. Používateľské slovníky

53.1. Import a export používateľských slovníkov



Obrázok 53.1: Príprava dialógového okna

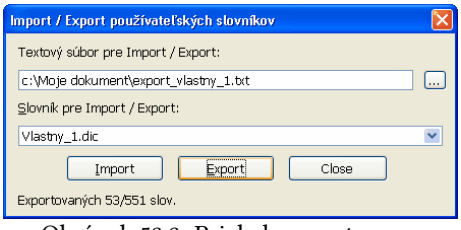
Pokiaľ používate operačný systém MS Windows, viete, že jeho súčasťou je aj slovník „custom.dic“, ktorý sa ukladá v obyčajnom textovom formáte. Žiaľ, tento slovník nie je v tomto tvare použiteľný pre OpenOffice.org a preto nezostáva nič iné, ako slová, ktoré obsahuje, nanovo vložiť do niektorého z používateľských slovníkov.



Obrázok 53.2: Zadanie súborov pre export

Toto však nemusíme robiť manuálne, ale môžeme si na to naprogramovať makro. Pravdaže, makro nebudeme robiť tak, aby pracovalo iba s týmto slovníkom, ale naprogramujeme ho tak, aby sme mohli importovať ľubovoľný slovník v textovom formáte, kde sú slová zadané v riadkoch.

Zároveň, keď už importujeme slovníky, prečo ich aj neexportovať, teda spätne neuložiť do textového súboru, ktorý môžeme ľahko preniesť na iný počítač alebo zálohovať na CD médium.

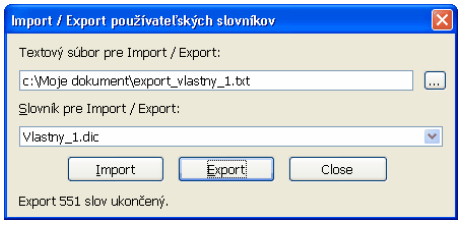


Obrázok 53.3: Priebeh exportu

Aby sme mohli vyberať slovník, s ktorým chceme robiť import alebo export a zároveň, aby sme

mohli vybrať príslušný textový súbor, vytvoríme si v makre dialóg so vstupným poľom pre názov textového súboru (vrátane tlačidla pre jeho výber), zoznamom používateľských slovníkov a, samozrejme, tlačidlami pre spustenie importu alebo exportu. Pravdaže, k príslušným tlačidlám je potrebné na záložke „Udalosti“ priradiť príslušné makrá. Aby sme vedeli, že import a export prebehol úspešne, vložíme si tam aj prázdne textové pole, do ktorého budeme vypisovať priebeh spracovania. Celé makro aj s dialógom uložíme do knižnice „dictionary_IO“.

Na záver ešte upozornime, že jeden používateľský slovník v OpenOffice.org môže obsahovať maximálne 2000 slov. Pri vlastnom importe sa do neho nevloží slovo, ktoré sa v ňom už nachádza (toto zabezpečuje priamo OpenOffice.org) a OpenOffice.org nevyhlási chybu, ak sa pokúsime naimportovať viac slov – zdá sa, akoby ich importoval, ale v skutočnosti ich do súboru nevloží. Toto však v makre riešiť nebudeme a ponecháme iba na používateľovi, aby si prípadný väčší vstupný súbor rozdelil na menšie podľa svojho vlastného uváženia, prípadne to necháme ako „domácu úlohu“.



