60. Язык функционального программирования Haskell: типы, функции и задание определяющих уравнений для функций. Образцы и сопоставление с ними. Рекурсивная обработка списков на примере поиска элемента в списке.

Háskell (русск. *Ха́скель, Ха́скелл*) — стандартизованный чистый функциональный язык программирования общего назначения. Типизация в Хаскеле строгая, статическая, с автоматическим выводом типов. Поскольку язык функциональный, то основная управляющая структура — это функция. Серьёзное отношение к типизации — ещё одна отличительная черта Хаскеля.

Базовые типы

Integer – для работы с целыми числами
Float – для работы с дробными числами
Bool – для логических величин (True, False)

Функции

Наиболее важным типом данных в функциональном программировании являются функции. Мы можем применить функцию к каким-либо аргументам и вывести результаты. Говоря языком математики, функция f – это закон соответствия, который сопоставляет каждому элементу из данного множества A один единственный элемент из множества B. Тип элементов из множества A называется исходным, тип элементов из множества B – целевым типом функции. Сокращенно мы будем записывать эту информацию как f :: A -> B. Например, функция square, ассоциирующая с каждым целым числом его квадрат, имеет следующий тип: square :: Integer -> Integer. О функции f :: A -> B говорят, что она берет аргумент из A и возвращает результат из B.

Например:

```
square :: Integer -> Integer
square x = x * x   //Определяющее уравнение для функции
```

Если функция должна изменить аргумент тогда и только тогда, когда выполняется условие, необходимо применить специальную конструкцию языка, очень похожую на стандартный условный оператор в процедурных языках. Общий вид конструкции:
Функция аргумент = if условие then значение1 else значение2
Пример:

```
square :: Integer -> Integer
square x = if x > 1
           then x * x else x
```

Сопоставление с образцом
Параметры (аргументы) функции в ее определении, подобные переменным x и y в определении f x y = x * y, называются формальными параметрами функции. При запросах функции передаются фактические параметры. Например, в запросе
F 17 (1+6)
17 – фактический параметр, согласованный с x и (1+6) фактический параметр, согласованный с y. При запросе функции фактические параметры заменили собой формальные параметры из определения. Поэтому результат запроса: 17 * (1+6).
Фактические параметры – это выражения. Формальные параметры это названия переменных. В Haskell формальный параметр может также быть образцом.
Рассмотрим следующее функциональное определение, где образец используется как формальный параметр:
f[1, x, y] = x+ y
Такая функция применяется лишь к спискам, содержащим три элемента, причем первый элемент должен быть равен 1. Второй и третий элементы в нем могут быть произвольными. Но эта функция не определена для более коротких или длинных списков, или списков, у которых первый элемент не равен 1. Функция sqrt не определена для отрицательных чисел, оператор / (деления) не определен при нулевом втором параметре.

Рекурсивная обработка списков на примере поиска элемента в списке.
Список – набор элементов, относящихся к одному и тому же типу. Тип списка задают как тип его элементов в квадратных скобках. Элементы списка при записи заключаются в квадратные скобки и, перечисляя их, отделяют друг от друга запятой. Пустой список – список не содержащий элементов []. Первый элемент списка называется головой (head). Только пустой

список не имеет головы. Остальная часть списка называется хвостом (tail). В отличие от головы списка, его хвост тоже является списком.

Операции:

- -добавление элемента в начало списка (1 : [])
- -объединение списков ([1, 2, 3]++[2,3])
- -возврат головы (head [1, 2, 3])
- -возврат хвоста (tail [1, 2, 3])
- -длина списка (length [1, 2, 3])

```
member :: Integer -> [Integer] -> Bool
member _ [] = False
member x (h : t) = (x == h) || (member x t)
```