Ejercicio evaluable Extraordinaria: Uso de ficheros y generación de histórico de teclas pulsadas

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 último ejercicio se pide implementar funciones para inicializar objetos con parámetros leídos desde un fichero de texto suministrado por el usuario al inicio del programa, usando argc y argv. Este fichero contendrá varias líneas de texto, usando el formato visto en el ejercicio evaluable 3. Cada línea de texto representa un objeto que habrá que colocar en el tablero.

El programa leerá todas las líneas del fichero de datos, parseará todos los objetos y los colocará en el tablero en función de los datos suministrados al inicio.

Adicionalmente, el programa guardará todas las teclas pulsadas a lo largo del juego en un archivo “histórico”. Para ello, se modificarán algunas funciones y se crearán algunas estructuras nuevas.

Parte 1: Implementar lectura de datos del juego desde fichero

Se pide implementar las siguientes funciones de lectura de datos de fichero:

Archivo “asciiMain.c”

```
//Función que, dado un archivo “f” abierto previamente, lee una línea del fichero
//retorna una cadena con la línea leída de fichero
char* leeLineaDinamicaFichero(FILE* f){

}
```

//Función que, dado el nombre de un fichero pasado en el parámetro “fichero”, abre ese fichero, lee todas las líneas, y las retorna como un array de cadenas

//la variable “numLineas” se usa para indicar el número de líneas leídas. Se pasa por referencia para poder modificarla dentro de la función

char leeLineas(const char* fichero, int* numLineas)**

{

//declaración de variables para el fichero, número de líneas y el array de líneas

//abrir fichero

//si no hay errores de fichero

//leer líneas de fichero hasta que sea final de fichero

//almacenar la nueva línea en el array de líneas, actualizar su tamaño

//ojo, solo se guardan líneas que tengan longitud mayor a "0"

//si es fin de fichero

//cerrar fichero

//actualizar variable numLineas

//retornar lineas

//si hay errores de fichero

//retornar NULL;

}

Parte 2: Usar los datos leídos de fichero

Anteriormente, se pidió crear funciones para parsear el array “argv” con datos de los objetos del programa en un formato dado. Ese formato se mantiene, pero ahora los objetos deben ser leídos de un archivo de texto.

Se pide modificar la función “main” del programa para que lea los datos de los objetos siguiendo las siguientes reglas:

- El nombre del fichero de datos se suministrará al programa usando el primer parámetro de argv.
- Usando las funciones pedidas en la sección anterior, el programa debe leer todas las líneas del fichero suministrado, guardándolas en un array de cadenas. Cada línea se pasará a una cadena dentro de ese array.

- Una vez leídas todas las líneas, invocará al método “iniciaTableroConDatos” usando el array de datos leído de fichero.
- El fichero contendrá un objeto por cada línea, usando el formato visto en la práctica anterior. Se puede usar el siguiente archivo “datos.txt” como ejemplo:

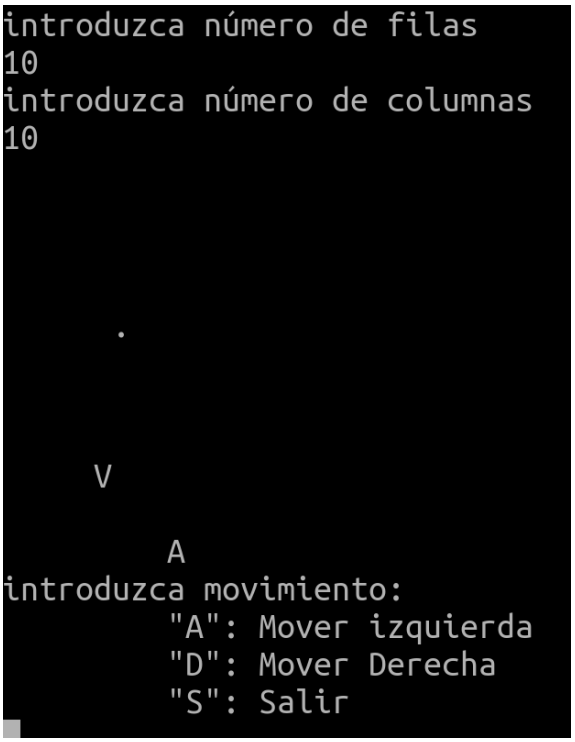
```
objeto[posX=9,posY=9,sprite=A,tipo=personaje,vida=5,puntuacion=0]
objeto[posX=6,posY=3,sprite=.,tipo=misil,direccion=ascendente,danio=10]
objeto[posX=5,posY=7,sprite=V,tipo=enemigo,vida=100,puntuacion=10,movimientos=x=0,y=1,x=1,y=0,x=-1,y=0,x=0,y=-1]
```

