

**Paradigmas de Programación
Paradigma imperativo Procedural
Manual de Usuario**

FELIPE MATURANA GUERRA

Profesor:
Roberto González

Ayudantes:
Giovanni Benussi
Mauricio Rojas
Esteban Contardo

Santiago - Chile
2-2016

TABLA DE CONTENIDOS

Tabla de Contenidos.....	I
Índice de Figuras	I
CAPÍTULO 1. introducción.....	1
1.1 Reglas.....	1
1.2 Cómo jugar.....	1
CAPÍTULO 2. Procesos de Compilación	3
2.1 Cómo compilar y ejecutar	3
CAPÍTULO 3. Funcionalidades ofrecidas	4
CAPÍTULO 4. Resultados de pruebas	6

ÍNDICE DE FIGURAS

Figura 2-1: Abrir ventana de comandos	3
Figura 2-2: Compilar MINGW32	3
Figura 3-1: Funcionalidades del proyecto	4
Figura 4-1: Prueba de creación tablero	7
Figura 4-2: Prueba de posicionamiento barcos Jugador.....	7
Figura 4-3: Prueba de posicionamiento superpuesto.....	8
Figura 4-4: Prueba de ataque en Fila Enemigo a Jugador.....	8
Figura 4-5: Ataque en área 3x3 de jugador a Enemigo	9

CAPÍTULO 1. INTRODUCCIÓN

Battleship es un juego de mesa también conocido como “Batalla de mesa”, es un juego de ingenio y estrategia el cual consiste en un tablero dividido en dos mitades, una de estas mitades corresponde al jugador y la otra al enemigo. Históricamente este juego se jugaba con lápiz y papel tras ser comercializado en 1943 por Milton Bradley Company, sin embargo, no fue hasta 1967 que fue posible conocerlo tal y como lo conocemos, en tres dimensiones. El objetivo del juego es derribar todos los barcos enemigos antes que él nos destruya los nuestros, aquí es donde se pone a prueba nuestro ingenio.

Cada jugador puede ver en su totalidad su tablero, sin embargo, el tablero enemigo sólo se vislumbrarán los lugares donde se han efectuado jugadas (disparos), es decir, antes de comenzar a jugar veremos sólo espacios en blanco en el tablero enemigo.

Antes de comenzar a jugar es necesario posicionar los barcos que tenemos a nuestra disposición en las casillas dentro de nuestro tablero, los barcos pueden estar orientados de forma vertical u horizontal nunca en diagonal, en esta implementación un jugador puede tener un barco de diferencia con respecto al otro, con 1 barco como mínimo, a diferencia de su implementación original donde cada jugador debe tener la misma cantidad de barcos.

1.1 REGLAS

Un jugador no podrá realizar dos jugadas en forma consecutiva.

Un jugador no podrá realizar jugadas sobre su mismo tablero ni en una posición donde ya se ha realizado un disparo.

La dimensión del tablero en general, deberá tener una cantidad par de columnas ya que éste será dividido en dos y a cada jugador le corresponderá una misma cantidad de celdas.

1.2 CÓMO JUGAR

Una vez que se decida quién juega primero, se jugará por turnos, señalando la posición en la cual se realizará un disparo y, en esta implementación, desde qué barco se realizará el disparo (y si es un ataque especial) hasta que alguno de los jugadores no le queden barcos disponibles.

CAPÍTULO 2. PROCESOS DE COMPILACIÓN

2.1 CÓMO COMPILAR Y EJECUTAR

En este proyecto se utilizó la herramienta “make”, la cual es una herramienta de gestión de dependencias entre los archivos que componen el código fuente de nuestro programa de tal forma que dirige las directivas de compilación. El compilador usado es “MINGW32”, el proyecto fue realizado en Windows siguiendo los estándares necesarios y propuestos en el proyecto.

Es necesario ubicar la carpeta raíz donde se encuentran los archivos y una vez dentro hacer un click derecho mientras se mantiene la tecla shift presionada, de ésta forma abriremos la ventana de comandos con la dirección de esta carpeta.

