Instituto Tecnológico de Costa Rica

Análisis y diseño de Algoritmos

Battleship

Integrantes:

Daniel Alvaro Sáenz Cordero

2020162527

Brayan De Jesus Barquero Madrigal

2021067205

Profesor:

Walter Jeancarlo Alvarez Grijalba

6 de abril de 2023

Introducción

El siguiente proyecto, "Battle ship"tiene como objetivo ser un programa desarrollado en el lenguaje c++, capaz de simular una partida del famoso juego del mismo nombre. En este juego se busca derribar las embarcaciones del contrincante mediante disparos continuos. Aunque, este programa posee requerimientos y restricciones, por lo que deberá cumplir con ciertas condiciones, de las cuales las más importantes o destacables son las siguientes:

- Barcos: El programa deberá ser capaz de leer un archivo de tipo .txt con la información necesaria para que inicie el juego, con el nombre de cada jugador y la posición de las embarcaciones del jugador.

- Tableros: Cada jugador tendrá 2 tableros, uno donde colocará sus barcos y registrará los tiros de su oponente. Mientras que el otro tablero será para registrar los tiros de su oponente.

- Desarollo del juego: En este apartado se desarrollaran varias acciones, entre las cuales se encuentra; El jugador indicará donde irá su disparo por medio de un sistema de coordenadas determinado por un sistema de letras en eje horizontal, y números en eje vertical. Si el disparo de un jugador impacta una embarcación, podrá continuar disparando hasta que falle. El tablero se debe actualizar en cada jugada y así como indicar lo ocurrido en cada turno.

- Fin de la batalla: El juego debe detectar cuando un jugador hunde por completo todas las embarcaciones de su contrincante, determinándolo como ganador.

- Resumen del juego: El programa deberá coleccionar una bitácora del juego, la cual indicara en la pantalla la cantidad de disparos de cada jugador, en que posición disparó y cuantos barcos de cada jugador fueron hundidos y por último el ganador de la partida. Esto deberá realizarse en una pila.

Ambiente de desarollo

Durante el desarrollo del proyecto se utilizaron varios ambientes y herramientas para desarrollar el proyecto, por lo tanto, a continuación, se indicarán las herramientas utilizadas y su propósito.

Se utilizaron tanto los sistemas operativos de linux como de Windows.

El Código para el juego fue desarrollado en el lenguaje de programación C++.

El editor de Código utilizado principalmente es Visual studio code.

Se utilizaron los navegadores de Chrome y Opera GX para la investigación de los materiales necesarios para el desarrollo del mismo.

Se utilizó Overleaf como programa de escritura con el fin de utilizar latex para el documento escrito.

Se utilizaron también los bloc de notas incorporados en windows para generar los archivos txt.

Estructuras de datos usadas

Primeramente, definimos un header file con las clases Player, Cell, Board, y Stack. La clase player tiene los atributos públicos de disparo tanto para el jugador 1, como para el jugador 2, un booleano que detecta si el disparo ha dado en el objetivo y finalmente una función de tipo vacío que lee el estado del jugador 1 y 2. Seguidamente la clase Cell define los métodos para determinar si una casilla está ocupada o ha sido disparada. La clase Board se encarga de tener los métodos para desplegar el tablero, colocar el submarino, el crucero y la lancha, así como registrar el disparo del jugador 1 y 2. Y finalmente si el impacto fue en el agua o en una embarcación. Finalmente, la clase Stack, crea una variable llamada items, que es un puntero a un objeto tipo string, asi como dos variables de capacidad y tamaño. Se define un void push para la pila y así como un pop. Así como 2 boleanos que determinan si la pila esta llena o vacía. Con respecto a las funciones y estructuras principales realizadas para la elaboración del proyecto, cabe recalcar que hubo dificultades a la hora de separar correctamente los métodos de battleship del main, por lo tanto, el main se encuentra en el archivo Battleship.cpp. Primeramente, tenemos varias definen los valores tanto de error, como de impactos entre otros, en 0. Esto con el fin de que no se redefinan en otros puntos del Código.
La primera función que tenemos en el Código es un booleano "game over"que determina que si alguno de los dos jugadores recibe 6 impactos el juego se termina, por lo que retorna un "true", puesto que esto determinara que no le quedan embarcaciones disponibles. En caso de no determinar los 6 impactos devolverá un "false".
Seguidamente tenemos la función main, que en caso de que el juego sea diferente a game over (básicamente siempre debería ocurrir al iniciar el Código) este lea el archivo txt. con la información del tablero del jugador 1 y 2. Esto lo logra por medio de la función read j1ƨ read j2", estas básicamente son funciones gemelas. que solo cambian el jugador que analizan. Por lo que con explicar una de ellas se determina el funcionamiento de ambas. La función read j1"lee el archivo txt. e introduce las líneas de Código a las variables respectivas (nombre del jugador y las coordenadas de las diferentes embarcaciones). Se

