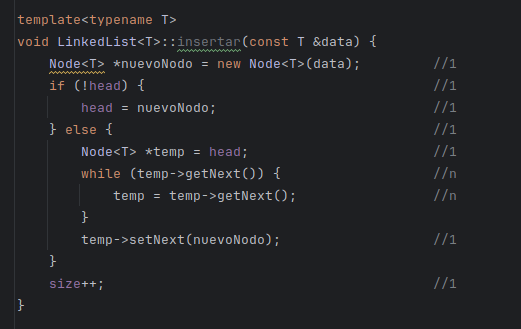
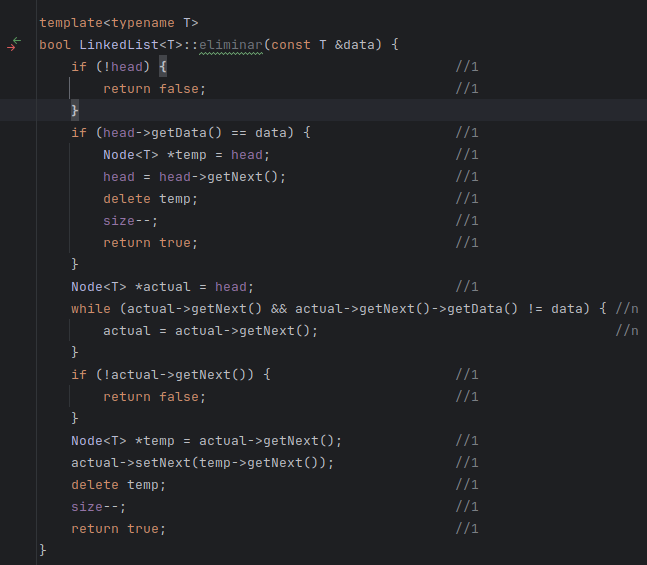
COMPLEJIDADES DE ALGORITMOS

CLASE LINKEDLIST

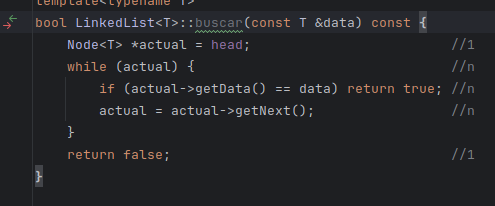
- Método insertar:

**Justificación de complejidad:** O(n) esta complejidad se da cuando se recorre los elementos de una lista una cantidad de n veces en el peor de los casos y 1 vez en el mejor de los casos.

- Método eliminar:

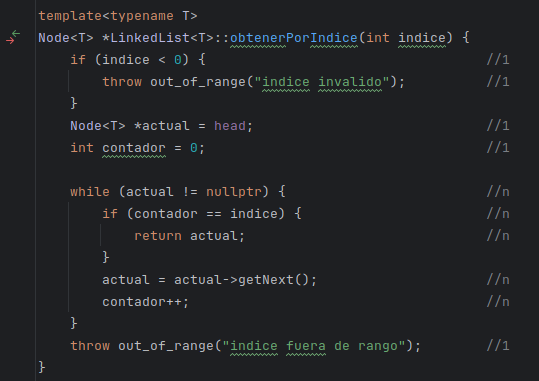
**Justificación de complejidad:** O(n) al igual que el método anterior esta complejidad se da cuando se debe recorrer los elementos de una lista una cantidad de n veces hasta que el elemento a eliminar sea igual al elemento encontrado en x posición.

-Método Buscar:



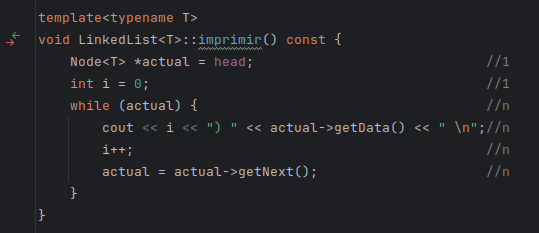
**Justificación de complejidad:** O(n) este método o algoritmo tiene esta complejidad dado que recorre una lista nodo por nodo una cantidad de n veces hasta que el elemento recibido sea igual al elemento que este en el nodo actual.

