Министерство науки и высшего образования Российской Федерации

Новосибирский Государственный технический университет

Кафедра автоматизированных систем управления



**Отчет по лабораторной работе 4**

**по дисциплине «Функциональное программирование»**

**Вариант 3**

Выполнил

студент группы АВТ-812:

Березин Дмитрий

Преподаватель:

Достовалов Дмитрий Николаевич,

к.т.н., доцент кафедры АСУ

г. Новосибирск

2021 г.

Содержание

[1 Цель работы 3](#_Toc84887484)

[2 Текст задания 3](#_Toc84887485)

[3 Текст программы 3](#_Toc84887486)

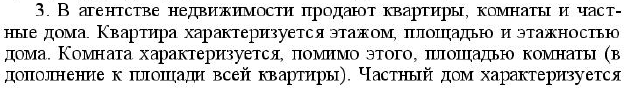
[4 Результаты тестов 5](#_Toc84887487)

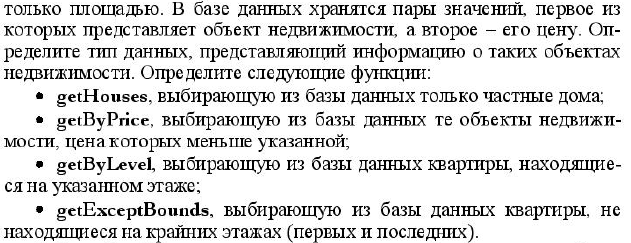
[5 Вывод 6](#_Toc84887488)

# 1 Цель работы

Научиться определять охраняющие условия, использовать сигнализацию об ошибках. Получить представление о механизме let-связывания. Приобрести навыки разработки пользовательских типов данных.

# 2 Текст задания





# 3 Текст программы

Main.hs

module Lab4 where

{--

Data types:

Flat level squareAll maxLevel

Room level squareAll maxLevel squareRoom

House squareAll

--}

data RealEstate = Null |

                  Flat Integer Double Integer |

                  Room Integer Double Integer Double |

                  House Double deriving(Eq, Show)

db = [(Flat 2 44.3 9, 40000),

      (Flat 5 56.1 5, 35000),

      (Room 3 77.0 8 12.5, 10000),

      (House 180.3, 70000)]

getHouses :: [(RealEstate, Integer)] -> [(RealEstate, Integer)]

getHouses [] = []

getHouses (((House squareAll), price):xs) = ((House squareAll), price) : (getHouses xs)

getHouses ((\_, \_):xs) = getHouses xs

{--Get all real estate items with price less than value --}

getByPrice :: [(RealEstate, Integer)] -> Integer -> [(RealEstate, Integer)]

getByPrice [] \_ = []

getByPrice (x:xs) value | (snd x) < value = x : (getByPrice xs value)

                        |  otherwise = (getByPrice xs value)

{--Get all flats with level = value --}

getByLevel :: [(RealEstate, Integer)] -> Integer -> [(RealEstate, Integer)]

getByLevel [] \_ = []

getByLevel (((Flat level squareAll maxLevel), price):xs) value | level == value = ((Flat level squareAll maxLevel), price) : (getByLevel xs value)

                                                               | otherwise = (getByLevel xs value)

getByLevel ((\_, \_):xs) value = (getByLevel xs value)

{--Get all flats excepting first and last levels --}

getExceptBounds :: [(RealEstate, Integer)] -> [(RealEstate, Integer)]

getExceptBounds [] = []

getExceptBounds (((Flat level squareAll maxLevel), price):xs) | ((level - 1) \* (maxLevel - level)) /= 0 = ((Flat level squareAll maxLevel), price) : (getExceptBounds xs)

                                                              | otherwise = (getExceptBounds xs)

getExceptBounds ((\_, \_):xs) = (getExceptBounds xs)

Здесь к каждой функции перед ее определением приведен краткий комментарий, описывающий выполняемую ею операцию. Кроме этого в самом начале исходного кода приведен комментарий, описывающий структуру данных.

# 4 Результаты тестов

Тестирование задания:

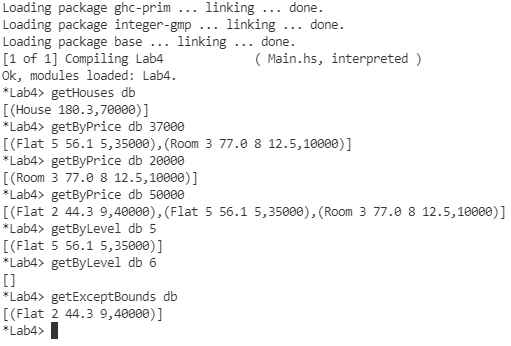


Рисунок 1 – тесты для задания

На рисунке 1 показана работа функций, указанные в задании. Работа этих функций продемонстрирована в том порядке, который был описан в задании.

# 5 Вывод

В ходе выполнения лабораторной работы я научился определять и использовать охраняющие условия, а также сигнализацию об ошибках. Также мною было получено представление о механизме let-связывания. Кроме этого я приобрел практические навыки разработки пользовательских типов в языке Haskell.