crea un ciclo while que determina tanto que las coordenadas dadas para el submarino no se salgan del tablero, así como que no superen el número determinado para el tamaño de este. Se repite el mismo proceso tanto como para crucero como para la lancha. Seguido a esto vuelve a comprobar que el juego no esté terminado. Luego crea un ciclo while que se ejecuta siempre y cuando se determine que el juego no se ha acabado y no se determine un error. Por lo tanto, concede el primer tiro al jugador 1 (ocurre exactamente lo mismo con el jugador 2). Este tiro se realiza mediante la función void "Shot J1"que entrega algunos mensajes de guía al jugador, luego despliega el tablero de tiro y el tablero de océano. Luego define las variables necesarias para registrar el tiro del jugador. Finalmente, si el tiro es valido lo registra tanto en el tablero de océano como en el tablero en de tiro. Como anteriormente se explicó, este mismo proceso se ejecuta para el jugador 2 en caso de que el impacto del jugador 1 no haya impactado en ninguna de las embarcaciones. Y si el jugador 2 tampoco impacta alguna de las embarcaciones, se comprueba nuevamente si el juego ha acabado o se da un error, de no ocurrir estas condiciones el ciclo se repite. En caso contrario finaliza el juego.

Seguidamente en el Código fuente tenemos varias variables menores que sirven para colocar correctamente los elementos dentro del tablero, así como el tablero mismo, estas corresponden a la clase Board, y serían las siguientes; place submarina, place cruise, place boat, print. Que simplemente realizan lo que su nombre dice, colocar las embarcaciones y pintar el tablero.

Pasos para ejecutar el proyecto

Los pasos necesarios para ejecutar el proyecto de manera correcta, así como de jugar sin complicaciones es el siguiente

- Primeramente Asegúrese de que su computador posee un sistema operativo compatible con c++, se recomienda usar linux o windows.
- Luego, descargue el archivo comprimido respectivo al proyecto.
- Seguidamente descomprima el contenido en una carpeta vacía, esto con el fin de evitar cualquier error a la hora de ejecutar los archivos.
- En un compilador o idle abra los archivos Battleship.cpp, Battleship.h y main.cpp.
- Ahora, ejecute el archivo en la terminal respectiva a su sistema operativo
- Este abrirá una terminal que le mostrará el tablero de tiro y el tablero de oceano del jugador 1. Abajo de esto se le pedirá introducir la coordenada del disparo, asegúrese de que el disparo este dentro del tablero.
- La coordenada deberá ser introducida en este formato: Letra mayúscula + Numero. Un ejemplo de cómo debe verse la entrada es el siguiente: A2
- Este desplegará la información de tiro, en caso de este golpee la embarcación dirá Ïmpacto!ƶ el jugador 1 podrá hacer otra jugada. ¡En caso contrario el juego dirá "Agua!ƶ el jugador 2 pasará a juego. Este ciclo continuará hasta derribar todas las embarcaciones de un jugador. Al final se desplegará una lista de todas las jugadas realizadas.

