Wordle

Изготвен от Милица Тончева, 2MI0600216

Реализиран на Haskell

Решението ми може да бъде разделено на 4 семантични части.

Режим “игра” (game mode), който се разделя на 3 - нормален режим (normal), който представлява същинската игра на Wordle; лесен режим (easy), който в допълнение на правилата предупреждава играча, ако думата, с която играе, не е в речника или си противоречи с предишни опити; експертен режим (hard), в който програмата има право веднъж да излъже играча, давайки грешни цветове като отговори на предположена дума.

Режим “помощник” (helper mode), в който програмата се опитва да познае дума, намислена от играча.

Думите, които се използват по време на игра, се взимат от файлове, намиращи се в същата директория. Думите оригинално са взети от следния сайт: <https://www.ef.com/wwen/english-resources/english-vocabulary/top-1000-words/>; и са организирани в отделни файлове спрямо дължината им.

Дефинирани са два типа:

* **ColoredLetters** - списък от наредени двойки, който указва какви са цветовете на всяка буква в предположена от играча дума спрямо конкретна тайна дума (цвят, буква)

**Пример**: тайна дума: **eagle**, предположение: **apple**

ColoredLetters letters = [(‘y’,’a’), (‘r’,’p’), (‘r’,’p’), (‘g’,’l’), (‘g’,’e’)]

* **Rule** - наредена тройка, която задава правило в думата (индекс, буква, цвят), спрямо досегашните предположения на играча. Най-често функциите връщат списък от **Rule** с всички досега известни правила за думата.

**Пример**: тайна дума: **eagle**, предположение: **apple**, при следващо предложение rules изглежда така

[Rule] rules = [(5, ‘а’,’y’), (5, ‘p’,’r’), (3, ‘l’,’g’), (4, ‘e’,’g’)]

Тоест в думата сме срещнали едно “жълто” ‘а’, “сиво” ‘p’, “зелено” ‘l’ на индекс 3 и “зелено” ‘e’ на индекс 4

‘g’ - зелено

‘y’ - жълто

‘r’ - сиво

В **Rule** първия елемент указва индекс. Ако правилото е за зелена буква, индексът е позицията на зелената буква в думата. Ако правилото е за жълта или сива буква, то индексът е равен на дължината на думата.

1. Меню:

Въвежда се ‘g’ или ‘h’, в зависимост от това в кой режим искаме да играем;

След това се въвежда дължината на думата, с която искаме да играем; Ако сме избрали режим “игра”, трябва да въведем ‘n’, ‘e’, или ‘h’ за нормален, лесен и експертен режим.

1. Режим “Игра” - нормален режим

След всяко въвеждане на предложена дума от играча тази дума се проверява спрямо тайната дума чрез функцията **checkGuess**, която първо проверява дали има зелени букви в предложението на играча, след това проверява дали има жълти и останалите маркира като сиви. Функцията връща **(Bool, ColoredLetters, [Rule]),** но в нормален режим се използва само първият елемент, който показва дали думата е позната или играта продължава. Думата се принтира от функцията **printGuess,** която се вика от **checkGuess.** Когато думата е позната, се принтира в зелено и играта приключва.

1. Режим “Игра” - лесен режим

След въвеждане на предположение от играча, програмата първо проверява дали дадената дума се съдържа в списъка от думи, с който разполага. Ако не, уведомява играча, който трябва да отговори дали въпреки това желае да продължи с това предположение. Ако да, програмата преминава към проверка дали предложената от играча дума си противоречи с предишните предположения. Това се осъществява чрез функциите **checkForContradiction** и **isContradiction.** Втората проверява 3 неща:

* за всяко “зелено” правило, което досега сме събрали - дали в предложената от играча дума на индекса, който правилото указва, се срещата същата “зелена” буква
* за всяко “жълто” правило - дали броят на “жълтите” правила за конкретна буква е равен на броя срещания на тази буква в предложената дума. **Пример**: правила: [(5, ‘a’, ‘y’), (2, ‘k’, ‘g’), (5, ‘a’, ‘y’)]; от правилата следва, че в думата се срещат поне 2 ‘а’-та; функцията проверява дали и двете са използвани в предположението на играча
* за всяко “сиво” правило - дали има буква от предположението на играча, която е била маркирана като сива спрямо тайната дума, за която вече имаме правило в правилата. **Пример**: правила: [(5, ‘a’, ‘y’), (2, ‘g’, ‘g’), (5, ‘n’, ‘r’)]; предположение: vegan -> ‘n’ вече сме я срещали като сива -> противоречие

