Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática

Tarea programada 3: Laberinto

Objetivos

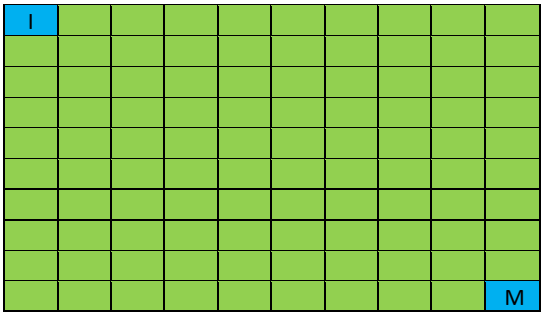
Resolver un problema de complejidad media utilizando todos los elementos aprendidos en clase que incluyen:

- Diseño orientado a objetos
- Programación por objetos
- Programación de métodos para reutilizar código
- Programación de objetos e interacción de los mismos
- Estructuras de control
- Estructuras de repetición
- Manejo de excepciones
- Entrada y salida de datos
- Estructuras de datos
- Entrada y salida de archivos

Instrucciones

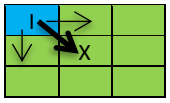
Se deberán programar clases y el programa principal para implementar un juego: **Longleat (laberinto).** Para implementar el juego, considere lo siguiente:

1. El juego tiene la modalidad de *hot seat* para dos jugadores.
2. Se juega en una cuadrícula de tamaño MxM ($8 \leq M \leq 30$).
3. En la superior izquierda es la salida y en la esquina opuesta, por tanto la inferior derecha, es la llegada. Por ejemplo, suponga una cuadrícula de 10x10 casillas, donde I marca el inicio del juego y M la meta:

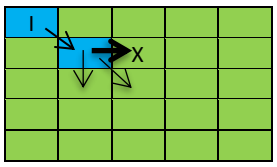


4. El juego se desarrolla por turnos. El primer jugador marca la casilla de salida y, a continuación, el segundo jugador marca otra casilla situada justo debajo, justo a la derecha o en diagonal (lo que equivale a mover una posición a la derecha y una posición abajo).

   Por ejemplo, a partir de la casilla de inicio, el jugador 1 puede moverse en las direcciones que marca la siguiente figura. Suponga que el jugador 1 escogió moverse diagonalmente desde el punto de inicio.



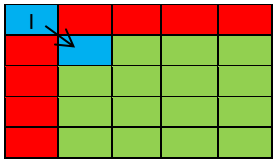
   Ahora suponga que el jugador 2 se movió a la derecha desde el punto X (el lugar seleccionado por el jugador 1). Tome en cuenta que el jugador 2 tenía las mismas opciones de moverse a la derecha, diagonalmente o hacia abajo. Luego el primer jugador hace lo mismo desde la última marca hecha por su oponente y así sucesivamente. Gana el que consiga marcar la casilla señalada como meta.



5. Para lograr los objetivos de programación del juego propuesto, el programa principal se basará en la utilización de: listas simple y doblemente enlazadas y árboles binarios.

La matriz propuesta en el ejemplo anterior deberá estar implementada como: una lista doblemente enlazada, donde cada uno de sus elementos será una lista simplemente enlazada. Es decir los nodos de la lista simplemente enlazada representarán las celdas de una fila apuntada por un nodo de la lista doblemente enlazada.

Con el objetivo de imprimir la submatriz de posiciones a las que se puede llegar en uno o más turnos, genere un árbol binario con los elementos de la matriz de juego. Seguidamente por cada turno jugado, elimine del árbol todas las posiciones que ya no se pueden acceder.
Por ejemplo en el caso anterior, después de hacer el siguiente movimiento:



Se deberán eliminar del árbol de despliegue de movimientos los nodos que representan las casillas marcadas en color rojo.

Para llenar este árbol haga lo siguiente:

- Genere un arreglo con cada uno de los elementos de la matriz (lista de listas).
- Revuelva el arreglo con los elementos de forma aleatoria.
- Recorra el arreglo y agregue cada uno de los elementos en el árbol binario.

