Programujeme v softwaru Statistica

díl druhý

Newsletter Statistica ACADEMY

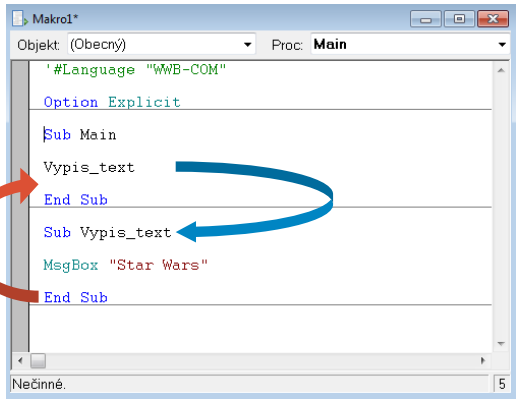
Téma: Programování, makra, skripty
Typ článku: Návody

*V tomto článku si ukážeme další možnosti při psaní maker v softwaru Statistica. Navazuje na úvodní **článek z minulého čísla** newsletteru, kde byly vysvětleny základní prvky programovacího prostředí, jejichž pochopení je nezbytné pro vytvoření základních maker. Dnes si popíšeme další prvky a pustíme se do tvorby komplexnějších maker, která budeme moci nasadit v reálných situacích.*

Již víme, jak a jaké proměnné můžeme používat, jak větvit kód programu do ucelených celků. V tomto článku si ukážeme, jak lze mezi jednotlivými celky navzájem komunikovat, a popíšeme si základní prvky pro práci s cykly.

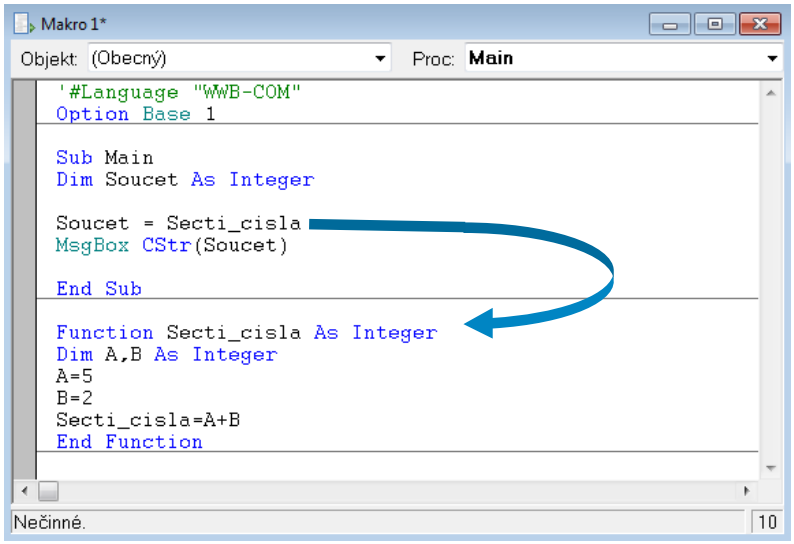
Volání procedur, předávání hodnot proměnných

Každý složitější program/makro se skládá z menších, dílčích celků, které jsou navzájem provázány a komunikují mezi sebou. Při spuštění makra Statistica se jako první zpracovává procedura s názvem *Main*. Ta tvoří základní část makra, což ale neznamená, že musí zákonitě obsahovat nejvíc řádků. Často jsou nosné části maker soustředěny do samostatných procedur nebo funkcí. Pokud chceme, aby byly také zpracovány, musíme je takzvaně „zavolat". Volat můžeme nejen z hlavní procedury *Main*, ale samozřejmě i jiných procedur a funkcí. Volání provedeme jednoduše tak, že v kódu uvedeme název dané procedury/funkce:



Výše uvedené makro se po spuštění postupně zpracovává, a jakmile se dostane na řádek s odkazem na funkci/proceduru (*Vypis_text*), začne se zpracovávat kód procedury. Po jeho vykonání se kurzor vrátí do místa, odkud byla procedura volána a pokračuje dál ve zpracování zbylého kódu.

