# Ejercicio evaluable 3: Parseo de cadenas

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 tercer ejercicio se pide implementar funciones para inicializar objetos con parámetros dados por el usuario al inicio del programa, usando argc y argv. El programa parseará una línea de datos con un formato dado y colocará los objetos en el tablero en función de los datos suministrados al inicio.

## Parte 1: Implementar funciones de inicialización de objetos con datos suministrados

Se pide implementar las siguientes funciones de inicialización de objetos usando parámetros dados:

**Archivo “enemigo.h”**

```
enemigo_t CrearEnemigoConDatos(int vida, int puntuacion,movimiento_t* movimientos);
```

**Archivo “enemigo.c”**

```
enemigo_t CrearEnemigoConDatos(int vida, int puntuacion,movimiento_t* movimientos){
//inicializar una estructura de tipo enemigo_t con los datos de vida, puntuacion
        //y movimientos suministrados
        //el array “movimientos” debe tener 4 elementos como mínimo
}
```

**Archivo “misil.h”**

```
misil_t CrearMisilConDatos(int danio,direccion_e dir);
```

**Archivo “misil.c”**

```
misil_t CrearMisilConDatos(int danio,direccion_e dir){
//inicializar una estructura de tipo misil _t con los datos de danio y puntuacion suministrados
}
```

**Archivo “personaje.h”**

```
personaje_t CrearPersonajePrincipalConDatos(int vida, int puntuacion);
```

**Archivo “personaje.c”**

```
personaje_t CrearPersonajePrincipalConDatos(int vida, int puntuacion){
//inicializar una estructura de tipo personaje_t con los datos de vida y puntuacion
}
```

## Parte 2: Implementar funciones de parseo de tipos enumerados de datos

Los datos “tipo de objeto” y “dirección de misil” son tipos de datos enumerados que se suministrarán como cadenas al programa. Se deben crear dos funciones que transformen una cadena con los valores de esos tipos al tipo enumerado correcto (Ej. Si el usuario escribe “tipo=misil”, se debe transformar la cadena “misil” al tipoObjeto_e “misil”). Para ello, añadir las siguientes funciones:

**Archivo “objetos.h”**

```
tipoObjeto_e dameTipo(char* strTipo);
direccion_e dameDireccion(char* strDir);
```

**Archivo “objetos.c”**

```
direccion_e dameDireccion(char* strDir)
{
//comparar la cadena “strDir”con las palabras "ascendente","descendente"
//devolver el tipo”direccion_e”  asociado
//si no es ninguno, devolver "2" (error)
}
tipoObjeto_e dameTipo(char* strTipo)
{
//comparar la cadena “strTipo”con las palabras "misil","enemigo","personaje"
//devolver el tipo asociado
//si no es ninguno, devolver "4" (error)
}
```

## Parte 3: Implementar el parseo de la cadena de entrada con los datos de un objeto

Al inicio del programa, se le suministrará por argc/argv una lista de palabras con una codificación que permitirá definir la posición, tipo y datos de cualquiera de los objetos que se pueden crear en nuestro programa (misil, enemigo ó personaje principal). Para ello se seguirá el siguiente formato (NO CONTIENE ESPACIOS, TODO EN UNA SOLA LÍNEA, TODAS LAS LETRAS EN MINUSCULA):

- **Objetos de tipo "personaje principal":**

objeto[posX=número,posY=número,sprite=carácter,tipo=personaje,vida=número,puntuacion=número]

- **Objetos de tipo "misil":**

objeto[posX=número,posY=número,sprite=carácter,tipo=misil,direccion=direccion_e,danio=número]

- **Objetos de tipo enemigo**

objeto[posX=número,posY=número,sprite=carácter,tipo=enemigo,vida=número,puntuacion=número,movimientos=x=número,y=número,x=número,y=número,x=número,y=número,x=número,y=número]

El formato mostrado anteriormente debe empezar por la palabra "objeto", y contener entre corchetes "[" y "]" los parámetros de inicio de los objetos dados. Por cada posición, vida, puntuación, etc... se tiene un valor numérico asociado. Los "tipos" de objeto válidos son las cadenas "enemigo","misil" y "personaje". Las direcciones de los misiles válidos son las cadenas "ascendente" y "descendente.

