ALGORITMO Y ESTRUCTURA DE DATOS II

PROYECTO INTEGRADOR

Informe Final

Integrantes del grupo:

German Saucedo - DNI: 45.456.552

Geronimo Serial - DNI: 42.603.578

Matias Jose Torres - DNI: 41843643

Nahuel Alejo Tomei - DNI: 43824602

PROYECTO SNAKE

COMISION C

Fecha de entrega: 27/11/2023

GERMAN SAUCEDO
GERONIMO SERIAL
MATIAS JOSE TORRES
NAHUEL ALEJO TOMEI

Introducción:

En este informe se desglosa el proceso de creación de un emocionante juego al estilo de la viborita. Desde la concepción inicial hasta la concreción de un programa completo, se detallan las diversas fases por las que atravesó este proyecto. Con un enfoque en desafiar las habilidades y destrezas del jugador, el juego fue elaborado por completo utilizando el potente " Lenguaje C". Cada etapa revela los procesos y cambios aplicados al código, permitiendo una identificación detallada de la evolución constante del juego y sus características

Descripción del Proyecto:

Nuestro juego, **Proyecto Snake**, desarrollado en lenguaje C, es una experiencia clásica y adictiva. Controlas una serpiente que crece a medida que consume elementos en el mapa, con el desafío de evitar chocar contra las paredes o contra sí misma. El objetivo es simple: sobrevivir el mayor tiempo posible mientras la serpiente se vuelve más larga y el juego se vuelve más rápido. La mecánica básica y adictiva lo convierte en un entretenimiento accesible para jugadores de todas las edades, ofreciendo diversión sin complicaciones y una curva de aprendizaje rápida para disfrutar al máximo.

Planificación del Proyecto:

El proyecto Snake esta completamente desarrollado en el lenguaje de programación C. Está compuesto por tres códigos auxiliares basados en el Tipo Abstracto de Datos (TAD) y un archivo principal.

El primer archivo, **"corte.h",** se dedica a manipular datos almacenados en archivos, generar informes sobre estos datos y facilitar la interacción con el usuario para llevar a cabo estas operaciones.

El segundo código, "**listaDinamica.h"**, se orienta a implementar una lista dinámica enlazada para gestionar usuarios y sus puntuaciones. Permite realizar inserciones, eliminaciones y visualizaciones dinámicas de estos elementos.

Por otro lado, **"ListaEstatica.h"** se enfoca en implementar una lista estática para almacenar y gestionar usuarios. Ofrece funciones para inicializar, ordenar, visualizar y eliminar elementos de la lista de manera estática.

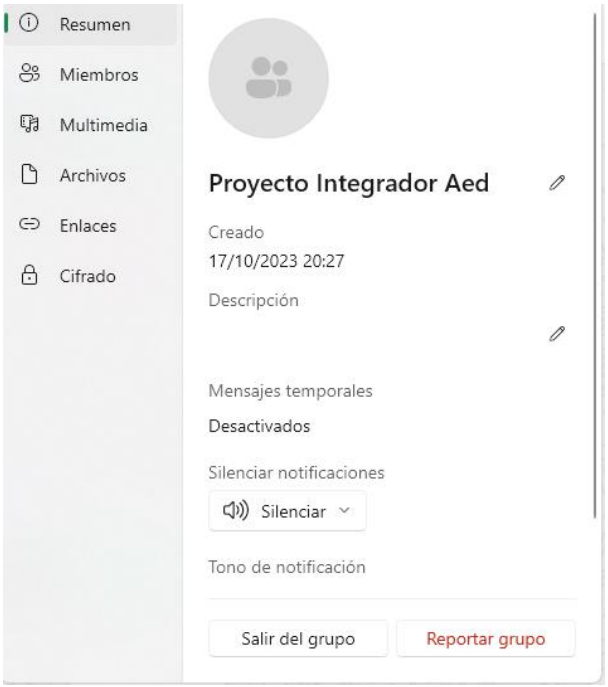
El archivo principal, **"Proyecto Snake.cpp**", integra la lógica del juego de la serpiente, manejo de usuarios y puntuaciones, menús interactivos y funciones para la interacción con archivos. Este conjunto crea una experiencia de juego completa dentro de la consola.

