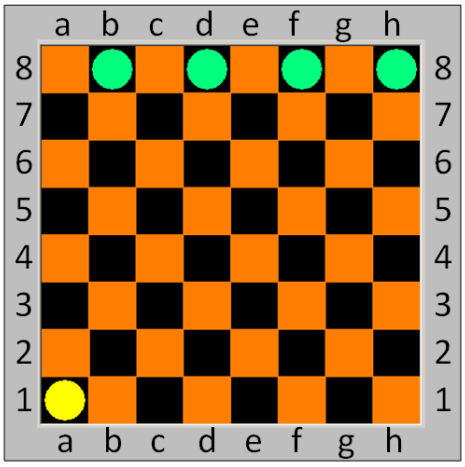
Игра "Овца и волки"

Правила

Волк и овцы — Википедия



Игра для двух игроков. На шахматной доске по краям расположены 4 волка (черные шашки) и 1 овца (белая шашка). Один игрок играет за волков, другой — за овцу. Игроки ходят по очереди, передвигая одну шашку за ход. Овца ходит первой. Шашки могут ходить по диагонали на ближайшую незанятую клетку — волки только вперед, овца — вперёд и назад.
Овца побеждает, если дошла до противоположного края доски (или у волков закончились доступные ходы).
Волки побеждают, если они овца не может больше ходить (она окружена или прижата к краю доски).

Описание типов данных

Положение шашек на доске и один ход описываются с помощью следующих типов данных:

```
-- Положение на доске
data Position = Position {f :: Char, r :: Int}
            deriving (Eq)

instance Show Position where
  show (Position f r) = f : show r

-- Один ход
data Step = Step {from :: Position, to :: Position}
            deriving (Eq)

instance Show Step where
  show (Step f t) = show f ++ " -> " ++ show t
```

Например, начальное положение овцы и ее единственный возможный ход:

```
ghci> Position 'a' 1
a1
ghci> Step {from=(Position 'a' 1), to=(Position 'b' 2)}
a1 -> b2
```

Текущее состояние доски и результат игры описываются с помощью следующих типов данных:

```
data Game = Game {sheep :: Position, wolfs :: [Position]}
            deriving (Eq)

data GameResult = SheepWin | WolfsWin
            deriving (Show)
```

Начальное состояние игры:

```
startSheep :: Position
startSheep = Position 'a' 1

startWolfs :: [Position]
startWolfs = [Position f 8 | f <- "bdfh"]

startGame :: Game
startGame = Game startSheep startWolfs
```

Задания

Задание 0 (0,5 балла)

Сделайте `Game` представителем класса типов `Show`.
Например:

```
ghci> startGame
Board:
 W W W W
. . . .
 . . . .
. . . .
 . . . .
. . . .
 . . . .
S . . .
```

Задание 1 (1 балл)

Напишите функции, возвращающие возможные ходы для каждого игрока согласно правилам игры.
Реализуйте два варианта:
1. с использованием механизма list comprehension
2. с использованием do-нотации (монада списка)

```
possibleSheepSteps :: Game -> [Step]
possibleSheepSteps = undefined
```

```
possibleWolfsSteps :: Game -> [Step]
possibleWolfsSteps = undefined
```

Например:

```
ghci> possibleSheepSteps startGame
[a1 -> b2]
ghci> possibleWolfsSteps startGame
[b8 -> a7,b8 -> c7,d8 -> c7,d8 -> e7,f8 -> e7,f8 -> g7,h8 -> g7]
```

Задание 2 (1,5 балла)

Напишите функцию, реализующую одну итерацию игры:

1. сначала ходит игрок, играющий за овцу
   - в случае отсутствия доступных ходов игра заканчивается победой волков
   - в случае, если овца дошла до противоположного края (ряд 8), игра заканчивается победой овцы
2. затем ходит игрок, играющий за волков (в случае отсутствия доступных ходов игра заканчивается победой овцы)

Если у игрока несколько доступных ходов, выбирайте первый из доступных.

```
simpleTurn :: Game -> Either GameResult Game
simpleTurn = undefined
```

Например:

```
ghci> simpleTurn startGame
Right Board:
 . W W W
W . . .
 . . . .
. . . .
 . . . .
. . . .
 S . . .
. . . .
```

Задание 3 (1 балл)

Напишите функцию, исполняющую итерацию игры до тех пор, пока игра не завершится:

```
run :: Monad m => (a -> m a) -> a -> m a
run = undefined
```

Например (результат зависит от порядка, в котором вы генерируете доступные ходы):

```
ghci> run simpleTurn startGame
```

```
Left WolfsWin
```

Задание 4 (2 балла)

Добавьте запись в лог (лог - список ходов). Напишите две версии функции, выполняющей одну итерацию игры:

```
loggedTurn :: Game -> (WriterT [Step] (Either GameResult)) Game
loggedTurn = undefined
loggedTurn' :: Game -> ExceptT GameResult (Writer [Step]) Game
loggedTurn' = undefined
```

Чем они отличаются? Какая предпочтительней? Проверьте работу с помощью функции `run`.

Задание 5 (2 балла)

Используя модуль `System.Random`, добавьте случайность в ходы. Теперь выбирается не первый доступный ход, а случайный. Самостоятельно придумайте сигнатуру функции реализующей одну итерацию игры (логгирование должно остаться). Проверьте работу с помощью функции `run`.

Задание 6 (2 балла)

Добавьте возможность пользователю определять ходы овцы. Теперь волки ходят случайно, а за овцу играет пользователь.
Самостоятельно придумайте сигнатуру функции реализующей одну итерацию игры (логгирование должно остаться). Проверьте работу с помощью функции `run`.