Examen Asincrónico 1

ITESO

Universidad Jesuita
de Guadalajara

Materia: Programación en Memoria Dinámica

Alumn@s: Angel Ramirez y José Luis Almendarez Gonzalez
Maestr@: Francisco Cervantes

Tabla de Contenidos

Introducción

En este primer examen asincrónico se nos pidió hacer un clon de batalla naval en base a ciertos parámetros e instrucciones y usando los “pointers” como método principal.
Este programa tenía que tener un menú donde pudiéramos:

- Cambiar la dimensión del tablero.
- Escoger el modo de juego
- Jugar
- Salir

Si no modificamos las dimensiones del tablero, el tamaño predeterminado será 10 x 10 casillas.

Los modos de juegos que podemos escoger son el didáctico y el fácil. En el didáctico podemos ver el tablero de nuestro contrincante aparte del nuestro y en el fácil únicamente vemos nuestro tablero.

Uno de los parámetros que necesitábamos antes era hacer una estructura de datos llamada “CELDA” que le asignaría un valor a la celda de 0 si está libre y 1 si está ocupado o impactada y donde le podríamos un identificador a cada una.

También, tendríamos que contar con otra estructura llamada “NAVE” que servirá para determinar la orientación de la nave, las casillas que ocupan estas naves y su estado de salud.

Como quedó claro en las opciones del menú tenemos que hacer una matriz en la cual nosotros determinamos el tamaño que va a guardar la estructura celda para poder hacer cambios y trabajar sobre ella, donde las naves van a ocupar alrededor de 30% de la pantalla y su posicionamiento será totalmente aleatorio.

Finalmente, el último requerimiento es el más importante, pues de nada sirve todo lo anterior si no podemos jugarlo, entonces tenemos que establecer un algoritmo adecuado para que la máquina apunte y dispare en el tablero, que nosotros lo podamos hacer y que haya ciertos parámetros que determinen que el juego sigue en pie, pues aún hay naves a flote. Estos turnos de ataque tienen se tienen que alternar turnos entre la computadora y tú.

Una vez ya dejados en claro los parámetros e instrucciones que tenemos que seguir para poder realizar el proyecto podemos continuar con la primera etapa del desarrollo y la segunda parte de este reporte, es decir, el análisis del problema.

Desarrollo

Análisis del problema

En un primer momento el problema era abrumador pues a pesar de que teníamos el conocimiento de que teníamos que trabajar una matriz y recorrer líneas para usar apuntadores y cambiar el estado de la celda para sobre eso trabajar el comportamiento del juego. Pues cuando uno inicia un trabajo desde cero realmente no sabe ni por dónde empezar, por ejemplo una de las preguntas que teníamos era sobre cómo íbamos a recorrer el arreglo para poder acomodarlo de manera vertical.

Así que dividimos los problemas en pequeños pedazos para imaginarnos cuales tenían que ser los input y los output que necesitábamos, nos repartimos qué funciones iba a hacer cada quien y así fuimos juntando las piezas del rompecabezas.

Primero nos quedó muy en claro que teníamos que antes que nada hacer la matriz y hacer las Structs, la verdad es que gracias a que en el problema planteado por el profesor venía muy especifico lo de los structs lo que nos permite también determinar incluso el tamaño máximo de la matriz, en qué parámetros se va a establecer.

Después de eso establecimos 3 funciones importantes que teníamos que realizar si queríamos avanzar en el proyecto, después de esas 3 nos imaginamos que todo iba a ser más sencillo y así fue pues todo salió lógico y natural a como lo desarrollamos, al final del dia todo se hizo con la básica idea de localizar la casilla, leer el estado de la celda y dependiendo de ese estado cambiarlo o tomar una decisión.

Entonces estas 4 funciones importantes fueron asignar una estructura CELDA a todas las casillas, repartir los barcos por el tablero, apuntar a la casilla e imprimir el tablero. Lo demás, las otras funciones fueron una consecuencia de lo que ya habíamos logrado.

- Para poder repartir la estructura CELDA a todas las casillas realmente fue fácil pues nuestro razonamiento fue recorrer toda la matriz e ir asignando la estructura casilla por casilla a través de un ciclo for anidado.

