МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Лабораторная работа №3**

по дисциплине: **«Функциональное программирование»**

на тему: «**Работа в интерпретатора Haskell**»

ВЫПОЛНИЛ студент группы 16 ИТ-3

Яблонский А.С.

ПРОВЕРИЛ преподаватель

Попкова Д.В.

Полоцк, 2018 г.

**Вариант №8**

**Условие**

В электронной записной книжке хранятся записи следующих видов: напоминания о днях рождения знакомых, телефоны знакомых и назначенные встречи. Напоминание состоит из имени знакомого и даты (день и месяц). Запись о телефоне должна содержать имя человека и его телефон. Информация о назначенной встрече содержит дату встречи (день, месяц, год) и краткое описание (можно представлять строкой). Разработайте тип данных, представляющий такую запись. Записная книжка является списком записей.

**Структуры данных:**

Для хранения записей данного варианты были реализованы следующие структуры данных:

data Date =

Dm {

d :: Int,

m :: Int

}

| Dmy {

d :: Int,

m :: Int,

y :: Int

}

deriving (Eq, Show)

data Record =

-- Date of birth

Dob {

t :: Char,

name :: String,

date :: Date

}

| Contact {

t :: Char,

name :: String,

number :: Int

}

| Event {

t :: Char,

name :: String,

date :: Date

}

deriving (Eq, Show)

Структура Record – родительский тип данных для хранения в списке.

Dob - напоминания о днях рождения знакомых.

Contact - телефоны знакомых.

Event - назначенные встречи.

Во всех структурах поле t::Char хранит тип структуры для последующего сравнения.

Date – структуры для храения даты. Имеет два конструктора Dm – для даты без указания года. Dmy – для даты с указанием года.

Для упрощения тестирования всех функция было принято решение создать статический список записей разных типов:

-- Спискок записей

myList = [

dContact "Vano" 123,

dContact "Penka" 223,

dDob "Vano" Dm { d = 18, m = 9 },

dEvent "Vano" Dmy { d = 1, m = 12, y = 2016 }

]

**Функция для определения №1:**

Функция getByName, возвращающая информацию о человеке с указанным именем (его телефон и дату рождения).

**Решение:**

getByName :: String -> [Record]

getByName name = searchByName name myList

{-

Поиск записией по имени

Параметры:

name :: String - Имя для которого ищим записи

records :: [Record] - Список записей

Возвращает:

Список записей с именем name

Если таких нет - пустой список

-}

searchByName :: String -> [Record] -> [Record]

searchByName name records = \_searchByName name records []

\_searchByName \_ [] resultList = resultList

\_searchByName name (firstRecord:restRecords) resultList =

if getName firstRecord == name

then \_searchByName name restRecords (firstRecord:resultList)

else \_searchByName name restRecords resultList

**Тестирование:**

В результате тестирования функции ошибок выявлено не было. Функция отработала, как и ожидалось.

Код вызова функции:

> getByName "Vano"

**Функция для определения №2:**

Функция getByLetter, возвращающая список людей, о которых есть информация в записной книжке и чье имя начинается на указанную букву

**Решение:**

getByLetter :: Char -> [Record]

getByLetter letter = searchByLetter letter myList

{-

Поиск записей по первому символу их имени

Параметры:

letter :: Char - первый символ имени для поиска

records :: [Record] - Список записей

Возвращает:

Массив записей, имя у которых начинается с letter (без учета регистра)

Если таких нет - пустой список

-}

searchByLetter :: Char -> [Record] -> [Record]

searchByLetter letter records = \_searchByLetter letter records []

\_searchByLetter \_ [] resultList = resultList

\_searchByLetter letter (firstRecord:restRecords) resultList =

let firstLetter = head (getName firstRecord)

in if toUpper firstLetter == toUpper letter

then \_searchByLetter letter restRecords (firstRecord:resultList)

else \_searchByLetter letter restRecords resultList

**Тестирование:**

В результате тестирования функции ошибок выявлено не было. Функция отработала, как и ожидалось.

Код вызова функции:

> getByLetter 'V'

**Функция для определения №3:**

Функция getAssignment, возвращающая по указанной дате список дел (информация о назначенных встречах и телефоны друзей, которых нужно поздравить в этот день). **Решение:**

getAssignment :: Date -> [Record]

getAssignment date = searchByDate date myList

{-

Поиск записей по по дате (только по )

Параметры:

date :: Date - дата для поиска

records :: [Record] - Список записей

Возвращает:

Массив записей(Dob и Event только), дата которых совпадает с Date

Если таких нет - пустой список

-}

searchByDate :: Date -> [Record] -> [Record]

searchByDate date records = \_searchByDate date records []

\_searchByDate \_ [] resultList = resultList

\_searchByDate date (firstRecord:restRecords) resultList =

if getType firstRecord /= contactType && getDate firstRecord == date

then \_searchByDate date restRecords (firstRecord:resultList)

else \_searchByDate date restRecords resultList

**Тестирование:**

В результате тестирования функции ошибок выявлено не было. Функция отработала, как и ожидалось.

Код вызова функции:

> getAssignment Dm {d = 3, m = 4}

или

> getAssignment Dm {d = 18, m = 9}

**Вывод:**

В ходе выполнения данной лабораторной я приобрел навыки создания и работы с типами данных на языке Haskell. Научился описывать собственные конструкторы для типов данных, работать с let-in связыванием и выводить собственные ошибки при выполнении функций используя error.