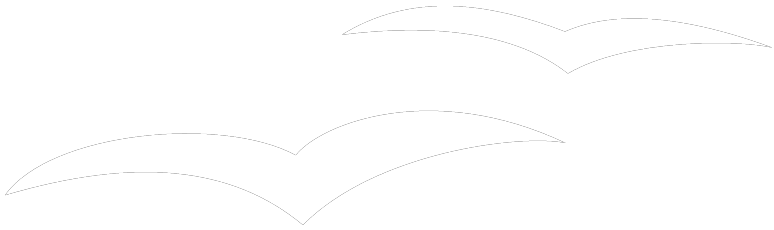
Makrá v OpenOffice.org
… začíname …



Vytvorené pre internetový denník www.inet.sk

© 2005 Július Pastierik

Obsah

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.

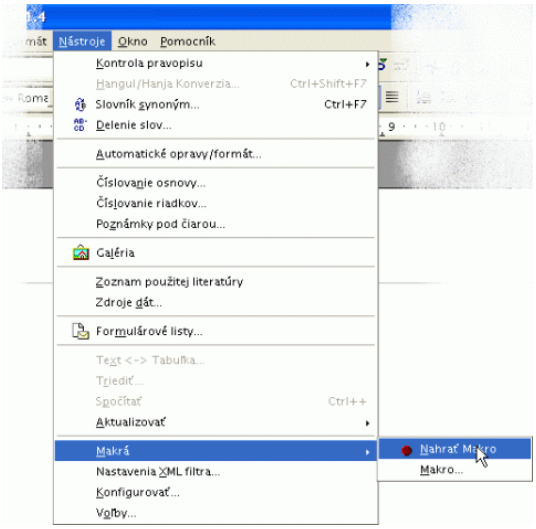
„Nudné“ ale potrebné informácie na úvod

Priznám sa, že nemám rád zbytočné definície, a preto si dovolím predstaviť makro v kancelárskom balíku OpenOffice.org je 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.

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.

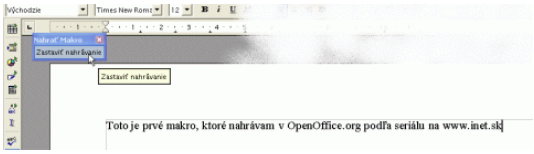


Obrázok 1: Spustíme nahrávanie makra

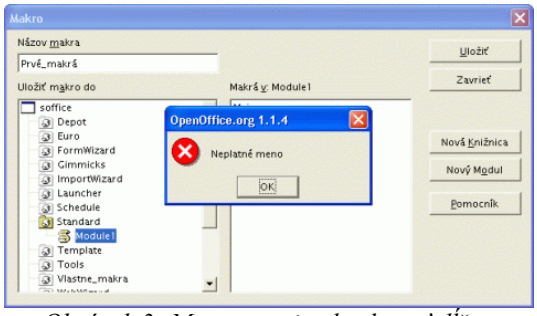
V hlavnom menu kancelárskeho balíka OpenOffice.org si vyberieme položku „Nástroje-Makrá-Nahrať makro“, ktorú si prípadne môžeme nechať zobrazovať aj na paneli funkcií. Po jej výbere sa nám zobrazí malé plávajúce okno „Nahrať makro“, kde nájdeme jediné tlačidlo „Zastaviť nahrávanie“, ktoré je, samozrejme, 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.



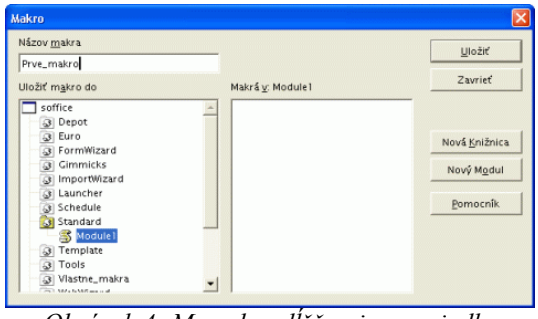
Obrázok 2: Zastavíme nahrávanie



Obrázok 3: Meno nesmie obsahovať dĺžne

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.

