Lenguajes Estructurados

01.TP - Programar un juego de Snake

¿Qué es Snake?

- Es un videojuego lanzado a mediados de la década de 1970 que ha mantenido su popularidad desde entonces, convirtiéndose en un clásico.
- En 1998, el Snake obtuvo una audiencia masiva tras convertirse en un juego estándar pre-grabado en los teléfonos Nokia.
- El objetivo del juego es guiar una serpiente por un área rectangular, recogiendo comida, evitando chocar con las paredes o con su propio cuerpo.
- Cada vez que la serpiente come una pieza de comida, su cuerpo crece un segmento, lo que hace que el juego sea cada vez más difícil.
- El juego termina cuando la serpiente choca con la pared o con su propio cuerpo.

Objetivo del ejercicio

Crear un programa en C que simule un tablero de juego de Snake utilizando punteros, programación estructurada y la librería SDL para la representación gráfica.

Descripción

El programa deberá cumplir con los siguientes requisitos:

- Utilizar una estructura para representar las coordenadas (x, y) en el tablero

```
typedef struct {
    int x;
    int y;
} Coordenada;
```

- Utilizar una estructura para representar la serpiente, que incluya un puntero a un arreglo dinámico de coordenadas y un entero que indique el tamaño de la serpiente

```
typedef struct {
    Coordenada* cuerpo;
    int longitud;
} Serpiente;
```

- Implementar las siguientes funciones:
  - `Serpiente* crear_serpiente(int longitud_inicial)`
  - `void destruir_serpiente(Serpiente* serpiente)`
  - `void mover_serpiente(Serpiente* serpiente, int direccion)`
  - `int colision(Serpiente* serpiente)`
  - `int comer(Serpiente* serpiente, Coordenada comida)`

- Utilizar punteros y programación estructurada en todas las funciones.
- Inicializar y configurar SDL para la representación gráfica del juego, con un tamaño de ventana adecuado y una representación simple de los elementos del juego (serpiente, comida).
- Crear un bucle principal que maneje la lógica del juego (mover serpiente, detectar colisiones, generar comida, etc.) y actualice la pantalla utilizando SDL.
- Controlar la velocidad del juego ajustando el tiempo de espera en cada iteración del bucle principal.

Evaluación

Se valorará la eficiencia y organización del código, así como la correcta gestión de memoria dinámica.

Implementación del juego SNAKE en C utilizando SDL2

Este programa implementa una versión simple del juego de la serpiente en C utilizando la biblioteca SDL2 para gráficos.

Estructuras de datos

Coordenadas

Esta estructura representa las coordenadas `x` e `y` en el área de juego.

```
typedef struct {
    int x;
    int y;
} Coordenadas;
```

Serpiente

Esta estructura representa la serpiente en el juego, incluyendo su longitud y las coordenadas de cada parte de su cuerpo.

```
typedef struct
{
    Coordenada *cuerpo;
    int longitud;
} Serpiente;
```

Funciones

inicializar_serpiente

Esta función inicializa la serpiente, estableciendo su longitud. Inicialmente su cuerpo estará formado por una única coordenada en el centro del área de juego.

```
Serpiente *crear_serpiente(int longitud_inicial)
{
    Serpiente *serpiente = (Serpiente *)malloc(sizeof(Serpiente));
    serpiente->longitud = longitud_inicial;
    serpiente->cuerpo = (Coordenada *)malloc(longitud_inicial * sizeof(Coordenada));
    for (int i = 0; i < longitud_inicial; i++)
    {
        serpiente->cuerpo[i].x = i + 1;
        serpiente->cuerpo[i].y = 1;
    }
    return serpiente;
}
```

mover_serpiente

Esta función actualiza la posición de la serpiente según la dirección dada.

```
void mover_serpiente(Serpiente* serpiente, int direccion) {
    // ...
}
```

colision

Esta función verifica si la serpiente colisiona con los límites del área de juego o consigo misma.

```
int colision(Serpiente* serpiente, int ancho, int alto) {
    // ...
}
```

comer

Esta función verifica si la serpiente come la comida y, en caso afirmativo, incrementa su longitud.

```
int comer(Serpiente* serpiente, Coordenadas comida) {
    // ...
}
```

generar_comida

Esta función genera una nueva posición de comida en el área de juego, asegurándose de que no aparezca dentro del cuerpo de la serpiente.

```
Coordenada generar_comida(Serpiente* serpiente, int ancho, int alto) {
    // ...
}
```

