Algoritmos y Programación I

Curso Mendez

.UBAfiuba
FACULTAD DE INGENIERÍA

TP Primera Parte

La Sombra de Mac

Fecha Presentación	07/10/2021
Fecha Entrega	28/10/2021

1. Introducción

Mac es un niño de ocho años con gran inteligencia y creatividad. Él es una persona sensata y moral, y puede ser a través de Bloo, su amigo imaginario, que hace y dice todas las cosas que quiere pero no puede. Por lo tanto, Bloo existe como un desafío a la moralidad de Mac. Pasa una gran porción de sus días involucrándose en sus travesuras dentro y fuera de Foster's. La **Mansión Foster** para amigos imaginarios es un inmenso orfanato con estilo victoriano al que los amigos imaginarios se van a vivir cuando ya no pueden pertenecerle a sus creadores. Ahí se muda Bloo, y Mac lo visita todos los días para asegurarse de que Bloo no sea adoptado.

Bloo es el amigo imaginario de Mac, a menudo egoísta, rebelde, inquieto, arrogante o en busca de atención. No obstante, puede cambiar su actitud en lealtad a su creador. Su cuerpo humanoide completamente azul es comparable a la de un guante de cocina o un fantasma del videojuego Pac-Man. A menudo, tiene una tendencia a aceptar ideas escandalosas como un hecho para explicar sucesos aparentemente mundanos, y somete a otros a su voluntad, yendo tan lejos como para hacerlos ir en contra de sus estándares morales. Bloo también muestra pasión por la pelota de pádel, aunque nunca logra hacer que la pelota golpee la paleta, a lo que insiste en que todas sus paletas están rotas.

2. Objetivo

El presente trabajo práctico tiene como objetivo evaluar a los alumnos en aspectos fundamentales de la programación. Estos aspectos son:

- Validación de datos ingresados por el usuario.
- Diseño y desarrollo de funcionalidades de una biblioteca con un contrato preestablecido.
- El correcto uso de estructuras de control.
- Tipos de dato simples y estructurados.
- Buenas prácticas de programación.
- Modularización.

3. Enunciado

Como desarrolladores de este juego, debemos ayudar a Mac y a Bloo a volver de la mansión luego de haber pasado una tarde explorando la ciudad.

Deberás guiarlos en su camino ingresando una dirección en la que caminar, para poder encontrar la llave que abre la puerta de la mansión. Pero cuidado! Bloo, para hacer la vuelta de forma divertida, va a caminar en espejo con respecto a Mac. Es decir, si Mac se dirige en algun sentido horizontal (izquierda o derecha), Bloo irá en el camino contrario.

Para comenzar, Mac **(M)** tendrá 3 vidas disponibles, sus puntos iniciarán en 0 y su posición será aleatoria. En cuanto a Bloo **(B)**, su sombra, empezará estando vivo en la misma fila que Mac, y la columna se calculará como la resta de las columnas totales del terreno y la columna de Mac.

Como el camino es largo, y no todo el camino es igual, deberán pasar por 3 niveles distintos, donde cada nivel estará delimitado por los bordes del terreno. En caso de que Bloo o Mac, se choque con alguno de estos bordes, el mismo no deberá moverse.

Deberán crearse su propio programa principal para que el juego quede inicializado y en un estado válido para empezar a jugar.

Podrás ayudarlos a volver a la mansión?