-Método obtenerPorIndice:



**Justificación de complejidad:** O(n) la complejidad de este método o algoritmo esta definida por el ciclo que se repite una cantidad de n veces en búsqueda del índice que es recibido para buscar el nodo que está en ese índice de la lista.

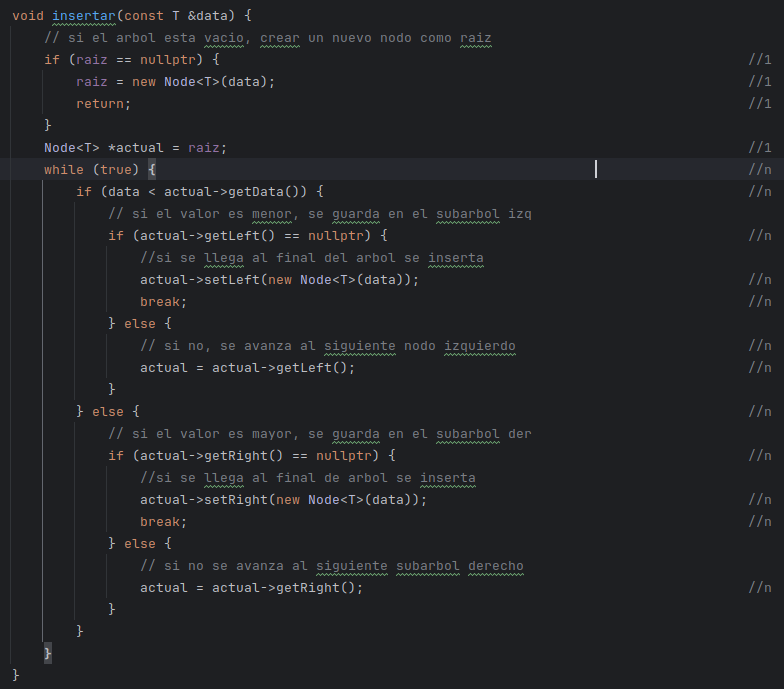
-Método imprimir:



**Justificación de complejidad:** O(n) esta complejidad esta definida por el recorrido de n veces que se le hace a una lista para ir imprimiendo cada nodo que pertenece a la lista.

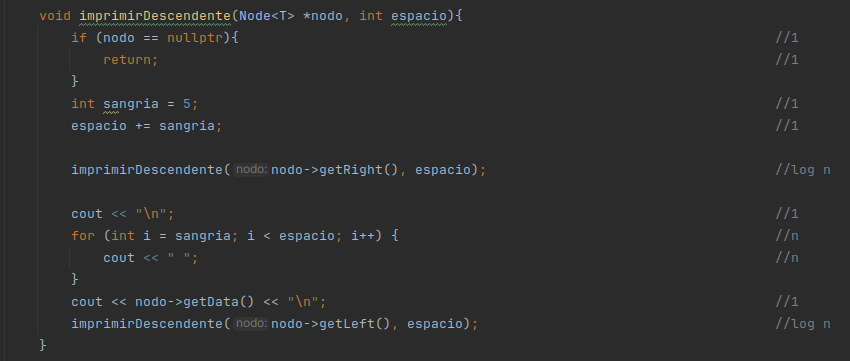
CLASE BST

-Método insertar:



**Justificación de complejidad:** Esta complejidad O(n) está definida también por el recorrido del árbol que puede tener una cantidad de n subárboles, esto en búsqueda del nodo que se ha recibido en la llamada del método.

