**Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования**

**Московский технический университет связи и информатики**

**КАФЕДРА ИНФОРМАТИКИ**

**Лабораторная работа по теме 6  
на языке Haskell**

**Выполнил:**

**Студент группы БВТ1701**

**Шабанов Борис**

**Вариант 22**

**Москва 2019**

**Цель работы:** закрепить материал по пройденным темам, повторить классы, типы, рекурсивные функции, моды, а также добавочные элементы let и where, чтение и запись файлов.

**1 Формулировка задания на лабораторную работу**

Текстовый файл содержит натуральные числа. В другой файл поместить только те числа, корень из которых - натуральное число.

**2 Техническое задание**

-Входные параметры: текстовый файл, записывающийся в переменную file1

-Выходные параметры: консольное окно и текстовый файл, записывающийся в переменную file2

**3 Функции программы**

- fun1 :: Int -> [String] -> [Double]

Функция, меняющая тип элементов списка со строкового до переменной с плавающей точкой.

- fun2 :: (Num i, Ord i) => i -> [Double] -> [Int]

Функция, определяющая удовлетворяют ли элементы условию задачи: добавляются в список только натуральные переменные.

- fun3 :: (Num i, Ord i) => i -> [Int] -> [String]

Функция, меняющая тип элементов списка с плавающей точкой до строкового.

- main = do

Основная функция, отвечающая за основной (общий) алгоритм программы, вызывающая все выше перечисленные функции.

1. **Код программы**

**4.1 Текстовый файл**

На рисунке 1 изображен текст №1 с входными параметрами, файл №2 – пустой.

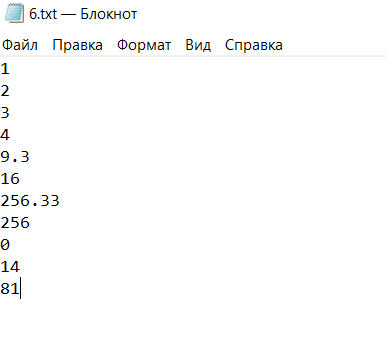


Рисунок 1 - Текстовый файл.

**4.2 Haskell**

module Main where

import System.IO

import System.Environment (getArgs)

import Control.Monad

fun1 :: Int -> [String] -> [Double]

fun1 n \_

| n <= 0 = []

fun1 \_ [] = []

fun1 n (x:xs) = a : fun1 (n - 1) xs

where a = read x + 0.0

fun2 :: (Num i, Ord i) => i -> [Double] -> [Int]

fun2 n \_

| n <= 0 = []

fun2 \_ [] = []

fun2 n (x:xs) = if (d > 0) && (x > 0) then fun2 (n - 1) xs else truncate x : fun2 (n - 1) xs

where d = x - fromIntegral (truncate x)

fun3 :: (Num i, Ord i) => i -> [Int] -> [String]

fun3 n \_

| n <= 0 = []

fun3 \_ [] = []

fun3 n (x:xs) = a : fun3 (n - 1) xs

where a = show x

main = do

[file1, file2] <- getArgs

text1 <- readFile file1

let lines1 = lines text1

l2 = fun1 (length lines1) lines1

l3 = map (sqrt) l2

l4 = fun2 (length l3) l3

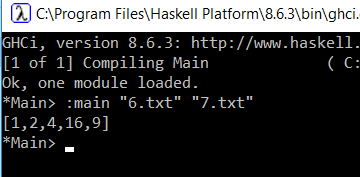
l5 = fun3 (length l4) l4

writeFile file2 (show l5)

print(l4)

1. **Результат программы**

На рисунке 2 изображен итог текста программы в приложении и в файле.



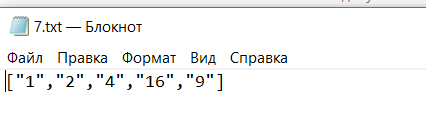


Рисунок 2 – Итог программы.