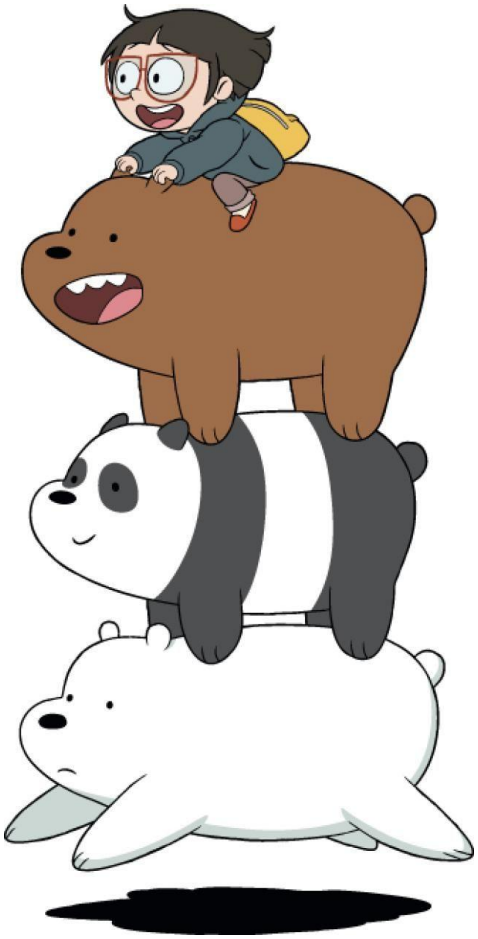
Algoritmos y Programación I
Curso Mendez

.UBAfiuba
FACULTAD DE INGENIERÍA

Osos Contra Reloj

TP2



Fecha de Presentación: 20/05/2021

Fecha de Vencimiento: 24/06/2021

1. Introducción

Los escandalosos son tres osos hermanos (de corazón): Pardo, Panda y Polar, cada uno con sus propias personalidades, intereses y talentos.

Pardo fue rescatado de un árbol por guardaparques, Panda escapó del cautiverio en China y Polar huyó del Ártico ruso por ser perseguido por unos cazadores. Luego de vagar de un lado a otro, Pardo y Panda se topan en algo que pareciera ser una vía de tren. Luego de una breve conversación escuchan que se acercaba un tren, a lo que de repente cae Polar del cielo para rescatarlos. Por una cuestión de rapidez, se ponen uno encima de los otros, y así, es como nace...la torre.

Los osos viajan en una torre cuando necesitan transportarse. Se forma con Pardo en la parte superior, Panda en el medio y Polar en la parte inferior, de acuerdo con sus respectivas edades.

Han estado juntos desde la primera infancia en busca de un nuevo hogar. Luego de vivir en cajas, México y hasta una isla, se asientan en una cálida cueva dentro del bosque de San Francisco.

Chloe Park es una niña humana prodigio de diez años que un día se coló en la cueva de los Osos Escandalosos para investigar y juntar información para su trabajo universitario. Con el tiempo se fueron haciendo muy amigos, y ahora disfrutan mucho pasar el tiempo juntos compartiendo numerosas aventuras.

2. Objetivo

El presente trabajo práctico tiene como objetivo que el alumno:

- Diseñe y desarrolle las funcionalidades de una biblioteca con un contrato preestablecido.
- Se familiarice con y utilice correctamente los tipos de datos estructurados.
- Desarrolle una interfaz gráfica amigable y entendible para el usuario.

Por supuesto, se requiere que el trabajo cumpla con las buenas prácticas de programación profesadas por la cátedra.

Se considerarán críticos la modularización, reutilización de código y la claridad del código.

3. Enunciado

Un día, aburridos de estar encerrados, los Osos y Chloe decidieron salir al bosque y jugar a las escondidas. Sin embargo, se olvidaron del toque de queda que rige debido a los aumentos de casos de Covid en la ciudad y ya está oscureciendo. Además tienen otro problema: Chloe sigue escondida y no sabe que ya tienen que volver a casa.

