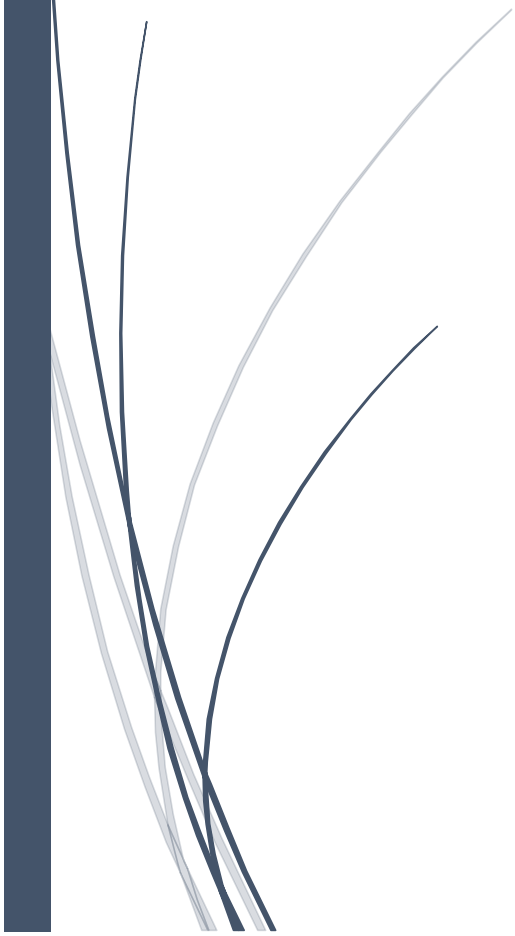
8-6-2023

MEMORIA

PF_Practica4.hs

(El poeta informático)

Sergio Heras Álvarez (20M025)
GRADO MATEMÁTICAS E INFORMÁTICA (UPM)



Funcionamiento general del programa

El objetivo del programa era a partir de un poema poder determinar sus características como el numero de sus versos, de qué tipo son cada uno, que rima se emplea como también la composición de cada párrafo en la que podremos saber la estructura final del poema.

Para ello, he hecho uso de la siguiente tabla :

NOMBRE	Nº DE VERSOS	Nº DE SÍLABAS	RIMA	ESQUEMA MÉTRICO
Pareado	Dos versos	Arte mayor o menor	Consonante	a, a A, A
Terceto	Tres versos	Arte mayor (endecasílabos)	Consonante	A, B, A B, C, B
Cuarteto	Cuatro versos	Arte mayor (endecasílabos)	Consonante	A, B, B, A
Redondilla	Cuatro versos	Arte menor (octosílabos)	Consonante	a, b, b, a
Serventesio	Cuatro versos	Arte mayor (endecasílabos)	Consonante	A, B, A, B
Cuarteta	Cuatro versos	Arte menor (octosílabos)	Consonante	a, b, a, b
Cuaderna vía	Cuatro versos	Arte mayor (alejandrinos)	Consonante	A, A, A, A
Quinteto	Cinco versos	Arte mayor (endecasílabos)	Consonante	A, B, A, B, A
Quintilla	Cinco versos	Arte menor (octosílabos)	Consonante	a, b, a, b, a
Lira	Cinco versos	Arte mayor y menor (endecasílabos y heptasílabos)	Consonante	7a, 11B, 7a, 7b, 11B
Octava real	Ocho versos	Arte mayor (endecasílabos)	Consonante	A, B, A, B, A, B, C, C
Décima o espinela	Diez versos	Arte menor (octosílabos)	Consonante	a, b, b, a, a, c, c, d, d, c
Soneto	Catorce versos	Arte mayor (endecasílabos)	Consonante	A, B, B, A A, B, B, A C, D, C D, C, D
Romance	Serie ilimitada	Arte menor (octosílabos)	Asonante	-, a ,-, a, -, a, -, a, ...
Silva y estancia	Serie ilimitada	Arte mayor y menor (heptasílabos y endecasílabos)	Consonante	

Mi programa se basa en una serie de funciones que he creado para cada apartado de la tabla, además de otras funciones auxiliares para la implementación de éstas. El usuario no tiene que ir llamando a cada una de las funciones. El programa está compuesto de un main en el que se le introduce un poema cualquiera y al ejecutarlo automáticamente hará el estudio de dicho poema. El programa te imprimirá :

1) El número de versos
2) El tipo de versos
3) Clasificación de cada párrafo
4) Nombre/Estructura del poema (pareado, terceto, soneto…)
5) El esquema métrico
6) El tipo de rima

Para aportar una mayor eficiencia al programa he obviado los resultados en los que no existe un caso contemplado, es decir, si por ejemplo se introduce un poema de 7 versos , en el apartado de Nombre/Estructura del poema no aparecerá nada en lugar de que pusiera un texto alternativo como “No existe” cuando en la realidad sí pudiera existir. He implementado los casos más relevantes de la literatura ya que hay infinidad de posibilidades. A continuación, explicaré el funcionamiento de cada función y cómo lo he planteado.

