Министерство науки и высшего образования Российской Федерации

Новосибирский Государственный технический университет

Кафедра автоматизированных систем управления



**Отчет по лабораторным работам 1-2**

**по дисциплине «Функциональное программирование»**

**Вариант-1**

Выполнили

студенты группы АВТ-812:

Антонянц Егор

Амельченко Артем

Преподаватель:

Достовалов Дмитрий Николаевич,

к.т.н., доцент кафедры АСУ

г. Новосибирск

2021 г.

Содержание

[1 Цель работы 3](#_Toc82789703)

[2 Текст заданий 3](#_Toc82789704)

[2.1 Текст заданий к лабораторной работе 1 3](#_Toc82789705)

[2.2 Текст заданий к лабораторной работе 2 3](#_Toc82789706)

[3 Тексты программ 4](#_Toc82789707)

[4 Результаты тестов 6](#_Toc82789708)

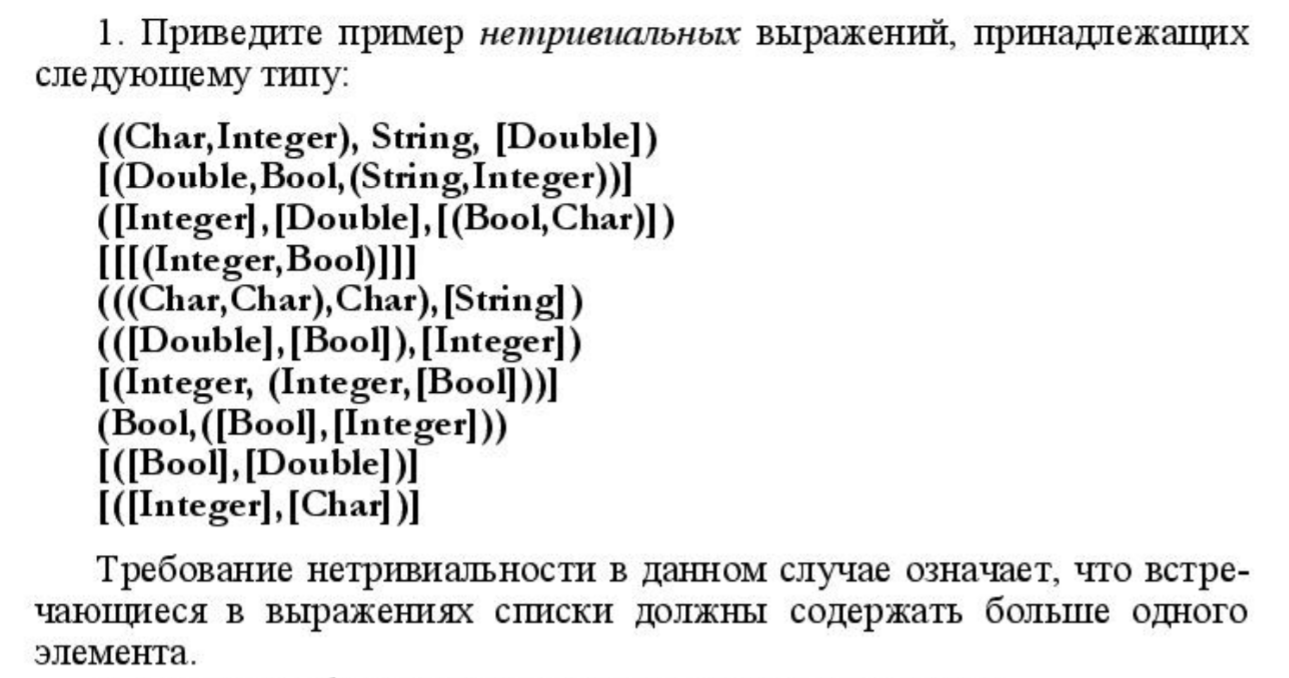
[5 Выводы 7](#_Toc82789709)

# 1 Цель работы

Приобрести навыки работы с интерпретатором Haskell. Получить представление об основных типах языка Haskell. Научиться определять простейшие функции. Научиться определять рекурсивные функции. Получить представление о механизме сопоставления с образцом. Приобрести навыки определения функций для обработки списков.