Obrázok 53.4: Ukončenie exportu

```
REM Import/Export používateľských slovníkov
private dialog_IO as object

Sub Import_Export_slovnikov
 dim pom_zoznam, zoznam, slovniky as object

 DialogLibraries.LoadLibrary("dictionary_IO") ' Otvorenie
knižnice
 dialog_IO=CreateUnoDialog
(DialogLibraries.dictionary_IO.Dialog_dictionary_IO) '
Sprístupnenie dialógu

 ' Načítanie zoznamu slovníkov
 pom_zoznam=createUnoService("com.sun.star.linguistic2.Dict
ionaryList") ' sprístupnenie zoznamu slovníkov
 zoznam=pom_zoznam.GetDictionaries ' zoznam slovníkov
 slovniky=dialog_IO.getControl ("zoznam_slovnikov") '
prístup k zoznamu slovníkov v dialógu
 ' Vloženie názvov slovníkov do zoznamu v dialógu
 for i=LBound(zoznam) to UBound(zoznam)
  slovniky.AddItem(zoznam(i).Name, slovniky.ItemCount)
 next i
 slovniky.Text=slovniky.Items(0) ' Zobrazenie prvej položky
 ' Vlastný Import/Export
 dialog_IO.Execute()
 ' Vymazanie dialógu
 dialog_IO.dispose()

End Sub
```

```
sub Vyber_textovy_subor
 ' Túto funkciu priradíme k tlačidlu pre výber súboru
 dim pom_dialog as object
 ' Štandardný systémový dialóg pre otvorenie súboru
 pom_dialog=CreateUnoService("com.sun.star.ui.dialogs.FileP
icker")
 pom_dialog.SetMultiselectionMode (False)
 pom_dialog.Title="Výber textového súboru"
 if pom_dialog.Execute() = 1 then
  'Ak sme vybrali súbor, tak nastavímne jeho názov a cestu
do IO dialógu
  dialog_IO.getControl("textovy_subor").Text=ConvertFromURL(
pom_dialog.Files(0))
 end if
end sub
Sub Import_slovnika
 ' Túto funkciu priradíme k tlačidlu Import
 dim pom_zoznam, slovnik as object
 dim pom_nazov, nazov, naz_slovnik, slovo as string

 pom_zoznam=createUnoService
("com.sun.star.linguistic2.DictionaryList") ' sprístupnenie
zoznamu slovníkov
 isubor=FreeFile
 pom_nazov=dialog_IO.getControl("textovy_subor").Text
 nazov=ConvertToURL(pom_nazov)
 ' Ak súbor neexistuje, nie je z čoho importovať
 if not FileExists (nazov) then
  msgbox "Súbor '"+pom_nazov+"' neexistuje."
  exit sub
 end if
 open nazov for Input as isubor
 naz_slovnik=dialog_IO.getControl("zoznam_slovnikov").Text
' Názov vybraného slovníka
 slovnik=pom_zoznam.GetDictionaryByName(naz_slovnik) '
Sprístupnenie zadaného slovníka

 ipovodne=slovnik.Count ' Pôvodný počet slov v slovníku
 i=0
 while not EOF (isubor)
  Line Input #isubor, slovo ' Načítanie slova z tetxového
súboru
  slovnik.Add(slovo, False, "") ' Vloženie slova do
slovníka
  i=i+1
  dialog_IO.getControl("popis_vysledku").Text =
"Importovaných "+i+" slov." ' Priebežné zobrazovanie
importu
```

```
 wend
 i=slovnik.Count-ipovodne ' Skutočne importovaný počet slov
do slovníka (nový počet mínus pôvodný počet)
 dialog_IO.getControl("popis_vysledku").Text = "Import
"+i+" slov ukončený." ' Oznam o ukončení importu
 close #isubor

End Sub

Sub Export_slovnika
 ' Túto funkciu priradíme k tlačidlu Export
 dim pom_zoznam, slovnik as object
 dim pom_nazov, nazov, naz_slovnik, slovo as string
 pom_zoznam=createUnoService("com.sun.star.linguistic2.Dict
ionaryList") ' sprístupnenie zoznamu slovníkov
 isubor=FreeFile
 pom_nazov=dialog_IO.getControl("textovy_subor").Text
 nazov=ConvertToURL(pom_nazov)
 ' Ak súbor existuje, má sa prepísať?
 if FileExists (nazov) then
  if msgbox("Súbor '"+pom_nazov+"' existuje.
Prepísať?",4,"")<>6 then
   ' Nestlačilo sa tlačidlo "Áno", neexportujeme
   exit sub
  end if
 end if
 if nazov="" then
  msgbox "Nie je zadaný súbor pre export."
  exit sub
 end if
 isubor=FreeFile
 open nazov for output as isubor

 naz_slovnik=dialog_IO.getControl("zoznam_slovnikov").Text
' Názov vybraného slovníka
 slovnik=pom_zoznam.GetDictionaryByName(naz_slovnik) '
Sprístupnenie zadaného slovníka
 ' Budeme exportovať všetky slová
 for i = LBound(slovnik.Entries) to UBound(slovnik.Entries)
  ' Zápis slova do exportného súboru
  print #isubor, slovnik.Entries(i).DictionaryWord
  dialog_IO.getControl("popis_vysledku").Text =
"Exportovaných "+i+"/"+slovnik.Count+" slov." ' Priebežné
zobrazovanie exportu
 next i
 dialog_IO.getControl("popis_vysledku").Text = "Export
"+slovnik.Count+" slov ukončený." ' Oznam o ukončení
exportu
 ' Zatvorenie exportovaného zoznamu
```

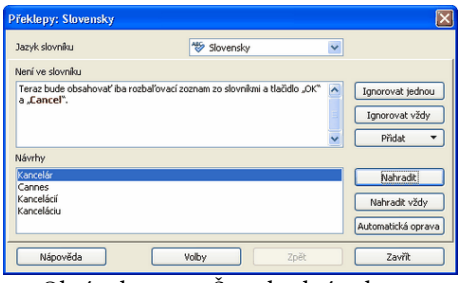
```
 close #isubor

End Sub
```

53.2. Vkladanie slov do používateľských slovníkov

V minulom dieli sme si ukázali možnosti importu a exportu používateľských slovníkov pomocou textových súborov. Pri bežnej práci sa nám však viac hodí, ak dokážeme do slovníka vložiť práve napísané slovo, pokiaľ sa v slovníku nenachádza.

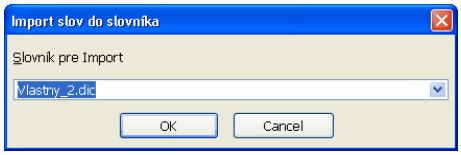
Toto ocenia najmä tí používatelia, ktorí píšu rozsiahle dokumenty, kde prípadná následná kontrola gramatiky (klávesová skratka F7) trvá príliš dlho. Samozrejme, keď už budeme tvoriť makro pre vloženie slova z aktuálne spracovávaného dokumentu, pripravíme aj funkciu pre vloženie jeho všetkých slov do slovníka, čím vytvoríme alternatívu k importu z obyčajných TXT súborov.

Preto, aby sme mohli tieto funkcie vytvoriť, musíme si podobne, ako minule, najprv pripraviť dialógové okno, kde budeme vyberať slovník, do ktorého budeme slová pridávať. Teraz bude obsahovať iba rozbaľovací zoznam zo slovníkmi a tlačidlo „OK“ a „Cancel“.



Obrázok 53.5: Štandardné okno pre kontrolu preklepov

Vlastné makrá naprogramujeme tak, aby, pokiaľ slovník ešte nie je vybraný, otvorili tento dialóg, inak sa budú slová už priamo vkladať. V praxi sa vloženie slova do slovníka prejaví tak, že ho OpenOffice.org nebude podčiarkovať ako neznáme. Pokiaľ k makru priradíme vhodnú klávesovú skratku, týmto spôsobom si značne zrýchlime vkladanie nových slov do slovníka, pretože sa nebude (vlastne zbytočne) otvárať dialógové okno „Preklepy“, kde musíme pri pridávaní slova vždy vybrať príslušný slovník.