Explicaciones / razonamientos

- **LIBRERIAS**

```
import Data.List (group, sort)
import Data.Char (toLower, isSpace, isAlpha)
import Data.List (isSuffixOf)
import Data.Char (toLower)
import Data.Char (toLower, isUpper)
import Data.List (findIndex)
import Data.List
import Data.Maybe (mapMaybe)
import Data.Maybe (fromMaybe)
```

Estas son las librerías que he importado en el programa. Cada una la utilizado para el correcto funcionamiento de las funciones que ahora iré explicando más detalladamente.

- **contarVocales**

```
esVocal :: Char -> Bool
esVocal c = toLower c `elem` "aeiouáéíóúh"

esSinalefa :: Char -> Char -> Bool
esSinalefa c1 c2 = esVocal c1 && esVocal c2

contarVocales :: String -> Int
contarVocales frase = (length . filter esVocal $ sinEspacios)
                    - length (filter (uncurry esSinalefa) $ zip sinEspacios (tail sinEspacios))
 where
  sinEspacios = filter (/=' ') frase
```

La función contarVocales cuenta el número de vocales en una cadena de texto para simular el conteo de silabas de un verso. Comienza filtrando los espacios en blanco de la cadena original y la asigna a la variable sinEspacios. Luego, utiliza la composición de funciones para contar las vocales en sinEspacios aplicando la función esVocal y tomando la longitud del resultado filtrado. A continuación, resta la longitud de las sinalefas detectadas en sinEspacios utilizando la función esSinalefa en la lista de caracteres resultante de combinar cada carácter con su siguiente mediante zip y tail.

La función esVocal toma un carácter como entrada y devuelve True si el carácter es una vocal (incluyendo las vocales acentuadas) o la letra "h" y False en caso contrario.

La función esSinalefa toma dos caracteres como entrada y devuelve True si ambos caracteres son vocales. Es utilizada para verificar si se produce una sinalefa entre dos vocales consecutivos.

- **tipoRima**

```
tipoRima :: String -> String -> String
tipoRima verso1 verso2
  | ultimaPalabra verso1 `rimaConsonante` ultimaPalabra verso2 = "Rima consonante"
  | ultimaPalabra verso1 `rimaAsonante` ultimaPalabra verso2 = "Rima asonante"
  | otherwise = "No hay rima"

longitudSufijo :: String -> Int
longitudSufijo palabra = case clasificarPalabra palabra of
  Just Aguda    -> 2
  Just Llana    -> 3
  Just Esdrújula -> 3
  Nothing       -> 0  -- este caso no debería ocurrir si la palabra es válida

rimaConsonante :: String -> String -> Bool
rimaConsonante palabra1 palabra2 =
  let n = max (longitudSufijo palabra1) (longitudSufijo palabra2)
  in lastN n palabra1 `isSuffixOf` lastN n palabra2

rimaAsonante :: String -> String -> Bool
rimaAsonante palabra1 palabra2 =
  let n = max (longitudSufijo palabra1) (longitudSufijo palabra2)
  in lastN n palabra1 `isSuffixOf` lastN n palabra2

-- Función auxiliar para obtener la última palabra de un verso
ultimaPalabra :: String -> String
ultimaPalabra verso = last (words verso)

-- Función auxiliar para obtener los últimos n caracteres de una cadena
lastN :: Int -> [a] -> [a]
lastN n = reverse . take n . reverse
```

La función tipoRima y sus funciones auxiliares se utilizan para determinar el tipo de rima entre dos versos en un poema (consonante , asonante o no hay rima).

La función tipoRima toma dos versos como entrada (verso1 y verso2) y devuelve un valor de tipo String que indica el tipo de rima entre ellos. La función utiliza las funciones auxiliares ultimaPalabra, rimaConsonante y rimaAsonante para determinar si las últimas palabras de los versos riman de manera consonante, asonante o no riman.

La función longitudSufijo toma una palabra como entrada y devuelve la longitud del sufijo que determina su acentuación (aguda, llana o esdrújula). La función utiliza la función clasificarPalabra para obtener la acentuación de la palabra y asigna un valor numérico correspondiente a cada tipo de acentuación.

Las funciones rimaConsonante y rimaAsonante se utilizan para verificar si dos palabras riman de manera consonante o asonante, respectivamente. Ambas funciones comparan los sufijos de

longitud máxima de las palabras y utilizan la función lastN para obtener los últimos caracteres de una cadena.

La función ultimaPalabra es una función auxiliar que toma un verso y devuelve la última palabra del mismo. Utiliza la función words para dividir el verso en palabras y selecciona la última palabra con la función last.

La función lastN es una función auxiliar que toma un entero n y una lista y devuelve los últimos n elementos de la lista. La función utiliza la función reverse para invertir la lista, luego toma los primeros n elementos con take y finalmente invierte nuevamente el resultado con otra llamada a reverse.