Desarrollo de las etapas del proyecto:

Etapa 1: Desarrollo de Idea

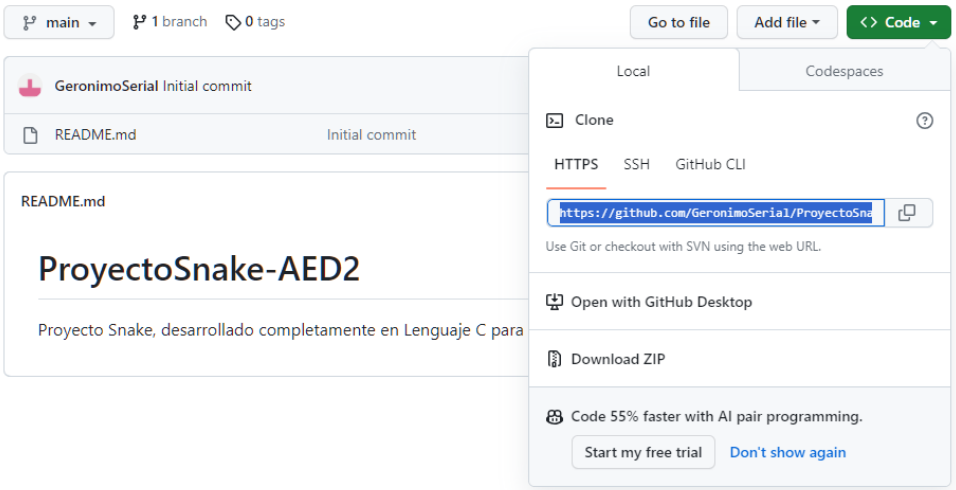
Primera parte

Creación de un grupo de WhatsApp como espacio de trabajo:



Se integraron los participantes del proyecto.

Creamos un repositorio de trabajo en GitHub para hacer la colaboración e implementación de las ideas acordadas:



Segunda parte

Creación del bosquejo para la creación del proyecto.

Se creó una estructura, en pseudocodigo, que usaríamos como guía para desarrollar el proyecto.

```
Inicio

Definir Estructura Usuario:
    Nombre
    Puntuación

Definir Función inicializarListaEstatica():
    Para cada usuario en listaEstatica:
        Establecer nombre como "vacio"
        Establecer puntuación como 0

Definir Función inicializarListaDinamica():
    Establecer puntuacionNivelIntermedio como Nulo

Definir Función abrirArchivoEscribir():
    Abrir archivo para escritura

Definir Función grabarDatos(datosUsuario):
    Guardar datosUsuario en archivo

Definir Función insertarElemento(usuario):
    Si la lista dinámica está vacia:
        Crear nuevo nodo con el usuario
        Establecer como primer elemento de la lista dinámica
    Sino:
        Crear nuevo nodo con el usuario
        Insertar nodo al inicio de la lista

Definir Función eliminarUsuarioNivelIntermedio():
    Mostrar menú para eliminar usuario:
    Si se elige eliminar:
        Si se elige eliminar el primero:
            Eliminar primer usuario de la lista dinámica
            Visualizar puntuaciones actualizadas
        Sino:
            Ingresar el número del usuario a eliminar
            Eliminar usuario especifico
            Visualizar puntuaciones actualizadas

Definir Función guardarPosicion():
    Guardar posición de la serpiente

Definir Función dibujarCuerpo():
    Para cada segmento de la serpiente:
        Dibujar segmento en pantalla

Definir Función borrarCuerpo():
    Borrar último segmento de la serpiente

Definir Función teclear():
    Si se presiona una tecla:
        Capturar tecla presionada

Definir Función cambiarVelocidad():
    Si la puntuación alcanza múltiplos de 20:
        Incrementar velocidad

Definir Función puntos():
    Mostrar puntuación actual en pantalla

Definir Función juego():
    Mientras no se presione tecla ESC y el juego no haya terminado:
        Actualizar puntuación
        Borrar segmento de la serpiente
        Guardar posición
        Dibujar serpiente
        Generar comida
        Capturar teclas y mover serpiente
        Si pierde o gana:
            Mostrar mensaje
            Actualizar puntuación de usuario
            Guardar datosUsuario

Definir Función menu():
    Mostrar menú principal:
    Si se elige una opción:
        Realizar la acción correspondiente
        Volver al menú

Fin
```

Etapa 2: Implementación - Estructuras compuestas: listas

1. *Primera parte*

Decidimos realizar una lista estática, para almacenar y gestionar los usuarios. Para ello, definimos los tipos de datos y los prototipos.

```c
//TIPOS DE DATOS

typedef char tString[20];

typedef struct {
    tString usuario;
    int puntuacionUsuario;
}tUsuario;

typedef tUsuario listaEstatica[USUARIOS];


//DECLARACION DE LISTA Y AUXILIAR
listaEstatica puntuacionesNivelInicial;
tUsuario auxUsuario;


//PROTOTIPADO
void inicializarListaEstatica(listaEstatica);
void ordenarListaEstatica(listaEstatica);
void mostrarListaEstatica(listaEstatica);
void mostrarListaEstaticaRecur(listaEstatica , int);

void eliminarUsuarioNivelInicial();
void eliminarUsuario(int);
```

2. *Segunda parte*

Para las puntuaciones, utilizamos lista dinámica Estructuras de datos: Define la estructura tUser para almacenar el nombre de usuario y su puntuación, y una estructura de nodo tListaDinamica que contiene un usuario y un puntero al siguiente nodo.