Pokud voláme funkci, která vrací hodnotu, obvykle ji voláme z místa, kde s danou hodnotou pracujeme, např. přiřazujeme nějaké proměnné:



Výše uvedený kód vypíše na obrazovku výsledek „7". Všimněme si, že název funkce vystupuje v rámci funkce jako proměnná a pokud má funkce vrátit požadovanou hodnotu, musíme jí v průběhu zpracování funkci přiřadit, v našem případě *Secti_cisla=A+B*.

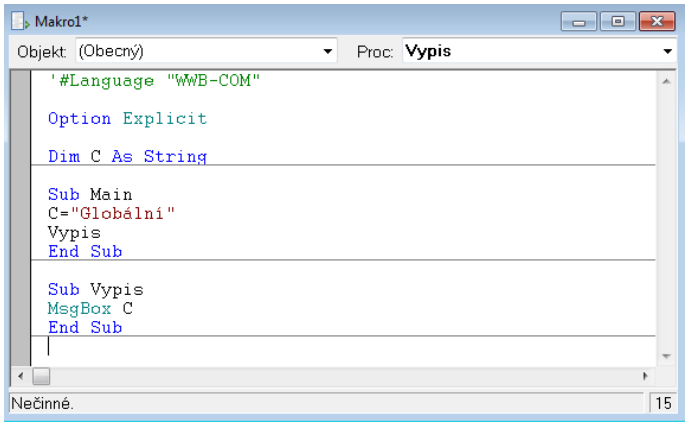
Tady je třeba si něco říci o dostupnosti jednotlivých proměnných v různých částech makra. Proměnné, deklarované v rámci jakékoli procedury/funkce, lze použít pouze V RÁMCI dané procedury/funkce. Pokud bychom chtěli ve výše uvedeném příkladu přistoupit k proměnné „A" v rámci procedury *Main*, potom by program zahlásil chybu, že proměnná není deklarována. Jako by pro tuto část programu vůbec neexistovala. Z toho plyne, že lze používat v jednotlivých procedurách/funkcích stejné názvy proměnných, aniž by nastal konflikt. Nicméně, z důvodu přehlednosti tento přístup nedoporučuji.

Jak tedy předáme hodnoty proměnných mezi procedurami/funkcemi? Jsou dvě možnosti:

- Globální proměnné
- Argumenty procedury/funkce

Globální proměnné

Globální proměnnou deklarujeme mimo jakoukoli proceduru/funkci, obvykle před hlavní procedurou *Main*. Takto deklarovaná proměnná je dostupná v rámci celého kódu makra, včetně procedur a funkcí:



Výše uvedené makro je bez problému zpracováno a vypíše text „Globální"

Doporučuji však používání globálních proměnných omezit na minimum, jelikož jde proti principům objektového programování, kdy jsou všechny hodnoty zapouzdřeny v objektech a jinak než v rámci objektů k nim přistupovat nelze. I procedury a funkce jsou v podstatě objekty. Tím lze dosáhnout integrity dat a zabránit nechtěným změnám hodnot globálních proměnných v rozsáhlejších makrech/programech.

Argumenty procedury/funkce

Argumenty procedur/funkcí můžeme chápat jak vstupní parametry (hodnoty), které proceduře předáváme a ty se obvykle účastní operací v rámci těchto procedur/funkcí. Argumenty definujeme při vytváření procedur/funkcí a zapisujeme je v závorce za názvem procedury/funkce:

Sub Volitelny_jednoslovny_nazev (Argument1 as datovy_typ, Argument1 as datovy_typ, atd.)

End sub

Takže např.

Sub Procedura1 (Text as string)

End sub

Argumenty potom vystupují v rámci procedury/funkce jako proměnné. Můžeme je číst i modifikovat. Při volání procedur/funkcí musíme na místě argumentu použít očekávanou hodnotu. Hodnotu můžeme zadat přímo:

Procedura1 ("Vítej!")