Ayudá a tu oso a encontrar a su amiga antes de que comience el confinamiento obligatorio.

¡Pero hay un problema más! Ya se hizo de noche en el bosque y el personaje no podrá ver nada de lo que hay a su alrededor, y deberá ayudarse de su linterna, velas y bengalas para poder llegar a encontrarse con su amiga.

3.1. Personajes

Cada personaje, detectado por el trabajo práctico 1, tendrá características particulares que le darán una ventaja en este reto de encontrar a Chloe a tiempo.

- **Polar**: por hacer artes marciales y estar muy entrenado, tiene la habilidad de poder saltar piedras.
- **Pardo**: por ser guardabosque y conocer muy bien los caminos, en caso de toparse con un árbol, podrá enfrentar la situación más rápido y perder menos tiempo.
- **Panda**: es un genio usando el GPS, y esto le da la ventaja de saber la ubicación de Chloe, luego de sumar 30 segundos al tiempo perdido.

El personaje arrancará con el tiempo en 0 segundos, en la columna 0 y en una fila aleatoria. A su vez, Chloe estará en una posición generada aleatoriamente.

3.2. Obstáculos

Los obstáculos son elementos que estarán en cualquier posición del mapa e intentarán hacer que el personaje pierda tiempo.

- **Árboles:** si el personaje choca con un árbol, sumará un segundo al tiempo perdido. Excepto Pardo que sumará medio segundo por cada árbol, al tiempo perdido. En el bosque habrá 350 árboles.
- **Piedras:** hacen tropezar al personaje y sumar dos segundos al tiempo perdido. Excepto a Polar, que puede evadirlas. En el bosque habrá 80 piedras.
- **Koala Nom Nom y sus sekoalaces:** Nom Nom es un personaje codicioso y competitivo a quien no le importa el toque de queda, que aparecerá desde el comienzo del juego. Éste organizó una fiesta clandestina e invitó a varios de sus amigos koalas, quienes irán apareciendo de a uno, aleatoriamente por el terreno, cada vez que el jugador prenda la linterna. Debido a que el oso no está invitado, si el jugador se choca con alguno de los koalas (incluido Nom Nom), éstos se encargarán de regresar al personaje al inicio del bosque en la columna 0 del terreno y una fila aleatoria.

Los obstáculos deben posicionarse aleatoriamente al inicializar el juego, a excepción de los amigos de Nom Nom quienes aparecerán a lo largo del juego. Cabe destacar que no pueden posicionarse distintos obstáculos en la misma posición que otro objeto, o personaje.

El orden de posicionamiento de los obstáculos al inicializar el nivel es indiferente, siempre y cuando se respete lo enunciado en el párrafo anterior.

3.3. Herramientas en la mochila

Las herramientas ubicadas en la mochila, estarán disponibles desde que comienza el nivel, y servirán de ayuda para encontrar a Chloe. Estos tienen un uso limitado y se irán desgastando a medida que el jugador se mueva en el terreno. Para activar alguna de estas herramientas, se utilizará la primera hallada en el vector.

Le permitirán al personaje visualizar parte del terreno pero cada una de ellas de una forma particular, con esto, se podrán ver tanto herramientas, como obstáculos y hasta Chloe si es que éstos se encuentran en el rango que la herramienta ilumina.