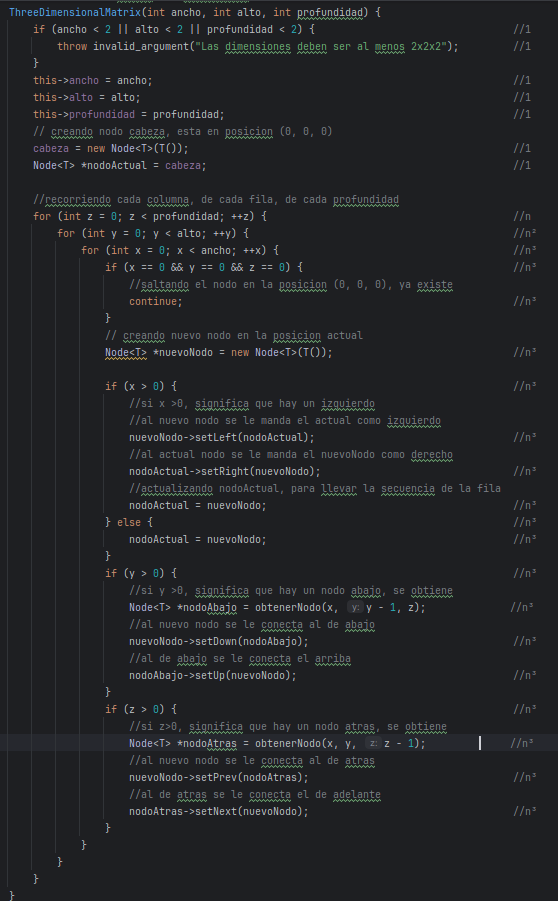
-Método imprimir:



**Justificación de complejidad:** O (log n) esta complejidad esta justificada en la llamada recursiva que se aplica de este método, pero en el momento que se hace esta llamada el árbol que se envía es cada vez mas pequeño, tanto del lado izquierdo como derecho, hasta que se llega a imprimir el árbol en su totalidad.

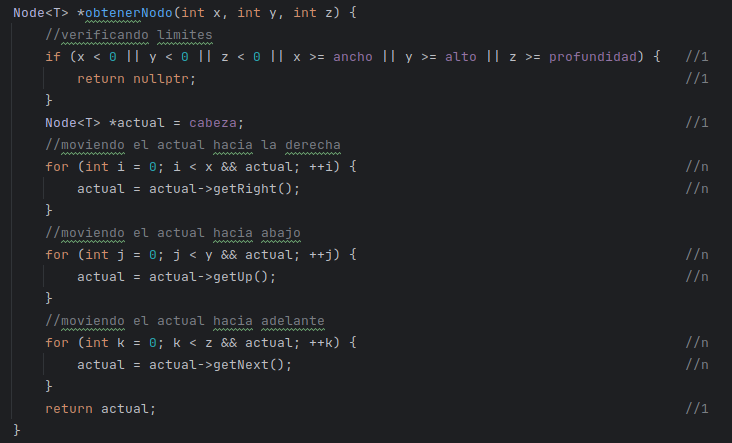
CLASE THREEDIMENSIONALMATRIX

-Constructor:



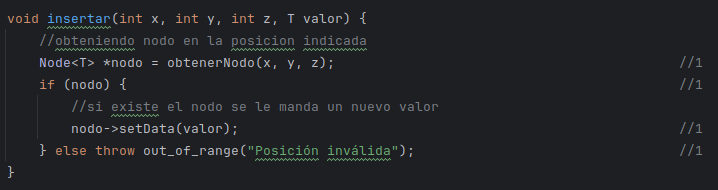
**Justificación de complejidad:** O (n^3) este método tiene esta complejidad para todos los casos, ya que se recorre un for dentro de otro for que esta dentro de un último for (3 fors en total), cada for representa una dimensión para el tablero y en cada iteración se van conectando los nodos.

-Método obtenerNodo:



**Justificación de complejidad:** La complejidad O (n) esta dada por los recorridos de los ciclos for que realizan cada iteración n veces dado una coordenada x, y o z.

-Método insertar:



**Justificación de complejidad:** O (1) este método tiene esta complejidad constante ya que dadas unas coordenadas inserta un dato en el nodo que coincide con estas coordenadas.

