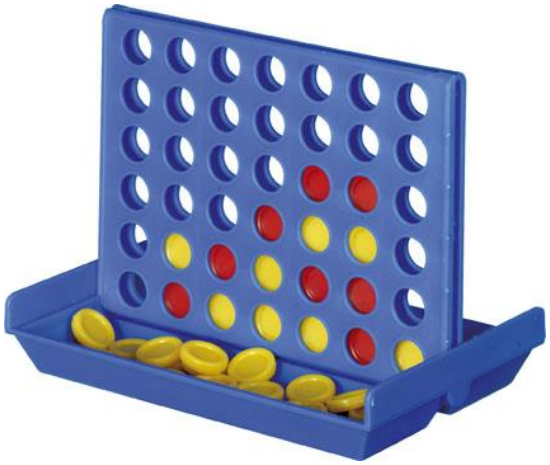
4 EN RAYA

PRÁCTICA DE LENGUAJES DE PROGRAMACIÓN



Iván Carrasco Alonso

DNI: 70907531-A

Lenguajes de programación

2º CURSO DE GIISI

EPSZ

UNIVERSIDAD DE SALAMANCA

Escuela politécnica superior de Zamora

VNiVERSiDAD D SALAMANCA

Contenido

EXPLICACIÓN DE LOS IMPORTS

import System.Environment

La importación de System.Environment se debe a la utilización de funciones de entrada y salida como:

- ***getArgs :: IO [String]***→ Permite la introducción de argumentos en el main.

import Data.List

La importación de Data.List se debe a la utilización de:

- ***transpose :: [[a]] -> [[a]]*** → Permite realizar la matriz transpuesta de una matriz es decir, devuelve una matriz en la cual las filas de la matriz introducida serán columnas, y las columnas serán ahora filas.

import Text.Printf

La importación de Text.Printf se debe a la utilización de:

- ***printf :: PrintfType r => String -> r*** → Permite imprimir por pantalla diversos tipos con saltos de línea y tabulaciones.

Las demás funciones utilizadas se encuentran dentro de las funciones básicas del conjunto de funciones de **Prelude.**

EXPLICACIÓN DE LAS FUNCIONES

Creación de la matriz

- ***crear_columna :: (Eq a, Num a, Num t) => a -> [t]***

  Esta función crea una lista de elementos N los cuales conformarán una columna de la matriz del tablero. Crea una columna de elementos N, los cuales son ceros.

- ***crear_tablero :: (Eq a, Eq a1, Num a, Num a1, Num t) => a1 -> a -> [[t]]***

  Esta función crea una lista de listas, es decir, creará una lista de M elementos por cada columna y N elementos de listas “columna”, conformando así la matriz.

Dibujo del tablero de juego

- ***pinta :: Show a => [a] -> Int -> Int -> IO ()***

  Esta función pinta el tablero, pasándole una lista el número de filas, y la fila en la que comienza a pintar, pero lo hace pintando por cada fila, una columna, por lo que se necesita pasar la matriz transpuesta a esta función para que nos dibuje el tablero correctamente.

Inserción de fichas

- ***inserta_previo :: (Eq a, Num a) => [a] -> t -> a -> [a]***

  Esta función inserta un elemento x en una lista fijándose en el elemento de mayor índice con valor x, y sustituyéndolo por el elemento num en la lista

- ***inserta :: (Eq a, Eq a1, Num a, Num a1) => [[a1]] -> a -> a -> a1 -> [[a1]]***

  Esta función utiliza la anterior para eliminar un elemento con valor 0 e insertar el nuevo, y a continuación unirlo con la cola de la función.

Cálculo del ganador

Cálculo de los valores verticales

- ***calcula_vertical:: (Eq a2, Num a, Num a1, Ord a1) => [[a2]] -> a2 -> a1 -> Int -> Int -> a***

  Esta función comienza con una rep=0, y va analizando cada columna, aumentando rep en 1 si existen valores repetidos iguales a num en el momento en el que exista uno que no se repite rep =0 también tiene un contador i el cual cuando llega a índice m cambia de columna analizando asi todas. En el momento en el que rep>=4 la función devuelve valor 1 y si llega al final de la lista [] entonces devuelve valor 0, ya que no ha encontrado 4 valores num repetidos y seguidos.

Cálculo de los valores horizontales

- ***calcula_horizontal:: (Eq a1, Num a) => [[a1]] -> a1 -> t -> Int -> t1 -> a***

  Esta función utiliza la función calcula vertical ya que si hacemos la transpuesta de la matriz y utilizamos esta función, nos devolverá si existen cuatro valores iguales seguidos en horizontal.

Cálculo de los valores diagonales

- ***diagonales :: [[t]] -> [[t]]***

  Esta función devuelve el conjunto de las diagonales de suroeste a noreste

- ***diagonales_totales :: [[t]] -> [[t]]***

  Esta función obtiene el total de las diagonales de una matriz, para ello utiliza la función anterior juntando la lista obtenida de diagonales, más la que se obtiene de hacer un reverse de la lista y obtener diagonales de esta, así se obtiene la lista completa en ambas direcciones de la matriz, noroeste a sureste, y suroeste a noreste.