Los enemigos tienen una lista de 4 movimientos en X/Y asociados, se agrupan en pares de datos numéricos "x" e "y". Un ejemplo de estas cadenas que crearía un objeto "personaje", otro "misil" y "enemigo" serían las siguientes:

objeto[posX=6,posY=9,sprite=A,tipo=personaje,vida=5,puntuacion=0]
objeto[posX=6,posY=3,sprite=.,tipo=misil,direccion=ascendente,danio=10]
objeto[posX=5,posY=7,sprite=V,tipo=enemigo,vida=100,puntuacion=10,movimientos=x=0,y=1,x=1,y=0,x=-1,y=0,x=0,y=-1]

De esta manera, la descripción de cada uno de los objetos includos en el juego va a estar recogida en distintos parámetros del main.

Se debe implementar una función que, dada una de las cadenas anteriores, averigüe el tipo del objeto e inicialice sus campos a partir de los datos de la cadena.

**Archivo “objetos.h”**

```
//Esta función recibe por parámetros una cadena con el formato de objeto visto anteriormente
//Se debe parsear la cadena y devolver un objeto con los datos inicializados según indique la cadena
        objeto_t CrearObjetoConDatos(char* cadena);
```

**Archivo “objetos.c”**

```
objeto_t CrearObjetoConDatos(char* cadena){
//Declaración de variables para cada tipo de dato
//Reserva de copia de cadena de entrada
//copiar cadena, para poder usar strtok
//la cadena debe empezar por la palabra "objeto" y acabar por "]". Si no, es un error
 //consumir final de objeto "]" (sustituir por '\0')
//si el primer token es la palabra “objeto”
                    //iterar iniciando cada una de las variables posibles. Se acaba cuando "token" o "valor"
                    // son null
                    //leer el token (posX, posY, vida,puntuación,
                    //                  tipo,direccion,danio,sprite,movimientos)
                   //si token no es NULL
                            //si no es un array movimientos (son especiales)
                                //leer valor asignado
                               //inicializar la variable adecuada en función del token leído (vida, posX, etc..)
//si es un array movimientos
                                //leer los cuatro movimientos y almacenarlos
      //una vez leidas todas las variables de la cadena
      ///Crear un objeto, iniciar posición, tipo, sprite, de objeto
      //en función del tipo del objeto, crear un objeto con sus datos de personaje,misil ó enemigo
      // usar crear[Personaje/Misil/Enemigo] con datos
```

//liberar datos

//Devolver el objeto

## Parte 4: Iniciar tablero con datos suministrados por argc/argv

Se debe crear una función para inicializar los objetos del tablero usando una lista de cadenas suministradas por argc/argv. Para ello se debe crear la siguiente función:

**Archivo "tablero.h"**

```
//esta función recibe un tablero ya iniciado, número de filas/columnas del mismo, y una lista de
//cadenas con el formato "objeto" visto anteriormente. El Número de cadenas está siministrado en la
//variable "numObjetos
void iniciaTableroConDatos(objeto_t** tablero, int numFilas, int numColumnas, int numObjetos, char** datos);
```

**Archivo "tablero.c"**

```
void iniciaTableroConDatos(objeto_t** tablero, int numFilas, int numColumnas, int numObjetos, char** datos){
//inicializar el tablero con objetos "no activos"
//por cada número de cadenas del array datos
    //iniciar un objeto con datos (llamar a la función "crearObjetoConDatos" pasándole una cadena
    //del array "datos")
    //colocarlo en la posición del tablero indicada por su "posX/Y"
      //antes de colocarlo, comprobar si las posiciones están dentro del tablero
      //si no están, se avisa al usuario
}
```

## Parte 5: Bucle principal del programa "main"

Modificar el programa principal "main" para que realice las siguientes operaciones:

```
Int main(int argc, char** argv){
        //pedir numFilas y numColumnas
```