- **tipoVerso**

```
tipoVerso :: String -> Maybe String
tipoVerso verso
  | numSilabas == 4 = Just "Arte menor (Tetrasilabos)"
  | numSilabas == 5 = Just "Arte menor (Pentasilabos)"
  | numSilabas == 6 = Just "Arte menor (Hexasilabos)"
  | numSilabas == 7 = Just "Arte menor (Heptasilabos)"
  | numSilabas == 8 = Just "Arte menor (Octosilabos)"
  | numSilabas == 9 = Just "Arte Mayor (Eneasilabos)"
  | numSilabas == 10 = Just "Arte Mayor (Decasilabos)"
  | numSilabas == 11 = Just "Arte Mayor (Endecasilabos)"
  | numSilabas == 14 = Just "Arte Mayor (Alejandrinos)"
  | otherwise = Nothing
  where
    numSilabas = contarVocales verso

tiposDeVersos :: String -> [Maybe String]
tiposDeVersos poema = map tipoVerso (lines poema)
```

La función tipoVerso se utiliza para determinar el tipo de verso en función del número de sílabas en un poema. Toma un verso como entrada y devuelve un valor de tipo Maybe String, que indica el tipo de verso correspondiente. La función verifica el número de sílabas del verso y devuelve una descripción del tipo de verso en función de ese número. Si el número de sílabas no coincide con ningún tipo de verso conocido, se devuelve Nothing. La función utiliza la función contarVocales para contar el número de sílabas en el verso.

La función tiposDeVersos toma un poema como entrada y devuelve una lista de valores de tipo Maybe String, que representa los tipos de verso correspondientes a cada verso del poema. La función utiliza la función tipoVerso en combinación con la función lines para aplicar tipoVerso a cada verso individual del poema.

- **NumeroDeVersos**

```
numeroDeVersos :: [String] -> Int
numeroDeVersos poema = length $ filter (not . null) poema
```

La función numeroDeVersos se utiliza para contar el número de versos en un poema. Toma un poema representado como una lista de cadenas de texto [String] como entrada.

Utiliza la función filter para eliminar las líneas vacías del poema, dejando solo los versos.

Luego, utiliza la función length para contar el número de elementos en la lista resultante, que representa el número de versos en el poema.

Finalmente, devuelve el número de versos como un valor entero (Int).

- **clasificarPalabra**

```
indiceCaracterAcentuado :: String -> Maybe Int
indiceCaracterAcentuado palabra = aux (reverse $ map toLower palabra) 1
 where
  esCaracterAcentuado c = c `elem` "áéíóú"
  aux [] _ = Nothing
  aux (c:cs) i
   | esCaracterAcentuado c = Just i
   | otherwise = aux cs (i+1)

esdrújula2 :: String -> Bool
esdrújula2 palabra = case indiceCaracterAcentuado palabra of
                      Just n  -> n >3
                      Nothing -> False


data TipoPalabra = Aguda | Llana | Esdrújula deriving Show

clasificarPalabra :: String -> Maybe TipoPalabra
clasificarPalabra palabra
 | esAguda palabra = Just Aguda
 | esLlana palabra = Just Llana
 | esEsdrújula palabra = Just Esdrújula
 | otherwise = Nothing
 where

  esAguda p = (not(esdrújula2 p ) && (terminaEnVocal p || terminaEnNS p) ) || ( all (`notElem` "áéíóú") (map toLower p) && not(terminaEnVocal p || terminaEnNS p) )
  esLlana p = ( esUltimaSilabaAcentuada p && terminaEnConsonanteNoNS p ) || ( all (`notElem` "áéíóú") (map toLower p) && (terminaEnVocal p || terminaEnNS p) )
  esEsdrújula p = esdrújula2 p
  esUltimaSilabaAcentuada = any (`elem` "áéíóúÁÉÍÓÚ") . last . palabras
  terminaEnVocal = (`elem` "aeiouáéíóúAEIOUÁÉÍÓÚ") . last . quitarAcentos
  terminaEnNS p = last (quitarAcentos p) `elem` "nsNS"
  terminaEnConsonanteNoNS p = last (quitarAcentos p) `notElem` "aeiouáéíóúAEIOUÁÉÍÓÚnsNS"
  quitarAcentos = map quitarAcento
  quitarAcento c
   | c `elem` "áÁ" = 'a'
   | c `elem` "éÉ" = 'e'
   | c `elem` "íÍ" = 'i'
   | c `elem` "óÓ" = 'o'
   | c `elem` "úÚ" = 'u'
   | otherwise = c
  palabras = words . map toLower
```

La función de clasificarPalabra y las funciones auxiliares se utilizan para clasificar una palabra en español en una de las tres categorías: Aguda, Llana o Esdrújula, según su acentuación y terminación silábica que utilizaremos en la funcion tipoRima.

indiceCaracterAcentuado es una función que recibe una palabra y devuelve el índice (contando desde el final) del primer carácter acentuado en la palabra. Si no hay carácter acentuado, devuelve Nothing.

esdrújula2 es una función que recibe una palabra y determina si es esdrújula. Utiliza indiceCaracterAcentuado para verificar si el índice del carácter acentuado es mayor a 3 empezando desde el final de la palabra. He asumido que la mayor parte de las esdrújulas recogen este caso.

TipoPalabra es un tipo de dato enumerado creado para representar las tres categorías de palabras: Aguda, Llana y Esdrújula.

clasificarPalabra es una función que recibe una palabra y la clasifica en una de las categorías de TipoPalabra. Utiliza varias funciones auxiliares para determinar la clasificación de la palabra:

- esAguda verifica si una palabra es aguda basándose en diferentes criterios, como si es esdrújula negada y si termina en vocal o en "n" o "s".
- esLlana verifica si una palabra es llana basándose en si la última sílaba está acentuada y si termina en consonante no "n" ni "s".
- esEsdrújula utiliza la función esdrújula2 para verificar si una palabra es esdrújula.
- esUltimaSilabaAcentuada verifica si la última sílaba de una palabra está acentuada.
- terminaEnVocal verifica si una palabra termina en vocal.
- terminaEnNS verifica si una palabra termina en "n" o "s".
- terminaEnConsonanteNoNS verifica si una palabra termina en una consonante que no es "n" ni "s".
- quitarAcentos reemplaza los caracteres acentuados por sus equivalentes sin acento.
- palabras separa la cadena de texto en palabras individuales, convierte todas las letras a minúsculas y devuelve una lista de palabras.

