Декларативное программирование

Функции высших порядков

Аккумулятор

```
factorial :: Integer -> Integer
factorial n = factorial' 1 n

factorial' :: Integer -> Integer -> Integer
factorial' acc n
  | n <= 1       = acc
  | otherwise    = factorial' (acc*n) (n - 1)
```

Функции как аргументы

```
doTwice :: (Int -> Int) -> Int -> Int
doTwice f x = f (f x)

plus1 :: Int -> Int
plus1 x = x + 1

baz = doTwice plus1 7
```

Лямбда-функции как аргументы

```
plus1 x = x + 1

minus1 x = x – 1

doTwice f x = f (f x)


baz = doTwice plus1 3

bar = doTwice minus1 7
```

Лямбда-функции как аргументы

```
plus1 x = x + 1
minus1 x = x – 1
doTwice f x = f (f x)

baz = doTwice plus1 3
bar = doTwice minus1 7

baz' = doTwice (\x -> x + 1) 3
bar' = doTwice (\x -> x - 1) 7
```

Лямбда-функции как аргументы

```
plus1 x = x + 1
minus1 x = x – 1
doTwice f x = f (f x)

baz = doTwice plus1 3
bar = doTwice minus1 7

baz' = doTwice (\x -> x + 1) 3
bar' = doTwice (\x -> x - 1) 7
```

Когда применять анонимные функции?

Лямбда-функции как аргументы

```
plus1 x = x + 1
minus1 x = x – 1
doTwice f x = f (f x)

baz = doTwice plus1 3
bar = doTwice minus1 7

baz' = doTwice (\x -> x + 1) 3
bar' = doTwice (\x -> x - 1) 7
```

Когда применять анонимные функции?

Когда функции небольшие и не используются повторно.

Возвращаем функции

```
plusn :: Int -> (Int -> Int)
plusn n = f
          where f x = x + n
```

Упражнение:

Перепишите с использованием let-выражения

Частичное применение

```
(plusn 25) 100 == plusn 25 100
```

Частичное применение

```
(plusn 25) 100 == plusn 25 100


plusn' :: Int -> Int -> Int
plusn' n x = x + n
```

Упражнение

Какие из следующих типов эквивалентны?

1. `Int -> Int -> Int`
2. `Int -> (Int -> Int)`
3. `(Int -> Int) -> Int`

Упражнение

Напишите функцию, которая суммирует все целые числа в интервале [a,b]

```
sumInts :: Int -> Int -> Int
sumInts a b = undefined

>sumInts 1 3
6
```

Упражнение

Напишите функцию, которая суммирует квадраты целых чисел в интервале [a,b]

```
sumSquares :: Int -> Int -> Int
sumSquares a b = undefined

> sumSquares 1 3
14
```

Упражнение

Напишите функцию, которая применяет функцию к каждому числу в интервале [a,b] и суммирует результаты.

```
higherOrderSum :: (Int -> Int) -> Int -> Int -> Int
higherOrderSum f a b | a > b = 0

> higherOrderSum (^2) 1 3
14
```

Упражнение

Функция sumInts выполняла операцию сложения между целыми числами в интервале.

Например, 1 + 2 +3. Обобщите эту идею и напишите функцию, которая выполняет заданную пользователем операцию между числами. Например, умножение: 1*2*3.

Выразите факториал через эту функцию.