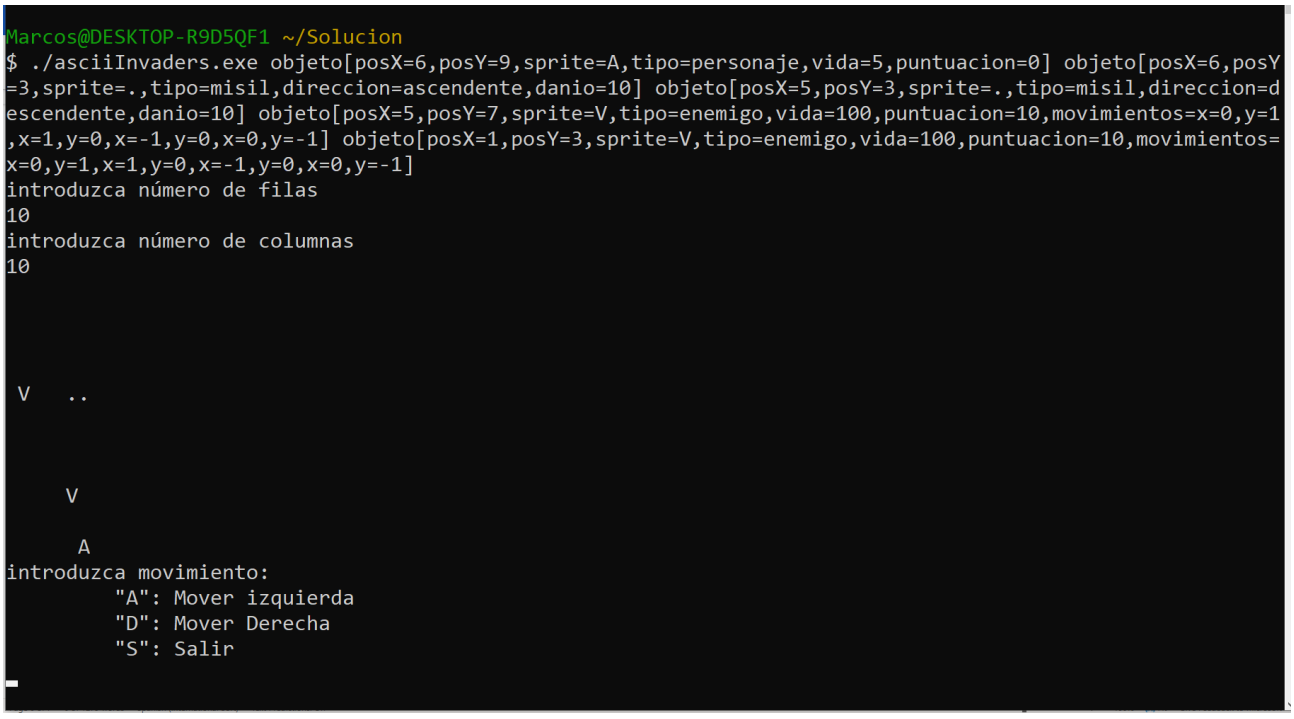
```
    //reservar tablero de tamaño numFilas numColumnas
    //iniciar tablero con datos suministrados por argc/argv
    //mientras haya un personaje activo
        //mostrar el tablero
        //actualizar tablero
    //repetir
    //liberar memoria del tablero
}
```

## Ejemplo de ejecución:

Suponiendo que se desea crear un escenario de 10 filas por 10 columnas, con un personaje principal, dos balas y dos enemigos con datos dados, este sería el comando con parámetros:

```
./asciiInvaders.exe objeto[posX=6,posY=9,sprite=A,tipo=personaje,vida=5,puntuacion=0]
objeto[posX=6,posY=3,sprite=.,tipo=misil,direccion=ascendente,danio=10]
objeto[posX=5,posY=3,sprite=.,tipo=misil,direccion=descendente,danio=10]
objeto[posX=5,posY=7,sprite=V,tipo=enemigo,vida=100,puntuacion=10,movimientos=x=0,y=1,x=1,y=0,x=-
1,y=0,x=0,y=-1]
objeto[posX=1,posY=3,sprite=V,tipo=enemigo,vida=100,puntuacion=10,movimientos=x=0,y=1,x=1,y=0,x=-
1,y=0,x=0,y=-1]
```

Se suministra el ejecutable “SolucionAscii.exe” para comparar los resultados y practicar con los parámetros de entrada. Para el ejemplo anterior, la salida sería la siguiente:



## 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
- Uso correcto de memoria dinámica
- Comprobaciones de accesos a arrays correctos.
- Parseo de cadenas argc/argv 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_IPIIEjEv3.zip