Z vlastnej praxe vám môžem doporučiť, že ak chcete používať prehľadné dlhé názvy s oddeľujúcou medzerou, namiesto nej je najvhodnejšie použiť 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“.

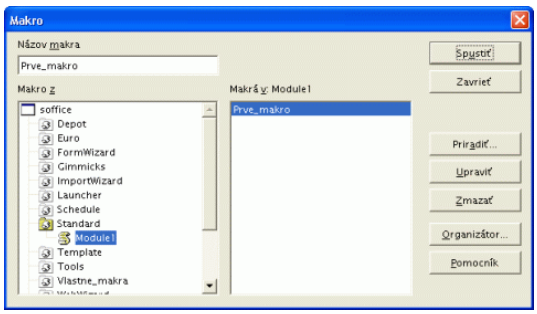


Obrázok 4: Meno bez dĺžňov je v poriadku

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 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.

Konfigurujeme OpenOffice.org

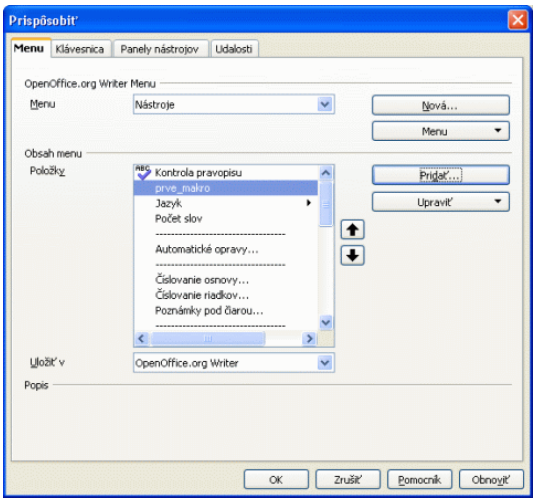
Ukázali sme si ako sa dá vytvoriť jednoduché makro bez toho, aby sme vôbec vedeli programovať. Zároveň sme si ukázali jednu z možností, ako ho spustiť. Samozrejme, táto možnosť nie je jediná a, priznajme si, ani tá najpohodlnejšia.

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 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.

Záložka „Menu“

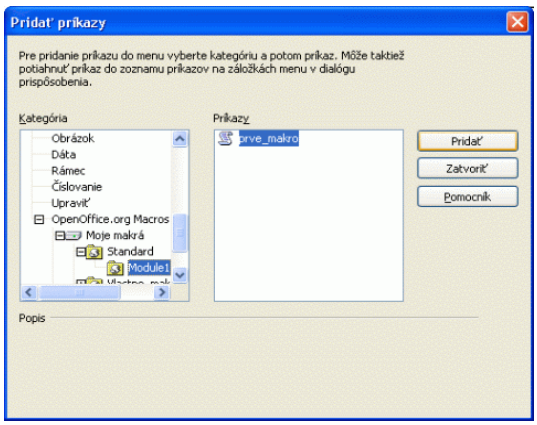
Na tejto záložke môžeme zmeniť menu OpenOffice.org. Vzhľadom na to, že tu vlastne môžeme zmeniť celé menu OpenOffice.org, musíme byť veľmi pozorní, aby sme si, náhodou, nezrušili prístup k dôležitým funkciám. Venujme sa však vlastnému zaradeniu nášho makra do menu. Záložka sa skladá z dvoch zoznamov.



