



Ingeniería de Software

TRABAJOS REALIZADOS EN EL PRIMER PARCIAL

MATERIA: Estructura de Datos
NRC: 2967

Tabla de contenido

Función Matemática (TDA)	- 4 -
Funciones Trigonométricas	- 10 -
Sudoku	- 14 -
Cubo mágico	- 24 -
Generador QR	- 29 -
Factorial recursivo	- 35 -
Ordenamiento por Quicksort.....	- 37 -
Ordenamiento ShellSort	- 42 -
Método de Ordenamiento Heapsort.....	- 45 -
Búsqueda Binaria.....	- 50 -
Método de búsqueda secuencial.....	- 53 -
Puzzle deslizante.....	- 56 -
Contar Vocales.....	- 61 -
Corrección de la prueba.....	- 63 -
Ordenamiento por Inserción	- 69 -
Ocho reinas.....	- 76 -
Generar e-mail:	- 82 -
Polaca Inversa	- 95 -
Prueba Segundo Parcial.....	- 105 -
Suma de a Dos dígitos, Primos.....	- 114 -
Snake	- 119 -
Pilas.....	- 125 -
Colas	- 128 -
Juego de Palabras.....	- 131 -
Números primos	- 141 -
Biblioteca	- 148 -
Sección de proyectos segundo parcial.....	- 168 -
Grupo Picado-Naula.....	- 168 -
Grupo Duy y Puco	- 184 -
Grupo Galarza-Zurita	- 204 -
Grupo Lopez-Zambrano.....	- 229 -
Grupo Bermúdez -Salazar	- 246 -

Grupo Toapanta-Naranjo	- 261 -
Grupo Baez_Cardenas	- 279 -
Grupo Carvajal_Llorente.....	- 288 -
Grupo Avila_Zurita	- 306 -
Grupo Alvarez-Garcia	- 325 -

Introducción

A continuación se presenta un informe detallado de los dieciocho programas elaborados a lo largo del primer parcial, sean deberes, trabajos en clase, pruebas o extras, los cuales constan de una definición, objetivo, código y ejecución (por medio de capturas).

El código ha sido embellecido por medio de una herramienta de software llamada “Notepad++” la cual permite ingresar código en formato de programación para un mejor entendimiento.

Función Matemática (TDA)

Descripción

La siguiente aplicación tiene como objetivo calcular las raíces de una ecuación de segundo grado usando un tipo de dato abstracto (TDA) para lograr eso se ocupará la fórmula general de una ecuación cuadrática, pidiendo al usuario los coeficientes de la ecuación a calcular.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Objetivo de la aplicación

Calcular las raíces de una ecuación de segundo grado mediante la fórmula general..

Código de la aplicación

- La clase Fórmula nos permite alojar los coeficientes de nuestra fórmula a desarrollar para luego ocuparlas en nuestra clase Raíz.

```
class Raiz;

class Formula
{
public:
    Formula();
    float getNum1(void);
    void setNum1(float newNum1);
    float getNum2(void);
    void setNum2(float newNum2);
    float getNum3(void);
    void setNum3(float newNum3);

    Raiz* raiz;
protected:
private:
    float num1;
    float num2;
```

```

    float num3;

};

#include "Raiz.h"
#include "Formula.h"

Formula::Formula()
{
}

float Formula::getNum1(void)
{
    return num1;
}

void Formula::setNum1(float newNum1)
{
    num1 = newNum1;
}

float Formula::getNum2(void)
{
    return num2;
}

void Formula::setNum2(float newNum2)
{
    num2 = newNum2;
}

float Formula::getNum3(void)
{
    return num3;
}

void Formula::setNum3(float newNum3)
{
    num3 = newNum3;
}

```

- La clase Raíz se encargará de calcular mediante la fórmula presentada anteriormente las raíces de nuestra ecuación

```

class Raiz
{
public:
    void calcular(float a, float b, float c);
    float getResult1(void);
    void setResult1(float newResult1);
    float getResult2(void);
    void setResult2(float newResult2);
    Raiz();

protected:
private:

```

```

        float result1;
        float result2;
    };

#include<iostream>
#include<math.h>
#include "Raiz.h"

using namespace std;

void Raiz::calcular(float a, float b, float c)
{
    Raiz raiz;

    float raizValidacion;

    raizValidacion=b*b-4*a*c;
    if(raizValidacion<0){
        cout<<"No existe solucion"<<endl;
    }else{

        raiz.setResult1((-b+sqrt(raizValidacion))/(2*a));
        raiz.setResult2((-b-sqrt(raizValidacion))/(2*a));
        cout<<"La primera solucion es:
x1=<<raiz.getResult1()<<endl;
        cout<<"La segunda solucion es:
x2=<<raiz.getResult2()<<endl;

    }
}

float Raiz::getResult1(void)
{
    return result1;
}

void Raiz:: setResult1(float newResult1)
{
    result1 = newResult1;
}

float Raiz::getResult2(void)
{
    return result2;
}

void Raiz:: setResult2(float newResult2)
{
    result2 = newResult2;
}

Raiz::Raiz()
{
}

```

- La librería “Ingreso.h” nos permitirá que el usuario ingrese los datos para nuestra aplicación

```
#include <iostream>
#include "Validacion.h"

using namespace std;

class Ingreso {

public:
    string leer(string,int);
};

string Ingreso::leer(string mensaje,int tipo) {
    Validacion validacion;
    string entrada;
    cout << mensaje << endl;
    cin >> entrada;
    while (validacion.validar(entrada, tipo)) {
        cout << "Valor no valido reingrese" << endl;
        cin >> entrada;
    }
    return entrada;
}
```

- La librería “Validar.h” nos ayudará a verificar que los datos ingresados sean correctos..

```
#include <iostream>
#include <ctype.h>
#include <string.h>

using namespace std;

class Validacion {
public:
    bool validar(string, int);
};

bool Validacion::validar(string entrada, int tipo) {
    int contador = 0;
    try {
        for (int i = 0; i < entrada.length(); i++) {
            if (isalpha(entrada[i])) {
                throw 1;
            }
            if (!isdigit(entrada[i]) && tipo == 1) {
                throw 1;
            }
            if (entrada[i] == '.') {
                contador++;
            }
            if ((isdigit(entrada[i]) == 0 && entrada[i] != '.') && entrada[i] != '-') ||
                (contador>1)) {
```

```

        throw 1;
    }
}
catch (int e) {
    return true;
}
return false;
}

```

- Este será nuestro aplicativo donde se ejecutará nuestro programa.

```

#include<iostream>
#include <sstream>
#include"Formula.cpp"
#include"Raiz.cpp"
#include"Ingreso.h"

using namespace std;

int main(){

    Formula obj;
    Raiz obj2;
    Ingreso ingreso;
    float a,b,c;
    string dim;

    cout<<">Ingrese los datos (ax^2+bx+c)"<<endl;

    dim=ingreso.leer("Ingrese el coeficiente a",2);
    istringstream (dim) >> a;
    obj.setNum1(a);

    dim=ingreso.leer("Ingrese el coeficiente b",2);
    istringstream (dim) >> b;
    obj.setNum2(b);

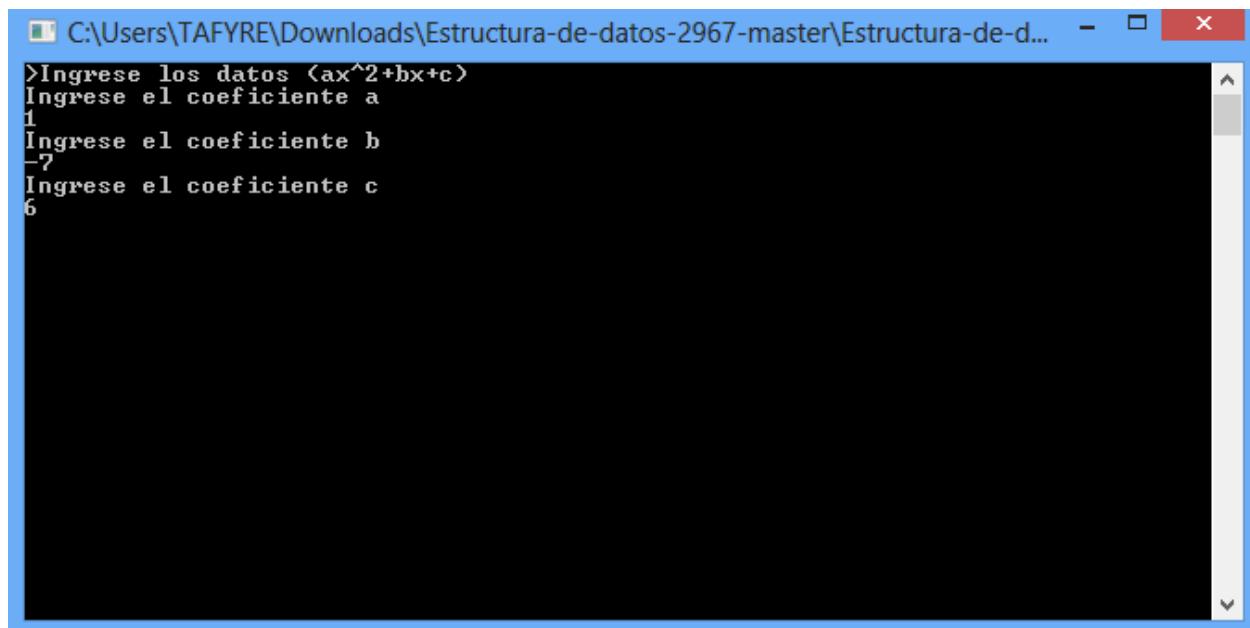
    dim=ingreso.leer("Ingrese el coeficiente c",2);
    istringstream (dim) >> c;
    obj.setNum3(c);

    obj2.calcular(obj.getNum1(),obj.getNum2(),obj.getNum3());

    return 0;
}

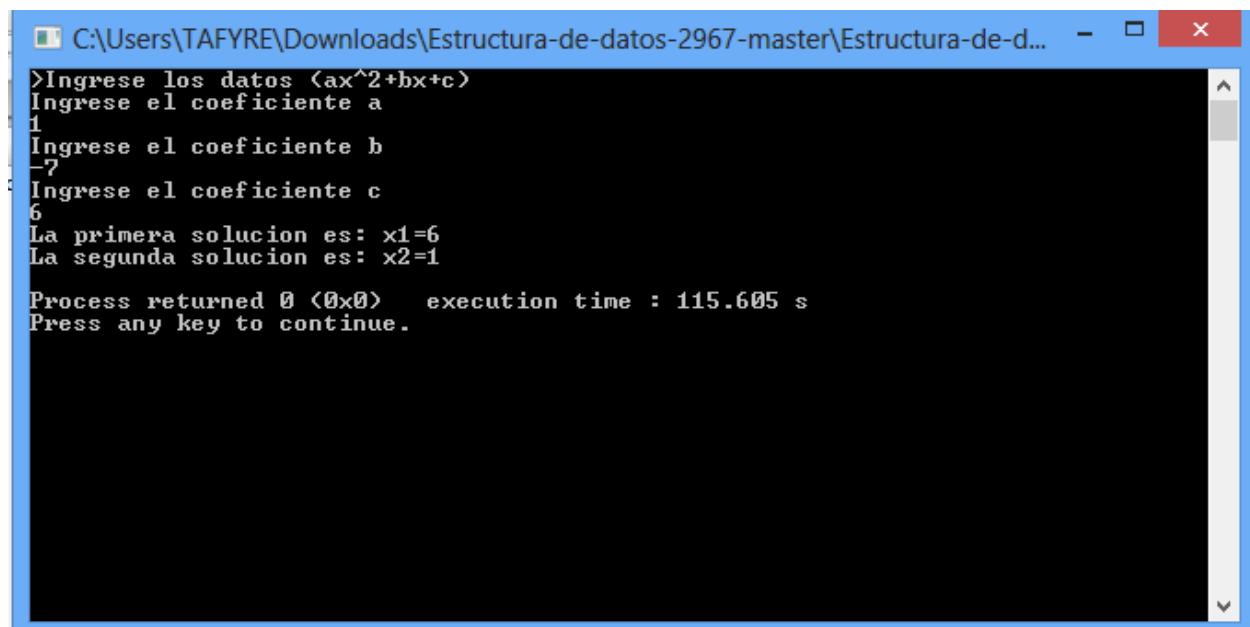
```

Ejecución del aplicativo



```
>Ingrese los datos <ax^2+bx+c>
Ingrese el coeficiente a
1
Ingrese el coeficiente b
-7
Ingrese el coeficiente c
6
```

Gráfico 1. Ingreso coeficientes por parte del usuario.



```
>Ingrese los datos <ax^2+bx+c>
Ingrese el coeficiente a
1
Ingrese el coeficiente b
-7
Ingrese el coeficiente c
6
La primera solucion es: x1=6
La segunda solucion es: x2=1

Process returned 0 (0x0)   execution time : 115.605 s
Press any key to continue.
```

Gráfico 2. Impresión de resultados.

Funciones Trigonométricas

Descripción

La siguiente aplicación tiene como objetivo desarrollar las funciones trigonométricas se definen comúnmente como el cociente entre dos lados de un triángulo rectángulo, asociado a sus ángulos.

Las funciones trigonométricas son funciones cuyos valores son extensiones del concepto de razón trigonométrica en un triángulo rectángulo trazado en una circunferencia unitaria (de radio unidad).

Objetivo de la aplicación

Calcular las Funciones trigonométricas básicas (seno, coseno, tangente) para la aplicación de las mismas en operaciones matemáticas.

Código de la aplicación

- La librería “Funciones Trigonométricas” donde se encuentran las funciones que permiten calcular los valores de las funciones trigonométricas (seno, coseno, tangente).

```
#include <iostream>
#include <stdio.h>
#include <math.h>
#define PI 3.14159265359
using namespace std;

class FuncionesTrigonometricas
{
public:
    int factorial(int angulo);
    float seno(float angulo);
    float coseno(float angulo);
    float tangente(float angulo);
    float radianes(float angulo);
    int potencia(int base,int exponente);
protected:
private:
};

//Factorial de un Número
int FuncionesTrigonometricas::factorial (int angulo)
{ int factor,i;
    factor=1;
    for (i=1;i<=angulo;i++) {
        factor=factor*i;
    }
    return (factor);
}
```

```

// Funcion Seno para un angulo radianes
float FuncionesTrigonometricas::seno(float angulo)
{
    float resultado;
    int i, posicion, precision;
    resultado= angulo;
    precision=5;
    for(i=1; i<=precision; i++) {
        posicion = i * 2 + 1;
        if(i%2==0)
            resultado += potencia(angulo, posicion) / factorial(posicion);
        else
            resultado -= potencia(angulo, posicion) /
factorial(posicion);
    }
    return resultado;
}

//Funcion Coseno para un angulo radianes
float FuncionesTrigonometricas::coseno(float angulo){
    float resultado;
    int i, posicion, precision;
    resultado= 1;
    precision=5;
    for(i=1; i<=precision; i++) {
        posicion = i * 2;
        if(i%2==0)
            resultado += potencia(angulo, posicion) / factorial(posicion);
        else
            resultado -= potencia(angulo, posicion) /
factorial(posicion);
    }
    return resultado;
}

//Funcion Tangente para un angulo radianes
float FuncionesTrigonometricas::tangente(float angulo){
    float coseno;
    float seno;
    float resultado;

    coseno=FuncionesTrigonometricas::coseno(angulo);
    seno=FuncionesTrigonometricas::seno(angulo);
    resultado=seno/coseno;
    return resultado;
}

int FuncionesTrigonometricas::potencia(int base,int exponente){
    int resultado = base;
    if(exponente>0){
        return resultado*potencia(base,exponente-1);
    }else{
        return 1;
    }
}

```

```

float FuncionesTrigonometricas::radianes(float angulo){
    float resultado;
    resultado = angulo*(PI/180);
    return resultado;
}

```

- La librería “Ingreso.h” es la que permite que el usuario digite los datos que van a ser procesados para el cálculo de las funciones trigonométricas.

```

#include <iostream>
#include "Validacion.h"

using namespace std;

class Ingreso {

public:
    string leer(string,int);
};

string Ingreso::leer(string mensaje,int tipo) {
    Validacion validacion;
    string entrada;
    cout << mensaje << endl;
    cin >> entrada;
    while (validacion.validar(entrada, tipo)) {
        cout << "Valor no valido reingrese" << endl;
        cin >> entrada;
    }
    return entrada;
}

```

- La librería “Validacion.h” verifica que el dato ingresado por el usuario sea correcto.

```

#include <iostream>
#include <ctype.h>
#include <string.h>

using namespace std;

class Validacion {
public:
    bool validar(string, int);
};

bool Validacion::validar(string entrada, int tipo) {
    int contador = 0;
    try {
        for (int i = 0; i < entrada.length(); i++) {
            if (isalpha(entrada[i])) {
                throw 1;
            }
            if (!isdigit(entrada[i]) && tipo == 1) {
                throw 1;
            }
            if (entrada[i] == '.') {

```

```

        contador++;
    }
    if ((isdigit(entrada[i]) == 0 && entrada[i] != '.') || 
(contador>1)) {
        throw 1;
    }
}
catch (int e) {
    return true;
}
return false;
}

```

- Este será nuestro aplicativo donde se ejecutará nuestro programa.

```

#include <iostream>
#include <stdlib.h>
#include "FuncionesTrigonometricas.h"
#include "Ingreso.h"
using namespace std;

int main(){
    FuncionesTrigonometricas funciones;
    Ingreso ingreso;
    string angulo1;
    float angulo2;
    float calculo1;
    float calculo2;
    float calculo3;
    cout<< "***CALCULO DE FUNCIONES TRIGONOMETRICAS : SENO, COSENO,
TANGENTE***"<<endl<<endl;
    angulo1 = ingreso.leer("Ingrese un angulo: ",2);
    angulo2 = funciones.radians(atof(angulo1.c_str()));
    calculo1 =funciones.seno(angulo2);
    calculo2 =funciones.coseno(angulo2);
    calculo3 =funciones.tangente(angulo2);
    system("cls");
    cout<< "***CALCULO DE FUNCIONES TRIGONOMETRICAS : SENO, COSENO,
TANGENTE***"<<endl<<endl;
    cout<< endl << "SENO DE: "<< angulo1 << " ES= " << calculo1 <<endl;
    cout<< "COSENO DE: "<< angulo1 << " ES= " <<calculo2 <<endl;
    cout<< "TANGENTE DE: "<< angulo1 << " ES= " <<calculo3 <<endl;
}

```

Ejecución del Aplicativo

```
C:\Users\HP\Desktop\manual\Aplicativo.exe
***CALCULO DE FUNCIONES TRIGONOMETRICAS : SENO, COSENO, TANGENTE***
Ingrese un angulo:
```

Figura 1 : Ingreso de datos por parte del usuario

```
C:\Users\HP\Desktop\manual\Aplicativo.exe
***CALCULO DE FUNCIONES TRIGONOMETRICAS : SENO, COSENO, TANGENTE***

SENO DE: 0 ES= 0
COSENO DE: 0 ES= 1
TANGENTE DE: 0 ES= 0

-----
Process exited after 112.7 seconds with return value 0
Presione una tecla para continuar . . .
```

Figura 2: Funciones trigonométricas

Sudoku

Descripción

El objetivo del sudoku es llenar una cuadrícula de 9×9 celdas (81 casillas) dividida en sub cuadrículas de 3×3 (también llamadas "cajas" o "regiones") con las cifras del 1 al 9 partiendo de algunos números ya dispuestos en algunas de las celdas. Aunque se podrían usar colores, letras, figuras, se conviene en usar números para mayor claridad, lo que importa, es que sean nueve elementos diferenciados, que no se deben repetir en una misma fila, columna o sub cuadrícula.

Objetivo de la aplicación

Simular el juego Sudoku de tal manera que el aplicativo siempre dé como resultado un juego correcto siguiendo las reglas del mismo.

Código de la aplicación

- En la clase “Sudoku.cpp” se encuentran todas las funciones necesarias para la resolución del juego en las cuales se encuentran todas las restricciones del mismo.

```
#include "Sudoku.h"
int** Sudoku::inicializar(int tamano) {
    matriz = (int**) malloc(sizeof (int *)*tamano);
    for (int i = 0; i < tamano; i++) {
        *(matriz + i) = (int*) malloc(sizeof (int*)*tamano);
    }
    return matriz;
}

void Sudoku::tablero(int **matriz) {
    cout << "-----SUDOKU-----\n";
    for (int i = 0; i < 9; i++) {
        printf("\n");
        if (i == 3 || i == 6) {
            cout << "-----\n";
        }
        for (int j = 0; j < 9; j++) {
            if (j == 3 || j == 6) {
                printf(" | ");
            }
            cout << " " << *(*(matriz + i) + j) << " ";
        }
    }
}
int** Sudoku::hacerSolucion(int **matriz) {
    srand(time(NULL));
    do {
        contador2 = 0;
        matriz = inicializar(10);
        //Enceramos la matriz
        for (int i = 0; i < 9; i++) {
            for (int j = 0; j < 9; j++) {
                *(*(matriz + i) + j) = 0;
            }
            contador1 = 0;
            do {
                //Se almacenan numeros aleatorios del 1 al 9
                aleatorio = rand() % 9 + 1;
                //Si la funcion controlar devuelve 1 se almacena el numero
                aleatorio generado, si devuelve 0 vuelve a generar un numero aleatorio
                detente = controlar(aleatorio, 9, i, j, matriz);
                contador1++;
                //Hace que termine los dos ciclos for
                if (contador1 > 15) {
                    i = 9;
                    j = 9;
                }
            } while (detente == 1);
            *(*(matriz + i) + j) = aleatorio;
            //Aumenta el numero de 1 en 1 hasta llegar a 81 para
            terminar el ciclo do-while
            contador2++;
        }
    }
}
```

```

        }
    } while (contador2 != 81);
    return matriz;
}

int Sudoku::controlar(int numero, int tamanio, int contador1, int contador2,
int **matriz) {
    detente = 0;
    for (int i = 0; (i < tamanio && detente == 0); i++) {
        //Se va desplazando y comprueba si el espacio en donde está ubicado es
        igual al numero aleatorio generado tanto en fila como columna
        if (*(*(matriz + contador1) + i) == numero || *(*(matriz + i) +
        contador2) == numero) {
            detente = 1;
        }
    }
    return detente;
}

```

- En la librería “Sudoku.h” se encuentra la funciones declaradas y las variables que serán utilizadas en el desarrollo de funciones para la aplicación.

```

#include <iostream>
#include <time.h>
#include <stdlib.h>

using namespace std;

class Sudoku {
private:
    int **matriz;
    int detente;
    int aleatorio;
    int contador1;
    int contador2;
public:
    int** inicializar(int);
    void tablero(int**);
    int** hacerSolucion(int**);
    int controlar(int, int, int, int, int**);
};

```

- Este será nuestro aplicativo donde se ejecutará nuestro programa.

```

#include "Sudoku.cpp"
int main() {
    Sudoku sudoku;
    int **matriz;
    matriz = sudoku.hacerSolucion(matriz);
    sudoku.tablero(matriz);
    return 0;
}

```

Ejecución del Aplicativo

```

----- SUDOKU -----
6 8 2 | 5 4 3 | 9 1 7
1 9 4 | 2 7 5 | 8 3 6
2 7 3 | 9 1 4 | 6 5 8
-----
8 1 6 | 4 9 7 | 5 2 3
7 2 8 | 1 6 9 | 3 4 5
9 5 1 | 6 3 2 | 7 8 4
-----
3 4 9 | 8 5 6 | 1 7 2
5 6 7 | 3 2 8 | 4 9 1
4 3 5 | 7 8 1 | 2 6 9
-----

Process exited after 1.01 seconds with return value 0
Presione una tecla para continuar . .

```

Figura 1 Tablero de solución Sudoku

Matriz Identidad

Descripción

Es una matriz que cumple la propiedad de ser el elemento neutro del producto de matrices. Esto quiere decir que el producto de cualquier matriz por la matriz identidad (donde dicho producto esté definido) no tiene ningún efecto.

Para obtener la matriz identidad se debe multiplicar la matriz con su inversa.

Objetivo de la aplicación

Calcular la matriz Identidad de cualquier matriz ingresada de dimensión n, mediante el uso de procesos matemáticos.

