Ejercicio evaluable 1: Estructuras y tipos enumerados

A lo largo del curso, se pedirá al alumno implementar un juego llamado “Ascii invaders”. Este programa replicará un juego de tipo “matamarcianos” usando la terminal para generar el escenario, enemigos, etc... Además, se le proporcionarán algunas funciones ya implementadas para facilitar el desarrollo del programa.

En este primer ejercicio se pide implementar las bases del juego:

- **Estructuras de dato**s que permitan modelar los objetos que participarán en el programa:
    - Enemigos
    - Personaje principal
    - Misiles
- **Tablero** de juego dividido en NxM casillas, donde por cada casilla podrá haber un objeto de los comentados anteriormente. Los objetos pueden estar “activos” o “inactivos” (solo interactuan los activos)
- **Sistema para dibujar el tablero**
- **Programa principal (main)** que creará un tablero de juego con posiciones de enemigos y personaje principal random y lo mostrará por pantalla.
- **NO HAY QUE IMPLEMENTAR LA LÓGICA DEL JUEGO,** en futuras prácticas se darán las instrucciones para implementar nuevas funciones para ello.

Un juego de tipo “Matamarcianos” tiene la siguiente estructura:

- La pantalla se divide en NxM casillas, donde puede haber objetos. En cada casilla sólo puede haber un objeto, si hubiera más de dos objetos se considera que están “colisionando”, y se deberá resolver esa colisión según marquen las reglas del juego:
    - Colisionan dos enemigos: No pasa nada
    - Colisiona un Enemigo con un Misil: El enemigo pierde vida, el misil desaparece
    - Colisiona un Enemigo con el Personaje Principal: Ambos pierden vida
    - Colisiona el Personaje Principal con un Misil: El personaje principal pierde vida
    - Colisiona un Misil con un Misil: Ambos desaparecen
- Un ejemplo de tablero de juegode tamaño “6Filas-4Columnas” se puede representar de la siguiente manera no es necesario dibujar las casillas). Los enemigos son “V”, los misiles son “I” y el personaje principal es la “A”:

	V		
		V	
	I		
		I	
		A	

- Los enemigos se pueden mover en cualquier dirección (Lo marca su “IA”)
- Los misiles sólo se mueven a lo largo del eje “Y”.

    - Misiles lanzados por los enemigos se mueven en dirección descendente.
    - Misiles lanzados por el personaje principal se mueve en dirección ascendente
- El personaje principal se puede mover en horizontal únicamente. Sus movimientos los marca el jugador, usando el teclado.

Estructuras de datos y tipos enumerados a implementar

Se deben implementar estructuras que permitan modelar el comportamiento de los objetos. Estos pueden tener las siguientes características:

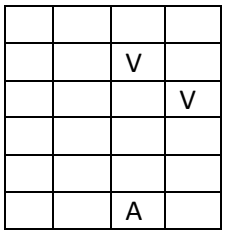
- **Estructura** "**movimiento**":
    - El tablero está formado por un conjunto de casillas en las que puede haber objetos. Los objetos tienen una posición "X, Y" dentro de ese tablero y se desplazarán hacia otra casilla cada cierto tiempo.
    - Un "movimiento" representa la cantidad de casillas que se moverá en cada eje (X,Y) cada vez que se actualice el tablero de juego.
    - Cada enemigo tiene una lista de movimientos "cíclica". Cuando le toque moverse, consultará esa lista para saber qué debe hacer.
    - **ESTO NO HAY QUE IMPLEMENTARLO AÚN,** sólo es una explicación de uso. Los enemigos deben moverse una vez por segundo. Para ello, consultarán su lista de movimientos, y actuarán según lo que indique. Ej:
        - Se tiene el siguiente estado del tablero con dos enemigos

	V		
		V	
		A	

- La lista de movimientos indica que deben moverse "una unidad a la izquierda (X+=1)" en su siguiente turno. El resultado sería el siguiente:

		V	
			V
		A	

- En el siguiente turno, la lista de movimientos indica que deben moverse “una unidad hacia abajo (Y+=1)”



- Así sucesivamente hasta acabar el array de movimientos. En ese momento volverá al inicio del array para consultar los siguientes movimientos de la misma manera

RESUMIENDO:

- La estructura “movimiento” contiene dos números, que representan el número de casillas que se desplaza el objeto en cada turno.

- **Estructura Enemigo_t**: Deben almacenar la vida del enemigo (número entre 0 y 99), puntuación que se consigue al destruirlo (número entre 0 y 1000), un array de 4 movimientos, una variable que indique el tamaño del array de movimientos (más adelante este array será de un tamaño variable), y una variable que indique el movimiento en el que se encuentra (posición dentro del array de movimientos, entre 0 y 3)
- **Estructura Misil_t**: Deben almacenar la cantidad de daño que producen (número entre 0 y 50), y la dirección en la que se mueven (ascendente o descendente, se deja libertad para elegir el tipo apropiado de esta propiedad).
- **Estructura Personaje_t**: Debe almacenar la vida del personaje (número entre 0 y 200), y la puntuación obtenida en este momento.

