**Федеральное агентство связи**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования «Московского технического университета связей и информатики»**

Кафедра Математической кибернетики и информационных технологий

Лабораторная работа №3

Знакомство с эффектами функций и создание программы на Haskell

Выполнил:

Студент группы БВТ1702

Рыбаков Д. В.

Вариант 26

**Формулировка задания:** разобрать различные эффекты языка Haskell и написать функцию, которая выполняет данную задачу: Описать функцию, которая для заданного списка проверяет, является ли он отсортированным по возрастанию.

**ТЗ:**

1. Написать функцию, которая принимает один аргумент – список – и возвращает boolean значение
2. Изучить и описать стандартные эффекты функций
3. **Код функции**

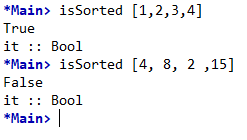
isSorted :: (Ord a) => [a] -> Bool

isSorted [] = True

isSorted [x] = True

isSorted (x:y:xs) = x <= y && isSorted (y:xs)

1. **Контрольное тестирование**



1. **Эффекты**
   1. **Эффект частичности**

Тип Maybe

data Maybe a = Just a | Nothing

Типы, которые могут вести себя как контейнеры по отношению

к другим типам, могут быть функторами. тип Maybe. Он может быть «пустой коробкой», и в этом случае имеет значение Nothing, или же в нём хранится какое-то одно значение, например, "ХА-ХА", и тогда он равен Just "ХА-ХА".

Пример:

f::Int -> Maybe Int

f 0 = Nothing

f x = Just x

Тип Either

data Either a b = Left a | Right b

На практике также применяется типы Either и Except, реализующие тот же эффект, но позволяющий добавить информацию о том, почему вычисление не может быть завершено

Пример:

data MyError = EmptyList

headE :: [a] -> Either MyError a

headE [] = Left EmptyList

headE (x:\_) = Right x

* 1. **Эффекты недетерминированности (неопределённости)**
     1. **Эффект нестабильности**

Если процедура для одного и того же значения аргумента может вернуть от раза к разу разные результаты, это значит, что на самом деле результат зависит от чего-то ещё.

Даже генератор случайных чисел (настоящий, аппаратный) — это «чистая» функция, зависящая от состояния источника энтропии.

Чтобы представить этот эффект чистой функцией, надо всего лишь неявную зависимость сделать явной.

type Reader r b = r -> b

p :: a -> Reader r b

* + 1. **Эффект множественности**

Здесь всё просто и очевидно. Функция, дающая много ответов сразу — это функция, имеющая единственный ответ-множество.

В Хаскеле есть тип Set для множеств, но для моделирования эффекта множественности оказывается более удобным список — [].

rollADie :: Int -> [Int]

rollADie n = [1..n]

* 1. **Побочный эффект**

Побочный эффект — это просто неявный результат. Сделаем же неявное явным!

p :: a -> (b, s)

newtype Putter s b = Putter (b, s)

p :: a -> Putter s b

* 1. **Эффект состояния**

Если соединить результат побочного эффекта и источник нестабильности, из их комбинации (композиции) получается эффект состояния — процедура, которая может и зависеть от текущего состояния «переменной», и задавать ей новое состояние.

Проведя рассуждения, аналогичные случаям Reader и Putter, получим

p :: a -> s -> (b, s)

newtype State s b = State (s -> (b, s))

p :: a -> State s b

* 1. **Отсутствие эффектов**

Рассмотрим тип

newtype Identity a = Identity a

Тип Identity a полностью аналогичен типу a. То есть это своего рода функция id, только на уровне типов.

Тип Identity не может выражать никаких эффектов. С другой стороны, можно сказать, что он выражает отсутствие эффектов.

Конечно же, конструктор типа Identity принадлежит Functor, Applicative, Monad и многим другим интересным и полезным классам.

* 1. **Все эффекты**

В Хаскеле есть специальный тип IO, реализующий сразу все возможные эффекты. В нём можно прерывать программу, обмениваться данными с ресурсами, не указанными явно в аргументах и возвращаемом значении. IO принадлежит Functor, Applicative, Monad и многим другим интересным и полезным классам.