- Funciones de lista: Ofrece operaciones básicas para manipular la lista dinámica:
- Inicialización de la lista.
- Verificación de si la lista está vacía.
- Inserción de usuarios en la lista (al principio, adelante o en una posición específica).
- Eliminación de usuarios (el primero, en una posición específica o un usuario en particular).

```c
#define USUARIOS 10

typedef char tString[20];

typedef struct {
    tString nombre;
    int puntuacionUsuario;
}tUser;

typedef struct nodo {
    tUser user;
    struct nodo * siguiente;
} tListaDinamica;

void inicializarLista();
bool listaVacia( tListaDinamica * );

void insertarPrimero( tUser );
void insertarAdelante( tUser );
void insertarElemento( tUser );

void eliminarPrimero();
void eliminarK( int k );
void eliminarUsuario();
```

PROYECTO SNAKE
COMISION C

GERMAN SAUCEDO
GERONIMO SERIAL
MATIAS JOSE TORRES
NAHUEL ALEJO TOMEI

3. *tercera parte - Corte*

Definición de estructuras: Define la estructura tDatosUsuario que contiene información sobre el nivel, el nombre del usuario y su puntuación. También define un tipo listaOrdenar que representa una lista estática de datos de usuarios.

Funciones de manejo de archivos: Proporciona funciones para abrir, escribir y cerrar archivos donde se almacenan los datos de los usuarios.

Funciones de procesamiento de datos: Ofrece funciones para llevar a cabo procesos específicos:

procesoCorte(): Realiza un proceso específico de corte en los datos de usuarios.

ordenarListaEstatica(): Ordena la lista estática de usuarios.

```c
#define USUARIOSTOTALES 20

typedef char tString[20];

typedef struct {
    int nivel;
    tString usuario;
    int puntuacionUsuario;
}tDatosUsuario;

typedef tDatosUsuario listaOrdenar[USUARIOSTOTALES];

void abrirArchivoEscribir();
void grabarDatos( tDatosUsuario );
void cerrarArchivoEscribir();

void abrirArchivoLeer();
void cerrarArchivoLeer();
void procesoCorte();
void principioCorte();
void unUsuario();
void finCorte();
void leerRegistro();

int buscarLugarArray(listaOrdenar);
void ordenarListaEstatica(listaOrdenar);

```


GERMAN SAUCEDO
GERONIMO SERIAL
MATIAS JOSE TORRES
NAHUEL ALEJO TOMEI

Etapa 3:

Unificación de archivos e implementación del Proyecto Snake.cpp

Configuración y Datos del juego:

1- Al principio, se incluyen los archivos creados anteriormente, como los necesarios para la implementación. Se definen las reglas y variables del juego, como las teclas para mover la serpiente y los datos de puntuación y nivel.

```cpp
#include<iostream>
#include<windows.h>
#include<conio.h>

#include "ListaEstatica.h"
#include "ListaDinamica.h"
#include "Corte.h"

#define ARRIBA    72
#define IZQUIERDA 75
#define DERECHA   77
#define ABAJO     80
#define ESC       27

#define USUARIOS 10

void inicializarPrograma();

int cuerpo [400][2];
int n;
int tam;
int x,y;
int dir;
int xc,yc;
int velocidad,h;
int puntuacion;
int w1;
int w2;
```


GERMAN SAUCEDO
GERONIMO SERIAL
MATIAS JOSE TORRES
NAHUEL ALEJO TOMEI

Funciones Esenciales.

Definimos las funciones principales que controlan la serpiente: Como se dibuja, como se come la comida, que pasa si choca y como se gestiona la puntuación. Se tienen dos niveles distintos con sus propias reglas y dificultades, cada uno con su propio diseño y condiciones de juego

```
1  void gotoxy (int, int);
2  void guardar_posicion();
3  void dibujar_cuerpo();
4  void borrar_cuerpo();
5  void teclear();
6  void cambiar_velocidad();
7  void puntos();
8  void titulo();
9  void menu();
10 void opciones();
11
12 //Nivel 1
13
14 void pintar ();
15 void comida();
16 bool game_over();
17 void game();
18 bool win1();
19
20 //Nivel 2
21 void pintar2();
22 void comida2();
23 bool game_over2();
24 void game2();
25 bool win2();
```

Opciones:

Se define un menú principal con diferentes opciones para jugar en distintos niveles, visualizar tablas de puntuación y generar informes. Para lograr una estética acorde a lo esperado, y debido a que no llegamos a un acuerdo, para de manera propia generar un menú adecuado, solicitamos a una IA que nos genere un formato de impresión para el menú de opciones según lo que habíamos desarrollado.

```c
void opciones(){
    system("cls");
    printf("\n\n\n");
    printf("\t\t\t\t  *=============================================*\n");
    printf("\t\t\t\t  |                                             |\n");
    printf("\t\t\t\t  | Ingrese la opcion que desea realizar        |\n");
    printf("\t\t\t\t  |                                             |\n");
    printf("\t\t\t\t  | 1 - Iniciar partida nivel inicial           |\n");
    printf("\t\t\t\t  |                                             |\n");
    printf("\t\t\t\t  | 2 - Iniciar partida nivel intermedio        |\n");
    printf("\t\t\t\t  |                                             |\n");
    printf("\t\t\t\t  | 3 - Mostrar tablero de puntuacion nivel inicial    |\n");
    printf("\t\t\t\t  |                                             |\n");
    printf("\t\t\t\t  | 4 - Mostrar tablero de puntuacion nivel intermedio |\n");
    printf("\t\t\t\t  |                                             |\n");
    printf("\t\t\t\t  | 5 - Emitir informe por nivel                |\n");
    printf("\t\t\t\t  |                                             |\n");
    printf("\t\t\t\t  | Otro numero - Salir                         |\n");
    printf("\t\t\t\t  |                                             |\n");
    printf("\t\t\t\t  *=============================================*");
    gotoxy(1,18);printf("\n\n\t\t\t\t   Eleccion: ");
}
```

Funciones game() y game2()

La función **game()** esta diseñada para controlar la lógica del juego en el nivel incial de Snake. La función opera con un bucle while que continua mientras el jugador no presione la tecla ESC. Y mientras el juego no cumpla las condiciones de la función **game_over()** o el jugador no haya ganado y cumpla las condiciones de la función **win().**

Se llama también, a las funciones de actualizacion y dibujado del juego. **borrar_cuerpo()** y **dibujar_cuerpo()** actualizan el cuerpo de la serpiente en pantalla y **puntos()** muestra la puntuación actual.

La serpiente se mueve en función de la tecla presionada por el usuario, definido esto en la función **teclear()**. Las coordenadas x e y se actualizan según la dirección elegida, **('dir')**

Si la función **game_over()** o la función **win()** se cumplen, se muestra un mensaje usando la función **messageBox()** que imprime un cartel externo, y se graba la puntuación del usuario.

```
void game(){
    while(tecla!=ESC&&game_over()&& win()){
    puntos();
    borrar_cuerpo();
    guardar_posicion();
    dibujar_cuerpo();
    comida();

    teclear();
    teclear();

    if(dir==1)y--;
    if(dir==2)y++;
    if(dir==3)x++;
    if(dir==4)x--;
    Sleep(velocidad);

if ( !game_over()){
    MessageBox(NULL,"PERDISTE","  JUEGO TERMINADO", MB_OK);
    system("cls");
    puntuacionesNivelInicial[lugarLibre].puntuacionUsuario = puntuacion;
    datosUsuario.puntuacionUsuario = puntuacion;
    grabarDatos(datosUsuario);
}
if ( !win()){
    MessageBox(NULL,"GANASTE"," JUEGO TERMINADO", MB_OK);
    system("cls");
    puntuacionesNivelInicial[lugarLibre].puntuacionUsuario = puntuacion;
    datosUsuario.puntuacionUsuario = puntuacion;
    grabarDatos(datosUsuario);
}
}
}
```

La funcion **game2()**, es una extension logica del juego, pero que llama a la función **comida2()** para ofrecer un desafio mas avanzado que el nivel inicial.

comida2() similar a **comida()** genera comida para la serpiente en un nivel mas avanzado. A diferencia de **comida()** llama a la función **cambiar_velocidad()** cada vez que la serpiente come la comida.

```c
void comida2(){
    if (x == xc && y == yc) {
        xc = (rand() % 73) + 4;
        yc = (rand() % 19) + 4;
        tam++;
        puntuacion += 10;
        cambiar_velocidad;
    }
    gotoxy(xc, yc);
    printf("%c", 'O');
}
```

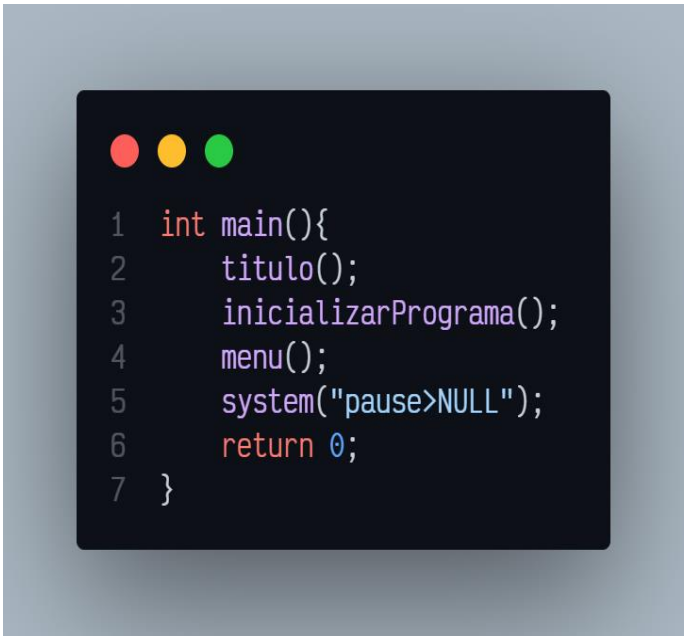

GERMAN SAUCEDO
GERONIMO SERIAL
MATIAS JOSE TORRES
NAHUEL ALEJO TOMEI

Parte de la gestión de la puntuación, también se realiza en **game() y game2()**, se llama a la función **puntos()**, que utiliza la información desarrollada en el archivo "**Corte.h"**, y que con la funcion **gotoxy()** envia a un sector de la consola la información sobre los puntos adquiridos.

```
void puntos(){
    gotoxy(3,1);printf("Puntos 00%d",puntuacion);
}
```