4. Funciones y Procedimientos a realizar

Como buenos amigos, vamos a ayudar a Mac y Bloo a cumplir su objetivo de volver a la mansión. Para poder lograr esto, se pedirá implementar algunas funciones y procedimientos.

```
#ifndef __LA_SOMBRA_DE_MAC__
#define __LA_SOMBRA_DE_MAC__

#include <stdbool.h>

#define MAX_FILAS 20
#define MAX_COLUMNAS 20
```

```c
#define MAX_ELEMENTOS 500

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

typedef struct elemento {
    coordenada_t posicion;
    char tipo;
}elemento_t;

typedef struct personaje {
    coordenada_t posicion;
    int vida;
    int puntos;
    bool tiene_llave;
    bool interruptor_apretado;
}personaje_t;

typedef struct sombra {
    coordenada_t posicion;
    bool esta_viva;
}sombra_t;

typedef struct juego {
    personaje_t personaje;
    sombra_t sombra;
    // se vienen cositas lindas.
}juego_t;

/*
*    Función que recibe dos coordenadas, devuelve true si las
* coordenadas son iguales, sino false.
*/
bool coordenadas_iguales(coordenada_t a, coordenada_t b);


/*
*   Procedimiento que dejará la estructura personaje_t en un
* estado válido para iniciar el juego.
* 'arranque_personaje' es un parámetro que representa la posición
* en la arrancará el personaje.
*/
void inicializar_personaje(personaje_t* ref_personaje, coordenada_t arranque_personaje);


/*
*   Procedimiento que dejará una estructura elemento_t que
* represente a la sombra, en un estado válido para iniciar el juego.
* 'arranque_sombra' es un parámetro que representa la posición
* en la arrancará la sombra.
*/
void inicializar_sombra(sombra_t* ref_sombra, coordenada_t arranque_sombra);


/*
*   La función recibe un caracter y devuelve true
* si el movimiento recibido concuerda con la convención
* propuesta, sino false.
* -> 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.
*/
bool es_movimiento_valido(char movimiento);


/*
*   Procedimineto que se encargará de pedirle al usuario
* que ingrese un movimiento hasta que sea válido.
*/
void pedir_movimiento(char* ref_movimiento);


/*
```

```
84 *    La función recibe una coordenada, devuelve true
85 * si la posición dentro de las dimensiones dadas, sino false.
86 * Una posición nunca podrá tener un valor negativo y no tendrá un
87 * valor más alto que los máximos.
88 */
89 bool esta_dentro_rango(coordenada_t posicion, int max_alto, int max_ancho);
90
91
92 /*
93 *    El procedimiento recibe el personaje y la sombra por referencia y el movimiento,
94 * y según este último, los mueve acordemente.
95 *
96 * El personaje se mueve de la forma:
97 * -> W: Si el personaje debe moverse para la arriba.
98 * -> A: Si el personaje debe moverse para la izquierda.
99 * -> S: Si el personaje debe moverse para la abajo.
100 * -> D: Si el personaje debe moverse para la derecha.
101 *
102 *   La sombra se mueve de igual forma para el eje vertical, pero
103 * para el eje horizontal se moverá de forma inversa:
104 *
105 * -> A: Si el personaje debe moverse para la derecha.
106 * -> D: Si el personaje debe moverse para la izquierda.
107 */
108 void mover_personaje(personaje_t* ref_personaje, sombra_t* ref_sombra, char movimiento);
109
110
111 /*
112 *   Función que dados una altura máxima y un ancho máximo
113 * devuelve una coordenada aleatoria. Una posición nunca podrá
114 * tener un valor negativo, ni ser mayor a sus máximos.
115 */
116 coordenada_t coordenada_aleatoria(int max_alto, int max_ancho);
117
118
119 /*
120 *    La función recibe un vector de elementos, su respectivo tope y una coordenada,
121 * devuelve la posición del elemento del vector, que coincida con la coordenada pasada 'posicion',
122 * si no se encuentra ningún elemento, se devolverá -1.
123 */
124 int buscar_elemento(elemento_t elementos[MAX_ELEMENTOS], int tope, coordenada_t posicion);
125
126
127 /*
128 *    Procedimiento que recibe el juego e imprime
129 * toda su información por pantalla.
130 */
131 void imprimir_terreno(juego_t juego);
132
133 #endif
```

Observación: Queda a critero del alumno/a el hacer o no, más funciones y/o procedimientos para resolver los problemas presentados. No se permite agregar dichas firmas al .h.

5. Resultado esperado

Se espera que el trabajo creado cumpla con las buenas prácticas de programación y todas las funciones y procedimientos pedidos funcionen acorde a lo solicitado, respetando las pre y post condiciones propuestas.

6. Compilación y Entrega

El trabajo práctico debe ser realizado en un archivo llamado la_sombra_de_mac.c, lo que sería la implementación de la biblioteca la_sombra_de_mac.h. El objetivo es que sea compilado sin errores al correr desde la terminal el comando:

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

Luego, juego.c será el archivo que ustedes deberán crear y utilizar para probar que las funcionalidades desarrolladas cumplan con lo pedido.