- **modoDeRima**

```haskell
modoDeRima :: [String] -> Maybe String
modoDeRima versos
  | length versos == 2 && rimaAA = Just "Rima AA"
  | length versos == 2 && rimaaa = Just "Rima aa"
  | length versos == 3 && rimaABA = Just "Rima ABA"
  | length versos == 4 && rimaABBA = Just "Rima ABBA"
  | length versos == 4 && rimaABAB = Just "Rima ABAB"
  | length versos == 4 && rimaabba = Just "Rima abba"
  | length versos == 4 && rimaabab = Just "Rima abab"
  | length versos == 4 && rimaAAAA = Just "Rima AAAA"
  | length versos == 5 && rimaABABA = Just "Rima ABABA"
  | length versos == 5 && rimaababa = Just "Rima ababa"
  | rimaABABABCC = Just "Rima ABABABCC"
  | rimaabbaaccddc = Just "Rima abbaaccddc"
  | otherwise = Nothing
  where
    todosVersos11Silabas = all (\verso -> contarVocales verso == 11) versos
    todosVersos8Silabas = all (\verso -> contarVocales verso == 8) versos
    rimaAA = todosVersos11Silabas && tipoRima (versos !! 0) (versos !! 1) == "Rima consonante"
    rimaaa = todosVersos8Silabas && tipoRima (versos !! 0) (versos !! 1) == "Rima consonante"
    rimaABA = todosVersos11Silabas && tipoRima (versos !! 0) (versos !! 2) == "Rima consonante"
    rimaABBA = todosVersos11Silabas && tipoRima (versos !! 0) (versos !! 3) == "Rima consonante" &&
               tipoRima (versos !! 1) (versos !! 2) == "Rima consonante"
    rimaABAB = todosVersos11Silabas && tipoRima (versos !! 0) (versos !! 2) == "Rima consonante" &&
               tipoRima (versos !! 1) (versos !! 3) == "Rima consonante"
    rimaAAAA = todosVersos11Silabas && all (== "Rima consonante") [ tipoRima v1 v2 | v1 <- versos, v2 <- versos, v1 /= v2 ]
    rimaabba = todosVersos8Silabas && tipoRima (versos !! 0) (versos !! 3) == "Rima consonante" &&
               tipoRima (versos !! 1) (versos !! 2) == "Rima consonante"
    rimaabab = todosVersos8Silabas && tipoRima (versos !! 0) (versos !! 2) == "Rima consonante" &&
               tipoRima (versos !! 1) (versos !! 3) == "Rima consonante"
    rimaABABA = todosVersos11Silabas && tipoRima (versos !! 0) (versos !! 2) == "Rima consonante" &&
               tipoRima (versos !! 1) (versos !! 3) == "Rima consonante" &&
               tipoRima (versos !! 0) (versos !! 4) == "Rima consonante"
    rimaababa = todosVersos8Silabas && tipoRima (versos !! 0) (versos !! 2) == "Rima consonante" &&
               tipoRima (versos !! 1) (versos !! 3) == "Rima consonante" &&
               tipoRima (versos !! 0) (versos !! 4) == "Rima consonante"
    rimaABABABCC = length versos == 8
    rimaabbaaccddc = length versos == 10

dividirEnParrafos :: [String] -> [[String]]
dividirEnParrafos [] = []
dividirEnParrafos xs = parrafo : dividirEnParrafos (dropWhile null resto)
  where
    (parrafo, resto) = break null xs


modosDeRima :: [String] -> [Maybe String]
modosDeRima poema = map modoDeRima (dividirEnParrafos poema)
```

La función modoDeRima se encarga de determinar el modo de rima presente en un conjunto de versos. El resultado es una cadena de texto que describe el modo de rima encontrado o Nothing en caso de que no se cumpla ningún modo de rima definido.

La función utiliza varias funciones auxiliares para realizar las comprobaciones necesarias:

- todosVersos11Silabas verifica si todos los versos tienen 11 sílabas.
- todosVersos8Silabas verifica si todos los versos tienen 8 sílabas.
- También implemento cada posible métrica correspondiente de la tabla inicial y sacará como resultado el caso que se cumpla(rimaAA,rimaaa,rimaABA,etc)

La función dividirEnParrafos se encarga de dividir el poema en párrafos, considerando que un párrafo está delimitado por líneas vacías. Retorna una lista de listas de versos, donde cada sublista representa un párrafo.

La función modosDeRima recibe un poema y utiliza la función dividirEnParrafos para obtener los párrafos del poema. Luego, aplica la función modoDeRima a cada párrafo y retorna una lista de resultados, donde cada resultado describe el modo de rima del respectivo párrafo o Nothing si no se cumple ningún modo de rima.