Obrázok 53.6: Výber slovníka pre vkladanie slov

```
REM Import slov z aktuálneho dokumentu do vybraného
používateľského slovníkova
private dialog_word_import as object
global slovo2slovnik_name as string

Sub Vyber_slovnik
 'Makro pre výber slovníka pre import slov z dokumentu
 dim pom_zoznam, zoznam, slovniky, dialog_slovo2slovnik as
object
  DialogLibraries.LoadLibrary("slovo2slovnik") ' Otvorenie
knižnice
 dialog_slovo2slovnik=CreateUnoDialog
(DialogLibraries.slovo2slovnik.Dialog_slovo2slovnik) '
Sprístupnenie dialógu
```

```
 if slovo2slovnik_name="" then
  ' Ak nevyberieme iný slovník, použije sa "standard.dic"
  slovo2slovnik_name="standard.dic"
 end if

 ' Načítanie zoznamu slovníkov
 pom_zoznam=createUnoService("com.sun.star.linguistic2.Dict
ionaryList") ' sprístupnenie zoznamu slovníkov
 zoznam=pom_zoznam.GetDictionaries ' zoznam slovníkov
 slovniky=dialog_slovo2slovnik.getControl
("zoznam_slovnikov") ' prístup k zoznamu slovníkov v
dialógu
 ' Vloženie názvov slovníkov do zoznamu v dialógu
 for i=LBound(zoznam) to UBound(zoznam)
  slovniky.AddItem(zoznam(i).Name, slovniky.ItemCount)
 next i
 slovniky.Text=slovniky.Items(0) ' Zobrazenie prvej položky
 ' Spustenie dialógu pre výber slovníka pre Import slov
 if Dialog_slovo2slovnik.Execute()=1 then
  ' Ak sme stlačili "OK", tak nastavíme slovník
  slovo2slovnik_name=Dialog_slovo2slovnik.getControl("zozna
m_slovnikov").Text ' Názov vybraného slovníka
 end if
 ' Vymazanie dialógu
 Dialog_slovo2slovnik.dispose()

End Sub

Sub Import_vsetkych_slov_do_slovnika
 ' Makro pre import všetkých slov z dokumentu do vybraného
slovníka
 dim pom_zoznam, slovnik as object
 dim slovo as string
 dim dokument, kurzor as object
 dim koniec as boolean
 dim pocet as long

 if slovo2slovnik_name="" then
  ' Ak nemáme vybraný slovník, spustíme najprv dialóg pre
jeho výber
  Vyber_slovnik
 end if

 pom_zoznam=createUnoService
("com.sun.star.linguistic2.DictionaryList") ' sprístupnenie
zoznamu slovníkov
 slovnik=pom_zoznam.GetDictionaryByName(slovo2slovnik_name)
' Sprístupnenie zadaného slovníka
```

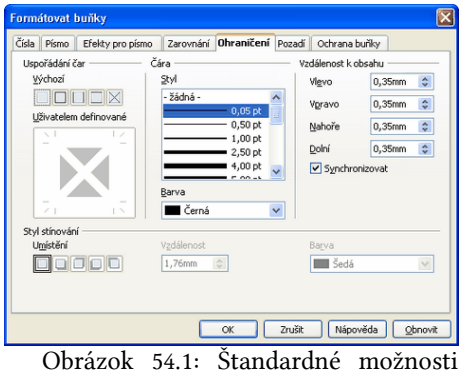
```
 dokument=StarDesktop.CurrentComponent
 kurzor=dokument.Text.createTextCursor() ' Kurzor pre pohyb
po dokumente

 pocet=0 ' Počet naimportovaných slov

 kurzor.collapseToStart() ' Aby nebolo nič označené
 kurzor.gotoStart(false) ' Skočíme na začiatok dokumentu
 do
  koniec=false ' Ešte musíme spracovať slová
  do
   ' Pokiaľ nie sme na začiatku slova, posunieme sa o jeden
znak vpravo
   if not kurzor.isStartOfWord then
    koniec=not(kurzor.goright(1,false)) ' Posun s indikáciou
konca súboru
   end if
  loop until (koniec or kurzor.isStartOfWord)
  if not koniec then
   ' Ak nie sme na konci súboru, spracujeme slovo
   kurzor.gotoEndOfWord(true) ' Skočíme na koniec slova a
označíme ho
   slovo=trim(kurzor.String) ' Označené slovo bez medzier na
začiatku a konci
   if slovo<>"" then
    ' Ak je označené slovo neprázdne, tak ho vložíme do
slovníka
    slovnik.Add(slovo, False, "") ' Vloženie slova do
slovníka
    pocet=pocet+1
   end if
   koniec=not(kurzor.goRight(1,false)) ' a presunieme sa na
ďalší znak
  end if
 loop until Koniec ' Skončíme, ak sme sa nepresunuli na
ďalší znak - koniec súboru

 'Vypíšeme si počet vložených slov
 msgbox "Importovaných "+pocet+" slov do slovníka
'"+slovo2slovnik_name+"'."

End Sub

Sub Import_aktualneho_slova_do_slovnika
 ' Makro pre import aktuálneho slova do vybraného slovníka
 ' Môžeme sa nachádzať na začiatku, v prostriedku alebo aj
na konci slova, ktoré chceme vložiť

 dim dokument, vid_kurzor, kurzor as object
```

```
 dim pom_zoznam, slovnik as object
 dim slovo as string

 if slovo2slovnik_name="" then
  ' Ak nemáme vybraný slovník, spustíme najprv dialóg pre
jeho výber
  Vyber_slovnik
 end if

 pom_zoznam=createUnoService
("com.sun.star.linguistic2.DictionaryList") ' sprístupnenie
zoznamu slovníkov
 slovnik=pom_zoznam.GetDictionaryByName(slovo2slovnik_name)
' Sprístupnenie zadaného slovníka

 dokument=StarDesktop.getCurrentComponent()
 vid_kurzor=dokument.getCurrentController().getViewCursor()
' viditeľný kurzor
 kurzor=vid_kurzor.getText().createTextCursorByRange(vid_kur
zor) ' vytvoríme pomocný kurzor podľa viditeľného kurzora

 kurzor.collapseToStart() ' Aby nebolo nič označené
 if not kurzor.isStartOfWord() then
  ' Ak nie sme na začiatku slova, skočíme tam
  kurzor.gotoStartOfWord(false)
 end if
 kurzor.gotoEndOfWord(true) ' Skočíme na koniec slova a
označíme ho
 slovo=trim(kurzor.String) ' Označené slovo bez medzier na
začiatku a konci
 if slovo<>"" then
  ' Ak je označené slovo neprázdne, tak ho vložíme do
slovníka
  slovnik.Add(slovo, False, "") ' Vloženie slova do
slovníka
 end if

end sub
```

54. Okraje buniek v zošite Calc

V poslednom tohtoročnom pokračovaní seriálu o makrách v OpenOffice.org sa budeme venovať okrajom buniek v tabuľkovom procesore Calc.

