75.41 - Algoritmos y Programación II

Cátedra Ing. Patricia Calvo - 1er cuatrimestre 2023

Trabajo Práctico 1: BATALLA DIGITAL V1.0

Objetivo

Generar una pieza de software que simule el funcionamiento del juego Batalla Digital en su versión Jugador contra Jugador.

Enunciado

Batalla digital es un juego de mesa para dos jugadores en el que se introducen 4 soldados por cada jugador en un tablero de 20 por 20 con el fin de que los soldados sobrevivan hasta el final, En cada turno el jugador deja una mina en un casillero del tablero, eliminando si hay un soldado en ella o dejando la casilla minada. Cuando la mina explota, deja la casilla inactiva por 5 turnos. El jugador, luego de dejar la mina, puede optar por mover un soldado, ya sea horizontal, vertical o diagonal. Si un soldado se mueve a una casilla minada, la mina explota y el soldado muere, dejando la casilla inactiva. El jugador no se puede mover a una casilla inactiva y si se mueve a una casilla con un soldado contrario, se eliminan los 2 soldados. El juego termina cuando un jugador se queda sin soldados, ganando el otro jugador.

Interfaz de usuario

Toda la interfaz de usuario debe estar basada en texto. El estado del tablero tiene que mostrarse usando caracteres dispuestos en filas y columnas. Por ejemplo, usando ‘ ’ para representar un casillero vacío, ‘X’ para representar un casillero inactivo, ‘1’ para indicar un

jugador y ‘2’ para el otro, solo se ven las minas propias. Además, después de cada turno, el programa debe exportar el tablero en un archivo de texto con el estado del tablero, uno para el jugador 1 y otro para el jugador 2, así no ven sus posiciones.

No es necesario que se limpie la pantalla, simplemente escribir el estado del tablero luego de cada jugada.

Cuestionario

Responder el siguiente Cuestionario:

1) ¿Que es un Debug?
2) ¿Que es un “Breakpoint”?
3) ¿Que es “Step Into”, “Step Over” y “Step Out”?

Normas de entrega

Trabajo práctico individual: 1 persona.

Reglas generales: respetar el Apéndice A.

Se deberá subir un único archivo comprimido al campus, en un link que se habilitará para esta entrega. Este archivo deberá tener un nombre formado de la siguiente manera:

Padron-TP1.zip

Deberá contener los archivos fuentes (no los binarios), el informe del trabajo realizado, las respuestas al cuestionario, el manual del usuario y el manual del programador (Todo en el mismo PDF).

La fecha de entrega vence el día lunes 31/03/23 a las 23.59hs.

Se evaluará: funcionalidad, eficiencia, algoritmos utilizados, buenas prácticas de programación, modularización, documentación, gestión de memoria y estructuras de datos.

Apéndice A

1) Usar las siguientes convenciones para nombrar identificadores.

a) Clases y structs: Los nombres de clases y structs siempre deben comenzar con la primera letra en mayúscula en cada palabra, deben ser simples y descriptivos. Se concatenan todas las palabras. Ejemplo: Coche, Vehiculo, CentralTelefonica.
b) Métodos y funciones: Deben comenzar con letra minúscula, y si está compuesta por 2 o más palabras, la primera letra de la segunda palabra debe comenzar con mayúscula. De preferencia que sean verbos. Ejemplo: arrancarCoche(), sumar().
c) Variables y objetos: las variables siguen la misma convención que los métodos. Por Ejemplo: alumno, padronElectoral.
d) Constantes: Las variables constantes o finales, las cuales no cambian su valor durante todo el programa se deben escribir en mayúsculas, concatenadas por "_". Ejemplo: ANCHO, VACIO, COLOR_BASE.

2) El lenguaje utilizado es C++, esto quiere decir que se debe utilizar siempre C++ y no C, por lo tanto una forma de darse cuenta de esto es no incluir nada que tenga .h, por ejemplo #include <iostream> .
3) No usar sentencias 'using namespace' en los .h, solo en los .cpp. Por ejemplo, para referenciar el tipo string en el .h se pone std::string.
4) No usar 'and' y 'or', utilizar los operadores '&&' y '||' respectivamente.
5) Compilar en forma ANSI. Debe estar desarrollado en linux con eclipse y g++. Utilizamos el estándar C++98.
6) Chequear memoria antes de entregar. No tener accesos fuera de rango ni memoria colgada.
7) Si el trabajo práctico requiere archivos para procesar, entregar los archivos de prueba en la entrega del TP. Utilizar siempre rutas relativas y no absolutas.
8) Entregar el informe explicando el TP realizado, manual de usuario y manual del programador.
9) Comentar el código. Todos los tipos, métodos y funciones deberían tener sus comentarios en el .h que los declara.
10) Modularizar el código. No entregar 1 o 2 archivos, separar cada clase o struct con sus funcionalidades en un .h y .cpp
11) No inicializar valores dentro del struct o .h.
12) El tablero en el TP 1 tiene que ser un array dinámico. En el TP 2 debe ser implementado con listas.
13) Si cualquier estructura de control tiene 1 línea, utilizar {} siempre, por ejemplo:

```
for(int i = 0; i < 10; i++) {
      std::cout << i;
}
```