Obrázok 6: Záložka "Menu"

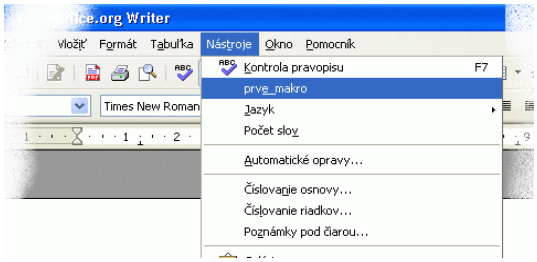
V prvom si vyberieme menu, do ktorého chceme pridať naše makro (napr. „Nástroje“). V druhom zozname sa nám následne zobrazí vlastný obsah menu. Prejdeme na miesto za ktoré chceme vložiť naše makro (napr. na položku „Kontrola pravopisu“) a stlačíme tlačidlo „Pridať. Zobrazí sa nám okno „Pridať príkazy“, 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“.

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 7: Vyhľadáme naše makro

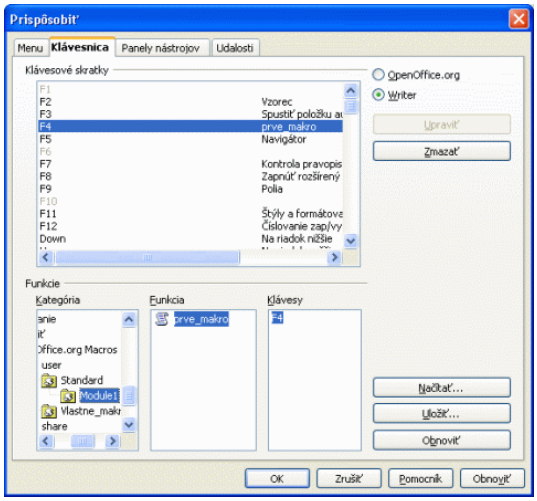
Ak sme postupovali správne, makro máme teraz priamo prístupné v menu „Nástroje-Prve_makro“. Pokiaľ chceme naše makro z menu premenovať alebo natrvalo odstrániť, prejdeme znovu do tejto záložky dialógového okna „Prispôsobiť“, prejdeme na našu položku v menu a stlačíme tlačidlo „Upraviť“, kde vyberieme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



Obrázok 8: Naše makro ako súčasť menu

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.

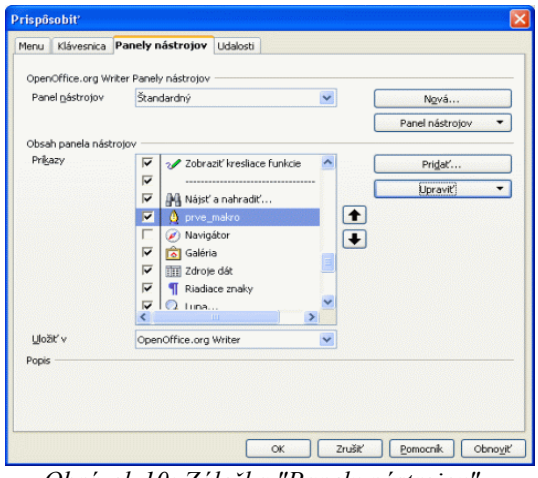


Obrázok 9: Záložka "Klávesnica"

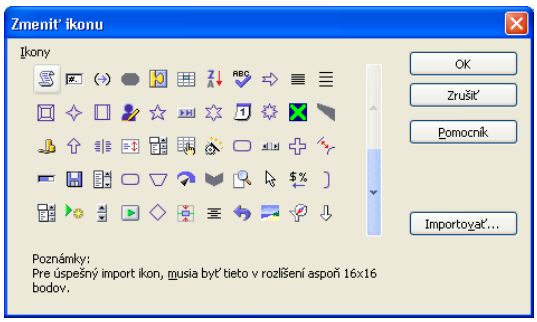
Zoznam makier máme v tomto prípade priamo prístupný v spodnej časti tejto záložky. Naše makro nájdeme, samozrejme, podobne ako v predchádzajúcom príklade v okienkach „Kategória“ a „Funkcia“. Potom stlačíme tlačidlo „Upraviť“, čím makro priradíme ku vybranej klávesovej skratke. Naopak, ak chceme makro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

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.



