U.A.B.C

Facultad de Ingeniería

Tecnologías de Programación

Práctica 3: Funciones

M.C. Pablo M. Navarro Álvarez

OBJETIVO:

Realizar un programa en el que se empleen funciones con el fin de que el alumno comprenda la importancia de la reutilización del código y la organización del mismo. Asimismo se trabajará con parámetros por valor y referencia para identificar su comportamiento y sus principales diferencias.

DESARROLLO:

El alumno desarrollará una simulación contra la computadora del famoso juego "***Battleship***", haciendo uso de los distintos tipos de funciones mostradas en clase. Para iniciar el juego, se requerirá pasar como argumentos en consola (al momento de correr el programa) el número de barcos a jugar (**10 MÁXIMO, 5 MÍNIMO**) y el nombre del jugador.

Nota: Si al ejecutar el programa, el usuario especifica un número de argumentos distinto al necesario (3), se deberá acabar el programa y notificar al usuario del error presentado.

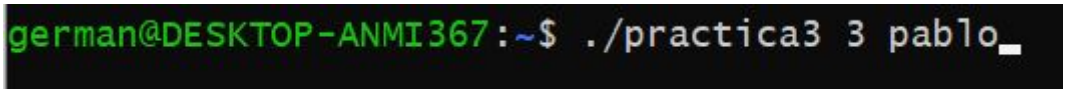


Figura 1.1. **Corrida del programa:** el primer argumento es el nombre del programa; el segundo el número de barcos y el tercero el nombre del jugador.

Antes de iniciar, se deberán repartir los tipos de barcos de acuerdo al número de unidades a utilizar. Se recomienda usar la siguiente estructura:

- 20% para los ***Buques de Carga***
- 30% para los ***Buques de Batalla***
- 40% para los ***Buques Destructores***

Por ejemplo, si se desea jugar con 7 barcos:

- 20% de 7 = 1.4 = 1 Buque de Carga
- 30% de 7 = 2.1 = 2 Buques de Batalla
- 40% de 7 = 3.5 = 4 Buques Destructores

Una vez terminada esta operación, se deberá generar un ***tablero***, que será una matriz de ***12 x 12***, las filas estarán numeradas del ***1 al 12*** y las columnas representadas por letras de la ***A a la L***, finalmente, el interior del tablero se deberá ver de la siguiente manera:

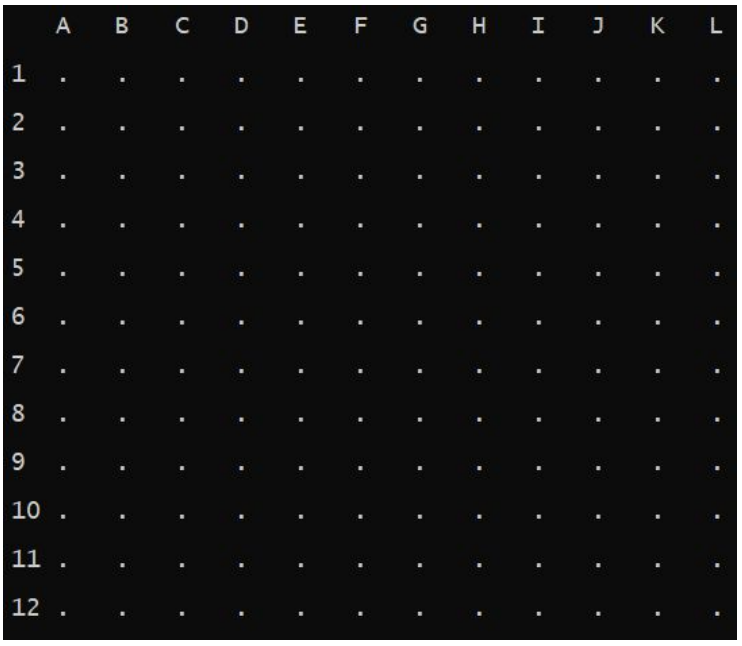


Figura 1.2. Tablero generado automáticamente.

Nota: Aunque el tablero sea representado por letras, no es necesario manejarlas en el código, dando oportunidad de manejar números durante la codificación.

Para generar el tablero se deben tomar en cuenta los siguientes aspectos:

1. Se deben utilizar funciones para generar el tablero, el *main* del programa solamente se dedica a llamar las funciones para manejar el juego, una sugerencia de la firma de la función para generar el tablero es la siguiente:

```
generarTablero(char tablero[][], int filas, int columnas);
```

Nota: Cualquier sugerencia de firma de función es solamente pensada para guiar al alumno, pueden existir más funciones, y la firma podrá cambiar de acuerdo al pensamiento de cada programador.