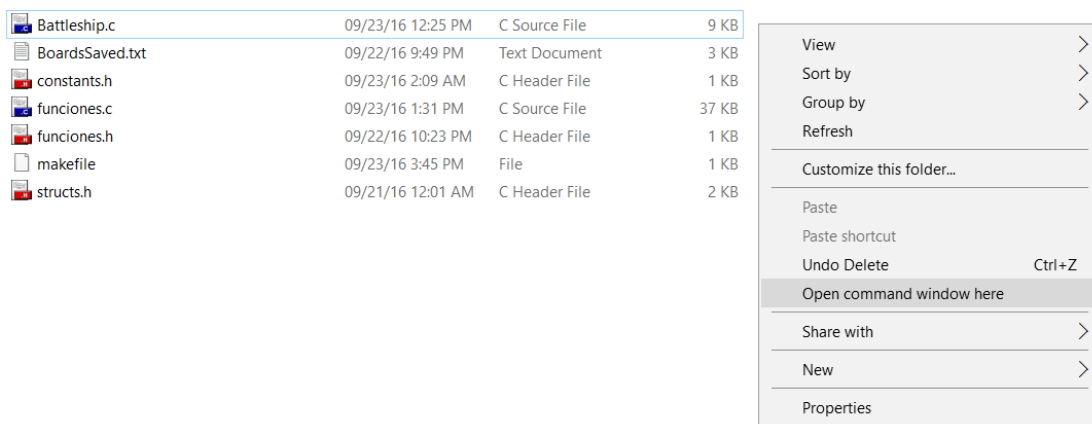


Figura 2-1: Abrir ventana de comandos

En la ventana de comandos debemos escribir “mingw32-make” para compilar a través del archivo makefile y generar de forma automática con todas las opciones que se encuentren en el archivo que dirige nuestra compilación.

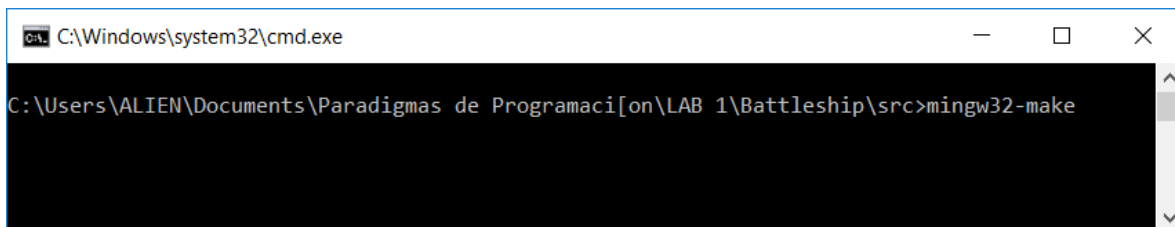


Figura 2-2: Compilar MINGW32

CAPÍTULO 3. FUNCIONALIDADES OFRECIDAS

El proyecto posee las siguientes funcionalidades:

```
Board* createBoard(int n, int m, Params params, code *statusCode);
void print(Board *b, int showComplete, code *statusCode);
int checkBoard(Board *b, code *statusCode);
void saveBoard(Board *b, int *id, code *statusCode);
Board * loadBoard(int id, code * statusCode);
void putShip(Board *b, Position p, Ship s, code *statusCode);
int play(Board *b, Ship * ship, Position * pArray, code *statusCode);

Position setPosition(int Fila, int Columna);
Ship cfgShips(int Largo, int Ancho, int ATKespecial, char charBarco, int DEF);
int BarcoAtacado(Position p, Comandante comandante, Board *b);
int * generateID();
Params setParams(int Dificultad, code *statusCode);
Position * AtaquesMasivos(Ship s, Position p, Board * b);
void destroyBoard(Board **b);
int getScore(Game g);
```

Figura 3-1: Funcionalidades del proyecto

Las primeras 7 funcionalidades son las requeridas de forma explícita por el proyecto planteado por la asignatura, a continuación, existen funcionalidades que surgieron de acuerdo al diseño propio que se hizo del proyecto.

