INF-253 Lenguajes de Programación
Tarea 2: C

Profesor: José Luis Martí Lara, Jorge Ariel Díaz Matte,
Wladimir Elías Ormazabal Orellana.
Ayudante Cátedra: Norman Cortés Vega, Lucas Morrison Osorio,
Jhossep Martínez Velásquez.
Ayudante Coordinador: Álvaro Rojas Valenzuela.
Ayudante Tareas: Fernanda Araya Zárate, Bryan González Ramírez,
Andrés Jablonca Peña, Blas Olivares Gutiérrez, Nicolás Paz Tralma,
Bastián Salomon Ávalos, Cristian Tapia Llantén, Cristóbal Tirado Morales.

26 de agosto de 2024

x es un puntero a una función, que retorna un puntero a una función, que retorna un puntero a una función, que retorna un puntero a una función...

1. La Defensa de Po Inter City

Advertencia, una gran flota de barcos de los SegFault se acerca rápidamente a Po Inter City. El General Vo Ider prepara las defensas en la gran costa de la ciudad, cinco grandes cañones que pueden disparar toda clase de misiles. Pero hay un problema, el antiguo controlador de los cañones fue comprometido y solo arroja Segmentation fault! Rápidamente, el General se acerca a usted, estudiante de lo Lenguaje de Programación de la UTFSM que programe un nuevo controlador para los cañones. Para ello, se tiene que usar el lenguaje de programación C.

2. City Defender 2.0

El controlador se compone de dos partes principales, el tablero y las cartas que controlan los cañones. El tablero sabe en todo momento a cuantos turnos les falta para que llegue la gran flota y el área en donde sabe que hay una posibilidad de acertar un misil. Mientras que las cartas permiten elegir uno de los cañones a disparar y las coordenadas relativo al área del tablero. A continuación se muestran los requerimientos del programa:

2.1. El Tablero

El tablero divide el área en una grilla cuadrada de celdas de 1x1 metro cuadrado. Cada celda puede saber si se ha disparado en ese sector o no, como también puede saber si acertó o no a una parte de un barco.
Para ello se le da el siguiente fragmento de código:

```
extern void *** tablero;

inicializarTablero(int tamano);
mostrarTablero();
```

Donde:

- `void *** tablero`: Almacena la información de cada celda en una matriz cuadrada.
- `inicializarTablero`: Inicializa el tablero cuadrado para un tamaño dado.
- `mostrarTablero`: Imprime por pantalla estándar el estado actual del tablero, cada celda tiene tres estados posibles. Si no se le ha disparado, si el disparo ha fallado o si ha acertado.

Como el General solo le interesa la información, él te ha permitido que crees un diseño a gusto. Unos técnicos te mostraron un ejemplo de como podría ser el tablero:
Ejemplo:

```
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |O|  |  |  |  |  |  |
|  |O|  |  |  |  |  |  |  |  |
|  |X|  |  |O|  |  |  |  |  |
|  |X|  |  |  |  |  |  |  |  |
|  |X|O|  |  |  |O|O|O|  |  |
|  |O|O|  |  |  |X|X|O|  |  |
|  |  |  |  |  |  |O|O|O|  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
```

Las `X` representan un acierto, los `O` representa un fallo y los espacios en blanco donde no se ha disparado.

2.2. Las Cartas

Estas cartas representa los distintos métodos de disparos que tiene disponible los cañones. Solo puede estar disponible cinco cartas a la vez, uno para cada cañón. Cuando el General selecciona una de las cartas, el sistema pide ingresar una coordenada del tablero como objetivo del misil y la carta es usada. Luego el sistema elige aleatoriamente otro tipo de carta (o la misma) dependiendo del estado actual del cañón producto del disparo

realizado.
Para la implementación de este sistema, cada carta puede ser representado por un puntero a una función que recibe dos parámetros (las coordenadas del tablero), y que retorne otro puntero a una función del mismo tipo de función. Las cartas son ilimitadas (a excepción de una). A continuación te entregan una lista de cada tipo de carta disponible, lo que debe hacer y cuáles son los posibles retornos:

```
void * disparoSimple(int x, int y);
void * disparoGrande(int x, int y);
void * disparoLineal(int x, int y);
void * disparoRadar(int x, int y);
void * disparo500KG(int x, int y);
```