2. Una vez generado el tablero, se distribuirán los barcos a lo largo y ancho de este, cada barco empieza y termina en una **coordenada específica**, tiene una ***orientación*** y un ***tipo*** que lo representa. La siguiente firma representa la función que distribuirá los barcos:

```
generarCoordenadas(char tablero[][], int filas, int columnas,
int barcosTipos[], char coordenadasInicio[][], char
orientacion[], char coordenadasFinal[][]);
```

Cada parámetro que permita saber la ubicación de un barco (tipos, coordenadas de inicio, orientación) se generarán aleatoriamente, dichos valores permitirán calcular la coordenada final del barco. Además, la función anterior también deberá verificar que las coordenadas de inicio y final se encuentren dentro del tablero.

3. Para solucionar el problema del dibujo de los barcos, se sugiere utilizar arreglos paralelos como los siguientes:

    - El primer arreglo se refiere al tipo de barco que se va a dibujar, cada tipo tiene una longitud específica (que además, representa el número de vidas de cada barco). Este arreglo va a contener un renglón por cada barco especificado.

barcosTipos[]	
Posición	Tipo
0	1
1	3
2	2

    - El segundo arreglo representa la posición donde inicia el barco, los barcos horizontales comienzan con **“|”** y los verticales con **“_”**.

coordenadasInicio[][]		
Posición	Coordenada	
0	A	12
1	F	5
2	D	6

    - El tercer arreglo verifica la orientación del barco.

orientacion[]	
Posición	Orientación
0	V
1	H
2	V

- Por último, el arreglo de coordenadas de término permite conocer el final de cada barco, para los barcos horizontales, el final es representado por “>”; los verticales, por “^”.

coordenadasFinal[][]		
Posición	Coordenada	
0	A	8
1	H	5
2	D	3

El objetivo de esto es poder verificar si una coordenada se encuentra dentro de la matriz, es indispensable que el dibujo del barco no se salga de las dimensiones especificadas, puesto que afectaría al funcionamiento del juego, además, será de gran ayuda para el programador al momento de verificar coordenadas durante el juego.

4. Una vez llenado el arreglo, se dibujarán los barcos de manera horizontal o vertical, pero nunca en diagonal, el dibujo del barco queda a disposición del programador, sin embargo, se recomienda utilizar el siguiente formato:

Figura 1.3. Tipos de barcos y sus orientaciones.

Existen 3 tipos distintos de barcos que pueden utilizarse, los cuales se muestran en el siguiente esquema:

- El ***buque destructor (tipo 3)*** corresponde al dibujo de 1 unidad de largo (***guiones o pipes***).
- El ***buque de batalla (tipo 2)*** corresponde al dibujo de 2 unidades de largo.
- El ***buque de carga (tipo 1)*** corresponde al dibujo de 3 unidades de largo.

5. Si implementamos este funcionamiento al tablero dibujado, se tendría la siguiente función y dibujo de barcos final:

```
dibujarBarcos(char tablero[][], int filas, int columnas, int barcosTipos[], char coordenadasInicio[][], char orientacion[], char coordenadasFinal[][]);
```

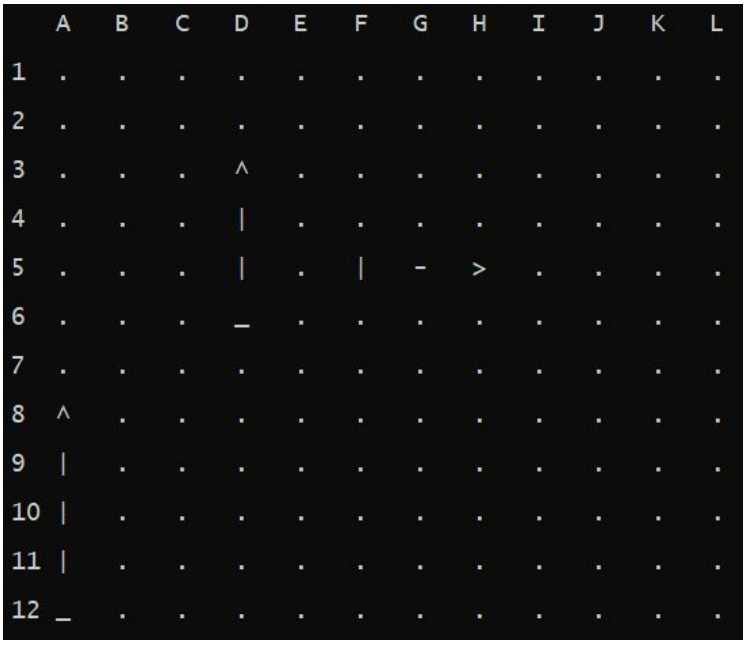


Figura 1.4. Tablero final con los barcos dibujados.