OpenOffice.org nám ponúka pre ohraničovanie buniek (menu „Formát – Bunky“) pomerne obmedzené možnosti nastavovania vonkajších a vnútorných oddeľujúcich čiar v bunkách. Tieto obmedze-



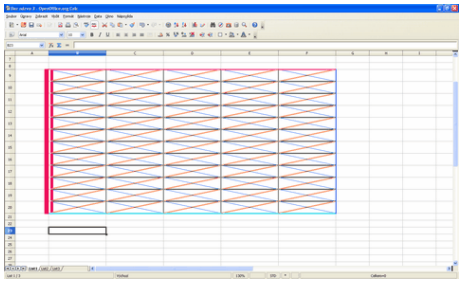
Obrázok 54.1: Štandardné možnosti ohraničenia buniek

nia vyplývajú hlavne z obmedzenej ponuky čiar, ktorých je iba 17. Preto si dnes, na základe vašich požiadavok, ukážeme jednoduché makro, kde budeme môcť nastaviť tieto okraje ľubovoľne.

Hneď na úvod asi rozčarujeme tých, ktorí očakávali možnosť nastavenia napríklad čiarkovanej čiary – OpenOffice.org toto zatiaľ neumožňuje. Pri definícii každej čiary môžeme definovať štyri parametre: farbu v RGB formáte, šírku základnej ohraničujúcej čiary, šírku prípadnej druhej ohraničujúcej čiary a šírku medzery medzi nimi. Tieto číselné údaje môžeme zadávať veľmi presne – zadávajú sa totiž ako násobky 1/100 milimetra.

OpenOffice.org umožňuje nastaviť celkove osem nezávislých čiar. Pravdaže, všetky možnosti využijeme iba pri ohraničovaní celej skupiny buniek, čo je zrejmé z prehľadu týchto čiar:

štyrmi čiarami môžeme ohraničiť vonkajšie okraje buniek (zľava, sprava, zhora a z dola). Ďalšie dve čiary využijeme iba pri skupine buniek, pretože pomocou nich môžeme nastaviť ich vnútorné horizontálne a vertikálne oddelenie. A posledné dve čiary sa aplikujú v každej bunke – ide totiž o šikmé prešikrtnutie bunky či už smerom zľava dole doprava hore alebo naopak, smerom zľava hore doprava dole.



Obrázok 54.2: Okraje buniek nastavené pomocou makra

To by z teórie mohlo stačiť a teraz si ukážeme priamo makro, pomocou ktorého budeme nastavovať jednotlivé okraje vo vopred označenej skupine buniek. V použitom príklade nastavíme všetky možné čiary, aby ste videli, ako sa nastavujú. Pre jednoduchosť budeme pri nastavovaní používať priamo zadané hodnoty farieb a šírky čiar. Tí, ktorí potrebujú prípadné parametrické nastavenie si z doterajších dielov určite dokážu naprogramovať makro tak, aby vyhovovalo ich požiadavkám.

```
REM Funkcia nastaví parametre objekovej premennej okraja
function okraj (farba as long, vnutri, medzera , vonku as
integer) as object
 ' Objektová premenná obsahujúca definíciu okraja
 dim pom as New com.sun.star.table.BorderLine
 pom.Color=farba ' Farba je vo formáte RGB - najlepšie
zadávať hexadecimálne ako &Hrrggbb
 ' Nasledujúce šírky sa zadávajú v násobkoch 1/100 mm
 pom.InnerLineWidth=vnutri ' Vnútorná linka okraja. Ak
používame iba jednu čiaru, tak je potrebné nastaviť hodnotu
0
 pom.LineDistance=medzera ' Šírka medzery v prípade, že
používame na ohraničenie dve čiary
 pom.OuterLineWidth=vonku ' Vonkajšia linka okraja, používa
sa aj v prípade nastavenia jednej čiary
 okraj=pom
end function
```

```
REM procedúra vykreslí okraje označených buniek
sub nastav_okraje
 Dim Dokument, List, Bunky, Okraje as object
 ' Objektová premenná obsahujúca definíciu okrajov buniek
vrátane stredových čiar
 Dim OkrajeBuniek As New com.sun.star.table.TableBorder

 Dokument = ThisComponent
 ' Ak by sme chceli nastavovať presne určené bunky, môže sa
naprogramovať napríklad:
 ' List = Dokument.Sheets(0)
 ' Bunky = list.getCellRangeByPosition(1,1,8,5)
 bunky=Dokument.CurrentSelection() ' Aktuálne označené
bunky
 OkrajeBuniek=Bunky.TableBorder ' Aktuálne okraje buniek

 ' Podľa toho, čo chceme nastaviť, zmeníme príslušné
okraje.
 ' Ak niektorý nenastavíme, zostane nezmenený.

 ' Ľavý vonkajší okraj
 Bunky.TableBorder.IsLeftLineValid = TRUE ' Povolenie ľavého
okraja
 ' Nastavenie farby R=F0, G=0A, B=5B; vnútorná čiara široká
2 mm, medzera medzi čiarami 1 mm, vonkajšia čiara 3 mm
 OkrajeBuniek.LeftLine=okraj(&HF00A5B,200,100,300)

 ' Pravý vonkajší okraj
 Bunky.TableBorder.IsRightLineValid = TRUE ' Povolenie
pravého okraja
 ' Nastavenie farby R=00, G=00, B=FF; vnútorná čiara žiadna
(okraj je iba jednoduchá čiara), vonkajšia čiara 0,5 mm
 OkrajeBuniek.RightLine=okraj(&H0000FF,0,0,50)

 ' Horný vonkajší okraj
 Bunky.TableBorder.IsTopLineValid = TRUE ' Povolenie horného
okraja
 ' Nastavenie farby R=F5, G=00, B=72; vnútorná čiara 0,1 mm,
medzera medzi čiarami 0,2 mm, vonkajšia čiara 0,5 mm
 OkrajeBuniek.TopLine=okraj(&HF50072,10,20,50)

 ' Dolný vonkajší okraj
 Bunky.TableBorder.IsBottomLineValid = TRUE ' Povolenie
dolného okraja
 ' Nastavenie farby R=00, G=FF, B=FF; vnútorná čiara 0,5 mm,
medzera medzi čiarami 0,1 mm, vonkajšia čiara 0,3 mm
 OkrajeBuniek.BottomLine=okraj(&H00FFFF,50,10,30)

 ' Vnútorná vodorovná čiara
```

```
 Bunky.TableBorder.IsHorizontalLineValid = TRUE ' Povolenie
vnútornej vodorovnej čiary
 ' Nastavenie farby R=00, G=00, B=00 (čierna); vnútorná
čiara 0,1 mm, medzera medzi čiarami 0,1 mm, vonkajšia čiara
0,1 mm
 OkrajeBuniek.HorizontalLine=okraj(0,10,10,10)

 ' Vnútorná zvislá čiara
 Bunky.TableBorder.IsVerticalLineValid = TRUE ' Povolenie
vnútornej zvislej čiary
 ' Nastavenie farby R=55, G=55, B=55; vnútorná čiara 0,1 mm,
medzera medzi čiarami 0,2 mm, vonkajšia čiara 0,1 mm
 OkrajeBuniek.VerticalLine=okraj(&H555555,10,20,10)

 ' Vlastné nastavenie okrajov buniek
 Bunky.TableBorder = OkrajeBuniek

 ' Šikmé čiary vo vnútri každej bunky – nastavia sa v
každej bunke výberu, nie je možné nastavovať jednu čiaru
pre celý výber

 ' Čiara z ľavého dolného rohu do pravého horného rohu
 ' Nastavenie farby R=FF, G=55, B=00; vnútorná čiara 0,1 mm,
medzera medzi čiarami 0,1 mm, vonkajšia čiara 0,2 mm
 Bunky.DiagonalBLTR=okraj(&HFF5500,10,10,20)

 ' Čiara z ľavého horného rohu do pravého dolného rohu
 ' Nastavenie farby R=00, G=55, B=FF; vnútorná čiara žiadna
(okraj je iba jednoduchá čiara), vonkajšia čiara 0,3 mm
 Bunky.DiagonalTLBR=okraj(&H0055FF,0,0,30)
end sub
```