- ***calcula_diagonales :: (Eq a2, Num a, Num a1, Ord a1) => [[a2]] -> a2 -> a1 -> Int -> Int -> a***

  Para el cálculo de diagonales de la matriz se le debe pasar como lista el resultado obtenido de la función diagonales_totales, así examinará cada lista de las obtenidas comprobando si existen 4 elementos iguales seguidos en cada lista "diagonal" que conforman la lista de listas, si la i supera el tamaño total de la lista es que se ha llegado al final por lo que se devuelve 0, si j supera a la longitud de xss!!i entonces hay que cambiar de lista de "diagonal" y se hace una recursiva con la siguiente, si no ocurre esto se comprueba si existen elementos seguidos aumentando en 1 rep y haciendo una recursiva con j+1 para acceder al siguiente elemento de la lista de "listas de diagonales" si es así, y en el momento en que exista un elemento distinto rep se vuelve 0

- ***calcula :: (Eq a, Num a1) => [[a]] -> a -> Int -> Int -> a1***

  Esta función comprueba si alguna de las funciones de cálculo de diagonales, horizontal o vertical ha devuelto 1 si es así esta función devolverá 1 sino devolverá 0.

- ***empate :: (Eq a1, Num a, Num a1) => [[a1]] -> a***

  Esta función comprueba si existe algún 0 en la fila más superior, puesto que si no existe existirá un empate devolviendo 1 y si existe no devolviendo 0.

Cálculos de la máquina

- ***calculo_maquina:: (Eq a, Num a) => [[a]] -> Int -> Int -> Int -> Int -> Int***

  Esta función en el caso en el que i no valga n, comprueba si en el elemento xss!!i!!0 ==0 es decir, si es posible insertar algo porque la lista no está llena, y si además la inserción de una ficha por el adversario, provoca una jugada ganadora, si esto ocurre, devuelve el valor de la columna, para que la máquina inserta allí, sino ocure esto, sigue examinando columnas hasta que una cumpla estas dos condiciones, o se llegue al final de la lista de listas, en donde la i==n, si ocurre esto, utilizaríamos busca_central, para poner una ficha en el elemento más al centro posible.

- ***busca_central:: (Eq a, Eq a1, Num a, Num a1) => [[a1]] -> Int -> Int -> Int -> a -> Int***

  Esta función busca si en el elemento más central (intro) puede insertar una ficha, si no es así debe optar por los valores más próximos a izquierda o derecha del mismo, si en estos no se puede, entonces debe añadirse a estos valores una unidad más y así sucesivamente hasta que se encuentre un hueco.

Turnos de los jugadores

Hombre contra hombre

- **hombrevshombre :: [[Int]] -> Int -> Int -> Int -> IO ()**

  Esta función cede los turnos a cada jugador y finaliza diciendo quien ha ganado o si han empatado. Lo primero que hace es comprobar si existe un empate si existe finaliza la partida, sino existe pasamos a atendiendo a la variable num, tendremos el turno del jugador 1 o del jugador2 , en el turno de cada jugador se le pedirá que diga una columna haciendo las comprobaciones pertinentes, y se insertara para después comprobar si tiene una jugada ganadora, si la tiene finaliza la partida y gana el jugador num si no, se realiza la llamada recursiva pasándole el parámetro num distinto , y la lista ya modificada por la inserción.

Hombre contra maquina

- ***hombrevsmaquina :: [[Int]] -> Int -> Int -> Int -> IO ()***

  En primer lugar se comprueba que en tablero no exista un empate, si existe, se indica y se finaliza la partida, sino atendiendo a el parámetro num le tocaría a la maquina o al jugador, en el turno del jugador, se le pide que ingrese una columna, la ingresa y si

está llena se le pide que ingrese otra sino se inserta y muestra el tablero, después se hacen las comprobaciones para saber si ha ganado, y si ha ganado entonces se muestra finalizando la partida, sino tocaría el turno de la máquina haciendo una llamada recursiva con el tablero ya modificado.

En el turno de la máquina primero se calcula el valor de intro que será el valor de la mitad de su tamaño es decir, el elemento más, después se llama a la función calculo máquna, la cual devolverá la columna en la que debe insertar, se pintará el tablero, y después se comprueba si ha ganado la máquina, si ha ganado se finaliza la partida, mostrando un mensaje, si no se cambia num y se realiza la llamada recursiva con el tablero modificado para así realizar el turno del hombre.

Funciones auxiliares

- ***quitaultima :: Eq t => t -> [t] -> [t]***

  Esta función quita el último elemento de la lista que tenga un elemento x, recorre la lista contando la cantidad de elementos x que existen y cuando solo se repita una vez en la lista ls y l de (l:ls) sea igual a ese x buscado, se devuelve ls sino [l]++quitaultima x ls.

- ***quitaultima2 :: Eq t => t -> [t] -> t -> [t]***
  Esta función quita el último elemento de la lista que tenga un elemento x y a este le añade [num]++ls en vez de l. Para ello, recorre la lista contando la cantidad de elementos x que existen y cuando solo se repita una vez en la lista ls y l de (l:ls) sea igual a ese x buscado, se devuelve [num]++ls sino [l]++quitaultima x ls.

- ***serepite :: (Eq a1, Num a) => a1 -> [a1] -> a***

  Esta función cuenta las veces que se repite un elemento en la lista, si el elemento l de (l:ls) coincide con el x que queremos contar, se devuelve (i+1) donde “i =serepite x ls” sino únicamente devuelve i.

- ***atoi :: String -> Int***

  Esta función lee un String y lo transforma en el tipo Int.

Función principal

- ***main :: IO ()***

  Esta función lee a través de la entrada de argumentos dos, a continuación crea el tablero dado por las dimensiones señaladas, m x n, si no se ajustan a n>=7 y m>=6 entonces muestra un error, si se ajustan, atendiendo a el número de argumentos introducidos, si se introducen más de dos el juego será de la maquina contra la persona y si se introducen 2 será una persona contra otra persona, llamándose en los dos casos a las funciones pertinentes.