.UBAfiuba
FACULTAD DE INGENIERÍA

ALGORITMOS Y PROGRAMACIÓN I - CURSO CAMEJO

Trabajo Práctico Nº 2

EL PLAN DE SÍNDROME

Fecha presentación	29/9/2022
Fecha entrega	20/10/2022

Segundo cuatrimestre 2022

1. Introducción

Bob, mejor conocido como Mr. Increíble, volvió a sus días como héroe luego de haber sido llamado a una misteriosa misión, durante la cual descubrió que la persona que lo había contratado era en realidad un supervillano llamado "Síndrome".

Gracias a tu ayuda, Mr. Increíble pudo desbloquear la computadora de Síndrome y descubrir cuál es su plan: crear robots para que ataquen a la ciudad y controlarlos para que parezca que es un héroe. Sin embargo, como el supervillano se da cuenta que Mr. Increíble ya conoce sus intenciones, lo toma prisionero. No es sino hasta que Helen visita a Edna para preguntarle sobre el paradero de su esposo, que esta le revela los nuevos trajes que diseñó para ella y sus hijos, y activa una radiobaliza con la que localizan a Bob.

Helen decide rápidamente llamar a sus hijos Violeta y Dash, dado que Jack-Jack es inofensivo todavía (o eso creen) e ir al rescate de su esposo.

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 y claridad del código.

3. Enunciado

Como desarrolladores de este juego, debemos ayudar a la familia Increíble a salvar a Bob y escapar.

El juego consiste en una habitación donde estará Bob y su familia, ésta habitación estará delimitada por bordes del terreno y dividida en 4 cuadrantes y cada integrante de la familia estará en uno diferente. Su objetivo es encontrarse entre ellos, uno por uno, para finalmente llegar hasta Bob y así poder escapar.
Dicha habitación tendrá dentro robots con lásers y varias pinzas, los cuales dificultarán la tarea de la familia de reunirse.

Cada integrante de la familia tendrá una cantidad de movimientos iniciales, los cuales se reinician cuando se llega al siguiente familiar. También cada integrante podrá contar con un poder particular, el cual será activado al obtener el supertraje.

Se iniciará el juego siendo ElasticGirl, y cada vez que se llegue a un nuevo personaje, éste pasará a ser el personaje con el que se juega, esto quiere decir que durante el juego podremos tener poderes distintos, dependiendo con cuál de los personajes estemos jugando en ese momento.

El orden de inicialización de los elementos debe ser en el mismo orden en el cual se presentan en este trabajo práctico.

3.1. Obstáculos

Los obstáculos deben posicionarse aleatoriamente al inicializar el juego, respetando los que deben estar dentro de un cuadrante en particular. Cabe destacar que no pueden posicionarse distintos obstáculos en la misma posición que otro objeto, o personaje.

3.1.1. Robots

Habrá 4 robots, cada uno estará en un cuadrante y tendrán dos lásers en forma de L los cuales girarán 45° en sentido horario luego de cada movimiento del personaje. La dirección de inicio de los lásers debe ser aleatoria entre las direcciones arriba, abajo, derecha e izquierda.

Dado que los lásers giran, pueden llegar a pisarse con algún otro elemento del juego (pinzas, supertraje, lásers de otro robot o un robot), en caso de que esto suceda, se pide que se priorice al momento de mostrar al láser excepto que con quien choque sea otro robot, en ese caso se pide mostrar el robot. También se pide que el efecto producido si el personaje llega a tocar ese lugar compartido sea el del láser. También se permite que los lásers de un robot queden dentro de un cuadrante al cual no pertenece ese robot.

La longitud de los lásers dependerán de si se pudo o no desactivar el sistema de seguridad. El sistema está desactivado si se logró adivinar la contraseña del trabajo práctico anterior.

Si la contraseña fue adivinada en su totalidad, la longitud de los lásers será 3. En caso de no haber adivinado la contraseña completa, la longitud de los lásers será 5.

Si el personaje pisa un láser o a uno de los robots, se termina el juego.

3.1.2. Pinzas

Habrá 4 pinzas por cuadrante, si el personaje pisa alguna de ellas, las pinzas lo llevarán aleatoriamente a una de las posiciones que están alrededor del robot que se encuentra dentro del cuadrante inicial de ese personaje.