- **clasificarPoema**

```
numeroDeParrafos :: String -> Int
numeroDeParrafos poema = length (separaParrafos (lines poema))

separaParrafos :: [String] -> [[String]]
separaParrafos [] = []
separaParrafos xs = parrafo : (separaParrafos $ dropWhile null rest)
 where
  (parrafo, rest) = break null xs

data TipoPoema = Pareado | Terceto | Cuarteto | Redondilla | Serventesio | Cuarteta | CuadernaVia | Quinteto | Quintilla | OctavaReal | Décima | Soneto deriving Show
clasificarPoema :: String -> Maybe TipoPoema
clasificarPoema poema
 | numVersos == 2 && todosVersos11Silabas && modoDeRima versos == Just "Rima AA"= Just Pareado
 | numVersos == 2 && todosVersos8Silabas && modoDeRima versos == Just "Rima aa"= Just Pareado
 | numVersos == 3 && todosVersos11Silabas && modoDeRima versos == Just "Rima ABA" = Just Terceto
 | numVersos == 4 && todosVersos11Silabas && modoDeRima versos == Just "Rima ABBA" = Just Cuarteto
 | numVersos == 4 && todosVersos8Silabas && modoDeRima versos == Just "Rima abba"= Just Redondilla
 | numVersos == 4 && todosVersos11Silabas && modoDeRima versos == Just "Rima ABAB" = Just Serventesio
 | numVersos == 4 && todosVersos8Silabas && modoDeRima versos == Just "Rima abab" = Just Cuarteta
 | numVersos == 4 && todosVersos14Silabas && modoDeRima versos == Just "Rima AAAA"= Just CuadernaVia
 | numVersos == 5 && todosVersos11Silabas && modoDeRima versos == Just "Rima ABABA"= Just Quinteto
 | numVersos == 5 && todosVersos8Silabas && modoDeRima versos == Just "Rima ababa"= Just Quintilla
 | numVersos == 8 && todosVersos11Silabas = Just OctavaReal
 | numVersos == 10 && todosVersos8Silabas = Just Décima
 | numVersos == 14 && todosVersos11Silabas = Just Soneto
 | otherwise = Nothing
 where
  versos = lines poema
  numVersos = length versos
  todosVersos11Silabas = all (\verso -> contarVocales verso == 11) versos
  todosVersos8Silabas = all (\verso -> contarVocales verso == 8) versos
  todosVersos14Silabas = all (\verso -> contarVocales verso == 14) versos

clasificarParrafos :: String -> [Maybe TipoPoema]
clasificarParrafos poema = map (clasificarPoema . unlines) (separaParrafos (lines poema))
```

La función clasificarPoema se encarga de clasificar un poema en un tipo de poema específico, según ciertas características definidas. El resultado es un valor de tipo Maybe TipoPoema,

donde Just representa el tipo de poema identificado y Nothing indica que el poema no se pudo clasificar en ningún tipo conocido.

La función utiliza varias funciones auxiliares para realizar las comprobaciones necesarias:

numeroDeVersos calcula el número de versos en un poema.

todosVersos11Silabas verifica si todos los versos del poema tienen 11 sílabas.

todosVersos8Silabas verifica si todos los versos del poema tienen 8 sílabas.

todosVersos14Silabas verifica si todos los versos del poema tienen 14 sílabas.

modoDeRima determina el modo de rima presente en un conjunto de versos.

separaParrafos divide el poema en párrafos, considerando que un párrafo está delimitado por líneas vacías. Retorna una lista de listas de versos, donde cada sublista representa un párrafo.

clasificarParrafos clasifica cada párrafo del poema en un tipo de poema utilizando la función clasificarPoema.

unlines une las líneas de un párrafo separadas por saltos de línea para formar un poema en formato de texto.

- **esConsonante**

```
esConsonante :: [String] -> Maybe String
esConsonante versos
  | any (\rima -> rima == Just "Rima ABBA" || rima == Just "Rima ABAB" || rima == Just "Rima AAAA"
          || rima == Just "Rima AA" || rima == Just "Rima aa"
          || rima == Just "Rima abab" || rima == Just "Rima abba" || rima == Just "Rima ABA" || rima == Just "Rima ABABA" || rima == Just "Rima ababa"
          || rima == Just "Rima ABABABCC" || rima == Just "Rima abbaaccddc" ) result
  = Just "Rima Consonante"
  | otherwise = Nothing
  where result = modosDeRima versos
```

La función esConsonante toma una lista de versos como entrada y devuelve un Maybe String para determinar si el patrón de rima del poema coincide con uno de los patrones de rima consonante especificados. Para hacer esto, utiliza la función modosDeRima para determinar el patrón de rima de los versos dados y luego verifica si este patrón de rima está entre los patrones de rima consonante predeterminados.

Si encuentra una coincidencia, la función devuelve Just "Rima Consonante". De lo contrario, devuelve Nothing, lo que indica que el poema no sigue uno de los patrones de rima consonante reconocidos.

