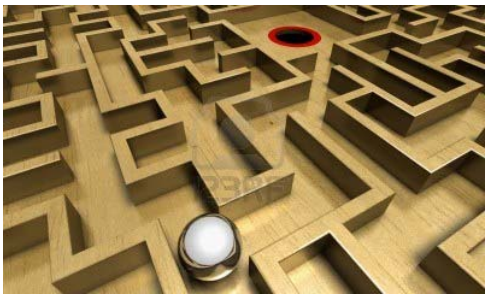
Práctica 3
Laberinto de bolas

Fecha de entrega: 7 de abril de 2013

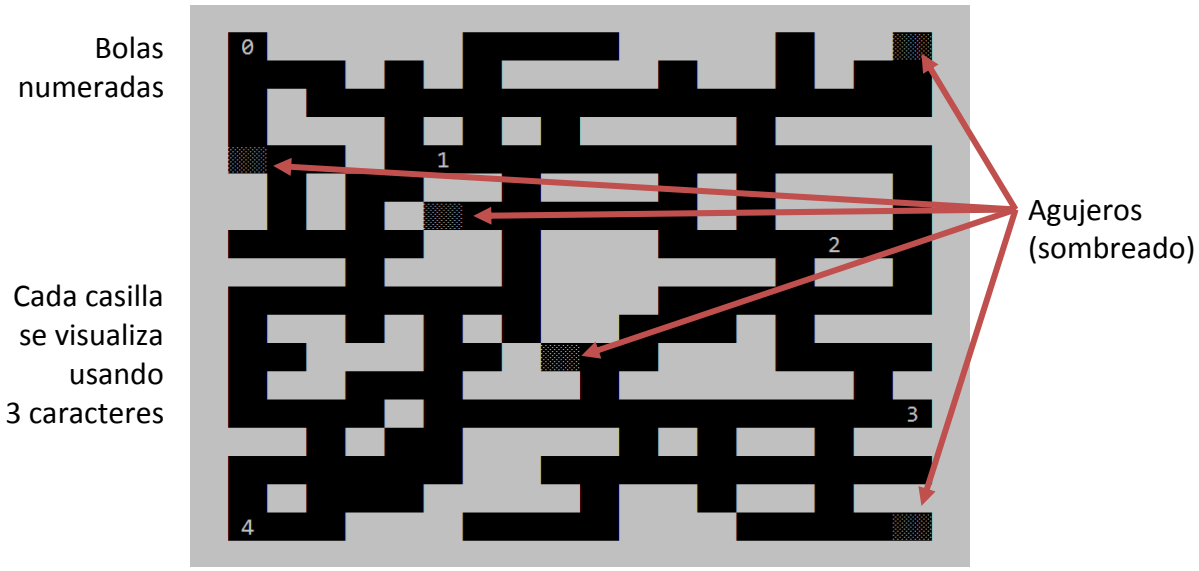


(Fuente: es.123rf.com)

1. Descripción de la práctica

En esta práctica implementaremos un juego que denominaremos *laberinto de bolas*. Tenemos un tablero con forma de laberinto en el que hay algunos agujeros y sobre el que se desplazan unas bolas. Se trata de inclinar el tablero sucesivamente hacia sus lados hasta conseguir que una de las bolas se detenga encima de un agujero.

Nosotros representaremos el tablero en la consola de esta forma:



El tablero está formado por una rejilla de DIM × DIM casillas, que pueden clasificarse en cuatro categorías: *libre*, *bola*, *pared* o *agujero*. En cada *turno* el jugador inclina el tablero en una de estas cuatro direcciones: *derecha*, *arriba*, *izquierda* o *abajo*. Al inclinar el tablero todas las bolas caen en línea recta,

siguiendo la correspondiente dirección, hasta que una pared u otra bola las detiene; es decir, pasan por encima de las casillas libres y de los agujeros. Si después de un turno alguna de las bolas ha quedado justo encima de un agujero, el juego ha terminado. El número de turnos empleados es entonces la puntuación obtenida por el jugador.

Se puede encontrar una versión más sofisticada de este juego en:
http://juegosdelogica.net/juegosdeestrategia/laberinto.php

2. Implementación de la práctica

2.1. Programa principal

El programa debe comenzar pidiendo el nombre del archivo que contiene la información del juego. La cargará y presentará el tablero en su situación inicial. A continuación mostrará un menú con las siguientes opciones:

```
1 - Inclinar hacia la derecha
2 - Inclinar hacia arriba
3 - Inclinar hacia la izquierda
4 - Inclinar hacia abajo
5 - Reiniciar el tablero
6 - Cargar nuevo tablero
0 - Salir
```

Si el usuario elige inclinar el tablero en la misma dirección del turno anterior, se ignorará esa solicitud. Reiniciar el tablero permitirá volver a la situación inicial, a partir del último archivo cargado en el juego. Cargar nuevo tablero permitirá cargar un nuevo tablero desde el archivo que indique el usuario.

Como de costumbre, debes incluir un subprograma `menu()` que muestre ese menú hasta que el usuario introduzca una opción válida. En el programa principal habrá un bucle que procesará las opciones que proporcione `menu()`, hasta que devuelva la opción 0.