Nuestro int main()

Se define un menú principal **int main()** donde imprimirá el titulo, llamaría a la funcion inicializar programa que contiene a su vez, subfunciones para llamar a las listas creadas anteriormente, y se muestra en pantalla el menú de opciones.



```
int main(){
    titulo();
    inicializarPrograma();
    menu();
    system("pause>NULL");
    return 0;
}
```

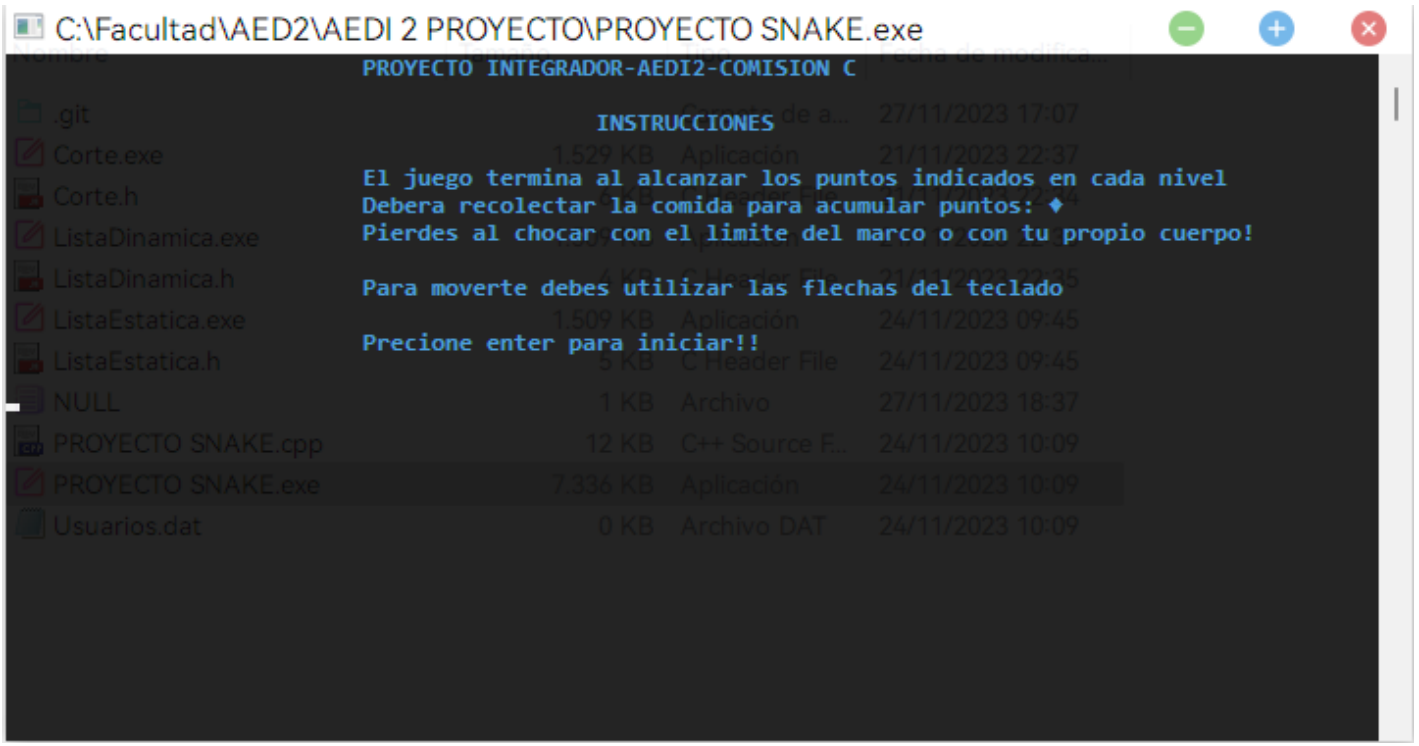

GERMAN SAUCEDO
GERONIMO SERIAL
MATIAS JOSE TORRES
NAHUEL ALEJO TOMEI

