Функции высшего порядка

Лекция 15: Функции как данные

Преподаватель: [Ваше имя]

Группа: 203

Семестр: Осенний 2024

План лекции

- 1. Что такое функции высшего порядка?
- 2. Map, filter, fold
- 3. Композиция функций
- 4. Частичное применение
- 5. Лямбда-функции
- 6. Практический пример: Игровая логика

Что такое функции высшего порядка?

Определение:

Функция высшего порядка — функция, которая принимает другие функции в качестве аргументов или возвращает функции.

Преимущества:

- Переиспользование кода
- Абстракция общих паттернов
- Читаемость и выразительность
- Композиция функций

Примеры:

Мар функция

Определение:

```
map :: (a -> b) -> [a] -> [b]
map _ [] = []
map f (x:xs) = f x : map f xs
```

Использование:

```
-- Применение функции к каждому элементу
doubleList :: [Int] -> [Int]
doubleList xs = map (*2) xs
-- Преобразование типов
getNames :: [Unit] -> [String]
getNames units = map unitName units
-- Работа с позициями
getXCoordinates :: [Position] -> [Int]
```

Filter функция

Определение:

Использование:

```
    — Фильтрация по условию aliveUnits :: [Unit] -> [Unit] aliveUnits units = filter isAlive units
    — Фильтрация по типу warriors :: [Unit] -> [Unit] warriors units = filter (\unit -> unitType unit == Warrior) units
    — Фильтрация по позиции
```

Fold функции

Foldr (справа налево):

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr _ z [] = z
foldr f z (x:xs) = f x (foldr f z xs)
-- Примеры использования
sumList :: [Int] -> Int
sumList xs = foldr (+) 0 xs
productList :: [Int] -> Int
productList xs = foldr (*) 1 xs
concatList :: [[a]] -> [a]
concatList xs = foldr (++) [] xs
-- Сложные примеры
countAliveUnits :: [Unit] -> Int
countAliveUnits units = foldr (\unit count ->
    if isAlive unit then count + 1 else count) 0 units
getTotalHealth :: [Unit] -> Int
getTotalHealth units = foldr (\unit total ->
    total + unitHealth unit) 0 units
findStrongestUnit :: [Unit] -> Maybe Unit
findStrongestUnit units = foldr (\unit strongest ->
```

Композиция функций

Оператор композиции:

```
(.) :: (b \rightarrow c) \rightarrow (a \rightarrow b) \rightarrow a \rightarrow c
(f \cdot g) \times = f (g \times)
-- Примеры использования
— Композиция простых функций
doubleAndIncrement :: Int -> Int
doubleAndIncrement = (+1) . (*2)
-- Композиция с функциями для работы со списками
getAliveUnitNames :: [Unit] -> [String]
getAliveUnitNames = map unitName . filter isAlive
getStrongUnitPositions :: [Unit] -> [Position]
getStrongUnitPositions = map unitPosition . filter (\unit -> unitHealth unit > 100)
-- Сложная композиция
formatAliveUnits :: [Unit] -> String
formatAliveUnits =
    concat . map (\unit -> unitName unit ++ " (HP: " ++ show (unitHealth unit) ++ ")\n")
    . filter isAlive
-- Композиция с преобразованием типов
getUnitHealths :: [Unit] -> [Int]
getUnitHealths = map unitHealth . filter isAlive
```

Частичное применение

Что такое частичное применение?

Частичное применение — создание новой функции путем фиксации части аргументов существующей функции.

Примеры:

isGreaterThan :: Int -> Int -> Bool

```
— Частичное применение с двумя аргументами add :: Int -> Int add :: Int -> Int add :: Int -> Int addFive :: Int -> Int addFive = add 5

— Частичное применение с функциями высшего порядка filterAlive :: [Unit] -> [Unit] filterAlive = filter isAlive

mapNames :: [Unit] -> [String] mapNames = map unitName

— Комбинирование частичного применения getAliveUnitNames :: [Unit] -> [String] getAliveUnitNames = map unitName . filter isAlive

— Частичное применение с операторами
```

Лямбда-функции

Синтаксис:

```
— Лямбда-функции (анонимные функции)
\xspace \xs
\unit -> unitHealth unit > 50 -- проверяет здоровье юнита
-- Использование с тар
incrementList :: [Int] -> [Int]
incrementList xs = map(x -> x + 1) xs
-- Использование c filter
strongUnits :: [Unit] -> [Unit]
strongUnits units = filter (\unit -> unitHealth unit > 100) units
-- Использование c fold
totalAttack :: [Unit] -> Int
totalAttack units = foldr (\unit total -> total + unitAttack unit) 0 units
-- Сложные лямбда-функции
formatUnitDetailed :: Unit -> String
formatUnitDetailed unit =
             let healthStatus = if unitHealth unit > 50 then "Healthy" else "Weak"
             in unitName unit ++ " (" ++ healthStatus ++ ", HP: " ++ show (unitHealth unit) ++ ")"
-- Лямбда-функции c pattern matching
qetUnitPositions :: [Unit] -> [Position]
getUnitPositions = map (\(\(\text{Unit} \_ \_ \_ \) pos \_ \_) -> pos)
-- Лямбда-функции c quards
classifyUnit :: Unit -> String
```

Практический пример: Игровая логика

