Ejercicio 2

Recorrido de matrices

Rodrigo Burdet

Para la resolución de este ejercicio se usa una matriz dinámica, cuyas dimensiones son tomadas de un archivo pasado como parámetro, o en caso de no existir dicho archivo, desde STDIN. Se reconocieron 3 tipos distintos de TDAs, ellos son:

Without specifying width for last column:

TDA	Header (.h)	Source (.c)
Matriz	myMatrix.h	myMatrix.c
Parseador	fileParser.h	fileParser.c
Lista	myList.h	myList.c

La implementación de lista se hizo usando otro TDA:

TDA	Header (.h)	Source (.c)
Nodo	myNode.h	myNode.c

La lista contiene un puntero al primer nodo y un puntero al nodo "actual" (llamado así para realizar operaciones más fácilmente). El parseador contiene las dimensiones de la matriz sacadas desde el archivo, un arreglo (también dinámico) para obtener los posibles movimientos desde el archivo y el tamaño del mismo para saber cuándo dejar de moverse sobre la matriz. En este ejercicio, al tener en cada línea del archivo, distintos tamaños de matriz, se optó por crear la matriz, hacer el procesamiento necesario y destruirla al final de la línea. El proceso se repite para cada línea del archivo. Se eligió hacerlo de esta forma porque la probabilidad de que dos matrices sean del mismo tamaño (o parecido) son chicas, lo cual "reallocar" memoria no me parecio meritorio.

Los archivos de entrada tiene formato aXb siendo a y b numeros naturales. Me encontré con el problema que el archivo de entrada `in_all_errors` el formato era axb, por lo que opté por aceptar una cadena de la forma $a$caracteres$b$, con cualquier valor de carácter posible. Al recorrer la matriz para imprimir los movimientos se optó porque la matriz cree nodos y estos sean agregados por la lista de movimientos, si bien esta no es la solución más "bonita" (en términos de la abstracción de datos) no quise hacer una función o un TDA que sólo se encargue de manejar otros TDAs.

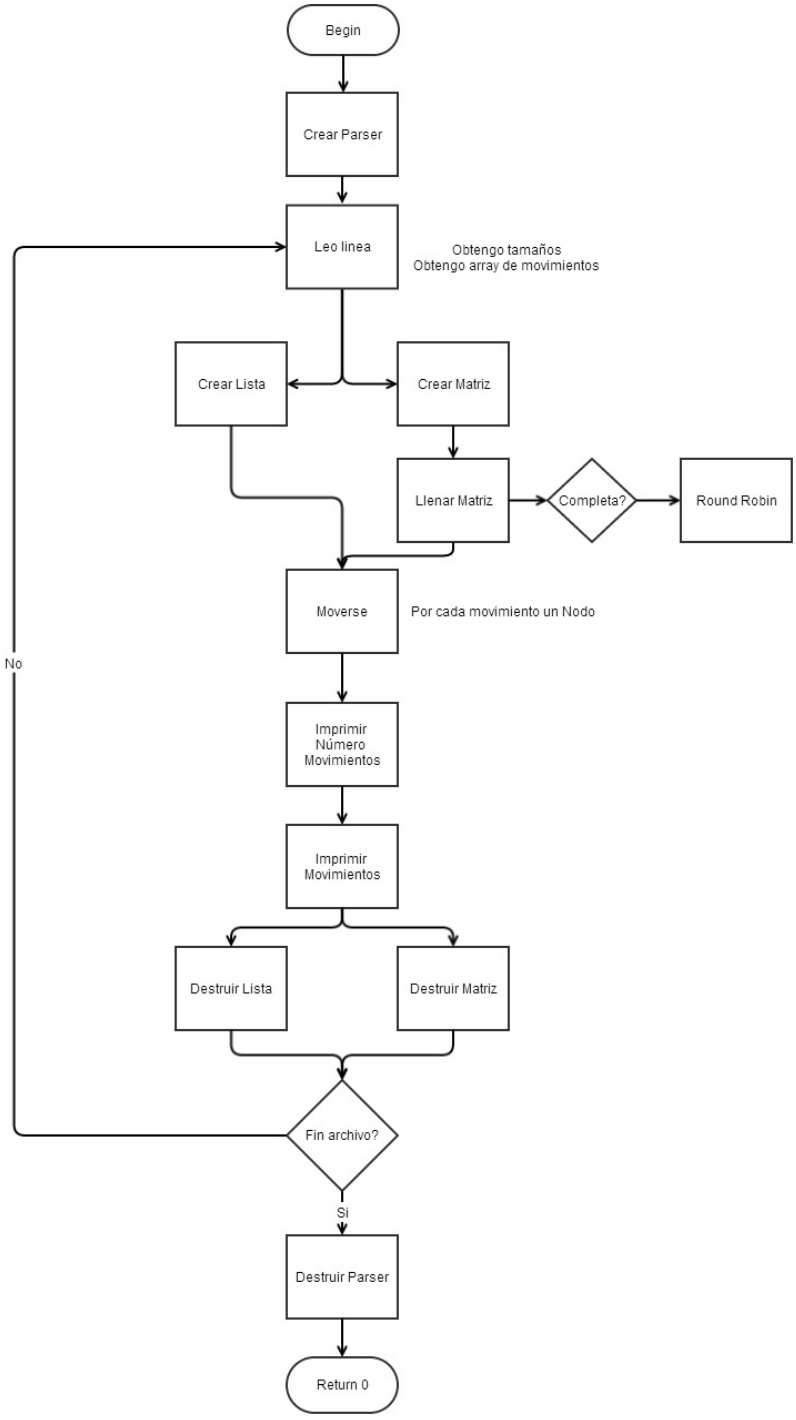

Figure 1: Diagrama de flujo