Ако открием противоречие, отново питаме играчът дали иска да продължи с предложената дума. Трябва да въведе ‘y’ за да или ‘n’ за не. Останалата част от функционалността за този режим се припокрива с нормален режим.

1. Режим “Игра” - експертен режим

В този режим след всяко въвеждане на предположение от играч се избира случайно число между 0 и 3. Ако числото е равно на 1, тогава програмата ще се опита да излъже играча с отговора си чрез функцията **lie**. В нея се избира отново на случаен принцип буква от вече маркираното спрямо тайната дума предложение на играча (**Пример**: тайна дума: **apple**, предположение: **eagle**), с която програмата да излъже. Оттам логиката се разделя на 3:

* ако избраната буква е “зелена” проверяваме

1. дали вече имаме правило, което казва, че тази “зелена” буква сме я срещали на същото място. **Пример**: избрали сме да излъжем с ‘**e**’. Ако в правилата вече има правило (4, ‘e’, ‘g’), значи вече знаем, че на индекс 4 стои ’е’.
2. дали буквата се среща изобщо в правилата

Ако някое от тези две условия е изпълнено, се търси нова буква, с която програмата да се опита да излъже. Ако и двете са грешни, програмата прави избраната “зелена” буква “сива”.

* ако избраната буква е “жълта”

1. ако буквата не се съдържа в правилата - програмата я прави “сива”
2. ако буквата се съдържа, проверяваме дали на позицията, на която е в момента, сме срещали “зелена” буква
   1. ако да, търсим нова буква, с която програмата да се опита да излъже
   2. ако не, проверяваме дали ако я направим “зелена”, няма да останат една “жълта” и всички останали “зелени” (**Пример**: тайна дума: **life,** предположение: **file;** Aко излъжем с **l** и я направим “зелена”, резултатът ще бъде **file** и ще имаме една буква на грешно място и 0 места, на които да я сложим)
      1. ако не - програмата я прави “зелена”
      2. ако да -програмата търси нова буква, с която да се опита да излъже

* ако избраната буква е “сива”

1. ако всички букви от тайната дума са ни известни - програмата търси нова буква, с която да се опита да излъже
2. ако освен буквата, с която сме решили да излъжем, няма поне още една “сива” буква или поне още две “жълти” букви - програмата търси нова буква, с която да се опита да излъже
3. ако буквата не се съдържа в правилата, програмата я прави “жълта”
4. ако буквата се съдържа в правилата като сива - програмата търси нова буква, с която да се опита да излъже
5. ако буквата се съдържа в правилата като жълта или зелена, проверяваме дали на позицията, на която е в момента, сме срещали “зелена” буква
   1. ако да, програмата търси нова буква, с която да се опита да излъже
   2. ако не, програмата я прави “зелена”

Ако програмата не успее да излъже с която и да е буква от думата, то тогава играта продължава. На следващия рунд може пак да се опита да излъже, стига случайно избраното число между 0 и 3 да е 1-ца. Ако успее да излъже, не може да опитва повече.

Когато лъже, програмата отпечатва “фалшивите” цветове на предложената от играча дума, но само реалните правила се запазват. При следващ опит буквата, с която програмата е излъгала, ще бъде показана с истинските си цветове.

1. Режим “помощник”

В този режим на играта целта на програмата е да познае думата, намислена от играча, като единственото, което знае за нея, е дължината ѝ. Алгоритъмът за избиране на най-оптимално предположение, което ще елиминира най-много от останалите възможности, може да се представи по следния начин:

* За всяка дума от списъка с възможни думи (тази дума ще служи за предположение)
  + За всяка дума (различна от горната) от списъка с възможни думи (тази дума ще служи за “тайна дума”)
    - изчислява се колко от останалите думи биха били противоречия като последвало предположение (елиминирани) за конкретната двойка предположение - тайна дума
* Извеждаме думата, която в ролята си на предположение е “елиминирала” най-много думи от списъка с възможности.