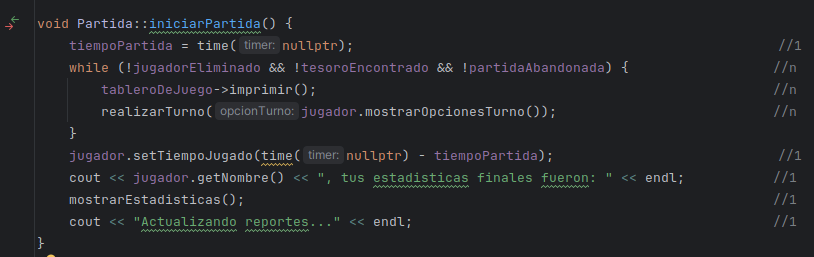
-Método imprimir:



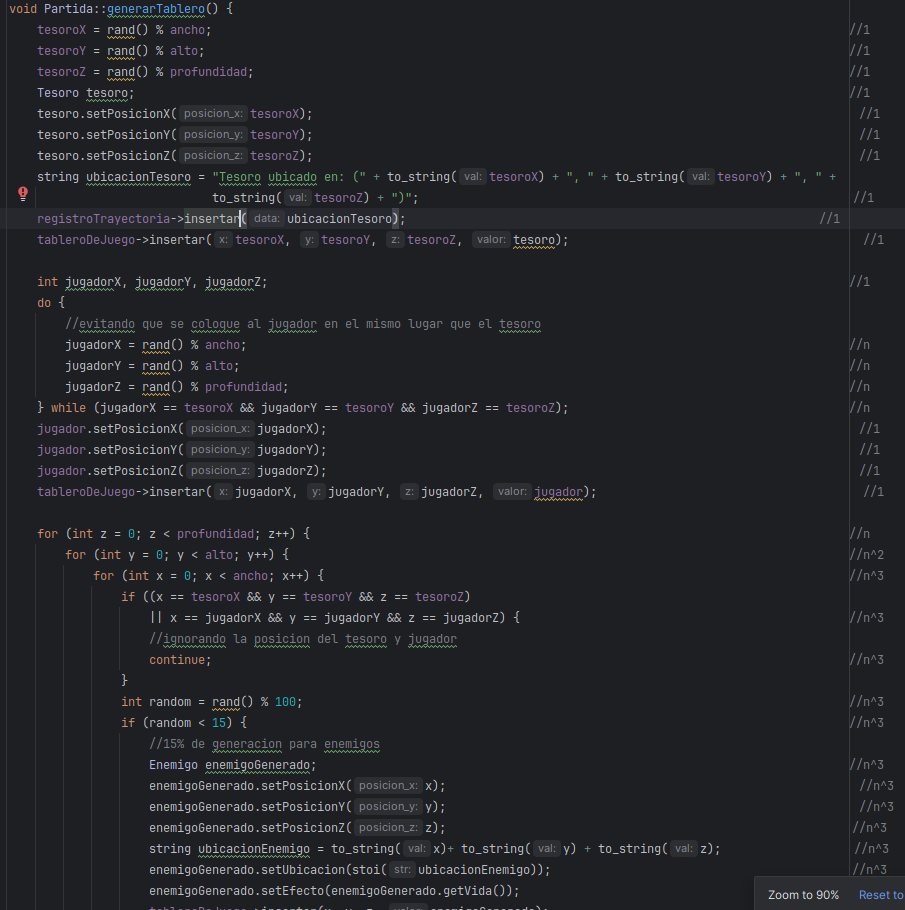
**Justificación de complejidad:** Este método tiene una complejidad de O (n^3) ya que se hace un recorrido de 3 fors anidados y por cada iteración se hace una impresión en el nodo que se esté posicionado.

CLASE PARTIDA

-Método iniciarPartida:



**Justificación de complejidad:** Este método tiene una complejidad de O (n) dado que se repite un while n veces mientras que la partida no haya sido abandonada, el tesoro no haya sido encontrado o el jugador no haya sido eliminado.