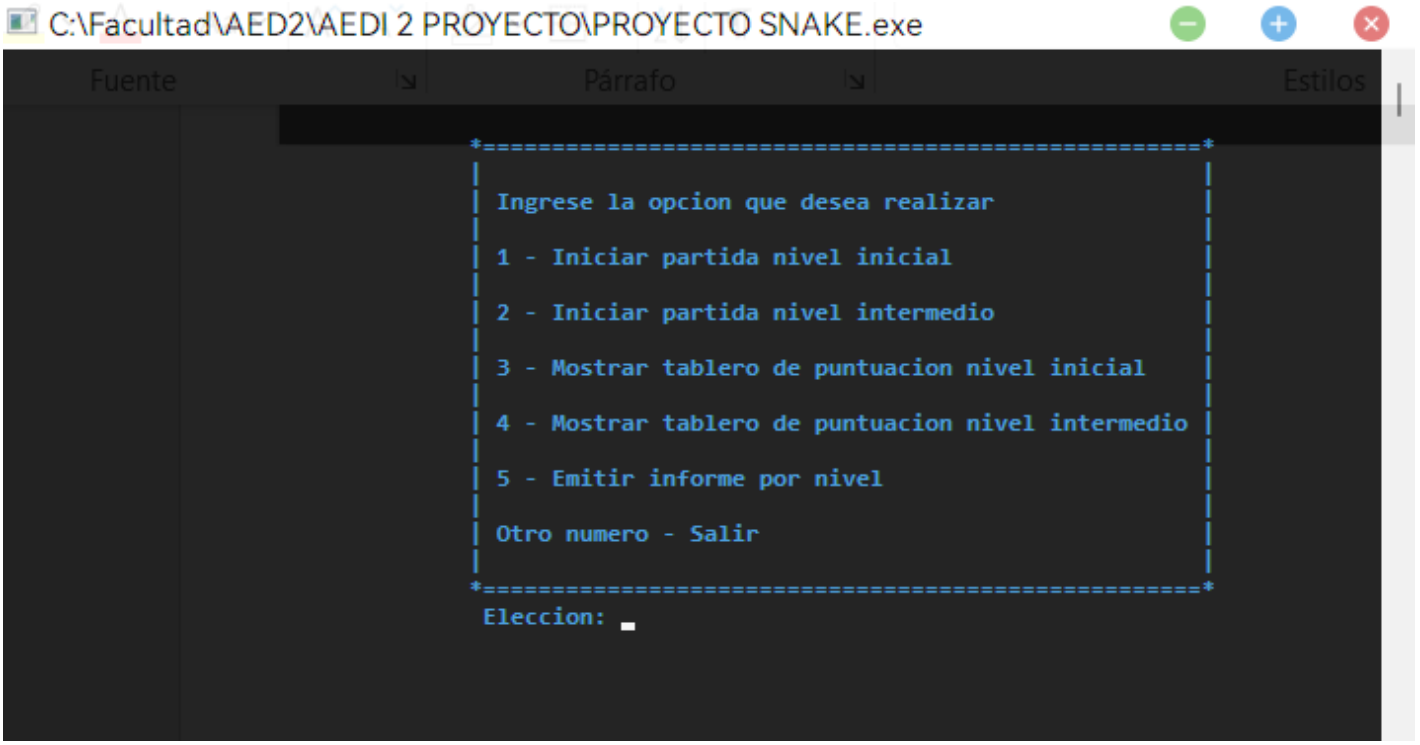
Conclusiones:

Comenzar este proyecto fue desafiante y, al principio, un tanto desalentador, debido a que no sabíamos por dónde comenzar ni creíamos que lograríamos algo acorde a las expectativas que teníamos. pero a medida que avanzaba, se convirtió en un viaje educativo. Ahora, al mirar el proyecto finalizado, podemos decir que a pesar de los obstáculos iniciales, ha sido satisfactorio completar este juego y ha resultado en una experiencia enriquecedor