Funcionalidades Obligatorias:

1. CreateBoard recibe los parámetros n y m las cuales son las dimensiones de fila y columna respectivamente, además de Params, estructura la cual contiene la dificultad de la partida además de la cantidad que tiene el enemigo de cada tipo de barco en su mitad de tablero.
2. Print imprime el tablero de forma parcial o completa según el parámetro entero showComplete 0 para parcial y 1 para completo.
3. CheckBoard devuelve un “booleano” si el board que recibe como argumento cumple con las condiciones necesarias para ser considerado válido. Número par de columnas, dimensiones consistentes y mayores que cero, etc.
4. Save y Load permiten guardar y cargar las características de un tablero para luego checkearlo y jugar sobre él.

5. Putship permite posicionar un barco que recibe por argumento en una posición p, esta función permite posicionar los barcos que cumplan las condiciones necesarias, por ejemplo, que no se superpongan o que sea la mitad correspondiente del tablero.
6. Play esta es la función que le da la magia al juego y permite realizar jugadas en las posiciones que contenga pArray, puede ser una o las necesarias. pArray es generado a través de la función AtaquesMasivos la cual retorna un puntero a posiciones (pArray) con todas las posiciones donde se realizarán ataques. La posición pArray[0] contiene la cantidad de disparos que realizará, se pidió explícitamente que esta función retornara un entero para las ocasiones en que éste impactara sin embargo para los disparos múltiples, éste requerimiento carece de sentido ya que las funciones sólo tienen un retorno y en éste caso se necesitaban n retornos donde n es la cantidad de disparos, es por esto que las líneas donde corresponde retorno se comentaron, de ser necesario descomentarlas para verificar su uso. 0 si cayó al agua, 1 si le dio sin destruirle, 2 si le destruyó y 3 si éste poseía defensa.

Funcionalidades Extras y adicionales.

- A. setPosition: Retorna una posición de acuerdo a la fila y columna que recibe como argumentos.
- B. cfgShips: Retorna un barco con la configuración declarada en los argumentos de la función.
- C. BarcoAtacado: Es una función en la cual se implementó un algoritmo de búsqueda lineal para buscar el barco el número correspondiente al puntero de barcos que contiene la estructura Board, del barco que se encuentra en la “posición p” de la celda del tablero.
- D. GenerateID: Retorna un entero correspondiente al ID necesario que se utiliza para guardar el tablero, este ID es generado de forma aleatoria.
- E. setParams: recibe la dificultad con la cual se va a jugar y de acuerdo a esto, selecciona la cantidad de cada barco del enemigo las cuales se guarda en CantidadXTipoBarco

contenido en la estructura Params, por ejemplo: La dificultad 1: Contiene 2 barcos normales, 1 barco acorazado, 1 porta aviones, 1 submarino y 1 buque pirata.

- F. Ataque Masivo: retorna un puntero a position (pArray) el cual es recibido por la función play ya que los barcos poseen ataques especiales como por ejemplo el porta aviones tiene un ataque por columnas el cual es retornado como pArray el cual contiene todas las posiciones a atacar, el primer valor de pArray tanto para fila y columna corresponde a la cantidad de disparos (pArray[0]) y los siguientes valores son las posiciones donde se realizarán disparos.
- G. destroyBoard: realiza una liberación de memoria del tablero que reciba como parámetro.
- H. getScore: obtiene el puntaje de una partida (estructura Game).

CAPÍTULO 4. RESULTADOS DE PRUEBAS

Para ilustrar de mejor forma este capítulo se realizarán las siguientes pruebas, estas pruebas estarán contenidas en el archivo main (Battleship.c):

1. Creación de un tablero y posicionar los barcos enemigos.
2. Verificar y Guardar el tablero.
3. Posicionar los barcos del jugador: probar las excepciones (Barcos superpuestos, fuera del tablero, etc)
4. Cargar un tablero que no se encuentre e imprimirlo.
5. Probar disparos (función play) y las excepciones (Jugadas por el mismo jugador de forma consecutiva, disparos sobre el mismo tablero, etc).

```

C:\Windows\system32\cmd.exe
*****FIN Creacion de Tablero*****
| | | | *|N|N|N| |
| | | | *|N|N|N| |
| | | | *|A|A|A|A|
| | | | *|P|P|P| |
| | | | *|S|S|S| |
| | | | *|B|B| |
| | | | *| | |
| | | | *| | |
| | | | *| | |
| | | | *| | |
*****INICIO Verificacion de Tablero Creado*****
El tablero cuenta con todas las caracteristicas para ser considerado valido
*****FIN Verificacion de Tablero Creado*****
El barco es propiedad de: E
*****INICIO Guardar Tablero Creado*****
Se ha Guardado de forma correcta el tablero con ID: 10690
*****FIN Guardar Tablero Creado*****

