Paradigme de programare

Laborator 6

Mihai Nan

Facultatea de Automatica si Calculatoare
Universitatea Politehnica din Bucuresti

Anul universitar 2015-2016

Cuprins

Cuprins

Haskell

- Limbaj **pur** functional
- Nu exista efecte laterale
- Limbaj cu tipare statica
- Limbaj cu evaluare lenesa
- Fiecare functie are un tip

Cuprins

QuickSort

```haskell
-- Type annotation (optional)
qsort :: Ord a => [a] -> [a]

qsort [] = []

qsort (p:xs) = (qsort lesser) ++ [p] ++ (qsort greater)
    where
        lesser  = filter (< p) xs
        greater = filter (>= p) xs
```

Liste

```haskell
let l1 = [1,2,3]
let l2 = [4,5,6]
(++) l1 l2 -- rezultat [1, 2, 3, 4, 5, 6]
(:) 0 [1,2,3] -- rezultat [0, 1, 2, 3]
rev :: [t] -> [t]
-- varianta 1
rev [] = []
rev xs = last xs : rev (init xs)
-- varianta 2
rev [] = []
rev (x:xs) = reverse xs ++ [x]
-- varianta 3
rev [] = []
rev xs = rev (tail xs) ++ [head xs]
```

Liste

```
head [1, 2, 3, 4] -- capul listei => 1
tail [1, 2, 3, 4] -- coada listei => [2, 3, 4]
last [1, 2, 3, 4] -- ultimul element din lista => 4
init [1, 2, 3, 4] -- lista fara ultimul element => [1, 2, 3]
[1, 2, 3, 4] !! 1 -- elementul de pe pozitia 1 => 2
```

Functionale

```haskell
-- Tipul functionalei map
map :: (a -> b) -> [a] -> [b]
map even [1..5] -- [False,True,False,True,False]
map (+5) [1..10] -- [6,7,8,9,10,11,12,13,14,15]
-- Implementarea functionalei map
map f [] = []
map f (x:xs) = (f x):(map f xs)

-- Tipul functionalei filter
filter :: (a -> Bool) -> [a] -> [a]
filter even [1..10] -- => [2,4,6,8,10]
-- Implementarea functionalei filter
filter p []                   = []
filter p (x:xs) | p x         = x : filter p xs
                | otherwise = filter p xs
```

Functionale

```
-- Functii lambda
map (\x -> x*x) [1..10] -- => [1,4,9,16,25,36,49,64,81,100]

-- Suma elementelor dintr-o lista
foldl (+) 0 [1, 2, 3] -- => 6
foldl (\acc x -> if x `elem` "aeiou"
                          then acc+1
                          else acc)  0 "hello world"

foldl (\acc item -> acc ++ [item]) [] [1, 2, 3, 4, 5]
foldr (\item acc -> [item] ++ acc) [] [1, 2, 3, 4, 5]
```

Functii

```
f :: Num a => a -> a -> a
f x y = x + y

g :: Num t => t -> (t, t)
g x = (x, 2 * x)

h :: Num a => a -> a
h x = x - 1

t :: Num a => a -> a
t x = x + 1

g(t(h(8)))
(g.t.h)(8)
```

Compunere

```
f1 :: Num a => (a, a) -> a
f1 x = (fst x) + (snd x)

z x = g(t (h (x)))

z = g . t . h

z (8)
```

List comprehensions

$\{ x \mid x \in [1..10],\ x > 5 \}$

$[1..10] = [1,2,3 \ldots 10]$

$[1,3..10] = [1,3,5,7,9]$

```
[x | x <- [1..10], x > 5] -- => [6, 7, 8, 9, 10]
[x*2 | x <- [1..10], x*2 >= 12]  -- => [12,14,16,18,20]
[ x | x <- [10..20], x /= 13, x /= 15, x /= 19]
[ x*y | x <- [2,5,10], y <- [8,10,11]]
```

$[(x,y) \mid x \leftarrow l_1, y \leftarrow l_2]$

Bibliografie I

Andrei Olaru
Suport de curs - Seria CC
Anul universitar 2014-2015

Mihnea Muraru
Suport de curs - Seria CA
Anul universitar 2014-2015

Echipa de PP
Breviar laborator
Anul universitar 2015-2016