- **Main**

```
main :: IO ()
main = do
 let poema1 =[ "Un soneto me manda hacer Violante"
             , "que en mi vida me he visto en tanto aprieto"
             , "catorce versos dicen que es soneto"
             , "burla burlando van los tres delante"
             , ""
             , "Yo pensé que no hallara consonante"
             , "y estoy a la mitad de otro cuarteto"
             , "mas si me veore en el primer terceto"
             , "no hay cosa en los cuartetos que me espante"
             , ""]
 let numeroVersos = numeroDeVersos poema1
 putStrLn $ "El poema tiene: " ++ show numeroVersos ++ " versos"

 case tipoVerso (filter (/= ',') (head poema1)) of
  Just tipo -> putStrLn $ "El poema está escrito en: " ++ show tipo
  Nothing -> return ()

 let poemaSinComas = map (filter (/= ',')) poema1

 putStrLn "La clasificación de los párrafos es:"
 mapM_ (\parrafo -> case parrafo of
  Just p -> putStrLn $ "- " ++ show p
  Nothing -> return ()) (clasificarParrafos (unlines poemaSinComas))

 let poemaSinComas2 = map (filter (/= ',')) poema1
 let poemaSinLineasVacias = filter (not . null) poemaSinComas2

 case clasificarPoema (unlines poemaSinLineasVacias) of
  Just estructura -> putStrLn $ "La estructura del poema es: " ++ show estructura
  Nothing -> return ()

 putStrLn "Sigue un esquema métrico de:"
 mapM_ (\esquema -> case esquema of
  Just e -> putStrLn $ "- " ++ show e
  Nothing -> return ()) (modosDeRima poema1)

 case esConsonante poema1 of
  Just rima -> putStrLn $ "La rima es: " ++ rima
  Nothing -> return ()
```

El main es la función principal del programa. Aquí se realiza la ejecución y se imprime el resultado de las funciones que analizan y clasifican un poema.

Se define un poema (poema1) como una lista de versos.

Se crea una nueva lista de versos (poemaSinComas) eliminando las comas del final de cada verso del poema original.

Se utiliza mapM_ para recorrer la lista de clasificaciones de párrafos (clasificarParrafos (unlines poemaSinComas))

Se crea otra lista de versos sin comas y líneas vacías (poemaSinLineasVacias).

Si se obtiene un resultado (Just estructura), se imprime la estructura del poema. Si el resultado es Nothing, no se imprime nada. En general, cuando no se puede determinar un resultado válido para alguna clasificación o estructura, se utiliza return () para no imprimir nada y pasar al siguiente paso.

Consultas / Pruebas

A lo largo de la creación del programa he ido realizando una serie de pruebas para comprobar el funcionamiento de cada función. Probaremos las funciones principales explicadas anteriormente. Utilizaremos el siguiente fragmento de poema para ir probando las funciones:

```
 "Un soneto me manda hacer Violante"
, "que en mi vida me he visto en tanto aprieto"
, "catorce versos dicen que es soneto"
, "burla burlando van los tres delante"
,""
,"Yo pensé que no hallara consonante"
,"y estoy a la mitad de otro cuarteto"
,"más si me vezo en el primer terceto"
,"no hay cosa en los cuartetos que me espante"
,""
```

- **contarVocales**

```
ghci> contarVocales "Un soneto me manda hacer Violante"
11
ghci> contarVocales "que en mi vida me he visto en tanto aprieto"
11
ghci> contarVocales "no hay cosa en los cuartetos que me espante"
```

- **tipoRima**

```
ghci> tipoRima "Un soneto me manda hacer Violante" "que en mi vida me he visto en tanto aprieto"
"No hay rima"
ghci> tipoRima "Un soneto me manda hacer Violante" "catorce versos dicen que es soneto"
"No hay rima"
ghci> tipoRima "Un soneto me manda hacer Violante" "burla burlando van los tres delante"
"Rima consonante"
```

- **tipoVerso**

```
ghci> tipoVerso "Yo pensé que no hallara consonante"
Just "Arte Mayor (Endecasilabos)"
ghci> tipoVerso "y estoy a la mitad de otro cuarteto"
Just "Arte Mayor (Endecasilabos)"
ghci> tipoVerso "Un soneto me manda hacer Violante"
Just "Arte Mayor (Endecasilabos)"
```

- **numeroDeVersos**

```
ghci> numeroDeVersos ["Un soneto me manda hacer Violante"]
1
ghci> numeroDeVersos ["Un soneto me manda hacer Violante\n","que en mi vida me he visto en tanto aprieto\n"]
2
ghci> numeroDeVersos ["Un soneto me manda hacer Violante\n","que en mi vida me he visto en tanto aprieto\n","catorce versos dicen que es soneto\n"]
3
```

- **modoDeRima**

```
ghci> modoDeRima ["Un soneto me manda hacer Violante\n","que en mi vida me he visto en tanto aprieto\n","catorce versos dicen que es soneto\n"]
Nothing
ghci> modoDeRima ["Un soneto me manda hacer Violante\n","que en mi vida me he visto en tanto aprieto\n","catorce versos dicen que es soneto\n","burla burlando van los tres delante"]
Just "Rima ABBA"
```

- **clasificarPoema**

```
ghci> clasificarPoema "Un soneto me manda hacer Violante\nque en mi vida me he visto en tanto aprieto\ncatorce versos dicen que es soneto\nburla burlando van los tres delante"
Just Cuarteto
ghci> clasificarPoema "Yo pensé que no hallara consonante\ny estoy a la mitad de otro cuarteto\nmas si me vezo en el primer terceto\nno hay cosa en los cuartetos que me espante"
Just Cuarteto
```