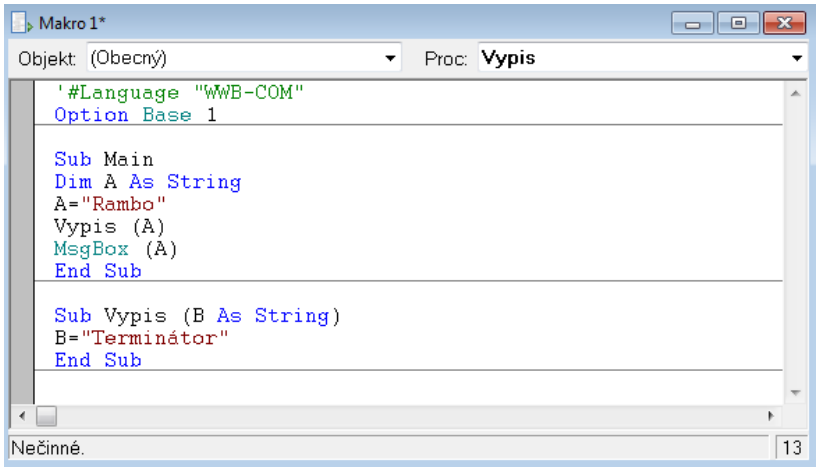
Nebo prostřednictví proměnné:

Dim Text as string

Text="Vítej!"

Procedura1 (Text)

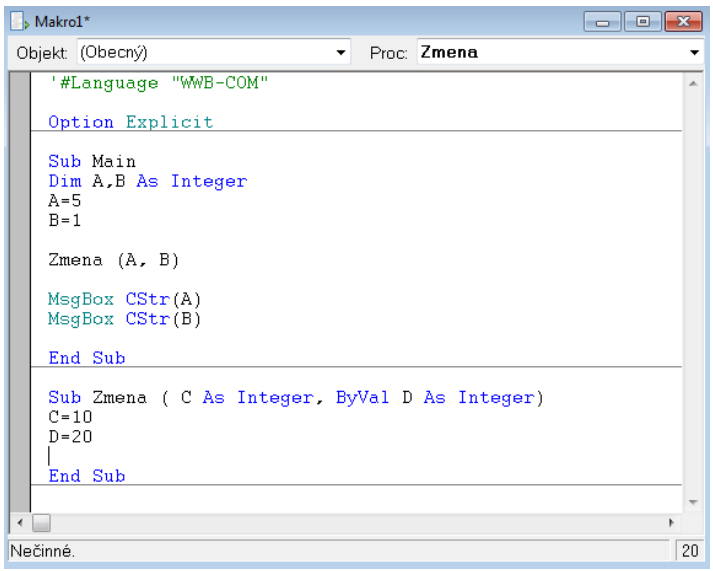
Pokud hodnotu proměnné v rámci volané procedury/funkce změníme, potom je tato upravená hodnota předána, po vykonání procedury, zpět do místa, odkud byla zavolána:



Výše uvedený kód vypíše „Terminátor". Můžeme si všimnout, že názvy proměnné a argumentu se nemusí shodovat, ale můžou mít i shodný název.