GERMAN SAUCEDO
GERONIMO SERIAL
MATIAS JOSE TORRES
NAHUEL ALEJO TOMEI

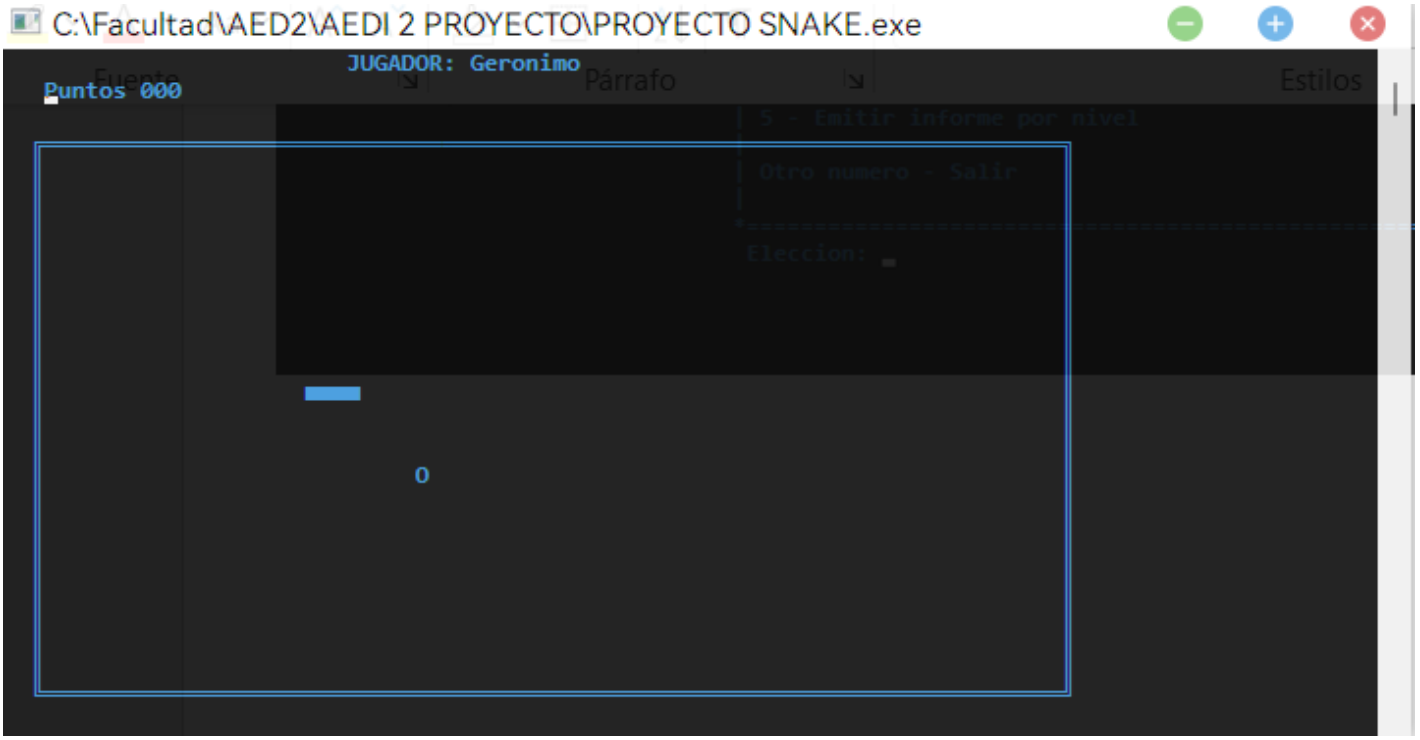
*Ejecución del juego "**Proyecto Snake**"*