```

Figura 4-1: Prueba de creación tablero

No se encontraron problemas al momento de crear el tablero de dimensión NxM.

```

C:\Windows\system32\cmd.exe
*****INICIO Cargar Tablero*****
El tablero ID: 8563 no existe.
*****FIN Cargar Tablero*****
*****INICIO Verificacion de Tablero Cargado*****
No se encuentra el tablero
*****FIN Verificacion de Tablero Cargado*****
No se ha encontrado un tablero para imprimir.
*****INICIO Posicionamiento en 0,1 BARCO: P*****
El barco es propiedad de: J
Fila 0 ///// Columna 1
Fila 1 ///// Columna 1
Fila 2 ///// Columna 1
|P| | | *|N|N|N| |
|P| | | *|N|N|N| |
|P| | | *|A|A|A|A|
| | | | *|P|P|P| |
| | | | *|S|S|S| |
| | | | *|B|B| |
| | | | *| | |
| | | | *| | |
| | | | *| | |
| | | | *| | |

```

Figura 4-2: Prueba de posicionamiento barcos Jugador

No se encontraron problemas al momento de Posicionar en el tablero un barco Porta Avion en la posición (0,1).

```

Select C:\Windows\system32\cmd.exe
*****INICIO Posicionamiento en 6,2 BARCO: A*****
| P|N| |A|*|N|N|N| | |
| P|N|B|A|*|N|N|N| |
| P|N|B|A|*|A|A|A|A| |
| | |S|A|*|P|P|P| |
| | |S| |*| |S|S|S| |
|A|S| |*| |B|B| |
|A| |*| | | | |
|A| |*| | | | |
|A| |*| | | | |
| | |*| | | | |

Hay 6 Barcos JUGADOR
*****FIN Posicionamiento en 6,2 BARCO: A*****
*****INICIO Posicionamiento en 3,2 BARCO: N*****
La posicion se encuentra ocupada, posicionamiento FALLIDO
Fila 1 //// Columna 2
Fila 0 //// Columna 2
Fila 2 //// Columna 2
| P|N| |A|*|N|N|N| | |
| P|N|B|A|*|N|N|N| |
| P|N|B|A|*|A|A|A|A| |
| | |S|A|*|P|P|P| |
| | |S| |*| |S|S|S| |
|A|S| |*| |B|B| |
|A| |*| | | | |
|A| |*| | | | |
| | |*| | | | |

