**UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA**
**DEPARTAMENTO DE INFORMÁTICA**
**CAMPUS VALPARAÍSO Y SAN JOAQUÍN**

# LENGUAJES DE PROGRAMACIÓN
# TAREA EXTRA

## 1. Objetivo

El estudiante debe analiza e implementa las reglas de un conocido juego utilizando el lenguaje de programación C, mediante el uso de listas enlazadas, punteros a funciones y punteros a void, entre otros elementos del lenguaje.

## 2. Descripción general del Trabajo

El trabajo a realizar consiste en la simulación del juego de la batalla naval, pero para un único jugador. Éste deberá intentar hundir los distintos barcos en el tablero utilizando una serie limitada de disparos con diversas áreas de efecto.

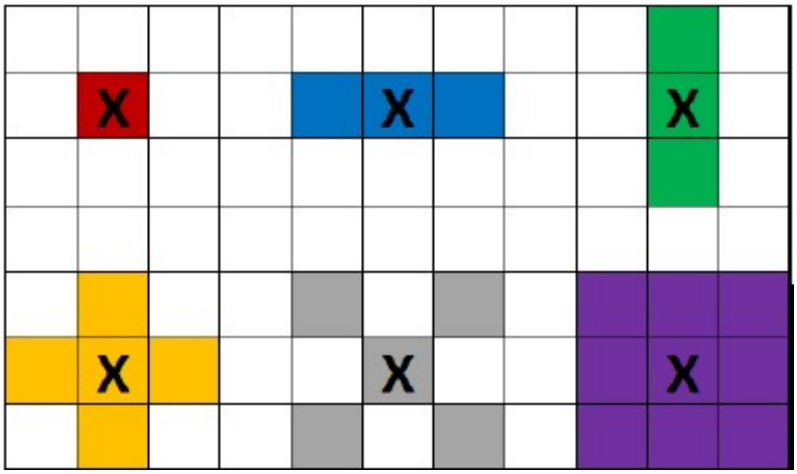
Los barcos estarán compuestos por varias partes, cada una posicionada en coordenadas específicas. El jugador ganará un punto cada vez que le acierte a una parte de barco a la cual no le había acertado antes. En el tablero se encontrarán algunos elementos *bonus*, los cuales permitirán disparos extras.

El juego terminará cuando el jugador se queda sin disparos (perdió) o destruye todos los barcos (ganó).

## 3. Requerimientos

El programa debe recibir un archivo de entrada el cual entregará la cantidad de disparos de cada tipo y el *layout* del tablero a jugar (siendo éste de 10x10 caracteres), que contendrá la posición de las partes de barcos y los bonus. Se deben crear partes de barcos, bonus y disparos utilizando las estructuras que se presentan más adelante.

Existen 6 tipos de disparos, los cuales se muestran en la figura 1, la X muestra el punto que es ingresado en coordenadas por el jugador.



Tipos de disparos:

- Simple (color rojo)
- Horizontal (color azul)
- Vertical (color verde)
- Cruz (color amarillo)
- Equis (color gris)
- Cuadrado (color morado

Figura 1: Ejemplo de *layout* del tablero y descripción de los tipos de disparos

El tablero inicialmente debe aparecer vacío, sólo con las coordenadas en los costados para guiarse, para luego ir mostrando gráficamente, sólo mediante caracteres, el efecto de los disparos en el tablero, uno a la vez. Se debe mostrar el puntaje actual del jugador y la cantidad de disparos de cada tipo que le quedan.

## 4. Simbología

El tablero será una matriz de caracteres de 10x10, que utilizará la siguiente simbología:
0: Espacio vacío.
1: Munición para disparos simples.
2: Munición para disparos horizontales.
3: Munición para disparos verticales.
4: Munición para disparos en cruz.
5: Munición para disparos en equis.
6: Munición para disparos cuadrados.
B: Parte de barco.

A modo de ejemplo, el archivo de entrada, que se debe llamar *entrada.txt*, puede verse como sigue:

```
2
3
3
1
1
1
0 0 0 0 0 0 0 0 0 0
0 0 B B B B B 0 0 0
0 0 3 0 0 0 0 0 0 0
0 0 0 0 0 0 B 0 0 0
0 B B B 1 0 B 0 0 0
0 0 0 0 0 0 B 0 B B
0 0 0 0 0 0 0 0 B 0
0 0 0 0 0 0 0 0 B 0
0 0 0 B B 0 0 0 B 0
0 0 0 0 0 0 0 0 B 0
```

donde las seis primeras 6 líneas indican la cantidad de disparos iniciales de cada tipo que el jugador poseerá, según el orden dado en la figura 1 (rojo, azul, verde, amarillo, gris, morado).

## 5. Estructuras Básicas propuestas

Se proponen las siguientes estructuras como mínimas para el código a desarrollar.

```
struct Disparo
{
  void (*Utilizar)(posicionX, posicionY);
  int tipo;
}

struct Jugador
{
  int Puntaje; //inicialmente igual a cero
  Disparo[ ] disparos; //conjunto de disparos a probar, por parte del Jugador
}
```

```
struct ParteBarco
{
  void (*acierto)(void *);
  int destruida;
}

struct Bonus
{
  void (*activar)(void *);
  int tipo;
  int cantidad;
}
```

Además, considerar que se debe implementar una lista enlazada simple para la creación y manejo de elementos del tablero (partes de Barco y *Bonus*), la que debe manejar un puntero a *void* al elemento correspondiente. No está permitido el uso de bibliotecas externas para el manejo de listas.

Como la lista enlazada no es parte del programa en sí, debe estar en archivos separados. Cada definición de estructuras y cabeceras de funciones deben estar en un archivo con extensión *.h*, mientras que cada función de elementos debe estar en archivos con extensión *.c*.

Sólo de manera justificada, cada estudiante puede agregar algún elemento adicional, siempre que no evite el uso de las funcionalidades a evaluar con la tarea (listas enlazadas, punteros a funciones y punteros a void).

## 6. Archivos a Entregar

Los archivos mínimos a entregar serian:

Disparos.c (funciones de configuración de disparos).
Disparos.h (Archivo con estructuras y prototipos de funciones de disparos).

PartesDeBarcos.c (funciones de configuración de partes de barcos).
PartesDeBarcos.h (Archivo con estructuras y prototipos de funciones de partes de barcos).

Bonus.c (funciones de configuración de bonos).
Bonus.h (Archivo con estructuras y prototipos de funciones de bonos).

main.c (Archivo con programa principal).

MAKEFILE (Archivo de compilación automática). // Si el makefile no está bien realizado, la tarea no se revisará.

listaElementos.c (Archivo con funciones de lista enlazada).
listaElementos.h (Archivo con estructuras y prototipos de funciones de lista enlazada).

Se pueden crear más archivos si lo estiman necesario, mientras tengan los nombrados anteriormente entre los enviados.

## 7. Sobre Entrega

El código debe venir indentado y sin *warnings*. y la entrega debe realizarse en tar.gz y debe llevar el nombre el estudiante (ej.: JoseLuisMartiLara.tar.gz). Debe incluirse un *readme.txt*, con instrucciones para la compilación y ejecución del programa.

La entrega debe hacer vía aula, a más tarde el miércoles 19 de julio, 23:59 horas.

**Entrega mínima**: el programa ejecutando todo lo pedido → 55 puntos
Adicionales:

- Documentación completa: 15 puntos
- Mejor interfaz del tablero (colores): 20 puntos
- Liberación de la memoria una vez que se ha dejado de usar ciertas estructuras, incluyendo al finalizar el programa: 10 puntos