-Método generarTablero:

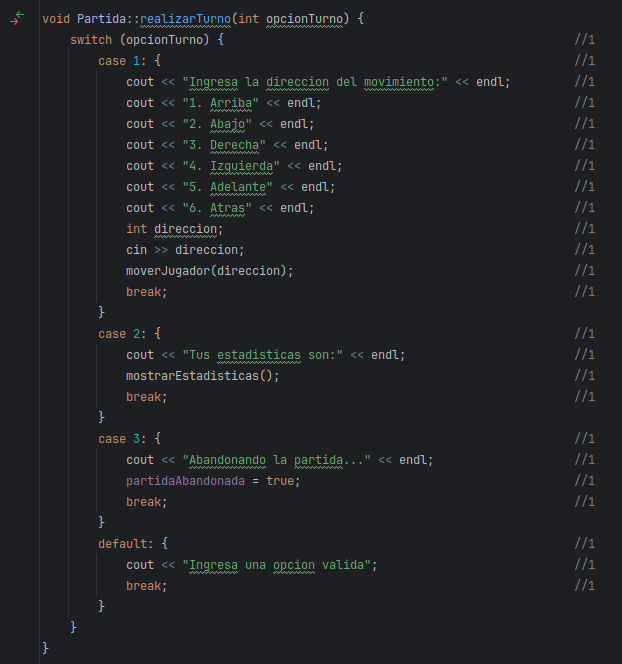
Justificación de complejidad

-Metodo realizarTurno:



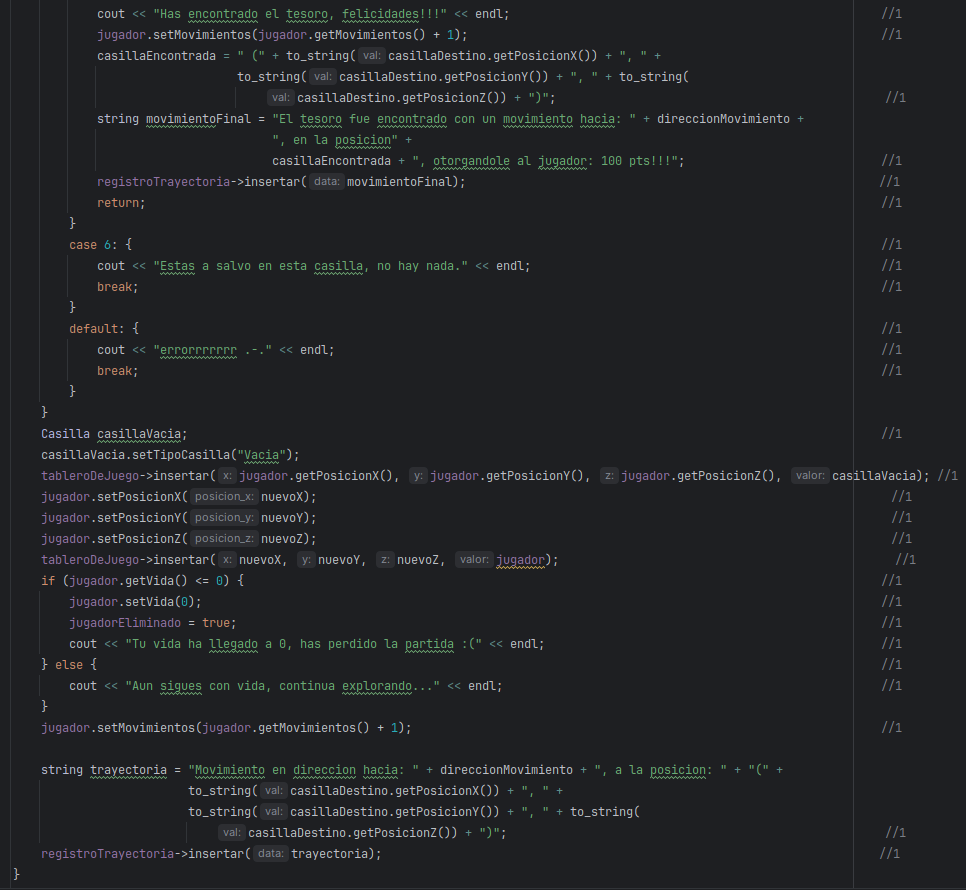
**Justificación de complejidad:** Este método tiene una complejidad de O (n^3) ya que se recorre nuevamente los 3 fors anidados y en cada posición se generan entidades aleatorias, ya sea Enemigos, Trampas, Pócimas o Pistas, además del jugador y el tesoro.

