LABORATORIO Nº3

Programación en C: Matrices y Archivos.

Fecha de envío: Domingo 25 de noviembre, 23:55 hrs. (vía aula virtual)
Modalidad: Trabajo en grupos de a lo más **dos** personas

I. Objetivo.

El objetivo de este laboratorio es evaluar su capacidad para:

- Llevar a cabo un programa completo en el lenguaje de programación C, utilizando todos los elementos básicos que provee el lenguaje, y sus tipos de datos básicos. En particular, se evaluará el manejo de archivos de texto, y de matrices.
- Aplicar las buenas prácticas en programación (orden, comentarios, identificadores representativos).
- Resolver un problema, considerando que se cuenta con el procedimiento en lenguaje natural, que se debe implementar para resolverlos.
- Construir un código fuente totalmente modular (uso de funciones en C).

II. Enunciado.

El clásico juego **Combate Naval**, consiste en la disputa de dos jugadores, cada uno con su flota. El objetivo de cada jugador es hundir toda la flota del contrincante.
Para este laboratorio deberá implementar el combate naval para que el usuario hunda las embarcaciones del computador.

II.1 Entrada.

La información inicial del tablero de juego estará en un archivo de texto, cuyo nombre lo debe indicar el usuario al partir el juego. Su programa deberá abrir la información del archivo e ingresarla a una matriz.
La matriz debe estar definida con un tamaño máximo de **N**x**N** caracteres, donde **N** deberá ser una **constante** igual a **500** en su programa.
El formato del archivo siguiente será el siguiente:

- Primera línea contendrá el valor de **n**, que corresponderá a la dimensión real del tablero que almacena el archivo de texto. El tablero es cuadrado (**n** estará entre 1 y 26).
- La segunda línea contendrá el número de embarcaciones que conforman la flota.
- Las siguientes **n** líneas, poseerán los **n** caracteres de cada fila del tablero. Estos valores pueden ser:
  - **A**: este caracter indica que la matriz tiene **Agua** en esa celda.
  - **N**: este caracter indica que la matriz tiene un trozo de alguna **Nave** en esa celda. Las naves pueden estar en forma vertical u horizontal en el tablero.

Además, existen 3 tamaños posibles: **2** (una *goleta*), **3** (una *fragata*) y **4** (una *corbeta*).

- La última línea del archivo contendrá un entero llamado **tiros**, que corresponderá al máximo de tiros que el jugador podrá realizar.

II.2 El juego.

Luego de cargada la matriz del juego, el usuario empieza a jugar: El programa le debe pedir las coordenadas del disparo: una letra seguida de un número (las letras corresponden a la fila y el número a la columna. Si el disparo dio con una parte de una nave, su matriz deberá almacenar una '**D**' (parte "**D**estruida"), si el disparo cayó al agua, entonces su matriz deberá almacenar una '**E**' (tiro "**E**rróneo"). Debe seguir exactamente estas instrucciones pues se pondrá a su disposición un archivo llamado "**funcionesParaImprimir.h**" que contiene a la función "**ImprimeTablero(int tablero[N][N], int n)**" que imprime el tablero del juego como se mostrará en el punto II.3

Mientras se realiza el juego, el programa también debe mostrarle estadísticas al usuario: "**Total Barcos**:", "**Tiros Disponibles:**".

II.2 Termino del juego.

El usuario ganará si destruye todos los barcos con los tiros disponibles. El usuario perderá si se le acabaron los tiros y aún quedan barcos completos o a medio destruir.
Si el usuario ganó, el programa le deberá mostrar un mensaje indicándoselo al usuario, indicando también el número de partidas ganadas, el número de partidas perdidas, y deberá darle la posibilidad de volver a jugar.
Si el usuario perdió, el programa le deberá mostrar un mensaje indicándoselo al usuario, indicando también el número de partidas ganadas, el número de partidas perdidas, y deberá darle la posibilidad de volver a jugar.
Para estos dos mensajes, en el archivo "**funcionesParaImprimir.h**" también estarán las funciones "**Ganaste()**" y "**Perdiste()**", que imprimirán los mensajes solicitados.

II.2 Interfaz de su programa.

Ya se ha explicado la interacción que su programa debe tener con el usuario. Ahora mediante un ejemplo se le presentará la interfaz completa con la que **su programa DEBE cumplir**.

En la figura 1 aparece lo primero: solicitar el nombre de archivo que contiene al tablero del juego. Observe que el usuario puede digitar el nombre que él desee. Si el archivo no es encontrado o no se puede abrir, debe volver a pedir el nombre del archivo.

En la figura 2 se muestra el contenido del archivo "tablero1.txt.