3.2. Poderes

En cada cuadrante, habrá un supertraje el cual, al pisarlo, permitirá que el personaje pueda activar su poder cuando quiera. Si se quiere activar sin antes haber pisado el supertraje, no se podrá acceder a los beneficios del poder.

Cada personaje deberá recolectar el supertraje de su cuadrante original, si pisa alguno de otro cuadrante, este no tendrá efecto.

Los supertrajes deben posicionarse de forma aleatoria al inicio del juego, uno por cuadrante y no deben estar en la misma posición que otro objeto. Una vez recolectado, el supertraje debe desaparecer.

El poder puede activarse una única vez por personaje y dura 5 movimientos del personaje desde la activación. Luego de esos 5 movimientos, el poder se desactiva y no se puede volver a utilizar.

3.3. Personajes

Los personajes deben posicionarse de forma aleatoria, siempre manteniéndose dentro de su cuadrante inicial.

Cuando un personaje se posiciona sobre el siguiente, se transforma en éste. El orden en el que deben encontrarse es el mostrado debajo:

- **ElasticGirl**:
  - **Poder**: su poder es estirarse, por lo que puede avanzar 3 posiciones en un solo movimiento.
  - **Cuadrante**: 1
  - **Movimientos iniciales**: 25
- **Violeta**:
  - **Poder**: su poder es crear un escudo, por lo que, al activar ese escudo, los lásers no la dañarán.
  - **Cuadrante**: 2
  - **Movimientos iniciales**: 30
- **Dash**:
  - **Poder**: al activarse su poder, se dirigirá a una posición random dentro del cuadrante donde se encuentra Mr. Increíble.
  - **Cuadrante**: 3

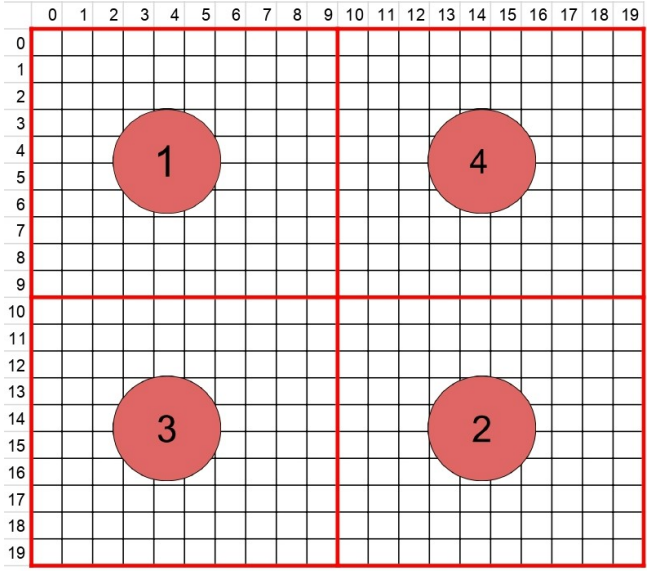
- **Movimientos iniciales**: 20

- **Mr. Increíble**:
  - **Poder**: su superfuerza evita que le afecten las pinzas.
  - **Cuadrante**: 4
  - **Movimientos iniciales**: 15

Esto quiere decir, que ElasticGirl debe ir a buscar a Violeta, Violeta a Dash y Dash a Mr. Increíble. En caso de que algún personaje llegue al lugar donde está otro que no es el siguiente a ese personaje, no debe cambiar.

3.4. Terreno

El terreno será representado con una matriz de 20x20, dividida en 4 cuadrantes de la siguiente forma:



3.5. Modo de juego

El personaje se podrá mover en 4 direcciones:

- **Arriba**: W
- **Abajo**: S
- **Derecha**: D
- **Izquierda**: A

Para **activar el poder** se usará la letra C, siempre y cuando el personaje haya recolectado su supertraje.

Cada movimiento realizado hará avanzar al personaje un casillero en la dirección elegida. Cada vez que el personaje se mueve, se debe restar un movimiento. No resta movimientos activar el poder.

Para ganar el juego, Mr. Increíble debe llegar al casillero de salida, el cual estará siempre ubicado en la esquina superior derecha del cuadrante 4. Cabe aclarar que ningún elemento puede inicializarse sobre el casillero de salida.

