# Лабораторная работа 5, задание 7.1 (по варианту)

## Условие

Добавить к заданию 6.1 методы ввода/вывода с файла, а также ввод/вывод с консоли во время исполнения программы

Условия 6.1

a\*

r\*

l

c

e\*

r

i

e\*

d\*

e\*

e

z

t\*

f

s

Рис. 1: trie-дерево

Пример trie-дерева приведен на рис. 1. Булевское значение, равное True отмечает конец слова, читаемого, начиная с корня дерева. На рисунке узлы с такими значениями помечены символом \*. Таким образом, в дереве представлены слова fa, false, far, fare, fact, fried, frieze. Определите следующие функции:

1)exists, которая проверяет, что заданное слово содержится в trie-дереве.

## Код

{-

ПИ19-3 Деменчук

Задание 6.1 (по варианту 5)

Пример trie-дерева приведен на рис. 1. Булевское значение, равное True отмечает конец слова, читаемого, начиная с корня дерева.

На рисунке узлы с такими значениями помечены символом \*.

Таким образом, в дереве представлены слова fa, false, far, fare, fact, fried, frieze. Определите следующие функции:

1)exists, которая проверяет, что заданное слово содержится в trie-дереве.

-}

-- Созданный кастомный тип дерева

data MyTree = Unit Char Bool [MyTree] deriving (Eq, Show, Read)

-- Метод для проверки на наличие элемента в дерева

exist :: MyTree -> String -> Bool

exist t s = s `elem` (treeToList t "")

--Конвертер дерева в список

treeToList :: MyTree -> String -> [String]

treeToList (Unit c True \_) acc = [acc ++ [c]]

treeToList (Unit c False ts) acc = concatMap (\ u -> treeToList u (acc++[c])) ts

--Сохранение струкутры в файл

save :: MyTree -> FilePath -> IO ()

save editor f = writeFile f $ show editor

-- Ввод данных

readData :: String -> IO String

readData (userInput) = if (userInput == "1") then getLine else readFile "word.txt"

-- Конвертер для преобразования String в массив Char

myconverter :: String -> [Char]

myconverter = id

--Main-метод

main :: IO ()

main = do

--Читаем наш тип дерева из файла

buf <- readFile "tree.txt"

let tree = read buf :: MyTree

-- Спрашиваем пользователя о том, как он хочет ввести данные

putStrLn "Как считать данные о слове для поиска?\n1. С клавиатуры\n2. С файла"

userInput <- getLine

-- Вызов readData

input <- readData userInput

putStrLn ("Ввод: " ++ input)

-- Преобразование в массив char

let converted = myconverter input

-- Получение результата

let result = exist tree converted

-- Спрашиваем как выводить данные будем

putStrLn "Как вывести данные?\n1. На экран\n2. В файл"

userInput <- getLine

-- Запись ворматированного вывода в переменную

let formatedResult = "Результат: " ++ show result

-- Если вывод на экран

if userInput == "1" then do

putStrLn formatedResult

-- Если вывод в файл

else do

writeFile "result.txt" formatedResult

putStrLn "Занесли данные в файл"

save tree "tree.txt"

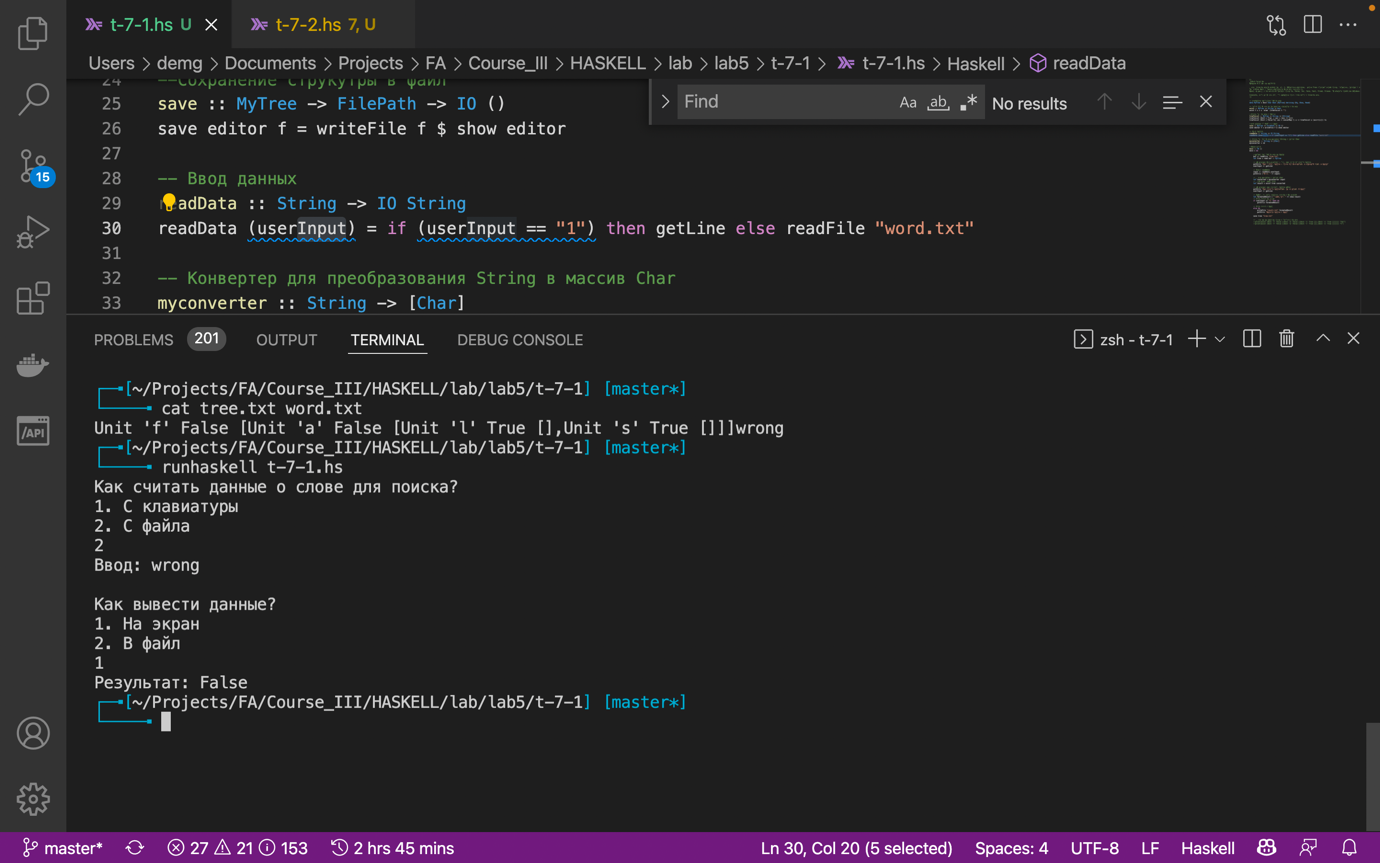
-- Проверка на наличие слова в тестовом дереве