ByRef, ByVal, Optional

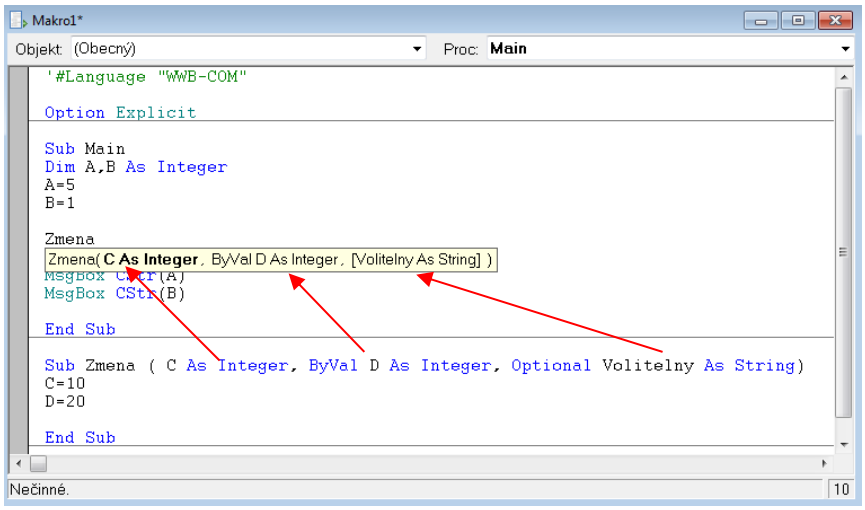
V některých případech je žádoucí, aby hodnota proměnné, kterou použijeme jako argument, zůstala nezměněna. Hodnotu totiž můžeme předávat buď jako hodnotu jako takovou (v paměti počítače se pro tuto hodnotu vyhradí nové místo), nebo pomocí odkazu, na již existující hodnotu proměnné v operační paměti počítače (odkazem). Pokud předáváme hodnotu odkazem, potom, dojde-li ke změně hodnoty v rámci procedury/funkce, potom je změněna i původní hodnota proměnné, která vstupuje to procedury jako argument. Pokud není definováno jinak, jako výchozí se ve Statistice předávají hodnoty odkazem. Pokud si chceme být jisti, že je předáván pouze odkaz na proměnnou, použijeme před definici argumentu klíčové slovo *ByRef*. Pokud chceme přenášet čistě hodnotu proměnné, potom použijeme *ByVal*:



Vlevo uvedené makro vypíše hodnotu proměnné *A* jako „10", *B* jako „1".

Při volání procedury/funkce musíme použít všechny argumenty, v rámci procedury/funkce definované. Tzn. je potřeba zavolat proceduru/funkci se všemi potřebnými hodnotami. To však neplatí vždy. Výjimku tvoří argumenty volitelné, definované pomocí klíčového slova *Optional*.

Vývojové prostředí Statistica Visual basic nám při psaní kódu pomáhá a napovídá (našeptává). Všechny argumenty, které volaná procedura/funkce vyžaduje, můžeme vidět při zápisu kódu v okamžiku, kdy za název volané procedury/funkce napíšeme mezeru:



Můžeme si všimnout, že nepovinné argumenty jsou v našeptávači uvedeny v hranatých závorkách a není je třeba při volání uvádět.

Opakování částí programu, cykly

Nyní bychom již měli být schopni napsat strukturované makro, kdy obvykle z hlavní nosné části voláme části ostatní, často i opakovaně. Pro vykonání opakovaných operací máme ve Statistice několik konstrukčních možností:

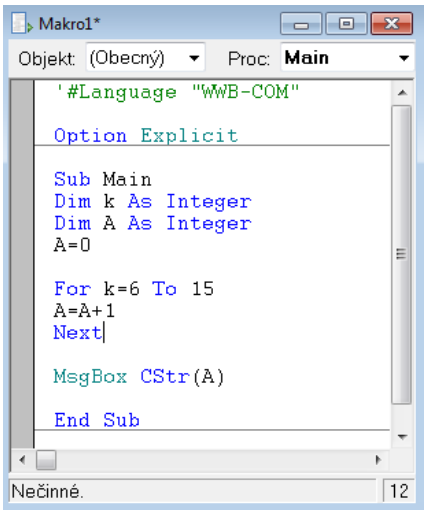
- For .. Next
- Do .. Loop
- While .. Wend
- For each .. Next

For .. Next

Často používaný konstruktor pro práci s cykly, který nalezneme i v jiných programovacích jazycích. Používá se v situacích, kdy potřebuje provést určitý počet cyklů, např. 10. Zápis je uvozen klíčovým slovem „*For*", následován čítačem, kterým říkáme, kolikrát chceme cyklus provést. Čítač musí být číselná proměnná, u které definujeme počáteční hodnotu, konečnou hodnotu a popřípadě přírůstek. Pokud přírůstek nedefinujeme, je použita hodnota 1. Definice čítače pro 10 opakování by vypadala následovně:

I=1 to 10

Hodnota proměnné „*I*" by tedy postupně nabývala hodnot od 1 do 10, s přírůstkem 1. Konstrukce cyklu je ukončena klíčovým slovem *Next*. Bude se tedy provádět vše, co je uvnitř bloku *For – Next*:



Výsledkem bude vypsání hodnoty 10.

