Deklaratívne programovanie webových aplikácií

1. prednáška

Ján Kľuka

Katedra aplikovanej informatiky
FMFI UK Bratislava

23. 9. 2008

Obsah

O predmete

- Úvod do jazyka Haskell
- Opakovanie techník deklaratívneho programovania v Haskelli (rekurzia, zoznamy, stromové štruktúry)
- XML ako stromová štruktúra
- Zobrazovanie a spracovanie XML
- SVG
- Vstup/výstup v Haskelli
- Interaktívne webové aplikácie
  - formulárové
  - s trochou AJAXu

Organizácia a pravidlá

- Učíme prvý rok
- Voľnejšia forma
- Prvých 8 týždňov: domáce úlohy po 5 bodov
- Zvyšok semestra a skúškové: projekt, priebežná kontrola (60 bodov)

Haskell

- Funkcionálny programovací jazyk
- Čisto deklaratívny
- Typovaný
- Prakticky orientovaný
- Imperatívne črty (polia, vstup/výstup) bez narušenia deklaratívnosti (vďaka funkcionálom)
- Viacero implementácií, interpretery aj kompilátory
- Veľká knižnica modulov
- http://haskell.org
- Začneme interpreterom Hugs, neskôr prejdeme na kompilátor GHC
- http://haskell.org/hugs

Jednoduché výrazy

Výrazy možno vyhodnocovať priamo v príkazovom riadku intepretera Hugs:

- Konštanty: celé čísla (`123`, `-5`), znaky (`'c'`), reťazce (`"abcd"`)
- Infixové aritmetické operátory so zvyčajnou prioritou:
  `2+(9-4)^2*21/7`
- Porovnania sú boolovské funkcie (vrátia True alebo False), nízka priorita:
  `11+2 < 55`
- Mená funkcií (a premenných):
  prvý znak: `a-z` alebo `_`, ďalšie znaky: `a-z`, `A-Z`, `0-9`, `_`
- Aplikácia funkcie: `fun arg`; vyššia priorita ako infixové operátory
  `sqrt 2+2 = (sqrt 2)+2`,
  zátvorkuje sa doľava
  `sqrt sqrt 2 = (sqrt sqrt) 2`,
  preto treba písať
  `sqrt (sqrt 2)`

Definície funkcií

Definície funkcií musia byť v súbore.

```
Hugs> :edit lec1.hs
```

Funkcie sa definujú rovnicami:

$$five = 5$$

$$sq\ x = x * x$$

$$isZero\ n = n == 0$$

```
Hugs> :load lec1.hs.
```

Definované funkcie potom môžeme používať vo výrazoch:
five, *sq five*, *isZero* (*sq* 20)

Komentáre:

-- jednoriadkové (do konca riadka)

{- viacriadkové -}

Zložené výrazy

if-then-else

$$sgn\ x = \textbf{if}\ x > 0\ \textbf{then}\ 1\ \textbf{else}\ 0$$

case-of

$$sgn_1\ x = \textbf{case}\ x\ \textbf{of}\ \{\ 0 \to 0;\quad \_ \to 1\ \}$$

„_“ znamená „čokoľvek“

Prehľadnejší zápis s odsadením:

$$sgn_2\ x = \textbf{case}\ x\ \textbf{of}$$
$$0 \to 0$$
$$\_ \to 1$$

Podobne ako v CL môžeme dosadiť možnosti za premennú x:

$$sgn_3\ 0 = 0$$
$$sgn_3\ \_ = 1$$

Výlučnosť

Možnosti v **case** nemusia byť výlučné (aj 0 je „čokoľvek“).
Použije sa prvá vyhovujúca:

$$sgn_4\ 0 = 0$$
$$sgn_4\ \_ = 1$$
$$sgn_4\ 9 = 999$$

```
Main> sgn4 9
1
```

Rekurzia

Nič prekvapujúce:

$$\begin{aligned} &\mathit{factorial}\ n = \textbf{case}\ n\ \textbf{of} \\ &\qquad\qquad 0 \to 1 \\ &\qquad\qquad k+1 \to n * \mathit{factorial}\ k \end{aligned}$$

alebo dvomi rovnicami

$$\begin{aligned} &\mathit{sumFirst}\ 0 = 0 \\ &\mathit{sumFirst}\ (n+1) = (n+1) + \mathit{sumFirst}\ n \end{aligned}$$

Notoricky známa funkcia počítajúca množenie králikov

$$\begin{aligned} &\mathit{fib}\ 0 = 1 \\ &\mathit{fib}\ 1 = 1 \\ &\mathit{fib}\ (x+2) = \mathit{fib}\ (x+1) + \mathit{fib}\ x \end{aligned}$$

Viac argumentov, pomocné premenné

Ďalší argument sa pripája medzerou

$$max\ x\ y = \textbf{if}\ x >= y\ \textbf{then}\ x\ \textbf{else}\ y$$

Rekurzia s akumulátorom

$$facta\ 0\ a = a$$
$$facta\ (n+1)\ a = facta\ n\ ((n+1)*a)$$

$$factorial_1\ n = facta\ n\ 0$$

Pomocné premenné – zložený výraz **let-in**

$$facta_1\ 0\ a = a$$
$$facta_1\ (n+1)\ a = \textbf{let}\ b = (n+1)*a\ \textbf{in}$$
$$facta_1\ n\ b$$

Viac a podrobnejšie o Haskelli

Hal Daume III et al.: *Yet Another Haskell Tutorial*

http://www.haskell.org/haskellwiki/Tutorials