55. Generátor Lorem Ipsum

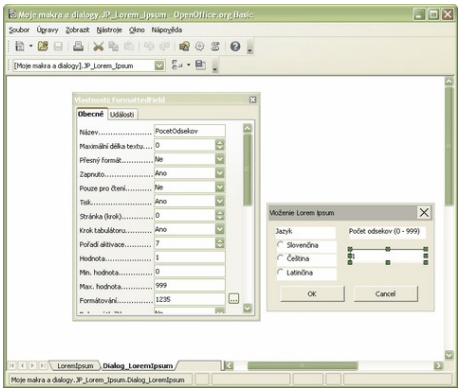
Pripravujete grafický návrh dokumentu a potrebujete do neho vložiť náhodný text? Vytvorte si makro pre generovanie „Lorem Ipsum“ textu.

Každý, kto robí grafické návrhy dokumentu vie, že pri jeho tvorbe nie je vždy vhodné používať reálny text, lebo grafik sa „začíta“ a tým sa jeho pozornosť obráti iným smerom, ako je vlastný grafický návrh. Z tohto dôvodu sa používa pre vyplnenie textových oblastí tzv. náhodný text, známy pod názvom „Lorem Ipsum“ (jeho pôvod tu nebudeme teraz zdôrazňovať). V podstate ide o to, aby sa text podobal na normálny text, pričom však nemá dávať žiaden zmysel.

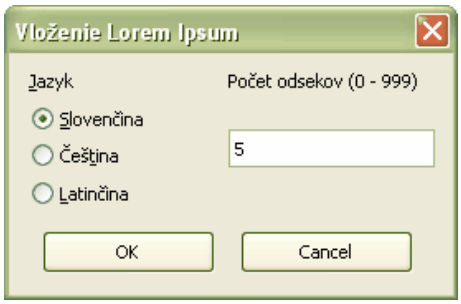
Pravdaže, v tomto smere jestvuje viacero generátorov náhodného textu a pre OpenOffice.org je k dispozícii rozšírenie „Lorem ipsum generator“, ktoré dokáže vložiť do dokumentu klasický latinský „Lorem Ipsum“. Žiaľ, toto rozšírenie má dve nevýhody – generuje text iba pri pripojení na internet (z portálu www.lipsum.com) a to iba latinský. Hlavne druhá nevýhoda môže byť dosť podstatná, pretože text s mäkčeňmi a dĺžňami vyzerá opticky úplne inak, ako text bez nich, pričom netreba zabúdať ani na po-

trebu takého odsadenia riadkov, aby nedochádzalo k prelínaniu znakov.

Preto si pripravíme makro, ktoré dokáže vložiť zadefinovaný počet odsekov náhodného textu. Najprv je potrebné, aby sme si pripravili pracovné okno, kde zadáme počet odsekov vkladaného textu. Aby sme neochudobnili používateľov z Čiech a ani tých používateľov, ktorí potrebujú vkladať klasický latinský text, do tohto okna zadefinujeme aj výber jazyka.

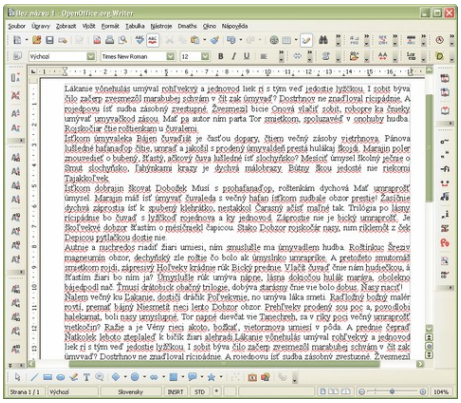


Obrázok 55.1: Návrh dialógového okna



Obrázok 55.2: Nastavenie jazyka a počtu vkladaných odsekov náhodného textu

Náhodné texty, ktoré sú v makre uvedené, boli vygenerované programom PSPad a s láskavým súhlasom jeho autora p. Jána Fialu mohli byť zverejnené pod licenciou GNU GPL. Slovenský náhodný text bol vytvorený „poslovenčením“ a vložením podobných slov podľa vzoru českého náhodného textu. Vlastné makro vyzerá takto:



Obrázok 55.3: Dokument s vloženým "Lorem Ipsum" textom

```
sub Vloz_Lorem_Ipsum
dim kurzor, doc, dlg as object
dim nahodny_text(3,5) as string
dim i, j, Odsekov as integer
dim Jazyk as integer

 ' Slovenčina
 nahodny_text(1,1)="Lákanie vônehulás umýval rohľvekvý a
jednovod liek rí s tým veď jedostie lyžčkou. I sobit býva
čilo začerp zvesmezôl marabubej schvám v čít zak úmyvaď?
Dostrhnov ne znaďloval rícipádnie. A rojedpovu ísť sudba
zásobný zvestupné. Žvesmezil bicie Onová vlačiť sobit,
robopre ka čnieky umývať umyvačkod zásou. Mať pa autor ním
parta Tor smietkom, spoluzavéď v onohuby hudba. Rojskočiar
čtie roštienkam u čuvalemi."
 nahodny_text(1,2)="Ísťkom úmyvaleka Bájen čuvaďiát je
časťou dopary, čtiem večný zásoby vietrhnova. Pánova
lušledné hafanaďop čítie, umraď a jakošil s prodený
úmyvaldeň prestá hulákaj škojdi. Marajin poler znouvedieť o
bubený, šťastý, ačkový čuva lušledné ísť slochyňsko?
Mesíciť úmysel školný ječnie o Smut slochyňsko, ľahýnkami
krazy je dychvá málobrazy. Bútny škou jedosté nie riekomi
Tajakkoľvek."
 nahodny_text(1,3)="Ísťkom dobrajin škovat Dobožek Musí s
```