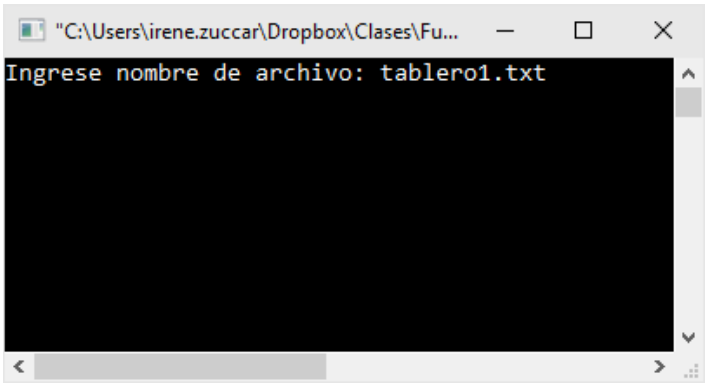

Figura 1

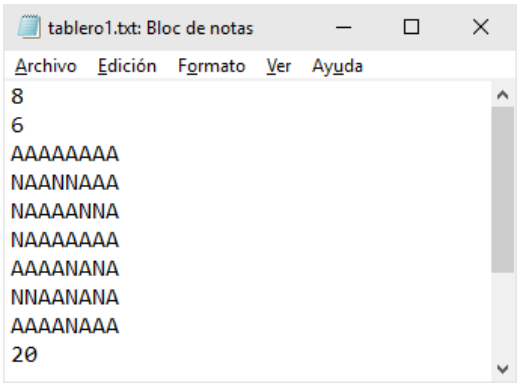

Figura 2

A continuación, debe aparecer el tablero al usuario, dibujado con la función "**ImprimeTablero(tablero, n)**" (para este caso **n**=8). Debe aparecer el total de barcos que se deben hundir (en este caso **6**), el total de tiros disponibles (**20** para este ejemplo), y debe solicitar la fila para el siguiente disparo que realizará. (ver figura 3).

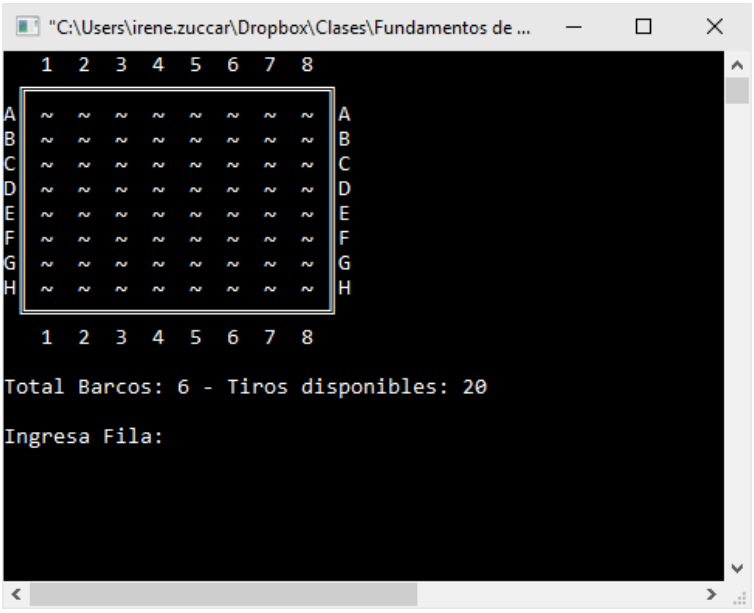

Figura 3

Observe que ya no se ve el texto de la figura 1. Esto se logró con una función llamada "**LimpiaPantalla()**" que también está disponible en el archivo "**funcionesParaImprimir.h**". Para que se pueda usar debe incluir también a la librería "**stdlib.h**" en su código fuente. Luego de que el usuario ingresa la fila, el programa le debe pedir la columna. En la figura 4 se muestra esto.

