KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI
www.inf.upol.cz

Michal Krupka
krupka.inf.upol.cz
michal.krupka@upol.cz
17. listopadu 12, 771 46 Olomouc

Paradigmata programování 1 ⋄ poznámky k přednášce

5. Vedlejší efekt

1 Rozšíření vyhodnocovacího procesu

Z minulých přednášek víme, jak probíhá vyhodnocovací proces ve Scheme. Teď si povíme o dvou jeho rozšířeních.

Víme, že základním principem práce programu ve Scheme je, že během práce programu se **vyhodnocují symbolické výrazy**. Získaná hodnota výrazu se buď použije jako argument pro další výraz (v případě složených symbolických výrazů), nebo je už konečným výsledkem výpočtu. Přitom se nic jiného s programem nestane.

Už jsme se ale setkali s případy, které tomuto čistému pojetí vyhodnocování výrazů nevyhovují. Při následujícím vyhodnocení se stanou hned dvě věci, které jsou s ním v rozporu:

```
> (define a 10)
>
```

Výraz nemá („nevrací“) žádnou hodnotu. Místo toho vykoná změnu, jejíž dopad bude trvat i po vyhodnocení. V tomto případě to změnou je, že se do výchozího (globálního) prostředí přidá nová vazba symbolu `a`. Taková okolnost se nazývá vedlejším efektem vyhodnocení výrazu.

K *vedlejšímu efektu* vyhodnocení výrazu dojde v případě, že kromě vrácení hodnoty výraz způsobí nějakou vnější změnu. Tou změnou může být (jako v uvedeném příkladě) změna výchozího prostředí, nebo třeba tiskový nebo jiný výstup.

V imperativním programovacím stylu (na rozdíl od našeho víceméně funkcionálního) je běžným vedlejším efektem změna hodnoty proměnné, případně jiného úložiště hodnot. Tento typ vedlejšího efektu používat nebudeme.

Jednoduchým příkladem procedury, která vykoná vedlejší efekt, je procedura `display`, která tiskne na obrazovku zadaný text. Než si tuto proceduru ukážeme, řekneme si rychle něco o textových řetězcích.

Textový řetězec (*string*) je libovolný text, zapisuje se do uvozovek. Z pohledu vyhodnocovacího procesu je textový řetězec **atom**, jeho hodnotou je vždy tentýž řetězec:

```
> "Dobrý den"
"Dobrý den"
```

Zpět k proceduře `display`. Procedura slouží k vypsání daného textu na obrazovku. Typické použití je výpis textového řetězce:

```
> (display "Dobrý den")
Dobrý den
```

Výsledkem vyhodnocení není žádná hodnota; došlo ale k vedlejšímu efektu, kterým je vytištění textu (fialovou barvou).

Procedura umožňuje vytisknout i jiné hodnoty než textové řetězce:

```
> (display (+ 1 1))
2
```

Je třeba si jasně uvědomovat rozdíl mezi vytisknutím a vracením hodnoty. Zdrojový kód k této přednášce vám v tom pomůže.

Drobnost k tisku: skok na nový řádek během tisku umožňuje procedura `newline` (aplikovaná bez argumentu).

A drobnost k nevracení hodnoty: na to se dá použít procedura `void`:

```
> (void)
>
```

Při práci s vedlejším efektem se hodí sekvenčně (tedy za sebou, pořadě) vyhodnocovat výrazy. K tomu slouží speciální operátor `begin`. Ten akceptuje libovolný nenulový počet argumentů. Při vyhodnocení výrazu

```
(begin expr1 expr2 ... exprn)
```

se postupně (zleva doprava) vyhodnotí všechny výrazy ***expr1 expr2 ... exprn***, hodnota posledního (tedy ***exprn***) se vrátí jako výsledek. Případné ostatní hodnoty se ignorují.

Příklady na použití speciálního operátoru `begin` najdete v kódu k této přednášce.

2 Želví grafika

Na přednášce jsme si vedlejší efekt demonstrovali také na želví grafice. Příklady najdete v příslušných zdrojových kódech, tady jen popíšu základní použité procedury.

Zdrojový soubor by měl na začátku obsahovat dva řádky:

```
(require graphics/turtles)
(turtles #true)
```

První zpřístupní procedury želví grafiky, druhý otevře kreslicí okno se želvou.

Popis procedur želví grafiky:

```
(draw length)
```

Želva se posune dopředu o délku *length* v pixelech. Při přesunu kreslí čáru.

```
(move length)
```

Želva se posune dopředu o délku *length* v pixelech. Při přesunu nic nekreslí.

```
(turn angle)
```

Želva se otočí proti směru hodinových ručiček o úhel *angle* (ve stupních).

```
(clear)
```

Kreslicí okno se vymaže a želva vrátí do počáteční polohy.

```
(tprompt expr)
```

Vyhodnotí výraz *expr*. Pokud během vyhodnocení želva změní polohu nebo směr, bude nakonec vrácena do polohy před vyhodnocením. (`tprompt` je speciální operátor.)

Otázky a úkoly na cvičení

1. Navrhněte a vyzkoušejte způsob, jak zjistit, jestli Scheme před aplikací procedury vyhodnocuje argumenty zleva doprava, nebo zprava doleva.
2. Vylepšete proceduru `fact-2` tak, aby při tisku lépe odsazovala řádky. Například výpočet faktoriálu 5 by měl před vrácením správného výsledku vytisknout toto:

```
fact volán s argumentem 5
 fact volán s argumentem 4
  fact volán s argumentem 3
   fact volán s argumentem 2
    fact volán s argumentem 1
     fact volán s argumentem 0
     fact vrátil 1
    fact vrátil 1
   fact vrátil 2
  fact vrátil 6
 fact vrátil 24
fact vrátil 120
```

3. Napište proceduru pro želví grafiku, která vykreslí obrázky podobné následujícím obrázkům: