|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería de Software | **GUÍA**  No. 02 | **TIEMPO ESTIMADO:**  1 mes |
| **ASIGNATURA:**  Estructura de datos  NRC: 2967 | **FECHA DE ELABORACION:** 10-12-2019  **SEMESTRE**: septiembre 2019 – febrero 2020 | |
| **TÍTULO:**  Lista | **DOCENTE:** Ing. Fernando Solís | |

**OBJETIVO**

Utilizar Listas (simples, dobles, etc.) para realizar un proyecto el cual se asemeje al clásico juego de Tetris.

**INSTRUCCIONES**

**Listas**

La lista enlazada es un TDA que nos permite almacenar datos de una forma organizada, al igual que los vectores pero, a diferencia de estos, esta estructura es dinámica, por lo que no tenemos que saber "a priori" los elementos que puede contener.

En una lista enlazada, cada elemento apunta al siguiente excepto el último que no tiene sucesor y el valor del enlace es null. Por ello los elementos son registros que contienen el dato a almacenar y un enlace al siguiente elemento. Los elementos de una lista, suelen recibir también el nombre de nodos de la lista.

Los operadores básicos de una son:

* Insertar: inserta un nodo con dato x en la lista, pudiendo realizarse esta inserción al principio o final de la lista o bien en orden.
* Eliminar: elimina un nodo de la lista, puede ser según la posición o por el dato.
* Buscar: busca un elemento en la lista.
* Localizar: obtiene la posición del nodo en la lista.
* Vaciar: borra todos los elementos de la lista

**ACTIVIDADES**

1. **Ubicación de recursos**
2. El grupo de 2 personas
3. Herramienta para C++, llamada Code::Blocks
4. **Planteamiento del problema**

Elaborar un programa el cual se asemeje a un Tetris clásico utilizando los diferentes tipos de listas asi como las operaciones básicas de las mismas.

1. **Entregable (s)**

**Clase Nodo.h**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Module: Nodo.h

\* Author: Luis Carvajal, Elian Llorente

\* Modified: sabado, 30 de noviembre de 2019 10:04:26

\* Purpose: Declaration of the class Nodo

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class Nodo

{

private:

int numero;

Nodo\* anteriorDireccion;

Nodo\* siguienteDireccion;

public:

void setNumero(int);

int getNumero();

void setAnteriorDireccion(Nodo\*);

Nodo\* getAnteriorDireccion();

void setSiguienteDireccion(Nodo\*);

Nodo\* getSiguienteDireccion();

};

void Nodo::setNumero(int numero\_)

{

numero=numero\_;

}

int Nodo::getNumero()

{

return numero;

}

void Nodo::setAnteriorDireccion(Nodo\* anteriorDireccion\_)

{

anteriorDireccion=anteriorDireccion\_;

}

Nodo\* Nodo::getAnteriorDireccion()

{

return anteriorDireccion;

}

void Nodo::setSiguienteDireccion(Nodo\* siguienteDireccion\_)

{

siguienteDireccion=siguienteDireccion\_;

}

Nodo\* Nodo::getSiguienteDireccion()

{

return siguienteDireccion;

}

**Clase ListaDoble.h**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Module: ListaDoble.h

\* Author: Luis Carvajal, Elian Llorente

\* Modified: sabado, 30 de noviembre de 2019 10:04:26

\* Purpose: Declaration of the class ListDoble

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "Nodo.h"

#include <iostream>

#include <cstdio>

using namespace std;

class ListaDoble

{

private:

Nodo \*lista;

string jugador;

int contadorNodo;

int puntaje;

public:

void insertarAlFinal(int);

void impresion();

void impresionLista();

void generarLista();

void listaTetris(int,int);

void juegoTetris();

int posicionAIndice(int x);

void deleteNumber(int,int );

void siguienteNumero(int,Nodo \* );

void deleteNodo(Nodo \*Actual);

ListaDoble()

{

lista=NULL;

contadorNodo=0;

puntaje=0;

}

};

/\*\*

\* @brief Funcion que imprime el resultado del Juego

\*/

void ListaDoble::impresion()

{

Nodo\* Aux=new Nodo();

Aux=lista;

if(Aux == NULL){

gotoxy(20,10);cout<<"\n\n\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||";

cout<<"\n|||| ||||\n";

cout<<"\n|||| NO HAS INICIADO EL JUEGO ||||";

cout<<"\n|||| ||||";

cout<<"\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\n\n\n";

}

else

{

fstream enter;

enter.open("tetrisPixel.txt",fstream::out); //para leer in, para salir es out escribir

gotoxy(20,15);

cout<<"Jugador: "<<jugador;

gotoxy(20,17);

cout<<"LA LISTA EN LA QUE PERDISTE ES: ";

gotoxy(20,19);

while(Aux != NULL)

{

printf("%d, ",Aux->getNumero());

Aux=Aux->getSiguienteDireccion();

}

gotoxy(20,21);printf("Su Puntaje es: %d \n\n",puntaje);

enter<<"Jugador: "<<jugador<<". Tu Puntaje es: "<<puntaje<<endl;

}

}

/\*\*

\* @brief Funcion que inserta al final de la lista la pieza en el Juego

\* @param numero que se agrega al final de la lista

\*/

void ListaDoble::insertarAlFinal(int numero)

{

Nodo\* Nuevo=new Nodo();

Nodo\* Actual=new Nodo();

if(contadorNodo==0)

{

Actual->setNumero(numero);

Actual->setAnteriorDireccion(NULL);

Actual->setSiguienteDireccion(NULL);

lista=Actual;

}

else

{

Actual=lista;

while(Actual->getSiguienteDireccion()!=NULL)

{

Actual=Actual->getSiguienteDireccion();

}

Nuevo-> setNumero(numero);

Nuevo-> setSiguienteDireccion(NULL);

Actual->setSiguienteDireccion(Nuevo);

Nuevo-> setAnteriorDireccion(Actual);

}

contadorNodo++;

}

/\*\*

\* @brief Funcion que genera la lista randomica en el fondo del tablero del Juego

\*/

void ListaDoble::generarLista()

{

Nodo\* Aux=new Nodo();

int numero;

for(int i=0;i<17;i++)

{

numero=rand()%10;

insertarAlFinal(numero);

}

}

/\*\*

\* @brief Funcion inserta los elementos en el tablero

\* @param posicion, lugar en el que se encuentra

\* @param numero, el elemento que se insertara en esa posicion

\*/

void ListaDoble::listaTetris(int posicion,int numero)

{

Nodo \*Aux=new Nodo();

Nodo \*Actual=new Nodo();

Nodo \*Nuevo=lista;

int cont;

while(Nuevo!=NULL)

{

if(numero==Nuevo->getNumero())

{

cont++;

break;

}

Nuevo=Nuevo->getSiguienteDireccion();

}

switch(posicion)

{

case 1:

if(cont>0)

{

Nuevo=Nuevo->getSiguienteDireccion();

Nuevo->setAnteriorDireccion(NULL);

lista=Nuevo;

contadorNodo--;

}

else

{

Aux->setNumero(numero);

Aux->setSiguienteDireccion(lista);

Aux->setAnteriorDireccion(NULL);

Actual->setAnteriorDireccion(Aux);

lista=Aux;

contadorNodo++;

}

break;

}

}

int ListaDoble::posicionAIndice(int x){

return (x-31)/4;

}

/\*\*

\* @brief Funcion que elimina el nodo Actual para apuntar al siguiente

\* @param \*Actual nodo

\*/

void ListaDoble::deleteNodo(Nodo \*Actual){

Nodo\* Siguiente=new Nodo();

Nodo\* Anterior=new Nodo();

if(Actual->getAnteriorDireccion()!=NULL&&Actual->getSiguienteDireccion()!=NULL){

Anterior=Actual->getAnteriorDireccion();

Siguiente=Actual->getSiguienteDireccion();

Anterior->setSiguienteDireccion(Siguiente);

Siguiente->setAnteriorDireccion(Anterior);

delete(Actual);

}else if(Actual->getAnteriorDireccion()==NULL){

lista=Actual->getSiguienteDireccion();

delete(Actual);

}else if(Actual->getSiguienteDireccion()==NULL){

Anterior=Actual->getAnteriorDireccion();

Anterior->setSiguienteDireccion(NULL);

delete(Actual);

}

}

/\*\*

\* @brief Funcion que hace que el anterior apunte al siguiente del que se eliminira

\* @param numero que coincide con otro elemento de la lista

\* @param \*aux guarda el siguiente y mueve la lista

\*/

void ListaDoble::siguienteNumero(int numero,Nodo \*aux){

if(aux==NULL||(aux->getNumero()!=(numero+1))){

return ;

}else{

if(aux->getNumero()==(numero+1)){

puntaje++;

deleteNodo(aux);

}

}

}

/\*\*

\* @brief Funcion que elimina cuando ambos elementos coninciden

\* @param indice posicion en la que se encuentra

\* @param numero numero en lista

\*/

void ListaDoble::deleteNumber(int indice,int numero){

int cont=0;

Nodo\* Actual=new Nodo();

Actual=lista;

while(Actual!=NULL){

if(indice==cont){

if(Actual->getNumero()==numero)//borra el numero

{

contadorNodo--;

puntaje++;

deleteNodo(Actual);

siguienteNumero(numero,Actual);

}

else

{

insertarAlFinal(numero);

}

Actual=NULL;

}

else

{

Actual=Actual->getSiguienteDireccion();

}

cont++;

}

}

/\*\*

\* @brief Funcion general del Juego

\*/

void ListaDoble::juegoTetris()

{

bool primera=true;

char tecla=0;

int x=31,y=4;

int col,fil,j=0;

int numero;

int i=0;

srand(time(NULL));

lista=NULL;

contadorNodo=puntaje=0;

gotoxy(20,15);

printf("NOMBRE DEL JUGADOR: ");

cin>>jugador;

fflush(stdin);

generarLista();

do

{

Sleep(200);

system("cls");

margen();

if(primera)

{

numero=0+rand()%9;

primera=false;

gotoxy(4,45);

}

if((x==10)&&(y==23)) /// aqui mandar cuando encuentre el numero

{

listaTetris(1,numero);

numero=0+rand()%9;

x=34;y=3;

}

impresionLista();

gotoxy(x,y);

numbers(numero,x,y);

color(15);

y++;

if(y>=37){

deleteNumber(posicionAIndice(x),numero);

y=4;

primera=true;

}

if(kbhit()){

switch(getch())

{

case TECLA\_DERECHA:

x+=4;

if(x>95)

x=31;

break;

case TECLA\_IZQUIERDA:

x-=4;

if(x<31)

x=95;

break;

case TECLA\_ENTER:

tecla = TECLA\_ENTER;

break;

}

}

if(contadorNodo==18)

tecla=TECLA\_ENTER;

}while(tecla!=TECLA\_ENTER);

system("cls");

}

/\*\*

\* @brief Funcion que imprime las piezas en el tablero

\*/

void ListaDoble::impresionLista()

{

int x=31,y;

Nodo\* Aux=new Nodo();

Aux=lista;

while(Aux != NULL)

{

y=38;

switch(Aux->getNumero())

{

case 0:

color(1);//num 0 azul

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c0%c%c",219,219,219);gotoxy(x,y++);

color(15);//num 0 azul

break;

case 1:

color(2);// 1 verde

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c1%c%c",219,219,219);gotoxy(x,y++);

break;

case 2:

color(11);// 2 celeste

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c2%c%c",219,219,219);gotoxy(x,y++);

break;

case 3:

color(12);// 3 rojo

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c3%c%c",219,219,219);gotoxy(x,y++);

break;

case 4:

color(5);// 4 morado

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c4%c%c",219,219,219);gotoxy(x,y++);

break;

case 5:

color(6);// 5 amarillo

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c5%c%c",219,219,219);gotoxy(x,y++);

break;

case 6:

color(15);// 6 blanco

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c6%c%c",219,219,219);gotoxy(x,y++);

break;

case 7:

color(8);// 7 gris

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c7%c%c",219,219,219);gotoxy(x,y++);

break;

case 8:

color(13);// 8 mas celeste

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c8%c%c",219,219,219);gotoxy(x,y++);

break;

case 9:

color(10);// 9 verde claro

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c9%c%c",219,219,219);gotoxy(x,y++);

break;

}

x+=4;//separacion entre cubos

Aux=Aux->getSiguienteDireccion();

}

}

**Clase** **PersonalLibrary.h**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Module: PersonalLibrary.h

\* Author: Luis Carvajal, Elian Llorente

\* Modified: sabado, 30 de noviembre de 2019 10:04:26

\* Purpose: Declaration of the class PersonalLibrary

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <conio.h>

#include <windows.h>

#include <ctype.h>

#include <string.h>

#include <fstream>

#include <string>

#include <time.h>

#include <pthread.h>

#include <fstream>

using namespace std;

#define TECLA\_ARRIBA 72

#define TECLA\_ABAJO 80

#define TECLA\_DERECHA 77

#define TECLA\_IZQUIERDA 75

#define TECLA\_ENTER 13

void gotoxy(int x, int y)

{

HANDLE hCon;

hCon=GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD dwPos;

dwPos.X=x;

dwPos.Y=y;

SetConsoleCursorPosition(hCon,dwPos);

}

void color(int x)

{

SetConsoleTextAttribute(GetStdHandle (STD\_OUTPUT\_HANDLE),x);

}

void AltEnter()

{

keybd\_event(VK\_MENU,

0x38,

0,

0);

keybd\_event(VK\_RETURN,

0x1c,

0,

0);

keybd\_event(VK\_RETURN,

0x1c,

KEYEVENTF\_KEYUP,

0);

keybd\_event(VK\_MENU,

0x38,

KEYEVENTF\_KEYUP,

0);

}

/\*\*

\* @brief Funcion que genera los cubos en el tablero del juego

\* @param numero cubo de un color especifico para cada numero del 1-9

\* @param x coordenada

\* @param y coordenada

\*/

void numbers(int numero,int x,int y)

{

switch(numero){

case 0:

color(1);//num 0 azul

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c0%c%c",219,219,219);gotoxy(x,y++);

color(15);//num 0 azul

break;

case 1:

color(2);// 1 verde

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c1%c%c",219,219,219);gotoxy(x,y++);

break;

case 2:

color(11);// 2 celeste

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c2%c%c",219,219,219);gotoxy(x,y++);

break;

case 3:

color(12);// 3 rojo

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c3%c%c",219,219,219);gotoxy(x,y++);

break;

case 4:

color(5);// 4 morado

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c4%c%c",219,219,219);gotoxy(x,y++);

break;

case 5:

color(6);// 5 amarillo

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c5%c%c",219,219,219);gotoxy(x,y++);

break;

case 6:

color(15);// 6 blanco

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c6%c%c",219,219,219);gotoxy(x,y++);

break;

case 7:

color(8);// 7 gris

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c7%c%c",219,219,219);gotoxy(x,y++);

break;

case 8:

color(13);// 8 mas celeste

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c8%c%c",219,219,219);gotoxy(x,y++);

break;

case 9:

color(10);// 9 verde claro

gotoxy(x,y++);

printf("%c%c%c%c",219,219,219,219);gotoxy(x,y++);

printf("%c9%c%c",219,219,219);gotoxy(x,y++);

break;

}

}

/\*\*

\* @brief Funcion que genera el margen del Juego

\*/

void margen()

{

color(15);

for(int i=30; i < 99; i++)

{

//PARTE SUPERIOR

gotoxy (i, 3); printf("%c",177);

//PARTE INFERIOR

gotoxy(i, 40); printf("%c",177);

}

for(int j=3; j < 40; j++)

{

//PARTE IZQUIERDA

gotoxy (30,j); printf("%c",177);

//PARTE DERECHA

gotoxy(99,j); printf("%c",177);

}

//ESQUINAS

color(15);

gotoxy(99,40); printf("%c",177);

}

**Clase Main.cpp**

#include <iostream>

#include <windows.h>

#include <conio.h>

#include "PersonalLibrary.h"

#include "ListaDoble.h"

#define ARRIBA 72

#define IZQUIERDA 75

#define DERECHA 77

#define ABAJO 80

using namespace std;

ListaDoble ObjJuego;

int i=21;

/\*\*

\* @brief Funcion que me permite selccionar una opcion del Menu

\* @param tecla que se mueve dentro del menu

\*/

void seleccionarOpcion(char tecla){

gotoxy(30, i); cout<<" ";

if( tecla == ABAJO && i <=24 ){

if(i==24){

i=20;

}

i++;

}

if( tecla == ARRIBA && i >= 21){

if(i==21){

i=25;

}

i--;

}

if( tecla == 13 )

{

if( i == 21 ){

system("cls");

ObjJuego.juegoTetris(); // metodo que comienza el juego

}

if( i == 22 ){

system("cls");

cout<<"CERRAR EL ARCHIVO AYUDA PARA SEGUIR EN EL MENU...."<<endl;

system("ayuda.chm");

//system("ayuda.pdf");

system("pause");

system("cls");

}

if( i == 23 ){

system("cls");

ObjJuego.impresion();

cout<<"\n\n\n\nABRIENDO ARCHIVO CON LAS SOLUCIONES"<<endl;

system("txt2pdf.exe tetrisPixel.txt respuestas.pdf -oao -pfs60 -pps43 -ptc0 -width3000 -height2000");

system("pause");

system("cls");

}

if( i == 24 ){

system("cls");

cout<<"\n\n\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||";

cout<<"\n|||| ||||\n";

cout<< "|||| ||||";

cout<<"\n|||| GRACIAS POR USAR ESTE PROGRAMA ||||";

cout<<"\n|||| ESPERAMOS QUE HAYA SIDO DE TU AGRADO ||||";

cout<<"\n|||| ||||";

cout<<"\n|||| ||||";

cout<<"\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\n\n\n";

system("pause");

exit(0);

}

}

gotoxy(30, i); cout<<">";

}

/\*\*

\* @brief Funcion que muestra el Menu principal

\* @param tecla que se mueve dentro del menu

\*/

void menuinicio(char tecla){

gotoxy(20, 4);cout<<"\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||";

cout<<"\n|||| ||||\n";

cout<< "|||| UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE ||||";

cout<<"\n|||| ESTRUCTURA DE DATOS ||||";

cout<<"\n|||| INTEGRANTES: CARVAJAL LUIS - LLORENTE ELIAN ||||";

cout<<"\n|||| NRC: 2967 ||||";

cout<<"\n|||| ||||";

cout<<"\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\n\n\n\n\n";

cout<<"\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||";

cout<<"\n|||| ||||\n";

cout<< "|||| MENU ||||";

cout<<"\n|||| ||||";

cout<<"\n|||| ||||";

cout<<"\n|||| ||||";

cout<<"\n|||| ||||";

cout<<"\n|||| ||||";

cout<<"\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\n\n\n";

seleccionarOpcion(tecla);

gotoxy(31, 21); cout<<" APLICATIVO";

gotoxy(31, 22); cout<<" AYUDA";

gotoxy(31, 23); cout<<" PUNTAJE";

gotoxy(31, 24); cout<<" SALIR";

}

/\*\*

\* @brief Funcion main

\*/

int main()

{

//PlaySound(TEXT("undertale.wav"),NULL,SND\_ASYNC);

AltEnter();

char tecla;

while(true){

system("color e");

if( kbhit() )

tecla = getch();

else

tecla = ' ';

menuinicio(tecla);

Sleep(100);

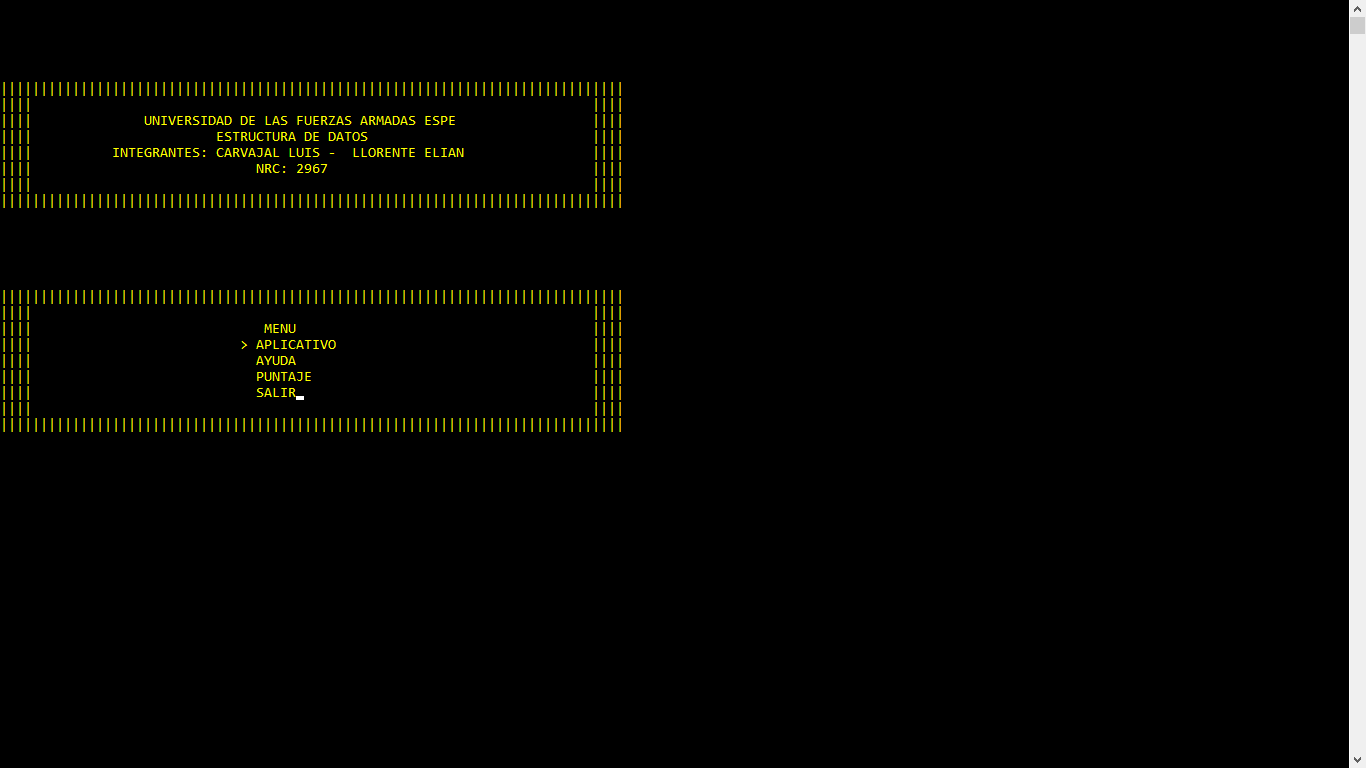
}

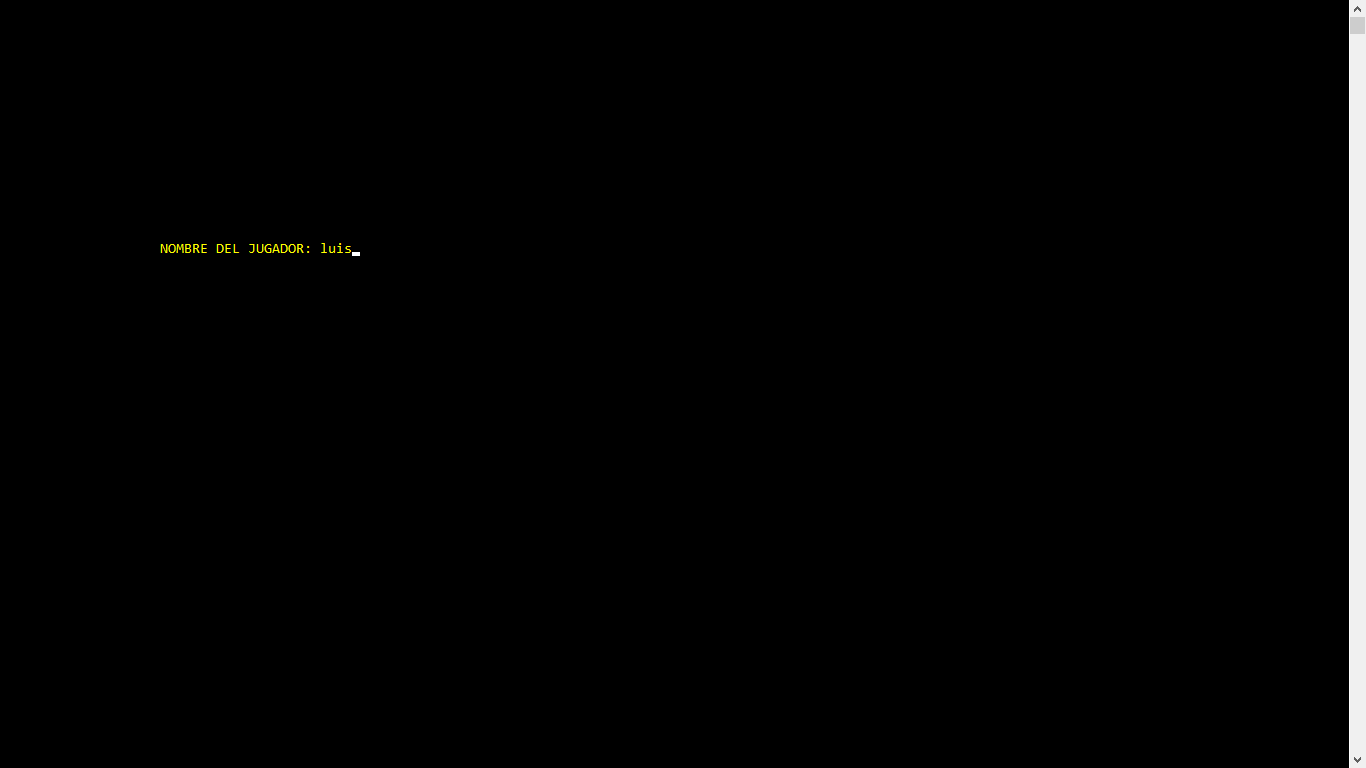
system("pause>null");

return 0;

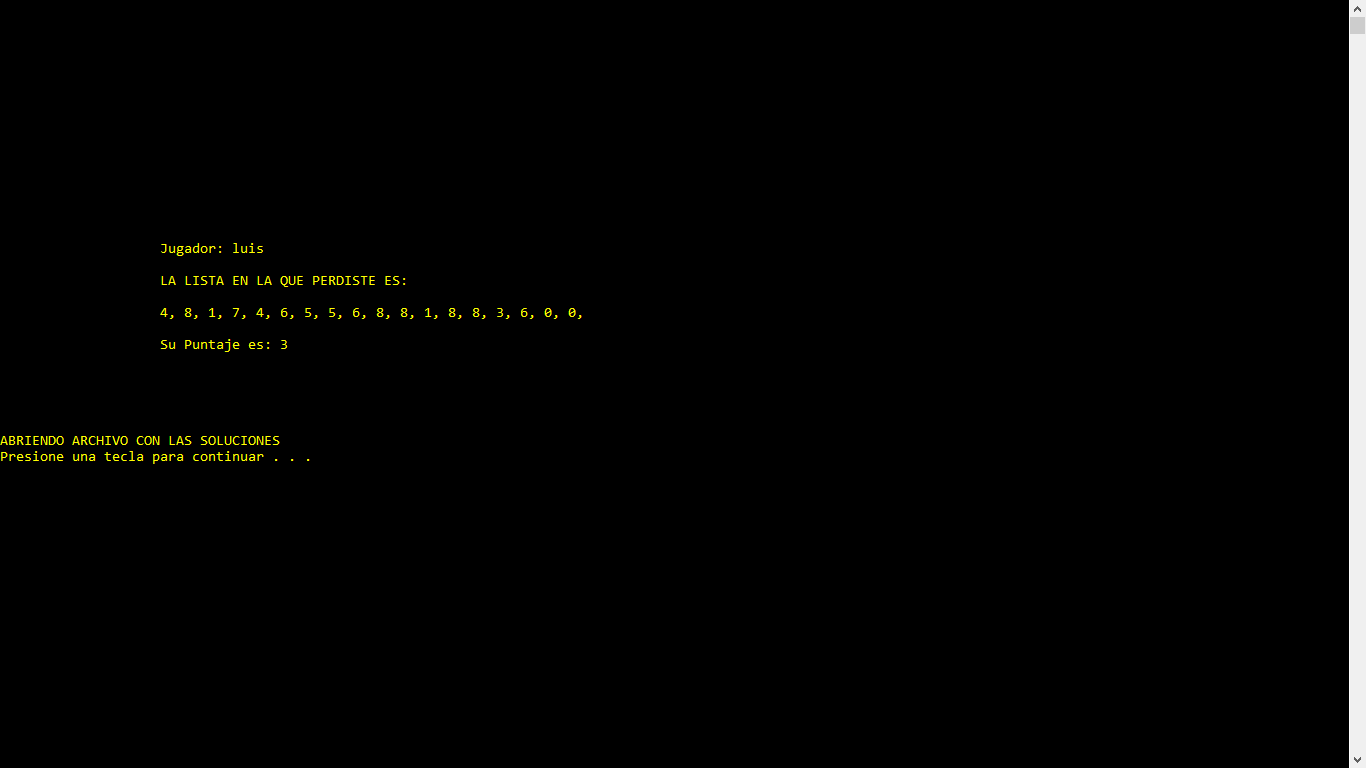
}

1. **Ejecución**











1. **Conclusiones**

En conclusión, se puedo apreciar de mejor manera el uso que se puede dar a las listas mediante el desarrollo de este proyecto, gracias a las cuales nos facilitan, dependiendo de cual se nuestro problema a resolver, encontrar cual de estas se adapta de mejor manera a las necesidades que se tenga en el desarrollo de cualquier proyecto.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DOCENTE RESPONSABLE COORDINADOR DE ÁREA

Ing. Fernando Solis. MsC. PhD. Rodrigo Fonseca.