Manual de Usuario - Juego Tesoro Binario

¡Bienvenido al juego Tesoro Binario! Este es un juego de mesa para dos jugadores donde tu objetivo es esconder tus tesoros y descubrir los tesoros de tu oponente.

Contenido:

Consejos Estratégicos

1. Reglas del Juego:

Tesoro Binario es un juego de estrategia en un tablero de 20x20 casillas. Cada jugador tiene 4 tesoros para esconder en el tablero y un espía para explorar el terreno del oponente. Las reglas son las siguientes:

Objetivo: El objetivo del juego es capturar todos los tesoros dej jugador enemigo. El juego termina cuando uno de los jugadores se queda sin tesoros.

Turnos: El juego se desarrolla por turnos, empezando por el jugador 1 y luego el jugador 2. Durante tu turno, puedes realizar una de las siguientes acciones:

a. Ingresar un tesoro $: Elige una casilla vacía en el tablero para ocultar uno de tus tesoros.

b. Ingresar un espía E: Coloca un espía en una casilla vacía para explorar el tablero del oponente.

c. Mover un tesoro: Si tienes un tesoro en juego, puedes moverlo una casilla en cualquier dirección (horizontal, vertical o diagonal).

Reglas de Ingreso:

Si ingreses un tesoro sobre un tesoro enemigo, se revela la ubicación del tesoro enemigo.
Si ingreses un tesoro sobre un espía enemigo, el tesoro se pierde.
Si un espía se encuentra con otro espía enemigo, ambos son eliminados.
Cada jugador solo puede tener un elemento por casilla.

2. Iniciar una Partida:
Para comenzar una partida, sigue estos pasos:

El juego te presentará las reglas. ¡Asegúrate de leerlas detenidamente!
Cada jugador debe ingresar sus 4 tesoros en el tablero de juego. Selecciona una casilla vacía para ocultar tu tesoro. Asegúrate de no poner dos tesoros en la misma casilla.

3. Desarrollo del Juego:
El juego se desarrolla por turnos, siguiendo estas reglas:

Turno del Jugador 1: Comienza el primer jugador. Durante su turno, puede realizar una acción válida (Ingresar tesoro o espía, o mover un tesoro). S

Turno del Jugador 2: Luego, es el turno del segundo jugador. Las mismas reglas se aplican aquí.

Visualización del Tablero: Después de cada turno, el tablero se actualizará para mostrar los cambios realizados por los jugadores.

4. Fin del Juego:

Terina cuando un jugador se queda sin tesoros

5. Consejos:
Algunos consejos que pueden ser utiles al momento de jugar:

Planea tus movimientos con anticipación.
Usar espias para explorar el tablero es una buena idea
Ten cuidado en mover tesoros, podria causar que tu tesoro sea eliminado

MANUAL DEL PROGRAMADOR

C++

C++ es un lenguaje de alto nivel con la particularidad de poder trabajar con Objetos (POO) y de poder trabajar memoria/hardware directamente por el funicionamiento de punteros y otros operadores varios / dado que c++ es una version “moderna” de C, cualquier codigo de C es compatible con C++.

El codigo esta desarollado en C++ cumpliendo la norma C++98, esta creado con clases para mantener el codigo ordenado y facilitar cualquier ctualizacion/modificacion que sea necesaria al codigo porque las funcionalidades estan encapsuladas en un solo lugar.
Esto permite agregar funciones, cambiarlas, agregar metodos, vectores y cualquier tipo de dato que sea constante en el juego/programa que se desee construit

Git.

Para hacer uso del juego o acceder al codigo fuente del programa, anadir funcionalidades o modificaciones basta con clonar el repositorio
Repo: https://github.com/AndresBlanco-Debug/AlgoDos.fiuba/tree/main/Juegocpp

Librerias.

El juego usa 3 librerias diferentes para su funcionamiento:

1. <iostream>: librería escencial para poder usar c++
2. <fstream>: librería que nos permite trabajar con archivos de texto
3. <vector>: libretia que nos permite trabajar con diferentes tipos de vectores.

Linux e IDE.

Este trabajo esta desarrollado en Linux Ubuntu 22.04, en conjunto con un IDE llamado Clion, compilado y debugeado con GNU.

Clases.
El trabajo esta desarollado usadndo el paradigma POO o Programacion Orientada a Objetos usando 3 clases principales que se encagran de la interaccion entre el jugador, el tablero y el juego.