- `disparoSimple` :
  - Disparo: Dispara un misil con área de efecto de 1x1 celdas en las coordenadas `(x,y)`.
  - Retorno: Puede retornar alguna de las siguientes cartas: `disparoSimple` 65 %, `disparoGrande` 20 %, `disparoLineal` 5 %, `disparoRadar` 10 %.
- `disparoGrande` :
  - Disparo: Dispara un gran misil con área de efecto de 3x3 celdas con centro en las coordenadas `(x,y)`.
  - Retorno: Puede retornar alguna de las siguientes cartas: `disparoSimple` 80 %, `disparoGrande` 3 %, `disparoLineal` 10 %, `disparoRadar` 5 %, `disparo500KG` 2 %.
- `disparoLineal` :
  - Disparo: Dispara múltiples misiles pequeños afectando un área de 1x5 o 5x1 celdas a elección con centro en las coordenadas `(x,y)`.
  - Retorno: Puede retornar alguna de las siguientes cartas: `disparoSimple` 85 %, `disparoGrande` 5 %, `disparoLineal` 2 %, `disparoRadar` 6 %, `disparo500KG` 2 %.
- `disparoRadar` :
  - Disparo: Dispara un misil radar que revela si hay (o no) presencia de barcos enemigos en un área de 5x5 con centro en las coordenadas `(x,y)`.
  - Retorno: Puede retornar alguna de las siguientes cartas: `disparoSimple` 75 %, `disparoGrande` 15 %, `disparoLineal` 5 %, `disparoRadar` 2 %, `disparo500KG` 3 %,

- `disparo500KG` :
  - Disparo: Este misil es el más grande de todos. Puede afectar a un área de 11x11 celdas con centro en las coordenadas `(x,y)`. **Solo se puede obtener una vez**.
  - Retorno: Debido al gran tamaño del misil, deja incapacitado el cañón que lo disparó. Por lo que no retorna una carta, dejando solo disponible los otros cañones.

Para una implementación de las cartas le entregan el siguiente fragmento de código:

```
typedef struct Mano{
    void ** carta;
    int disponibles;
} Mano;

extern Mano Cartas;
void inicializarMano();
void mostrarMano();
void usarCarta();
```

Donde:

- `Mano Cartas` : Variable que contiene una estructura `Mano`, que contiene un array con las cartas actuales.
- `inicializarMano()` : Inicializa la mano inicial correspondiente a cinco cartas de **disparo simple**.
- `mostrarMano()` : Muestra por pantalla una lista de las cartas en mano disponible.
- `usarCarta()` : Usa una carta de la mano disponible a elección.

El General espera que la interfaz sea lo suficientemente explícita para saber qué inputs ingresar por el terminal.

3. Las Pruebas

Para que el General apruebe tu controlador, necesita que sea capaz de simular tres escenarios diferentes al momento de iniciar el programa, estos son:

- **Fácil**: Un tablero de 11×11, con: 2 barcos de tamaño 1×2, 1 de tamaño 1×3, 1 de tamaño 1×4 y 1 de tamaño 1×5. Y un límite de 30 turnos.
- **Medio**: Un tablero de 17×17, con: 3 barcos de tamaño 1×2, 2 de tamaño 1×3, 1 de tamaño 1×4 y 1 de tamaño 1×5. Y un límite de 25 turnos.
- **Difícil**: Un tablero de 21×21, con: 3 barcos de tamaño 1×2, 2 de tamaño 1×3, 2 de tamaño 1×4 y 2 de tamaño 1×5. Y un límite de 15 turnos.

Un turno consiste en usar solo una carta. Y al momento de inicializar el tablero, se posicionan todos los barcos correspondientes a la dificultad de manera aleatoria. Los barcos pueden estar verticales u horizontalmente.

Si se logra hundir todos los barcos antes del límite de turnos, significará que será posible realizar una defensa exitosa. De lo contrario, la defensa fracasará. En ambos caso se debe **revelar el tablero final con los barcos que sobrevivieron** y terminar el programa.

Unos técnicos te mostraron un ejemplo de como podría ser una ejecución completa: Ejemplo:

```
Selecciona la Dificultad:
1. Facil -> 11 X 11, 5 Barcos
2. Medio -> 17 X 17, 7 Barcos
3. Dificil -> 21 X 21, 9 Barcos
Ingrese un numero: 1
Turno 1
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
Cartas:
Simple | Simple | Simple | Simple | Simple
Selecciona una Carta: 3
Selecciona Coordenadas
X : 3
Y : 3
MISS!
Turno 2
| | | | | | | | | | |
| | | | | | | | | | |
| | |O| | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
Cartas:
Simple | Simple | Grande | Simple | Simple
Selecciona una Carta: 3
Selecciona Coordenadas
```

```
X : 4
Y : 4
HIT!
HIT!
Turno 3
| | | | | | | | | | |
| | | | | | | | | | |
| | |O|O|O| | | | | |
| | |X|O|O| | | | | |
| | |X|O|O| | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
Cartas:
Simple | Simple | 500KG | Simple | Simple
Selecciona una Carta: 3
Gran Misil Preparado!!! Se perdera el canon.
Selecciona Coordenadas:
X : 6
Y : 6
ULTRA HIT!
TODOS LOS BARCOS DESTRUIDOS, DEFENZA EXITOSA!
Tablero Final:
|O|O|O|O|O|O|X|O|O|O|O|
|O|O|O|O|O|O|X|O|O|O|O|
|O|O|O|O|O|O|X|O|O|O|O|
|O|O|X|O|O|O|X|O|O|O|O|
|O|O|X|O|O|O|X|O|O|O|O|
|O|O|X|O|O|O|O|O|O|O|O|
|O|O|O|O|O|O|X|X|X|X|O|
|O|O|O|O|O|O|O|O|O|O|O|
|O|O|O|O|X|O|O|O|O|O|O|
|O|O|O|O|X|O|O|O|O|O|O|
|O|O|O|O|O|O|O|O|O|X|X|
```

4. Sobre la Entrega

- El código debe venir indentado y ordenado. De no existir orden, se realizará descuento.
- Se debe entregar los siguientes archivos:
  - CityDefender.c, Cartas.c, Cartas.h, Tablero.c, Tablero.h
  - Makefile
  - README.txt

Donde `CityDefender.c` debe contener la función **main**, los archivos `.h` debe contener las **declaraciones** de las estructuras, funciones, variables y definiciones. Y los archivos `.c` debe contener todas las **implementaciones** de las funciones de su header correspondiente.