- **Linterna**: tendrá una vida útil de 10 movimientos. Sin embargo, como Pardo es experto en acampar y andar de noche en los bosques, tendrá 15 usos en total. La iluminación la dará por fila o por columna, dependiendo cual sea su último movimiento desde donde está posicionado el personaje y hasta el final del bosque, es decir que no iluminará hacia atrás (ver sección 3.5 Modo de juego).
- **Velas**: iluminan en todas las posiciones adyacentes al personaje. Su duración será de 5 movimientos. Cada personaje comenzará la partida con 4 velas en su mochila. Excepto Polar, que por hacer yoga y estar siempre listo para meditar, siempre lleva velas de más consigo, por lo que comenzará con 6 velas en su mochila (ver sección 3.5 Modo de juego).
- **Bengalas**: iluminan con una distancia Manhattan de 3 posiciones, que se dará en una ubicación random. Su duración será de 3 movimientos, donde se deberá actualizar la posición en cada uno. Cada personaje empezará sin bengalas, excepto por Panda, que como es amante de las fiestas, le sobraron 2 de Año Nuevo (ver sección 3.5 Modo de juego).

3.4. Herramientas recolectables

Las herramientas recolectables son elementos que estarán en el mapa a lo largo del juego y ayudarán a los osos a encontrar a Chloe.

- **Pilas**: al agarrar una pila, se le sumarán 5 movimientos a la linterna de la mochila. En el bosque habrá 30 pilas.
- **Velas**: al agarrar una vela, se guardará en la mochila del oso. En el bosque habrá 30 velas.
- **Bengalas**: al agarrar una bengala, se guardará en la mochila del oso. En el bosque habrá 10 bengalas.

El hecho de agarrar cualquiera de estos objetos, está dado por pararse en la posición en la que se encuentra la herramienta.

Las herramientas deben posicionarse aleatoriamente al inicializar el juego, cabe destacar que no pueden posicionarse distintas herramientas en la misma posición o en una posición donde ya existe un obstáculo, o la posición de uno de los dos personajes.

El orden de posicionamiento de las herramientas al inicializar el juego es indiferente, siempre y cuando se respete lo enunciado en el párrafo anterior.

Es importante remarcar que al posicionarse sobre uno de estos objetos, éste debe ser eliminado de las herramientas.

3.5. Modo de juego

El orden de posicionamiento al iniciar el juego de todos los elementos debe ser el siguiente:

1. Personaje.
2. Chloe.
3. Obstáculos.
4. Herramientas.

El objetivo del juego es encontrar a Chloe en un terreno de 20x30 antes de que comience el toque de queda, por lo que el oso tendrá solo dos minutos (120 segundos) para hallar a su amiga. Para ello, una vez encontrada la niña, se deberá sumar el tiempo perdido a causa de los obstáculos (acumulado a lo largo del juego) al tiempo transcurrido desde que se inició el cronómetro.

Como se hizo tarde y ya está oscureciendo, el oso no podrá ver el bosque. Además, en el mismo hay obstáculos que dificultarán su misión de encontrar a Chloe. Para ayudarse, cuenta con herramientas en su mochila y con elementos ubicados a lo largo del mapa que podrá recoger a medida que vaya avanzando.

El oso puede moverse de a una posición, y en cada turno, podrá elegir moverse o activar/desactivar sus herramientas (linterna, bengalas o velas). Para activar una herramienta debe presionar la tecla que se indica en las especificaciones y para desactivarla pueden suceder tres cosas: por un lado, la herramienta puede desactivarse automáticamente cuando se terminen los movimientos de ella, por otro, puede ser desactivada por el jugador presionando la misma tecla que usó para encenderla, y por último, activando otra herramienta, a excepción de la bengala, que no puede ser desactivada por el usuario y se apagará automáticamente cuando no tenga más movimientos.

A continuación se muestran algunos ejemplos del movimiento del oso y el funcionamiento de las herramientas:

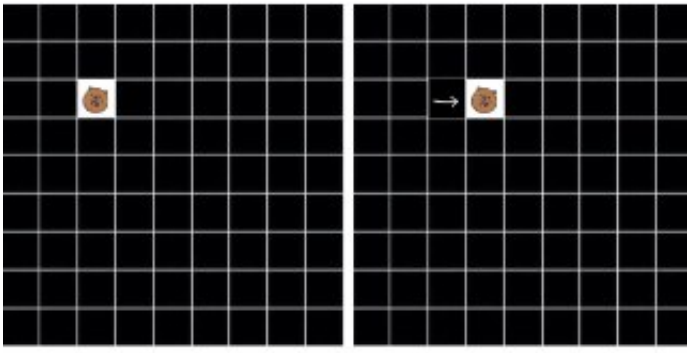


Figura 1: Oso se mueve hacia la derecha

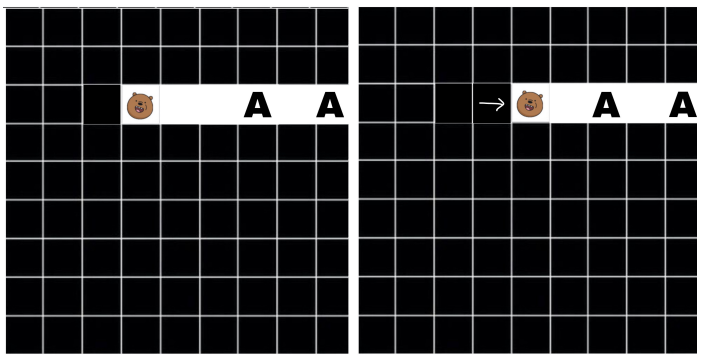


Figura 2: Prende la linterna y vuelve a moverse hacia la derecha

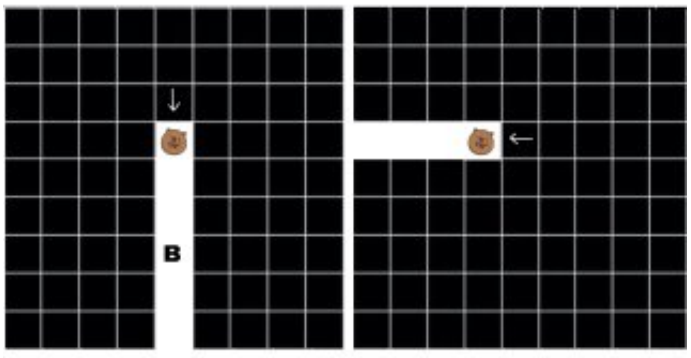


Figura 3: Con la linterna prendida, se mueve hacia abajo y luego hacia la izquierda