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** que contenga únicamente los archivos antes mencionados, sin carpetas internas ni otros archivos. De lo contrario, la entrega no será validada por la plataforma.

IMPORTANTE! Obtener la etiqueta **¡Exito!** en **Chanutron2021** (patente pendiente) no implica necesariamente haber aprobado el trabajo. El trabajo será corregido por un colaborador que verificará que se cumplan las buenas prácticas de programación.

7. Anexos

7.1. Ejemplos

Ejemplo 1:

```
bool coordenadas_iguales(coordenada_t primera_coordenada, coordenada_t segunda_coordenada);
```

Al realizar el llamado a la función deberá devolver:

```
coordenada_t primera_coordenada;
coordenada_t segunda_coordenada;

primera_coordenada.fila = 3;
primera_coordenada.col = 2;
segunda_coordenada.fila = 3;
segunda_coordenada.col = 2;

coordenadas_iguales(primera_coordenada, segunda_coordenada);
```

- Devuelve como resultado true.

```
primera_coordenada.fila = 1;
primera_coordenada.col = 2;
segunda_coordenada.fila = 3;
segunda_coordenada.col = 4;

coordenadas_iguales(primera_coordenada, segunda_coordenada);
```

- Devuelve como resultado false.

Ejemplo 2:

```
bool esta_dentro_rango(coordenada_t personaje, int alto, int ancho);
```

Al realizar el llamado a la función deberá devolver:

```
coordenada_t coordenada_personaje;

coordenada_personaje.fila = -1;
coordenada_personaje.col = 31;

esta_dentro_rango(coordenada_personaje, MAX_FILAS, MAX_COLUMNAS);
```

- Devuelve como resultado false.

```
coordenada_personaje.fila = 1;
coordenada_personaje.col = 1;

esta_dentro_rango(coordenada_personaje, MAX_FILAS, MAX_COLUMNAS);
```

- Devuelve como resultado true.

Ejemplo 3:

```
bool es_movimiento_valido(char jugada);
```

Al realizar el llamado a la función deberá devolver:

```
char movimiento = 'W';
es_movimiento_valido(movimiento);
```

- 'movimiento' es un caracter preestablecido como un movimiento válido, por lo tanto la función devuelve true.

```
movimiento = 'F';
es_movimiento_valido(movimiento);
```

- En este caso el valor de 'movimiento' no pertenece a la convención utilizada, por lo tanto la función devuelve false.

Ejemplo 4:

```
int buscar_elemento(elemento_t elementos[MAX_ELEMENTOS],int tope, coordenada_t posicion);
```

Al realizar el llamado a la función deberá devolver:

```
// Ignoren los tipos por el momento.
elemento_t elementos[3];
elementos[0].posicion = {1, 1};
elementos[0].tipo = 'A';

elementos[1].posicion = {2, 2};
elementos[1].tipo = 'A';

elementos[2].posicion = {3, 3};
elementos[2].tipo = 'A';

int tope = 3;
coordenada_t pos_buscado = {5, 0};

buscar_elemento(elementos, tope, pos_buscado);
```

- Devuelve como resultado -1, no se encuentra ningún elemento con la posición buscada.

```
// Ignoren los tipos por el momento.
elemento_t elementos[3];
elementos[0].posicion = {1, 1};
elementos[0].tipo = 'A';

elementos[1].posicion = {2, 2};
elementos[1].tipo = 'A';

elementos[2].posicion = {3, 3};
elementos[2].tipo = 'A';

int tope = 3;
coordenada_t pos_buscado = {2, 2};

buscar_elemento(elementos, tope, pos_buscado);
```

- Devuelve como resultado 1, la posición del elemento en el vector.

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 mismos.

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 esto, 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 29 (inclusives).

```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)); // Genera la semilla aleatoria.
        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;
}
```

Referencias

Links:

https://fostershomeforimaginaryfriends.fandom.com/wiki/Mac
https://fostershomeforimaginaryfriends.fandom.com/wiki/Bloo