Do .. Loop

Jedná se o cyklus s podmínkou. Nejedná se tedy o cyklus s předem daným počtem cyklů a je prováděn do okamžiku, kdy je splněna/nesplněna námi definovaná podmínka. Podmínka může být definována na začátku nebo na konci cyklu. Pozor, tento cyklu lze zapsat i bez podmínky, potom ale cyklus nikdy neskončí a běh makra je třeba ručně zastavit. Konstrukce cyklu s podmínkou na začátku:

Do podmínka

Loop

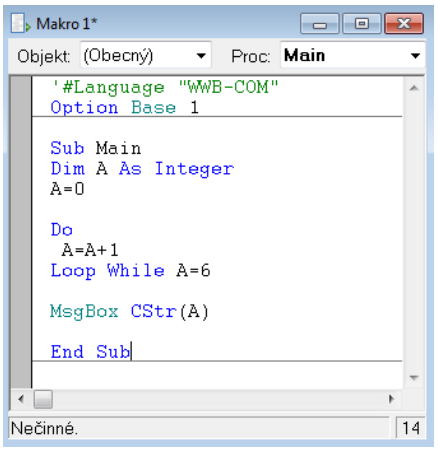
S podmínkou na konci:

Do

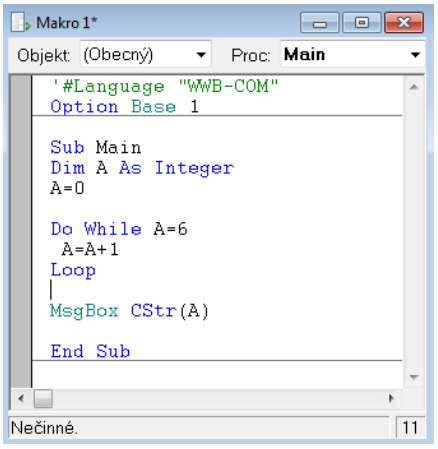
Loop podmínka

Podmínka může být dvojího druhu – *Until* a *While*. *While* použijeme v případech, kdy chceme zachovat běh cyklu tak dlouho, dokud je podmínka splněna, tzn. tak dlouho, dokud je podmínka *True*. V případě *Until* běží cyklus tak dlouho, než je podmínka splněna, tzn. do okamžiku, kdy podmínka nabyde hodnoty *True*.

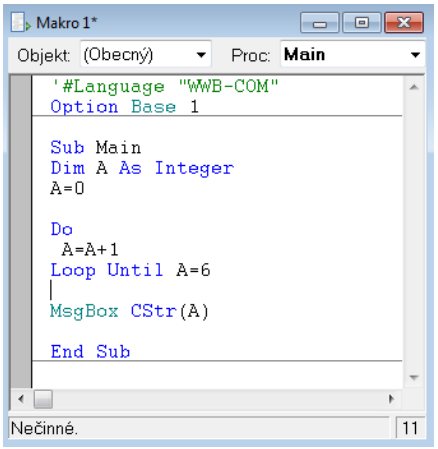
Příklady:



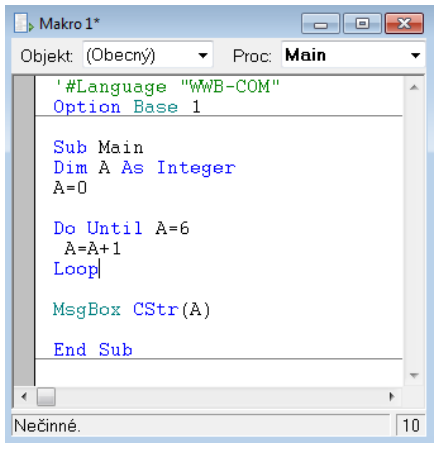
„A" je 1, jelikož cyklus prošel pouze jednou a na konci nebyla splněna podmínka, jelikož „A" bylo v té době rovno 1.