- Para poder repartir los barcos a través del tablero supusimos que teníamos que recorrer todas las casillas y basado en una variable random le vaya asignando un estado de celda especial del 25% al 35% de las casillas. Ese fue nuestro primer análisis y pensamiento del problema, tuvimos ciertas complicaciones donde tuvimos que invertir algunos valores, pero eso está documentado en el código. La idea que se expone en este texto era la idea inicial y lo que buscábamos hacer desde un comienzo.

- Para poder encontrar la casilla específica al apuntar y revisar el estado de la celda encontramos la fórmula matemática ((size - y-1) * size) + x donde size es el tamaño del tablero que rápidamente concluimos que para poder decidir el tamaño era cuestión de con un simple scanf poner la variable donde correspondiera.

- Finalmente para imprimir el tablero únicamente hicimos nuevamente un ciclo for anidado donde dependiendo de qué estado tenga la casilla correspondiente se imprimirá con un distinto símbolo para que podamos distinguir los distintos estados de la celda de manera visual.

Lo demás como fue el chequear si ya habíamos ganado o perdido, el comportamiento de nuestro oponente o los distintos modos de juego únicamente fueron consecuencia de lo que ya habíamos realizado, si teníamos que cambiar algún comportamiento o algún valor en una casilla podíamos hacerlo gracias a las anteriores funciones. Por ejemplo, la función de game over es casi por así decirlo una copia de la función shoot(que es la que dispara) pero recorriendo y revisando todas las casillas.

En conclusion del analisis de soluciones, una vez establecido el tablero y habiendo asignado los Structs a cada casilla todo fue cuestión de jugar con los estados de la celda.

Propuesta de Solución

Acerca de los Structs: estos objetos, por así llamarlos. Tienen parámetros que usaremos en los métodos pero primero describiremos cuales son estos y para qué van a servir.

Struct CELDA: Dentro de esta struct, los parámetros que pueden cambiar son la ID, el estado de la celda y el detector de impacto. En este caso tuvimos que hacer dos Structs, pero ambos organizados de la misma manera. Uno para el tablero de la computadora y otro para nuestro propio tablero, esto para evitar errores innecesarios. Esta celda se localiza en un tablero, que es una matriz.

Struct NAVE: Dentro de este struct, los parámetros que podemos decidir son: el tipo de nave, a través de caracteres; la orientación de la nave, también a través de caracteres y un parámetro que puede cambiar para indicar que la nave fue impactada y hundida.

choose_ship(): Para este método vamos a escoger un número random de 1 a 5 y dependiendo de esto se determinará el tamaño del barco.

fill_board(): Dentro de un arreglo al estar todos los espacios de memoria una a lado de otro lo único que tenemos que hacer es hacer un ciclo for (i <= size) para poder recorrer toda la matriz y asignar los estados de celda, id y está de celda que necesitamos. Esto es únicamente para el tablero del oponente.

fill_board_ships(): En este método primero vamos a apuntar al tablero, a la posición [0][0]. Después vamos a determinar un rango para que una variable random, pueda escoger un dato dentro de ese random. Vamos a usar una variable random otra vez para poder escoger la orientación del barco, este cálculo nos va a regresar un dato tipo String que puede ser ‘v’ para vertical y ‘h’ para horizontal. Asignamos la posición de la nave mediante un random y dependiendo de si es ‘v’ o ‘h’ la fórmula para recorrer la matriz y cambiar el estado de la celda. Esto es únicamente para el tablero del oponente.

shoot(): Dentro de este método le vamos a pedir al usuario dos distintas coordenadas si estas variables Integer son mayores al tamaño designado del tablero entonces vamos, si está dentro de los parámetros entonces dentro de “x” y ”y” vamos a aplicar una fórmula para alcanzar la locación en la matriz. Si el estado de esta celda es igual a 0 entonces cambiamos el estado a 1 e imprimimos “fallaste”, cualquier otro resultado indicará que atinaste el disparo, se cambiará el estado de la celda a 1 y se imprimirá “atinaste”.

check_ship_sunk(): Esta función es muy sencilla, pues revisa las casillas para revisar si identificador y compararlo. De esa manera se comprueba el estado de la celda y si el barco está hundido.

print_board_and_impact(): Esta función lo que hace es imprimir el tablero del oponente y los impactos recibidos. Esto a través de un ciclo for anidado donde dependiendo del estado de la celda se imprimir “_” si la casilla está vacía; “0” si tiene un barco y “X” si está impactada. De nuevo, todo dependiendo del estado de la celda de nuestro Struct CELDA.

