Trabajo Práctico 1: Tesoro Binario V1.0

Datos personales

Alumno: Ontiveros Emilio
DNI: 36089085
Legajo: 109048
Mail: eontiveros@fi.uba.ar
Nombre del archivo zip entregado: "109048-TP1.zip"

Versión

Entregado: 9/10/23
Ultimo cambio realizado a la documentación: 9/10/23
Repositorio del proyecto: https://github.com/Greyluck/Tesoro
(Solo la versión Linux-Eclipse esta actualizada)

Cuestionario

1) ¿Que es un Debug?:
Es la herramienta utilizada y procedimiento realizado para purgar los errores en el código. Un "bug" es un error en el código, de ahí la palabra "Debug"
2) ¿Qué es un "Breakpoint"?:
Es un punto intermedio que se setea para para detener la ejecución durante el debuging con la finalidad de poder analizar el estado actual del programa en esa precisa instrucción.
3) ¿Qué es "Step Into", "Step Over" y "Step Out"?
Son acciones que el desarrollador puede utilizar para recorrer el código durante el debuging.

- **Step into** permite acceder a función para ver el correcto procesamiento de las instrucciones dentro de ella.
- **Step over** permite "saltearse" la revisión de la instrucción actual para continuar revisando la siguiente.
- **Step out** le permite "volver" de la instrucción que se está revisando hacia aquella que se encuentra por fuera. Así como el step into se utiliza para entrar a una función, el step out se utiliza para salir de ella.

Manual de usuario

Indicaciones previas: Se asume que el usuario leyó y entendió el siguiente documento, que sus acciones son bienintencionadas y que está intentando jugar el juego de la manera que corresponde.

Tablero: El juego se realiza sobre un tablero de 20 casillas por 20 casillas numeradas del 0 al 19.
Casillas: Los jugadores ingresaran los valores de las casillas siempre colocando primero la columna, y luego la fila. (Es importante que el usuario ingrese valores enteros de 0 a 19).
Esta será la forma en que los usuarios colocaran tanto espías como tesoros.

Inicio de juego:

1. Al inicio del juego, el tablero estará vacío.
2. **Fase de colocar tesoros iniciales:** El jugador blanco comenzara colocando uno de sus tesoros en el tablero. Luego, continuara el jugador negro. Los jugadores irán alternando sucesivamente hasta haber colocado todos sus tesoros.
3. **Sucesión de turnos:** Una vez colocados los tesoros iniciales, los turnos irán ocurriendo sucesivamente hasta que alguno de los 2 jugadores haya cumplido con la condición de victoria (dejar sin tesoros al rival). Durante cada turno ocurrirán los siguientes eventos.
    - **Colocación de un espía en un casillero:** El jugador del turno actual colocara un espía en un casillero donde no aun no haya un espía propio.
        - **Enfrentamiento:** Si el jugador coloco a su espía sobre un casillero en el cual ya había un espía enemigo, ambos espías morirán liberando la casilla.
        - **Patrullaje:** Si el jugador coloco su espía en un casillero vacío, este se quedará en ese casillero patrullando.
        - **Desenterrar:** Si el jugador coloco su espía en un casillero que contiene un cofre enemigo. Se le avisara que el espía ha comenzado a cavar para desenterrar el cofre, inhabilitando la casilla para ambos jugadores por 5 turnos.
        - **Reorganizar:** Si un jugador decidió colocar su espía sobre uno de sus cofres, deberá primero mover ese cofre hacia otro casillero adyacente. Este movimiento podrá ser vertical u horizontal (no diagonal). Es importante tener en cuenta que, al mover el cofre, no se debe elegir una casilla que ya posea cofres o espías para poder colocarlo.
            - Si luego de colocar el espía el jugador no movió ningún cofre, se la dará la opción de mover cualquiera de sus cofres a una casilla adyacente valida.
    - **Recolectando el tesoro:** Al final del turno, todos los espías que estén cavando avanzar un turno hacia el tesoro. Aquellos que recolecten un tesoro, se irán a su casa dejando la casilla vacía.
    - **Condición de victoria:** Si un jugador desenterró todos los cofres de su oponente, habrá ganado y la partida habrá terminado. En caso de que aun queden cofres por desenterrar, comenzara el turno del jugador contrario.

Manual de programador

Introducción

- **IMPORTANTE:** El programa esta principalmente en inglés para su reutilización posterior, dejando en español solo los comentarios relevantes.
- A fines prácticos el programa se comporta sobrescribiendo una matriz en la que se almacenan valores enteros positivos y negativos que determinan el tipo de elemento en la casilla. Solo puede existir un elemento por casilla.
- El programa reescribe los valores en cada posición con un formato de similar a "Tablero[columna][fila]=nuevoValor".
- A cada posición se la llama celda. A cada celda se le asignan diferentes valores que representaran el tipo de celda es, por ejemplo "Espía" o "Tesoro".
- En el programa algunas secciones que deben ser revisadas o pueden ser mejoradas llevan la indicación "TODO". Eso permite su rápida búsqueda. En general algunos entornos de desarrollo lo marcan con un color diferente para que sea fácil de encontrar.

Partes del programa

El programa se encuentra divido en:

- "Main" donde ira el flujo general del programa.
- 2 clases principales.
    - "Board" para el Tablero
    - "Coordinate" para manejar la devolución de coordenadas
- En los archivos.h están las declaraciones de las funciones y variables, mientras que en los archivos.cpp las implementaciones de cada función y el seteo de las variables.

Clase "Coordinate" (Coordenada)

En esta clase se utiliza principalmente para definir la función **askForCoordinate()** la cual solicita al usuario ingresar los valores horizontales y verticales para identificar filas y columnas y verifica que sean válidos. A lo largo del programa se la utiliza siempre que se requiere que el usuario ingrese información nueva por lo que es una de las funciones principales a entender dentro del código.