„A" je 0, jelikož podmínka nebyla splněna již při vstupu do cyklu a cyklus neproběhl ani jednou.



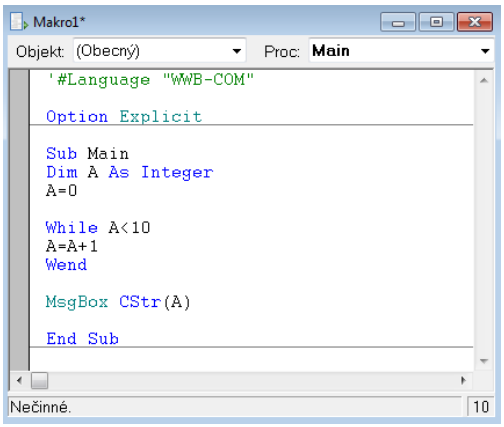
„A" je 6. Cyklus skončil právě tehdy, kdy „A" nabylo hodnoty 6. Pozor na to, abychom testovali hodnoty, které opravdu mohou nastat, abychom se nedostali do nekonečné smyčky.



"A" je opět 6.

While .. Wend

Je obdobou cyklu *Do .. Loop* s podmínkou *While* na začátku a cyklus mezi klíčovými slovy *While .. Wend* je vykonáván tak dlouho, dokud je podmínka rovna *True*:



„A" nabyde hodnoty 10.

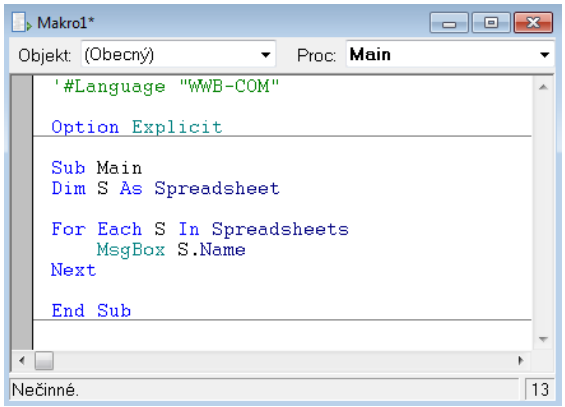
For each .. Next

Jedná se o velice zajímavý cyklus, který prochází všechny položky v nějaké kolekci. Můžeme tak např. projít všechny otevřené tabulky Statistica. Konstrukce toho cyklu je:

For each typ_polozky in kolekce_polozek

Next

Jako *typ_polozky* musíme použít již nadeklarovanou proměnnou, typově odpovídající prvku kolekce. Pro příklad s tabulkami Statistica by to vypadalo následovně:



Makro nám bude postupně procházet a vypisovat názvy aktuálně otevřených tabulek Statistica, a to i těch, které jsou otevřeny pouze na pozadí a nejsou viditelné (vlastnost *Visible=false*).

Závěrem

Dnes jsme si ukázali práci s procedurami a funkcemi, jakožto základními prvky strukturovaných maker. Také jsme se naučili předávat hodnoty proměnných mezi jednotlivými bloky programu, což je důležité zejména u rozsáhlejších projektů. V příštím díle si snad konečně ukážeme, jak získat makra generovaná programem Statistica, což již bylo avizováno již v minulém cyklu. Látka je však obsáhlejší a není možné vynechat části, které jako celek do sebe neoddělitelně zapadají.

Dell | Information Management | Software Group
V Parku 2325/16, 148 00 Praha 4
t +420 233 325 006, **e** info@statistica.cz
www.dell.cz/statsoft, www.statsoft.cz