check_game_over(): Este método de aquí lo que va a hacer es recorrer todas las celdas para saber si queda algún estado de celda que aún contenga una nave, dependiendo de esto van a regresar la variable game_over que tiene un resultado booleano. Este únicamente es para el game over del tablero del oponente.

attack_user_board(): Este función es similar a la función shoot() con la ligera diferencia de que aquí las posiciones :"x" y "y" serán determinadas por una elección random dentro de un rango de valores.

print_user_board_and_impact(): Esta función es la misma que print_board_and_impact() pero para uso del usuario (Es decir, usamos distinto tablero como referencia).

fill_board_user():Esta función es la misma que fill_board() pero para uso del usuario(Es decir, usamos distinto tablero como referencia).

fill_board_ships_user(): Esta función es la misma que fill_board_ships() pero para uso del usuario(Es decir, usamos distinto tablero como referencia).

check_game_over_user(): Esta función es la misma que check_game_over() pero para uso del usuario(Es decir, usamos distinto tablero como referencia).

play_game_didactico() / play_game_facil(): Estos dos modos son prácticamente el mismo. Pues esta función final es lo que nos permitirá juntar todos los métodos en un solo lugar y hacer que el juego aunque suene redundante, se pueda jugar. La única diferencia es que entre el didáctico y el fácil es que en el último no se puede ver el tablero del oponente, únicamente el tuyo. Es un ciclo while que se va a repetir siempre y cuando game_over == 0, es decir que se haya comprobado que aún hay naves sin hundir. Este ciclo while consiste en generar el tablero del oponente, generar el tuyo, preguntar a donde quieres disparar, verificar si se atino o no, despues revisar si el juego termino y si no ha terminado entonces la máquina dispara aleatoriamente a tu tablero y se vuelve a revisar si el juego ya termino. Así hasta que todas las naves se hundan.

menu(): Esto únicamente es una serie de printf, más que nada imprimos cómo está organizado el menú por aparte para no sobrecargar el menú.

main(): Aquí es donde haríamos la parte funcional del menú, al ser opciones que se pueden escoger a través de un simple input únicamente haríamos un switch dentro de un do-while.

Pruebas

Este es el menú principal.

```
Bienvenido a batalla naval
A. Modificar dimension del tablero
-. Seleccionar modo de juego
    a. Didactico
    b. facil
C. Iniciar juego
D. salir
introduce tu eleccion:
```

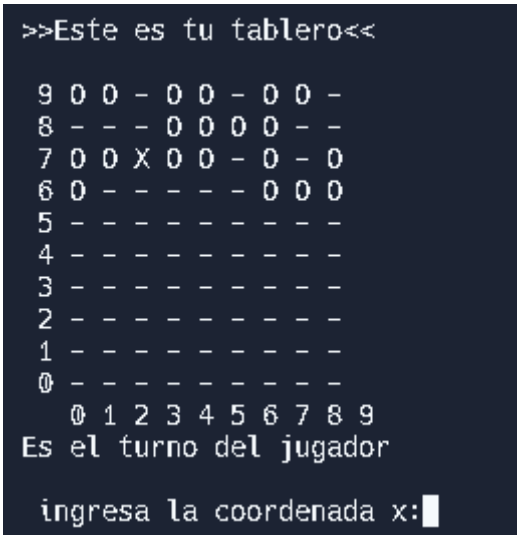
Si presionamos A. Entonces, nos permitirá modificar el tamaño del table. Son **n x n** casillas.

```
introduce tu eleccion:
A
introduce la dimension del tablero
9
```

El modo predeterminado donde se va a iniciar el juego será de manera “Didáctica”, como podemos ver se imprime el tablero propio y el de la computadora.

```
>>Este es tu tablero<<

 9 0 0 - 0 - - 0 0 0
 8 0 - - 0 X 0 0 0 0
 7 0 - 0 0 0 X 0 - 0
 6 0 0 - 0 - - 0 - -
 5 - - - - - - - - -
 4 - - - - - - - - -
 3 - - - - - - - - -
 2 - - - - - - - - -
 1 - - - - - - - - -
 0 - - - - - - - - -
   0 1 2 3 4 5 6 7 8 9
Es el turno del jugador

>>Este es el tablero de tu oponente<<

 9 - - - - 0 0 - - -
 8 0 0 0 - - 0 0 0 -
 7 - 0 0 0 - - - 0 -
 6 0 - - - 0 - 0 0 -
 5 - - - - - - - - -
 4 - - - - - - - - -
 3 - - - - - - - - -
 2 - - - - - - - - -
 1 - - - - - - - - -
 0 - - - - - - - - -
   0 1 2 3 4 5 6 7 8 9
 ingresa la coordenada x:
```

