Лекция 5. Работа со списками в Haskell

Пережогин А.С.

8 октября 2012 г.

Создание нового списка по заданному

- Создание списка из списка осуществляется с помощью

  ```
  [x^2 | x <- [1..5]]
  ```

  Результат

  ```
  [1,2,3,4,5] преобразуется в [1,4,9,16,25]
  ```

- Генератор списка

  ```
  x <- [1..5]
  ```

- Можно использовать несколько генераторов. Список кортежей (x,y)

  ```
  [(x,y) | x <- [1,2,3], y <- [4,7]]

  [(1,4), (1,7), (2,4) , (2,7), (3,4) , (3,7)]
  ```

- Можно изменить порядок элементов. Список кортежей (x,y)

  ```
  [(x,y) | y <- [4,7], x <- [1,2,3]]

  [(4,1), (4,2), (4,3) , (7,1), (7,2) , (7,3)]
  ```

Зависимые генераторы с условием

- Генераторы списка элементов могут быть зависимы.

```
[(x,y) | x <- [1..3], y <- [x..3]]
[(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)] - список чисел при x<=y
```

- Функция объединения concat

```
concat    :: [[a]] <- [a]
concat xss = [x | xs <- xss, x <- xs]

> concat [[1,2,3],[4,5],[6]]
[1,2,3,4,5,6]
```

- В генераторе списка можно накладывать условиях на аргумент x

```
[x | x <- [1..10], even x]
[2,4,6,8,10]
```

- Используя условия, можно определить функцию нахождения списка сомножителей

Зависимые генераторы с условием

```
factors :: Int -> [Int]
factors n = [x | x <- [1..10], n `mod` x == 0]

> factors 15
[1,3,5,15]
```

- Нахождение простого числа

```
prime :: Int -> Bool
prime n = factors n == [1,n]

>prime 15
False
>prime 7
True
```

Далее можно сгенерировать список простых чисел

```
primes :: Int -> [Int]
primes n = [x| x<-[2..n],prime x ]

>primes 20
[2,3,5,7,11,13,17,19]
```

Zip функция

- zip функция создает из двух списков список кортежей

```
zip :: [a] -> [b] -> [(a,b)]

>zip ['a','b','c'] [1,2,3]
[('a',1),('b',2),('c',3)]
```

- Функция генерирования пар элементов из списка

```
pairs   :: [a] -> [(a,a)]
pairs xs = zip xs (tail xs)
```

- Создадим функцию проверки является список отсортированным или нет

```
sorted   :: Ord a => [a] -> Bool
sorted xs =
   and [x <= y | (x,y) <- pairs xs]
```

- Список позиций заданного значения в списке

```
positions :: Eq a => a -> [a] -> [Int]
positions x xs =
   [i | (x',i) <- zip xs [0..n], x == x']
   where n = length xs - 1
```

Представление строк в Haskell

- Строка представляет собой список символов

```
"abc" :: String

['a','b','c'] :: [Char]
```

- Пример функции для вычисления числа строчных букв с строке

```
lowers   :: String -> Int
lowers xs =
   length [x | x <- xs, isLower x]
```

Контрольные задания

- Создать программу, которая отвечает удовлетворяет ли тройка чисел условию $x^2 + y^2 = z^2$
- Найти скалярное произведение двух векторов