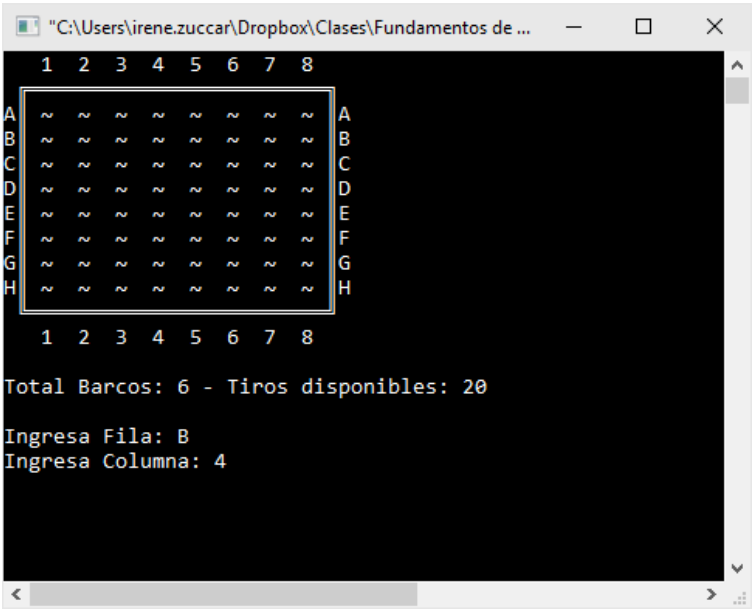

Figura 4

El usuario escogió **B4**. En esa posición hay una parte de la nave, por lo que en su matriz usted debe ingresar una '**D**' en esa posición, para que la función "**ImprimeTablero(tablero, n)**" muestre la parte del barco que destruyó, como se muestra en la figura 5. Observe que se muestra un tiro menos de los disponibles.

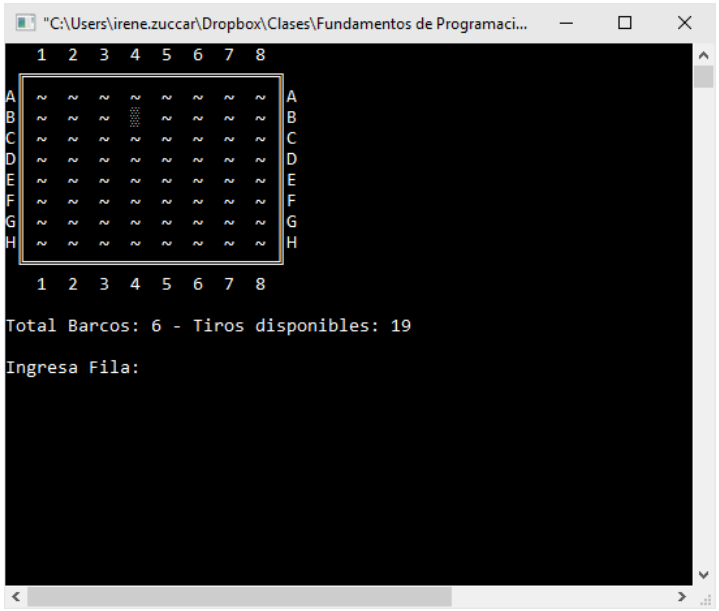

Figura 5

Si el usuario ahora digita **C4.** En esa posición hay solo agua, por lo que en su matriz usted debe ingresar una '**X**' en esa posición, para que la función "**ImprimeTablero(tablero, n)**" muestre un tiro fallido, como se muestra en la figura 5. Observe que nuevamente el programa

descontó un tiro más. Note también que cada vez que se vuelve a imprimir el tablero se "limpia la pantalla". Esto lo debe lograr usted en su programa también.

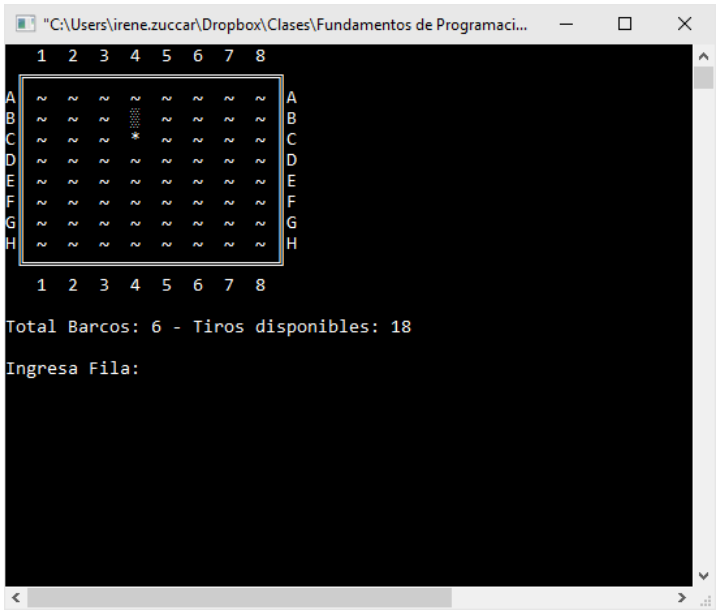

Figura 6

Finalmente, si el usuario gana, la pantalla que debe aparecer es la mostrada en la figura 7, si el usuario pierde, aparecerá la interfaz de la figura 8. Observe que aparecen las estadísticas solicitadas de las partidas anteriores jugadas, y la opción de volver a jugar. Si el usuario digita 'N', su programa se debe cerrar, si digita 'S', deberá volver a la interfaz de la figura 1.

