Лекция 5: функции как значения

Функциональное программирование на Haskell

Алексей Романов

7 марта 2018

МИЭТ

Функции как значения

- Как упоминалось в начале курса, одно из оснований ФП состоит в том, что функции могут использоваться как значения.
- В Haskell можно выразиться сильнее:

Функции как значения

- Как упоминалось в начале курса, одно из оснований ФП состоит в том, что функции могут использоваться как значения.
- В Haskell можно выразиться сильнее:
- Функции это и есть просто значения, тип которых имеет форму `ТипПараметра -> ТипРезультата` для каких-то ТипПараметра и ТипРезультата.

Функции как значения

- Как упоминалось в начале курса, одно из оснований ФП состоит в том, что функции могут использоваться как значения.
- В Haskell можно выразиться сильнее:
- Функции это и есть просто значения, тип которых имеет форму `ТипПараметра -> ТипРезультата` для каких-то ТипПараметра и ТипРезультата.
- Мы уже видели примеры этого в равноправии функций и переменных.

Функции как значения

- Как упоминалось в начале курса, одно из оснований ФП состоит в том, что функции могут использоваться как значения.
- В Haskell можно выразиться сильнее:
- Функции это и есть просто значения, тип которых имеет форму `ТипПараметра -> ТипРезультата` для каких-то ТипПараметра и ТипРезультата.
- Мы уже видели примеры этого в равноправии функций и **других** переменных.

Функции высших порядков

- В частности, функции могут принимать на вход функции.
- То есть тип параметра сам может быть функциональным типом.
- Тривиальный пример:

```
foo :: (Char -> Bool) -> Bool
foo f = f 'a'

Prelude Data.Char> foo isLetter
```

Функции высших порядков

- В частности, функции могут принимать на вход функции.
- То есть тип параметра сам может быть функциональным типом.
- Тривиальный пример:

```
foo :: (Char -> Bool) -> Bool
foo f = f 'a'

Prelude Data.Char> foo isLetter
True
```

- Скобки вокруг типа параметра здесь необходимы.

Функции высших порядков

- В частности, функции могут принимать на вход функции.
- То есть тип параметра сам может быть функциональным типом.
- Тривиальный пример:

```
foo :: (Char -> Bool) -> Bool
foo f = f 'a'

Prelude Data.Char> foo isLetter
True
```

- Скобки вокруг типа параметра здесь необходимы.
- Функции, параметры которых — функции, называются *функциями высших порядков (ФВП)*.

Функции высших порядков

- В частности, функции могут принимать на вход функции.
- То есть тип параметра сам может быть функциональным типом.
- Тривиальный пример:

```
foo :: (Char -> Bool) -> Bool
foo f = f 'a'

Prelude Data.Char> foo isLetter
True
```

- Скобки вокруг типа параметра здесь необходимы.
- Функции, параметры которых — функции, называются *функциями высших порядков (ФВП)*.
- Часто ими также считают функции, возвращающие функции, но в Haskell нет (скоро увидим почему).

Лямбда-выражения

- TODO

Функции применения и композиции функций

- TODO

Избавление от скобок

- TODO

Частичное применение

- TODO

η-редукция (сокращение аргументов)

- TODO

Сечения операторов

- TODO

Ограничение мономорфизмом

- TODO

Основные функции 2-го порядка над списками

- TODO

Бесточечный стиль

- TODO (нужно ли?)