Código de la aplicación

- A continuación presentamos el código de la aplicación “identidad.h”

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

class Identidad {
private:
    float **matriz;
    float *matriz1;
    float elemento;
    float coeficiente;
    float *vector;
    float **respuesta;

```

```

public:
    float* inicializar1(int);
    float** inicializar(int);
    float** encerar(int,float**);
    float* encerar1(int,float*);
    float** ingresarNumeros(int,float**);
    float** calcularInversa(int,float**);
    float** multiplicar(int,float**,float**);
    void mostrarInversa(int,float**);
    void mostrar(int,float**);
};

float* Identidad::inicializar1(int tamano) {
    matriz1 = (float*)malloc(tamano *sizeof(float));
    return matriz1;
}

float* Identidad::encerar1(int tamano,float *matriz) {
    for(int i=0; i<tamano; i++) {
        for(int j=0; j<tamano; j++) {
            *(matriz+i) = 0;
        }
    }
    return matriz;
}

float** Identidad::inicializar(int tamano) {
    matriz = (float**)malloc(sizeof(float *)*tamano);
    for(int i=0; i<tamano; i++) {
        *(matriz+i) = (float*)malloc(sizeof(float)*tamano);
    }
    return matriz;
}

float** Identidad::encerar(int tamano,float **matriz) {
    for(int i=0; i<tamano; i++) {
        for(int j=0; j<tamano; j++) {
            *(*(matriz+i)+j) = 0;
        }
    }
    return matriz;
}

float** Identidad::ingresarNumeros(int tamano,float **matriz) {
    matriz = inicializar(tamano);
    matriz = encerar(tamano,matriz);
    for(int i=0;i<tamano;i++) {
        for(int j=0;j<tamano;j++) {
            scanf("%f",*(matriz+i)+j);
        }
    }
    return matriz;
}

float** Identidad::calcularInversa(int tamano,float **matriz) {
    vector = inicializar1(tamano);
    vector = encerar1(tamano,vector);
}

```

```

        for(int i=0; i<tamanio; i++) {
            for(int j=tamanio; j<2*tamanio; j++) {
                if(i==(j-tamanio)) {
                    *(*(matriz+i)+j) = 1;
                } else {
                    *(*(matriz+i)+j) = 0;
                }
            }
        }
        for(int s=0; s<tamanio; s++) {
            elemento = *(*(matriz+s)+s);
            for(int j=0; j<2*tamanio; j++) {
                *(*(matriz+s)+j) = *(*(matriz+s)+j) / elemento;
            }
            for(int i=0; i<tamanio; i++) {
                if(i==s) {
                    ;
                } else {
                    coeficiente = *(*(matriz+i)+s);
                    for(int j=0; j<2*tamanio; j++)
                        *(vector+j) = *(*(matriz+s)+j) * (coeficiente*-1);
                    for(int j=0; j<2*tamanio; j++)
                        *(*(matriz+i)+j) = *(*(matriz+i)+j) + *(vector+j);
                }
            }
        }
    }
    return matriz;
}

void Identidad::mostrar(int tamanio, float **matriz) {
    for(int i=0; i<tamanio; i++) {
        for(int j=0; j<tamanio; j++) {
            printf(" %.0f ",*(*(matriz+i)+j));
        }
        printf("\n");
    }
}

void Identidad::mostrarInversa(int tamanio, float **matriz) {
    for(int i=0; i<tamanio; i++) {
        for(int j=tamanio; j<2*tamanio; j++) {
            printf(" %.0lf ",*(*(matriz+i)+j));
        }
        printf("\n");
    }
}

float** Identidad::multiplicar(int tamanio, float **matriz1, float **matriz2)
{
    respuesta = inicializar(tamanio);
    respuesta = encerar(tamanio,respuesta);
    for(int i=0;i<tamanio;i++) {
        for(int j=0;j<tamanio;j++) {
            for(int k=0;k<tamanio;k++) {
                *(*(respuesta+i)+j) = *(*(respuesta+i)+j) +
                (*(*(matriz1+i)+k)) * (*(*(matriz2+k)+j));
            }
        }
    }
}

```

```

        }
    }
    return respuesta;
}

```

- Luego tenemos la clase ingreso la cual valida nuestras entradas hacia el programa “ingreso.h”

```

#include <stdio.h>
#include <iostream>
#include <string.h>
#include <sstream>
#include <stdlib.h>

using namespace std;

class Ingreso{

public:
    char* ingresar(char*);
    string ingresarSoloTexto(char*);
    int ingresarEntero(char *);
    float ingresarFlotante(char *);
};

char* Ingreso::ingresar(char* msg){
    char* texto;
    cout<<msg<<endl;
    cin>>texto;
    return texto;
}

string Ingreso::ingresarSoloTexto(char *msg){
    float valor;
    string texto;
    string res;

    while (1)
    {
        bool is_valid = true;
        cout << msg << endl;
        getline(cin, texto);
        try{
            for (size_t i = 0; i < texto.length(); i++) {

                if (!isalpha(texto[i])) {
                    if(texto[i] == ' ') {
                        continue;
                    }else{
                        cout << "Se debe ingresar solo letras\n";
                        is_valid = false;
                        break;
                    }
                }
            }
        }
    }
}

```

```

        }
    }catch(exception e){
        cout<<"error";
    }

    if (is_valid){
        res = texto.c_str();
        break;
    }
}
return res;
}

float Ingreso::ingresarFlotante(char *msg){
    float valor;
    string numero;

    while (1)
    {
        bool is_valid = true;
        cout << msg << endl;
        cin >> numero;
        try{
            for (size_t i = 0; i < numero.length(); i++) {

                if (!isdigit(numero[i])) {
                    if(!(numero[i]=='.')){
                        cout << "Se debe ingresar numeros\n";
                        is_valid = false;
                        break;
                    }
                }
            }
        }catch(exception e){
            cout<<"error";
        }

        if (is_valid){
            stringstream geek(numero);
            geek>>valor;
            break;
        }
    }
    return valor;
}

int Ingreso::ingresarEntero(char *msg){
    int valor;
    string numero;
    char *res;

    while (1)
    {
        bool is_valid = true;
        cout << msg << endl;
        cin >> numero;
        try{

```

```

        for (size_t i = 0; i < numero.length(); i++) {
            if (!isdigit(numero[i])) {
                cout << "Se debe ingresar numeros\n";
                is_valid = false;
                break;
            }
        }
    }catch(exception e){
        cout<<"error";
    }

    if (is_valid){
        res = (char *)numero.c_str();
        valor=atoi(res);
        break;
    }
}
return valor;
}

```

- Finalmente tenemos la implementación del método principal (Main) “main.cpp”

```

#include <iostream>
#include "identidad.h"
#include "ingreso.h"
using namespace std;

int main(){
    Ingreso lee;
    Identidad obj;
    float **matriz;
    float **inversa;
    float *matriz1;
    float elemento;
    float coeficiente;
    float *vector;
    float **respuesta;
    int dimension=0;
    do{
        system("cls");
        dimension = lee.ingresarEntero("Ingrese dimension de la matriz: ");
    }while(dimension == 0 || dimension < 0);

    //RESERVO MEMORIA PARA LAS MATRICES
    matriz = obj.inicializar(dimension);
    inversa = obj.inicializar(dimension);
    respuesta = obj.inicializar(dimension);

    //ENCERO LAS MATRICES PARA EVITAR CUALQUIER FALLA
    obj.encerar(dimension,matriz);
    obj.encerar(dimension,respuesta);
}

```

```

obj.encerar(dimension,inversa);

//PIDO QUE SE INGRESEN LOS VALORES DE LA MATRIZ
matriz = obj.ingresarNumeros(dimension,matriz);
cout<<"Matriz ingresada: "<<endl;
obj.mostrar(dimension,matriz);

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

//CALCULO LA INVERSA DE LA MATRIZ ANTES INGRESADA
inversa = obj.calcularInversa(dimension,matriz);
cout<<"Matriz inversa: "<<endl;
obj.mostrarInversa(dimension,inversa);

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

//MULTIPLICO LA MATRIZ CON SU INVERSA Y GUARDO EL RESULTADO (MATRIZ
IDENTIDAD) EN RESPUESTA
respuesta = obj.multiplicar(dimension,matriz,inversa);
cout<<"Matriz resultante (identidad): "<<endl;
obj.mostrar(dimension,respuesta);

system("pause");
return 0;
}

```

Ejecución de la aplicación

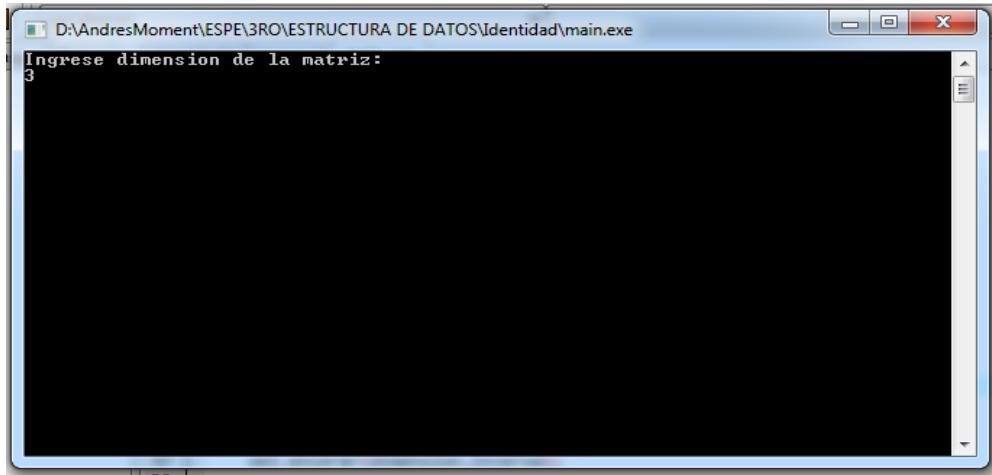


Figura 1: Ejecución de la aplicación

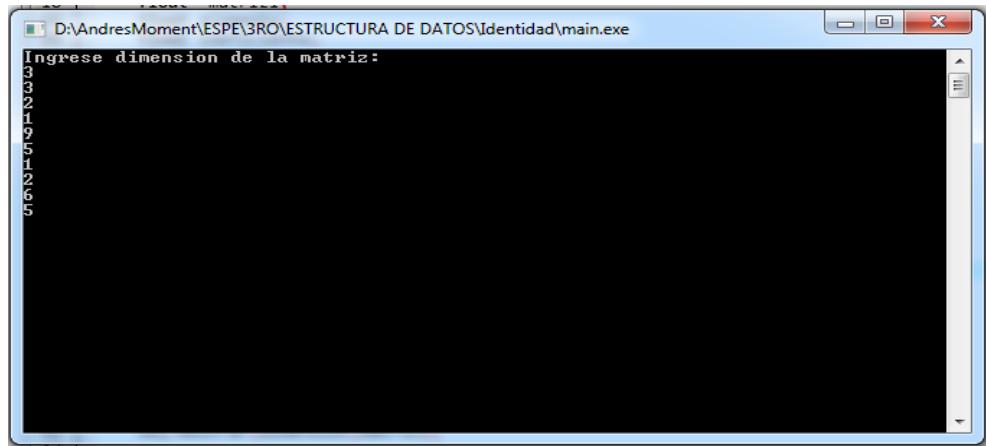


Figura 2: Ingreso de Datos

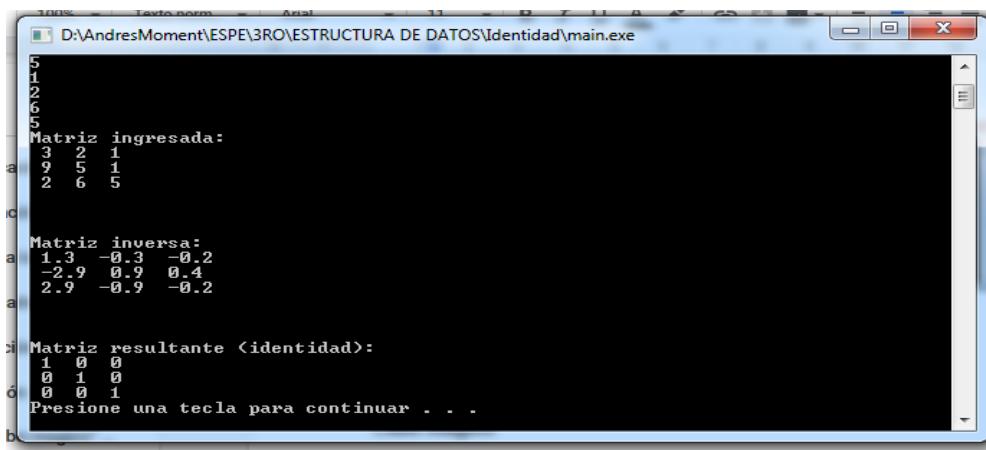


Figura 3: Cálculo de la Matriz identidad

Cubo mágico

Descripción

Un cubo mágico se obtiene colocando una serie de números naturales en una matriz cuadrada de tal forma que todas las filas, todas las columnas y las diagonales sumen el mismo número: la constante mágica.

Generalmente suelen colocarse los números entre 1 y n^2 , siendo n el número de filas y columnas del cuadrado. A este número n se le denomina orden del cubo mágico.

Objetivo de la aplicación

La aplicación nos permite ingresar un número impar para la dimensión de la matriz y este posteriormente generará números en el que la suma de sus filas, columnas y diagonal van a ser iguales.

Código de la aplicación

- La librería “CuboMagico” nos permite encontrar los valores de filas y columnas que cumplan con la condición de que la suma de ellas y su diagonal sean iguales.

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
using namespace std;

class CuboMagico{
private:
    int **cuadrado;
public:
    CuboMagico(int **);
    void llenar(int,int);
    void mostrar(int, int);
    int **getCuboMagicoCuadrado();
    void setCuboMagicoCuadrado(int **);
};

CuboMagico::CuboMagico(int **_cuadrado){
    cuadrado=_cuadrado;
}

int **CuboMagico::getCuboMagicoCuadrado () {
    return cuadrado;
}

void CuboMagico::setCuboMagicoCuadrado(int **_cuadrado) {
    cuadrado=_cuadrado;
}

void CuboMagico::llenar(int a,int b){
    int x=0,k=0,p=1,j=0,t=0,s=0,d=0;
    x=(b-1)/2;
    t=((a+1)/2);
    s=(a-x-1)/2;
    d=x;

    for(int g=0; g<((a+1)/2);g++) {
        j=g;
        k=g;
        for( int i=x;i>=j;i--) {
            *(*(cuadrado+i)+k)=p;
            if( k<(d-s) ){
                *(*(cuadrado+i)+(k+t))=p;
            }
        }
    }
}
```

```

        if(k>(d+s)){
            *(*(cuadrado+i)+(k-t))=p;
        }
        if(i>(d+s)){
            *(*(cuadrado+(i-t))+k)=p;
        }
        if(i<(d-s)){
            *(*(cuadrado+(i+t))+k)=p;
        }

        k++;
        p++;
    }
    x++;
}
}

void CuboMagico::mostrar(int a,int b){
    system("cls");
    int x=0,t=0,s=0;
    t=((a+1)/2);
    x=(b-1)/2;
    s=(a-x-1)/2;

    for(int l=0;l<a;l++){
        cout<<"\n\t";
        for(int h=0;h<b;h++){
            if(h>=(x-s) && h<=(x+s) && l<=(x+s) && l>=(x-s)){
                cout<<"\t"<<*(*(cuadrado+l)+h);
            }
        }
    }
}
}

```

- La librería “Ingreso.h” es la que permite al usuario dar el valor de las dimensiones del cubo mágico

```

#include <iostream>
#include "Validacion.h"

using namespace std;

class Ingreso {

public:
    string leer(string,int);
};

string Ingreso::leer(string mensaje,int tipo) {
    Validacion validacion;
    string entrada;
    cout << mensaje << endl;
    cin >> entrada;
    while (validacion.validar(entrada, tipo)) {
        cout << "Valor no valido reingrese" << endl;
        cin >> entrada;
    }
}

```

```

    }
    return entrada;
}

```

- La librería “Validacion.h” en la que verifica que el valor ingresado sea correcto y cumpla con la condición de que solo puede ingresar un número

```

#include <iostream>
#include <ctype.h>
#include <string.h>

using namespace std;

class Validacion {
public:
    bool validar(string, int);
};

bool Validacion::validar(string entrada, int tipo) {
    int contador = 0;
    try {
        for (int i = 0; i < entrada.length(); i++) {
            if (isalpha(entrada[i])) {
                throw 1;
            }
            if (!isdigit(entrada[i]) && tipo == 1) {
                throw 1;
            }
            if (entrada[i] == '.') {
                contador++;
            }
            if ((isdigit(entrada[i]) == 0 && entrada[i] != '.') || (contador>1)) {
                throw 1;
            }
        }
        catch (int e) {
            return true;
        }
        return false;
    }
}

```

- Este será nuestro aplicativo donde se ejecutará nuestro programa.

```

#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
#include <sstream>
#include "CuboMagico.h"
#include "Ingreso.h"

using namespace std;

```

```

int main()
{
    int **cuadrado;
    Ingreso ingreso;
    int num;
    string dim;
    do{
        dim=ingreso.leer("Digite el tamano del arreglo, solo numeros impares:
",2);
        istringstream (dim) >> num;
    }while(num%2==0 || num<0);
    num=2*num-1;

    cuadrado=(int**)calloc(num,sizeof(int*));
    for(int i=0;i<num;i++){
        *(cuadrado+i)=(int*)calloc(num,sizeof(int));
    }

    CuboMagico cubo=CuboMagico(cuadrado);
    cubo.llenar(num,num);
    cubo.mostrar(num,num);

    system("pause");
    return 0;
}

```

Ejecución del aplicativo

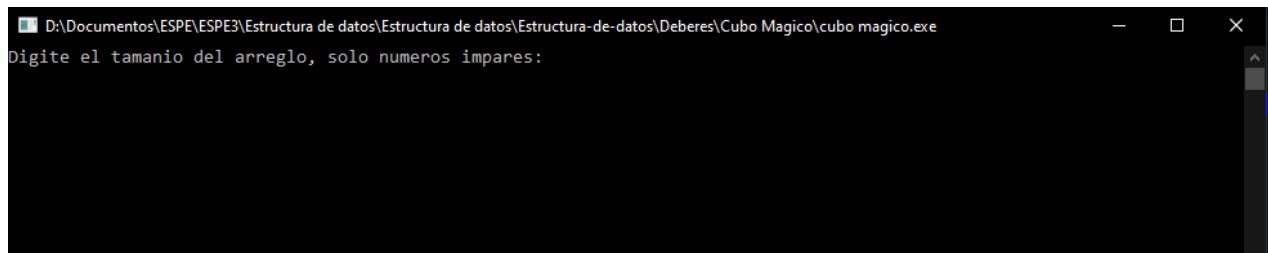


Figura 1. Ejecución de la aplicación

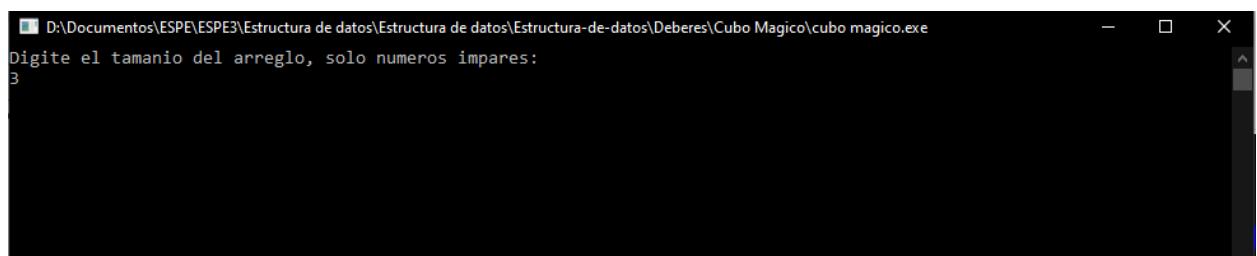


Figura 2. Ingreso de datos

D:\Documentos\ESPE\ESPE3\Estructura de datos\Estructura de datos\Deberes\Cubo Magico\cubo magico.exe

2	7	6
9	5	1
4	3	8

Presione una tecla para continuar . . .

Figura 3. Cubo mágico

Generador QR

Un código QR es un código de barras bidimensional cuadrada que puede almacenar los datos codificados. La mayoría del tiempo los datos es un enlace a un sitio web. Un código QR es la evolución del código de barras. Es un módulo para almacenar información en una matriz de puntos o en un código de barras bidimensional.

Objetivo de la aplicación

Ingresar el tamaño que deseemos para el QR y este nos genera automáticamente una representación gráfica del código QR.

Código de la aplicación

- La librería “Ingreso.h” es la que permite al usuario dar el valor de las dimensiones del cubo mágico.

```
#include <iostream>
#include <stdlib.h>
#include <string.h>

using namespace std;

class Ingreso {
public:
    int ingresarEntero();
    double ingresarDouble();
    float ingresarFloat();
    string ingresarLetra();
    bool validar(string);
    bool validarEntero(string);
    bool validarLetra(string);
};

int Ingreso::ingresarEntero() {
```

```

        string numero;
        bool valido = false;
        while(!valido) {
            try {
                getline(cin,numero);
                valido = validarEntero(numero);
                if(!valido) {
                    throw numero;
                }
            }catch(string e) {
                cout << "El numero " << e << " no es valido" << endl;
            }
        }
        return atoi(numero.c_str());
    }

double Ingreso::ingresarDouble() {
    string numero;
    bool valido = false;
    while(!valido) {
        try {
            getline(cin,numero);
            valido = validar(numero);
            if(!valido) {
                throw numero;
            }
        }catch(string e) {
            cout << "El numero " << e << " no es valido" << endl;
        }
    }
    return atof(numero.c_str());
}

float Ingreso::ingresarFloat() {
    string numero;
    bool valido = false;
    while(!valido) {
        try {
            getline(cin,numero);
            valido = validar(numero);
            if(!valido) {
                throw numero;
            }
        }catch(string e) {
            cout << "El numero " << e << " no es valido" << endl;
        }
    }
    return atof(numero.c_str());
}

string Ingreso::ingresarLetra() {
    string palabra;
    bool valido = false;
    while(!valido) {
        try {
            getline(cin,palabra);
            valido = validarLetra(palabra);
        }
    }
}

```

```

        if(!valido) {
            throw palabra;
        }
    }catch(string e) {
        cout << "La palabra " << e << " no es valida" <<
endl;
    }
}

return palabra;
}

bool Ingreso::validar(string numero) {
    int inicio = 0;
    if(numero.length() == 0) {
        return 0;
    }
    if(numero[0] == '+' || numero[0] == '-') {
        inicio = 1;
        if(numero.length() == 1) {
            return 0;
        }
    }
    for(int i=inicio; i<numero.length(); i++) {
        if(!isdigit(numero[i]) && numero[i] != '.') {
            return 0;
        }
    }
    return 1;
}

bool Ingreso::validarEntero(string numero) {
    int inicio = 0;
    if(numero.length() == 0) {
        return 0;
    }
    if(numero[0] == '+' || numero[0] == '-') {
        inicio = 1;
        if(numero.length() == 1) {
            return 0;
        }
    }
    for(int i=inicio; i<numero.length(); i++) {
        if(!isdigit(numero[i])) {
            return 0;
        }
    }
    return 1;
}

bool Ingreso::validarLetra(string palabra) {
    char c;
    for(int i=0; i<palabra.size(); i++) {
        c=palabra[i];
        if(isalpha(c) == 0) {
            if(isspace(c) == 0) {
                return 0;
            }
        }
    }
}

```

```

        }
    }
    return 1;
}

```

- La librería “generar” realiza gráficamente el código QR con el tamaño que nosotros ingresemos

```

#include <iostream>
#include <time.h>
#include <stdlib.h>

using namespace std;

class QR {
private:
    char **matriz;
    int caracter;
public:
    char** inicializar(int);
    char** encerar(int,char**);
    void generarQR(int,char**);
};

char** QR::inicializar(int tamano) {
    matriz = (char**)malloc(sizeof(char *)*tamano);
    for(int i=0; i<tamano; i++) {
        *(matriz+i) = (char*)malloc(sizeof(char)*tamano);
    }
    return matriz;
}

char** QR::encerar(int tamano,char **matriz) {
    for(int i=0; i<tamano; i++) {
        for(int j=0; j<tamano; j++) {
            *(*(matriz+i)+j) = 0;
        }
    }
    return matriz;
}

void QR::generarQR(int tamano, char **matriz) {
    matriz = inicializar(tamano);
    matriz = encerar(tamano,matriz);
    cout << ' ';
    for(int i=0; i<tamano*2; i++) {
        cout << '-';
    }
    cout << endl;
    srand(time(NULL));
    for(int i=0; i<tamano; i++) {
        cout << '|';
        for(int j=0; j<tamano ; j++) {
            caracter = rand()%2;
            *(*(matriz+i)+j) = caracter;
            if(*(*(matriz+i)+j)==0){

```

```

        cout << ' ';
    }else {
        cout << '*';
    }
    cout << ' ';
}
cout << '|';
for(int i=0; i<tamanio*2; i++) {
    cout << '-';
}
cout << endl;
}

```

- Este será nuestro aplicativo donde se ejecutará nuestro programa.

```

#include "generador.h"
#include <iostream>
#include "ingreso.h"

using namespace std;

int main() {
    Ingreso leer;
    QR obtener;
    int tamanio;
    char **matriz;
    cout << "Ingrese el tamaño que desea el QR" << endl;
    tamanio = leer.ingresarEntero();
    obtener.generarQR(tamanio, matriz);

    return 0;
}

```

Ejecución de la aplicación

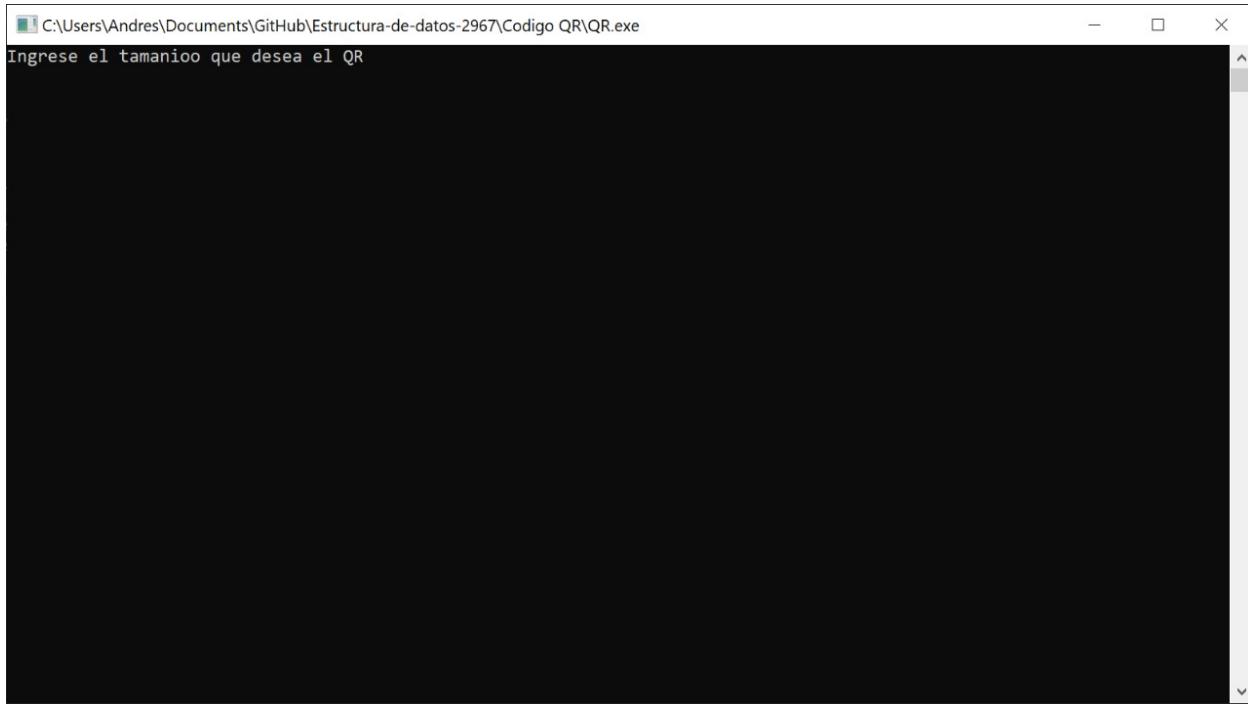


Figura 1. Ingreso del tamaño del QR

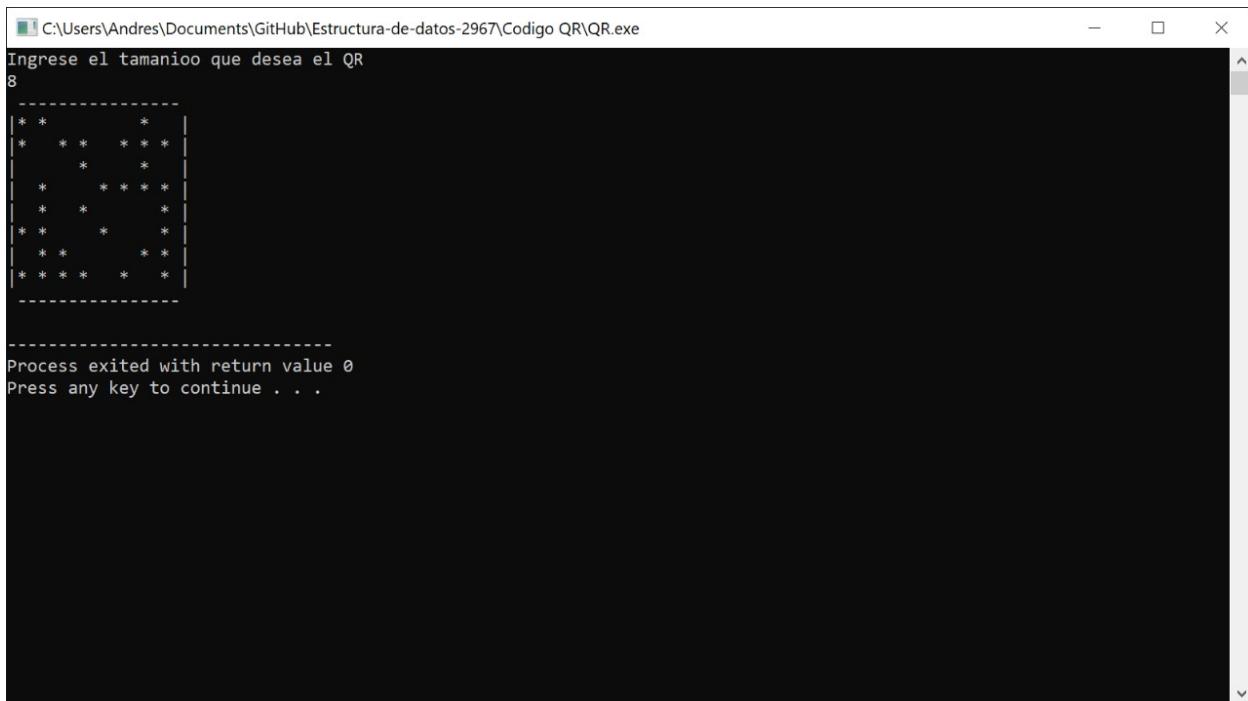


Figura 2. Representación del código QR

Factorial recursivo

Descripción

La operación de factorial aparece en muchas áreas de las matemáticas, particularmente en combinatoria y análisis matemático. De manera fundamental el factorial de n representa el número de formas distintas de ordenar n objetos distintos (elementos sin repetición).

El factorial del número entero positivo n , denotado $n!$, se define como el producto de todos los números enteros positivos menores o iguales que n .

$$n! = 1 \times 2 \times 3 \times 4 \times \dots \times (n - 1) \times n$$

Objetivo de la Aplicación

- Realizar una función para nuestro aplicativo que nos dé como resultado el factorial de un número usando la definición de recursividad.

Código de la aplicación

- A continuación se mostrará la librería “Dato.h” donde se encontrará nuestra función factorial

```
class Dato
{
    private:
        int valor;
    public:
        Dato(int);
        Dato();
        int getValor();
        void setValor(int);
        int factorial(int);
};

Dato::Dato(int valor)
{
    this->valor=valor;
}
Dato::Dato()
{
    this->valor=0;
}
int Dato::getValor()
{
    return valor;
}
void Dato::setValor(int valor)
{
    this->valor=valor;
}
```

```

int Dato::factorial(int valor)
{
    if(valor==1)
    {
        return 1;
    }
    else{
        return valor*factorial(valor-1);
    }
}

```

- Este será nuestro main para ejecutar el aplicativo

```

#include<iostream>
#include "Ingreso.h"
#include<stdlib.h>
#include "Dato.h"

using namespace std;

int main()
{
    Ingreso leer;
    char valor;
    Dato dato=Dato();
    dato.setValor(leer.ingresarInt(&valor));
    dato.setValor(dato.factorial(dato.getValor()));
    cout<<dato.getValor()<<endl;
    system("pause");
    return 0;
}

```

Ejecución de la aplicación

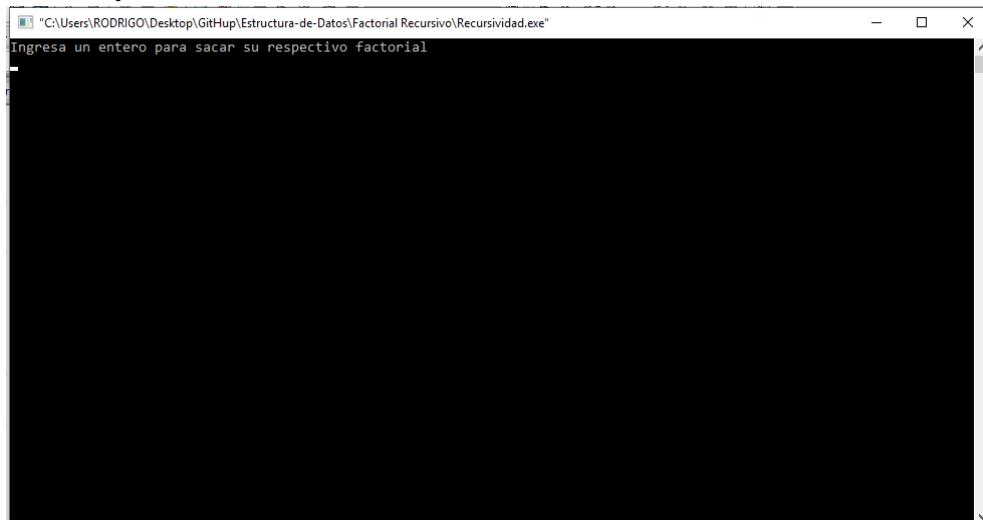


Figura 1. Ejecución de la aplicación

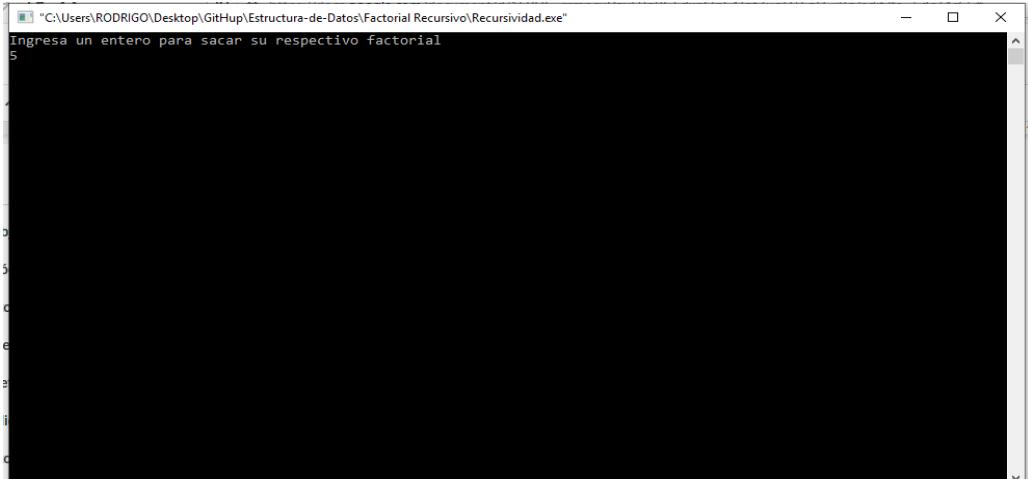


Figura 2. Inserción de Datos

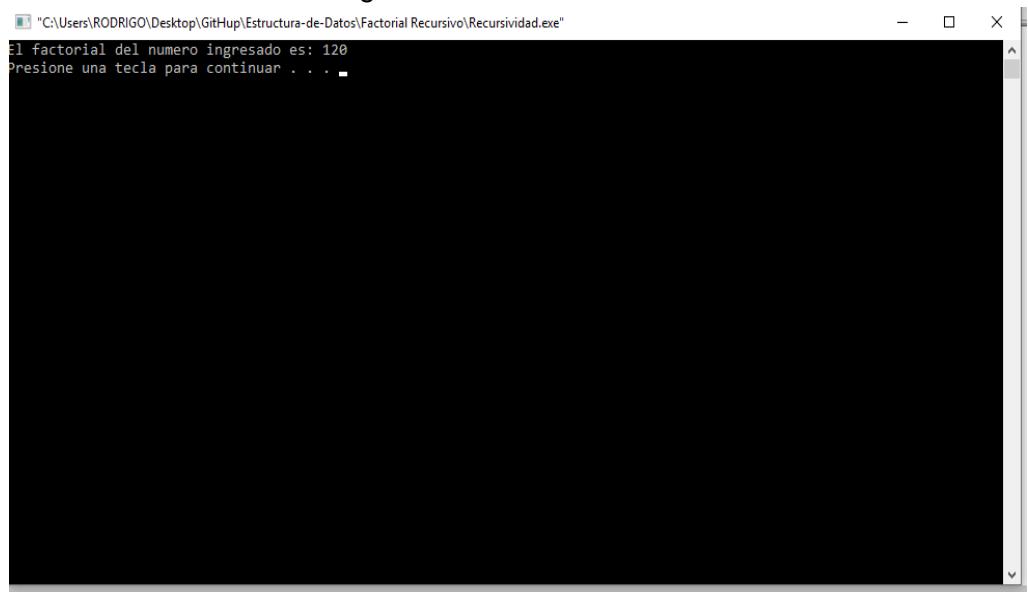


Figura 3. Factorial Recursivo

Ordenamiento por Quicksort

Descripción

Es un algoritmo creado por el científico británico en computación Tony Hoare y basado en la técnica de divide y vencerás. Esta es la técnica quizás la más eficiente y en ella que en la mayoría de los casos da mejores resultados

Objetivo de la aplicación

El objetivo de estudiar los algoritmos de ordenamiento es que permite ejemplificar la importancia del estudio de la eficiencia de los algoritmos tanto en rapidez como en implementación lleva este método.

Código de la aplicación

- En la librería “Quicksort.h” se encuentran todas las funciones necesarias para que el proceso de ordenamiento se realice de una manera correcta.

```
include <iostream>
#include <stdlib.h>
#include "Ingreso.h"
using namespace std;
void leeCadena(int cant, int* n) {
    Ingreso ingreso;

    for (int i = 0; i < cant; i++) {
        *n + i) = atoi(ingreso.ingresar("Ingrese valor entero: ").c_str());
    }
}

void muestraCadena(int cant, int* n) {
    cout << "Cadena ordenada por Quicksort" << endl;
    for (int i = 0; i < cant; i++) {
        printf("%d ", *(n + i));
    }
}

void quicksort(int* A, int izq, int der) {
    int aux;
    int i = izq;
    int j = der;
    int x = *(A + ((izq + der) / 2));
    do {
        while ((*A + i) < x) && (j <= der)) {
            i++;
        }

        while ((x < *(A + j)) && (j > izq)) {
            j--;
        }

        if (i <= j) {
            aux = *(A + i);
            *(A + i) = *(A + j);
            *(A + j) = aux;
            i++;
            j--;
        }
    } while (i <= j);

    if (izq < j)
        quicksort(A, izq, j);
    if (i < der)
        quicksort(A, i, der);
}
```

- La librería “Ingreso.h” es la que permite que el usuario digite los datos que van a ser procesados para el cálculo de las funciones trigonométricas.

```
#ifndef INGRESO_H
#define INGRESO_H
#include <iostream>
#include <string>

using namespace std;

class Ingreso {
public:
    string ingresar(string);
    bool validarTipoFloat(string);
    bool validarTipoInt(string);
    bool validarTipoString(string);
};

string Ingreso::ingresar(string msg) {
    string dato_a_validar;
    bool esValido = false;

    while (!esValido) {
        try {
            cout << msg;
            getline(cin, dato_a_validar);
            if (msg.find("flotante") != std::string::npos) {
                esValido = validarTipoFloat(dato_a_validar);
            }
            else if (msg.find("entero") != std::string::npos) {
                esValido = validarTipoInt(dato_a_validar);
            }
            else if (msg.find("cadena") != std::string::npos) {
                esValido = validarTipoString(dato_a_validar);
            }

            if (!esValido) {
                throw dato_a_validar;
            }
        }
        catch (string e) {
            cout << "El dato (" << e << ") no es valido" << endl;
        }
    }
    return dato_a_validar;
}

bool Ingreso::validarTipoFloat(string numero) {
    int inicio = 0;
    if (numero.length() == 0) {
        return 0;
    }
    if (numero[0] == '+' || numero[0] == '-') {
        inicio = 1;
    }
    if (numero.length() == 1) {

```

```

        return 0;
    }
}
for (int i = inicio; i < numero.length(); i++) {
    if (!isdigit(numero[i]) && numero[i] != '.') {
        return 0;
    }
}
return 1;
}

bool Ingreso::validarTipoInt(string numero) {
    int inicio = 0;
    if (numero.length() == 0) {
        return 0;
    }
    if (numero[0] == '+' || numero[0] == '-') {
        inicio = 1;
    }
    if (numero.length() == 1) {
        return 0;
    }
}
for (int i = inicio; i < numero.length(); i++) {
    if (!isdigit(numero[i])) {
        return 0;
    }
}
return 1;
}

bool Ingreso::validarTipoString(string numero) {
    if (numero.length() == 0) {
        return 0;
    }

    for (int i = 0; i < numero.length(); i++) {
        if (isdigit(numero[i])) {
            return 0;
        }
    }
}
return 1;
}

```

- Este será nuestro main para ejecutar nuestro aplicativo.

```

#include <iostream>
#include "Quicksort.h"
#include "Ingreso.h"
using namespace std;
int main() {
    Ingreso ingreso;
    int n = atoi(ingreso.ingresar("Indique valor entero de numeros a
ingresar: ").c_str());
    int* A = (int*) malloc(n * sizeof (int));

```

```
    leeCadena(n, A);
    quicksort(A, 0, n - 1);
    muestraCadena(n, A);
    return 0;
}
```

Ejecución del Aplicativo

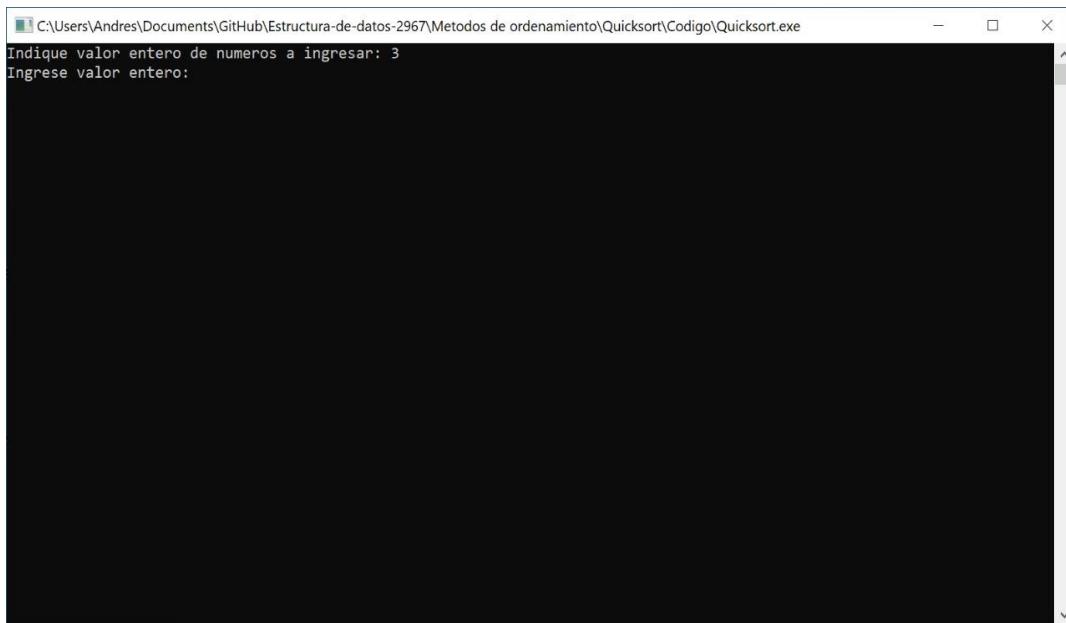


Figura 1: Se ingresa los valores a ordenar

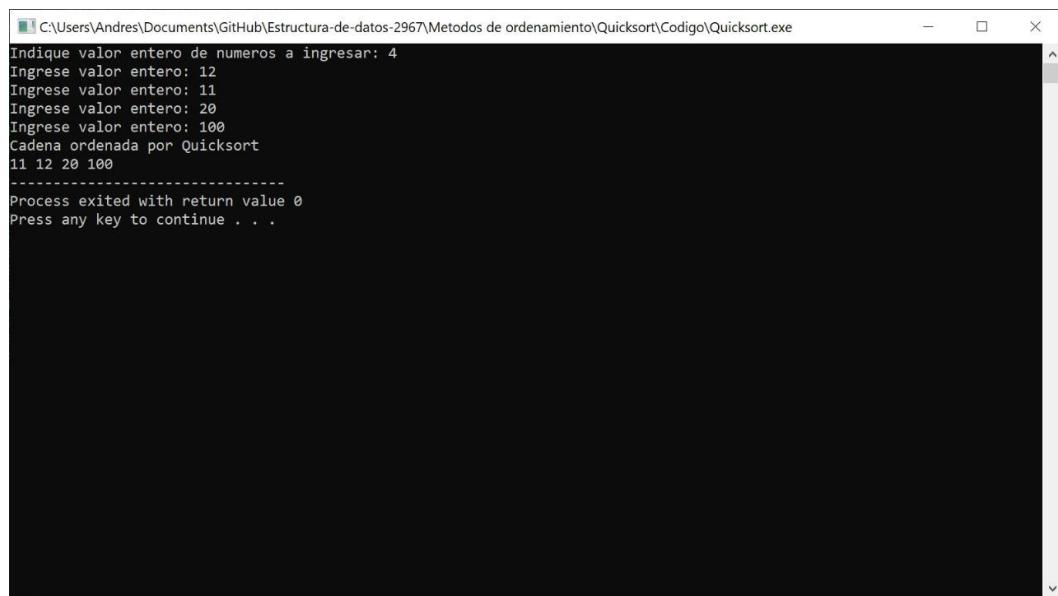


Figura 2: Indica la cadena de caracteres ordenados

Ordenamiento ShellSort

Descripción

El método se denomina así en honor de su inventor Donald Shell.

El algoritmo Shell es una mejora de la ordenación por inserción, donde se van comparando elementos distantes, al tiempo que se los intercambian si corresponde. A medida que se aumentan los pasos, el tamaño de los saltos disminuye; por esto mismo, es útil tanto como si los datos desordenados se encuentran cercanos, o lejanos.

Es bastante adecuado para ordenar listas de tamaño moderado, debido a que su velocidad es aceptable y su codificación es bastante sencilla. Su velocidad depende de la secuencia de valores con los cuales trabaja, ordenándolos.

Objetivo de la aplicación

Ordenar los elementos ingresados por el usuario mediante el uso del algoritmo shellsort.

Código de la aplicación

- A continuación mostraremos la librería “Ordenamiento.h” la cual contiene los prototipos de nuestro algoritmo de ordenamiento Shell sort

```
#include <iostream>
#include <stdlib.h>
#include "Ingreso.h"

using namespace std;

class Ordenamiento {
public:
    void ingresarDatos(int dim, int* arreglo);
    void imprimir(int* arreglo, int dim);
    void ordenarShell(int* arreglo, int dim);
    int* inicializarVector(int dim);
    Ordenamiento();
    ~Ordenamiento();

protected:
private:
};
```

- En la clase “Ordenamiento.cpp” se crear los métodos declarados en “Ordenamiento.h”.

```
#include "Ordenamiento.h"

void Ordenamiento::ingresarDatos(int dim, int* arreglo) {
    Ingreso ingreso;
```

```

        for (int i = 0; i < dim; i++) {
            *(arreglo + i) = atoi(ingreso.ingresar("Ingrese valor entero:
").c_str());
        }
    }

void Ordenamiento::imprimir(int* arreglo, int dim) {
    for (int i = 0; i < dim; i++)
        cout << *(arreglo + i) << " ";
}

int* Ordenamiento::inicializarVector(int dim) {
    int* a;
    a = (int*) calloc(dim, sizeof (int));
    return a;
}

void Ordenamiento::ordenarShell(int* arreglo, int dim) {
    for (int gap = dim / 2; gap > 0; gap /= 2) {
        for (int i = gap; i < dim; i += 1) {
            int temp = *(arreglo + i);
            int j;
            for (j = i; j >= gap && *(arreglo + (j - gap)) > temp; j -= gap)
                *(arreglo + j) = *(arreglo + (j - gap));
            *(arreglo + j) = temp;
        }
    }
}

Ordenamiento::Ordenamiento() {
}

Ordenamiento::~Ordenamiento() {
}

    • Este será nuestro main para ejecutar el aplicativo.

```

```

#include <iostream>
#include "Ordenamiento.cpp"

using namespace std;

int main(int argc, char** argv) {
    Ingreso ingreso;
    Ordenamiento ordenar;
    int* arreglo;
    int dimension = atoi(ingreso.ingresar("Indique valor entero de numeros
a ingresar: ").c_str());
    arreglo = ordenar.inicializarVector(dimension);
    ordenar.ingresarDatos(dimension, arreglo);
    cout << "\nArreglo antes de ordenar\n";
    ordenar.imprimir(arreglo, dimension);
    ordenar.ordenarShell(arreglo, dimension);
    cout << "\nArreglo ordenado con el metodo ShellSort\n";
    ordenar.imprimir(arreglo, dimension);
}

```

```
    free(arreglo);  
  
    return 0;  
}
```

Ejecución de la aplicación

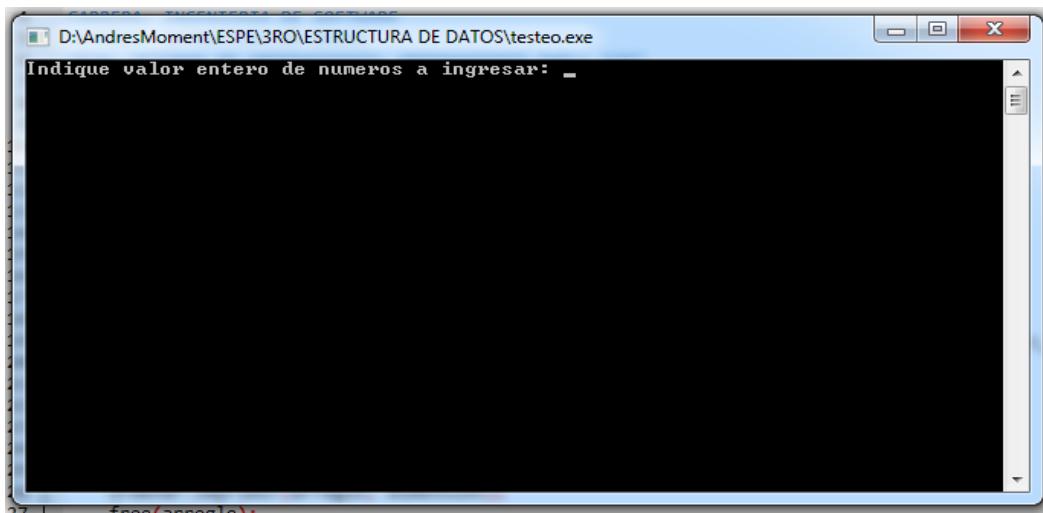


Figura 1. Ejecución de la aplicación

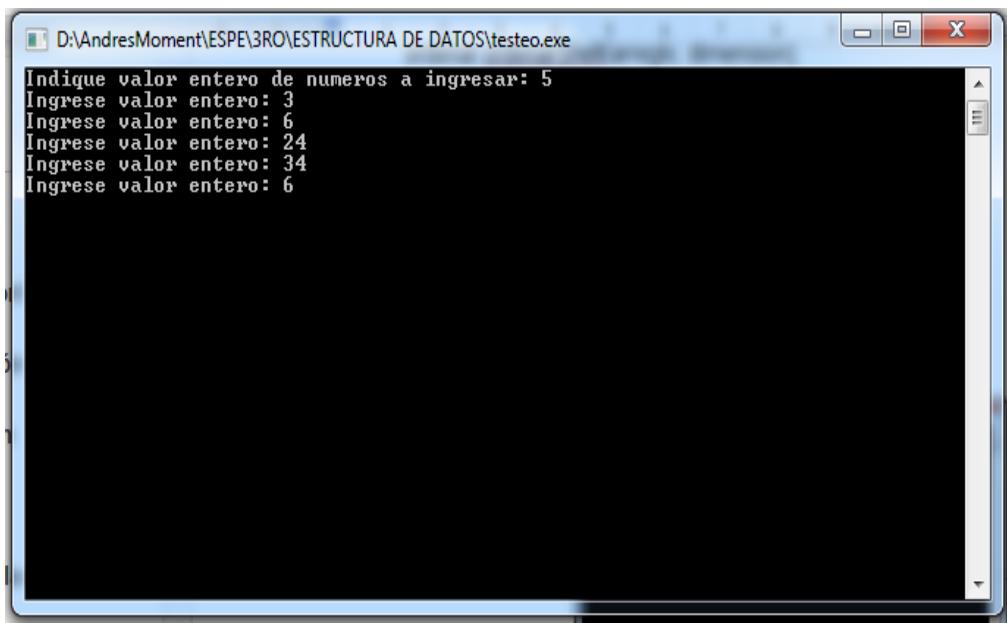


Figura 2. Inserción de Datos

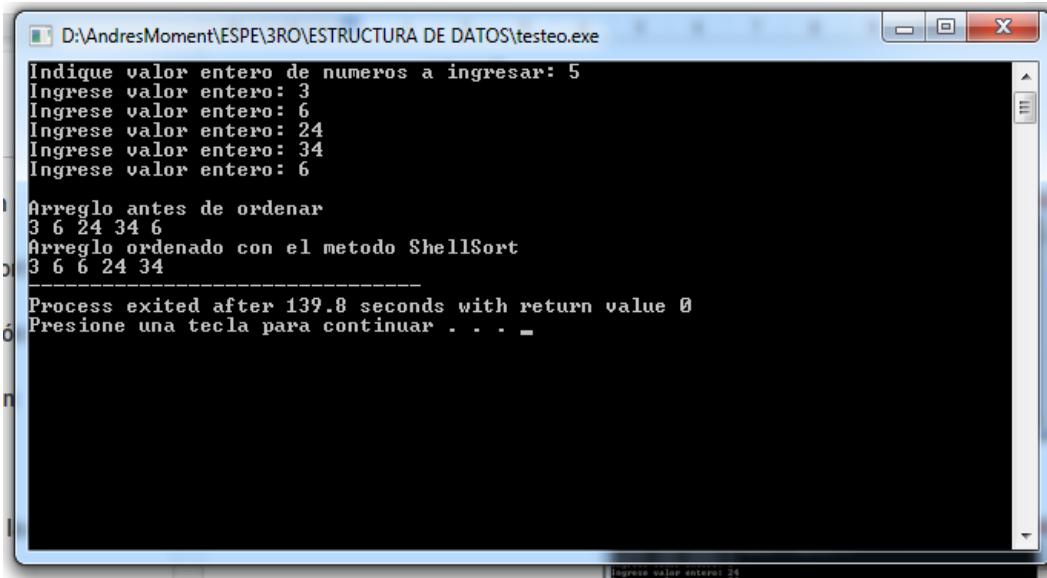


Figura 3. Ordenamiento Shell

Método de Ordenamiento Heapsort

Descripción

Es un método de ordenamiento basado con comparación, usa el Montículo o Heap como estructura de datos. Este método es más lento que otros métodos, pero es más eficaz en escenarios más rigurosos.

Objetivo de la aplicación

Ordenar los elementos de un vector utilizando el método heapsort

Código de la aplicación

- A continuación mostraremos el código necesario para la aplicación:

```
#include <iostream>
using namespace std;

class Ingreso {
public:
    string ingresar(string msg);
};

string Ingreso::ingresar(string msg) {
    string valor;
    cout << msg << endl;
```

```

    cin >> valor;
    return valor;
}

```

- Ahora para validar datos tenemos la siguiente librería:

```

#include <iostream>
using namespace std;

class Validacion
{
public:
    int validar(string val);
};

int Validacion::validar(string val) {
    int validez = 0;
    int punto = 0;
    int letras = 0;

    for (int i = 0; i < val.size(); i++)
    {
        if (!isdigit(val[i])) {
            letras++;
        }

        if (val[i] == '.') {
            punto++;
        }
    }

    if ((letras - punto) > 0 || punto > 1) {
        validez = 1;
    }
}

return validez;
}

```

- Ahora tenemos la clase que tiene el algoritmo de ordenamiento heapsort

```

#include <iostream>
#include <stdlib.h>
#include <time.h>

using namespace std;

class Ordenar{

private:
    int *matriz;
public:
    Ordenar();
    void setMatriz(int *);
    int* getMatriz();
    void inicializarMatriz(Ordenar,int);
}

```

```

        void heapify(Ordenar, int , int );
        void heapSort(Ordenar , int );
void imprimir(Ordenar , int );
void llenar(Ordenar ,int );

};

Ordenar::Ordenar() {

}

void Ordenar::setMatriz(int *v) {
    matriz = v;
}

int* Ordenar::getMatriz() {
    return matriz;
}

void Ordenar::inicializarMatriz(Ordenar v,int tam){
    v.matriz = (int*)malloc(tam*sizeof(int));
    v.setMatriz(v.matriz);
}

void Ordenar::heapify(Ordenar ord, int n, int i){
    // Encuentra más grande entre la raíz, el hijo izquierdo y el derecho
    int mayor = i;
    int izquierda = 2*i + 1;
    int derecha= 2*i + 2;

    if (izquierda< n && *(ord.matriz+izquierda)> *(ord.matriz+mayor))
        mayor = izquierda;

    if ((derecha<n)&& (*(ord.matriz+derecha)>*(ord.matriz+mayor)))
        mayor = derecha;

    // Intercambiar y continuar la heapificación si la raíz no es más grande
    if (mayor != i)
    {
        swap(*(ord.matriz+i), *(ord.matriz+mayor));
        heapify(ord, n, mayor);
    }
}

// Función principal para hacer la ordenación del montón
void Ordenar:: heapSort(Ordenar ord, int n){
    // Construir el montón máximo
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(ord, n, i);

    // tipo de pila
    for (int i=n-1; i>=0; i--)
    {
        swap(*(ord.matriz+0), *(ord.matriz+i));
}

```

```

        // Heapify elemento raíz para obtener el elemento más alto en la raíz
de nuevo
        heapify(ord, i, 0);
    }
}

void Ordenar:: imprimir (Ordenar ord, int n)
{
    for (int i=0; i<n; ++i){

        cout << *(ord.matriz+i) << " ";
        cout << " ";
    }
}

void Ordenar:: llenar(Ordenar ord,int tam){

srand(time(NULL));
for(int i=0;i<tam;i++){

    *(ord.matriz+i)=1+rand()%100;
    cout<<*(ord.matriz+i)<<" ";
}
cout<<endl;
}

```

- Finalmente tenemos el método principal (Main) en el que usaremos las librerías ya descritas anteriormente.

```

#include <iostream>
#include <sstream>
#include "Ingreso.h"
#include "Validacion.h"
#include "heapsort.h"

using namespace std;

int main(int argc, char** argv) {
    Ingreso ingreso;
    Validacion validacion;
    string numero;
    int val;

    int tam;
    int *matr;
    Ordenar ordenar;

    do{
        numero=ingreso.ingresar("Ingrese el tamaño de la matriz: ");
        val=validacion.validar(numero);
    }while(val==1);
    istringstream ( numero ) >> tam;

    matr= (int*)malloc(tam*sizeof(int));

```

```

ordenar.setMatriz(matr);
cout<<"Arreglo Desordenado" << endl;
cout<< endl;
ordenar.llenar(ordenar, tam);
cout<< endl;
ordenar.heapSort(ordenar, tam);
cout << "Arreglo Ordenado \n";
cout<< endl;
ordenar.imprimir(ordenar, tam);
cout<< endl;

system("PAUSE");
return 0;
}

```

Ejecución de la aplicación

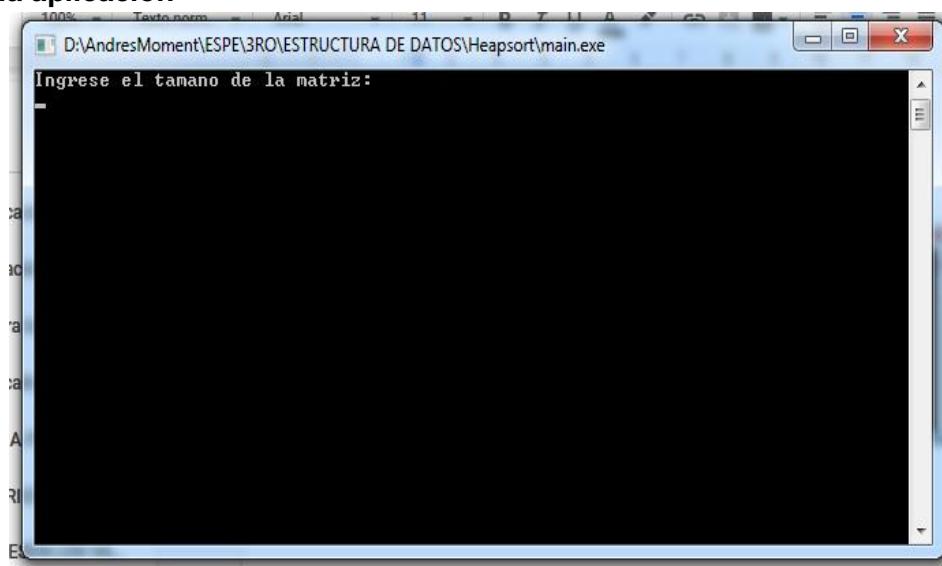


Figura 1: Ejecución de la aplicación

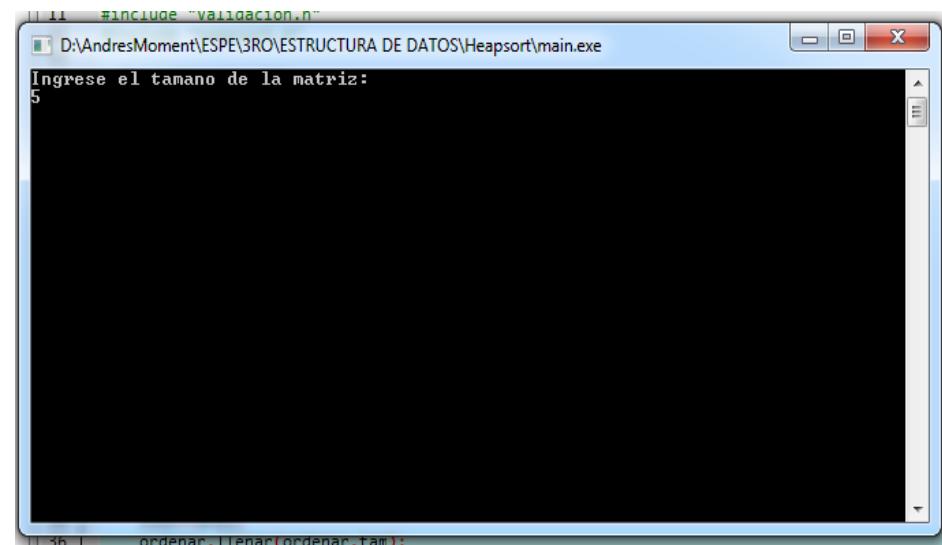


Figura 2: Ingreso de Datos

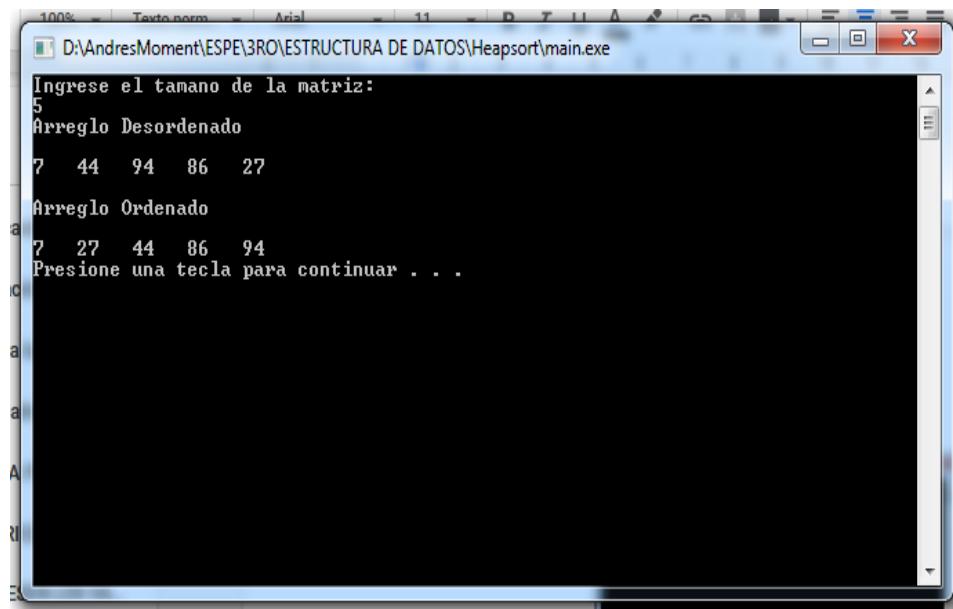


Figura 3: Ordenamiento Heapsort

Búsqueda Binaria

Descripción

Es un algoritmo de búsqueda que encuentra la posición de un valor en un array ordenado. Compara el valor con el elemento en el medio del array, si no son iguales, la mitad en la cual el valor no puede estar es eliminada y la búsqueda continúa en la mitad restante hasta que el valor se encuentre. La búsqueda binaria es computada en el peor de los casos en un tiempo logarítmico, realizando $O(\log n)$ comparaciones, donde n es el número de elementos del arreglo y \log es el logaritmo. La búsqueda binaria requiere solamente $O(1)$ en espacio, es decir, que el espacio requerido por el algoritmo es el mismo para cualquier cantidad de elementos en el array.

Código de la aplicación

- Aquí se encuentra el main de la aplicación que nos permite ejecutarla

```
#include <iostream>
#include "Busqueda.h"

using namespace std;

int main(int argc, char** argv) {
    Ingreso leer;
    Busqueda b;

    int dim, numero, pos;
```

```

int* vector;
int* vectorOrdenado;

vector = b.inicializarVector(dim);
vectorOrdenado = b.inicializarVector(dim);

cout << "Ingrese la dimension deseada del arreglo: " << endl;
dim = leer.ingresarEntero();

b.ingresoDatos(dim, vector);
vectorOrdenado = b.ordenamientoBurbuja(dim, vector);

cout << "Ingrese el elemento a buscar:" << endl;
numero = leer.ingresarEntero();
pos = b.busquedaBinaria(numero, 0, dim - 1, vectorOrdenado);
cout << "El elemento: " << "[" << numero << "]" << " se encuentra en la
posicion: " << "[" << pos << "]" << " del arreglo" << endl;

free(vector);
free(vectorOrdenado);
return 0;
}

```

- Se implementan los métodos generados en la clase “Busqueda.h”.

```

#include "Busqueda.h"

int* Busqueda::ordenamientoBurbuja(int dim, int* a) {
    int aux;
    for (int i = 0; i < dim - 1; i++) {
        for (int j = 0; j < dim - 1; j++) {
            if (*(a + j) > *(a + j + 1)) {
                aux = *(a + j);
                *(a + j) = *(a + j + 1);
                *(a + j + 1) = aux;
            }
        }
    }

    cout << "El arreglo ordenado por burbuja es: " << endl;
    for (int i = 0; i < dim; i++) {
        cout << "[" << *(a + i) << "] " << " ";
    }
    cout << endl;
    return a;
}

int* Busqueda::inicializarVector(int dim) {
    int* a;
    a = (int*) calloc(dim, sizeof (int));
    return a;
}

int* Busqueda::ingresoDatos(int dim, int* a) {

```

```

Ingreso leer;

int aux;

for (int i = 0; i < dim; i++) {
    cout << "Ingrese el elemento de la posicion: " << "[" << i << "]"
    endl;
    aux = leer.ingresarEntero();
    *(a + i) = aux;
}
}

int Busqueda::busquedaBinaria(int x, int inicio, int final, int *array) {
    int q;
    q = (inicio + final) / 2;
    if (x == *(array + q)) {
        return q;
    } else if (x > *(array + q)) {
        busquedaBinaria(x, q + 1, final, array);
    } else if (x < *(array + q)) {
        busquedaBinaria(x, inicio, q - 1, array);
    }
    else if (inicio >= final)
        return -1;
}

```

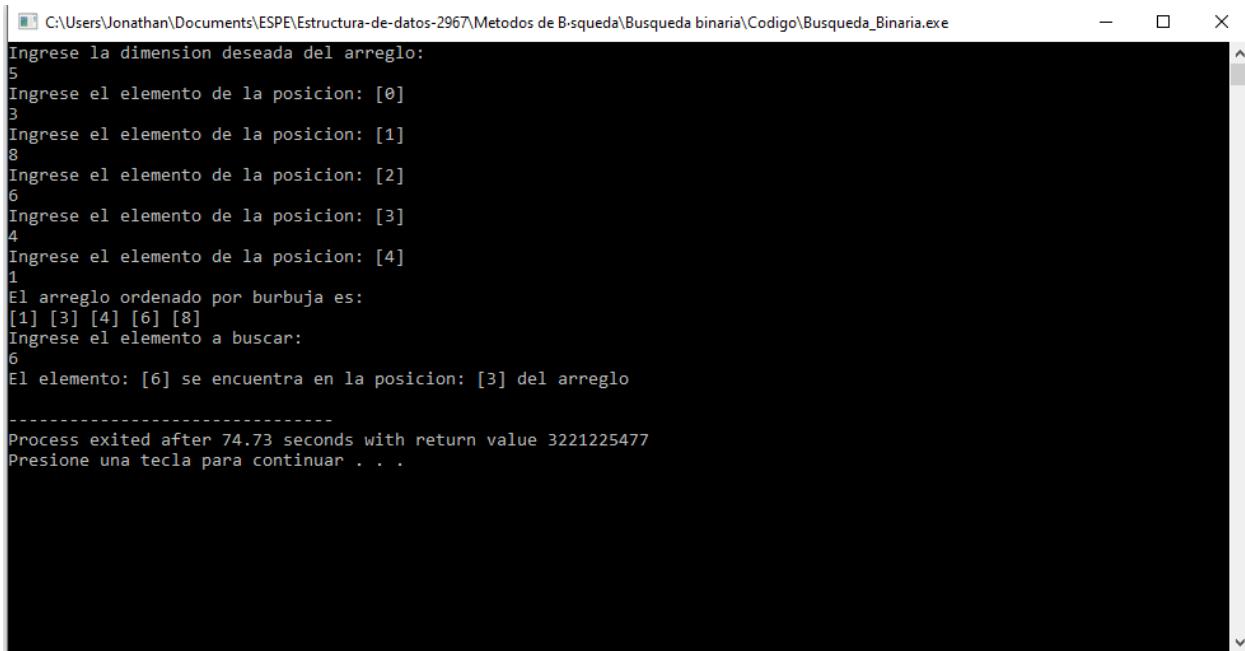
Ejecución de la aplicación

```

C:\Users\Jonathan\Documents\ESPE\Estructura-de-datos-296\Metodos de B-squeda\Busqueda binaria\Codigo\Busqueda_Binaria.exe
Ingrese la dimension deseada del arreglo:
5
Ingrese el elemento de la posicion: [0]
3
Ingrese el elemento de la posicion: [1]
8
Ingrese el elemento de la posicion: [2]
6
Ingrese el elemento de la posicion: [3]
4
Ingrese el elemento de la posicion: [4]
1
El arreglo ordenado por burbuja es:
[1] [3] [4] [6] [8]
Ingrese el elemento a buscar:

```

Figura : Ingreso de datos por parte del usuario



```
C:\Users\Jonathan\Documents\ESPE\Estructura-de-datos-2967\Metodos de Búsqueda\Busqueda binaria\Código\Busqueda_Binaria.exe
Ingrese la dimension deseada del arreglo:
5
Ingrese el elemento de la posicion: [0]
3
Ingrese el elemento de la posicion: [1]
8
Ingrese el elemento de la posicion: [2]
6
Ingrese el elemento de la posicion: [3]
4
Ingrese el elemento de la posicion: [4]
1
El arreglo ordenado por burbuja es:
[1] [3] [4] [6] [8]
Ingrese el elemento a buscar:
6
El elemento: [6] se encuentra en la posicion: [3] del arreglo
-----
Process exited after 74.73 seconds with return value 3221225477
Presione una tecla para continuar . . .
```

Figura : Resultado esperado para el usuario

Método de búsqueda secuencial

Objetivo del método secuencial

Revisar un arreglo definido, elemento por elemento hasta encontrar el elemento que está buscando o hasta llegar al final del arreglo.

Código de la aplicación:

- A continuación, se muestra el código fuente que se utilizó para construirá la búsqueda secuencial:

```
class Busqueda {
public:
    Busqueda(int* arrg, int n);
    void busquedaSecuencial(int clave);
int* getArreglo(void);
    void setArreglo(int* newArreglo);
    int getTamanio(void);
    void setTamanio(int newTamanio);
protected:
private:
    int* arreglo;
    int tamanio;
};
```

- Construcción de nuestros prototipos que fueron definidos en la clase Busqueda.h.

```
#include<iostream>
#include "Busqueda.h"
using namespace std;

Busqueda::Busqueda(int* arrg, int n) {
    this->arreglo = arrg;
    this->tamanio = n;
}
void Busqueda::busquedaSecuencial(int clave) {
    bool encontrado = true;
    //Buscar clave o dato en el arreglo.
    for (int j = 0; j < tamanio; j++) {
        if (*(arreglo + j) == clave) {
            cout << "Se encontró el " << clave << " en la posición [" << j +
1 << "] " << endl;
            encontrado = false;
        }
    }
    delete[] arreglo;

    if (encontrado)
        cout << "No se encontró el dato" << endl;
}
int* Busqueda::getArreglo(void) {
    return arreglo;
}
void Busqueda::setArreglo(int* newArreglo) {
    arreglo = newArreglo;
}
int Busqueda::getTamanio(void) {
    return tamanio;}

void Busqueda::setTamanio(int newTamanio) {
    tamanio = newTamanio;
}
```

- A continuación se muestra el main del programa

```
#include<iostream>
#include<stdlib.h>
#include <conio.h>
#include <iostream>
#include "Busqueda.h"
#include "Ingreso.h"
using namespace std;
int main() {

    int clave, tamanio, num;
    int * arreglo;
    Ingreso ingresar;
    cout << "Busqueda Secuencial" << endl;
    tamanio = ingresar.ingresarInt("Ingrese la dimensión del arreglo: ");
```

```

cout << endl;

arreglo = (int *) malloc(tamanio * sizeof (int));

//Ingreso de datos al arreglo
for (int i = 0; i < tamanio; i++) {
    cout << "\nIngrese dato [" << i + 1 << "]: ";
    (*arreglo + i)) = ingresar.ingresarInt("");
}

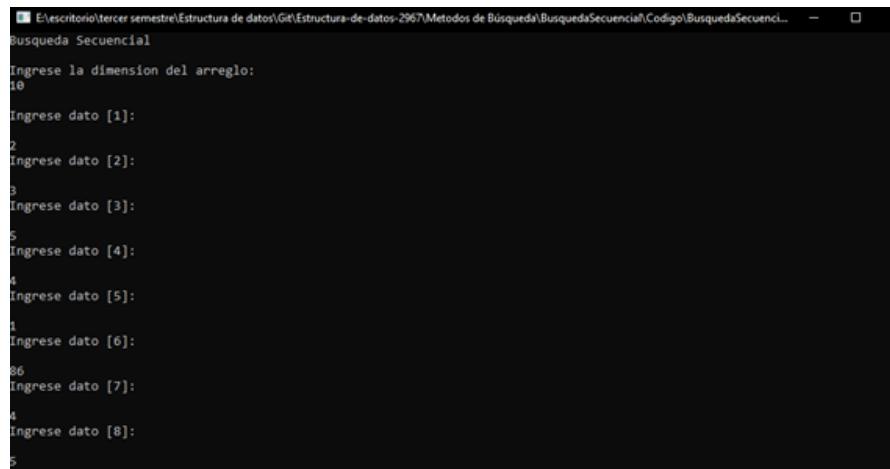
clave = ingresar.ingresarInt("Ingrese el numero que desea buscar: ");
cout << endl;

Busqueda buscar = Busqueda(arreglo, tamanio);
buscar.busquedaSecuencial(clave);

system("pause");
return 0;
}

```

Ejecución del programa:



```

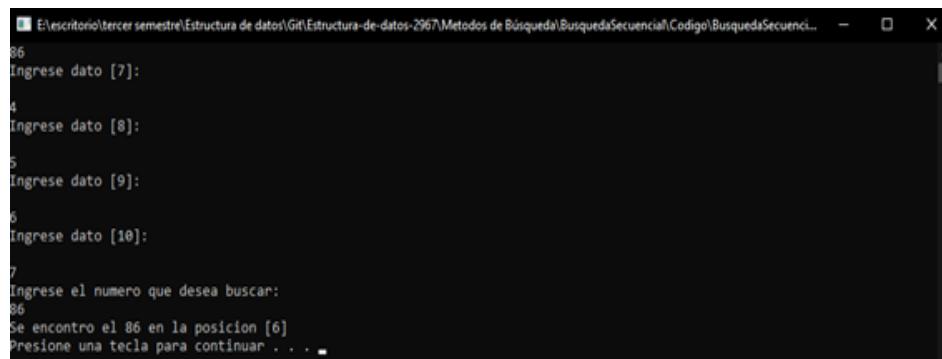
E:\Escritorio\tercer semestre\Estructura de datos\Git\Estructura-de-datos-2967\Metodos de Búsqueda\BusquedaSecuencial\Código\BusquedaSecuenc...
Busqueda Secuencial

Ingrese la dimension del arreglo:
10

Ingrese dato [1]:
2
Ingrese dato [2]:
3
Ingrese dato [3]:
5
Ingrese dato [4]:
6
Ingrese dato [5]:
86
Ingrese dato [6]:
4
Ingrese dato [7]:
5
Ingrese dato [8]:
6
Ingrese dato [9]:
5

```

Figura 1. ejecución de la aplicación y ingreso de datos.



```

E:\Escritorio\tercer semestre\Estructura de datos\Git\Estructura-de-datos-2967\Metodos de Búsqueda\BusquedaSecuencial\Código\BusquedaSecuenc...
86
Ingrese dato [7]:
4
Ingrese dato [8]:
5
Ingrese dato [9]:
6
Ingrese dato [10]:
7
Ingrese el numero que desea buscar:
86
Se encontro el 86 en la posicion [6]
Presione una tecla para continuar... .

```

Figura 2. Ingreso de un número que se desea buscar

Puzzle deslizante

Objetivo del programa:

Deslizar los datos de uno en uno, colocarlos ordenadamente en una distribución establecida de antemano (ordenándolos del 1 al número de piezas, alfabéticamente si se utilizan letras, restableciendo el orden que se espera).

Código:

- A Continuación se presentan las clases y librería necesarias para el correcto funcionamiento del puzzle deslizante:

```
#include <iostream>
#include <ctype.h>
#include <string.h>

using namespace std;

class Validacion {
public:
    bool validar(string, int);
};

bool Validacion::validar(string entrada, int tipo) {

    if (tipo == 3) {
        return false;
    }
    int contador = 0;
    try {
        for (int i = 0; i < entrada.length(); i++) {
            if (isalpha(entrada[i])) {
                throw 1;
            }
            if (!isdigit(entrada[i]) && tipo == 1) {
                throw 1;
            }
            if (entrada[i] == '.') {
                contador++;
            }
            if ((isdigit(entrada[i]) == 0 && entrada[i] != '.') || (contador
> 1)) {
                throw 1;
            }
        }
    } catch (int e) {
        return true;
    }
}
```

```

    return false;
}

```

- Metodo que funciona para definir el tablero del puzzle deslizante.

```

class Tablero {
public:
    Tablero(int filas, int columnas);
    ~Tablero();
    void inicializar(int filas, int columnas);
    void mover(int filaActual, int columnaActual, int filaNueva, int
columnaNueva);
    void llenar(int filas, int columnas);
    void desordenar(int filas, int columnas);
    void imprimir(int filas, int columnas);
    bool verificar(int filas, int columnas);
    int buscarColumna(int filas, int columnas);
    int buscarFila(int filas, int columnas);
    int** getMatriz(void);
    void setMatriz(int** newMatriz);

protected:
private:
    int** matriz;
};

```

- La clase Ingreso.h sirve para validar los ingresos por teclado del usuario

```

#include <iostream>
#include <string>
#include "Validacion.h"

using namespace std;

class Ingreso {

public:
    string leer(string,int);
};

string Ingreso::leer(string mensaje,int tipo) {
    Validacion validacion;
    string entrada;
    cout << mensaje << endl;
    getline(cin,entrada);
    while (validacion.validar(entrada, tipo)) {
        cout << "Valor no valido reingrese" << endl;
        cin >> entrada;
    }
    return entrada;
}

```

- Implementación de los prototipos declarados en tablero.h

```

#include "Tablero.h"
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;

void Tablero::inicializar(int filas, int columnas) {
    matriz = (int**) malloc(filas * sizeof (int));
    srand(time(NULL));

    for (int i = 0; i < columnas; i++) {
        (*matriz + i)) = (int*) malloc(columnas * sizeof (int));
    }
}

void Tablero::mover(int filaActual, int columnaActual, int filaNueva, int
columnaNueva) {
    int aux;
    aux = *(*(matriz + filaActual) + columnaActual);
    *(*(matriz + filaActual) + columnaActual) = *(*(matriz + filaNueva) +
columnaNueva);
    *(*(matriz + filaNueva) + columnaNueva) = aux;
}

void Tablero::llenar(int filas, int columnas) {
    int numeros = 1;
    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            *(*(matriz + i) + j) = numeros;
            numeros++;
        }
    }
    *(*(matriz + filas - 1) + columnas - 1) = -1;
}

void Tablero::desordenar(int filas, int columnas) {
    int filaNueva;
    int columnaNueva;
    int auxiliar;
    srand(time(NULL));
    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            filaNueva = rand() % (filas);
            columnaNueva = rand() % (columnas);
            auxiliar = (*(*(matriz + i) + j));
            *(*(matriz + i) + j) = *(*(matriz + filaNueva) + columnaNueva);
            (*(*(matriz + filaNueva) + columnaNueva)) = auxiliar;
        }
    }
}

int** Tablero::getMatriz(void) {
    return matriz;
}

void Tablero::setMatriz(int** newMatriz) {
    matriz = newMatriz;
}

```

```

}

Tablero::Tablero(int filas, int columnas) {
    inicializar(filas, columnas);
    llenar(filas, columnas);
    desordenar(filas, columnas);
}

Tablero::~Tablero() {
    free(matriz);
}

bool Tablero::verificar(int filas, int columnas) {
    int numero = 1;
    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            if (*(*(matriz + i) + j) != numero) {
                return false;
            }
            numero++;
        }
    }
    return true;
}
int Tablero::buscarFila(int filas, int columnas) {

    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            if (*(*(matriz + i) + j) == -1) {
                return i;
            }
        }
    }
}

int Tablero::buscarColumna(int filas, int columnas) {
    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            if (*(*(matriz + i) + j) == -1) {
                return j;
            }
        }
    }
}

void Tablero::imprimir(int filas, int columnas) {
    system("cls");
    cout << endl << endl;
    for (int i = 0; i < filas; i++) {
        cout << "\t\t";
        for (int j = 0; j < columnas; j++) {
            if ((*(*(matriz + i) + j)) != -1) {
                cout << *(*(matriz + i) + j) << "\t";
            } else {
                cout << "  \t";
            }
        }
        cout << endl;
    }
}

```

```
    }
}
```

- Se muestra el main que sirve para ejecutar el aplicativo

```
#include<iostream>
#include<stdlib.h>
#include <conio.h>
#include <iostream>
#include "Busqueda.h"
#include "Ingreso.h"

using namespace std;

int main() {

    int clave, tamano, num;
    int * arreglo;
    Ingreso ingresar;
    cout << "Busqueda Secuencial" << endl;
    tamano = ingresar.ingresarInt("Ingrese la dimension del arreglo: ");
    cout << endl;

    arreglo = (int *) malloc(tamano * sizeof (int));

    //Ingreso de datos al arreglo
    for (int i = 0; i < tamano; i++) {
        cout << "\nIngrese dato [" << i + 1 << "]: ";
        (*arreglo + i)) = ingresar.ingresarInt("");
    }

    clave = ingresar.ingresarInt("Ingrese el numero que desea buscar: ");
    cout << endl;

    Busqueda buscar = Busqueda(arreglo, tamano);
    buscar.busquedaSecuencial(clave);

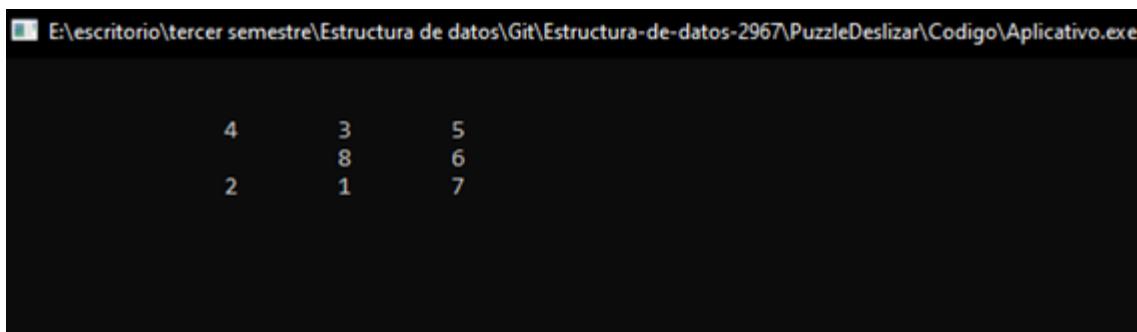
    system("pause");
    return 0;
}
```

Ejecución de programa:



```
E:\escritorio\tercer semestre\Estructura de datos\Git\Estructura-de-datos-2967\PuzzleDeslizar\Codigo\Aplicativo.exe
Ingrese el numero de filas
3
Ingrese el numero de columnas
3
```

Figura 1. Ingreso de la dimensión de la matriz.



The screenshot shows a terminal window with the path E:\escritorio\tercer semestre\Estructura de datos\Git\Estructura-de-datos-2967\PuzzleDeslizar\Codigo\Aplicativo.exe at the top. Below the path, there is a 3x3 grid of numbers:

4	3	5
8		6
2	1	7

Figura 2. Se genera la matriz con los respectivos números

Contar Vocales

Objetivo

A partir de una cadena ingresada por el usuario u al obtener el dato del nombre, el programa obtendrá la cantidad de vocales que se encuentran en dicha cadena.

Código:

- A continuación se presentan las clases y librería necesarias para el correcto funcionamiento:

```
#include <stdio.h>
#include <stdlib.h>
#include<string.h>
#include "Ingreso.h"
#include "ContadorVocales.h"
int main() {
    //declaracion de las clases Ingreso y lectura
    Ingreso lectura;
    ContadorVocales calcular;
    int contador;
    char * cadena=lectura.leer("Ingrese oracion: ");
    strupr(cadena); //conversion de de cadena de minuscula a mayuscula
    contador=calcular.contarVocales(cadena);//llamando al metodo de la clase
    ContadorVocales
    printf("\n\nHay %i Vocales \n\n", contador);
    free(cadena); //liberando la memoria
    system("pause");
    return 0;
}
```

- Método cuyo propósito es implementar los métodos generados en ContadorVocales.h

```
#include "ContadorVocales.h"
int ContadorVocales::contarVocales(char * cadena)
{
```

```
if(*cadena=='\0')
{
    return 0;
}
else
{
    switch(*cadena)
    {
    case 'A':
    case 'E':
    case 'I':
    case 'O':
    case 'U':
        cadena++;
        return 1+contarVocales(cadena);
    }
    cadena++;
    return contarVocales(cadena);
}
}
```

Ejecución de programa:

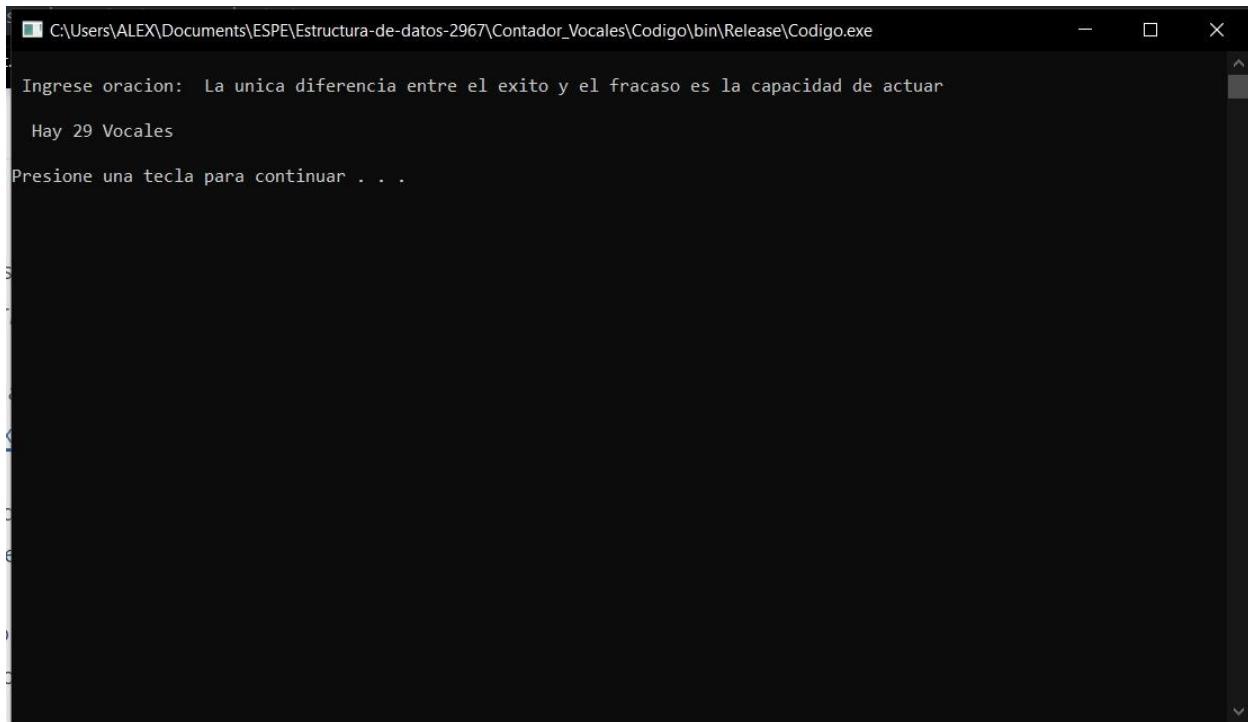


Figura . Ejecución de código

Corrección de la prueba

Objetivo:

Obtener el resultado de la Multiplicación de las matrices según el exponente que se ingrese.

Código:

- Implementación de las diferentes clases y funciones para el correcto funcionamiento de la multiplicación de matrices.

```
class Matriz
{
public:
    int** getmatriz(void);
    void setmatriz(int** newMatriz);
    int getfilas(void);
    void setfilas(int newFilas);
    int getexp(void);
    void setexp(int newExp);
    int getcolumnas(void);
    void setcolumnas(int newColumnas);
    Matriz();
    void crear(void);
    void llenar(void);
    void imprimir(void);
    void multiplicar(int exponente);

protected:
private:
    int filas;
    int columnas;
    int exp;
    int** matriz;
};
```

- Para validar los datos que ingresa en la matriz.

```
#include <iostream>
#include <string>
#include <stdlib.h>
using namespace std;
class Ingreso{
public:
    float ingresarFloat(char*);
    bool validarFloat(string);
    int ingresarInt(char*);
    bool validarInt(string);
```

```

    string ingresarString(char*);
    bool validarString(string);
    string ingresar10Digitos(char*);
    bool validar10Digitos(string);
};

float Ingreso::ingresarFloat(char* msg){
    string flotante;
    bool flag;
    do{
        try{
            cout<<msg;
            getline(cin,flotante);
            flag = validarFloat(flotante);
            if(flag){
                throw flotante;
            }
        }catch(string e){
            cout<<"Dato invalido "<<e<<endl;
        }
    }while(flag);

    return atof(flotante.c_str());
}
bool Ingreso::validarFloat(string valor){
    bool flag;
    for(int i = 0; i<valor.length(); i++){
        if(isdigit(valor[i])){
            flag = false;
        }else{
            if(valor[i]== '.'){
                flag = false;
            }else{
                flag = true;
                break;
            }
        }
    }
    return flag;
}
int Ingreso::ingresarInt(char* msg){
    string numero;
    bool flag;
    do{
        try{
            cout<<msg;
            getline(cin,numero);
            flag = validarInt(numero);
            if(flag){
                throw numero;
            }
        }catch(string e){
            cout<<"Dato invalido "<<e<<endl;
        }
    }while(flag);

    return atoi(numero.c_str());
}

```

```

bool Ingreso::validarInt(string valor){
    bool flag;
    for(int i = 0; i<valor.length(); i++){
        if(isdigit(valor[i])){
            flag = false;
        }else{
            return true;
        }
    }
    return flag;
}
string Ingreso::ingresarString(char* msg){
    string dato;
    bool flag;
    do{
        try{
            cout<<msg;
            getline(cin,dato);
            flag = validarString(dato);
            if(flag){
                throw dato;
            }
        }catch(string e){
            cout<<"Dato invalido "<<e<<endl;
        }
    }while(flag);
    return dato;
}
bool Ingreso::validarString(string valor){
    bool flag;
    for(int i = 0; i<valor.length(); i++){
        if(isalpha(valor[i]) || valor[i] == 32){
            flag = false;
        }else{
            return true;
        }
    }
    return flag;
}
string Ingreso::ingresar10Digitos(char* msg){
    string dato;
    bool flag;
    do{
        try{
            cout<<msg;
            getline(cin,dato);
            flag = validar10Digitos(dato);
            if(flag){
                throw dato;
            }
        }catch(string e){
            cout<<"Dato invalido "<<e<<endl;
        }
    }while(flag);
    return dato;
}
bool Ingreso::validar10Digitos(string valor){

```

```

    bool flag;
    int cont = 0;
    for(int i = 0; i<valor.length(); i++){
        if(isdigit(valor[i])){
            cont++;
            if(cont == 10){
                flag = false;
            }else{
                flag = true;
            }
        }else{
            return true;
        }
    }
    return flag;
}

```

- Construcción de las funciones declaradas en la matriz.h para utilizar en el main.

```

#include "Matriz.h"
#include <stdio.h>
#include <iostream>
#include <time.h>
#include <stdlib.h>

int** Matriz::getmatriz(void)
{
    return matriz;
}

void Matriz::setmatriz(int** newMatriz)
{
    matriz = newMatriz;
}

int Matriz::getfilas(void)
{
    return filas;
}

void Matriz::setfilas(int newFilas)
{
    filas = newFilas;
}

int Matriz::getexp(void)
{
    return filas;
}

void Matriz::setexp(int newExp)
{
    exp = newExp;
}

```

```

}

int Matriz::getcolumnas(void)
{
    return columnas;
}
void Matriz::setcolumnas(int newColumnas)
{
    columnas = newColumnas;
}
Matriz::Matriz()
{
}
void Matriz::crear(void)
{
    matriz =(int **)calloc(filas,sizeof(int *)*filas);
    for(int j=0;j<columnas;j++)
        *(matriz+j)=(int *)calloc(columnas,sizeof(int*)*columnas);
}
void Matriz::llenar(void)
{
    srand(time(NULL));
    for(int i=0;i<filas;i++)
        for(int j=0;j<columnas;j++) {
            *(*(matriz+i)+j)=1;//1+rand()%9;
        }
}

void Matriz::imprimir(void)
{
    for(int i=0;i<filas;i++) {
        for(int j=0;j<columnas;j++)
        {
            printf("%5d",*(*(matriz+i)+j));

        }
        printf("\n");
    }
}
void Matriz::multiplicar(int exponente)
{
    int** mT1;

    mT1 =(int **)calloc(filas,sizeof(int *)*filas);
    for(int j=0;j<columnas;j++){
        *(mT1+j)=(int *)calloc(columnas,sizeof(int*)*columnas);
    }

    for(int i=0;i<filas;i++)
        for(int j=0;j<columnas;j++) {
            *(*(mT1+i)+j)=*(*(matriz+i)+j);
        }

    for(int k=1;k<exponente;k++) {
        int ** mT3;
        mT3 =(int **)calloc(filas,sizeof(int *)*filas);
    }
}

```

```

        for(int j=0;j<columnas;j++)
            *(mT3+j)=(int *)calloc(columnas,sizeof(int*)*columnas);

            for(int i=0;i<filas;i++) {
                for(int j=0;j<filas;j++) {
                    for(int h=0;h<filas;h++) {

                        *((mT3+i)+j)=*((mT3+i)+j)+(*((mT1+i)+h))*(*((matriz+h)+j));
                    }
                }
            matriz=mT3;
        }
    }
}

```

- Función principal en la se declara las clases para poder utilizar todas las funciones para poder hacer la multiplicación de matrices.

```

#include <stdio.h>
#include <iostream>
#include <time.h>
#include <stdlib.h>
#include "Matriz.cpp"
#include "ingreso.h"
using namespace std;
int main(){
    main:
    int row,col,exp;
    Matriz p;
    Ingreso lee;
    row= lee.ingresarInt("Ingrese # filas: ");
    p.setfilas(row);
    col= lee.ingresarInt("Ingrese # columnas: ");
    p.setfilas(col);
    cout<<endl;

    if(row>10&&row>10||row!=col){
        cout<<"la matriz tiene q ser cuadrada y su dimension menor o igual
q 10\n"<

```

```

        system("pause");
        system("cls");
        goto main;
        return 0;
    }
}

```

Ejecución:

```

C:\Users\naula\Desktop\presentar\corrección prueba\pruebaarecu.exe
ingrese numero de filas :4
ingrese numero de columnas :4
ingrese el exponente3
1   1   1   1
1   1   1   1
1   1   1   1
1   1   1   1
16   16   16   16
16   16   16   16
16   16   16   16
16   16   16   16

Process exited after 4.437 seconds with return value 0
Presione una tecla para continuar . .

```

Figura 1. Ejecución de código

Ordenamiento por Inserción

Descripción

Es una manera muy natural de ordenar para un ser humano, y puede usarse fácilmente para ordenar un mazo de cartas numeradas en forma arbitraria. Requiere $O(n^2)$ operaciones para ordenar una lista de n elementos.

Inicialmente se tiene un solo elemento, que obviamente es un conjunto ordenado. Después, cuando hay k elementos ordenados de menor a mayor, se toma el elemento $k+1$ y se compara con todos los elementos ya ordenados, deteniéndose cuando se encuentra un elemento menor (todos los elementos mayores han sido desplazados una posición a la derecha) o cuando ya no se encuentran elementos (todos los elementos fueron desplazados y este es el más pequeño). En este punto se *inserta* el elemento $k+1$ debiendo desplazarse los demás elementos.

Objetivo de la aplicación

Ingreso de una serie de números que decida el usuario para que la aplicación devuelva estos números ordenados.

Código de la aplicación

- La librería “Ingreso.h” es la que permite al usuario dar el valor de las dimensiones del cubo mágico.

```
#include <iostream>
#include <stdlib.h>
#include <string.h>

using namespace std;

class Ingreso {
public:
    int ingresarEntero();
    double ingresarDouble();
    float ingresarFloat();
    string ingresarLetra();
    bool validar(string);
    bool validarEntero(string);
    bool validarLetra(string);
};

int Ingreso::ingresarEntero() {
    string numero;
    bool valido = false;
    while (!valido) {
        try {
            getline(cin, numero);
            valido = validarEntero(numero);
            if (!valido) {
                throw numero;
            }
        } catch (string e) {
            cout << "El numero " << e << " no es valido" << endl;
        }
    }
    return atoi(numero.c_str());
}

double Ingreso::ingresarDouble() {
    string numero;
    bool valido = false;
    while (!valido) {
        try {
            getline(cin, numero);
            valido = validar(numero);
            if (!valido) {
                throw numero;
            }
        } catch (string e) {
            cout << "El numero " << e << " no es valido" << endl;
        }
    }
}
```

```

        }
    }
    return atof(numero.c_str());
}

float Ingreso::ingresarFloat() {
    string numero;
    bool valido = false;
    while (!valido) {
        try {
            getline(cin, numero);
            valido = validar(numero);
            if (!valido) {
                throw numero;
            }
        } catch (string e) {
            cout << "El numero " << e << " no es valido" << endl;
        }
    }
    return atof(numero.c_str());
}

string Ingreso::ingresarLetra() {
    string palabra;
    bool valido = false;
    while (!valido) {
        try {
            getline(cin, palabra);
            valido = validarLetra(palabra);
            if (!valido) {
                throw palabra;
            }
        } catch (string e) {
            cout << "La palabra " << e << " no es valida" << endl;
        }
    }
    return palabra;
}

bool Ingreso::validar(string numero) {
    int inicio = 0;
    if (numero.length() == 0) {
        return 0;
    }
    if (numero[0] == '+' || numero[0] == '-') {
        inicio = 1;
        if (numero.length() == 1) {
            return 0;
        }
    }
    for (int i = inicio; i < numero.length(); i++) {
        if (!isdigit(numero[i]) && numero[i] != '.') {
            return 0;
        }
    }
    return 1;
}

```

```

bool Ingreso::validarEntero(string numero) {
    int inicio = 0;
    if (numero.length() == 0) {
        return 0;
    }
    if (numero[0] == '+' || numero[0] == '-') {
        inicio = 1;
        if (numero.length() == 1) {
            return 0;
        }
    }
    for (int i = inicio; i < numero.length(); i++) {
        if (!isdigit(numero[i])) {
            return 0;
        }
    }
    return 1;
}

bool Ingreso::validarLetra(string palabra) {
    char c;
    for (int i = 0; i < palabra.size(); i++) {
        c = palabra[i];
        if (isalpha(c) == 0) {
            if (isspace(c) == 0) {
                return 0;
            }
        }
    }
    return 1;
}

```

- La librería “recursivo.h” es la que ordena los números que ingresamos

```

#include <stdio.h>
#include <stdlib.h>

class Insercion {
private:
    int *vector;
public:
    int* inicializar(int);
    int* encerar(int, int*);
    int* ingresaNumeros(int, int*);
    void whileRecursivo(int, int, int*);
    void insercionRecursivo(int, int*);
    void mostrar(int, int*);
};

int* Insercion::inicializar(int tamanio) {
    vector = (int*) malloc(tamanio * sizeof (int));
    return vector;
}

```

```

int* Insercion::encerar(int tamano, int *vector) {
    for (int i = 0; i < tamano; i++) {
        *(vector + i) = 0;
    }
    return vector;
}

int* Insercion::ingresaNumeros(int tamano, int *vector) {
    vector = inicializar(tamano);
    vector = encerar(tamano, vector);
    for (int i = 0; i < tamano; i++) {
        scanf("%d", vector + i);
    }
    return vector;
}

void Insercion::whileRecursivo(int fin, int j, int *vector) {
    while (j >= 0 && *(vector + j) > fin) {
        *(vector + (j + 1)) = *(vector + j);
        j--;
    }
    *(vector + (j + 1)) = fin;
}

void Insercion::insercionRecursivo(int n, int* vector) {
    if (n <= 1) {
        return;
    }

    insercionRecursivo(n - 1, vector);

    int fin = *(vector + (n - 1));
    int j = n - 2;

    whileRecursivo(fin, j, vector);
}

void Insercion::mostrar(int tamano, int *vector) {
    for (int i = 0; i < tamano; i++) {
        printf("%d ", *(vector + i));
    }
}

```

- A continuacion se muestra el main del programa

```

#include <stdio.h>
#include "recursivo.h"
#include "ingreso.h"

int main() {
    Ingreso leer;
    Insercion insercion;
    int *vector, tamano;
    printf("Ingrese el tamano del vector: ");

```

```
tamanio = leer.ingresarEntero();
printf("Ingrese los datos del vector: ");
vector = insercion.ingresaNumeros(tamanio, vector);
insercion.mostrar(tamanio, vector);
printf("\nEl vector ordenado es: \n");
insercion.insercionRecursivo(tamanio, vector);
insercion.mostrar(tamanio, vector);
}
```

Ejecución del programa

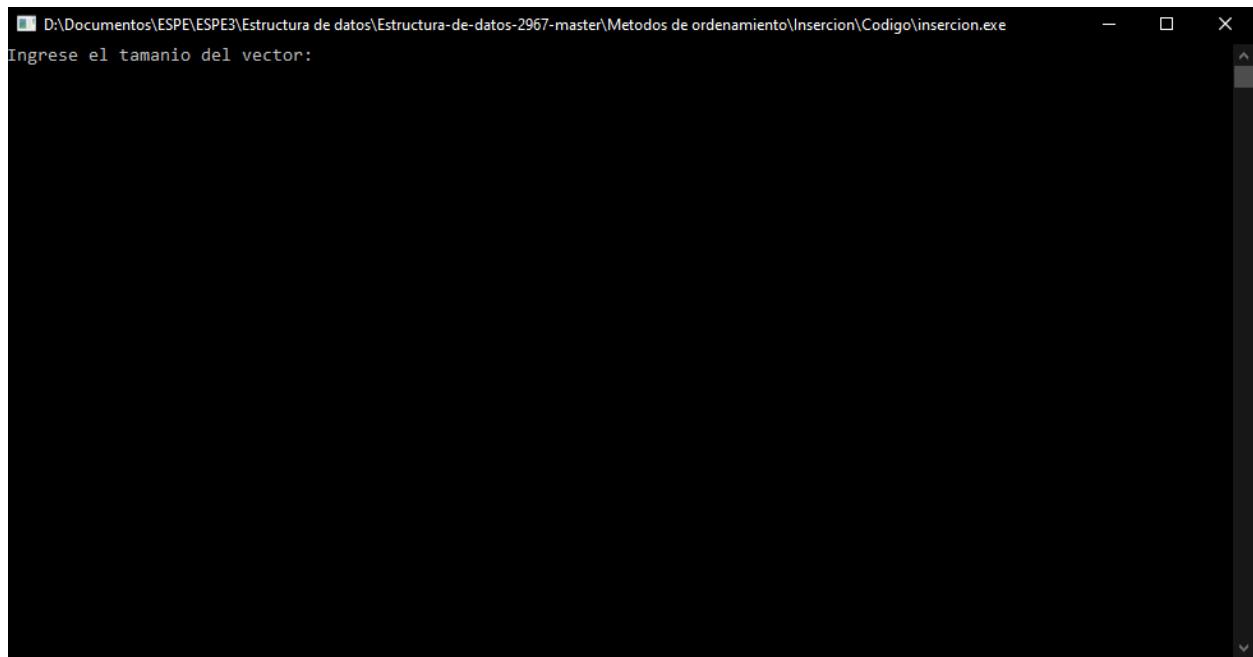
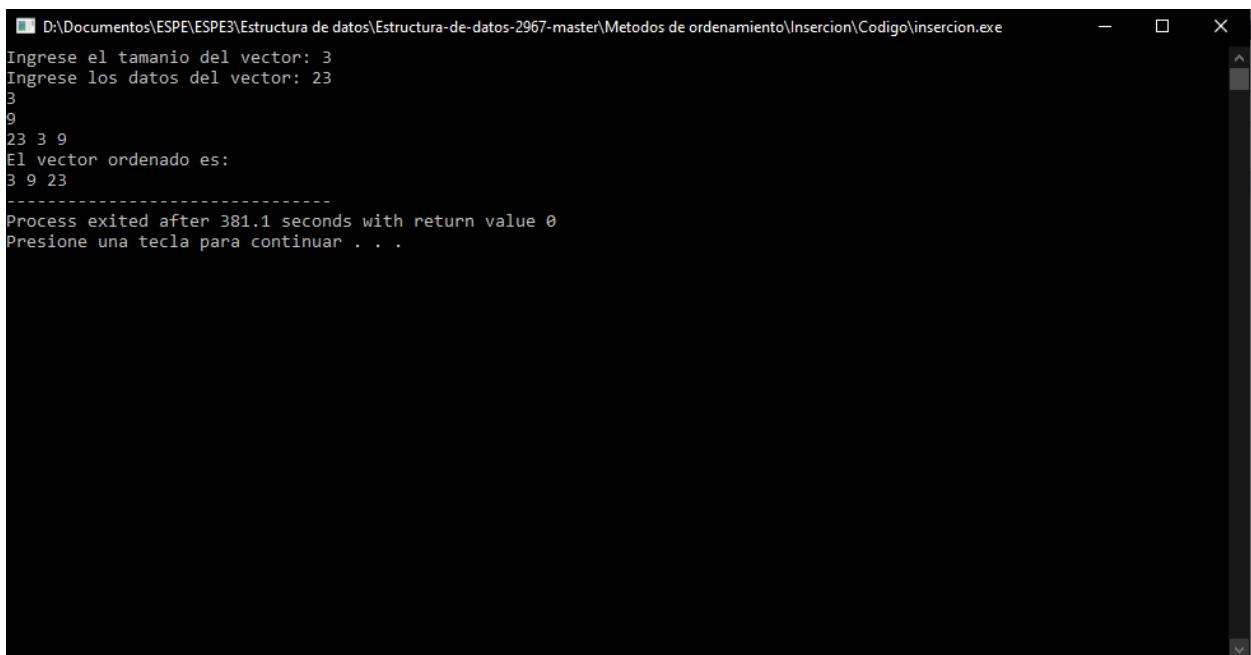


Figura 1. Ingreso del tamaño del vector



```
D:\Documentos\ESPE\ESPE3\Estructura de datos\Estructura-de-datos-2967-master\Metodos de ordenamiento\Insercion\Codigo\insercion.exe
Ingrese el tamano del vector: 3
Ingrese los datos del vector: 23
3
9
```

Figura 2. Ingreso de los números para ordenar



```
D:\Documentos\ESPE\ESPE3\Estructura de datos\Estructura-de-datos-2967-master\Metodos de ordenamiento\Insercion\Codigo\insercion.exe
Ingrese el tamano del vector: 3
Ingrese los datos del vector: 23
3
9
23 3 9
El vector ordenado es:
3 9 23
-----
Process exited after 381.1 seconds with return value 0
Presione una tecla para continuar . . .
```

Figura 3. Números ordenados

Ocho reinas

Objetivo:

Generar diferentes tipos de soluciones posibles para las ocho o menos reinas en un tablero de ajedrez.

Código del programa:

- se generan los prototipos de funciones, necesarias para el correcto funcionamiento de la aplicación.

```
#include <iostream>
#include <fstream>

using namespace std;

class Reina
{
public:
    Reina(int n1);
    ~Reina();
    void solucion(int x, int y, int n1);
    void bloquear(int x, int y);
    void quitarRelleno(int x, int y);
    void mostrar();
    void solucionReinas();

protected:
private:
    bool** validar;
    char** tablero;
    fstream enter;
    int contador;
    int n;
};
```

- Construir los prototipos que fueron declarados en Reina.h

```
#include "Reina.h"
#include <iostream>
#include <fstream>
Reina::Reina(int n1)
{
    n=n1;
    validar = new bool *[n];
    tablero= new char *[n];
    for(int i=0;i<n;i++){
        validar[i]=new bool [n];
        tablero[i]=new char[n];
        for(int j=0;j<n;j++){
            *(*(validar+i)+j)=false;
```

```

        *(*(tablero+i)+j)='.' ;
    }
}
enter.open("solucion.txt",fstream::out); //para leer in, para salir es
out escribir
enter<<"solucion "<<n<<"*"

```

```

        *(* validar+aux1)+aux2)=true;
        aux1--;
        aux2++;
    }
}
void Reina::quitarRelleno(int x, int y)
{
    *(*(tablero+x)+y)='.';
    for(int i=0;i<n;i++) {
        for(int j=0;j<n;j++) {
            *(* validar+i)+j)=false;
        }
    }
    for(int i=0;i<n;i++) {
        for(int j=0;j<n;j++) {
            if(*(*tablero+i)+j)=='R')
                bloquear(i,j);
        }
    }
}
void Reina::mostrar()
{
    contador++;
    enter<<"Solucion N "<<contador<<endl;
    for(int i=0;i<n;i++) {
        for(int j=0;j<n;j++) {
            enter<<*(*tablero+i)+j)<< " ";
        }
        enter<<endl;
    }
    enter<<endl;
}
void Reina::solucionReinas()
{
    for(int i=0;i<n;i++) {
        solucion(i,0,1);
    }
    enter.close();
}

```

- sirve para validar la entrada de datos al programa según el tipo que se necesite.

```

#include <iostream>
#include <string>

using namespace std;

class Ingreso{
public:
    string ingresar(string);
    bool validarTipoFloat(string);
    bool validarTipoInt(string);
    bool validarTipoString(string);
};

string Ingreso::ingresar(string msg){

```

```

string dato_a_validar;
bool esValido = false;

while(!esValido){
    try{
        cout << msg;
        getline(cin,dato_a_validar);
        if(msg.find("flotante")!= std::string::npos){
            esValido = validarTipoFloat(dato_a_validar);
        }
        else if(msg.find("entero")!= std::string::npos){
            esValido = validarTipoInt(dato_a_validar);
        }
        else if(msg.find("cadena")!= std::string::npos){
            esValido = validarTipoString(dato_a_validar);
        }

        if(!esValido){
            throw dato_a_validar;
        }
    }catch(string e){
        cout << "El dato (" << e << ")" no es valido" << endl;
    }
}
return dato_a_validar;
}

bool Ingreso::validarTipoFloat(string numero){
    int inicio = 0;
    if(numero.length() == 0){
        return 0;
    }
    if(numero[0] == '+' || numero[0] == '-'){
        inicio = 1;
        if(numero.length() == 1){
            return 0;
        }
    }
    for(int i = inicio; i<numero.length(); i++) {
        if (!isdigit(numero[i]) && numero[i] != '.') {
            return 0;
        }
    }
    return 1;
}

bool Ingreso::validarTipoInt(string numero){
    int inicio = 0;
    if(numero.length() == 0){
        return 0;
    }
    if(numero[0] == '+' || numero[0] == '-'){
        inicio = 1;
        if(numero.length() == 1){
            return 0;
        }
    }
    for(int i = inicio; i<numero.length(); i++) {
        if (!isdigit(numero[i])) {

```

```

        return 0;
    }
}
return 1;
}
bool Ingreso::validarTipoString(string numero){
    if(numero.length() == 0){
        return 0;
    }

    for(int i = 0; i<numero.length(); i++) {
        if (isdigit(numero[i])) {
            return 0;
        }
    }
    return 1;
}

```

- Se arma el programa con todas las clases para generar las soluciones de las ocho Reinas.

```

#include <iostream>
#include <fstream>
#include "Reina.h"
#include "Ingreso.h"

/* run this program using the console pauser or add your own getch,
system("pause") or input loop */

using namespace std;

int main(int argc, char** argv) {
    int n;
    cout<<"\tCuantas reinas: "<<endl;
    cin>>n;
    Reina *obj= new Reina(n);
    obj->solucionReinas();
    cout<<"Se creo el archivo solucion.txt"<<endl;

    return 0;
}

```

Ejecucion de programa:

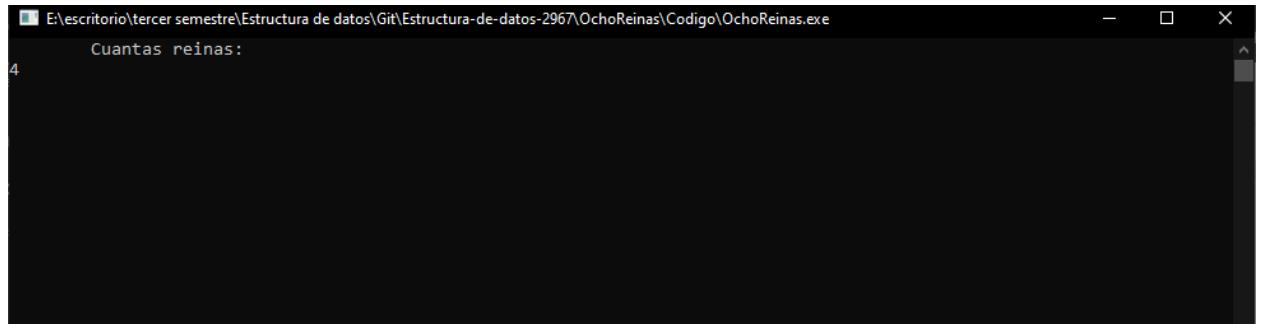


Figura 1. ingreso del número de reinas.

- A continuacion se muestra el archivo “soluciones.txt” donde se almacenan todas las soluciones brindadas por backtracking.

solución 4*4

Solución N 1

R . . .
. R . .
. . R .
. . . R

Solución N 2

R . . .
. . . R
. R . .
. . R .

Solución N 3

R . . .
. . R .
. . . R
. R . .

Solución N 4

. . R .
R . . .
. R . .
. . . R

Solución N 5

. . R .
R . . .
. . . R
. R . .

Solución N 6

. R . .
. . R .
R . . .
. . . R

Solución N **7**

```
. R . .
. . . R
R . . .
. . R .
```

Generar e-mail:

Objetivo:

Generar el correo con los datos ingresados por el usuario que son los nombre y apellidos y comprobar si la fecha es palindroma.

Código:

- generar los prototipos de las funciones.

```
#include <string>
#include "Provincia.h"
#include "FechaNacimiento.h"
using namespace std;

class Persona: public Provincia, public FechaNacimiento
{
public:
    string getCedula(void);
    void setCedula(string newCedula);
    string getNombre(void);
    void setNombre(string newNombre);
    string getApellido(void);
    void setApellido(string newApellido);
    string getNacionalidad(void);
    void setNacionalidad(string newNacionalidad);
    string getGenero(void);
    void setGenero(string newGenero);
    string getEstCivil(void);
    void setEstCivil(string newEstCivil);
    string getTelf(void);
    void setTelf(string newTelf);
    Persona();

private:
    string cedula;
    string nombre;
    string apellido;
    string nacionalidad;
    string genero;
    string estCivil;
```

```
        string telf;
    };

string Persona::getCedula(void)
{
    return cedula;
}

void Persona::setCedula(string newCedula)
{
    cedula = newCedula;
}

string Persona::getNombre(void)
{
    return nombre;
}

void Persona::setNombre(string newNombre)
{
    nombre = newNombre;
}

string Persona::getApellido(void)
{
    return apellido;
}

void Persona::setApellido(string newApellido)
{
    apellido = newApellido;
}

string Persona::getNacionalidad(void)
{
    return nacionalidad;
}

void Persona::setNacionalidad(string newNacionalidad)
{
    nacionalidad = newNacionalidad;
}

string Persona::getGenero(void)
{
    return genero;
}

void Persona::setGenero(string newGenero)
{
    genero = newGenero;
}

string Persona::getEstCivil(void)
{
    return estCivil;
}
```

```

void Persona::setEstCivil(string newEstCivil)
{
    estCivil = newEstCivil;
}

string Persona::getTelf(void)
{
    return telf;
}

void Persona::setTelf(string newTelf)
{
    telf = newTelf;
}

Persona::Persona()
{
}

```

- generar prototipos de las funciones.

```

#include <string>
using namespace std;
#include "Persona.h"

class Alumno : public Persona
{
public:
    string getIdAlumno(void);
    void setIdAlumno(string newIdAlumno);
    string getEmail(void);
    void setEmail(string newEmail);
    Alumno();
private:
    string idAlumno;
    string email;
};

string Alumno::getIdAlumno(void)
{
    return idAlumno;
}

void Alumno::setIdAlumno(string newIdAlumno)
{
    idAlumno = newIdAlumno;
}

string Alumno::getEmail(void)
{
    return email;
}

```

```

}

void Alumno::setEmail(string newEmail)
{
    email = newEmail;
}

Alumno::Alumno()
{
}

Provincia.h
#include <string>
#include "Canton.h"
using namespace std;
class Canton;

class Provincia : public Canton
{
public:
    string getIdProvincia(void);
    void setIdProvincia(string newIdProvincia);
    string getNombreProvincia(void);
    void setNombreProvincia(string newNombreProvincia);
    Provincia();
protected:
    string idProvincia;
    string nombreProvincia;
};

string Provincia::getIdProvincia(void)
{
    return idProvincia;
}

void Provincia::setIdProvincia(string newIdProvincia)
{
    idProvincia = newIdProvincia;
}

string Provincia::getNombreProvincia(void)
{
    return nombreProvincia;
}
void Provincia::setNombreProvincia(string newNombreProvincia)
{
    nombreProvincia = newNombreProvincia;
}

Provincia::Provincia()
{
}

#include <string>
#include "Parroquia.h"
using namespace std;

```

```

class Canton: public Parroquia
{
public:
    string getIdCanton(void);
    void setIdCanton(string newIdCanton);
    string getNombreCanton(void);
    void setNombreCanton(string newNombreCanton);
    Canton();
protected:
    string idCanton;
    string nombreCanton;
};

string Canton::getIdCanton(void)
{
    return idCanton;
}

void Canton::setIdCanton(string newIdCanton)
{
    idCanton = newIdCanton;
}

string Canton::getNombreCanton(void)
{
    return nombreCanton;
}

void Canton::setNombreCanton(string newNombreCanton)
{
    nombreCanton=newNombreCanton;
}

Canton::Canton()
{
}

#include <string>
using namespace std;

class Parroquia
{
public:
    string getIdParroq(void);
    void setIdParroq(string newIdParroq);
    string getNombreParroq(void);
    void setNombreParroq(string newNombreParroq);
    Parroquia();
protected:
    string idParroq;
    string nombreParroq;
};


```

```

string Parroquia::getIdParroq(void)
{
    return idParroq;
}
void Parroquia::setIdParroq(string newIdParroq)
{
    idParroq = newIdParroq;
}

string Parroquia::getNombreParroq(void)
{
    return nombreParroq;
}
void Parroquia::setNombreParroq(string newNombreParroq)
{
    nombreParroq = newNombreParroq;
}

Parroquia::Parroquia()
{
}

#include <string>
#include <stdlib.h>
using namespace std;

class FechaNacimiento
{
public:
    int getDia(void);
    void setDia(int newDia);
    int getMes(void);
    void setMes(int newMes);
    int getAnio(void);
    void setAnio(int newAnio);
    FechaNacimiento();
    bool fechaPalindroma();

private:
    int dia;
    int mes;
    int anio;
};

int FechaNacimiento::getDia(void)
{
    return dia;
}

void FechaNacimiento::setDia(int newDia)
{
    dia = newDia;
}

int FechaNacimiento::getMes(void)
{

```

```

        return mes;
    }

void FechaNacimiento::setMes(int newMes)
{
    mes = newMes;
}

int FechaNacimiento::getAnio(void)
{
    return anio;
}

void FechaNacimiento::setAnio(int newAnio)
{
    anio = newAnio;
}
FechaNacimiento::FechaNacimiento()
{
}

bool FechaNacimiento::fechaPalindroma() {
    string palindromo;
    char* conv;
    conv=(char*)malloc(8*sizeof(char));
    sprintf(conv,"%d%d%d",dia,mes,anio);
    palindromo.append(conv);
    free(conv);
    int x=palindromo.length();
    for(int j=0;j<(palindromo.length()/2);j++) {
        if(palindromo.at(j)!=palindromo.at(x-1)){
            return false;
        }
        x--;
    }
    return true;
}

```

```

#include <string>
using namespace std;

class GenerarEmail
{
public:
    virtual string genEmail(string nombre, string apellido)=0;
};

#include <iostream>
#include "Excluir.h"
#include <string>
using namespace std;

class Ingreso{
    public:

```

```

        string ingresarString(char* );
        int ingresar(char* );
        string ingresarStringN(char* );
};

string Ingreso::ingresarString(char *msg) {
    string valor;
    bool flag;
    Excluir comprobante;
    cout<<msg<<endl;
    do{
        cin>>valor;
        flag=comprobante.controls(valor);
        }while(flag==false);
    return valor;
}

string Ingreso::ingresarStringN(char *msg) {
    string valor;
    bool flag;
    Excluir comprobante;
    cout<<msg<<endl;
    do{
        cin>>valor;
        flag=comprobante.controlSN(valor);
        }while(flag==false);
    return valor;
}

int Ingreso::ingresar(char *msg) {
    string valor;
    bool flag;
    Excluir comprobante;
    cout<<msg<<endl;
    do{
        cin>>valor;
        flag=comprobante.control(valor);
        }while(flag==false);
    int num;
    num = strtod((valor).c_str(),0);
    return num;
}

#include <iostream>
#include <string>
#include <algorithm>
#include <vector>
#include <ctime>
#include <random>
#include "GenerarEmail.h"
using namespace std;

class PruebaEmail : public GenerarEmail
{
public:
    string genEmail(string nombre, string apellido);
    PruebaEmail();
};

string PruebaEmail::genEmail(string nombre, string apellido)
{

```

```

        string email, min;
        char* alocator;
        alocator=(char*)malloc(1*sizeof(char));
        int x=nombre.find(" ");
        if(x>1){
            sprintf(alocator,"%c%c",nombre.at(0),nombre.at(nombre.find(" ")+1));
        }
        else{
            sprintf(alocator,"%c",nombre.at(0));
        }
        email.append(alocator);
        if(apellido.find(" ")>0){
            email.append(apellido.substr(0,apellido.find(" ")).c_str());
        }
        else{
            email.append(apellido.c_str());
        }
        for(int i = 0; i<email.length();i++){
            int y=(int)email.at(i);
            char c=tolower(y);
            char* minus;
            minus=(char*)malloc(1*sizeof(char));
            sprintf(minus,"%c",c);
            min.append(minus);
            free(minus);
        }
        return min;
    }

PruebaEmail::PruebaEmail()
{
}

#include <iostream>
#include <string>
#include <bits/stdc++.h>
#include <stdlib.h>

using namespace std;

class Excluir{
public:
    bool control(string s){
        int d=0;
        for(int i=0;i<s.size();i++){
            if(!isdigit(s[i])){
                cout<<"Error"<<endl;
                return false;
            }
        }
        return true;
    }

    bool controls(string s){
        int d=0;

```

```

        for(int i=0;i<s.size();i++)
        {
            if(isdigit(s[i])&&s[i]!=' ')
                cout<<"Error"=>endl;
                return false;
            }
        }
        return true;
    }

    bool controlSN(string s){
        int d=0;
    for(int i=0;i<s.size();i++)
    {
        if(!isdigit(s[i])){
            cout<<"Error"=>endl;
            return false;
        }
    }
    return true;
}
};


```

- construcción del programa con las funciones ya declaradas.

```

#include <stdio.h>
#include <iostream>
#include <string>
#include <fstream>
#include "Alumno.h"
#include "PruebaEmail.h"
#include "Ingreso.h"

int main(){
    Ingreso ex;
    string idPa,idPr,idCa, idAlumno, nombre,apellido;
    Alumno alum = Alumno();
    PruebaEmail pru = PruebaEmail();
    cout<<"Ingrese su nombre"=>endl;
    getline(cin,nombre);
    alum.setNombre(nombre);
    cout<<"Ingrese su apellido"=>endl;
    getline(cin,apellido);
    alum.setApellido(apellido);
    alum.setCedula(ex.ingresarString("Ingrese su cedula"));
    alum.setEstCivil(ex.ingresarString("Ingrese su estado civil"));
    alum.setGenero(ex.ingresarString("Ingrese su genero"));
    alum.setTelf(ex.ingresarString("Ingrese su telefono"));
    alum.setNacionalidad(ex.ingresarString("Ingrese su nacionalidad"));
    alum.setAnio(ex.ingresar("Ingrese su anio de nacimiento"));
    alum.setMes(ex.ingresar("Ingrese su mes de nacimiento"));
    while(alum.getMes()>12){
        alum.setMes(ex.ingresar("Ingrese su mes de nacimiento"));
    }
}


```

```

alum.setDia(ex.ingresar("Ingrese su dia de nacimiento"));
while(alum.getDia()>28&&alum.getMes()==2) {
    alum.setDia(ex.ingresar("Ingrese su dia de nacimiento"));
}
while(alum.getDia()>31) {
    alum.setDia(ex.ingresar("Ingrese su dia de nacimiento"));
}
cout<<"Ingrese el id de su provincia"<<endl;
cin>>idPr;
alum.setIdProvincia(idPr);
alum.setNombreProvincia(ex.ingresarString("Ingrese el nombre de su
provincia de residencia"));
cout<<"Ingrese el id de su canton"<<endl;
cin>>idCa;
alum.setIdCanton(idCa);
alum.setNombreCanton(ex.ingresarString("Ingrese el nombre de su canton
de residencia"));
cout<<"Ingrese el id de su parroquia"<<endl;
cin>>idPa;
alum.setIdParroq(idPa);
alum.setNombreParroq(ex.ingresarString("Ingrese el nombre de su
parroquia de residencia"));
cout<<"Ingrese su id de alumno"<<endl;
cin>>idAlumno;
alum.setIdAlumno(idAlumno);
alum.setEmail(pru.genEmail(alum.getNombre(),alum.getApellido()));
string line;
fstream out;
out.open("example.txt",fstream::app);
int i=0;
ifstream myfile ("example.txt");
if (myfile.is_open())
{
    while ( getline (myfile,line) )
    {

        if(pru.genEmail(alum.getNombre(),alum.getApellido()).compare(line.substr(0,pru.genEmail(alum.getNombre(),alum.getApellido()).length())==0){
            i++;
        }
    }
    myfile.close();
    line=pru.genEmail(alum.getNombre(),alum.getApellido());
    if(i!=0){
        char* num;
        num=(char*)malloc(1*sizeof(char));
        sprintf(num,"%d",i);
        line.append(num);
    }
    out<<line<<endl;
    out.close();
}
else{
}
if(alum.fechaPalindroma()){
    cout<<"es palindromo"<<endl;
}else{
}

```

```
        cout<<"Nel prro"<<endl;
    }

    return 0;
}
```

Ejecución:

```
E:\escritorio\tercer semestre\Estructura de datos\Git\Estructura-de-datos-2967\Generar correo\Codigo\Prueba.exe
Ingrese su nombre
cesar jacobo
Ingresar su apellido
naula maji
Ingresar su cedula
1725544405
Ingresar su estado civil
s
Ingresar su genero
m
Ingresar su telefono
1234455
Ingresar su nacionalidad
ecuatoriana
Ingresar su anio de nacimiento
1999
Ingresar su mes de nacimiento
06
Ingresar su dia de nacimiento
22
Ingresar el id de su provincia
23
Ingresar el nombre de su provincia de residencia
Pichincha
Ingresar el id de su canton
23
Ingresar el nombre de su canton de residencia
Quito
```

Figura 1. ejecución de programa.

```
E:\escritorio\tercer semestre\Estructura de datos\Git\Estructura-de-datos-2967\Generar correo\Codigo\Prueba.exe
1234567
Ingrese su nacionalidad
ecuatoriana
Ingrese su anio de nacimiento
1999
Ingrese su mes de nacimiento
06
Ingrese su dia de nacimiento
f
Error
f
Error
22
Ingrese el id de su provincia
12
Ingrese el nombre de su provincia de residencia
Pichincha
Ingrese el id de su canton
34
Ingrese el nombre de su canton de residencia
Quito
Ingrese el id de su parroquia
56
Ingrese el nombre de su provincia de residencia
Quniche
Ingrese su id de alumno
1000394534
Nel prrro
```

Figura 2. ingreso de datos

```
example: Bloc de notas
kzurita
kmateo
kmzurita
cjnaula
cjnaula1
```

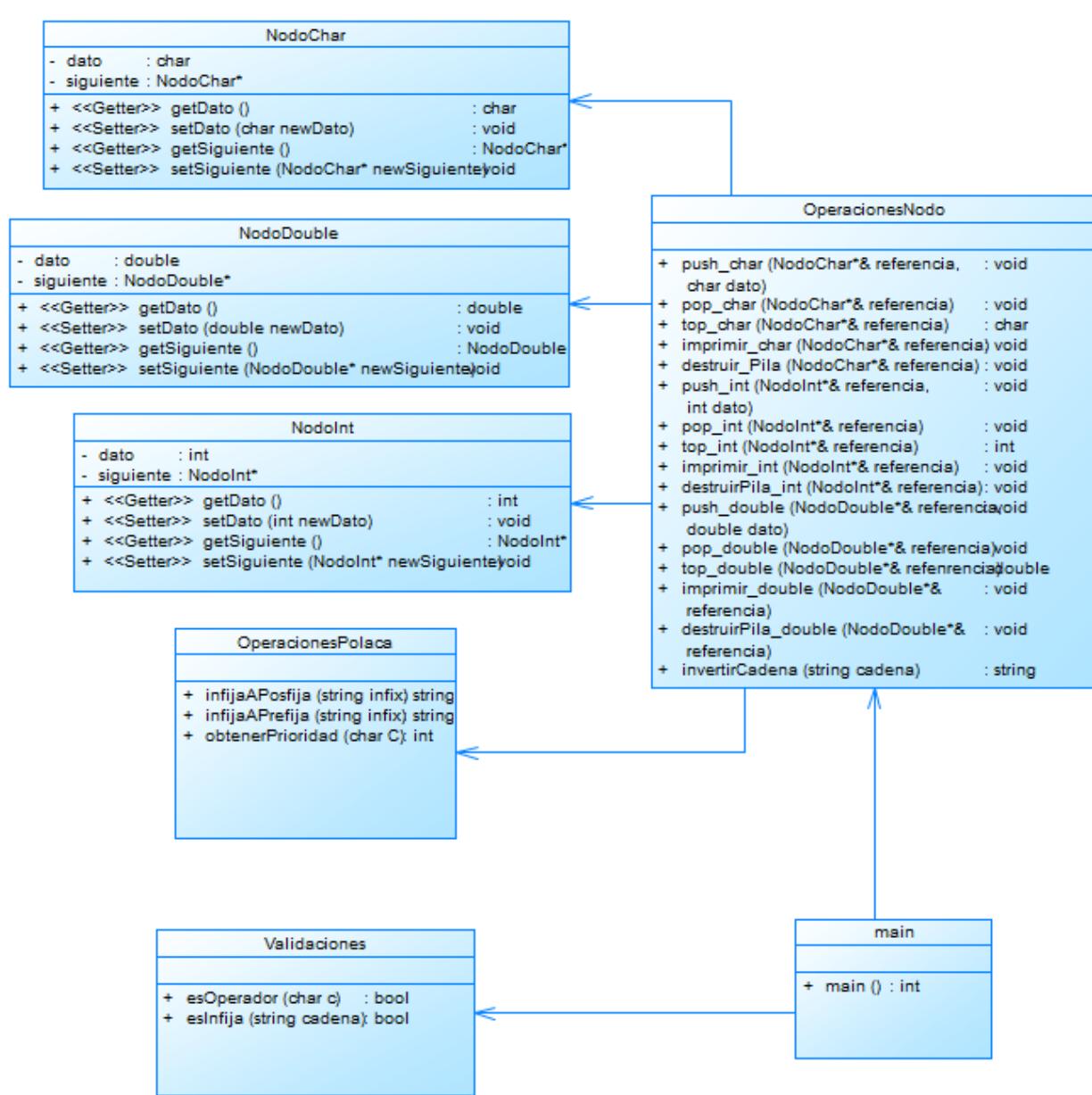
Figura 3. Correos generados.

Polaca Inversa

Descripción

La aplicación tiene como objetivo transformar una expresión algebraica o en forma infija a la notación prefijo y postfixa.