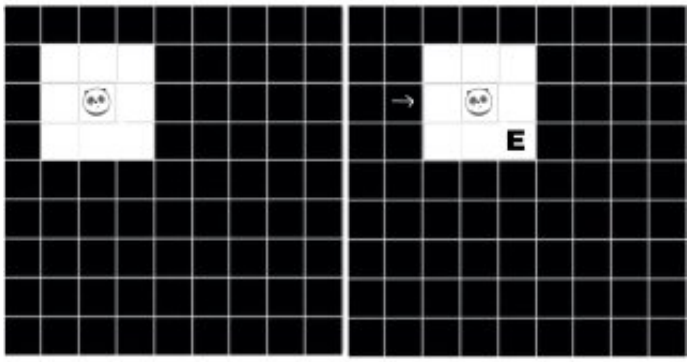


Figura 4: Prende una vela y se mueve hacia la derecha

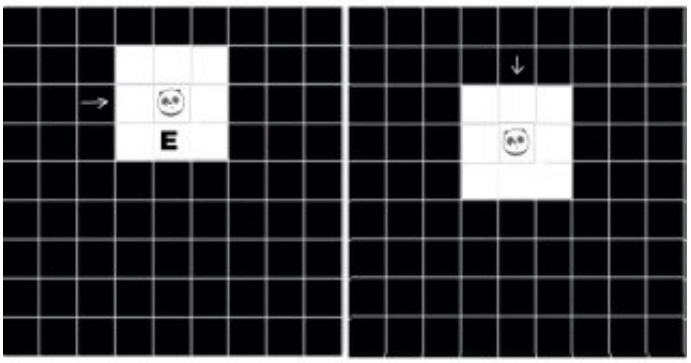


Figura 5: Con la vela encendida, se mueve hacia la derecha y luego hacia abajo

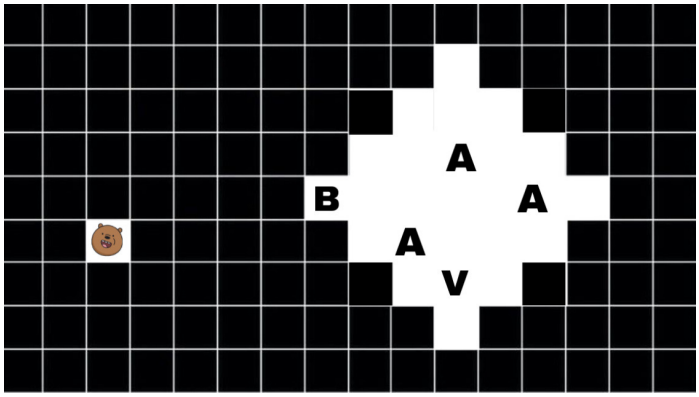


Figura 6: Se enciende una bengala

4. Especificaciones

4.1. Biblioteca osos_contra_reloj.h

```c
#ifndef __OSOS_CONTRA_RELOJ_H__
#define __OSOS_CONTRA_RELOJ_H__

#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>

#define MAX_OBSTACULOS 600
#define MAX_HERRAMIENTAS 600


typedef struct coordenada {
        int fil;
        int col;
} coordenada_t;

typedef struct elemento_del_mapa {
        char tipo;
        coordenada_t posicion;
        bool visible;
} elemento_del_mapa_t;

typedef struct elemento_mochila {
        char tipo;
        int movimientos_restantes;
} elemento_mochila_t;

typedef struct personaje {
        char tipo;
        coordenada_t posicion;
        elemento_mochila_t mochila[MAX_HERRAMIENTAS];
        int cantidad_elementos;
        int elemento_en_uso; // -1 si no hay nada en uso, o posicion del vector de lo que esta en uso.
        double tiempo_perdido;
        char ultimo_movimiento;
} personaje_t;

typedef struct juego {
        elemento_del_mapa_t obstaculos[MAX_OBSTACULOS];
        int cantidad_obstaculos;
        elemento_del_mapa_t herramientas[MAX_HERRAMIENTAS];
        int cantidad_herramientas;
        personaje_t personaje;
        coordenada_t amiga_chloe;
        bool chloe_visible;
} juego_t;


/*
 * Inicializará el juego, cargando toda la información inicial
 * y los datos del personaje.
 */
void inicializar_juego(juego_t* juego, char tipo_personaje);

/*
 * Recibe un juego con todas sus estructuras válidas.
 *
 * El juego se dará por terminado, si el personaje encontró a Chloe.
 * Devolverá:
 * -> 0 si el estado es jugando.
 * -> -1 si el estado es terminado.
 */

int estado_juego(juego_t juego);

/*
 * Mueve el personaje en la dirección indicada por el usuario o habilita
 * cualquiera de las herramientas y actualiza el juego según los elementos
 * que haya en el camino del personaje.
 * El juego quedará en un estado válido al terminar el movimiento.
 * El movimiento será:
```

```
 * -> W: Si el personaje debe moverse para la arriba.
 * -> A: Si el personaje debe moverse para la izquierda.
 * -> S: Si el personaje debe moverse para la abajo.
 * -> D: Si el personaje debe moverse para la derecha.
 * -> L: Si el personaje quiere encender una linterna.
 * -> V: Si el personaje quiere encender una vela.
 * -> E: Si el personaje quiere encender la bengala.
 * -> T: Si el personaje quiere ver el tiempo restante.
 * En caso de que querer activar una herramienta, y no tenga mas movimientos, no deberá
 * activarse ninguna ventaja.
 * Si se aprieta una tecla de iluminación y esta ya está siendo usada, se desactivará colocando
 * el int elemento_en_uso en -1.
 */
void realizar_jugada(juego_t* juego, char jugada);

/*
 * Mostrará el juego por pantalla.
 * Se recomienda mostrar todo lo que sea de utilidad para el jugador.
 */
void mostrar_juego(juego_t juego);

#endif /* __OSOS_CONTRA_RELOJ_H__ */
```

