## Задание 1. Построить парсер для данных {indent = integer}

x1 = 100

abc = 4

x1 = 10

дополнительно: получить сумму значений для указанного имени (для x1 - 110)

## 

-- 1. Построить парсер для данных {indent = integer}

splitOn :: String -> String -> (String, String)

splitOn sub str = let

splitOnA' [] s = ([], s)

splitOnA' (y:ys) (x:xs)

| x == y = let (xs', xs'') = splitOnA' ys xs in (xs', xs'')

| otherwise = let (xs', xs'') = splitOnA' sub xs in (x:xs', xs'')

splitOnA' \_ \_ = ([], [])

in splitOnA' sub str

w = [';', '\n', ' ', '\t', ',']

lstrip :: String -> String

lstrip = dropWhile (`elem` w)

rstrip :: String -> String

rstrip = reverse . lstrip . reverse

strip :: String -> String

strip = rstrip . lstrip

findVars :: String -> [(String, String)]

findVars [] = []

findVars s = let

(varName, xs) = splitOn "=" s

(value, x) = span (`notElem` w) (lstrip xs)

in

(strip varName, strip value):findVars x

parseValuesToInt :: [(a, String)] -> [(a, Integer)]

parseValuesToInt = map (\(x, y) -> (x, read y::Integer))

findVarsAndParseValuesToInt :: String -> [(String, Integer)]

findVarsAndParseValuesToInt = parseValuesToInt . findVars

sumBy :: Num a => String -> [(String, a)] -> a

sumBy varName = foldl (\acc (\_, value) -> acc + value) 0 . filter (\(x, \_) -> x == varName)

main :: IO ()

main = do

let fileInPath = "test.txt"

content <- readFile fileInPath

putStrLn "data"

let vars = findVars content

print (findVars content)

putStrLn "parsed data"

let varsInInt = parseValuesToInt vars

print varsInInt

putStrLn "sum x1"

print (sumBy "x1" varsInInt)

Вывод:

Main> :main

data

[("x1","100"),("abc","4"),("x1","10"),("x2","34"),("x1","503"),("cba","-4"),("x1","-1")]

parsed data

[("x1",100),("abc",4),("x1",10),("x2",34),("x1",503),("cba",-4),("x1",-1)]

sum x1

612