4. Especificaciones

4.1. Convenciones

Se deberá utilizar la siguiente convención para los obstáculos:

- **Robot**: R.
- **Láser**: L.

- **Pinza**: P.
- **Supertraje**: T.
- **Salida**: S.

Y para los personajes:

- **Mr. Increíble**: I.
- **Violeta**: V.
- **Dash**: D.
- **ElasticGirl**: E.

4.2. Biblioteca increidle.h

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

```
void adivinar_contrasenia(char contrasenia_adivinada[MAX_CONTRASENIA]);
```

4.3. Funciones y procedimientos

Para poder ayudar a la familia Increíble, se pedirá implementar las siguientes funciones y procedimientos.

```
#ifndef __KRONOS_H__
#define __KRONOS_H__

#include <stdbool.h>

#define MAX_LASERS 100
#define MAX_ROBOTS 10
#define MAX_PERSONAJES 10
#define MAX_PINZAS 100
#define MAX_SUPERTRAJES 10

typedef struct coordenada{
    int fila;
    int columna;
} coordenada_t;

typedef struct robot{
    coordenada_t posicion;
    coordenada_t lasers[MAX_LASERS];
    int tope_lasers;
} robot_t;

typedef struct supertraje{
    coordenada_t posicion;
    int cuadrante;
    bool recolectado;
    bool usado;
} supertraje_t;

typedef struct personaje{
    bool poder_activado;
    bool tiene_supertraje;
    coordenada_t posicion;
    int movimientos;
    int movimientos_con_poder;
    int cuadrante_inicial;
} personaje_t;

typedef struct juego{
    personaje_t personajes [MAX_PERSONAJES];
    int tope_personajes;
    robot_t robots [MAX_ROBOTS];
    int tope_robots;
    coordenada_t pinzas [MAX_PINZAS];
    int tope_pinzas;
    supertraje_t supertrajes [MAX_SUPERTRAJES];
    int tope_supertraje;
```

```c
    int longitud_laser;
    int id_personaje_actual;
} juego_t;


/*
 * Inicializará el juego, cargando toda la información inicial de los robots, los supertrajes, el personaje, los lásers y las pinzas.
 * El campo id_personaje_actual comienza en 1.
 */
void inicializar_juego(juego_t* juego, bool contrasenia_completa);

/*
 * Moverá el personaje y se realizarán todas las acciones necesarias en caso de chocas con algún elemento.
 */
void realizar_jugada(juego_t* juego, char movimiento);

/*
 * Imprime el juego por pantalla.
 */
void imprimir_terreno(juego_t juego);

/*
 * El juego se dará por ganado si se está sobre el casillero de salida siendo el personaje Mr. Increíble y perdido si el personaje (cualquiera sea) se queda sin movimientos.
 * Devuelve:
 * -> 1 si se ganó el juego.
 * -> 0 si el juego aún se está jugando.
 * -> -1 se se perdió el juego.
 */
int estado_juego(juego_t juego);

#endif /* __KRONOS_H__ */
```

Observación: Queda a criterio del alumno/a hacer o no más funciones y/o procedimientos para resolver los problemas presentados. No se permite agregar funciones al .h presentado por la cátedra, como tampoco modificar las funciones dadas.

5. Resultado esperado

Se espera que se creen las funciones y procedimientos para que el juego se desarrolle con fluidez. Así mismo, se espera que se respeten las buenas prácticas de programación.

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 y todos sus elementos.
- 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 kronos.c, lo que sería la implementación de la biblioteca kronos.h. Además, deberán crear una biblioteca con lo resuelto en el trabajo práctico 1. El trabajo debe ser compilado sin errores al correr el siguiente comando:

```
gcc *.c -o juego -std=c99 -Wall -Wconversion -Werror -lm
```

Aclaración: El main tiene que estar desarrollado en un archivo llamado juego.c, el cual manejará todo el flujo del programa.

Lo que nos permite ***.c** es agarrar todos los archivos que tengan la extensión .c en el directorio actual y los compile todos juntos. Esto permite que se puedan crear bibliotecas a criterio del alumno, aparte de las exigidas por la cátedra.

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

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.

Para la entrega en **AlgoTrón** (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.

7. Anexos

7.1. 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;
}
```

7.2. 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;
}
```