4.2. Convenciones

Se deberá utilizar la siguiente convención para los obstáculos y herramientas:

- **Árbol**: A.
- **Piedra**: R.
- **Koalas**: K.
- **Linterna**: L.
- **Vela**: V.
- **Bengala**: E.
- **Pila**: B.

Y para los personajes:

- **Polar**: I.
- **Pardo**: G.
- **Panda**: P.
- **Chloe**: C.

4.3. Biblioteca test_de_personalidad.h

Se debe crear una biblioteca con el trabajo práctico 1, ésta biblioteca solo tendrá un procedimiento con la siguiente firma:

```
void test_de_personalidad(char* personalidad_detectada);
```

5. Resultado esperado

Se espera que se creen las funciones y procedimientos para que el juego se desarrolle con fluidez.

Muchas de las funcionalidades quedan a criterio del alumno, solo se pide que se respeten las estructuras y especificaciones brindadas.

El trabajo creado debe:

- Interactuar con el usuario.
- Mostrarle al usuario, de forma clara el terreno, logrando que éste pueda interpretar que parte del terreno está oculta y que parte está iluminada.
- Mostrarle al jugador la información del juego a cada momento.
- Informarle al jugador correctamente cualquier dato que haya sido ingresado incorrectamente.
- Informarle al jugador si ganó o perdió.
- Cumplir con las buenas prácticas de programación.
- Mantener las estructuras propuestas actualizadas a cada momento.

6. Compilación y Entrega

El trabajo práctico debe ser realizado en un archivo llamado juego.c, y la biblioteca de funciones para jugarlo osos_contra_reloj.c y osos_contra_reloj.h, y debe poder ser compilado sin errores con el comando:

```
gcc juego.c osos_contra_reloj.c test_de_personalidad.c utiles.o -o juego -std=c99 -Wall -Wconversion
    -Werror -lm
```

test_de_personalidad.c y test_de_personalidad.h corresponden a la biblioteca creada con la parte del trabajo práctico 1 para obtención del personaje que nos representará durante la partida.

utiles.o es un archivo compilado realizado por la cátedra, que pondrá a su disposición 3 funciones que pueden ser, justamente, útiles y su funcionamiento se explica en el anexo.

Por último debe ser entregado en la plataforma de corrección de trabajos prácticos **Chanutron2021** (patente pendiente), en la cual deberá tener la etiqueta **¡Exito!** significando que ha pasado las pruebas a las que la cátedra someterá al trabajo.

Para la entrega en **Chanutron2021** (patente pendiente), recuerde que deberá subir un archivo **zip** conteniendo únicamente los archivos antes mencionados, sin carpetas internas ni otros archivos. De lo contrario, la entrega no será validada por la plataforma.

IMPORTANTE! Esto no implica necesariamente haber aprobado el trabajo ya que además será corregido por un colaborador que verificará que se cumplan las buenas prácticas de programación.

7. Anexos

7.1. Distancia Manhattan

Para obtener la distancia entre 2 puntos mediante este método, se debe conocer a priori las coordenadas de dichos puntos.

Luego, la distancia entre ellos es la suma de los valores absolutos de las diferencias de las coordenadas. Se ve claramente en los siguientes ejemplos:

- La distancia entre los puntos (0,0) y (1,1) es 2 ya que: $|0 - 1| + |0 - 1| = 1 + 1 = 2$
- La distancia entre los puntos (10,5) y (2,12) es 15 ya que: $|10 - 2| + |5 - 12| = 8 + 7 = 15$
- La distancia entre los puntos (7,8) y (9,8) es 2 ya que: $|7 - 9| + |8 - 8| = 2 + 0 = 2$

7.2. Obtención de números aleatorios

Para obtener números aleatorios debe utilizarse la función **rand()**, la cual está disponible en la biblioteca stdlib.h.