-Método realizarTurno:



**Justificación de complejidad:** este método tiene la complejidad de O (1) ya que todo el algoritmo se ejecuta una sola vez, ya que el jugador solo puede hacer una acción por turno.

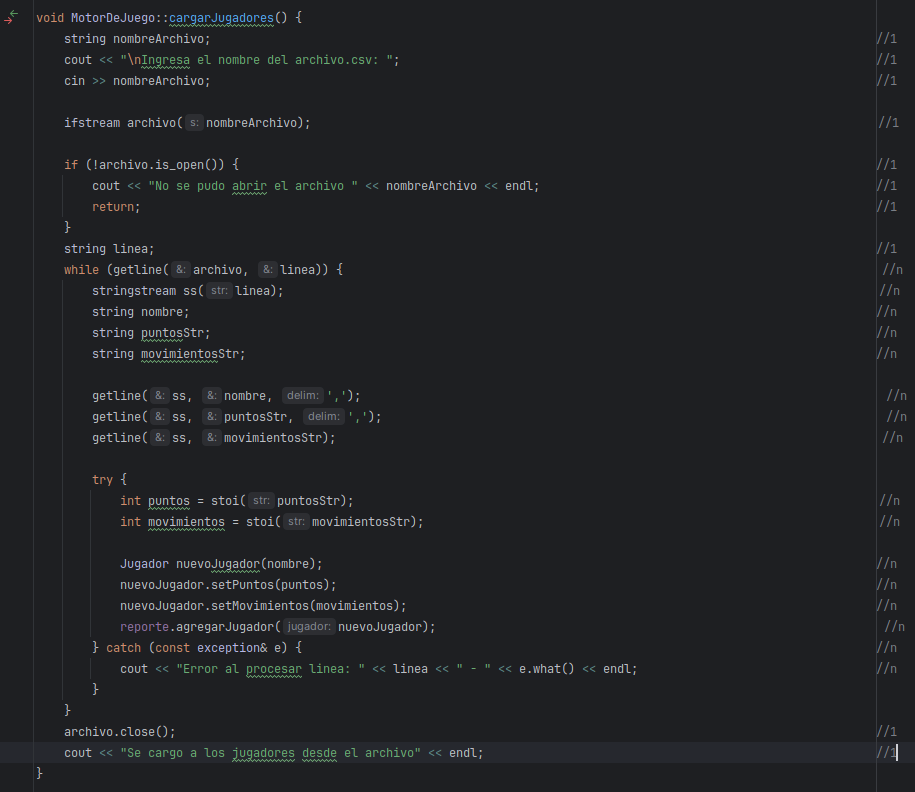
-Método moverjugador:



**Justificación de complejidad:** este método tiene también una complejidad de O (1) ya que el movimiento del jugador a través del mapa se hace un paso a la vez, así mismo como las interacciones del jugador con la entidad que este en la casilla a donde se movió el jugador.