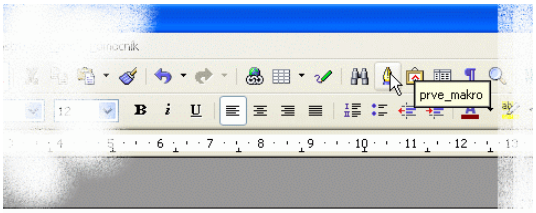
Obrázok 10: Záložka "Panely nástrojov"



Obrázok 11: Vyberáme si ikonu

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“.

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“.

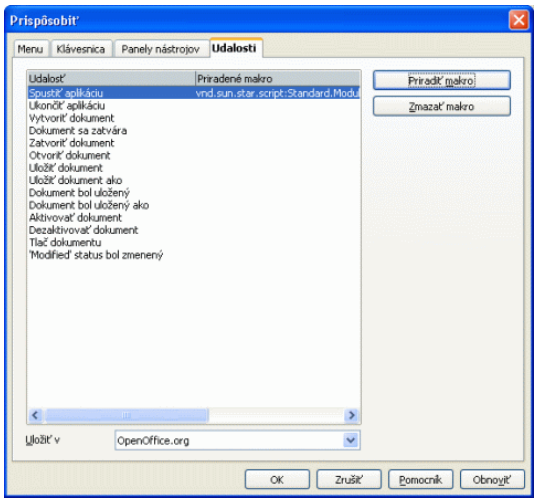


Obrázok 12: Naše makro v paneli nástrojov

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.

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“.



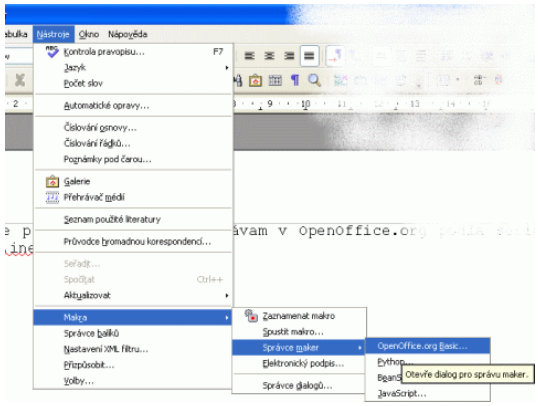
Obrázok 13: Záložka "Udalosti"

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.

Čo sme to naprogramovali?

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

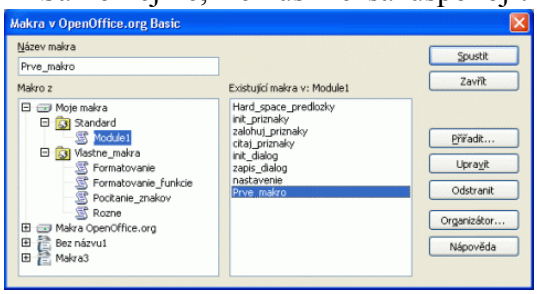
Hoci sme už uviedli, že nahrávanie makier nie je priame programovanie, vytvorili sme takýmto postupom procedúru v jazyku StarOffice Basic. Je to tak preto, že potrebné inštrukcie za nás vložil priamo OpenOffice.org. Pozrime sa teda na to, čo napísal za nás. Otvoríme si dialógové okno „Makrá“. Na rozdiel 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“.



