ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Цель лабораторной работы: Приобрести навыки работы с интерпретатором языка Haskell. Получить представление об основных типах языка Haskell. Научиться определять простейшие функции. Выполнить индивидуальные задания и составить отчёт со скриншотами программы.

Ход работы:

1.Приведите пример нетривиальных выражений, принадлежащих следующему типу:
1) ((Char,Integer), String, [Double])
(('A',15),"fp",[23.0,11.0])
2) [(Double,Bool,(String,Integer))]
[(4.0,True,("I learn language",7)), (5.0,False,("haskell",8))]
3) ([Integer],[Double],[(Bool,Char)])
([1..100],[1.0..10.0],[(False, 'H')])
4) [[[(Integer,Bool)]]]
[[[(2, False)]]]
5) (((Char,Char),Char),[String])
((('a','b'),'c'),["abc","def"])
6) (([Double],[Bool]),[Integer])
(([1.0,2.0],[True,False]),[7,14])
7) [Integer, (Integer,[Bool])]
список не может содержать элементы разных типов, для этого используются кортежи, н-р:
(Integer, (Integer, [Bool]))
(1,(2,[True,True]))
8) (Bool,([Bool],[Integer]))
(True,([True,False],[1,2,3,4,5]))
9) [([Bool],[Double])]
[([True,False,False],[15.0,16.0,17.0])]
10) [([Integer],[Char])]
[([6,7,8,9,10,11],"abcdefghijklmnopqrstuvwxyz")]

2. Дано 3 числа. Определить максимальное из них.
```
max' a b c = max (max a b) c
```

3. Дано 3 числа. Определить минимальное из них.
```
min' a b c = minimum [a,b,c]
```

4. Даны числа 23, 5, 43, 17. Вычислить их среднее арифметическое.
```
avg = sum [23,5,43,17] / 4
```

5. Проверьте на упорядоченность последовательности чисел, используя знаки логических операций:
   а) 1 2 33 4 5
   б) 1 2 3 4 5
   в) 5 4 3 2 0
   г) 5 4 33 2 0
```
checkSeq = do
        let list = [ [1,2,33,4,5], [1,2,3,4,5], [5,4,3,2,0], [5,4,33,2,0] ]
        [checkSeqSort a b c d e | [a,b,c,d,e] <- list]

checkSeqSort a b c d e
        | a > b && b > c && c > d && d > e = "Decreasing sequence"
        | a < b && b < c && c < d && d < e = "Increasing sequence"
        | otherwise = "undefined sequence"
```

6. Вычислите абсолютную величину отрицательного и положительного числа.
bounds = (minBound :: Int, maxBound :: Int)

7. Проверте на четность и на нечетность целые числа 35, 30.
checkNumbers = [(35, checkNumber 35), (30, checkNumber 30)]
checkNumber number = if odd number then "odd" else "even"

8. Выберите наибольший общий делитель для чисел 12, 24, 36.
gcd' = gcd (gcd 12 24) 36

9. Выберите наименьшее общее кратное для чисел 124, 56.
lcm' = lcm 124 56

10. Создать выражение, проверяющее, кортеж следующего вида:
((Integer,Integer), (Integer,Integer)) на выполнение условия: первый аргумент первого кортежа строго больше второго аргумента первого кортежа, и первый аргумент второго кортежа меньше либо равен второго аргумента второго кортежа.

```
checkTuple :: ((Integer, Integer), (Integer, Integer)) -> Bool
checkTuple ( (a,b), (c,d) )
        | a > b && c <= d = True
        | otherwise = False

sampleCheckTuple = do
        let list = [((2,1),(4,4)), ((2,1),(3,4)), ((2,1),(5,4)), ((1,2),(3,4))]
        [checkTuple tuple | tuple <- list]
```

11. По трём введённым числовым значениям на экран выводится строка, представляющая собой квадратное уравнение $a*x^2 + b*x + c = 0$, где коэффициенты a, b и c являются введёнными числами в соответствующем порядке.

```
quadraticEquation a b c =
        show a ++ "*x^2 + " ++ show b ++ "*x + " ++ show c ++ " = 0"
```

12. Дано три числа x, y, z, обозначающие длины отрезков. Определить, можно ли построить треугольник из этих отрезков.

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

13. Дано три числа x, y, z, обозначающие длины отрезков. Определить, является ли треугольник со сторонами x, y, z прямоугольным.

```
checkRightTriangle x y z
        | checkTriangle x y z && (x^2 + y^2 == z^2) = True
        | checkTriangle x y z && (z^2 + y^2 == x^2) = True
        | checkTriangle x y z && (x^2 + z^2 == y^2) = True
        | otherwise = False
```

14. Дано 5 чисел. Определить, отсортированы ли они в порядке убывания.

```
checkSort a b c d e
        | a > b && b > c && c > d && d > e = True
        | otherwise = False
```

15. Дано трёхзначное число. Заменить в нем число сотен на число десятков, число десятков на число единиц, а число единиц на число сотен.

```
switch number = do
        let
                a = number `div` 100
                b = (number `div` 10) `rem` 10
                c = number `rem` 10
        a*10 + b + c*100
```

Пример работы:

```
*Main> max' 5 (-2) 7
7
*Main> min' 4 1 2
1
*Main> avg
22.0
*Main> checkSeq
["undefined sequence","Increasing sequence","Decreasing sequence","undefined seq
uence"]
*Main> bounds
(-9223372036854775808,9223372036854775807)
*Main> gcd'
12
*Main> lcm'
1736
*Main> sampleCheckTuple
[True,True,False,False]
*Main> quadraticEquation 3 5 1
"3*x^2 + 5*x + 1 = 0"
```

```
*Main> checkTriangle 1 2 3
False
*Main> checkTriangle 5 5 5
True
*Main> checkTriangle 6 8 10
True
*Main> checkRightTriangle 5 5 5
False
*Main> checkRightTriangle 5 5 25
False
*Main> checkRightTriangle 6 8 10
True
*Main> checkSort 1 2 3 4 5
False
*Main> checkSort 5 4 3 2 1
True
*Main> switch 123
312
*Main> switch 937
793
```

Вывод: приобрел навыки работы с интерпретатором языка Haskell. Получил представление об основных типах языка Haskell. Научился определять простейшие функции. Выполнил индивидуальные задания и составил отчёт со скриншотами программы.