**Пример**:

речник: ["apple", "peach", "berry", "grape"]

“apple” елиминира общо 5 думи;

“peach” елиминира общо 5 думи

“berry” елиминира общо 5 думи;

“grape” елиминира общо 6 думи

Резултат: “grape”

След извеждането на предположение от страна на програмата, играчът трябва да въведе последователност от символи, които указват коя буква от предположението на програмата с какъв цвят трябва да бъде маркирана. Приемливите символи са: ‘**g**’ за зелено, ‘**y**’ за жълто и ‘**r**’ за сиво. Ако броят символи не съответства на дължината на търсената дума, играчът трябва да въведе отговора си наново. Ако бъдат въведени само ‘**g**’-та програмата приема, че е познала, и играта приключва.

Чрез въведените от играча символи се създават списък от тип **ColoredLetters** и списък от правила [**Rule**], които се използват за последвалите предположения на програмата. При последвалите изпълнения на алгоритъма на всяка итерация, която избира предположение, програмата ще проверява дали то противоречи на правилата, събрани дотук. Ако да, програмата преминава към следващата итерация. Ако програмата не успее да намери дума, която да не противоречи на събраните правила, уведомява играча, че не съществува такава дума в речника, и играта приключва.

Код, взет от генеративен модел (GPT)

1. Принтиране на оцветените букви

-- used model OpenAi ChatGPT

-- prompt: i have this list [('g','a'),('y','t'),('r','i'),('r','m'),('y','b')]

-- i want to print each letter in the color that the fst of each pair says (g for green, y for yellow and r for grey)

-- original response:

-- Define color codes for terminal output

-- green = "\x1b[32m" -- Green

-- yellow = "\x1b[33m" -- Yellow

-- reset = "\x1b[0m" -- Reset color to default

-- grey = "\x1b[90m" -- Grey

-- -- Function to print each letter in its respective color

-- printColoredLetters :: [(Char, Char)] -> IO ()

-- printColoredLetters [] = return () -- Base case for empty list

-- printColoredLetters ((color, letter):xs) = do

-- let colorCode = case color of

-- 'g' -> green

-- 'y' -> yellow

-- 'r' -> grey

-- \_ -> reset

-- putStr (colorCode ++ [letter] ++ reset) -- Print the letter with the color

-- printColoredLetters xs -- Recursive call to print the rest of the list

-- changes:

-- removed comments; changed type to ColoredLetters

green :: String

green = "\x1b[32m"

yellow :: String

yellow = "\x1b[33m"

reset :: String

reset = "\x1b[0m"

grey :: String

grey = "\x1b[90m"

printColoredLetters :: ColoredLetters -> IO ()

printColoredLetters [] = return ()

printColoredLetters ((color, letter):xs) = do

let colorCode = case color of

'g' -> green

'y' -> yellow

'r' -> grey

\_ -> reset

putStr (colorCode ++ [letter] ++ reset)

printColoredLetters xs

1. Събиране на всички думи от файл в списък

-- used model OpenAi ChatGPT

-- prompt: how do i put all the lines of file in an array

-- original response:

-- main :: IO ()

-- main = do

-- -- Open the file for reading

-- handle <- openFile "example.txt" ReadMode

-- -- Read the entire file content lazily and split it into lines

-- contents <- hGetContents handle

-- let linesArray = lines contents

-- -- Print the lines stored in the array

-- print linesArray

-- -- Close the file

-- hClose handle

-- changes:

-- only used "let linesArray = lines contents"; changed variable name

let wordlist = lines content

1. Генериране на случайно число

-- used model OpenAi ChatGPT

-- prompt: how do i generate a random number in an interval

-- original response:

-- randomNumber <- randomRIO (1, 100) :: IO Int

-- changes:

-- variable name; interval

pos <- randomRIO (0, length wordlist - 1) :: IO Int