Obrázok 14: Otvoríme si okno "Makrá"

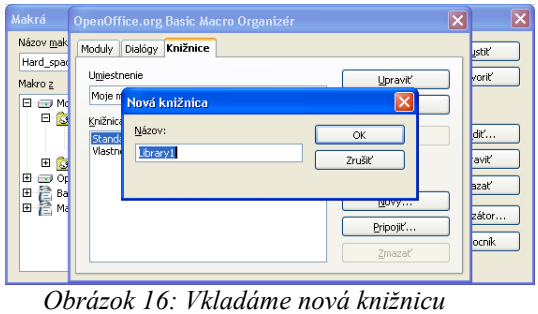
A znovu niečo konfigurujeme

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“.

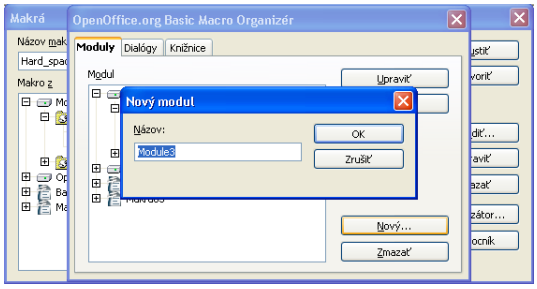


Obrázok 15: Určite budeme mať viac makier

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 16: Vkladáme nová knižnicu



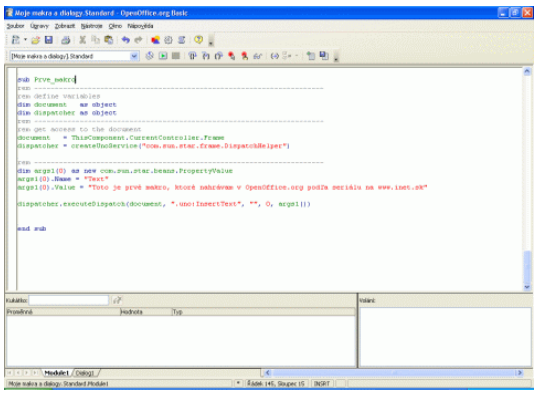
Obrázok 17: Vkladáme nový modul

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.

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"


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


  end sub
```



Obrázok 18: 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 (SUB*routine*) 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 (REM*ark*). 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 (DIM*ension*) 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())
```

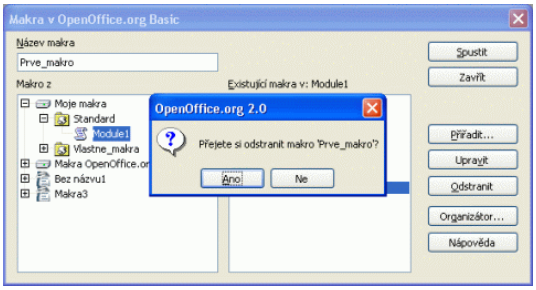
Hoci na to tento riadok na prvý pohľad vôbec nevyzerá, je to vlastne volanie iného podprogramu, či lepšie povedané v tomto prípade štandardnej UNO metódy. Prístup k UNO rozhraniu sme získali jeho priradením do premennej „dispatcher“ a preto príslušnú metódu „executeDispatch“ máme prístupnú priamo cez 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 SUB*routine*).

Rozobrali sme podrobne príkazy nášho prvého makra. Na koniec práce nezabudneme 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.



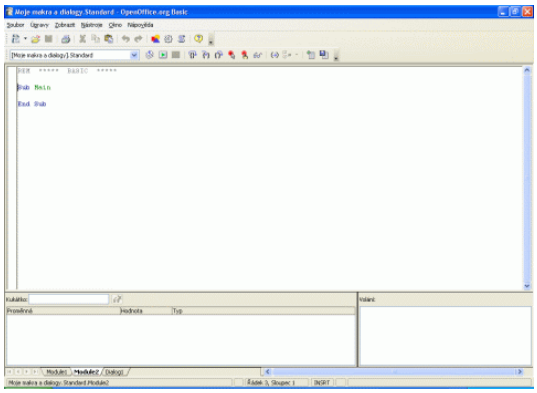
Obrázok 19: Mažeme makro

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.

