Haskell 简介

张凯

2015 年 8 月 5 日

# 介绍

## 开胃菜

* pandoc -- 文件格式转换的利器
  + 读入: Markdown，reStructuredText, LaTeX, DocBook，EPUB，Word docx 等等格 式
  + 输出: DZSlides, reveal.js, beamer; HTML5, pdf, Word docx, LaTeX
* xmonad -- 平铺式窗口管理器
  + 资源占用少
  + 基本功能一应俱全
  + 配置文件即 Haskell 程序，定制灵活

## Haskell 的优点

* 没有副作用 -> 易于查错，测试
* 变量不可变 -> 易于执行并发操作
* 惰性求值 -> 提高效率，表达无穷列表
* 静态类型 + 强大的类型推导系统 -> 节省代码量
* 尾递归优化 -> 代码更简洁

## Haskell 的缺点

* 非主流 -> 目前主要在学术界流行，工业界用的较少
* 执行效率不如 C、C++ 高
* 学习曲线陡峭

# 细节

## 基本语法

* doubleUs x y = x\*2 + y\*2
  + doubleUs 4 9 -> 26
* list
  + :, ++, !!;
  + head, tail; last, init;
  + take, drop;
  + maximum, minimum; sum, product
  + elem, .., repeat
  + [x\*2 | x <- [1..10], x\*2 >= 12]

## 基本语法 2

* tuple
  + fst (8, 11)
  + snd
  + zip

## type

addThree :: Int -> Int -> Int -> Int  
addThree x y z = x + y + z

* curry
  + Int -> (Int -> (Int -> Int))
  + ((addThree x) y) z

## Function

* Pattern matching

lucky :: (Integral a) => a -> String  
lucky 7 = "LUCKY NUMBER SEVEN!"  
lucky x = "Sorry, you're out of luck, pal!"

* Guards

bmiTell :: (RealFloat a) => a -> String  
bmiTell bmi  
 | bmi <= 18.5 = "You're underweight, you emo, you!"  
 | bmi <= 25.0 = "You're supposedly normal. Pffft, I bet you're ugly!"  
 | bmi <= 30.0 = "You're fat! Lose some weight, fatty!"  
 | otherwise = "You're a whale, congratulations!"

## Function 2

* case

describeList :: [a] -> String  
describeList xs =  
 "The list is " ++ case xs of [] -> "empty."  
 [x] -> "a singleton."  
 xs -> "a longer list."

## 递归

quicksort :: (Ord a) => [a] -> [a]  
quicksort [] = []  
quicksort (x:xs) =  
 let smallerSorted = quicksort [a | a <- xs, a <= x]  
 biggerSorted = quicksort [a | a <- xs, a > x]  
 in smallerSorted ++ [x] ++ biggerSorted

## Map and filter

map :: (a -> b) -> [a] -> [b]  
map \_ [] = []  
map f (x:xs) = f x : map f xs  
filter :: (a -> Bool) -> [a] -> [a]  
filter \_ [] = []  
filter p (x:xs)  
 | p x = x : filter p xs  
 | otherwise = filter p xs

ghci> map (+3) [1,5,3,1,6]  
[4,8,6,4,9]  
ghci> filter (>3) [1,5,3,2,1,6,4,3,2,1]  
[5,6,4]

## Fold, ($) and (.)

sum' :: (Num a) => [a] -> a  
sum' xs = foldl (\acc x -> acc + x) 0 xs  
  
map' :: (a -> b) -> [a] -> [b]  
map' f xs = foldr (\x acc -> f x : acc) [] xs

($) :: (a -> b) -> a -> b  
f $ x = f x

(.) :: (b -> c) -> (a -> b) -> a -> c  
f . g = \x -> f (g x)

## Modules

Geometry/  
 |--Sphere.hs  
 |--Cuboid.hs  
 |--Cube.hs

Sphere.hs

module Geometry.Sphere  
( volume  
, area  
) where  
  
volume :: Float -> Float  
volume radius = (4.0 / 3.0) \* pi \* (radius ^ 3)  
  
area :: Float -> Float  
area radius = 4 \* pi \* (radius ^ 2)

## Define our type

data Car = Car { company :: String  
 , model :: String  
 , year :: Int  
 } deriving (Show)  
  
ghci> Car {company="Ford", model="Mustang", year=1967}  
Car {company = "Ford", model = "Mustang", year = 1967}

type String = [Char]

newtype CharList = CharList { getCharList :: [Char] }  
 deriving (Eq, Show)

## 闭包

class Monoid m where  
 mempty :: m  
 mappend :: m -> m -> m  
 mconcat :: [m] -> m  
 mconcat = foldr mappend mempty

## Monad

* 带 context 的类型
* 只要实现 class Monad m，就不必关心 context 细节了，只需考虑传递的变量
* Monad 实现了 Haskell pure 部分与非 pure 部分的隔离