Modelo



Código

Nodoint.h

```

class NodoInt
{
public:
    int getData(void);
    void setData(int newData);
    NodoInt* getSiguiente(void);
    void setSiguiente(NodoInt* newSiguiente);
protected:
private:
    int dato;
    NodoInt* siguiente;
};

Nodoint.cpp
#include "NodoInt.h"
int NodoInt::getData(void)
{
    return dato;
}
void NodoInt::setData(int newData)
{
    dato = newData;
}
NodoInt* NodoInt::getSiguiente(void)
{
    return siguiente;
}
void NodoInt::setSiguiente(NodoInt* newSiguiente)
{
    siguiente = newSiguiente;
}

nodoDouble.h
class NodoDouble
{
public:
    double getData(void);
    void setData(double newData);
    NodoDouble* getSiguiente(void);
    void setSiguiente(NodoDouble* newSiguiente);
protected:
private:
    double dato;
    NodoDouble* siguiente;
};
Nododouble.cpp

#include "NodoDouble.h"
double NodoDouble::getData(void)
{
    return dato;
}
void NodoDouble::setData(double newData)
{
    dato = newData;
}
NodoDouble* NodoDouble::getSiguiente(void)
{
    return siguiente;
}

```

```

}

void NodoDouble::setSiguiente(NodoDouble* newSiguiente)
{
    siguiente = newSiguiente;
}
nodOperaciones.h
#include "NodoChar.h"
#include "NodoInt.h"
#include "NodoDouble.h"
#include <string>
using namespace std;
class OperacionesNodo
{
public:
    void push_char(NodoChar*& referencia, char dato);
    void pop_char(NodoChar*& referencia);
    char top_char(NodoChar*& referencia);
    void imprimir_char(NodoChar* referencia);
    void destruirPila_char(NodoChar* referencia);
    //Nodo con tipo de dato entero
    void push_int(NodoInt*& referencia, int dato);
    void pop_int(NodoInt*& referencia);
    int top_int(NodoInt*& referencia);
    void imprimir_int(NodoInt* referencia);
    void destruirPila_int(NodoInt* referencia);
    //Nodo con tipo de dato double
    void push_double(NodoDouble*& referencia, double dato);
    void pop_double(NodoDouble*& referencia);
    double top_double(NodoDouble*& referencia);
    void imprimir_double(NodoDouble* referencia);
    void destruirPila_double(NodoDouble* referencia);
    string invertirCadena(string cadena);
};

Nodooperaciones.cpp
#include "OperacionesNodo.h"
#include <iostream>
using namespace std;
void OperacionesNodo::imprimir_char(NodoChar* referencia)
{
    NodoChar* auxiliar = referencia;
    cout << "Pila: ";
    while (auxiliar != NULL)
    {
        cout << auxiliar->getdato() << " ";
        auxiliar = auxiliar->getsiguiente();
    }
}
void OperacionesNodo::destruirPila_char(NodoChar* referencia)
{
    NodoChar* aux = referencia;
    while (referencia != NULL)
    {
        referencia = aux->getsiguiente();
        delete(aux);
    }
}
void OperacionesNodo::pop_char(NodoChar*& referencia)

```

```

{
    NodoChar* aux = new NodoChar();
    aux = referencia;
    referencia = aux->getSiguiente();
    delete(aux);
}
char OperacionesNodo::top_char(NodoChar*& referencia)
{
    NodoChar* aux = new NodoChar();
    aux = referencia;
    char tope = aux->getData();
    return tope;
}
void OperacionesNodo::push_char(NodoChar*& referencia, char dato)
{
    NodoChar* aux = new NodoChar();
    aux->setData(dato);
    aux->setSiguiente(referencia);
    referencia = aux;
}
void OperacionesNodo::imprimir_int(NodoInt* referencia)
{
    NodoInt* auxiliar = referencia;
    cout << "Pila: ";
    while (auxiliar != NULL)
    {
        cout << auxiliar->getData() << " ";
        auxiliar = auxiliar->getSiguiente();
    }
}
void OperacionesNodo::destruirPila_int(NodoInt* referencia)
{
    NodoInt* aux = referencia;
    while (referencia != NULL)
    {
        referencia = aux->getSiguiente();
        delete(aux);
    }
}
void OperacionesNodo::pop_int(NodoInt*& referencia)
{
    NodoInt* aux = new NodoInt();
    aux = referencia;
    referencia = aux->getSiguiente();
    delete(aux);
}
int OperacionesNodo::top_int(NodoInt*& referencia)
{
    NodoInt* aux = new NodoInt();
    aux = referencia;
    int tope = aux->getData();
    cout << tope;
    return tope;
}
void OperacionesNodo::push_int(NodoInt*& referencia, int dato)
{
    NodoInt* aux = new NodoInt();

```

```

aux->setDatos(dato);
aux->setSiguiente(referencia);
referencia = aux;
}
void OperacionesNodo::imprimir_double(NodoDouble*& referencia)
{
    NodoDouble* auxiliar = referencia;
    cout << "Pila: ";
    while (auxiliar != NULL)
    {
        cout << auxiliar->getDatos() << " ";
        auxiliar = auxiliar->getSiguiente();
    }
}
void OperacionesNodo::destruirPila_double(NodoDouble*& referencia)
{
    NodoDouble* aux = referencia;
    while (referencia != NULL)
    {
        referencia = aux->getSiguiente();
        delete(aux);
    }
}
void OperacionesNodo::pop_double(NodoDouble*& referencia)
{
    NodoDouble* aux = new NodoDouble();
    aux = referencia;
    referencia = aux->getSiguiente();
    delete(aux);
}
double OperacionesNodo::top_double(NodoDouble*& referencia)
{
    NodoDouble* aux = new NodoDouble();
    aux = referencia;
    double tope = aux->getDatos();
    return tope;
}
void OperacionesNodo::push_double(NodoDouble*& referencia, double dato)
{
    NodoDouble* aux = new NodoDouble();
    aux->setDatos(dato);
    aux->setSiguiente(referencia);
    referencia = aux;
}
string OperacionesNodo::invertirCadena(string cadena) {

    int dim = cadena.length();
    char* temporal1 = (char*)cadena.c_str();
    char* temporal2 = new char[dim];
    int pivot = 0;
    for (int i = dim - 1; i >= 0; i--) {
        *(temporal2 + i) = *(temporal1 + pivot);
        pivot++;
    }
    string cadenaInvertida(temporal2, dim);
    return cadenaInvertida;
}

```

```

Validaciones.h
#pragma once
#include <string>
class Validaciones
{
public:
    bool esOperador(char);
    bool esInfija(std::string);
};

Validaciones.cpp
#include "Validaciones.h"
#include <string>
bool Validaciones::esOperador(char c) {
    return (!isalpha(c) && !isdigit(c));
}

bool Validaciones::esInfija(std::string cadena) {
    for (int i = 0; i < cadena.size(); i++) {
        /*Validacion de simbolos especiales y letras*/
        if ((cadena.at(i) > 32 && cadena.at(i) < 40) ||
            (cadena.at(i) == 44) ||
            (cadena.at(i) > 57 && cadena.at(i) < 94) ||
            (cadena.at(i) > 94 && cadena.at(i) < 255))
            return false;
    }

    return true;
}

operacionesPolaca.h
#include <string>
using namespace std;
class OperacionesPolaca
{
public:
    string infijaAPosfija(string);
    string infijaAPrefija(string);
    int obtenerPrioridad(char);
};

operacionesPolaca.cpp
#include "OperacionesPolaca.h"
#include "NodoChar.h"
#include "OperacionesNodo.h"
#include "Validaciones.h"
#include <iostream>
OperacionesNodo operacion;
Validaciones validacion;

string OperacionesPolaca::infijaAPosfija(string infix) {
    infix = '(' + infix + ')';
    NodoChar* nodo = NULL;
    string resultado;
    for (int i = 0; i < infix.size(); i++) {
        if (isalpha(infix.at(i)) || isdigit(infix[i]))
            resultado += infix.at(i);
        else if (infix.at(i) == '(')
            operacion.push_char(nodo, '(');
        else if (infix.at(i) == ')') {

```

```

while (operacion.top_char(nodo) != '(') {
    resultado += operacion.top_char(nodo);
    operacion.pop_char(nodo);
}
operacion.pop_char(nodo);
}
else {
    if (validacion.esOperador(operacion.top_char(nodo))) {
        while (obtenerPrioridad(infix.at(i)) <=
obtenerPrioridad(operacion.top_char(nodo))) {
            resultado += operacion.top_char(nodo);
            operacion.pop_char(nodo);
        }
        operacion.push_char(nodo, infix.at(i));
    }
}
}
return resultado;
}

string OperacionesPolaca::infijaAPrefija(string infix) {

operacion.invertirCadena(infix);
for (int i = 0; i < infix.size(); i++) {
    if (infix.at(i) == ')') {
        infix.at(i) = ')';
        i++;
    }
    else if (infix.at(i) == '(') {
        infix.at(i) = '(';
        i++;
    }
}
string prefix = infijaAPosfija(infix);
operacion.invertirCadena(prefix);
return prefix;
}

int OperacionesPolaca::obtenerPrioridad(char C) {
if (C == '-' || C == '+')
    return 1;
else if (C == '*' || C == '/')
    return 2;
else if (C == '^')
    return 3;
else
    return 0;
}

Principal
#include<iostream>
#include "NodoChar.h"
#include "OperacionesPolaca.h"
#include "Validaciones.h"
#include <cstdio>
#include <conio.h>
#include <windows.h>
#include <cstdlib>
using namespace std;

```

```

#define ARRIBA 72
#define ABAJO 80
#define ENTER 13
void menuCursor();
void menu();
void gotoxy(int x, int y);
void Selector(int i);
int main() {

    for (;;) {
        menuCursor();
    }
    return 0;
}
void menuCursor() {

    int i = 0;
    Selector(i);
    menu();
    gotoxy(3, 1);
    char tecla;
    while (true) {
        tecla = _getch();
        switch (tecla) {
            case ARRIBA:
                i--;
                if (i < 0) {
                    i = 3;
                }
                Selector(i);
                break;
            case ABAJO:
                i++;
                if (i == 0) {
                    i = 1;
                }
                if (i == 3) {
                    i = 0;
                }
                Selector(i);
                break;
            case ENTER:
                Selector(5);
                Validaciones validacion;
                string cadena, aux;
                OperacionesPolaca operacion;
                switch (i) {
                    case 0:
                        do{
                            cout << "Ingrese la expresion en infijo: ";
                            cin >> cadena;
                        } while (!validacion.esInfija(cadena));

                        aux = operacion.infijaAPosfija(cadena);
                        cout << "Posfija: " << endl << aux << endl;
                        system("pause");
                }
        }
    }
}

```

```

        break;
    case 1:
        do {
            cout << "Ingrese la expresion en infijo: ";
            cin >> cadena;
        } while (!validacion.esInfija(cadena));
        aux = operacion.infijaAPrefija(cadena);
        cout << "Prefija: " << endl << aux << endl;

        system("pause");
        break;
    case 2:
        cout << endl << endl;
        cout << "Saliendo del programa" << endl;
        exit(0);
        break;
    }
    system("cls");
    menu();
    gotoxy(3, 1);
    break;
}
}
}

void menu() {
    system("cls");
    cout << "|*****Menu*****| " << endl;
    cout << "\tNotacion Posfijo" << endl;
    cout << "\tNotacion Prefijo" << endl;
    cout << "\tSalir del Programa" << endl;
    cout << "|*****| " << endl;
}

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 Selector(int i) {
    gotoxy(1, 1 + i);
}

```

Ejecución

```
C:\Users\fleia\Desktop\polocalInversa\Debug\polocalInversa.exe
*****
Menu*****
Notacion Posfijo
Notacion Prefijo
Salir del Programa
*****
Ingrese la expresion en infijo: 5*9/6+2^3
Posfija:
59*6/23^+
Press any key to continue . . .
```

Figura 1. Ingreso de operaciones

```
C:\Users\fleia\Desktop\polocalInversa\Debug\polocalInversa.exe
*****
Menu*****
Notacion Posfijo
Notacion Prefijo
Salir del Programa
*****
Ingrese la expresion en infijo: 5/8+6-12^3-6
Prefija:
58/6+123^-6-
Press any key to continue . . .
```

Figura 2. Operación polaca inversa

Prueba Segundo Parcial

Con la ayuda de la realización del listas crear un programa que de manera aleatoria almacene números que caen en la pantalla de la consola (parecido a un juego) .

Objetivo de la aplicación

Con la ayuda de la realización del programa “Tetris” y “Snake” realizar una aplicación en donde de manera aleatoria caigan números y con la ayuda de las listas ir almacenando al final de ella. Además de que el movimiento del cero permita atrapar los números que se añadirán a la lista.

Modelado

ListaComida	
-	ListaComida() : int
-	insertar() : void
-	insertarPrimero() : void
-	listaVacia() : boolean
-	crearLinea() : char*
-	imprimirComida() : void
-	comprobarElementos : boolean
-	guitarElementos : int
+	cnodo Primero () : int

ListaEnlazada	
-	ListaEnlazada() : int
-	insertar : void
-	insertarPrimero : void
-	insertarVacia : boolean
-	numeroElementos : int
-	crearLinea() : char*
+	pnode primero () : int

NodoComida	
-	valorTetris : int
-	filas : int
-	columna : int
-	NodoComida* siguiente : int
+	NodoComida () : int

NodoSnake	
-	numero : int
+	NodoSanke (int NodoSanke*)
+	NodoSanke *siguiente () : int

Código de la aplicación

- La librería “NodoSnake” es la que permite obtener los numero que caerán en la consola.

```
#include <iostream>
#include <stdio.h>
#include <string>

using namespace std;

class NodoSnake{
public:
    NodoSnake(int, NodoSnake*);
    int getNumero();
private:
```

```

        int numero;
        NodoSnake *siguiente;

    friend class ListaEnlazada;
};

NodoSnake::NodoSnake(int num, NodoSnake *sig=NULL) {
    numero=num;
    siguiente=sig;
}

int NodoSnake::getNumero() {
    return numero;
}

```

- La librería “NodoComida” setea los valores de la lista y la lista que se va a almacenar

```

#include <iostream>
#include <stdio.h>

using namespace std;

class NodoComida{
private:
    int valorTetris;
    int filas;
    int columna;
    NodoComida* sig;
public:
    NodoComida(int,int,int);
    int getValorTetris();
    void setValorTetris(int);
    int getColumnas();
    void setColumnas(int);
    int getFilas();
    void setFilas(int);
    NodoComida* getSiguiente();
    void setSiguiente(NodoComida*);
    NodoComida* getAnterior();
    void setAnterior(NodoComida*);
    void imprimir();
    friend class ListaComida;
};

NodoComida::NodoComida(int value, int row, int column) {
    valorTetris=value;
    filas=row;
    columna=column;
    sig=NULL;
}

```

```

int NodoComida::getValorTetris() {
    return valorTetris;
}

void NodoComida::setValorTetris(int value){
    valorTetris=value;
}

int NodoComida::getFilas(){
    return filas;
}

void NodoComida::setFilas(int row){
    filas=row;
}

int NodoComida::getColumnas(){
    return columna;
}

void NodoComida::setColumnas(int column){
    columna=column;
}

NodoComida* NodoComida::getSiguiente(){
    return sig;
}

void NodoComida::setSiguiente(NodoComida* after){
    sig=after;
}
• La librería "ListaComida" almacena los numero en una lista y las va llenando conforme caen los números en la consola, además de que permite el movimiento del cero que adjuntara los números a la lista

#include <iostream>
#include <stdio.h>
#include <string>
#include <stdlib.h>
#include <windows.h>
#include "NodoComida.h"

void gotoxy(int x, int y)
{
    HANDLE hCon;
    COORD dwPos;

    dwPos.X = x;
    dwPos.Y = y;
    hCon = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleCursorPosition(hCon,dwPos);
}

typedef NodoComida *cnodo;

class ListaComida{

```

```

public:
    ListaComida();
    void insertar(int,int,int);
    void insertarPrimero(int,int,int);
    bool listaVacia();
    char* crearLinea();
    void imprimirComida(int);
    bool comprobarElementos(int,int,int,int);
    int quitarElemento();
private:
    cnodo primero;
};

ListaComida::ListaComida() {
    primero=NULL;
}

void ListaComida::insertar(int value, int row, int column){
    cnodo anterior;
    if(listaVacia()) {
        primero = new NodoComida(value, row, column);
    } else {
        anterior = primero;
        primero = new NodoComida(value, row, column);
        primero->sig =anterior;
    }
}

bool ListaComida::listaVacia(){
    return primero == NULL;
}

char* ListaComida::crearLinea(){
    string serpiente;
    char* temp;
    cnodo aux;
    aux = primero;
    while(aux) {
        temp=(char*)malloc(1*sizeof(int));
        sprintf(temp,"%d",aux->getValorTetris());
        serpiente.append(temp);
        aux = aux->sig;
        free(temp);
    }
    temp=(char*)malloc(serpiente.length()*sizeof(int));
    sprintf(temp,"%s",serpiente.c_str());
    return temp;
}

void ListaComida::insertarPrimero(int value, int row, int column){
    cnodo aux;
    aux=primero;
    if(listaVacia()) {
        primero = new NodoComida(value, row, column);
    } else {
        while(aux->sig) {

```

```

        aux = aux->sig;
    }
    aux->sig=new NodoComida(value, row, column);
}
}

void ListaComida::imprimirComida(int elementos){
    cnodo aux=primero;
    while(aux!=NULL) {
        gotoxy(aux->getColumnas(),aux->getFilas()+elementos*2);
        printf("%d",aux->getValorTetris());
        elementos--;
        aux=aux->sig;
    }
}

bool ListaComida::comprobarElementos(int xCuerpo, int yCuerpo, int elementos,
int cuerpo){
    cnodo aux=primero;
    if(aux->getFilas()+elementos*2==yCuerpo) {
        //      if(aux->getColumnas()<=xCuerpo+cuerpo&&aux-
>getColumnas()>=xCuerpo){ //comentado para poder ganar el juego y demostrar
funcionalidad
            return true;
        //      }
    }
    return false;
}

int ListaComida::quitarElemento(){
    cnodo aux=primero;
    primero=primero->sig;
    return aux->getValorTetris();
}

```

- La librería “ListaEnlazada” declarada para el cuerpo de la lista adjunta los elementos en este caso los números que caen ya sea al inicio al final o entre.

```

#include <iostream>
#include <stdio.h>
#include <string>
#include <stdlib.h>
#include "NodoSnake.h"

typedef NodoSnake *pnodo;

class ListaEnlazada{
public:
    ListaEnlazada();
    void insertar(int);
    void insertarPrimero(int);
    bool listaVacia();
    char* crearLinea();
    int numeroElementos();
}

```

```

private:
    pnodo primero;
};

ListaEnlazada::ListaEnlazada() {
    primero=NULL;
}

void ListaEnlazada::insertar(int num) {
    pnodo anterior;
    if(listaVacia()) {
        primero = new NodoSnake(num,primero);
    } else {
        anterior = primero;
        primero = new NodoSnake(num,primero);
        primero->siguiente =anterior;
    }
}

bool ListaEnlazada::listaVacia() {
    return primero == NULL;
}

char* ListaEnlazada::crearLinea() {
    string serpiente;
    char* temp;
    pnodo aux;
    aux = primero;
    while(aux) {
        temp=(char*)malloc(1*sizeof(int));
        sprintf(temp,"%d",aux->getNumero());
        serpiente.append(temp);
        aux = aux->siguiente;
        free(temp);
    }
    temp=(char*)malloc(serpiente.length()*sizeof(int));
    sprintf(temp,"%s",serpiente.c_str());
    return temp;
}

void ListaEnlazada::insertarPrimero(int num) {
    pnodo aux;
    aux=primero;
    if(listaVacia()) {
        primero = new NodoSnake(num,primero);
    } else {
        while(aux->siguiente) {
            aux = aux->siguiente;
        }
        aux->siguiente=new NodoSnake(num,aux->siguiente);
    }
}

int ListaEnlazada::numeroElementos() {
    pnodo aux=primero;
    int i=1;
}

```

```

        while(aux->siguiente){
            i++;
            aux=aux->siguiente;
        }
        return i;
    }
}

```

- “Sanake” Es el main y es ahí en donde llaman a las funciones y las clases para que pueda funcionar el programa. A continuación se detalla más específico las funcionalidades dentro del main.

```

#include <windows.h>
#include <iostream>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#include "ListaEnlazada.h"
#include "ListaComida.h"

void OcultaCursor() {
    CONSOLE_CURSOR_INFO cci = {100, FALSE};

    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cci);
}

int main()
{
    srand (time(NULL));
    int xCuerpo=30;
    int xComida, yComida=5, valorComida, control=0,elementos=0,
gameover=yComida;
    ListaEnlazada listE;//Lista declara para el cuerpo
    ListaComida list; //lista declarada para los números que caen
    listE.insertarPrimero(0);//Inicio del cuerpo
    OcultaCursor(); //se oculta el cursor para que no se vea feo
    gotoxy(xCuerpo,23);
    printf("%s",listE.crearLinea()); // se usa una funcion para transformar
los numeros en una linea char* para facilidad de impresion
    char c;
    while(c!=27&&gameover<=23){ //condicion de salida persionar esc y
perdida sii los numeros sobreapan cierto punto, ademas que repite las
acciones
        while(!kbhit()){//condicion para que se mueva el juego mientras
no se haga nada
            system("cls"); //se borra la pantalla ne cada
iteracion para reimprimir los nueros con el efecto de caida
            gotoxy(xCuerpo,23); //posivion del cuerpo
            printf("%s",listE.crearLinea());
            valorComida=rand()%10; //valor aleatorio de la comida
            xComida=rand()%77+1; //posicion aleatoria de la
comida
            list.insertarPrimero(valorComida,yComida,xComida); //se
inserta la comida con posicion y valor dey
            elementos++; //se suma la cuenta de elementos
            list.imprimirComida(elementos); //se indica cuantos
elementos se puede imprimir ya que se basa en eso apra las posiciones
        }
    }
}

```

```

        if(list.comprobarElementos(xComida,23,elementos,listE.numeroElementos())
        ){ //comprueba si la posicion de el cuerpo y la comida mas inferior es la
        misma

            listE.insertarPrimero(list.quitarElemento());//elimina el ultimo
            elemento de la comida y lo agrega al cuerpo
            elementos--;//resta un elemento
        }
        gameover=yComida+(elementos*2);//comprobacion de game over
        Sleep(150);//Timer para la generacion de elementos
    }

    c=getch();//getch y switch para los movimientos
    switch(c){

        case 75://movimiento de izquierda y reimpresion de cuerpo
            xCuerpo-=1;
            if(xCuerpo<1) xCuerpo=1;
            gotoxy(xCuerpo,23);
            printf("%s",listE.crearLinea());
            gotoxy(xCuerpo+1,23);
            printf(" ");
            break;

        case 77://movimiento de derecha y reimpresion del cuerpo
            xCuerpo+=1;
            if(xCuerpo>78) xCuerpo=78;
            gotoxy(xCuerpo,23);
            printf("%s",listE.crearLinea());
            gotoxy(xCuerpo-1,23);
            printf(" ");
            break;

    }

    if(list.comprobarElementos(xComida,23,elementos,listE.numeroElementos())){//comprobacion en caso de movimientos
        listE.insertarPrimero(list.quitarElemento());
        elementos--;
    }
}

return 0;
}

```

Ejecución de la aplicación

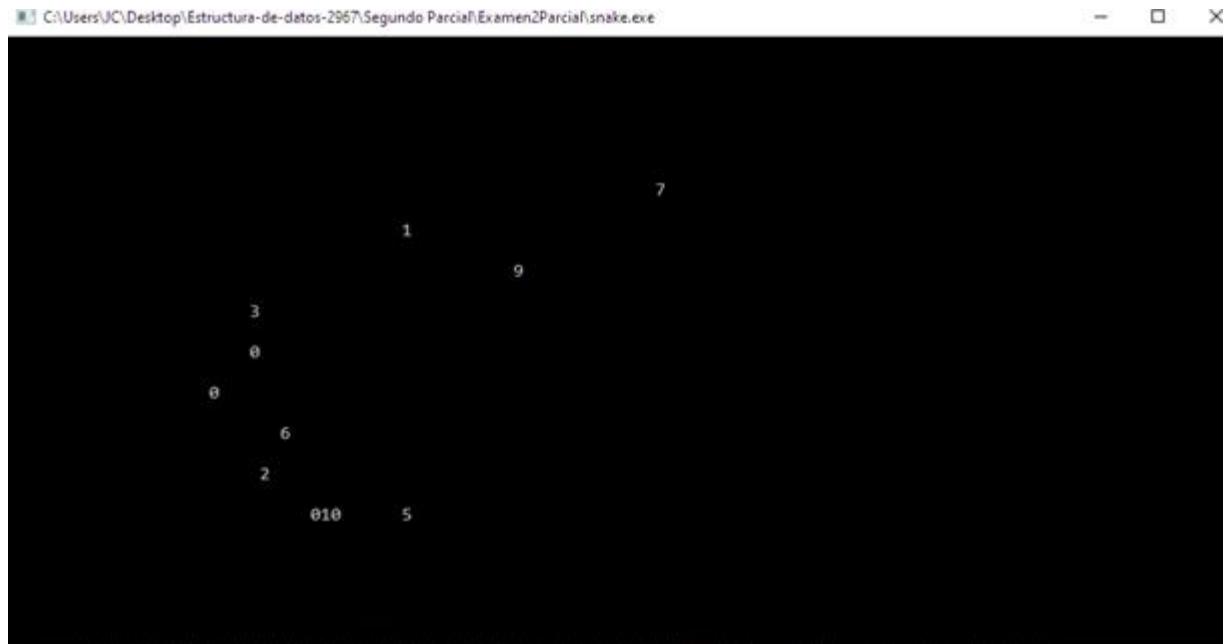


Figura 1. Los números caen de forma aleatoria cada cierto tiempo determinado

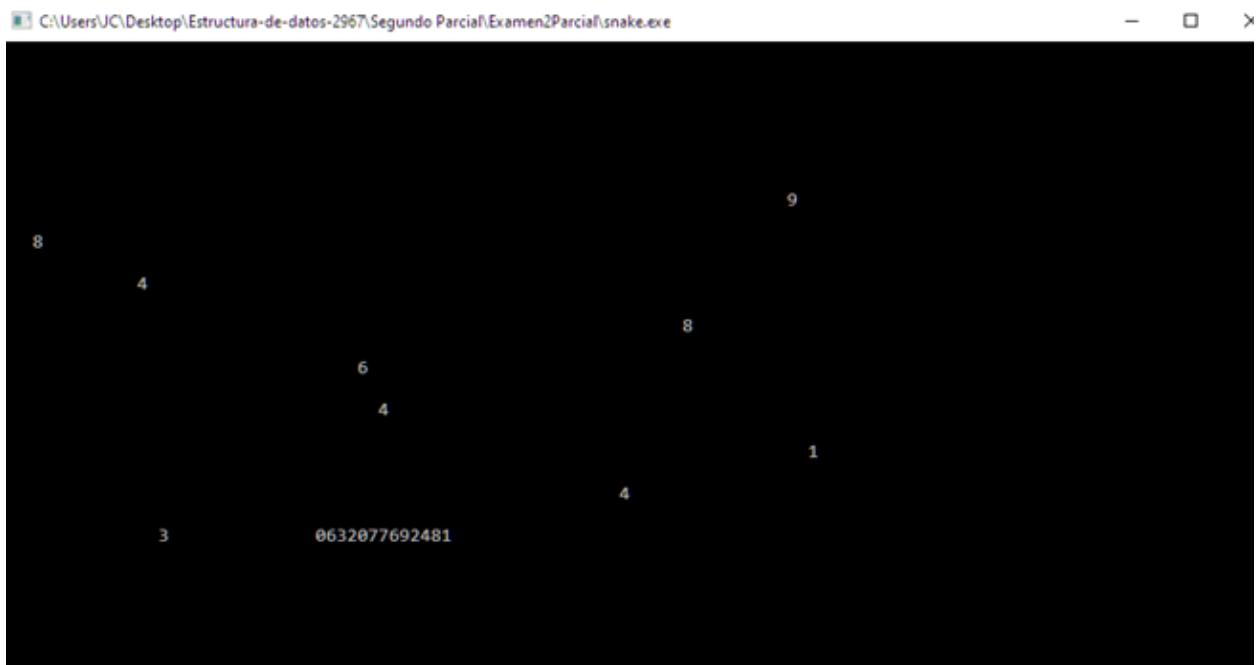


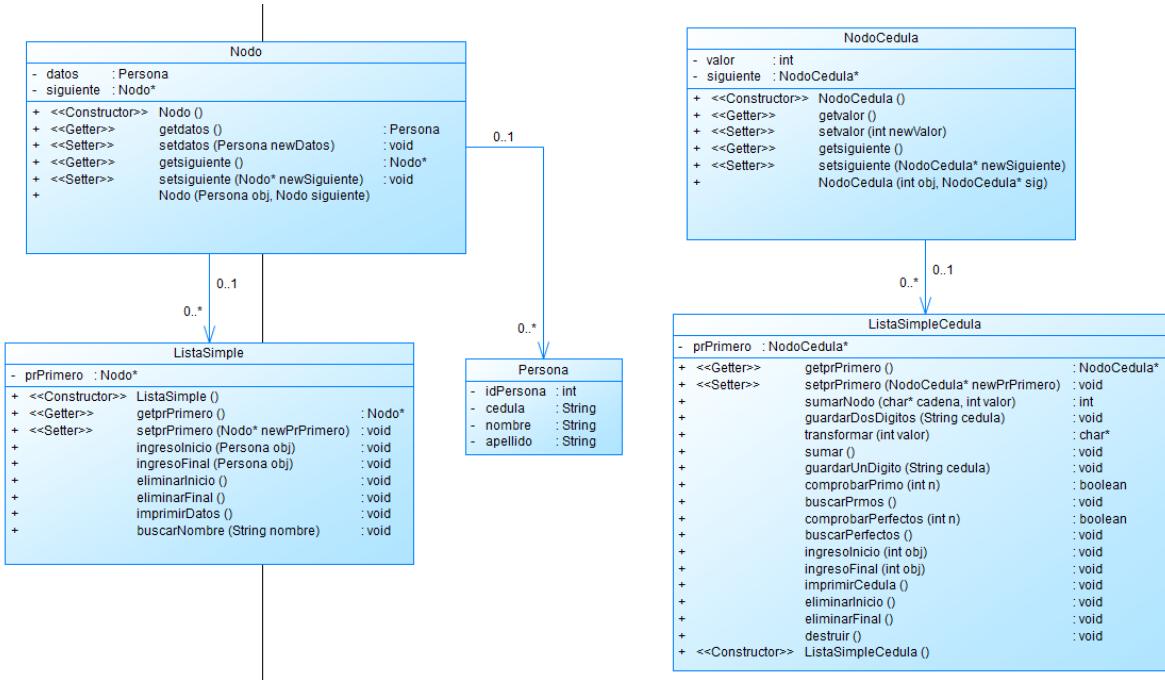
Figura 2. Los números que caen son atrapados en una lista

Suma de a Dos dígitos, Primos

Descripción

La aplicación consiste en el ingreso de datos de una persona del cual el los números de la cédula se extraen la suma de dos dígitos, se comprueba si existen números perfectos y la cantidad de número primos que existan.