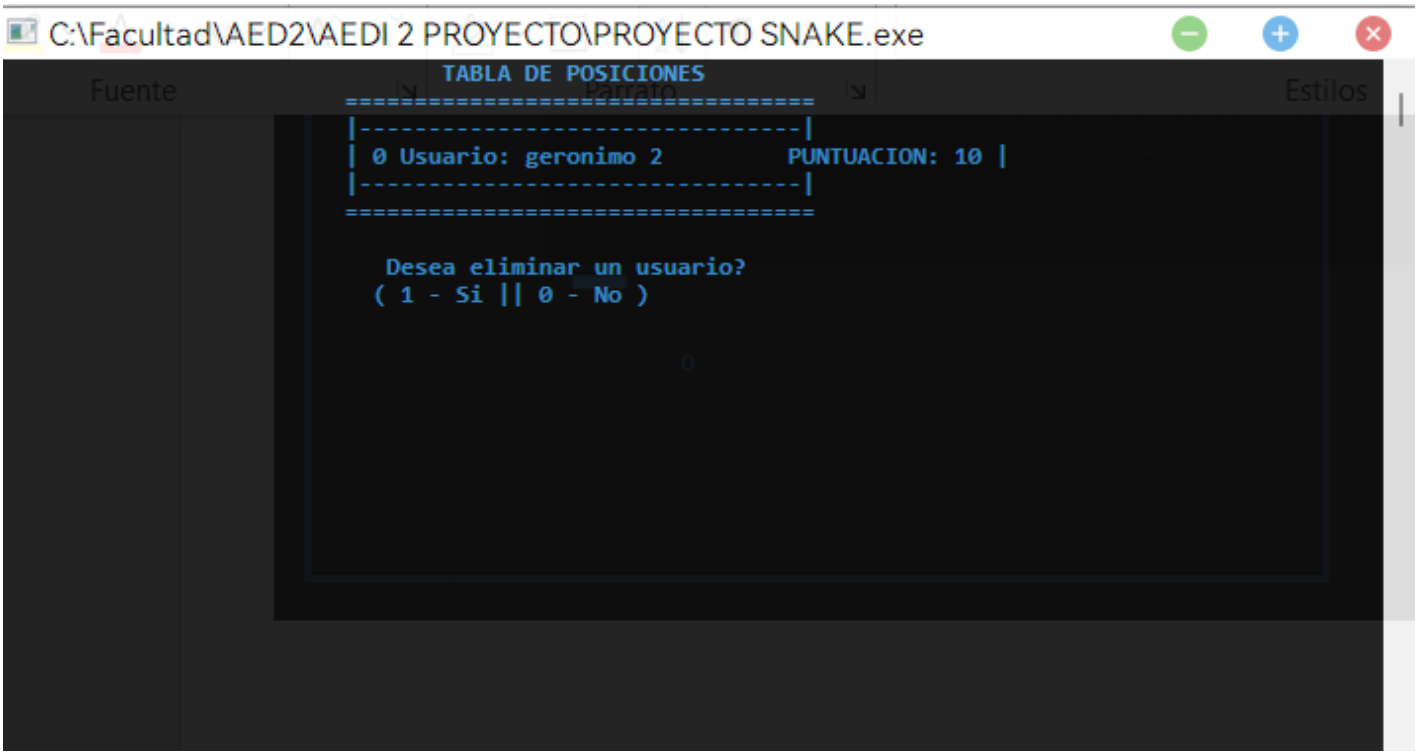




PROYECTO SNAKE
COMISION C

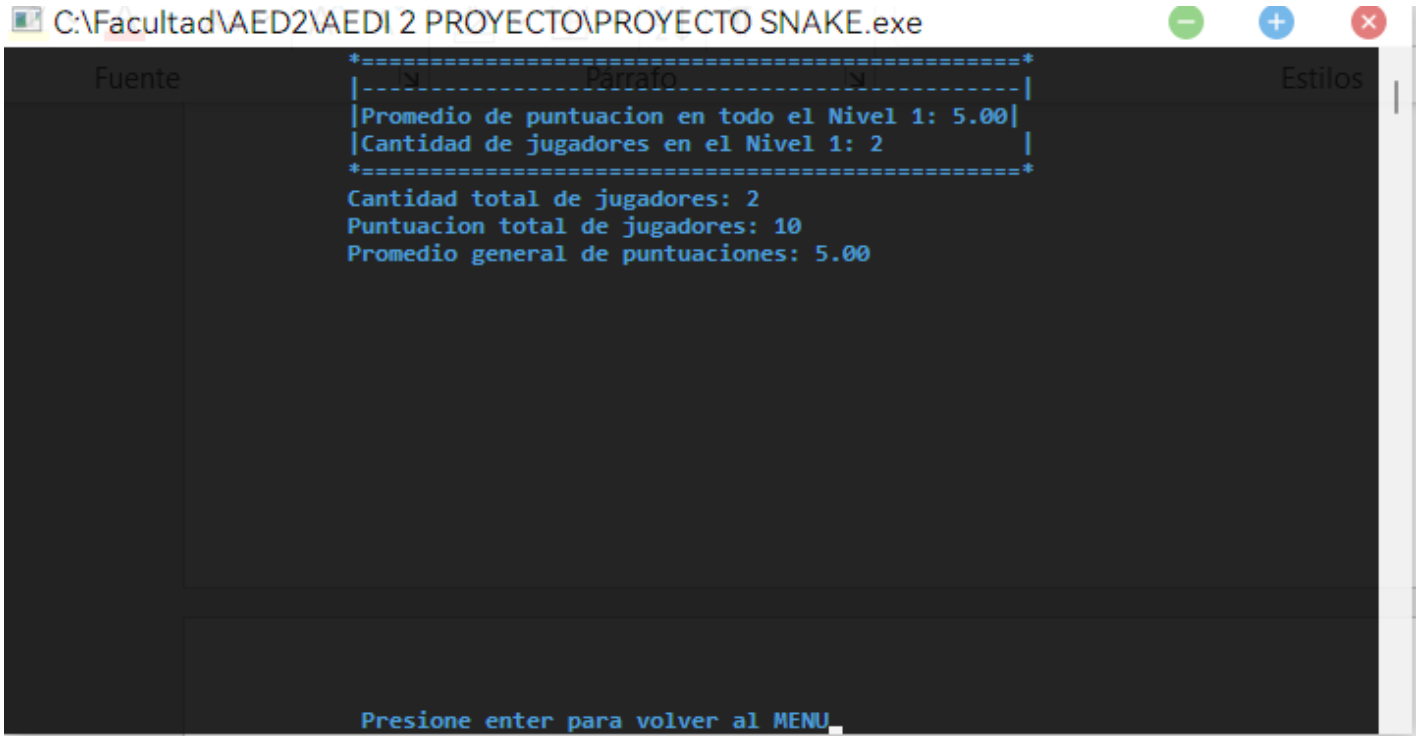
GERMAN SAUCEDO
GERONIMO SERIAL
MATIAS JOSE TORRES
NAHUEL ALEJO TOMEI





PROYECTO SNAKE
COMISION C

GERMAN SAUCEDO
GERONIMO SERIAL
MATIAS JOSE TORRES
NAHUEL ALEJO TOMEI



PROYECTO SNAKE
COMISION C

GERMAN SAUCEDO
GERONIMO SERIAL
MATIAS JOSE TORRES
NAHUEL ALEJO TOMEI