КУРСОВАЯ РАБОТА

по дисциплине «Языки и средства функционального программирования»

**Разработка приложения с использованием принципов функционального программирования**

Выполнила:

Студент гр. 3530904/80003 Шестакова А.Ю.

Проверил: Лукашин А. А.

**Оглавление**

[Введение 2](#_Toc27513024)

[Описание задачи 4](#_Toc27513025)

[Описание решения 5](#_Toc27513026)

[Cкриншоты программы 6](#_Toc27513027)

[Вывод 7](#_Toc27513028)

[Исходный код 8](#_Toc27513029)

# Введение

Функциональное программирование – это ветвь программирования, при котором программирование ведется с помощью определения функций.

В функциональном программировании нет ни процедур, ни циклов, нет даже переменных. Почти одни только функции. Функциональное программирование обладает рядом очень существенных преимуществ, которые не только позволяют ему существовать наряду с традиционным программированием, но и иметь своих поклонников, свою нишу задач и хорошие перспективы на будущее.

Ветвь программирования, начатая созданием Лиспа, понемногу развивалась с начала 60-х годов 20 века и привела к появлению целой плеяды очень своеобразных языков программирования, которые удовлетворяли всем требованиям, необходимым для исполнения программ несколькими параллельными процессорами. Во-первых, алгоритмы, записанные с помощью этих языков, допускают сравнительно простой анализ и формальные преобразования программ, а во-вторых, отдельные части программ могут исполняться независимо друг от друга. Языки, обладающие такими замечательными свойствами – это и есть языки функционального программирования. Помимо своей хорошей приспособленности к параллельным вычислениям языки функционального программирования обладают еще рядом приятных особенностей. Программы на этих языках записываются коротко, часто много короче, чем в любом другом традиционном (императивном) языке. Описание алгоритмов в функциональном стиле сосредоточено не на том, как достичь нужного результата (в какой последовательности выполнять шаги алгоритма), а больше на том, что должен представлять собой этот результат.

Единственный серьезный недостаток функционального стиля программирования состоит в том, что этот стиль не универсальный. Многие действительно последовательные процессы, такие как поведение программных моделей в реальном времени, игровые и другие программы, организующие взаимодействие компьютера с человеком, не выразимы в функциональном стиле.

Функциональное программирование позволяет несколько по-иному взглянуть вообще на процесс программирования, а некоторые приемы программирования, которые, предназначены для написания программ в чисто функциональном стиле, могут с успехом использоваться и в традиционном программировании.

# Описание задачи

Разработать графическое приложение “Лабиринт” с использованием принципов функционального программирования.

# Описание решения

Цель игры – дойти до “х”, не задевая “о”.

Поле по умолчанию задается следующее:

\ # \

\ ### \

\ I # x \

\ ### \

Требуется из изначальной позиции(I) дойти до конечной позиции(х).

Далее программа обрабатывает передвижения курсора:

EventSpecialKey KeyUpArrow -> move (0, -1)

EventSpecialKey KeyDownArrow -> move (0, 1)

EventSpecialKey KeyLeftArrow -> move (-1, 0)

EventSpecialKey KeyRightArrow -> move ( 1, 0)

И проверяет текущее положение по отношению к бортам и цели, производит подсчет сделанных шагов и координат:

move::(Int, Int) -> Game()

move(dx, dy) = do

GameState cur <- get

field <- ask

case cur of

Right(x, y) -> do

put $

case field V.!(y + dy)V.!(x + dx) of

Wall -> GameState $ cur

Road -> GameState $ Right (x + dx, y + dy)

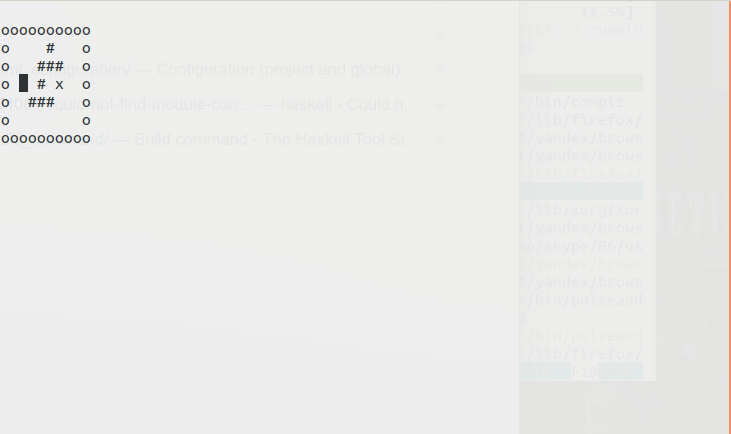
StartPos -> GameState $ Right (x + dx, y + dy)

Border -> GameState $ Left Died

Target -> GameState $ Left Win

Left \_ -> return ()

# Cкриншоты программы



# Вывод

В ходе выполнения курсовой работы были закреплены навыки использования принципов функционального программирования.

Использование принципов функционального программирования является очень удобным и продуктивным при написании программ. Функциональные программы не содержат операторов присваивания, а  переменные, получив однажды значение, никогда не изменяются. Более того, функциональные программы вообще не имеют побочных эффектов. Обращение к функции не вызывает иного эффекта кроме вычисления  результата. Это устраняет главный источник ошибок, и делает порядок выполнения функций несущественным: так как побочные эффекты не могут изменять значение выражения, оно может быть вычислено в любое время. Программист освобождается от бремени описания потока управления. Поскольку  выражения могут быть вычислены в любое время, можно свободно заменять переменные их значениями и наоборот, то есть программы "прозрачны по ссылкам". Эта прозрачность ссылок делает функциональные программы более удобными для математической обработки, по сравнению с общепринятыми аналогами.

# Исходный код

https://github.com/mycelium/hsse-fp-2019-2/tree/3530904/80003\_alina-shestakova/tasks/haskell/course\_project