```
\label{eq:period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_period_peri
      getUnitNames :: [Unit] -> [String]
getUnitNames = map unitName
      getUnitPositions :: [Unit] -> [Position]
getUnitPositions = map unitPosition
      getHealthyUnitPositions :: [Unit] -> [Position]
getHealthyUnitPositions = map unitPosition . filter isHealthy
getStronginitsInRange :: Position \rightarrow Int \rightarrow [Unit] \rightarrow [Unit] getStronginitsInRange center range : filter isHealthy . getUnitsInRange center range
             findstrongestimit:: [Buil] -> Raybe limit

[Line alivemint = fitter inline units
in find lardemints
the Marching
the March
                    Findmonthis: : [Bat1] > Maybe Unit
findmonthis units :
in four alivebiate : fitter inline units
in four latematic
in four latematic
else Just (four) (val a2 >>
if unitmatin a4 combination a2 then units as
if unitmatin a4 combination a2 then units as a) alivebiate)
      getBestTarget : Unit - (Unit) - Maybe Unit
getBestTarget attacks =
Let targets = getAttackableTargets attacker enemies
in targets = getAttackableTargets attacker enemies
                    — бумоцыя для форматирования
formathiat unit =
unitiams unit =
unitiams unit += (" ++ show (unitType unit) ++
", 197: "++ show (unitTealth unit) ++ ")" ++ show (unitMasNealth unit) ++ ")"
      formatGendinamary:: [Bail] > String
formatGendinamy: IBail] > String
formatGendinamy: unit:
totalCountry unit:
totalCountry length unit:
"White:" = share alsocate == "" == shor
"Voltail Realth: " == share length unit:
"White: Bailth: " == share alsocate == ""
"Voltail Realth: " == share alsocate == ""
"Voltail Realth: " == share length unit:
"White: Bailth: " == share problems == ""
"The segment of the segment == ""
"The segment =

    Bichosoratename dynames
machattacDistance: Position → Position → Int
machattacDistance (Position x1 y1) (Position x2 y2) =
abs (x1 - x2) + abs (y1 - y2)

             — Функции для работы с ресурсаны
data Resource = Resource String Int deriving (Show, Eq)
      getResourceAmount :: Resource -> Int
getResourceAmount (Resource _ amount) = amount
      getResourceName :: Resource -> String
cetResourceName (Resource name ) = name
      calculateTotalResources :: [Resource] \rightarrow Int calculateTotalResourceResource = foldr (\text{tree total} \rightarrow total + getResourceResout res) 0
             getResourcesDyName :: String \Rightarrow [Resource] \Rightarrow [Resource] getResourcesDyName targetName = filter (\res \Rightarrow getResourceName res == targetName)
optimization and positive interpretation in inter (vice — dynamic pass positive in processor continues data Camedivent in Unitified Unit Positive Positive | Unitified Unit Positive Interpretation | United Units Positive Interpretation | United Un
      - Symmat gas anness organos coformil → [Caselesst] getterothy/pps targettenet filter (worst → case targettenet, went) of [Caselesst] potterothy/pps targettenet filter (worst → case targettenet, went) of [Caselesst] → Trus (Caselesstanet) → [Caselesstanet] → [Case
```

Лучшие практики функций высшего порядка

Что делать:

- Использовать map, filter, fold для работы со списками
- Применять композицию функций для создания цепочек преобразований
- Использовать частичное применение для создания специализированных функций
- Создавать читаемые лямбда-функции с понятными именами
- Комбинировать функции для сложных операций

X Чего избегать:

- Создавать слишком сложные композиции
- Использовать лямбда-функции там, где можно использовать именованные

Домашнее задание

Задача 1:

Реализовать функции для работы с игровыми картами

Задача 2:

Создать функции для анализа игрового состояния

Задача 3:

Реализовать функции для игровых действий

Что дальше?

На следующей лекции:

- Монады и Ю
- Работа с файлами
- Обработка ошибок
- Тестирование

Подготовка:

- Изучить главу 27-28 из учебника
- Выполнить домашнее задание
- Подготовить вопросы по текущей теме

Вопросы?

Контакты:

- Email: [ваш.email@university.edu]
- Telegram: [@username]
- Офис: [номер кабинета]

Следующая лекция: Монады и Ю