Функции высшего порядка (Higher-order functions, HOFs)

Данил Браун

2022

Определение 1

Функцией высшего порядка называют функцию, которая принимает или[1] возвращает функцию.

[1] или — не исключительное!

Определение 1

Функцией высшего порядка называют функцию, которая принимает или[1] возвращает функцию.

Пример 1

```
fun fun-plus-one(num :: Number,
                 func :: (Number -> Number)) -> Number:
  func(num) + 1
end
```

[1] или — не исключительное!

Пример 2

```
check:
  f-to-c([list: 131, 77, 68]) is [list: 55, 25, 20]
  check-temp([list: 131, 77, 68]) is [list: "слишком
      жарко",
    "ок", "очень холодно"]
  # >90, <70, ...
end
```

Пример 2

```
check:
  f-to-c([list: 131, 77, 68]) is [list: 55, 25, 20]
  check-temp([list: 131, 77, 68]) is [list: "слишком
      жарко",
    "ок", "очень холодно"]
  # >90, <70, ...
end
```

Упражнение 1

Напишите определения `f-to-c` и `check-temp`.

Пример 2

```
check:
  f-to-c([list: 131, 77, 68]) is [list: 55, 25, 20]
  check-temp([list: 131, 77, 68]) is [list: "слишком
      жарко",
    "ок", "очень холодно"]
  # >90, <70, ...
end
```

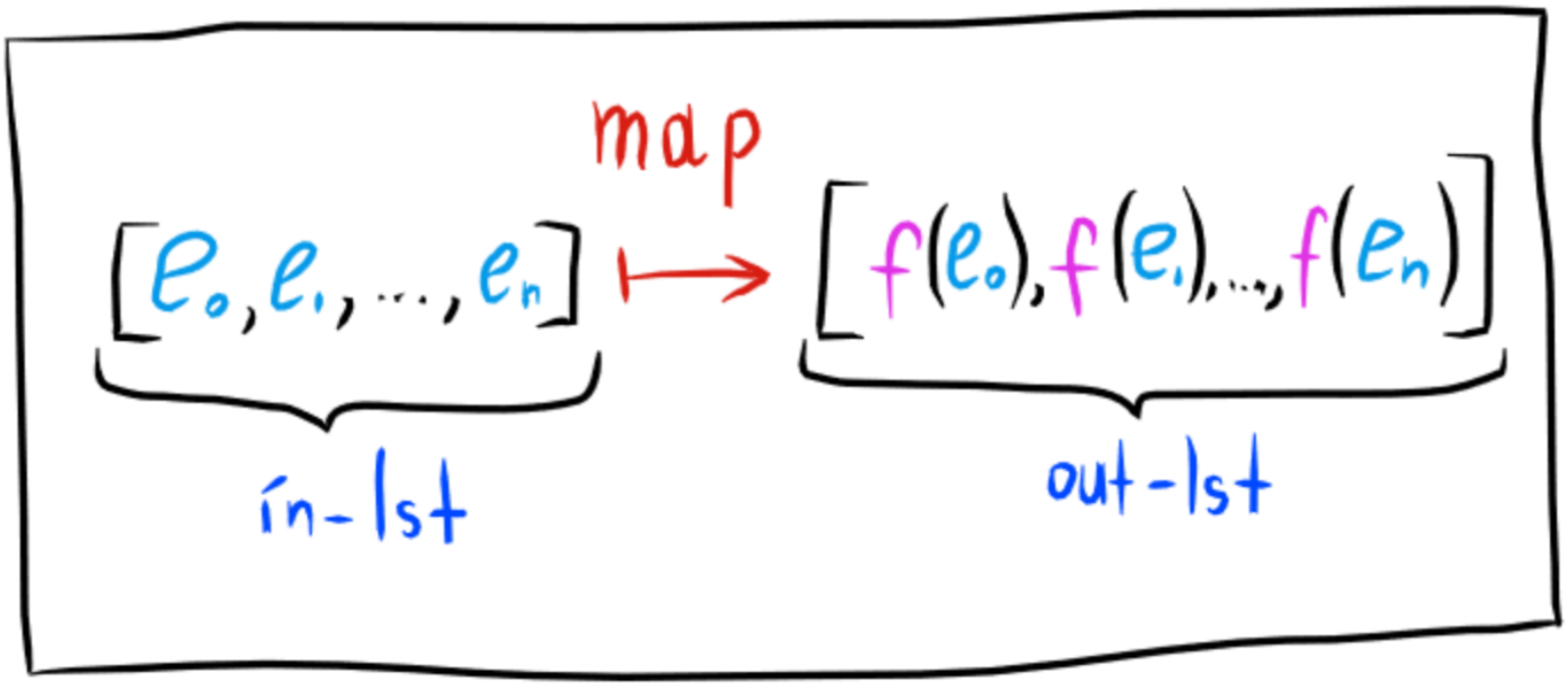
Упражнение 1

Напишите определения `f-to-c` и `check-temp`.

