Lectura del tablero de juego

El juego se desarrolla en un tablero de 8*8 casillas. Las casillas pueden estar vacías o contener un *muro* de piedra, una pared de *hielo*, una *caja*, una *joya* o una *tortuga*. Las tortugas se desplazarán por el tablero hasta llegar a una joya. Para poderse desplazar la tortuga lleva asociada una dirección que indica hacia que parte del tablero se dirige.

En este problema leeremos los datos de un tablero de la entrada y mostraremos su representación por pantalla. Cada entrada consta de varios tableros, cada uno para un número diferente de tortugas. Hemos de encontrar el tablero correspondiente al número de jugadores que desean jugar, guardarlo en memoria y despues mostrarlo por pantalla. La representación de los elementos del tablero en la entrada y salida es diferente. En la entrada cada elemento se representa mediante un caracter. En la salida cada casilla del tablero se representa con dos caracteres.

Para realizar los problemas del juez asociados con la práctica tres se proporciona en el campus virtual para cada problema una plantilla y los datos de entrada del ejemplo. La plantilla proporciona la lectura de los datos de entrada y la salida que pide el juez. Solo tenéis que realizar funciones que luego utilizaréis en la práctica. El fichero con los datos de entrada lo podéis utilizar como entrada al programa sin necesidad de teclear los datos en cada prueba, utilizando el redireccionamiento de la entrada estandar `cin` a un fichero. Este redireccionamiento está implementado en la plantilla. Solo hay que nombrar al fichero como `datos1.txt`.

Se recomienda comentar bien las funciones que hagáis para que luego las podáis utilizar sin problema en la práctica.

Para implementar el problema:

1. Declarar constantes para la dimensión del tablero y para el número de tipos de casillas.
2. Declarar un tipo enumerado `tEstadoCasilla` con los posibles valores de cada casilla: `VACIA`, `HIELO`, `MURO`, `CAJA`, `JOYA`, `TORTUGA`.
3. Declarar un tipo enumerado `tDir` que permita diferenciar la dirección en que se mueven las tortugas. Sus valores serán `UP`, `DOWN`, `RIGHT` y `LEFT`.
4. Declarar un `struct`, `tTortuga`, con información sobre la tortuga. La información es un identificador numérico y la dirección en la que se mueve.
5. Declarar un `struct`, `tCasilla` con información sobre cada casilla, debe tener en un campo el contenido de la casilla y en otro la información sobre la tortuga. El segundo campo solo contendrá información cuando el valor del primer campo sea `TORTUGA`.
6. Declarar un tablero como una matriz de casillas.
7. Implementar funciones para:
   (a) Transformar un carácter en una casilla. La función recibirá también un parámetro con el identificador de la tortuga. Este parámetro solo llevará valor cuando el carácter corresponda a una tortuga.
   (b) Transformar una casilla en una cadena de caracteres.
   (c) Mostrar por pantalla o fichero una casilla.
   (d) Mostrar por pantalla o fichero un tablero.
   (e) Cargar un tablero a partir de unos datos dados por teclado o fichero.
8. Vamos a utilizar el tipo `stringstream` para leer todos los datos de entrada con la función getline y evitar el uso del operador de extracción. Este tipo se encuentra definido en la librería `sstream`. Se comporta de forma parecida a un buffer de datos y lo podremos manejar con los operadores de extracción e inserción. Para usarlo:
   (a) Declarar una variable de tipo `string` y leer en ella una línea de datos utilizando la función `getline`.

(b) Declarar una variable de tipo `stringstream` y copiar en ella el contenido de la cadena de caracteres leida: `ss << linea`.

(c) Extraer la información contenida en la línea utilizando el operador `>>`.

(d) Limpiar la variable antes de copiar otra línea con la función : `ss.clear()`.

(e) La lectura de los datos de la función `resuelveCaso` de la plantilla se ha realizado utilizando este tipo.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso comienza con un valor que indica el número de jugadores que quieren jugar. A continuación aparece la descripción de varios tableros, cada uno comienza con una línea en que se indica el número de tortugas que tiene. Después del número de tortugas aparecen 8 líneas con 8 valores cada una. Cada valor es un posible elemento del tablero. Cada caso acaba con una línea con tres guiones. El último caso se marca con un número de jugadores cero.

El tablero es de 8*8. El número de jugadores y de tortugas es mayor que cero y menor que 5. En todos los casos se garantiza que existe un tablero con un número de tortugas igual al número de jugadores. Los caracteres que representan cada elemento del juego son: `'#'`, es un muro de piedra, `'@'`, es una pared de hielo, `' '`, es una casilla vacía, `'$'`, es una joya, y `'C'` es una caja. Para representar una tortuga aparece una letra que indica la dirección en la que mira: `'U'`, `'D'`, `'R'`, `'L'` (UP, DOWN, RIGHT, LEFT).

Salida

Para cada caso de prueba se dibuja el tablero con el número de jugadores pedido. Cada casilla se dibuja con dos caracteres. Se utilizan los siguientes símbolos para visualizar los elementos: `"||"`, para los muros de piedra, `"**"`, para los muros de hielo, `"[]"`, para las cajas, `"00"`, para las joyas; y para las tortugas que miran en las cuatro direcciones posibles usa: `">>"`, `"<<"`, `"^^"`, `"vv"`

Entrada de ejemplo

```
4
1
DC    CC
CC    CC
 #@@@@#
  @  @
 #@ $@#
  @@@@
CC    CC
CC    CC
2
DC    CD
CC    CC
 #@@@@#
  @$ @
 #@ $@#
  @@@@
CC    CC
CC    CC
```

```
3
DC    CD
CC    CC
 #@@@@#
  @$ @
 #@$$@#
  @@@@
CC    CC
RC    CC
4
DC    CD
CC    CC
 #@@@@#
  @$$@
 #@$$@#
  @@@@
CC    CC
RC    CU
---
2
4
DC    CD
CC    CC
 #@@@@#
  @$$@
 #@$$@#
  @@@@
CC    CC
RC    CU
2
DC    CD
CC    CC
 #@@@@#
  @$ @
 #@ $@#
  @@@@
CC    CC
CC    CC
1
DC    CC
CC    CC
 #@@@@#
  @  @
 #@ $@#
  @@@@
CC    CC
CC    CC
---
0
```

Salida de ejemplo

```
vv[]        []vv
[][]        [][]
  ||********||
    **0000**
  ||**0000**||
    ********
[][]        [][]
>>[]        []^^

vv[]        []vv
[][]        [][]
  ||********||
    **00  **
  ||**  00**||
    ********
[][]        [][]
[][]        [][]
```

Autor: Isabel Pita.