Adicionalmente considere lo siguiente:

- Por cada turno jugado, muestre el estado del árbol en pantalla.
- Cuando se finaliza el juego, muestre la matriz de juego completa con el recorrido hecho por los jugadores.
- Al final deberá indicar qué jugador ganó, cuántas veces ha ganado y perdido durante una sesión de juego.
- Deberá guardar los records en un archivo de texto (con todos los jugadores que han competido) incluyendo las veces ganadas y perdidas.
- El orden estará dado por la resta entre la cantidad de veces ganadas menos las perdidas.
- Al inicio deberá solicitar el nombre de cada jugador. Si ese nombre ya existe en el record, deberá advertirle al jugador que ya ha jugado antes y si está de acuerdo deberá tomar las estadísticas previas, acumularlas en esta partida y al final guardarlas de nuevo en archivo de forma ordenada.
- Por cada ronda deberá preguntar el tamaño del tablero de juego.
- Deberá diseñar las clases que considere necesarias. Por ejemplo clase jugador, nodo, lista simplemente enlazada, doblemente enlazada, árbol, lector, escritor, etc.

Opcional extra (10%)

Puede optar por un 10% extra en la tarea al realizar una interfaz gráfica utilizando BufferedImage (como referencia puede utilizar el objeto ObraDeArte explicado en clase).

La interfaz debe contar con:

Una matriz o grid (la forma de una tabla) para el tamaño del tablero, por ejemplo, uno de 8x8 mostrará la cuadrícula:

Por cada movimiento del usuario, deberá ir actualizando la cuadrícula con los movimientos restantes, por lo que se irá decrementando la cantidad de filas o columnas pintadas.

Una vez finalizado el juego, deberá dibujar el recorrido. Para esto debe rellenar los cuadrados con un color para cada jugador. El jugador podrá escoger el color que desea utilizar para rellenar los cuadrados. Por simplicidad se manejarán los siguientes colores:

- Rojo
- Verde
- Azul
- Amarillo
- Anaranjado
- Morado

Importante

Considere las siguientes observaciones:

1. Respetar las reglas de estilo del código: márgenes, nombres utilizan una convención consistente (tipo Java), comentarios para los atributos y variables de métodos.
2. Utilizar un diseño apropiado basado en objetos. Evitar utilizar clases “hace todo”.
3. Modularizar el código lo más posible evitando el procesamiento duplicado de datos.

4. Utilizar doxygen para la documentación interna. Cada método deberá tener su documentación apropiada con las etiquetas de: requiere, descripción y retorna.

Entregables

Se deberá realizar un análisis y diseño del problema. Se espera que en un documento se especifique el problema, los pasos para llevar a cabo la solución, un diseño de clases (en formato UML) y una explicación de los algoritmos complejos utilizando pseudocódigo. Un plan de pruebas y los resultados de las pruebas.

Adicionalmente agregar una sección de problemas encontrados a la hora de resolver la tarea y un porcentaje de distribución del 100% del trabajo de cada uno de los integrantes. Por ejemplo si el grupo es de dos personas y cada estudiante trabajo la misma cantidad, el porcentaje será 50% cada uno.

El código fuente deberá contar con documentación interna y deberá generar la documentación utilizando Doxygen.

Generar un manual de usuario con instrucciones para el uso del juego.

Deberá programar el juego para que sea completamente funcional y utilizar todas las técnicas de programación que considere necesarias y entregar el código fuente.

Adicionalmente, agregar el nombre de su repositorio que utilizó el Github. Se espera que utilice el sistema de control de versiones para agregar frecuentemente los cambios que está haciendo mediante el uso de commits y push al servidor de Github.

Se evaluará el uso de la herramienta de control de versiones usando la herramienta Git.

Notas adicionales

- Se debe crear una cuenta en Github utilizando el correo dado en clase.
- Se recomienda descargar Sourcetree o Git para llevar a cabo los commits.
- Se recomienda usar la estructura con branches.
- Recuerde que no se debe commitear a master código inestable.
- Esta tarea se puede realizar en parejas.
- Deberá generar la documentación externa usando Doxygen.

Desglose

- Documentación interna y externa 25% (documentación interna del código fuente usando Doxygen y los demás puntos explicados anteriormente para la elaboración de la documentación).
- Codificación de la solución 75% (incluye uso del sistema de control de versiones, solución propuesta, funcionalidad, buen diseño, apego a los estándares de código, detección de errores, efectividad y eficiencia de las estructuras de datos y los algoritmos).