El otro modo que tenemos que seleccionar si queremos jugar con él, es el modo fácil. Como podemos ver de este modo únicamente podemos ver nuestro tablero.



Para poder disparar tenemos que usar la coordenada en “X” y “Y”. Como podemos ver si indicamos la coordenada adecuada y le damos nos da imprime un “le diste” y lo marca en el tablero del oponente, de igual manera se puede fallar el tiro como vemos que lo falló la máquina y le imprimió un “Fallo el tiro”, además de marcarlo en el tablero.

```
 ingresa la coordenada x:7

 ingresa la coordenada y:7

 le diste

turno de la pc

 Fallo el tiro

>>Este es tu tablero<<

 10 0 - - 0 0 0 - - 0 0
 9 X - - 0 0 X 0 0 0 0
 8 0 0 0 0 - - - 0 0 -
 7 0 0 0 0 - 0 0 0 0 0
 6 0 0 - - - - - - 0 -
 5 - - - - - - - - - -
 4 - - - - - - - - - -
 3 - - - - - - - - - -
 2 - X - - - - - - - -
 1 - - - - - - - - - -
 0 - - - - - - - - - -
   0 1 2 3 4 5 6 7 8 9 10
Es el turno del jugador

>>Este es el tablero de tu oponente<<

 10 - 0 0 - - 0 0 0 0 0
 9 - 0 0 0 0 - - 0 0 0
 8 0 0 0 0 0 0 0 0 0 -
 7 0 0 0 0 0 0 - - X 0 0
 6 0 0 0 0 0 0 0 - - 0 -
 5 - - - - - - - - - -
 4 - - - - - - - - - -
 3 - - - - - - - - - -
 2 - - - - - - - - - -
 1 - - - - - - - - - -
 0 - - - - - - - X - - -
   0 1 2 3 4 5 6 7 8 9 10
 ingresa la coordenada x:
```

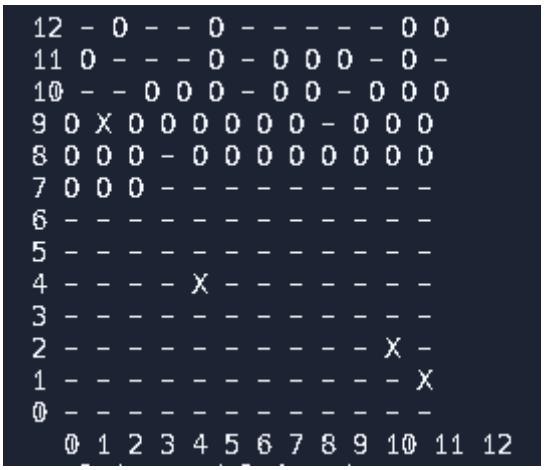
Si llegamos a atinar a todos los barcos del enemigo nos indica que ganamos y nos muestra en cuantos intentos lo hicimos.

```
Ganaste el juego!

SOLO TE TOMO 3 GANAR EL JUEGO!
```