Juego:

Cuando inicie el juego, se le mostrará al jugador su ***nombre***, ***movimientos restantes***, ***número de barcos hundidos*** hasta ahora, ***número de barcos restantes***, ***coordenadas fallidas***, ***coordenadas atinadas*** y una leyenda que pida la coordenada donde se piense que el oponente tenga alguno de sus buques.

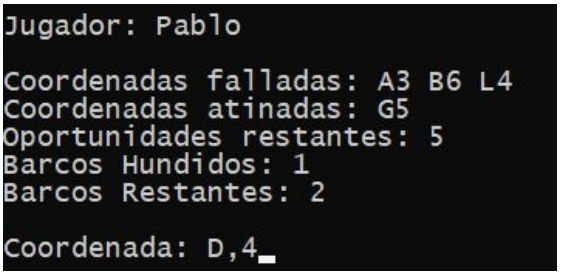


Figura 1.7. Pantalla del jugador, como se puede apreciar, se muestran todos los datos necesarios para guiarse durante el juego, haciendo más fácil la detección de barcos.

Para las coordenadas, primero se escribirá la letra, seguido del número, separados por coma. En caso de que exista un buque en esa coordenada (ya sea las orillas o cualquier parte de él), se deberá mostrar un mensaje que diga ***"Me diste"***, caso contrario, ***"Fallaste"***.

Figura 1.5. Coordenada provista por el jugador.

Si al especificar una coordenada se escribe un -1 en cualquiera de los dos ejes, se mostrará el tablero del oponente (para fines de revisión). La manera en que se puede hacer esto es simplemente imprimir en pantalla el tablero con una función como la siguiente:

```
imprimirTablero(char tablero[][], int filas, int columnas);
```

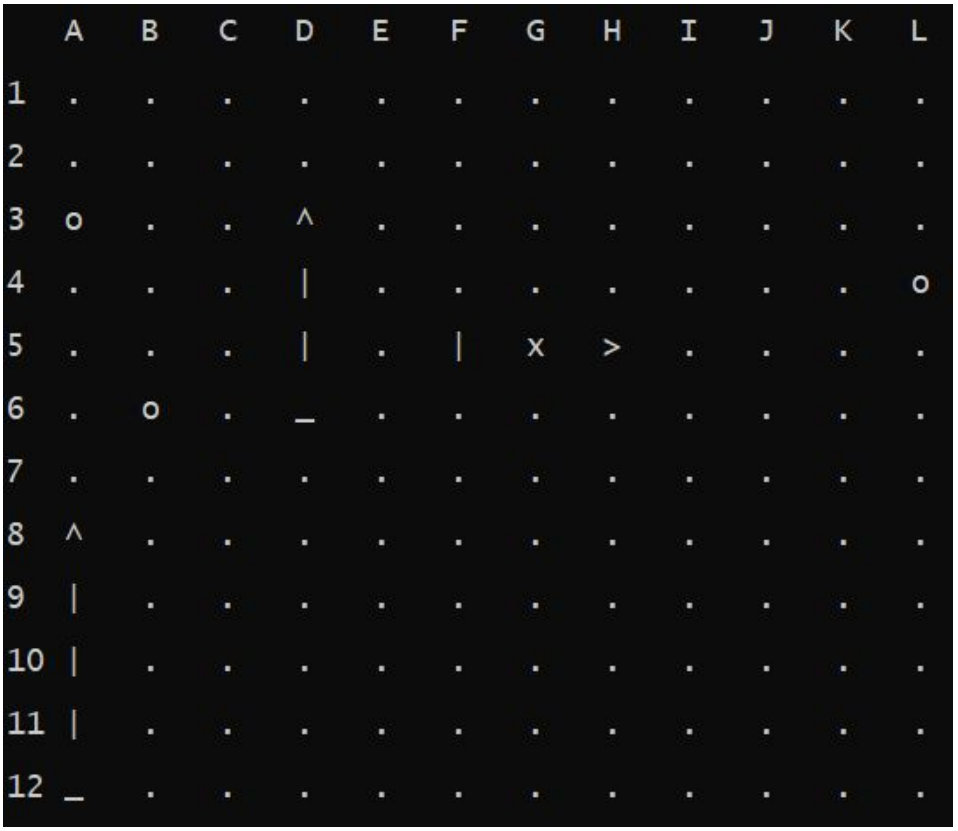


Figura 1.6. Tablero de la computadora, el barco atacado abarca 3 casillas, cualquiera de esas casillas podrá ser tocada e indicará que se le ha dado, la ***"x"*** representa una bandera. Las ***"o"*** representan los fallos.