--print(exist (Unit 'f' False [(Unit 'a' False [(Unit 'l' True []),(Unit 's' True [])])]) "fal")

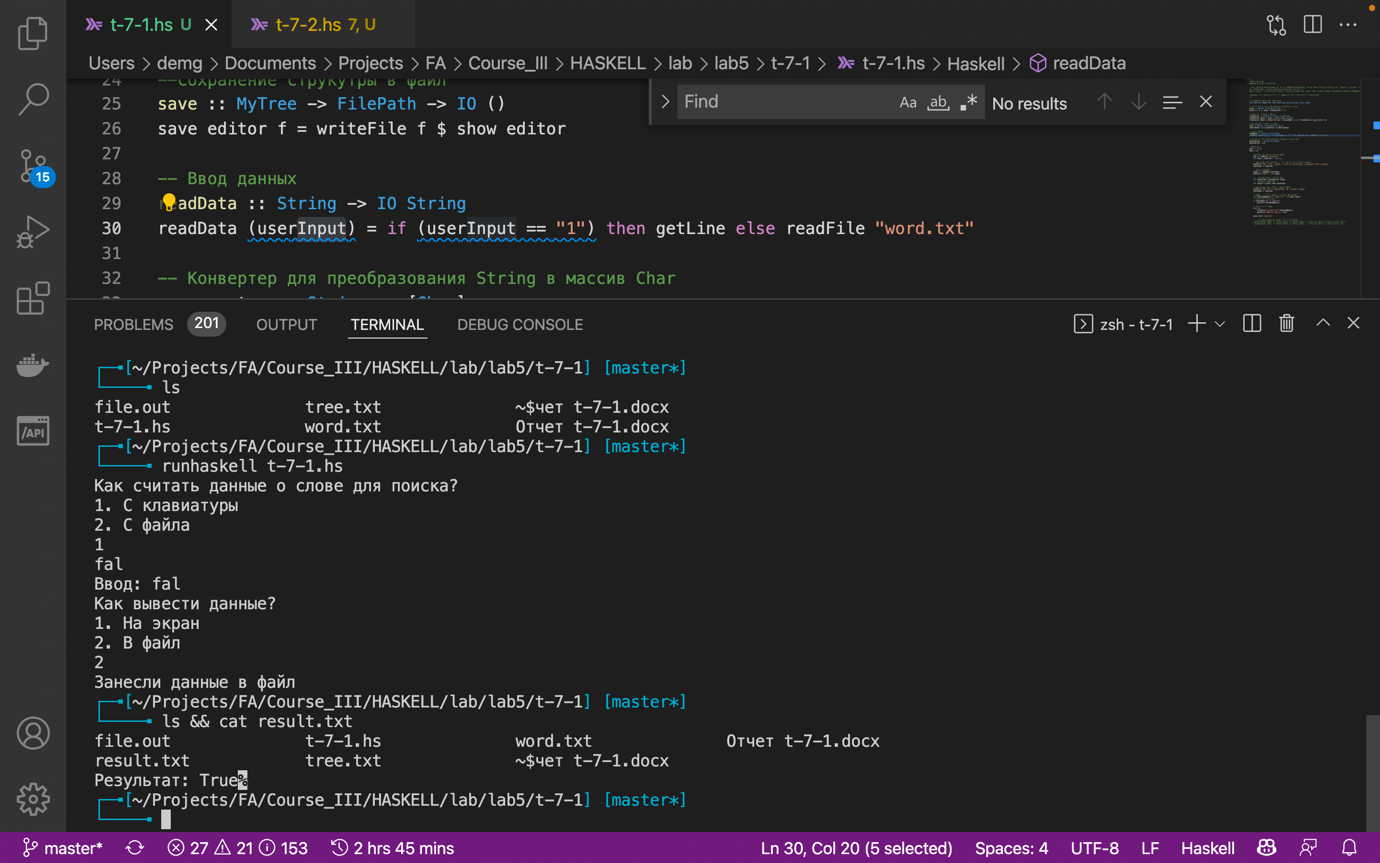
--print(exist (Unit 'f' False [(Unit 'a' False [(Unit 'l' True []),(Unit 's' True [])])]) "t")

