Язык программирования Haskell

Перменные ,Типы данных,Функции

Типы и значения

- Строгая статическая типизация
  - x :: Int
  - x = 5

-- Нельзя потом присвоить строку: x = "hello" вызовет ошибку

- Type inference (вывод типов)
  - y = 3.14 -- тип автоматически определится как Double
  - z = True -- тип Bool

- Базовые типы: Int, Integer, Float, Double
  - a :: Int ; a = 42
  - b :: Integer ; b = 12345678901234567890
  - c :: Float ; c = 3.14
  - d :: Double ; d = 2.718281828

Типы и значения

Bool, Char, String

```
flag :: Bool
flag = False

ch :: Char
ch = 'A'

str :: String
str = "Hello, Haskell!"
```

Списки [a], кортежи (a,b,…)

```
nums :: [Int]
nums = [1,2,3,4]

pair :: (Int, String)
pair = (7, "days")
```

Лексические соглашения

- Чувствителен к регистру
  - x = 5
  - X = 10 -- разные переменные
- Имена функций/переменных: строчные буквы
  - sumList :: [Int] -> Int
  - sumList xs = sum xs
- Комментарии: '--' однострочные, '{- -}' многострочные

Функции

- **Определение функции**
  - add :: Int -> Int -> Int
  - add x y = x + y
- **Функции первого класса** — можно передавать как аргументы

map (*2) [1,2,3,4] -- [2,4,6,8]

filter (>2) [1,2,3,4] -- [3,4]

foldl (+) 0 [1,2,3,4] -- (((0+1)+2)+3)+4 = 10

Foldl (*) 1 [1,2,3,4] -- (((1*1)*2)*3)*4 = 24

Каррирование

Каждая функция в Haskell принимает ровно один аргумент и возвращает новую функцию, если требуется больше.

```
add :: Int -> Int -> Int
add x y = x + y

inc = add 1
inc 5    -- 6
```

Особенности Haskell

- Чистота: нет побочных эффектов

  Функция всегда возвращает одно и то же значение при одинаковых аргументах, не изменяет внешние переменные.

Ленивые вычисления

Значения вычисляются только когда они действительно нужны. Это позволяет:

работать с бесконечными структурами;

экономить ресурсы, если часть результата не используется.

```
naturals = [1..]          -- бесконечный список
take 5 naturals           -- [1,2,3,4,5]
```

Примеры программ

- **Факториал:**
- fact :: Int -> Int
- fact n = if n == 0 then 1 else n * fact (n-1)
- **Фибоначчи:**
- fib 0 = 0
- fib 1 = 1
- fib n = fib (n-1) + fib (n-2)

Итог

- Haskell = функциональный язык
- Строгая типизация + вывод типов
- Ленивые вычисления
- Чистота и функции высшего порядка