Основные возможности Haskell

Цели

- Приобрести навыки работы с интерпретатором языка Haskell. Получить представление об основных типах языка Haskell. Научиться определять простейшие функции.

Задание

1. Изучить теоретические сведения
2. Выполнить задания в соответствии с вариантом

Теоретические сведения

Типы

- Типы Integer и Int используется для представления целых чисел, причем значения типа Integer не ограничены по длине.
- Типы Float и Double используется для представления вещественных чисел.
- Тип Bool содержит два значения: True и False, и предназначен для представления результата логических выражений.
- Тип Char используется для представления символов.

Списки Чтобы задать список в Haskell, необходимо в квадратных скобках перечислить его элементы через запятую. Все эти элементы должны принадлежать одному и тому же типу. Тип списка с элементами, принадлежащими типу a, обозначается как [a]. Примеры: [1,2]; ['1','2','3'].

Операции со списками Оператор : (двоеточие) используется для добавления элемента в начало списка. Его левым аргументом должен быть элемент, а правым - список:

```
> 1:[2,3]
[1,2,3] :: [Integer]
> '5':['1','2','3','4','5']
['5','1','2','3','4','5'] :: [Char]
> False:[]
[False] :: [Bool]
```

Listing 1: Пример

С помощью оператора (:) и пустого списка можно построить любой список:

```
> 1:(2:(3:[]))
[1,2,3] :: Integer
>1:2:3:[]
[1,2,3] :: Integer
```

Listing 2: Пример

Элементами списка могут быть любые значения — числа, символы, кортежи, другие списки и т.д.

```
> [(1,'a'),(2,'b')]
[(1,'a'),(2,'b')] :: [(Integer,Char)]
> [[1,2],[3,4,5]]
[[1,2],[3,4,5]] :: [[Integer]]
```

Listing 3: Пример

Для работы со списками в языке Haskell существует большое количество функций. В данной лабораторной работе рассмотрим только некоторые из них.

- Функция head возвращает первый элемент списка.
- Функция tail возвращает список без первого элемента.
- Функция length возвращает длину списка.

Функции head и tail определены для непустых списков. При попытке применить их к пустому списку интерпретатор сообщает об ошибке. Примеры работы с указанными функциями:

```
> head [1,2,3]
1 :: Integer
> tail [1,2,3]
[2,3] :: [Integer]
> tail [1]
[] :: Integer
> length [1,2,3]
3 :: Int
```

Listing 4: Пример

Для соединения (конкатенации) списков в Haskell определен оператор ++.

```
> [1,2]++[3,4]
[1,2,3,4] :: Integer
```

Listing 5: Пример

Функции Рассмотрим пример:

```
square :: Integer -> Integer
square x = x * x
```

Listing 6: Пример

Первая строка (square :: Integer -> Integer) объявляет, что мы определяем функцию square, принимающую параметр типа Integer и возвращающую результат типа Integer. Вторая строка (square x = x * x) является непосредственно определением функции. Функция square принимает один аргумент и возвращает его квадрат.

В общем виде тип функции, принимающей n аргументов, принадлежащих типам t1, t2, . . . , tn, и возвращающей результат типа a, записывается в виде t1->t2->...->tn->a.

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

Listing 7: Пример

Условное выражение В общем виде выглядит так:
if условие then выражение else выражение.

Функцию signum, вычисляющую знак переданного ей аргумента:

```
signum :: Integer -> Integer
signum x = if x > 0
        then 1
        else if x < 0
                then -1
                else 0
```

Listing 8: Пример

Следует обратить внимание на отступы. Именно по отступам компилятор определяет к какому if'у относится тот или иной else.

Условие в определении условного оператора представляет собой любое выражение типа Bool. Примером таких выражений могут служить сравнения. При сравнении можно использовать следующие операторы:

- <, >, <=, >= - эти операторы имеют такой же смысл, как и в языке Си (меньше, больше, меньше или равно, больше или равно);
- == - оператор проверки на равенство;
- /= - оператор проверки на неравенство.

Пример

Задание Написать функцию на языке Haskell возвращающую знак переданного целого числа.

```
signum :: Integer -> Integer
signum x = if x > 0
        then 1
        else if x < 0
                then -1
                else 0

main = print $ Prelude.signum 1
```

Listing 9: Пример

Состав отчета

- Титульный лист (фамилия, группа, номер варианта, наименование работы, задание)
- Текст программы на языке Haskell
- Результат работы программы на языке Haskell

Варианты заданий

1. Функция max3, по трем целым возвращающая наибольшее из них.
2. Функция min3, по трем целым возвращающая наименьшее из них.
3. Функция sort2, по двум целым возвращающая пару, в которой наименьшее из них стоит на первом месте, а наибольшее - на втором.
4. Функция bothTrue :: Bool -> Bool -> Bool, которая возвращает True тогда и только тогда, когда оба ее аргумента будут равны True. Не используйте при определении функции стандартные логический операции (&& , || и т.п.).

5. Функция solve2::Double->Double->(Bool,Double), которая по двум числам, представляющим собой коэффициенты линейного уравнения ax + b = 0, возвращает пару, первый элемент которой равен True, если решение существует и False в противном случае; при этом второй элемент равен либо значению корня, либо 0.0.

6. Функция isParallel, возвращающая True, если два отрезка, концы которых задаются в аргументах функции, параллельны (или лежат на одной прямой). Например, значение выражения isParallel (1,1) (2,2) (2,0) (4,2) должно быть равно True, поскольку отрезки (1, 1) - (2, 2) и (2, 0) - (4, 2) параллельны.

7. Функция isIncluded, аргументами которой служат параметры двух окружностей на плоскости (координаты центров и радиусы). Функция возвращает True, если вторая окружность целиком содержится внутри первой.

8. Функция isRectangular, принимающая в качестве параметров координаты трех точек на плоскости, и возвращающая True, если образуемый ими треугольник - прямоугольный.

9. Функция isTriangle, определяющая, можно ли их отрезков с заданными длинами x, y и z построить треугольник.

10. Функция isSorted, принимающая на вход три числа и возвращающая True, если они упорядочены по возрастанию или по убыванию.

11. Функция принимает два числа, и если их сумма чётна, то возвращает их разницу, иначе - сумму.

12. Функция принимает список и число, и добавляет число к списку.

13. Функция принимает список и число, и добавляет число к списку с конца.

14. Функция принимает 3 числа и возвращает список с этими числами

15. Функциця принимает 2 числа, и возвращает наиболшее кратное двойке.