Actividades realizadas por estudiante:

Brayan Barquero Madrigal:

- Planteo de objetivos e investigación. 8 horas. Sábado 25 de marzo.
- Investigación. 4 horas. Martes 28 de marzo.
- Definición de clases y atributos. 4 horas. Jueves 30 de marzo.
- Lectura de archivo de texto. 5 horas. Domingo 2 de abril.
- Implementacion de tablero. 10 horas. Lunes 3 de abril.
- Colocación de barcos. 10 horas. Martes 4 de abril.
- Funciones de actualizar tableros. 10 horas.
- Disapros, turnos y fin de juego. 8 horas. Jueves 6 de abril.

  Total de horas: 59 horas.

Daniel Sáenz Cordero:

- Determinación de objetivos, consultas, investigación, creación de seudocódigos. 9 horas. Lunes 26 de marzo hasta el jueves 30 marzo.
- Investigación sobre lectura de archivos, 2 horas. Viernes 31 de marzo.
- Intento de implementación de Código de lectura de archivo. 4 horas. Sábado 1 de abril.
- Investigación sobre listas de listas e implementación. 4 horas. Domingo 2 de abril.
- Investigación sobre pilas e implementación. 5 horas. miércoles 5 de abril.

- Investigación de uso sobre latex, redacción y finalización del documento. 14 horas. miércoles 5 de abril y Jueves 6 de abril.

  Total de horas: 38 horas.

Total de horas entre los 2 estudiantes: 97 horas.

Comentarios finales

El proyecto funciona correctamente casi en su totalidad. A excepción de la pila, que a la hora de ser integrada al proyecto genera muchos errores, por ejemplo, durante la recolección de las coordenadas de tiros, el programa inicia la pila. Sin embargo a la hora de dar el resumen de las jugadas. Simplemente despliegua el mensaje del método, mas no ninguno de los elementos de la pila. Esto se debe seguramente a un mal entendimiento de la manera correcta de intregrar el codigo de una pila al trabajo realizado.
Además, durante el proyecto se encontraron muchos problemas a la hora de crear una lista de listas con los metodos vistos en clase, por lo cual se optó por una solucion más sencilla.
Algunas de las limitaciones encontradas en el proyecto son las siguientes:

- El programa se cuelga en caso de que las coordenadas introducidas sean minusculas.
- El programa detecta el primer numero de la coordenada. Pero aún así permite el segundo aunque no lo toma encuenta para la misma.
- La pila no guarda los tiros realizados por los jugadores.

Conclusiones y recomendaciones

Las estructuras de datos son sumamente importantes para el manejo de archivos, ya que nos permiten organizar de manera correcta la información. Así manejar diferentes tipos de datos de manera fluida. Gracias a este trabajo se logro entender y mejorar los conocimientos respectivos a C++, listas y manejo de datos. Además, se aprendió como usar latex, por medio de overleaf, para la creación de documentos.

De acuerdo con todo lo expuesto anteriormente, así como lo realizado en la parte programada. Se recomienda repasar en gran manera las estructuras de datos referentes a listas, así como de punteros. Investigar y desarrollar pilas capaces de recibir datos de diferentes tipos y funciones. Finalmente se recomienda usar más la herramienta de latex, puesto que corresponde a una muy buena forma de redactar documentos estructurados. En este caso se utilizó únicamente para un documento básico de proyecto. Pero este también posee un montón de ventajas, ya que al ser multiplataforma permite que varias personas en diferentes sistemas operativos o dispositivos puedan trabajar sin ningún inconveniente.

Bibliografía

Codigazo. (2020, 21 junio). Manejo de archivos de texto en C++ (abrir, leer y escribir) [Vídeo]. YouTube. https://www.youtube.com/watch?v=RBZidsPGkfs

GitHub: Let's build from here. (s. f.). GitHub. https://github.com

Joyanes, L., Zahonero, I., Sanchez, L. (2007). ESTRUCTURA DE DATOS EN C++ (1.a ed.). McGraw-Hill.

L. (2023, 7 abril). Contenidos del blog. http://minisconlatex.blogspot.com