- **main**

```
El poema tiene: 8 versos
El poema está escrito en: "Arte Mayor (Endecasilabos)"
La clasificación de los párrafos es:
- Cuarteto
- Cuarteto
La estructura del poema es: OctavaReal
Sigue un esquema métrico de:
- "Rima ABBA"
- "Rima ABBA"
La rima es: Rima Consonante
```

También he realizado pruebas generales con fragmentos de poemas de diversos autores importantes de la literatura española. Si desea introducir dichos fragmentos al código, debe tener cuidado con la identación y las comillas ya que podría no compilar bien debido a ella. Le dejo una imagen de cómo se vería en el main y debajo el texto que debería introducir.

- Pareado de Antonio Machado.

```
poema1 = [ "La primavera ha venido"
         , " Nadie sabe como ha sido"]
```

poema1 = ["La primavera ha venido"

, " Nadie sabe como ha sido"]

```
El poema tiene: 2 versos
El poema está escrito en: "Arte menor (Octosilabos)"
La clasificación de los párrafos es:
- Pareado
La estructura del poema es: Pareado
Sigue un esquema métrico de:
- "Rima aa"
La rima es: Rima Consonante
```

- Terceto de Garcilaso de la Vega

```
poema1 = [ "En medio del invierno está templada"
         , "el agua dulce desta clara fuente"
         , "y en el verano más que nieve helada"]
```

poema1 = ["En medio del invierno está templada"
, "el agua dulce desta clara fuente"
, "y en el verano más que nieve helada"]

```
El poema tiene: 3 versos
El poema está escrito en: "Arte Mayor (Endecasilabos)"
La clasificación de los párrafos es:
- Terceto
La estructura del poema es: Terceto
Sigue un esquema métrico de:
- "Rima ABA"
La rima es: Rima Consonante
```

- Redondilla que pertenece a poema de Miguel de Unamuno.

```
poema1 = [ "Quevedo, qué recia lidia"
         , "trabaste en tu triste España"
         , "con la entrada de su entraña"
         , "carcomida de la envidia"]
```

poema1 = ["Quevedo, qué recia lidia"
, "trabaste en tu triste España"
, "con la entrada de su entraña"
, "carcomida de la envidia"]

```
ghci> main
El poema tiene: 4 versos
El poema está escrito en: "Arte menor (Octosilabos)"
La clasificación de los párrafos es:
- Redondilla
La estructura del poema es: Redondilla
Sigue un esquema métrico de:
- "Rima abba"
La rima es: Rima Consonante
```

- Quintilla de Garcilaso de la Vega

```
poema1 = [ "Por quereros, ser perdido"
         , "pensaba, que no culpado"
         , "mas que todo lo haya sido"
         , "así me lo habéis mostrado"
         , "que lo tengo bien sabido"]
```

poema1 = ["Por quereros, ser perdido"
, "pensaba, que no culpado"
, "mas que todo lo haya sido"
, "así me lo habéis mostrado"
, "que lo tengo bien sabido"]

```
El poema tiene: 5 versos
El poema está escrito en: "Arte menor (Octosilabos)"
La clasificación de los párrafos es:
- Quintilla
La estructura del poema es: Quintilla
Sigue un esquema métrico de:
- "Rima ababa"
La rima es: Rima Consonante
```

- Soneto de Lope de Vega

```
poema1 = [ "Un soneto me manda hacer Violante"
        , "que en mi vida me he visto en tanto aprieto"
        , "catorce versos dicen que es soneto"
        , "burla burlando van los tres delante"
        ""
        ,"Yo pensé que no hallara consonante"
        ,"y estoy a la mitad de otro cuarteto"
        ,"mas si me vero en el primer terceto"
        ,"no hay cosa en los cuartetos que me espante"
        ""
        ,"Por el primer terceto voy entrando"
        ,"y parecere que entré con pie derecho"
        ,"pues fin con este verso le voy dando"
        ""
        ,"Ya estoy en el segundo y aun sospecho"
        ,"que voy a los trece versos acaban"
        ,"contad sila son catorce y está hecho"]
```

```
poema1=[ "Un soneto me manda hacer Violante"
        , "que en mi vida me he visto en tanto aprieto"
        , "catorce versos dicen que es soneto"
        , "burla burlando van los tres delante"
        , ""
        , "Yo pensé que no hallara consonante"
        , "y estoy a la mitad de otro cuarteto"
        , "mas si te vero en el primer tercerto"
        , "no hay cosa en los cuartetos que me espante"
        ,""
        , "Por el primer terceto voy entrando"
        , "y parecere que entré con pie derecho"
        , "pues fin con este verso le voy dando"
        , ""
        ,"Ya estoy en el segundo y aun sospecho"
        ,"que voy a los trece versos acaban"
        ,"contad sila son catorce y está hecho"]
```

```
ghci> main
El poema tiene: 14 versos
El poema está escrito en: "Arte Mayor (Endecasilabos)"
La clasificación de los párrafos es:
- Cuarteto
- Cuarteto
- Terceto
- Terceto
La estructura del poema es: Soneto
Sigue un esquema métrico de:
- "Rima ABBA"
- "Rima ABBA"
- "Rima ABA"
- "Rima ABA"
La rima es: Rima Consonante
```