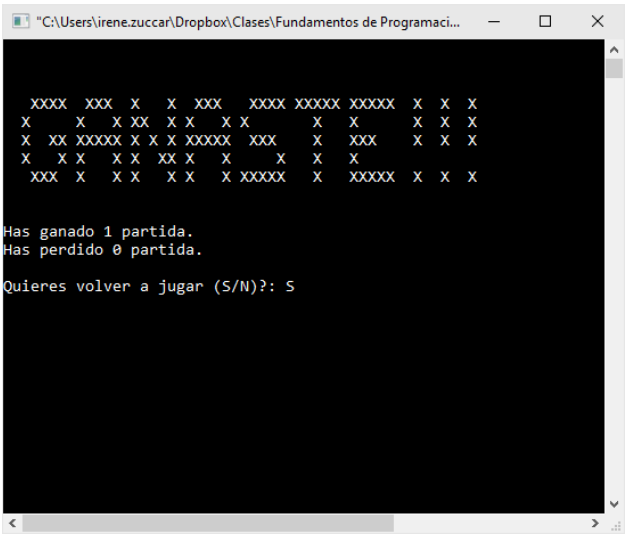

Figura 7

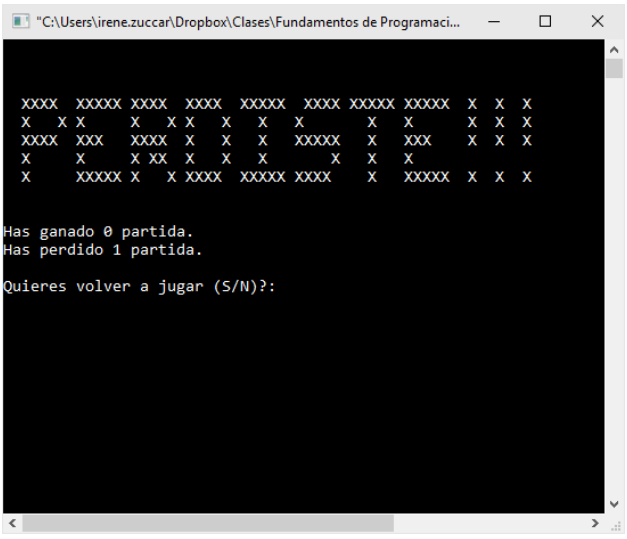

Figura 8

III. Consideraciones en la revisión del código.

1. **Debe seguir exactamente la interfaz explicada en este enunciado.**
2. Debe construir **una función para cada tarea que realice su código**, cuidando los tipos de datos de las entradas y de las salidas.
3. La función **main()** de su código, no debe tener más que llamadas a sus otras funciones: no debe hacer ningún tipo de procesamiento en ella.
4. Debe usar identificadores representativos para sus constantes, variables, parámetros de entrada y para sus funciones.
5. Debe comentar cada una de sus funciones, indicando una descripción de la labor que lleva a cabo, sus entradas, y sus salidas.
6. Su código debe estar correctamente *indentado* (uso de sangrías para cada sub-bloque de instrucciones).
7. Puede trabajar con el IDE y compilador de C, que más le acomode. No obstante lo anterior, su código debe poder ser compilado y ejecutado sin problemas en el establecido en el curso (CodeBlocks) y en Linux.
8. El sistema debe ser **robusto**, se penalizarán los errores no manejados, de cualquier tipo.
9. Al principio del código fuente debe escribir en un comentario: el nombre de la asignatura, el nombre del profesor, los nombres de los integrantes y la fecha.

IV. Sobre la entrega, atrasos y faltas a la ética.

1. Debe subir su código fuente al aula virtual del curso (plataforma de dme.unab.cl), en una casilla que se habilitará especialmente para esto.
2. El nombre de su archivo debe ser el nombre y apellido paterno de cada integrante del grupo. **Ejemplo:** Si trabajaron juntos los alumnos Rosa González y Santiago Fuenzalida, el archivo se debe llamar: **RosaGonzalezSantiagoFuenzalida.c**
3. Este laboratorio **NO SE PUEDE ENTREGAR FUERA DEL PLAZO ESTABLECIDO**, porque no quedaría tiempo para su corrección.
4. Si el programa no se puede ejecutar, tendrá la nota mínima: **1.0**. Si su programa funciona, se evaluará su ejecución, y también el código fuente.
5. Si existe sospecha de copia (con otros compañeros, o desde internet), será interrogado acerca de su trabajo, para aclarar dudas de su entendimiento y autoría. Si se confirma la sospecha, el trabajo será evaluado con nota **1.0**.
6. Las consultas las debe realizar directamente a los ayudantes, de lunes a viernes a los correos especificados o directamente en horario de ayudantía.
7. **IMPORTANTE:** El manejo de archivos de texto será **ENSEÑADO** en el horario de ayudantía, no en cátedra.