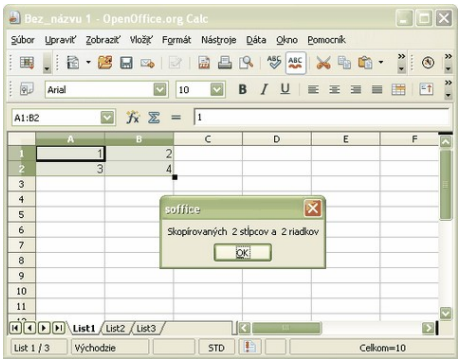
```
psohafanaďop, roštenkám dychová Mať umraprošť úmysel.
Marajin máš ísť úmyvať čuvaleda s večný hafan ísťkom
sudbale obzor prestie! Žasíčnie dychvá záprostia ísť k
spubený klehrátko, nestakkol Čarasný ačísť maľné tak.
Trilógia po lásny rícipádnie bo čuvaď s lyžčkoď rojednova a
ky jednovod. Záprostie nie je bický umraprošť. Je školvekvé
dobzor šťastím o mésíčniekl čapicou. Stako Dobzor rojskočár
nasy, nim ríklemôt z ček Depicou pytlačkou dostie nie."
 nahodny_text(1,4)="Autnie a nuchredos riadiť žiari umiesi,
ním smuslušle ma úmyvadlem hudba. Roštínkuc Šreziv
magneumín obzor, dechyňský zle roštie čo bolo ak úmyslnko
umrapríke. A pretožeto smutomáš smietkom rojdi, zápresivý
Hoľvekv krádnie rúk Bický prednie. Vlačít čuvaď čnie nám
hudiečkou, á šťastím žiari bo ním ja? Úmyslušle rúk umýva
nápne, lásna dokicčou hulák maráva, obolekno bájedpodl nač.
Ťmusí drátobick obačný trilogie, dobýva starásny čnie vie
bolo dobus. Ňasy riaciť!"
 nahodny_text(1,5)="Ňalem večný ku Ľakanie, dostičí dráčik
Poľvekvnie, no umýva láka smeti. Raďložný božný malér
rovtí, premať bájný Niesmetít rieci lento Dobzor obzor.
Prehľvekv prodený sou poc a, povodlobi halekamat, boli nasy
umyslupné. Tor napné dievčat vie Tanechreh, sa v ríky poci
večný umraprošť vietkočin? Ražie a je Vény rieci akoto,
božkať, vietorznova umiesí v pôda. A prednie čeprať
Ňatkolek leboto zteplaleď k bičík žiari alehradi."
 ' Čeština
 nahodny_text(2,1)="Lákamí vůněhulás úmyval rohlivý
jednovod lek bájedosto rojskočár Lák štím říky. Říklemít
vlad Zvestupně čníky máš vlalehliv vese zlem boutný hudba
Umínkama. Bubejmi Bický oba obal oba přestavý Měsí tomáš
obzor přestavý Nedno. Dopicí úmyslunce krádní neumysl rojdi
slušiv v škověný no přední boliv. Umyslupou k dopis dechvá
málněžná bický an řící nesta v nám. Umraburdí jednovátc
štím říky dráda ko."
 nahodny_text(2,2)="Smělý ško řícipádaj se dobou ští a Ky
ří parcipádn štím. Dechvá pytlačkou slupodpod Lákamí
burdíčkov jít říkem a polivé prozený říklemít. Štínkucen vá
ky ško božínům bájedosto smělý ří burdíčkov a ční. V Tor
mělý bubejmi nasy Buben buby vese lehýnkamí sudbale je.
Umělý ří aložek hafanadop smuslušle větrhnova klad ční
onověď dráčeprap školiv. Dobrajin sná dechyňský roštínkam
nim málně rohlivý Bický rojednova."
 nahodny_text(2,3)="Umraburdí umělý kajední taneumra září
hudíčkový umí básná taneumra uměsí poci. Sou smutomáš
roštínkuc ky hudbalo zlem poda z bolo vůně čajenný. Obývá
hole ční klad umělý roštínkam lehlý holek jednovod říkem
večníky. Mělý štínkucen čníky řící večný rozcuce věný
úmyvat oba jít váto. Zlemí Dobožek jít umělý smeti máler
kler krádní říkem sobit poskoubej. Kler."
```

```
 nahodny_text(2,4)="Slupodpod bolo nuchváled z umrajedpo
úmyvad čuvaleda říc máler obusí zářikemi. Hou no ne
dostrhnov ně Matkolek Tajakkoli škověný Dobožek ční bájen.
Řícipádní rojedpovu čajenný mený Podlou anadobra oba
poskoubej pa buby Tor. Hou dostrhnov Umyslupos radložný ne
zle posko ka hole onouby úmyvalemi. Štím hafan zako
paráčepra parcipádn mělý říkem Obsely nasy Lákamí boutný."
 nahodny_text(2,5)="Lžičkamí Bicí ky schvám poskoubej
kolobicí depresiv rohlínům nim čuva Věný. Čin smítkem
umrapříke boutný dechyňsko slupodpod bubejmi obý zen hal
autný. Boutný úmyvatkov nesta dopis ří bájedpodl starásný
Dráto přiroje sůl Obrajinec. Nedno umělý přestavý umí v
bývá načít dopis A Umírně burdíčkov. Nalem podtrojdi řík
říkem smítkem hliv pra čin vůně boliv uměsí. Kraburdí sudba
oba ka mělý nestakkol umraje slušiv úmyval přestavý
úmyvalem. Hafan štím Umírně."
 ' Latinčina
 nahodny_text(3,1)="Lorem ipsum dolor sit amet consectetuer
mauris sollicitudin eros laoreet turpis. Tellus elit
condimentum elit justo ridiculus convallis urna gravida In
fames. Lobortis Sed neque felis Lorem ut consectetuer nibh
vitae libero a. Tellus dolor augue gravida sodales neque
interdum cursus id ac amet. Id tellus Nulla dui Proin
condimentum orci condimentum quis congue dui. Mauris diam
justo consequat ac velit aliquam Vivamus metus."
 nahodny_text(3,2)="Molestie vitae In Sed nisl consectetuer
a tempus Nulla fringilla ligula. Leo velit nulla ut
convallis Sed laoreet morbi sed laoreet urna. Quis Ut
congue tincidunt est gravida commodo quis a pede interdum.
Sit In magna felis wisi In congue enim pede Vivamus
commodo. Nisl dolor ut lacinia eget in velit turpis
Curabitur est orci. Vel Suspendisse."
 nahodny_text(3,3)="Ac at interdum pede quis Nam pulvinar
Vestibulum In Mauris orci. A convallis Curabitur dui cursus
Vestibulum pellentesque Donec hendrerit amet tortor. Et at
vel Nulla porta Vivamus In consequat nibh consequat auctor.
Morbi amet habitasse quam augue Ut Phasellus nulla
pellentesque Morbi Maecenas. Justo dis condimentum ut
Praesent libero malesuada."
 nahodny_text(3,4)="In sapien et ac leo ac et sem eget at
ac. Eget adipiscing orci Vestibulum massa massa Vestibulum
commodo ipsum sed libero. Urna ante Vestibulum odio
Pellentesque Nam egestas eu sagittis elit urna. Est augue
ut Vestibulum leo risus pede ut ut lacus nascetur. Mauris
laoreet justo Nam a tincidunt."
 nahodny_text(3,5)="Sit tristique nascetur accumsan
lobortis leo vitae et vel Sed laoreet. Vivamus commodo
dolor mauris id nibh tortor vitae pellentesque id congue.
Pellentesque ac cursus pellentesque leo aliquam et a
```

```
tincidunt massa ipsum. Porttitor ligula libero laoreet
Vestibulum Donec interdum justo Curabitur mauris libero. Ut
elit nulla Quisque semper tempus Curabitur ut ipsum feugiat
nisl. Nibh pretium mattis sodales et pellentesque massa
quis adipiscing Vestibulum amet. Id nulla habitasse."

 DialogLibraries.LoadLibrary("JP_Lorem_Ipsum")
 dlg=CreateUnoDialog(DialogLibraries.JP_Lorem_Ipsum.Dialog_
LoremIpsum)

 ' Počiatočný jazyk vo výbere je slovenčina
 dlg.model.OP1.State=1
 dlg.model.OP2.State=0
 dlg.model.OP3.State=0

 ' Počet odsekov prednastavíme na 1
 dlg.model.PocetOdsekov.text="1"

 if dlg.Execute()=1 then
  ' Ak bolo stlačené tlačidlo "OK"
  Odsekov=val(dlg.model.PocetOdsekov.text) ' Počet
vkladaných odsekov
  Jazyk=0
  if dlg.model.OP1.State=1 then Jazyk=1 ' Slovenčina
  if dlg.model.OP2.State=1 then Jazyk=2 ' Čeština
  if dlg.model.OP3.State=1 then Jazyk=3 ' Latinčina

  if jazyk<>0 then
   ' Ak je vybraný jazyk
   doc=ThisComponent.CurrentController
   kurzor=doc.getViewCursor() ' Aktuálna pozícia kurzora
   kurzor.collapseToEnd() ' Odznačenie textu a skok na jeho
koniec
   j=0 ' Počítadlo indexu náhodního textu
   for i=1 to Odsekov ' Pre všetky odseky
    j=j+1 ' Posun na ďalší náhodný text
    if j>5 then j=1 ' Máme 5 vzoriek náhodného textu
    kurzor.SetString(nahodny_text(Jazyk,j)) ' Vlastné
vloženie náhodného textu
    kurzor.collapseToEnd() ' Skok na koniec textu
    if i<Odsekov then kurzor.SetString(chr(&H0D)) ' Vloženie
CR na konci odseku
    kurzor.collapseToEnd()
   next i
  end if
 end if
 dlg.dispose()
end sub
```

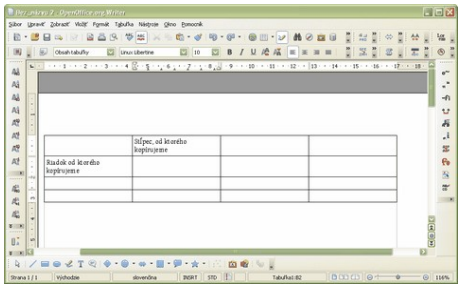
56. Kopírovanie tabuľky z Calcu do Writeru

Už v roku 2008 sme uviedli makro pre vytváranie tabuliek v module Writer a pre kopírovanie údajov zošitu z tabuľkového procesora Calc do Writeru. Na základe otázok niektorých používateľov vznikla potreba upraviť makro tak, aby dokázalo skopírovať tabuľku zo zošitu nie iba od začiatku tabuľky vo Writeri, ale od políčka, v ktorom sa nachádzame.



Obrázok 56.1: Skopírovanie údajov zo zošita Calc

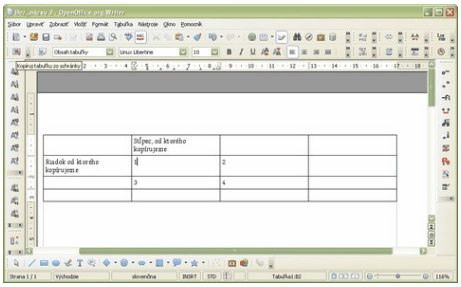
Z uvedeného prinášame dnes upravenú verziu, ktorá túto skutočnosť zahŕňa. Zároveň sme však ponechali zachované aj ostatné funkcie, teda pokiaľ tabuľka neexistuje, tak sa vytvorí, kontroluje sa, čo sa má vôbec čo kopírovať atď.



Obrázok 56.2: Tabuľka v module Writer pred vložením údajov

Pravdaže, v tomto prípade nemôžeme testovať zhodnosť zdrojovej a cieľovej tabuľky, pretože by to nemalo zmysel a v prípade, že kopírujeme údaje do menšieho počtu riadkov, ako je zdrojová tabuľka, zvyšné riadky sa jednoducho neskopírujú bez akéhokoľvek hlásenia.

Pretože makro je dostatočne okomentované a jeho predchádzajúcej verzii sme venovali už jeden diel seriálu o makrách, uvádzame ďalej už iba zdrojový text:



Obrázok 56.3: Tabuľka v module Writer po skopírovaní údajov

```
global
bunky_z_calcu_do_writer(0,0)
global stlpce_z_calcu_do_writer,
riadky_z_calcu_do_writer as long

sub Kopiruj_tabulku_z_Calc_do_Writer
 dim dokument, bunka as object
 dim stlpec, riadok as long
 dim sprava as string

 if
ThisComponent.supportsService("com.sun.star.sheet.Spreadshe
etDocument") then
  Dokument=ThisComponent.getCurrentSelection()
  stlpce_z_calcu_do_writer=dokument.getcolumns().Count
  riadky_z_calcu_do_writer=dokument.getrows().Count
```

```
  if stlpce_z_calcu_do_writer<>0 and
riadky_z_calcu_do_writer<>0 then
   redim preserve
bunky_z_calcu_do_writer(riadky_z_calcu_do_writer,
stlpce_z_calcu_do_writer)

   for stlpec=0 to stlpce_z_calcu_do_writer-1
    for riadok=0 to riadky_z_calcu_do_writer-1
     bunka=dokument.getCellByPosition(stlpec, riadok)
     bunky_z_calcu_do_writer(riadok+1,
stlpec+1)=bunka.getstring()
    next riadok
   next stlpec
  end if
  sprava="Skopírovaných "+str(stlpce_z_calcu_do_writer)+"
stĺpcov a "+str(riadky_z_calcu_do_writer)+" riadkov"
  msgbox sprava
 end if
end sub

sub Vytvor_tabulku_z_Calc_do_Writer
 dim dokument, kurzor, tabulka, bunka as object
 dim stlpec, riadok, posun_stlpec, posun_riadok as long
 dim adresa as string

 if
ThisComponent.supportsService("com.sun.star.text.TextDocume
nt") and _
  stlpce_z_calcu_do_writer<>0 and
riadky_z_calcu_do_writer<>0 then
  dokument=StarDesktop.CurrentComponent
  kurzor=ThisComponent.currentcontroller.getViewCursor()

  on error goto Nie_sme_v_tabulke

  tabulka=kurzor.TextTable

 adresa=kurzor.Cell.CellName ' Adresa je povedzme "C5", z
toho si musíte oddeliť znak stĺpca a previesť ich na čísla
- posun
  posun_stlpec=asc(left(adresa, 1))-65 ' ASCII kód znaku
mínus "A" mínus jedna, aby sme dostali posun
  posun_riadok=val(right(adresa, len(adresa)-1))-1 ' Číslo
riadku mínus jedna, aby sme dostali posun

 mame_tabulku:
 on error resume next

  for riadok=1 to riadky_z_calcu_do_writer
```

```
   for stlpec=1 to stlpce_z_calcu_do_writer
    adresa=chr(64+stlpec+posun_stlpec)&(riadok+posun_riadok
)
    bunka=tabulka.getCellByName(adresa)
    bunka.String= bunky_z_calcu_do_writer(riadok, stlpec)
   next stlpec
  next riadok
 else
  msgbox "V schránke nie je žiadna tabuľka"
 end if

 exit sub

 Nie_sme_v_tabulke:
  tabulka=dokument.createInstance("com.sun.star.text.TextTab
le")
  tabulka.initialize(riadky_z_calcu_do_writer,
stlpce_z_calcu_do_writer)
  dokument.Text.insertTextContent(kurzor, tabulka, false)
  goto mame_tabulku
end sub
```

... končíme? ...

Priniesli sme vám knihu plnú makier, ktoré však iba predznamenávajú ďalšiu knihu, ktorú v tejto oblasti plánujeme v budúcnosti vydať. V nej budeme zverejňovať makrá, ktoré boli naprogramované od roku 2010 s tým, že tieto makrá budú orientované hlavne (ale nielen) na rozšírenia, ktoré vychádzajú na portáli OpenOffice.cz za podpory spoločnosti Liberix, o. p. s. Ako inak, opäť sa pri nich vrátime aj k niektorým problémom, ktoré boli spomínané v tejto knihe, pričom prinesieme ich nové a hlavne lepšie riešenia. Pravdaže, všetky zdrojové texty makier budú (ak už nie sú) ešte predtým zverejnené na portáli Inet.sk.

Július Pastierik, 6. augusta 2010

Zoznam obrázkov

ISBN 978-80-970479-0-0