Programujeme v StarOffice Basic



Obrázok 20: Makro "Main"

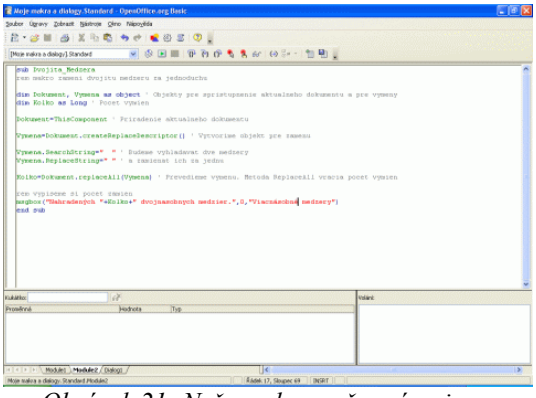
Nové makrá môžeme do OpenOffice.org vkladať priamo tak, že si už známym postupom, o ktorom sme písali minule (menu „Nástroje-Makrá-Usporiadať makrá…-OpenOffice.org Basic…“) otvoríme dialógové okno pre správu makier. Tu prejdeme na modul, kde chceme vložiť naše makro a stlačíme tlačidlo „Upraviť“.

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:

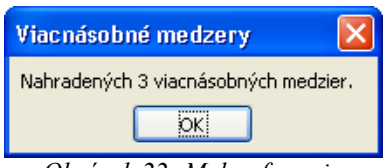
```
sub Dvojita_Medzera
rem makro zameni dvojitu medzeru za jednoduchu

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 21: 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.

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



Obrázok 22: Makro funguje

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 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 objekt 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 23: 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.

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.

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.

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 …

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 nejaké 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 definície, pretože sme sa ešte nezoznámili s typmi premenných.

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
  ENDIF
```

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 premenná
    CASE hodnota 1:
      ' … príkazy, ak sa premenná rovná tejto hodnote
    CASE hodnota 2:
      ' … príkazy, ak sa premenná rovná tejto hodnote
    CASE rozsah hodnôt 1
      ' … príkazy, ak sa premenná rovná niektorej z uvedených
hodnôt
    CASE rozsah hodnôt 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
```

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, …

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.

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
```

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
```

Typy premenných

V treťom dieli tohto seriálu (Makrá v OpenOffice.org III. – 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.

Reťazcové premenné

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

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). Príklady reťazcov:

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

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 10^{38} (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 10^{308} (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.

Logické premenné

```
DIM premenna AS BOOLEAN
```

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

Dátumové premenné

```
DIM premenna AS DATE
```

Dátumové premenné obsahujú hodnotu dátumu a času.

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 <základný 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())
```

Objektové premenné

```
DIM premenna AS OBJECT
```

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.

Operátory

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

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.

Logické operátory

AND Logický súčin.

OR Logický súčet.

XOR Exkluzívny logický súčet.

NOT Negácia.

EQV Ekvivalencia.

IMP Implikácia.

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.

(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 odstavca 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 odstavca
    Vymena.SearchString="^ *"
    ' ktoré vymažeme
    Vymena.ReplaceString=""
    kolko=kolko+Dokument.replaceAll(Vymena)
    Vymena.SearchRegularExpression=True

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

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 odstavca. 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 odstavca. Napríklad pomocou regulárneho výrazu `"^ "` dokážeme nájsť všetky jednoduché medzery na začiatku odstavca.

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

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

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

`*` – 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 odstavca.

`+` – 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 odstavca.

`?` – 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 opačným lomítkom sa nepovažuje za špeciálny (zástupný) znak regulárneho výrazu, ale za obyčajný. Takto môžeme v regulárnych výrazoch vyhľadať napríklad znak bodky `"\."`, doláru `"\$"`, opačného lomítka `"\\"` 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 odstavca, 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 hexadecimálnym 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 desaťdielnu 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.).