Funcion de la clase jugador que pide coordenadas al usuario

```
void Player::pedirCoordenadas(int &fila, int &columna) {
    cout << "Ingrese la fila deseada: " << '\n';
    cin >> fila;
    cout << "Ingrese la columna deseada: " << endl;
    cin >> columna;
}
```

Funcion de la clase tablero que se encarga de validar un una casilla [x][y] es valida

```
bool Tablero::guardarIngresoTesoroP1(int fila, int columna) {
    bool ingresoVal = true;
    pair<int, int> coordenadas( &: fila, &: columna);
    int len = getLongJugador1();
    if(getCasillaActual(fila,columna) == '$'){
        for(int i = 0; i < len; i++){
            if(coordenadas == tesorosPrimerJugador[i]){
                ingresoVal = false;
            }
        }
    }
    else{
        tesorosPrimerJugador.push_back(coordenadas);
        cout << "Tesoro ingresado exitosamente" << endl;
        tablero[fila][columna] = '$';
    }
    return ingresoVal;
}
```

Una rapida explicacion a una de las muchas validaciones que tiene la clase tablero.
Se delcara como booleano para poder de forma eficiente si la funcion se ejecuta o no.

la funcion en cuestion funciona de la siguiente manera:

1. se recibe como parametro la fila y columna (casilla) donde se desea poner el tesoro
2. se crea una variable ingresoVal como true, suponiendo que el ingreso es valido
3. se crea un vector de tipo par <int,int> llamaddo coordenadas que almacena fila y columna
4. se revisa si la casilla deseada tiene un tesoro en ella
5. si lo tiene recorre el vector donde se almacenan las coordenadas de los tesoros del primer jugador
6. si las coordenadas estan en alguna parte del vector regresa false, significanfo que el tesoro en la casilla pertenece al mismo jugador, lo cual no es un ingreso valido y retorna false
7. Si cualquiera de las condiciones arriba no se cumple, guarda las coordenadas en el vector de tesoros
8. avisa al usuario que el ingreso ha sifo exitoso, actualiza el tablero y regresa true

Problemas y como se resuelven.

```cpp
bool Juego::ingresoTesP1(int fila, int columna) {
    if(tableroJuego.guardarIngresoTesoroP1(fila,columna)){
        return true;
    }
    return false;
}
```

```cpp
void Juego::ingresoInicialP1() {
    int ingresos = 0;
    while(ingresos < 4){
        int fila, columna;
        primerJugador.pedirCoordenadas( &: fila, &: columna);
        while(!ingresoTesP1(fila,columna)){
            cout << "Error! el jugador ya posee un tesoro en la casilla" << endl;
            primerJugador.pedirCoordenadas( &: fila, &: columna);
        }
        if(tableroJuego.compararTesoros()){
            cout << "SE HAN INGRESADO 2 TESOROS EN LA MISMA CASILLA" << '\n';
            cout << fila << ", " << columna << endl;
        }
        primerJugador.depositarTesoro();
        ingresos++;
    }
}
```

Veamos la funcion ingresoInicialP1: esta pensada para el ingreso inicial según las reglas del juego, hay que tener en cuenta que el uusuario puede proporcionar una casilla que no sea valida, puede puede poner todos los tesoros en una casilla y afectar completamente el funcionamiento del juego

Se soluciona usando un while(NOT(ingresoTes1)), donde ingresoTes1 es una funcion booleana que regresa true si se cumplen con las normas para guardar un tesoro en el tablero, de lo contrario se queda en el while informandole al jugador que la casilla no es valida y pidiendole otra casilla, el jugador no puede salir del ciclo hasta proporcionar una casilla valida. Para evitar un ciclo infinito de un usuario malicioso se podria implementar un contador llamado fallos que se incremente cada vez que el jugador pone una casilla incorrecta, si se llega a 3 se rompe el ciclo y termina el turno del jugador.

Cuestionario

Que es un Debug?
Se refiere al Proceso de identificar y corregir errores o bugs en el programa, el objetivo es econtrar y solucionar comportamientos inesperados que pueden ser de varios tipos como: logico, sintactico o problemas en el control de flujo de control.

Que es un breakpoint?
Es una herramienta que se usa para indicarle al programa donde parar a la hora de hacer el debuggin, cuando se llega al breakpoint el programa se dentendra y le da la oportunidad al desarrollador de ver que esta provocando el comportamiento inesperado en el codigo, como ver que pasa con variables, condicionales o hasta ciclos infinitos.

Que es Step Into, Step Over, Step Out?
Step Into: Cuando se esta ejeutando una linea de codigo y esa linea llama a otra funcion, el depurador “entra” en la funcion y se coloca al inicio, esto permite ver como se comporta la funcion, seguir su ejecucion y comprender mejor su comportamiento interno

Step Over: Permite ejecutar la siguiente linea de codigo pero si esa línea contiene una llamada a una función, la función se ejecuta en su totalidad sin entrar en su interior, esto es util cuando ya se conoce el comportamiento o no es relevante en ese momento.

Step Out: Se utiliza cuando se esta dentro de una funcion durante la depuracion y se quiere regresar al punto inicial de la funcion actual.