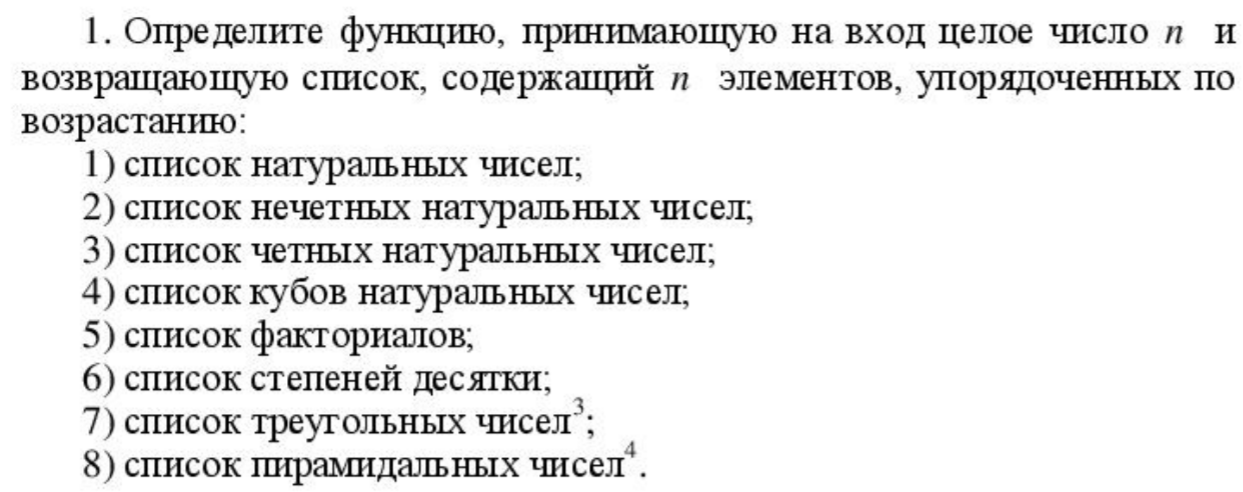
# 2 Текст заданий

## 2.1 Текст заданий к лабораторной работе 1



2. Напишите функцию, реализующую задачу: Три точки А, В, С лежат на одной прямой. Даны длины отрезков АВ, ВС, АС. Может ли точка А лежать между В и С?

## 2.**2 Текст заданий к лабораторной работе 2**



2. Определить следующие функции: функцию, вычисляющую сумму квадратов чисел от 1 до N; функцию f типа Char -> String -> String, которая принимает на вход строку и символ и возвращает строку, в которой удалены все вхождения символа. Пример: f ‘l’ “Hello world!” должно возвращать “Heo word!”

# 3 Тексты программ

Примеры нетривиальных выражений к заданию 1.1 приведен на рисунке 1.

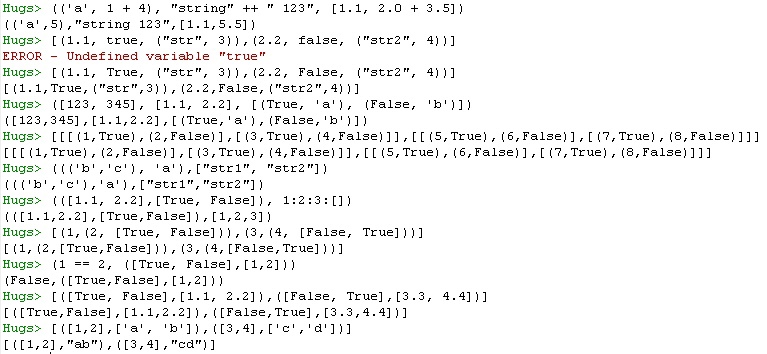


Рисунок 1 – Примеры нетривиальных выражений

Текст программы к заданию 1.2:

checkPoints :: Double -> Double -> Double -> Bool

checkPoints ab bc ac = ab + ac == bc

Текст программы к заданию 2.1:

getListTask1 :: Integer -> [Integer]

getListTask1 n = [i|i<-[0..n-1]]

getListTask2 :: Integer -> [Integer]

getListTask2 n = [2\*i+1|i<-[0..n-1]]

getListTask3 :: Integer -> [Integer]

getListTask3 n = [2\*i|i<-[0..n-1]]

getListTask4 :: Integer -> [Integer]

getListTask4 n = [i^3|i<-[0..n-1]]

getListTask5 :: Integer -> [Integer]

getListTask5 n = [factorial i|i<-[0..n-1]]

getListTask6 :: Integer -> [Integer]

getListTask6 n = [10^i|i<-[0..n-1]]

getListTask7 :: Integer -> [Integer]

treug 1 = 1

treug n = n + treug(n - 1)

getListTask7 0 = []

getListTask7 n = getListTask7(n - 1) ++ [treug(n)]

getListTask8 :: Integer -> [Integer]

piram 1 = 1

piram n = treug n + piram(n - 1)

getListTask8 0 = []

getListTask8 n = getListTask8(n - 1) ++ [piram n]

factorial :: Integer -> Integer

factorial 0 = 1

factorial n = n \* factorial(n - 1)

Текст программы к заданию 2.2:

f :: Char -> String -> String

f c [] = []

f c s = if c == head s

then f c (tail s)

else head s : f c (tail s)

sumSquares :: Integer -> Integer

sumSquares 0 = 0

sumSquares n = n \* n + sumSquares(n - 1)

# 4 Результаты тестов

Примеры тестов к заданиям 1.2, 2.1 и 2.2 представлены на рисунках 2-4.



Рисунок 2 – Тест к заданию 1.2

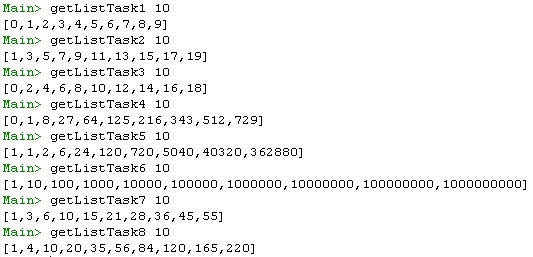


Рисунок 3 – Тест к заданию 2.1

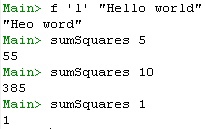


Рисунок 4 – Тест к заданию 2.2

# 5 Выводы

Были приобретены навыки работы с интерпретатором Haskell – HUGS. Получены представления об основных типах языка Haskell. Было получено представление о механизме сопоставления с образцом и рекурсивных функциях на языке Haskell. Приобретены навыки определения функций для обработки списков.