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

**Московский технический университет связи и информатики**

**КАФЕДРА ИНФОРМАТИКИ**

**Лабораторная работа по теме 5  
Игра «Быки и коровы» Haskell**

**Выполнил:**

**Студент группы БВТ1701**

**Шабанов Борис**

**Вариант 22**

**Москва 2019**

**Цель работы:** написать игру на Haskell «Быки и Коровы»

**1 Формулировка задания на лабораторную работу**

В данной работе необходимо разработать программу для игры «Быки и коровы». Правила игры: один игрок выбирает четырехзначный код, каждая цифра должна быть от 1 до 9 (например, [5, 2, 2, 1]). Другой игрок должен получить код с помощью нескольких догадок такого же вида (например, [2, 2, 3, 2]). Первый игрок сообщает количество быков и коров для каждой догадки. Бык — это правильная цифра на правильном месте. Корова — это правильная цифра на неправильном месте. Одна и та же цифра может быть посчитана только один раз — как бык или как корова. Зная эту информацию второй игрок пытается получить код за минимальное число попыток.

Определите функцию bulls :: [Int] -> [Int] -> Int, которая по секретному коду и догадке возвращает количество быков:

bulls [1, 2, 3, 4] [1, 1, 2, 2] == 1

bulls [1, 2, 3, 4] [2, 1, 2, 1] == 0

bulls [1, 1, 2, 2] [1, 2, 1, 2] == 2

Определите функцию cows :: [Int] -> [Int] -> Int, которая по секретному коду и догадке возвращает количество коров:

cows [4, 3, 2, 1] [1, 1, 1, 1] == 0

cows [4, 3, 2, 1] [1, 2, 2, 2] == 1

cows [1, 2, 2, 1] [1, 2, 1, 2] == 2

Возможно будет удобно определить вспомогательную функцию, которая вычисляет количество правильных цифр вообще (т.е. быков и коров вместе), а затем вычесть из этого числа количество быков. Помните, что каждая цифра в коде учитывается только один раз.

Определите game :: IO (), которая играет в качестве первого игрока (загадывает код) против человека, который отгадывает код. Напишите интерактивную программу, которая выдает догадки, считывает счет, сообщает код, когда он становится ясным. Пример диалога, если человек загадал [4, 3, 2, 1]

Main>*game*

Предположение: [1, 1, 1, 1]. Каков счет? *(1, 0)*

Предположение: [1, 2, 2, 2]. Каков счет? *(1, 1)*

Предположение: [3, 1, 2, 3]. Каков счет? *(1, 2)*

Предположение: [3, 2, 1, 4]. Каков счет? *(0, 4)*

Предположение: [4, 1, 3, 2]. Каков счет? *(1, 3)*

Секретный код: [4, 3, 2, 1].

Для алгоритма можно использовать следующую простую стратегию: создайте список всех возможных кодов, предложите первый из них, после получения счета удалите из списка все варианты, которые не соответствуют счету. Продолжайте итерации до тех пор, пока в списке не останется один код, который можно вывести в качестве угаданного.

**2 Техническое задание**

-Входные параметры: n1

-Выходные параметры: putStrLn $ "Cows : " ++ show ((bulls\_and\_cows n1 n2) - (bulls n1 n2)) , putStrLn $ "Bulls : " ++ show (bulls n1 n2) , if (bulls n1 n2 == 4) then putStrLn "You won!" else (putStrLn "Try away!" >> game n1)

**3 Функции программы**

-bulls\_and\_cows :: Int -> Int -> Int

«Быки и коровы» присчитывания всех правильных ответов для нахождения «Коров»

-bulls :: Int -> Int -> Int

«Быки» нахождения правильных ответов, находящихся на правильном месте

-game :: Int -> IO ()

Основная функция, которая осуществляет связку всех функций, принимает ввод 4 элемент от игрока и вывод сообщение пользователю, выиграл он или нет.

-result :: Int -> Int -> IO ()

Высчитывает быков и коров и выводит их в консоль

-main = do

Функция, которая вызывает основную функцию game

1. **Код программы**

import System.IO (hSetBuffering,stdout,BufferMode(NoBuffering))

import System.Random (randomRIO)

import Control.Monad (when)

bulls\_and\_cows :: Int -> Int -> Int

bulls\_and\_cows n1 n2 = length $ filter (`elem` s2) s1

where s1=show n1

s2=show n2

bulls :: Int -> Int -> Int

bulls n1 n2 = length $ filter (uncurry (==)) z

where z=zip (show n1) (show n2)

game :: Int -> IO ()

game n1 = do

putStr "Enter 4-Number : "

n2 <- readLn

result n1 n2

if (bulls n1 n2 == 4) then putStrLn "You won!" else (putStrLn "Try away!" >> game n1)

result :: Int -> Int -> IO ()

result n1 n2 = do

putStrLn $ "Cows : " ++ show ((bulls\_and\_cows n1 n2) - (bulls n1 n2))

putStrLn $ "Bulls : " ++ show (bulls n1 n2)

main = do

hSetBuffering stdout NoBuffering

n1 <- randomRIO(1111, 9999)

print n1

game n1

1. **Результат программы**

На рисунке 2 изображен итог текста программы – игра между ноутом и человеком.

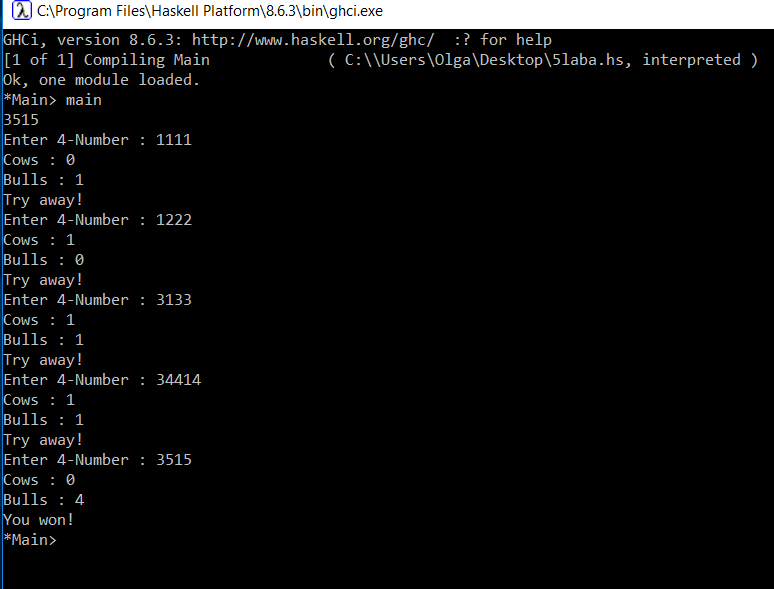


Рисунок 1 - Игра.