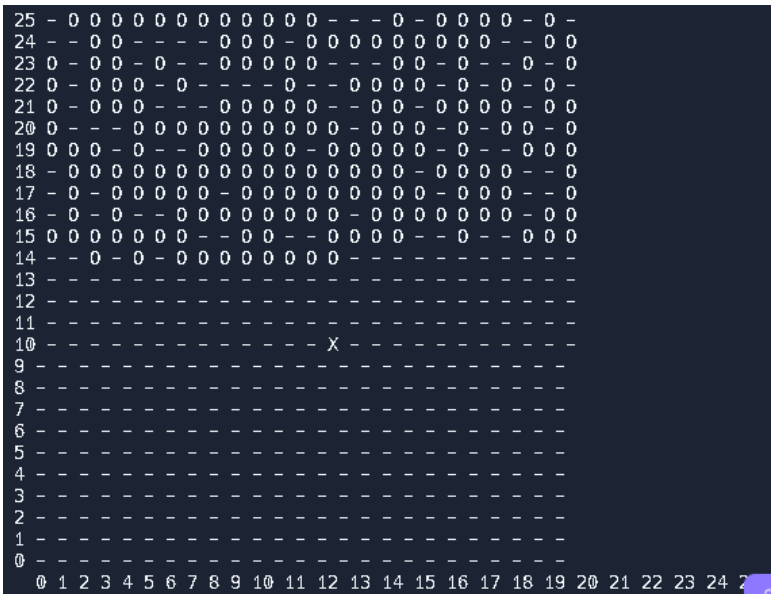
Si una vez jugado queremos salir, una vez presionada la tecla nos dará un agradable mensaje de despedida.

```
Bienvenido a batalla naval
A. Modificar dimension del tablero
-. Seleccionar modo de juego
    a. Didactico
    b. facil
C. Iniciar juego
D. salir
introduce tu eleccion:
D

 Gracias por jugar
```

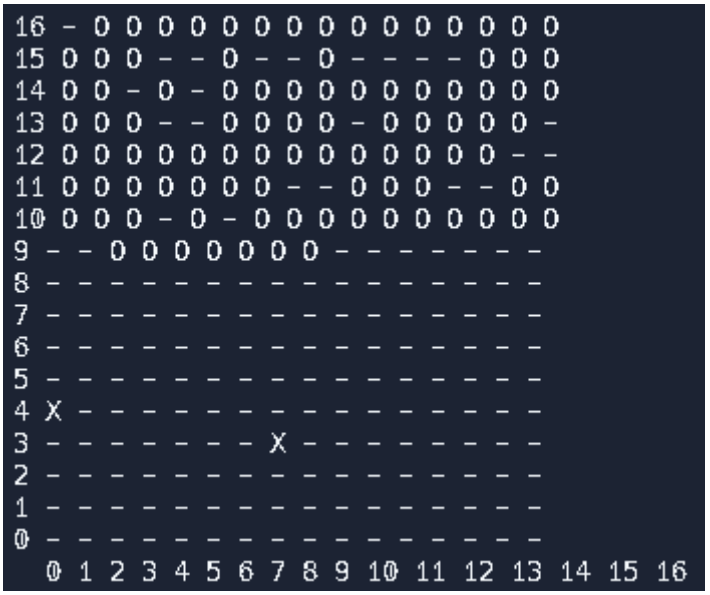
Más pruebas de tableros de distintos tamaños:



De 12 x 12 casillas.



De 25 x 25 casillas.



De 16 x 16 casillas.

Conclusión

En este proyecto en cada función que hicimos nos tuvo que quedar muy en claro el uso de los operadores *,&,-> dentro de los apuntadores, para poder manejarlo dentro de las estructuras repetitivas, pues al tratarse de una matriz no teníamos otra manera de hacer todo que no fuera ciclos for anidados. Además de eso tuvimos que repasar los structs para poder cumplir con todos los requerimientos del programa, aunque se nos habían ocurrido otras maneras de poder solucionar los problema de las celdas y las naves(tal vez un poco más engorrosas, no sabemos pues nunca las desarrollamos completamente al final del dia.

Este examen asincrónico fue un poco complicado, un poco enrevesado y a pesar de que nuestro código podría ser optimizado y mejorado pues no está exento de errores si uno los busca o hay cosas que podrían cambiarse para que sea un proyecto más “limpio” por así decirlo. Estamos orgullosos de que la funcionalidad y los métodos para llegar a esa meta que buscábamos fueran los correctos.

Aparte de esto no tenemos mucho más que concluir, fue un ejercicio importante para que nos quedara claro el uso de los apuntadores y para refrescar nuestra habilidad de programación en C, con esto creo que nos encontramos listos y preparados para poder pasar al próximo tema de la materia sin arrastrar este y tener problemas de comprensión y análisis en un futuro.

Bibliografía

Cervantes, F. (2022). *PROGRAMACIÓN CON MEMORIA DINÁMICA EVALUACIÓN I*. iteso.instructure.com. Recuperado 21 de septiembre de 2022, de https://iteso.instructure.com/courses/28055/pages/ii-manejo-dinamico-de-la-memoria-s6