- Se puede crear nuevas funciones, estructuras y/o definiciones que estime conveniente. Como también agregar nuevos atributos a las estructuras existentes.
- Es de uso obligatorio el `void *** tablero` para la entrega mínima, de lo contrario significará un 0 en la nota final de la tarea.
- No se permite cambiar las firmas, parámetros y retorno, como también agregar nuevos parámetros en las funciones mencionadas en el enunciado.
- Las funciones deben ir comentadas, explicando clara y brevemente lo que realiza, los parámetros que recibe y los que devuelve (en caso de que devuelva algo). Se deja en libertad al formato del comentario.
- Debe estar presente el archivo **MAKEFILE** para que se efectúe la revisión, este debe compilar TODOS los archivos mediante compilación separada.
- Se utilizará Valgrind para detectar las fugas de memoria.
- El trabajo es individual obligatoriamente.
- **La entrega debe realizarse en un archivo comprimido en tar.gz y debe llevar el nombre: Tarea2LP_RolAlumno.tar.gz**.
  Ej.: `Tarea2LP_202273000-k.tar.gz`.
- El archivo **README.txt** debe contener **nombre y rol del alumno**, junto a instrucciones detalladas para la correcta **compilación y ejecución** del programa.
- La entrega es vía aula y el plazo máximo de entrega es hasta el **11 de septiembre a las 23:55 hora aula**.
- Las consultas se deben realizar mediante el foro de la tarea disponible en AULA.
- Por cada día (o fracción) de atraso se descontará 20 puntos. 10 puntos dentro de la primera hora.
- Las copias serán evaluadas con nota 0 y se informará a las respectivas autoridades.

5. Clasificación

5.1. Entrega

Para la clasificación de su tarea, se debe realizar una entre con requerimientos mínimos que otorgarán hasta 30 pts base. Luego se le entregará puntaje dependiendo de los otros requerimientos que llegue a cumplir.

5.1.1. Entrega Mínima

Para obtener el puntaje mínimo en la entrega, el programa de cumplir con los siguientes requerimientos:

- Inicializa el tablero de dificultad fácil, asignando espacio en la memoria **heap** en la variable `void ***tablero` con acceso global. (10 pts)
- El tablero se debe inicializarse y borrarse a través de *malloc* y *free*, no puede haber *leaks* de memoria. (10 pts)
- Genera todos los barcos con posición y orientación aleatoria en el tablero. Y el programa termina mostrando el tablero revelando la posición de los barcos. (10 pts)

5.1.2. Entrega

Luego de cumplir con la Entrega Mínima, puede obtener más puntaje cumpliendo con los siguientes puntos (puede haber puntaje parcial por cada punto):

- **Características y Funcionalidad**:
  - Utiliza los Header para declarar las funciones, estructuras, variables y definiciones correspondientes. (5 pts)
  - Programa y usa al menos la carta de disparo simple, marcando si el disparo falló o acertó en el tablero. (5 pts)
  - Programa y usa todas las cartas disponibles según lo descrito en el enunciado. (15 pts)
  - Almacena y utiliza los punteros a funciones en la estructura *Mano* para la funcionalidad de seleccionar y usar las cartas. (20 pts).
  - El programa termina cuando se acierten a todos los barcos del tablero o si supera el límite de turnos. (15 pts)
  - El programa permite seleccionar las dificultades *Fácil, Medio* y *Difícil*, creando el tablero, los barcos y el límite de turnos correspondiente. (5 pts)
  - Utiliza una interfaz clara y precisa, sin la necesidad de predecir las entradas o salidas del usuario. (5 pts)

5.2. Descuentos

- Falta de orden (Max -20 pts)
- Falta de comentarios (-10 pts c/u, Max -30 pts)
- Código no compila (-100 pts)
- Compilación Unificada (-5 pts)
- Warnign (-5 pts c/u)
- Falta de Makefile (-100 pts)
- Porcentaje de leak de memoria ((1 - 5) % -3 pts, (6 - 50) % -15 pts, (51 - 100) % -30 pts)
- Falta de README (-20 pts)
- Falta de alguna información obligatoria en el README (-5 pts c/u)
- Día de atraso (-20 pts por día (o fracción), -10 pts dentro de la primera hora)
- Mal nombre en algún archivo entregado o información errónea (-5 pts c/u)

En caso de existir nota negativa, esta será remplazada por un 0.