Esta función devuelve números pseudo-aleatorios, esto quiere decir que, cuando uno ejecuta nuevamente el programa, los números, aunque aleatorios, son los mismo.

Para resolver este problema debe inicializarse una semilla, cuya función es determinar desde donde empezarán a calcularse los números aleatorios.

Los números arrojados por **rand()** son enteros sin signo, generalmente queremos que estén acotados a un rango (queremos números aleatorios entre tal y tal). Para ésto, podemos obtener el resto de la división de **rand()** por el valor máximo del rango que necesitamos.

Aquí dejamos un breve ejemplo de como obtener números aleatorios entre 10 y 30.

```c
#include <stdio.h>
#include <stdlib.h> // Para usar rand
#include <time.h>   // Para obtener una semilla desde el reloj

int main(){
        srand ((unsigned)time(NULL));
        int numero = rand() % 20 + 10; // la amplitud del rango es 20 y el valor mínimo es 10.
        printf("El valor aleatorio es: %i\n", numero);

        return 0;
}
```

7.3. Limpiar la pantalla durante la ejecución de un programa

Muchas veces nos gustaría que nuestro programa pueda verse siempre en la pantalla sin ver texto anterior.

Para ésto, podemos utilizar la llamada al sistema **clear**, de esta manera, limpiaremos todo lo que hay en nuestra terminal hasta el momento y podremos dibujar la información actualizada.

Y se utiliza de la siguiente manera:

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
        printf("Escribimos algo\n");
        printf("que debería\n");
        printf("desaparecer...\n");

        system("clear"); // Limpiamos la pantalla

        printf("Solo deberiamos ver esto...\n");
        return 0;
}
```

7.4. utiles.o

En esta oportunidad la cátedra pondrá a disposición una biblioteca con 3 funciones, una para iniciar el cronómetro, otra para obtener los segundos que pasaron desde que se inició el cronómetro y la última para detener el cronómetro.

Es pertinente aclarar que se envían 2 archivos **utiles.o**, uno para arquitecturas de 32 bits y otra para arquitecturas de 64 bits, use la que corresponda a su sistema operativo.

El .h de la biblioteca se muestra a continuación con las firmas propuestas.

```c
#ifndef __UTILES_H__
#define __UTILES_H__

/*
 * Inicia el cronómetro.
 */
void iniciar_cronometro();

/*
 * Retorna el tiempo actual en segundos desde que se inicio el crónometro.
 * En caso de error, retorna un valor menor a 0.
 */
double tiempo_actual();

```

```c
/*
 * Detiene el cronómetro, devolviendo el tiempo total desde que se inicio en segundos.
 * En caso de error, retorna un valor menor a 0.
 */
double detener_cronometro();


#endif /* __UTILES_H__ */
```

8. Referencias

https://escandalosos.fandom.com/es/wiki/Escandalosos_Wiki

9. Cambios

9.1. 23/05

- La posición de las bengalas irán variando aleatoriamente por cada movimiento gastado.
- La función estado_juego() devolverá -1 si el juego finalizó, es decir, si se encontró a Chloe, o 0 si aún se sigue jugando.
- Se cambió en el utiles.h del enunciado int por double como retorno de las funciones.
- Chloe se posicionará aleatoriamente por todo el terreno.
- Las teclas para ingresar tendrán que ser en mayúscula.
- Panda comienza a ver a Chloe cuando tenga 30 segundos o más en el tiempo perdido.
- Se agregó un bool visible en el struct del elemento, y un bool chloe_visible en el struct del juego.
- Se cambió la tecla para agregar una bengala. En lugar de 'B', será 'E'.