Лекция 7. Функции высших порядков

Пережогин А.С.

8 октября 2012 г.

Функции высших порядков

Функция называется функцией высшего порядка, если в качестве аргумента принимает функцию или возвращает функцию как результат.

```
twice :: (a -> a) -> a -> a
twice f x = f ( f x )
```

Функция map

Библиотечная функция высшего порядка map применяется к каждому элементу списка

```
map :: (a -> b) -> [a] -> [b]
```

Пример.

```
> map (+1) [1,3,5,7]
[2,4,6,8]
```

Функция map может быть определена с помощью генератора списка

```
map f xs = [f x | x <- xs]
```

или с помощью рекурсии

```
map f [] = []
map f xs = f x : map f xs
```

Функция filter

Функция фильтрации высокого порядка выбирает значения из списка, которые удовлетворяют условию.

```
filter :: (a -> Bool) -> [a] -> [a]
```

Например,

```
> filter even [1..10]
[2,4,6,8,10]
```

Объявление функции filter с помощью генератора списка

```
filter p xs = [x | x <- xs, p x]
```

или с помощью рекурсии

```
filter p [] = []
filter p (x:xs)
    | p x = x : filter p xs
    | otherwise = filter p xs
```

Функция foldr

Функции обработки списков могут быть представлены следующей простой схемой

```
f [] = v
f (x:xs) = x (+) f xs
```

Примеры.

```
sum [] = v
sum (x:xs) = x + sum xs
```

```
product [] = v
product (x:xs) = x * product xs
```

```
and [] = True
and (x:xs) = x && and xs
```

Для того, чтобы объединить все приведенные примеры, существует функция высокого порядка foldr.

```
sum     = foldr (+) 0
product = foldr (*) 1
or      = foldr (||) False
and     = foldr (&&) True
```

Функция foldr

Сама функция foldr определяется с помощью рекурсии:

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f v [] = v
foldr f v (x:xs) = f x (foldr f v xs)
```

Пример. length

```
length = foldr (\_ n -> 1+n) 0
```

То есть f – (λ _ n -> 1+n), v = 0.

Пример. reverse

```
reverse = foldr (\_ x xs -> xs ++ [x]) []
```

С помощью функции foldr многие функции выглядят более компактно и удобно для дальнейшего анализа.

Другие библиотечные функции

Вычисление композиции двух функций выполняется с помощью (.)

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

Пример.

```
odd :: Int -> Bool
odd  = not . even
```

Пример. all – определяет удовлетворяют ли все элементы списка условию

```
all :: (a -> Bool) -> [a] -> Bool
all p xs = and [p x | x <- xs]

> all even [2,4,6]
True
```

Контрольные задания

Переопределить функцию map f и filter p через функцию foldr.