Упражнение 2

Что у них общего?

Map



λ-выражения

Определение 2

λ-выражением называется выражение, создающее **анонимную функцию**. λ-выражение имеет вид

```
lam(<parameters>): <function body> end
```

Замечание 1

Существует короткая версия λ-выражения:

```
{(<parameters>): <function body>}
```

λ-выражения

Пример 3

```
1  >>> lam(x): x * 2 end
2  <function:anonymous>
3  >>> (lam(x): x * 2 end)(4)
4  8
5  >>> f = {(a, b): string-append(a, string-append(";", b))}
6  >>> f("qwe", "rty")
7  "qwe;rty"
```

Map

Упражнение 3

Напишите определения `f-to-c` и `check-temp` с помощью `map`.

Map

Упражнение 3

Напишите определения `f-to-c` и `check-temp` с помощью `map`.

Упражнение 4

Напишите свою реализацию `map` — `my-map`.

Filter

```
check:
  lon = [list: -1, 6, -2, 5, 7, 0, -4]
  filter(lam(num): num > 0 end, lon) is [list: 6, 5, 7]
end
```

Filter

Упражнение 5

```
fun tl-dr<T>(lol :: List<List<T>>,
             length-thresh :: Number) -> List<List<T>>:

fun eliminate-e(words :: List<String>) -> List<String>:
```

Filter

Упражнение 5

```
fun tl-dr<T>(lol :: List<List<T>>,
             length-thresh :: Number) -> List<List<T>>:

fun eliminate-e(words :: List<String>) -> List<String>:
```

Упражнение 6

Напишите свою версию `filter` с именем `my-filter`.

Fold (aka Reduce)

```
fun fold<T1, T2>(f :: (T2, T1 -> T2),
                 base :: T2, lst :: List<T1>) -> T2:
```

Fold (aka Reduce)

```
fun fold<T1, T2>(f :: (T2, T1 -> T2),
                 base :: T2, lst :: List<T1>) -> T2:

check:
  lon = [list: 7, 3, 2]
  fold(lam(acc, cur): acc + cur end, 0, lon2) is 12
end
```

acc	cur	f(acc, cur)
0	7	7
7	3	10
10	2	12

Fold (aka Reduce)

```
fun add-boolean(acc :: Number, cur :: Boolean) -> Number:
  if cur: acc + 1 else: acc end
end

check:
  attempts = [list: true, false, false, true]
  fold(add-boolean, 5, attempts) is 7
end
```

acc	cur	add-boolean(acc, cur
5	true	6
6	false	6
6	false	6
6	true	7

Fold (aka Reduce)

Упражнение 7

Определите следующие функции с использованием `fold`.

```
fun list-product(lon :: List<Number>) -> Number:
```

```
fun list-max(lon :: List<Number>) -> Number:
```

Fold (aka Reduce)

Упражнение 7

Определите следующие функции с использованием `fold`.

```
fun list-product(lon :: List<Number>) -> Number:
```

```
fun list-max(lon :: List<Number>) -> Number:
```

Упражнение 8

Напишите собственную реализацию `fold`. Назовите её `my-fold`.

Map2

```
check:
  noun-list = [list: "kitten", "puppy", "student"]
  adj-list = [list: "fluffy", "ugly", "smart"]
  map2(lam(n, a): a + " " + n end, noun-list, adj-list) is
       [list: "fluffy kitten",
         "ugly puppy",
         "smart student"]
end
```

Map2

Упражнение 9

```
fun who-passed(score :: List<Number>,
               extra-credit :: List<Boolean>
) -> List<Boolean>:
```

Map2

Упражнение 9

```
fun who-passed(score :: List<Number>,
               extra-credit :: List<Boolean>
) -> List<Boolean>:
```

Упражнение 10

Напишите Вашу собственную реализацию `map2`. Назовите её `my-map2`.