```

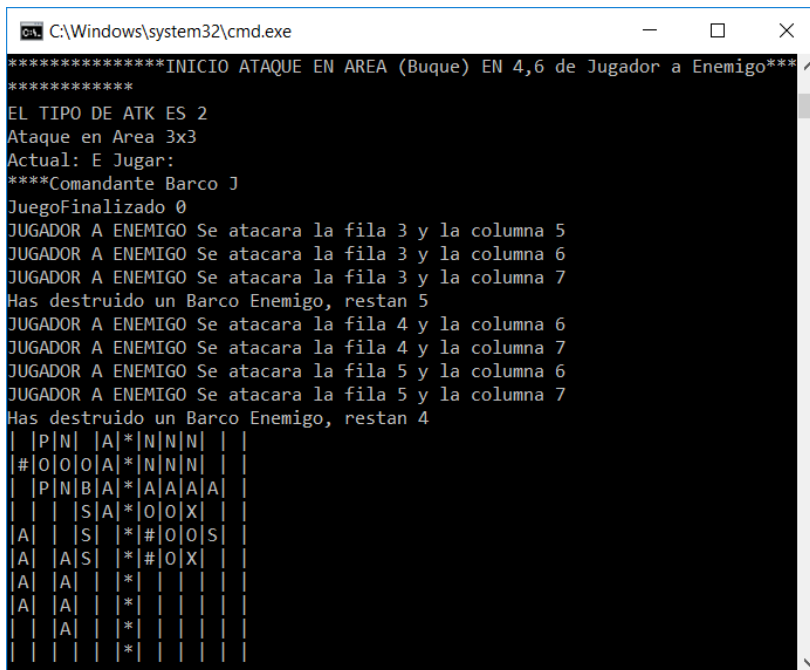
Figura 4-3: Prueba de posicionamiento superpuesto

```

C:\Windows\system32\cmd.exe
*****INICIO DE ATAQUES*****
Barcos Enemigos 6 Actual 6 Barcos Jugador 7 Actual 7
*****INICIO ATAQUE FILA (Porta avion) EN 1,1 de Enemigo a Jugador
*****
EL TIPO DE ATK ES 3
****Comandante Barco E
JuegoFinalizado 0
Enemigo a Jugador Se atacara la fila 1 y la columna 1
BUSQUEDAaaaaaaaaaaaaa
Enemigo a Jugador Se atacara la fila 1 y la columna 2
BUSQUEDAaaaaaaaaaaaaa
Enemigo a Jugador Se atacara la fila 1 y la columna 3
BUSQUEDAaaaaaaaaaaaaa
Enemigo a Jugador Se atacara la fila 1 y la columna 4
Le has dado a un barco que posee defensa, le restan 1 puntos de defensa
Le has dado a un barco que posee defensa, le restan 0 puntos de defensa
No puedes atacar tu mismo tablero
No puedes atacar tu mismo tablero
No puedes atacar tu mismo tablero
| P|N| |A|*|N|N|N| | |
|#|O|O|O|A|*|N|N|N| |
| P|N|B|A|*|A|A|A|A| |
| | |S|A|*|P|P|P| |
|A| |S| |*| |S|S|S| |
|A| |A|S| |*| |B|B| |
|A| |A| |*| | | | |

```

Figura 4-4: Prueba de ataque en Fila Enemigo a Jugador



```

C:\Windows\system32\cmd.exe
*****INICIO ATAQUE EN AREA (Buque) EN 4,6 de Jugador a Enemigo*****
*****
EL TIPO DE ATK ES 2
Ataque en Area 3x3
Actual: E Jugar:
***Comandante Barco J
JuegoFinalizado 0
JUGADOR A ENEMIGO Se atacara la fila 3 y la columna 5
JUGADOR A ENEMIGO Se atacara la fila 3 y la columna 6
JUGADOR A ENEMIGO Se atacara la fila 3 y la columna 7
Has destruido un Barco Enemigo, restan 5
JUGADOR A ENEMIGO Se atacara la fila 4 y la columna 6
JUGADOR A ENEMIGO Se atacara la fila 4 y la columna 7
JUGADOR A ENEMIGO Se atacara la fila 5 y la columna 6
JUGADOR A ENEMIGO Se atacara la fila 5 y la columna 7
Has destruido un Barco Enemigo, restan 4
| |P|N| |A|*|N|N|N| | |
|#|O|O|O|A|*|N|N|N| | |
| |P|N|B|A|*|A|A|A|A| |
| | |S|A|*|O|O|X| | |
|A| |S| |*|#|O|O|S| |
|A| |A|S| |*|#|O|X| | |
|A| |A| |*| | | | |
|A| |A| |*| | | | |
| | |A| |*| | | | |
| | | |*| | | | |

```

Figura 4-5: Ataque en área 3x3 de jugador a Enemigo

Todas las pruebas realizadas fueron exitosas ya que en el diseño y la implementación fueron considerados todos y cada uno de éstos casos. En caso de fallos es necesario verificar, en primera instancia, el compilador que se está utilizando ya que a pesar de haber seguido todos los parámetros de estandarización y haber utilizado sólo las librerías estándares y de petición de memoria, es posible que en otros ambientes no compile de forma correcta, a pesar de que el código fue probado en ambiente Linux (Kali) y Windows, existen casos de proyectos que no compilan de forma correcta en algunos ambientes o distintas distribuciones. Principalmente pueden existir problemas en la asignación de memoria.

Existe una potencial falla la cual consiste en que el jugador puede colocar barcos incluso cuando ya se ha comenzado a disparar. Para suplir esta falla basta con poner un entero en la estructura board que indique con un 1 si el juego ya ha comenzado y ésta condición verificarla en la función putShip.