dibujar_serpiente

Esta función dibuja la serpiente en el área de juego utilizando el renderizador SDL.

```
void dibujar_serpiente(SDL_Renderer *renderer, Serpiente *serpiente, int size_celda)
{
    SDL_SetRenderDrawColor(renderer, 0, 255, 0, 255);
    SDL_Rect rect;
    rect.w = size_celda;
    rect.h = size_celda;

    for (int i = 0; i < serpiente->longitud; i++)
    {
        rect.x = serpiente->cuerpo[i].x * size_celda;
        rect.y = serpiente->cuerpo[i].y * size_celda;
        SDL_RenderFillRect(renderer, &rect);
    }
}
```

dibujar_comida

Esta función dibuja la comida en el área de juego utilizando el renderizador SDL.

```
void dibujar_comida(SDL_Renderer* renderer, Coordenada comida, int size_celda) {
    // ...
}
```

manejar_eventos

Esta función maneja los eventos de SDL y actualiza las variables salir, iniciar_movimiento y direccion.

```
void manejar_eventos(int *salir, int *iniciar_movimiento, int *direccion)
{
    SDL_Event evento;
    while (SDL_PollEvent(&evento))
    {
        switch (evento.type)
        {

        case SDL_QUIT:
            *salir = 1;
            break;
```

```
        case SDL_KEYDOWN:

            switch (evento.key.keysym.scancode)
            {
            case SDL_SCANCODE_UP:
                if (*direccion != 2)
                {
                    *direccion = 0;
                    *iniciar_movimiento = 1;
                }
                break;
            case SDL_SCANCODE_RIGHT:
                if (*direccion != 3)
                {
                    *direccion = 1;
                    *iniciar_movimiento = 1;
                }
                break;
```

```
            case SDL_SCANCODE_DOWN:
                if (*direccion != 0)
                {
                    *direccion = 2;
                    *iniciar_movimiento = 1;
                }
                break;
            case SDL_SCANCODE_LEFT:
                if (*direccion != 1)
                {
                    *direccion = 3;
                    *iniciar_movimiento = 1;
                }
                break;
            }
        }
    }
}
```

Programa principal

El programa principal inicializa SDL y crea una ventana y un renderizador. Luego, inicializa la serpiente y la comida, y entra en un bucle principal que maneja los eventos, actualiza la posición de la serpiente, verifica las colisiones y la comida, y dibuja la serpiente y la comida en la ventana.

```
int WinMain(int argc, char *argv[])
{
    srand(time(NULL));
    int ancho = 20;
    int alto = 15;
    int size_celda = 32;
    int velocidad = 100;

    if (SDL_Init(SDL_INIT_EVERYTHING) != 0) {
        printf("error initializing SDL: %s\n", SDL_GetError());
    }

    SDL_Window *window = SDL_CreateWindow("Snake",
                                          SDL_WINDOWPOS_CENTERED,
                                          SDL_WINDOWPOS_CENTERED,
                                          ancho * size_celda,
                                          alto * size_celda, 0);

    SDL_Renderer *renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);

    Serpiente *serpiente = crear_serpiente(3);
    Coordenada comida = generar_comida(serpiente, ancho, alto);

    int direccion = 1; // Inicializamos la dirección en -1 (ninguna dirección)
    int iniciar_movimiento = 0 ;
    int salir = 0;
```

```
while (!salir)
{
    manejar_eventos(&salir, &iniciar_movimiento, &direccion);

    if (iniciar_movimiento)
    {
        mover_serpiente(serpiente, direccion);

        if (colision(serpiente, ancho, alto))
        {
            printf("Colision \n");
            while (!salir)
            {
                SDL_Event evento;
                while (SDL_PollEvent(&evento))
                {
                    if (evento.type == SDL_QUIT)
                    {
                        salir = 1;
                    }
                }
                SDL_Delay(100);
            }
        }
```

```
            if (comer(serpiente, comida))
            {
                printf("Comer \n");
                comida = generar_comida(serpiente, ancho, alto);
            }
        }

        SDL_SetRenderDrawColor(renderer, 0, 0, 0, 255);

        SDL_RenderClear(renderer);

        dibujar_serpiente(renderer, serpiente, size_celda);

        dibujar_comida(renderer, comida, size_celda);

        SDL_RenderPresent(renderer);

        SDL_Delay(velocidad);
    }

    destruir_serpiente(serpiente);
    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
```