Para tener los distintos objetos deberá crear la **estructura “objeto_t”** que tendrá las siguientes características:

- **Posición** X, Y del objeto
- **Está activo:** Variable para indicar si el objeto está activo o no. Los objetos no activos son aquellos que han sido destruidos y se deberán retirar del tablero.
- **Tipo** del objeto: Puede ser Enemigo, Misil o Personaje Principal. Se aconseja crear un **tipo enumerado** para representarlo.
- **Sprite**: Un carácter que dependerá del tipo del objeto:
    - Enemigo: ‘V’
    - Misil: ‘.’
    - Personaje Principal: ‘A’

- **Estructuras** de datos Enemigo, Misil y Personaje Principal: En función del tipo de enemigo, una de estas tres estructuras anidadas estará inicializada. (Se pueden usar **uniones** para ahorrar espacio en memoria).

Funciones a implementar

Se os entrega una plantilla de código a rellenar con la implementación pedida. **ESTE CÓDIGO NO COMPILA** , deberéis arreglarlo y completar las partes que faltan. Cada archivo contiene una pequeña descripción del código que hay que completar. A continuación se hace un resumen:

Archivo “tipos.h”:

- Implementar las estructuras indicadas en la sección anterior, con los nombres de estructura dados.

Resto de archivos “.h”:

- No es necesario editarlos, en cada uno de ellos hay una cabecera de función a implementar que no se debe modificar.
- Si lo consideráis necesario, está permitido añadir cualquier función nueva no especificada que os ayude.

Archivos “.c”:

- Cada archivo “.c” tiene un comentario con la función que se debe implementar.
- Se deben crear funciones que inicialicen los tipos de estructuras descritos anteriormente. Estas funciones inicializarán las estructuras a datos “por defecto” (de momento, todo a “0”), y devolverán una variable del tipo indicado:
  - enemigo_t CrearEnemigo()
  - misil_t CrearMisil ()
  - personaje_t CrearPersonajePrincipal()

- Dado que esas estructuras estarán anidadas en una estructura más grande llamada “objeto_t”, es necesario un método para inicializar objetos según su tipo. Además, estos objetos creados con este método deberán estar activos:
  - objeto_t crearObjeto(tipoObjeto tipo)

Archivo “tablero.c”:

- También es necesario crear un tablero que contenga objetos. Se usará un tablero de 20 filas y 5 columnas. Las posiciones iniciales de los objetos serán aleatorias, siguiendo el siguiente criterio:
  - Entre 1 y 3 Enemigos: Deben estar en la mitad superior del tablero, en posiciones aleatorias
  - Entre 2 y 5 Misiles: Pueden estar en cualquier posición random del tablero

- Un personaje principal: Debe estar en la fila más inferior del tablero, en una columna aleatoria.

- Se debe crear una función que inicialice un tablero de NFILAS x NCOLUMNAS suministrado por parámetros. **Se inicializará "vacío"** (todos los objetos inactivos), y luego **se rellenará** siguiendo los criterios anteriormente descritos:
  - void iniciaTablero(Objeto_t tablero[NFILAS][NCOLUMNAS], int numFilas, int numColumnas)

- Por último, se debe implementar una función que dibuje el tablero en pantalla, en función de las coordenadas de los objetos y del sprite que usen. No se debe dibujar la rejilla del tablero, únicamente los objetos que contiene en las posiciones "X" "Y" que indiquen.
  - void dibujaTablero(Objeto_t tablero[NFILAS][NCOLUMNAS], int numFilas, int numColumnas)

	V		
		V	
	I		
		I	
		A	

Programa principal "main" (archivo asciiMain.c):

Se debe realizar un programa "main" que realice las siguientes operaciones:

- Reservar memoria para el array bidimensional "tablero", de 20 filas por 10 columnas.
- Inicializar el tablero con objetos en posiciones aleatorias, siguiendo el criterio comentado anteriormente.
- Mostrar el tablero por pantalla (sólo una vez).
- Finalizar.

Evaluación y entrega:

Se deben implementar todas las partes pedidas anteriormente. Se valorará lo siguiente:

- División del programa en varios ficheros temáticos:
  - Por cada tipo de objeto (enemigo, misil y personaje), un archivo ".h" y ".c" que contenga las funciones de inicialización pedidas.
  - Un archivo ".h" y ".c" para las funciones de inicialización y dibujado del tablero.

    - Un archivo “asciiMain.c” que contenga la función “main” del programa
- Definiciones adecuadas de estructuras y enumerados: Deben ajustarse a lo pedido en el enunciado
- Funcionamiento correcto: Generación de posiciones aleatorias para los objetos y dibujado del tablero sin errores

TODOS los archivos generados deben contener el nombre del alumno y grupo al que pertenece.

El ejercicio es individual. **Está prohibido compartir, incluir o copiar código no desarrollado por el alumno**. En caso de detección de copias, el alumno se expone a suspender la asignatura, y dependiendo de la gravedad de la copia, podría considerarse la apertura de expediente y expulsión del curso.

El alumno deberá subir un archivo comprimido en formato **“zip” (no se admite otro formato)** a la actividad abierta en blackboard para realizar la entrega. El archivo deberá nombrarse de la siguiente manera:

NombreAlumno_Apellido_IPIIEjEv1.zip