Clase "Board" (Tablero)

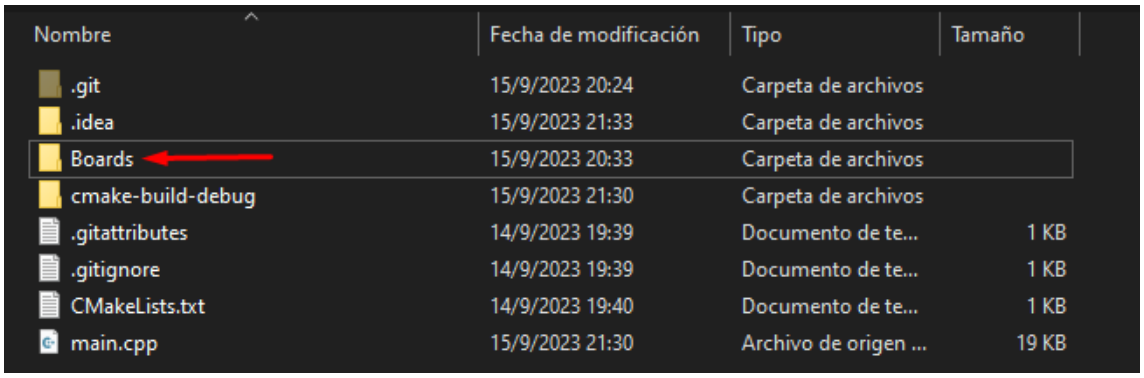
En esta clase se utiliza principalmente para crear y manipular el tablero donde se almacenan los datos. Se podría decir que es la clase principal del programa ya que, exceptuando el manejo de las coordenadas, el resto de las funciones están aquí.

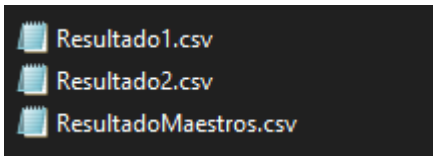
- **Salida por consola:** La función "printBoard()" permite al desarrollador ver por consola el tablero turno a turno, pero se encuentra deshabilitada por defecto. Para habilitarla setear el modo debug en true dentro de la clase.

Formato de salida

La función “exportBoard()” exportar 3 archivos.csv. Uno para cada jugador con la información que pueden ver y un archivo maestro con toda la información de las celdas.
Los tableros se encuentran en la carpeta boards.

- **IMPORTANTE:** Es necesario aclarar que el programa no obtiene información de ellos, simplemente sirven para para mostrar la información al usuario.
- **Aclaración de version:** La función fue programada utilizando CLion en un entorno Windows. Cuando se la trato de utilizar en la máquina virtual de Linux usando eclipse, este dejo de funcionar. Hasta que sea corregida, se deja esta indicación.





Condición de victoria

Cada turno, luego de que el jugador haya completado todas sus acciones, se ejecuta una especie de sort que recorrerá todos los valores del tablero actualizándolos.

Cuando se ejecute el "UpdateBoard()" de cada turno, se volverá a verificar la condición de victoria.

Inicialmente el programa regresara la cantidad de cofres de cada equipo a 0. Y luego ira sumando los que haya encontrado en el tablero. Al terminar la revisión, si se da el caso de que un jugador no posee tesoros enterrados, el juego le dará la victoria al oponente.

```
void updateBoard(int** board){
    // Vacia el contador de cofres, para ir sumandolos a medida que aparescan
    playerOneRemainingChests = 0;
    playerTwoRemainingChests = 0;
```

Modo debug

El código cuenta con un modo debug para ver el tablero por consola a medida que se juega, para eso se debe pasar el booleano a "true" dentro de la clase tablero. Actualmente esta deshabilitado para que los jugadores solo puedan ver la consola y sus tableros.

Casillas

Casillas compartidas: Las casillas (o celdas) pueden poseer diferentes valores dependiendo del tipo de elemento que halla en ellas. Los valores numéricos de la celda determinan el estado en la que se encuentra, (Vacía, con tesoro, con espía o inhabilitadas).

- **0 vacío:** No hay espías ni cofres en el casillero y la casilla esta habilitad para ser elegida.
- **1 a 5 Inhabilitada:** La casilla se encuentra bloqueada por tantos turnos como indica su número. Representan el tiempo que se requiere para desenterrar el cofre de esa celda. Estos valores son siempre positivos. Al final de cada turno su número baja en uno, quedando nuevamente libre al llegar a cero.
- **|6| Cofre:** Representa un cofre enterrado por un jugador. ***(Ver casillas con signo)***
- **|7| Espía:** Representa un espía oculto. ***(Ver casillas con signo)***
- **8 múltiples cofres:** La función de cofres múltiples fue deprecada.

Casillas con signo (Cofres y espías): En el caso de las celdas de cofre simple y espías, el signo de las celdas determina a que jugador pertenecen, siendo el positivo el jugador 1 (blanco) y el negativo para el 2 (negro).

- **6 y -6 Cofre:** Representa un cofre enterrado por un jugador. El "6" para el jugador 1 y "-6" para el jugador 2.
- **7 y -7 Espía:** Representa un espía oculto. El "7" para el jugador 1 y "-7" para el jugador 2.

- Es importante tener en cuenta, que cuando hay un espía se coloca sobre un cofre rival, el signo no es relevante, ya que comenzara a cavar y los turnos siempre son contados en forma de enteros positivos.
- Es importante tener en cuenta que cuando el programa exporta los tableros de los jugadores. Los valores de -6 y -7 del jugador 2, serán pasados a positivos para mostrarlos de forma más amena a su tablero. Esto se realizará automáticamente en el proceso de exportación.