Parte 3: Test de lectura de parámetros desde fichero

Se debe probar el programa para asegurar que hasta el momento funciona sin problemas. Para ello, una vez compilado, se debe ejecutar de la siguiente manera:

./asciiInvaders.exe datos.txt

El archivo “datos.txt” debe estar en el mismo directorio. Si todo está correcto, el programa pedirá el número de filas y número de columnas del tablero, y empezará la ejecución del programa con 3 objetos.



Parte 4: Crear histórico de teclas pulsadas

Se pide crear una estructura "historico_t" que almacene un array dinámico de teclas pulsadas junto con su tamaño. Este array debe poder redimensionarse cada vez que se pulse una nueva tecla, y se debe guardar el número de teclas pulsadas en la estructura.

Este histórico se reservará en la función "main", y se actualizará en la función "muevePersonaje", ya que es la única que tiene acceso a la terminal para saber qué tecla se ha pulsado durante el juego. Para ello, se deben realizar los siguientes cambios:

Archivo "tipos.h"

//añadir estructura "historico_t", que contiene un array dinamico de caracteres junto con su tamaño

Archivo "personaje.h"

Modificar (no hay que crear una función nueva, sólo modificar la cabecera) la cabecera de la función "muevePersonaje" para que pueda recibir por referencia una estructura de tipo histórico:

void muevePersonaje(historico_t* historico, objeto_t* objeto, int numFilas, int numColumnas);

Archivo "personaje.c"

Modificar la función "muevePersonaje" para que, antes de acabar, guarde la tecla pulsada dentro del array de teclas de la estructura "histórico":

```
void muevePersonaje (historico_t* historico, objeto_t* objeto, int numFilas, int numColumnas){
//al final:
   //actualizar el historico:
      //redimensionar el array de teclas
      //introducir la tecla presionada en la última posición (las anteriores teclas no se
      // pierden)
      //actualizar numero de teclas;
}
```

Archivo “tablero.h”

Modificar la cabecera de la función “actualizaTablero” para que pueda recibir por referencia

void actualizaTablero(historico_t* historico, objeto_t** tablero, int numFilas, int numColumnas);

Archivo “tablero.c”

Modificar la función “actualizaTablero” para que le pase el histórico a la función “ muevePersonaje” cuando haya que moverlo

void actualizaTablero(historico_t* historico, objeto_t** tablero, int numFilas, int numColumnas) {

…..

}

Parte 5: Bucle principal del programa “main”

Modificar el programa principal “main” para que cree una estructura de tipo histórico, se la pase al método “actualizaTablero”. Al final del programa, el histórico se debe guardar en el archivo suministrado por el usuario en el segundo parámetro de argv:

```
Int main(int argc, char** argv){
        //Testear datos correctos en argc/argv
        //pedir numFilas y numColumnas
        //reservar tablero de tamaño numFilas numColumnas
        //leer datos del fichero suministrado en argc/argv
        //iniciar tablero con datos leídos de fichero
        //crear variable de tipo “historico_t”
        //mientras haya un personaje activo
            //mostrar el tablero
            //actualizar tablero usando historico
        //repetir
        //liberar memoria del tablero
        //guardar historico en el archivo suministrado por argc/argv
        //liberar historico
}
```

Ejemplo de ejecución:

./asciiInvaders.exe datos.txt historico.txt

En el ejemplo anterior, el programa leerá los objetos del archivo “datos.txt” y guardará todas las teclas pulsadas durante el juego (teclas “A”, “D” y “S”) en una sola línea en el archivo “historico.txt”.

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 sin errores
- Lectura/escritora de datos usando ficheros
- Actualización de array dinámico guardado en “histórico”

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_IPIIEjEvExtra.zip