## Скриншоты

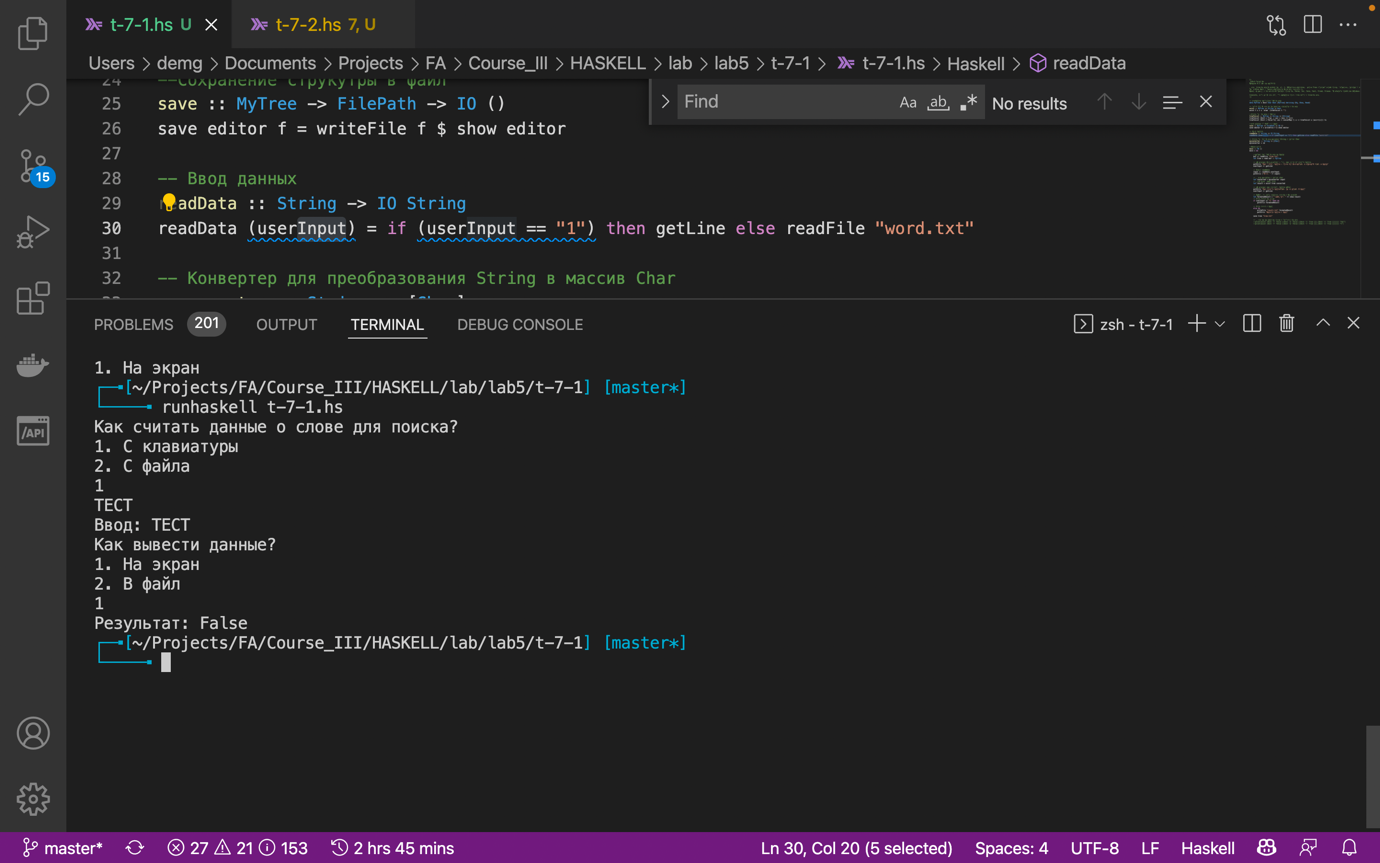
Пример ввода с файла и выводом в консоль (false)



Пример ввода с клавиатуры и выводом в файл (true)



Пример с вводом с клавиатуры и выводом в консоль (false)



Пример ввода с файла и выводом в файл (true)