- Octava Real de Francisco de Quevedo

```
poema1 = [ "El firmamento duplicado en flores"
         , "se ve en constelaciones olorosas;"
         , "ni mustias envejecen con calores,"
         , "ni caducan con nieves rigurosas;"
         , "naturaleza admira en las labores;"
         , "con respeto anda el aire entre las rosas:"
         , "que solo toca en ellas, manso, viento"
         , "lo que basta a robarlas el aliento"]
```

poema1 = ["El firmamento duplicado en flores"

, "se ve en constelaciones olorosas;"

, "ni mustias envejecen con calores,"

, "ni caducan con nieves rigurosas;"

, "naturaleza admira en las labores;"

, "con respeto anda el aire entre las rosas:"

, "que solo toca en ellas, manso, viento"

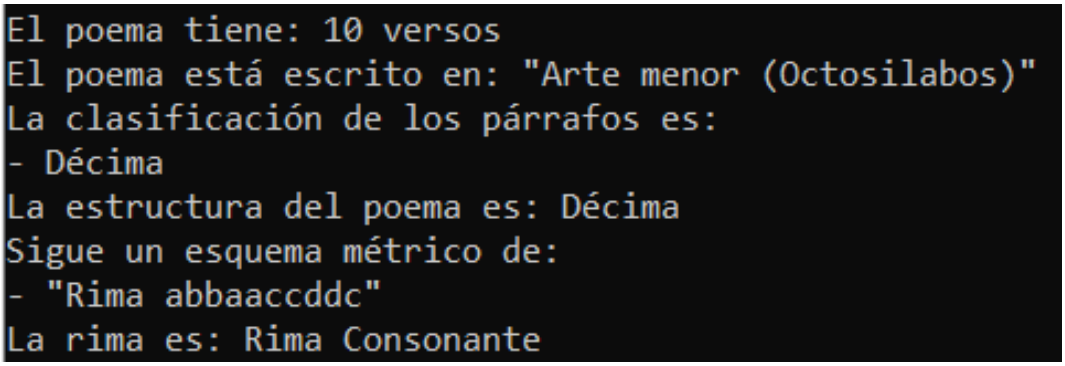
, "lo que basta a robarlas el aliento"]

```
El poema tiene: 8 versos
El poema está escrito en: "Arte Mayor (Endecasilabos)"
La clasificación de los párrafos es:
- OctavaReal
La estructura del poema es: OctavaReal
Sigue un esquema métrico de:
- "Rima ABABABCC"
La rima es: Rima Consonante
```

- Décima Xavier Villaurrutia

```
poema1 = [ "Qué prueba de la existencia"
         , "habrá mayor que la suerte"
         , "de estar viviendo sin verte"
         , "y muriendose en tu presencia"
         , "Esta lúcida conciencia"
         , "de amar a lo menos visto"
         , "y de esperarse lo imprevisto"
         , "este caerse mal sin llegar"
         , "es esa angustia de pensar"
         , "que puesto que muero existo"]
```


```
poema1 = [ "Qué prueba de la existencia"
         , "habrá mayor que la suerte"
         , "de estar viviendo sin verte"
         , "y muriendose en tu presencia"
         , "Esta lúcida conciencia"
         , "de amar a lo menos visto"
         , "y de esperarse lo imprevisto"
         , "este caerse mal sin llegar"
         , "es esa angustia de pensar"
         , "que puesto que muero existo"]
```



Limitaciones

Crear un poema en Haskell, como en cualquier otro lenguaje de programación, está sujeto a ciertas limitaciones. Estas limitaciones no están relacionadas con la creatividad poética , sino más bien con la forma en que se pueden manipular y analizar los textos a través de la programación. Algunas de estas limitaciones del programa y que me han ido surgiendo son:

1) **Reconocimiento de la métrica**: El reconocimiento de la métrica y la rima de un poema ha sido un proceso bastante complejo, ya que las palabras pueden rimar de formas que no son obvias para un algoritmo común (por ejemplo, rimas imperfectas, rimas internas, etc.). Esto significa que tienes contemplar los casos más relevantes o comunes, pero no puedes abordar todos los de la literatura española.

2) **Ambigüedad en la interpretación**: La poesía a menudo juega con la ambigüedad y las múltiples interpretaciones posibles de las palabras y frases. Un programa informático, en cambio, se basa en la lógica y en reglas claramente definidas, lo que puede limitar su capacidad para comprender e interpretar correctamente un poema. Si desea probar algún ejemplo poco común, el programa no fallará, pero probablemente no sea capaz de realizar el estudio completo.

3) **Procesamiento del lenguaje natural**: Haskell, como otros lenguajes de programación, no tiene la capacidad de entender el lenguaje humano. Para tareas como el análisis del poema, he tenido que recurrir a bibliotecas de procesamiento del lenguaje.

4) **Excepciones en la literatura:** A la hora de crear el tipo de palabra , los tipos de rima , etc, me he basado en las reglas clásicas y generales de la RAE , sin tener en cuenta todas las excepciones que ésta recoge. El tratamiento de algunas palabras con hiatos o diptongos no se contempla en todos los casos luego esto podría generar algún resultado no esperado.