Se proporcionará una demo de la ejecución deseada del programa.

2.2. Estructura de la información

Se ha de declarar la constante `DIM=20`. Y utilizar al menos los siguientes tipos:

- `tTablero`: para guardar la rejilla de DIM × DIM casillas que componen el tablero. En cada casilla se almacenará uno de los cuatro estados posibles (libre, bola, pared y agujero). No obstante, en este juego es importante distinguir las bolas entre sí, por ejemplo para comprobar que no se

adelantan al inclinar el tablero. Por ello, las bolas deberán numerarse. Nunca habrá más de NB bolas, así que numeraremos las bolas en el intervalo {0, 1, 2, ..., NB-1}. En definitiva, como necesitamos 3 valores más para las casillas libres, de pared y de agujero, los datos que habrá en el tablero serán enteros en el intervalo {-3, -2, -1, 0, 1, 2, ..., NB-1}. Se usarán adecuadamente las constantes `LIBRE=-1`, `PARED=-2` y `AGUJERO=-3` para codificar los datos, y `NB=10` para indicar el número máximo de bolas.

- `tDireccion`: para representar las cuatro inclinaciones que pueden aplicarse en un turno (derecha, arriba, izquierda y abajo).
- tJuego: para almacenar todos los datos que describen el estado actual del juego: tablero, número concreto de bolas que hay sobre el tablero, nombre del archivo con la situación inicial del tablero, inclinación del último turno y número de turnos que ha empleado el jugador hasta el momento.
- `tBolasMarcadas`: se trata de un tipo auxiliar que sirve para comprobar, durante la carga inicial del tablero, que cada bola aparece una única vez. Para cada bola, un `bool` que indique si ya se ha detectado su presencia.

2.3. Estructura algorítmica

El programa debe estar debidamente organizado en subprogramas. Será imprescindible tener al menos los subprogramas que se describen a continuación.

Inicialización y carga

- `inciaJuego(juego)`: para inicializar el juego.
- `cargaJuego(juego)`: para cargar el tablero del juego, y el número de bolas que tiene, desde un archivo. El nombre del archivo ya está registrado en el juego, por lo que aquí no hay que pedirlo. El formato de los archivos es el siguiente: en una primera línea aparecerá el número de bolas que hay en el tablero; después habrá DIM líneas, una por cada fila del tablero, incluyendo en cada una el estado (entero) de las DIM casillas que componen esa fila.

  Durante el proceso de carga se deben hacer las siguientes comprobaciones:

  - ✓ Que el número de bolas n del tablero inicial es válido: cumple $n \leq NB$.
  - ✓ Que los datos del tablero son válidos: están en el intervalo {-3, -2, -1, 0, 1, 2, ..., n-1}.
  - ✓ Que cada bola en el intervalo {0, 1, 2, ..., n-1} aparece una única vez en el tablero.

  Si alguna comprobación falla el programa deberá terminar, informando del motivo.

- `validarFronteras(juego)`: para comprobar si el tablero del juego está rodeado de paredes. Si no lo estuviera, el tablero sería inestable porque las bolas se podrían escapar de él. En este caso el programa debe terminar, informando del motivo.

Visualización

- `muestraLaberinto(juego)`: para mostrar en la consola el estado actual del tablero. Para visualizar los estados libre, pared y agujero, se pueden usar caracteres ASCII (p.e., con los de los códigos 176, 177, 178 y 219 para paredes y agujeros). Para las bolas, se usará el ordinal de cada una. Además del tablero, se debe mostrar el número de turnos realizados sobre la esquina superior derecha del tablero. Se puede mostrar el tablero con colores si se quiere. Para ello, basta incluir la biblioteca `Windows.h` en el programa y usar la siguiente función (se supone que el parámetro `color` es un valor entre 0 y 15):

```
void setColor(unsigned int color) {
   HANDLE handle = GetStdHandle(STD_OUTPUT_HANDLE);
   SetConsoleTextAttribute(handle, color);
}
```

Lógica del juego

- `desplazaBola(juego, fila, col, direc, terminado)`: desplaza la bola que ocupa la fila `fila` y la columna `col` en el tablero del juego `juego`. El parámetro de entrada `direc` corresponde a la inclinación que se debe aplicar. En el parámetro `terminado` se devuelve si la bola acaba deteniéndose sobre un agujero, lo que supondría el final del juego.
- `inclinaTablero(juego, direc, terminado)`: desplaza todas las bolas que hay en el tablero de `juego` siguiendo la dirección indicada en `direc`. En el parámetro `terminado` se devuelve si alguna de las bolas acaba deteniéndose sobre algún agujero, lo que supondría el final del juego.

3. Entrega de la práctica

La práctica se entregará en el Campus Virtual (tarea **Entrega de la Práctica 3**). Sólo se subirá el archivo con el código fuente (.cpp). En dicho archivo debe figurar el nombre de los miembros que integren el grupo de prácticas.

Fecha de entrega: 7 de abril de 2013