En caso de que el jugador atine a cualquier parte de un barco, se le asignará una bandera que indique que se le ha dado al barco en cuestión (representado por una ***x***), cada barco tiene un número determinado de vidas (indicado por el número de guiones), si un barco pierde todas sus vidas, se "hundirá". El otro caso representa cuando se falla al especificar una coordenada, en tal situación, se reemplazará la coordenada por la letra ***"o"***.

Nota: Decir que el barco se hunda solamente significa que ya no existirá en los arreglos de coordenadas, pero seguirá apareciendo en el tablero.

La computadora tiene un número de vidas finito representado por el número de vidas en total de los barcos, es decir, se suma el número de guiones que tiene cada barco. Siguiendo el ejemplo de 3 barcos:

- 1 ***buque destructor*** = 3 vidas
- 1 ***buque de batalla*** = 4 vidas
- 1 ***buque de carga*** = 5 vidas
- Total = 12 vidas

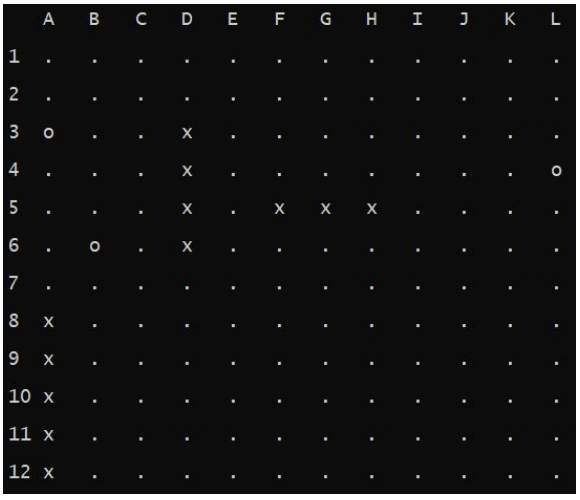
El jugador gana cuando haya hundido toda la flota del oponente, pero pierde si falla un número de veces determinado por la siguiente fórmula:

$$movimientosRestantes = (vidasComputadora * numBarcos) / 2$$

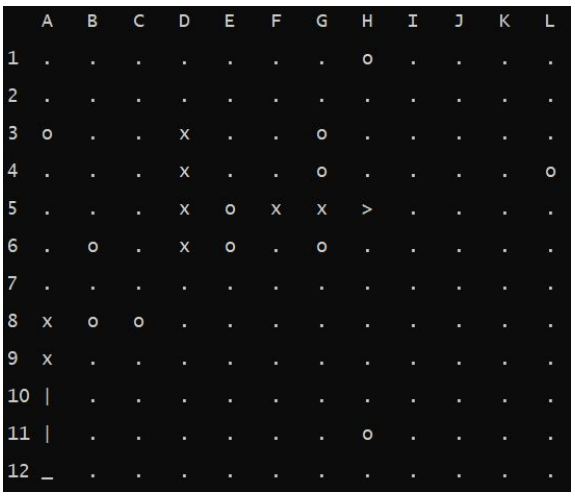
Los movimientos restantes se decrementarán cuando el jugador falle al especificar una coordenada, la manera en que el jugador puede verificar esto es recorriendo los arreglos de coordenadas, tipos y orientaciones para determinar si el punto provisto está entre la primera y última coordenada (recuerde que se están manejando números, así que comprobar que una coordenada le dio a un barco no debería ser un reto para el alumno), la firma de la función para verificar el daño se sugiere de la siguiente manera:

```
verificarGolpe(char tablero[][], int filas, int columnas, int barcosTipos[], char coordenadasInicio[][], char orientacion[], char coordenadasFinal[][], char coordenadaJugada[][], int *vidasOponente, int *movimientosRestantes);
```

El juego termina cuando el tablero se vean cualquiera de los siguientes dos casos:



Caso 1: Todos los barcos han sido hundidos



Caso 2: Todas las oportunidades restantes han sido agotadas

En la pantalla del jugador, simplemente se pondrá un mensaje de ***"ganaste"*** o bien, ***"perdiste"***.