ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«ПРОГРАММИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ФУНКЦИЙ»

Цель лабораторной работы: ознакомиться с синтаксисом функций в Haskell. Выполнить индивидуальные задания и составить отчёт со скриншотами программы.

Ход работы:

1. Функция max3, по трем целым возвращающая наибольшее из них.

```
max3 a b c = maximum [a,b,c]
```

2. Функция min3, по трем целым возвращающая наименьшее из них.

```
min3 a b c = minimum [a,b,c]
```

3. Функция sort2, по двум целым возвращающая пару, в которой наименьшее из них стоит на первом месте, а наибольшее – на втором.

```
sort2 a b
    | a > b = (a,b)
    | otherwise = (b,a)
```

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 логические операции (&&, || и т.д.).

```
bothTrue :: Bool -> Bool -> Bool
bothTrue True True = True
bothTrue _    _    = False
```

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.

```
solve2 a b
    | a == 0 = (False, 0.0)
    | otherwise = (True, -b / a)
```

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.

```
isParallel (x1,y1) (x2,y2) (a1,b1) (a2,b2)
    | (a2-a1)*(y2-y1) - (b2-b1)*(x2-x1) == 0 = True
    | otherwise = False
```

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.

```
isIncluded (x1,y1,r1) (x2,y2,r2)
    | sqrt ( (x1-x2)^2 + (y1-y2)^2 ) <= r1-r2 = True
    | otherwise = False
```

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

```
isRectangular (x1,y1) (x2,y2) (x3,y3)
    | (x2-x1)*(x3-x2) + (y2-y1)*(y3-y2) == 0 = True
    | (x2-x1)*(x3-x1) + (y2-y1)*(y3-y1) == 0 = True
    | (x2-x3)*(x1-x3) + (y2-y3)*(y1-y3) == 0 = True
    | otherwise = False
```

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

```
isTriangle x y z
    | (x + y > z) && (z + y > x) && (x + z > y) = True
    | otherwise = False
```

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

```
import Data.List
isSorted a b c
    | [a,b,c] == ( sort [a,b,c] ) = True
    | otherwise = False
```

Пример работы:

```
*Main> max3 1 2 3
3
*Main> min3 1 2 3
1
*Main> sort2 4 (-2)
(-2,4)
*Main> bothTrue True True
True
*Main> solve2 0 1
(False,0.0)
*Main> solve2 4 (-1)
(True,0.25)
```

```
*Main> isParallel (1,1) (2,2) (2,0) (4,2)
True
*Main> isParallel (0,0) (4,0) (0,2) (4,2)
True
*Main> isParallel (1,0) (0,3) (3,5) (1,7)
False
*Main> isIncluded (0,0,5) (0,0,4)
True
*Main> isIncluded (0,0,5) (0,0,10)
False
*Main> isIncluded (0,0,5) (2,2,1)
True
```

```
*Main> isRectangular (0,0) (0,6) (10,0)
True
*Main> isRectangular (1,2) (0,5) (3,0)
False
*Main> isTriangle 1 2 3
False
*Main> isTriangle 5 5 5
True
*Main> isTriangle 6 8 10
True
*Main> isSorted 1 2 3
True
*Main> isSorted 3 2 1
False
```

Вывод: ознакомился с синтаксисом функций в Haskell. Выполнил индивидуальные задания и составил отчёт со скриншотами программы.