CLASE MOTORDEJUEGO

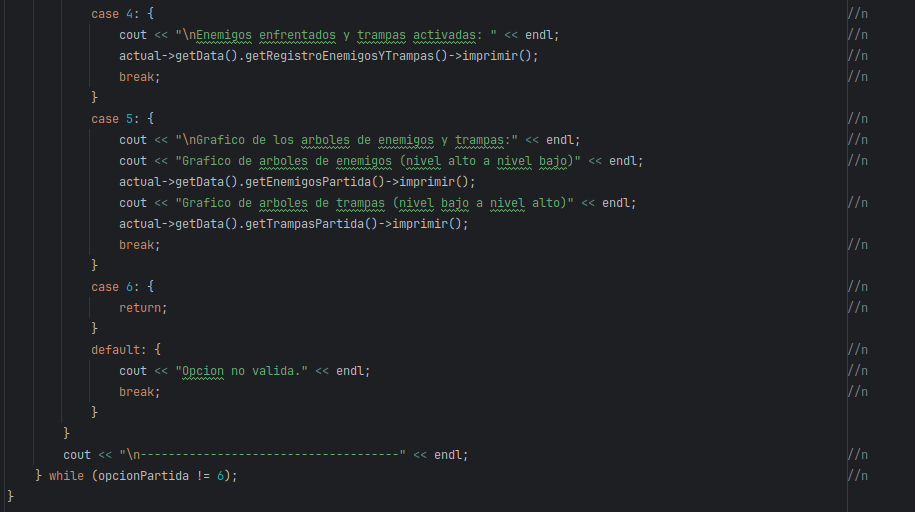
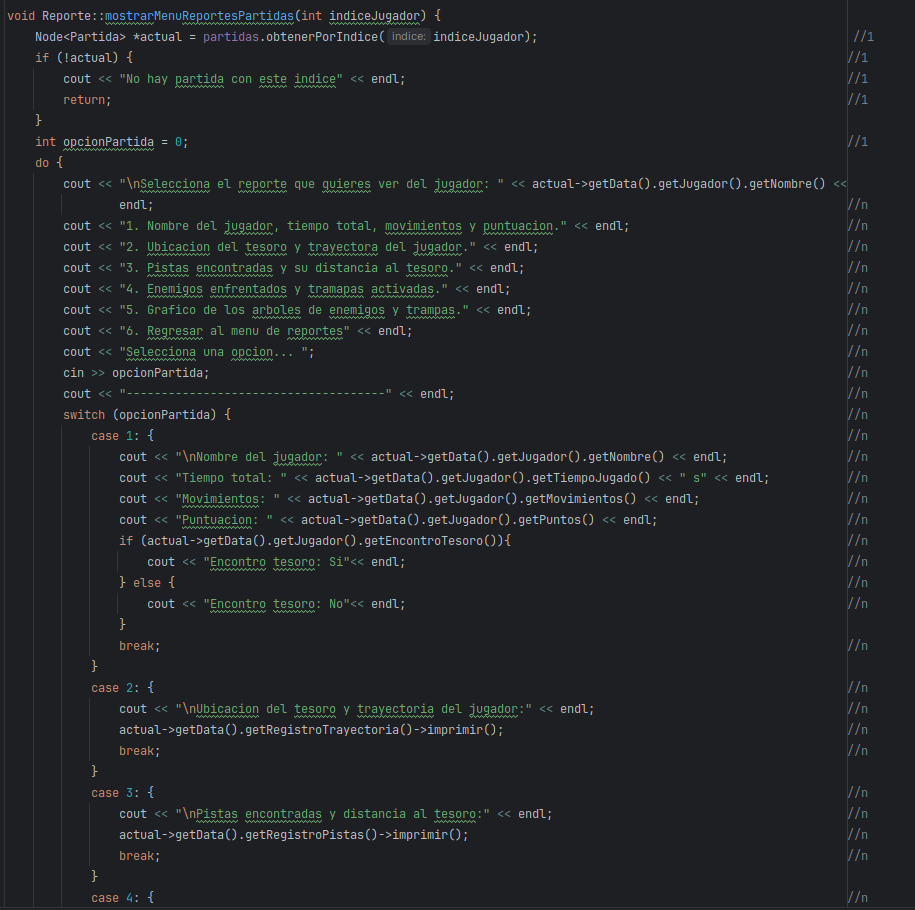
-Método cargarJugadores:



**Justificación de complejidad:** Este método tiene una complejidad de O (n) dado que se lee una línea una cantidad de n veces, hasta que el archivo ya no tenga líneas para leer la información.

CLASE REPORTE

-Método mostrarMenuReporteJugadores:



**Justificación de complejidad:** Este método tiene una complejidad de O (n) ya que las opciones de los reportes que el jugador puede acceder se repiten n veces hasta que el jugador desee regresar al menú principal.

-Método mostrarTablaJugadores:



**Justificación de complejidad:** este método tiene la complejidad de O (n) dado que realiza la impresión de datos de los jugadores n veces, dependiendo de la cantidad de jugadores que haya registrados en la ejecución actual.

-Método ordenarTablaJugadores:



**Justificación de complejidad:** este método de ordenamiento (Bubble-Sort Mejorado) tiene la complejidad de O (n^2) dado que por cada elemento de la lista recorre una vez la lista ordenando los valores por los puntos, hace una iteración de la lista por cada elemento de la lista que se está iterando.