Modelado



Código

```

Clase Main
#include <iostream>
#include "Persona.cpp"
#include "ingreso.h"
#include "Nodo.cpp"
#include "NodoCedula.cpp"
#include "ListaSimple.cpp"
#include "ListaSimpleCedula.cpp"

using namespace std;

int main(){
    Persona persona;
    Ingreso leer;
    ListaSimple *lista = new ListaSimple();

```

```

ListaSimpleCedula *listacedula = new ListaSimpleCedula();
string opc,buscar;
int id = 1;
do{
    cout<<"\t\tBienvenido\n";
    persona.setNombre(leer.ingresarString("Ingrese nombre: "));
    persona.setApellido(leer.ingresarString("Ingresar apellido: "));
    persona.setCedula(leer.ingresar10Digitos("Ingresar cedula: "));
    persona.setIdPersona(id);
    lista->ingresoFinal(persona);
    cout<<endl;
    listacedula->guardarDosDigitos(persona.getCedula());
    listacedula->guardarUnDigito(persona.getCedula());
    listacedula->buscarPrimos();
    listacedula->guardarUnDigito(persona.getCedula());
    listacedula->buscarPerfectos();
    id++;
    opc = leer.ingresarString("Desea ingresar otra persona (s/n): ");
    system("cls");
}while(opc == "s" || opc == "S");
system("pause");
//lista->imprimirDatos();
lista->~ListaSimple();
return 0;
}
Clase ListaSimpleCedula
#include <sstream>
#include "ListaSimpleCedula.h"
#include "NodoCedula.h"
#pragma once
void ListaSimpleCedula::ingresoInicio(int obj){
    prPrimero = new NodoCedula(obj,prPrimero);
}

void ListaSimpleCedula::ingresoFinal(int obj){
    NodoCedula *p = prPrimero;
    NodoCedula *nuevo;

    if(p == NULL){
        ingresoInicio(obj);
    }else{
        while(p->getSiguiente() != NULL){
            p = p->getSiguiente();
        }
        nuevo = new NodoCedula(obj, NULL);
        p->setSiguiente(nuevo);
    }
}

char* ListaSimpleCedula::transformar(int valor){
    stringstream a;
    a<<valor;
    return (char*)a.str().c_str();
}

int ListaSimpleCedula::sumarNodo(char* cadena,int valor){
    valor = *(cadena+0) - '0' + *(cadena+1) - '0';
}

```

```

    if(valor < 10){
        return valor;
    }else{
        sumarNodo(transformar(valor), 0);
    }
}

void ListaSimpleCedula::sumar(){
    NodoCedula *p = prPrimero;
    cout<<"La suma de cada dos digitos es: "<<endl;
    while(p != NULL){
        if(p->getValor()>9){
            int valor = sumarNodo(transformar(p->getValor()), 0);
            cout<<valor<< " ";
        }else{
            cout<<p->getValor()<< " ";
        }
        p = p->getSiguiente();
    }
    cout<<endl<<endl;
}

void ListaSimpleCedula::imprimirCedula(void)
{
    NodoCedula *p = prPrimero;
    cout<<"Cedula"<<endl;
    while(p != NULL){
        cout<<p->getValor()<< " ";
        p = p->getSiguiente();
    }
    cout<<endl;
}
void ListaSimpleCedula::eliminarInicio(){
    NodoCedula *p = prPrimero;
    if(p == NULL){
        cout<<"Nada que eliminar"<<endl;
        return;
    }else{
        prPrimero = p->getSiguiente();
    }
    delete p;
}

void ListaSimpleCedula::eliminarFinal(){
    NodoCedula *p = prPrimero;
    NodoCedula *aux;
    if(p == NULL){
        cout<<"Nada que eliminar"<<endl;
        return;
    }
    if(p->getSiguiente() == NULL){
        eliminarInicio();
        //cout<<"Nada que eliminar"<<endl;
        return;
    }else{
        while(p->getSiguiente() != NULL){
            aux = p;

```

```

        p = p->getSiguiente();
    }
    aux->setSiguiente(NULL);
}
delete p;
}

void ListaSimpleCedula::setPrPrimero(NodoCedula *primero) {
    prPrimero = primero;
}

NodoCedula* ListaSimpleCedula::getPrPrimero() {
    return prPrimero;
}

ListaSimpleCedula::ListaSimpleCedula() {
    prPrimero = NULL;
}

void ListaSimpleCedula::destruir() {
    NodoCedula *p = prPrimero;
    NodoCedula *actual;
    while(p != NULL) {
        actual = p->getSiguiente();
        delete p;
        p = actual;
    }
    prPrimero = NULL;
}

void ListaSimpleCedula::guardarDosDigitos(string cedula) {
    string cadena = "";
    int cont=0,aux=0;
    for(int i = aux; i<10; i++) {

        if(cont != 2){
            cadena += cedula.at(i);
            cont++;
        }

        if(cont == 2){
            ingresoFinal(atoi(cadena.c_str()));
            cadena = "";
            cont = 0;
        }
    }
    imprimirCedula();
    sumar();
    destruir();
}

void ListaSimpleCedula::buscarPrimos() {
    NodoCedula *p = prPrimero;
    int cont = 0;
    while(p != NULL) {
        if(comprobarPrimo(p->getValor())){

```

```

        cont++;
        cout<<p->getValor()<<" ";
    }
    p = p->getSiguiente();
}
cout<<"Existen "<<cont<<" Numeros primos"<<endl;
cout<<endl;
destruir();
}

bool ListaSimpleCedula::comprobarPrimo(int n)
{
    int acum=0;
    for(int i=1;i<=n;i++)
    {
        if(n%i==0)
        {
            acum++;
        }
    }

    if(acum == 2)
    {
        return true;
    }else{
        return false;
    }
}

void ListaSimpleCedula::guardarUnDigito(string cedula){
    for(int i = 0; i<10; i++){

        ingresoFinal(cedula.at(i)-'0');
    }
    imprimirCedula();
}

void ListaSimpleCedula::buscarPerfectos(){
    NodoCedula *p = prPrimero;
    int cont = 0;
    while(p != NULL){
        if(comprobarPerfecto(p->getValor())){
            cont++;
            cout<<p->getValor()<<" ";
        }
        p = p->getSiguiente();
    }
    cout<<"Existen "<<cont<<" Numeros perfectos"<<endl;
    cout<<endl;
    destruir();
}

bool ListaSimpleCedula::comprobarPerfecto(int n)
{
    int acum=0;
    if(n == 0)
        return false;

```

```

for(int i=1; i<n;i++)
{
    if(n%i==0)
    {
        acum=acum+i;
    }
}

if(acum==n)
{
    return true;
}
else{
    return false;
}
}

```

Ejecución

```

                Bienvenido
Ingresar nombre: jonathan
Ingresar apellido: picado
Ingresar cedula: 1715005581

Cedula
17 15 0 55 81
La suma de cada dos digitos es:
8 6 0 1 9

Cedula
1 7 1 5 0 0 5 5 8 1
7 5 5 5 Existen 4 Numeros primos

Cedula
1 7 1 5 0 0 5 5 8 1
Existen 0 Numeros perfectos

Desea ingresar otra persona (s/n):

```

Snake

Descripción

Juego realizado en C++ llamado “Snake” que utiliza como principal librería a la windows.h para poder simular la función gotoxy(). El juego consiste en una serpiente conformado por una matriz

de 0 y la comida que simula ser los números del 1 al 9, cada vez que la serpiente coma un número este se le sumará al cuerpo de la serpiente para poder ir creciendo constantemente, el juego termina cuando la serpiente choca con su propio cuerpo o cuando se excede de los límites permitidos.

Objetivo

Crear una lista de números que cada vez que la serpiente coma un número este se le añada tanto a su cuerpo como a la lista simple, en donde la inserción sea por cabeza, aplicando los conocimientos de una lista simple.

Modelo

ListaEnlazada	NodoSnake
<ul style="list-style-type: none"> - primero : pnodo* + insertar () : void + listaVacia () : bool + crearLinea () : char* + insertarPrimero () : void + <<Constructor>> ListaEnlazada () + <<Destructor>> ~ListaEnlazada () 	<ul style="list-style-type: none"> - numero : int - siguiente : int + <<Getter>> getSiguiete () : int + <<Setter>> setSiguiete (int newSiguiete) : void + <<Constructor>> NodoSnake () + <<Destructor>> ~NodoSnake ()

Código

- Clase ListaEnlazada.h

```
#include <iostream>
#include <stdio.h>
#include <string>
#include <stdlib.h>
#include "NodoSnake.h"

typedef NodoSnake *pnodo;

class ListaEnlazada{
    public:
        ListaEnlazada();
        void insertar(int);
        void insertarPrimero(int);
        bool listaVacia();
        char* crearLinea();

    private:
        pnodo primero;
};

ListaEnlazada::ListaEnlazada(){
    primero=NULL;
}

void ListaEnlazada::insertar(int num){
    pnodo anterior;
    if(listaVacia()) {
```

```

        primero = new NodoSnake(num,primero);
    } else {
        anterior = primero;
        primero = new NodoSnake(num,primero);
        primero->siguiente =anterior;
    }
}

bool ListaEnlazada::listavacia(){
    return primero == NULL;
}

char* ListaEnlazada::crearLinea(){
    string serpiente;
    char* temp;
    pnodo aux;
    aux = primero;
    while(aux) {
        temp=(char*)malloc(1*sizeof(int));
        sprintf(temp,"%d",aux->getNumero());
        serpiente.append(temp);
        aux = aux->siguiente;
        free(temp);
    }
    temp=(char*)malloc(serpiente.length()*sizeof(int));
    sprintf(temp,"%s",serpiente.c_str());
    return temp;
}

void ListaEnlazada::insertarPrimero(int num){
    pnodo aux;
    aux=primero;
    if(listavacia()) {
        primero = new NodoSnake(num,primero);
    } else {
        while(aux->siguiente) {
            aux = aux->siguiente;
        }
        aux->siguiente=new NodoSnake(num,aux->siguiente);
    }
}
● Clase NodoSnake.h
#include <iostream>
#include <stdio.h>
#include <string>

using namespace std;
class NodoSnake{
public:
    NodoSnake(int, NodoSnake* );
    int getNumero();
private:
    int numero;
    NodoSnake *siguiente;

    friend class ListaEnlazada;
};

```

```

NodoSnake::NodoSnake(int num, NodoSnake *sig=NULL) {
    numero=num;
    siguiente=sig;
}

int NodoSnake::getNumero() {
    return numero;
}
● Clase Snake.cpp
#include <windows.h>
#include <iostream>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#include "ListaEnlazada.h"

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

char tecla;

void gotoxy(int x, int y)
{
    HANDLE hCon;
    COORD dwPos;

    dwPos.X = x;
    dwPos.Y = y;
    hCon = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleCursorPosition(hCon,dwPos);
}

void OcultaCursor() {
    CONSOLE_CURSOR_INFO cci = {100, FALSE};

    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cci);
}

void pintar(){
    for(int i=2; i < 78; i++){
        gotoxy(i, 3); printf ("%c", 205);
        gotoxy(i, 23); printf ("%c", 205);
    }
    for(int v=4; v < 23; v++){
        gotoxy(2,v); printf ("%c", 186);
        gotoxy(77,v); printf ("%c", 186);
    }
    gotoxy(2,3); printf ("%c", 201);
    gotoxy(2,23); printf ("%c", 200);
    gotoxy(77,3); printf ("%c", 187);
    gotoxy(77,23); printf ("%c", 188);
}
void guardar_posicion(int *n, int *tam, int *x, int *y, int **cuerpo ){
    *(*cuerpo+*n)+0 = *x;
    *(*cuerpo+*n)+1 = *y;
}

```

```

*n=*n+1;
if(*n == *tam) *n = 1;
}
void dibujar_cuerpo(int *tam, int **cuerpo, char* line){

for(int i = 1; i < *tam; i++){
gotoxy(*(*(cuerpo+i)+0) , *(*(cuerpo+i)+1)); printf("%c",line[i-1]);
}
}
void borrar_cuerpo(int **cuerpo, int *n){
gotoxy(*(*(cuerpo+*n)+0) , *(*(cuerpo+*n)+1)); printf(" ");
}
void teclear(int *dir){
if(kbhit()){
tecla = getch();
switch(tecla){
case ARRIBA : if(*dir != 2) *dir = 1; break;
case ABAJO : if(*dir != 1) *dir = 2; break;
case DERECHA : if(*dir != 4) *dir = 3; break;
case IZQUIERDA : if(*dir != 3) *dir = 4; break;
}
}
}
void comida(int *x, int *y, int *xc, int *yc, int *tam, ListaEnlazada *listE,
int *com)
{
if(*x == *xc && *y == *yc)
{
srand (time(NULL));
listE->insertar(*com);
*xc = (rand() % 73) + 4;
*yc = (rand() % 19) + 4;
*com = rand()%10;
*tam=*tam+1;
gotoxy(*xc, *yc); printf("%d", *com);
}
}
bool game_over(int *tam, int *x, int *y, int** cuerpo)
{
if(*y == 3 || *y == 23 || *x == 2 || *x == 77) return false;
for(int j = *tam - 1; j > 0; j--){
if(*(*(cuerpo+j)+0) == *x && *(*(cuerpo+j)+1) == *y)
return false;
}
return true;
}
int main()
{
ListaEnlazada listE;
ListaEnlazada *listEn=&listE;
int **cuerpo;
int n = 1, tam = 10, dir = 3, com, x = 10, y = 12, xc = 30, yc = 15,
velocidad = 60;
int *n1=&n, *tam1=&tam, *dir1=&dir, *com1=&com, *x1=&x, *y1=&y,
*xcl=&xc, *ycl=&yc;
cuerpo =(int **)malloc(200*sizeof(int *));
for(int j=0;j<200;j++)
*(cuerpo+j)=(int *)malloc(2*sizeof(int));
}

```

```

        for(int i=0; i<10; i++) {
            listE.insertar(0);
        }
        srand (time(NULL));
        com=rand()%10;
        OcultaCursor();

        pintar();
        gotoxy(xc, yc); printf("%d", com);

    while(tecla != ESC && game_over(tam1, x1, y1, cuerpo))
    {
        borrar_cuerpo(cuerpo, n1);
        guardar_posicion(n1, tam1, x1, y1, cuerpo);
        dibujar_cuerpo(tam1, cuerpo, listE.crearLinea());
        comida(x1, y1, xcl,ycl, tam1, listEn, com1);
        teclear(dir1);
        teclear(dir1);

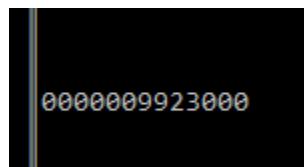
        if(*dir1 == 1) *y1=*y1-1;
        if(*dir1 == 2) *y1=*y1+1;
        if(*dir1 == 3) *x1=*x1+1;
        if(*dir1 == 4) *x1=*x1-1;

        Sleep(velocidad);
    }
    pintar();
    return 0;
}

```

Ejecución

La siguiente figura demuestra la inserción de los números al cuerpo de la serpiente.



La siguiente figura demuestra el fin del juego porque la serpiente se salió de los límites permitidos.

```

0000000000
8

-----
process exited after 4.577 seconds with return value 0
presione una tecla para continuar . .

```

Pilas

Descripción

La siguiente aplicación tiene como objetivo ingresar una expresión algebraica de manera que esta se transforme a notación polaca inversa mediante acoplamientos de pilas.

Objetivo de la aplicación

Utilizar los métodos de inserción para una Pila.

Modelado

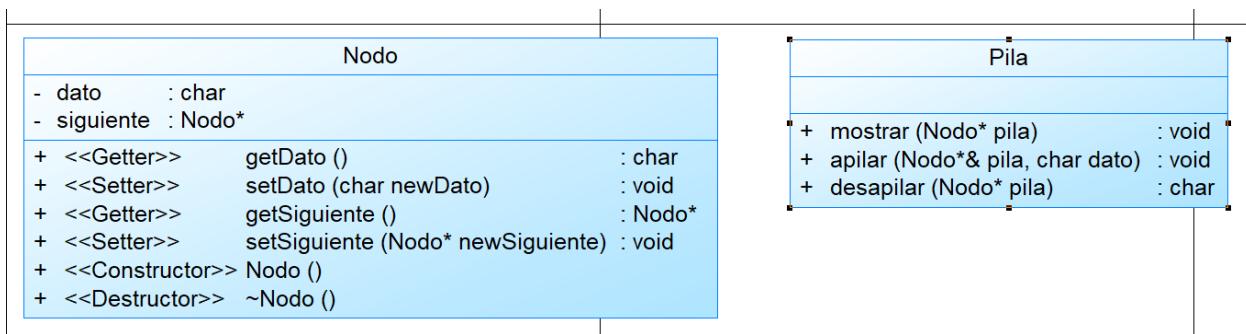


Gráfico 1. Modelo del aplicativo

Código de la aplicación

Main

```
#include <cstdlib>
#include <iostream>
#include <string.h>
#include "Nodo.h"
#include "Pila.cpp"
#include "IngresoBloqueTeclas.h"
#include "TransformacionString.h"
using namespace std;

int main() {
    Nodo *pila = NULL;
    char valor, c;
    Ingreso ig;
    string ecuacion;
    Pila opila;
    cout << "Ingrese un ecuacion matematica" << endl;
    ecuacion = ig.ingresarNumeros(&valor);
    for (int i = 0; i < ecuacion.length(); i++) {
        opila.apilar(pila, ecuacion.at(i));
    }
    opila.mostrar(pila, ecuacion.length());
    system("pause");
    return 0;
}
```

Clase Pila

```
#include "Pila.h"
#include <iostream>
using namespace std;

void Pila::mostrar(Nodo* pila, int n) {
    Nodo *actual = new Nodo();
    actual = pila;
    int p = 0;
    while (p != n) {
        cout << actual->getdato() << endl;
        actual = actual->getsiguiente();
        p++;
    }
}

void Pila::apilar(Nodo*& pila, char dato) {
    Nodo *nuevo_nodo = new Nodo();
    nuevo_nodo->setdato(dato);
    Nodo *aux;
    aux = pila;
    pila = nuevo_nodo;
    nuevo_nodo->setsiguiente(aux);
```

```

}

char Pila::desapilar(Nodo*& pila) {

}

Clase Nodo
#if !defined(__Nodo_Nodo_h)
#define __Nodo_Nodo_h

class Nodo {
public:
    char getData(void);
    void setData(char newData);
    Nodo* getSiguiente(void);
    void setSiguiente(Nodo* newSiguiente);
    Nodo();
    ~Nodo();

protected:
private:
    char dato;
    Nodo* siguiente;

};

#endif

```

Ejecución del aplicativo

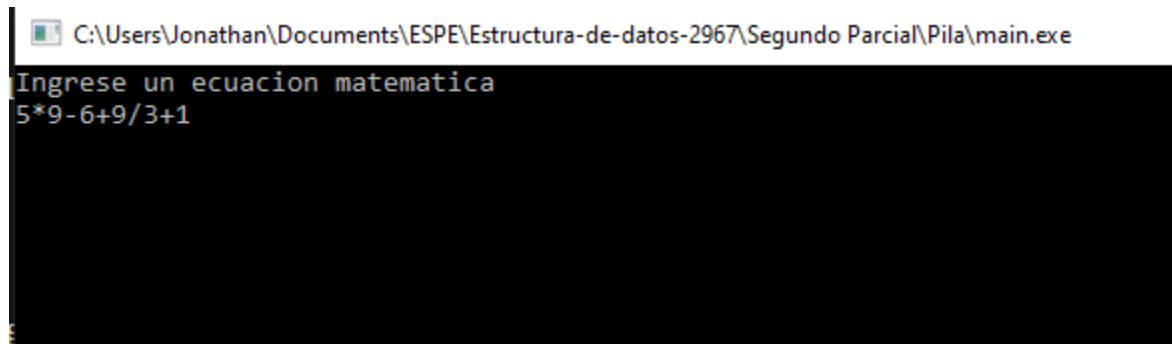


Gráfico 2. Ingreso de la expresión matemática por parte del usuario.

```

C:\Users\Jonathan\Documents\ESPE\Estructura-de-datos-2967\S
1
+
3
/
9
+
5
-
9
*
5
Presione una tecla para continuar . .

```

Gráfico 3. Resultado Final.

Colas

Descripción

La siguiente aplicación tiene como objetivo mostrar el funcionamiento de proceso de las colas en lenguaje c++.

Objetivo de la aplicación

Utilizar algoritmo de insercion y eliminacion para colas.

Modelado

Elemento	Cola
<ul style="list-style-type: none"> - dato : int - sig : Elemento* + <<Constructor>> Elemento () + <<Destructor>> ~Elemento () 	<ul style="list-style-type: none"> - tope : Elemento* - cola : Elemento* - total : int - Attribute_4 : int + <<Getter>> getElemTotales () : int + <<Setter>> setElemTotales (int newAttribute_4) : void + <<Constructor>> Cola () + <<Destructor>> ~Cola () + Eliminar () : int

Codigo

Funcion Principal

```

#include <iostream>
#include <string>
#include "Elemento.h"
#include "Cola.h"

using namespace std;

int main() {
    Cola* stack = new Cola();
    stack->total = 15;
    cout << "Llenando la cola" << endl;
    for (int i = 0; i < 10; i++) {
        Elemento* node = new Elemento();
        node->dato = i + 1;
        cout << "insertando: " << i + 1 << endl;
        stack->push(node);
    }
    cout << "Vaciando cola" << endl;
    Elemento * c = stack->pop();
    while (c != NULL) {
        cout << c->dato << endl;
        c = stack->pop();
    }

    return 0;
}
Clase Cola
#include "Elemento.h"
#include <iostream>
using namespace std;
class Cola {
public:
    Elemento* tope;
    Elemento* cola;
    int total;
    int elementos_existentes = 0;
    Cola() : tope(NULL), cola(NULL), total(0) {}

    Cola(int n) {
        this->tope = NULL;
        this->cola = NULL;
        this->total = n;
    }

    Elemento* pop() {
        if (this->elementos_existentes > 0) {
            Elemento* sacarme = this->tope;
            this->tope = sacarme->siguiente;
            this->elementos_existentes -= 1;
            return sacarme;
        } else {
            cout << "Nada para sacar" << endl;
            return NULL;
        }
    }
}

```

```

void push(Elemento* nuevo) {
    if (this->total >= this->elementos_existentes) {
        if (this->elementos_existentes == 0) {
            this->tope = this->cola = nuevo;
        } else {
            this->cola->siguiente = nuevo;
            this->cola = nuevo;
        }
        this->elementos_existentes += 1;
    } else {
        cout << "Error! cola llena" << endl;
    }
}
Clase Elemento
#include <iostream>
class Elemento {
public:
    int dato;
    Elemento* siguiente;
    Elemento() : dato(0), siguiente(NULL) {
    }
};

```

Ejecución del aplicativo

```
C:\Users\Jonathan\Documents\ESPE\Estructura-de-datos-2967\Segundo Parcial
Llenando la cola
insertando: 1
insertando: 2
insertando: 3
insertando: 4
insertando: 5
insertando: 6
insertando: 7
insertando: 8
insertando: 9
insertando: 10
Vaciando cola
1
2
3
4
5
6
7
8
9
10
Nada para sacar
-----
Process exited after 0.5203 seconds with return value 0
Presione una tecla para continuar . . .
```

Gráfico 1. Demostración de insercion y eliminacion en la cola.

Juego de Palabras

Descripción

La siguiente aplicación consiste en el clásico “Juego de la oración” en el que dos o mas personas pueden jugarlo. Se inicia por un participante diciendo una palabra, el segundo debe decir la palabra del participante anterior y adicionar una palabra más, el tercero debe decir las dos palabras anteriores más una nueva y así sucesivamente con todos los participantes.

Si un participante olvida la oración o la dice incorrectamente, queda eliminado.

El Juego finaliza cuando queda solo un participante.

Objetivo de la aplicación

Implementar el “Juego de la oración” previamente descrita, mediante la utilización de listas simples y sus procesos, para el correcto funcionamiento del juego.

Modelo



Figura 1: Modelado

Código de la aplicación

- La clase "ingreso.h" permite la validación de ingresar solamente datos de tipo "string" en la ejecución del programa.

```

#include <stdio.h>
#include <iostream>
#include <string.h>
#include <sstream>
#include <stdlib.h>
using namespace std;
class Ingreso{
public:
    char* ingresar(char* );
    string ingresarSoloTexto(string);
    int ingresarEntero(char * );
    float ingresarFlotante(char * );
    bool validarCedula(int cedula);
};

char* ingresar(char* msg){
    char* texto;
    cout<<msg<<endl;
    cin>>texto;
    return texto;
}
string ingresarSoloTexto(string msg){
    float valor;
    string texto;
    string res;
    while (1)
    {
        bool is_valid = true;
        cout << msg << endl;
        getline(cin, texto);
        try{
            for (size_t i = 0; i < texto.length(); i++) {
                if (!isalpha(texto[i])) {
                    if(texto[i] == ' '){
                        continue;
                    }
                    else{
                        is_valid = false;
                    }
                }
            }
            if(is_valid)
                break;
        }
        catch(exception e)
        {
            cout << "Error: " << e.what() << endl;
        }
    }
    return res;
}

```

```

        }else{
            cout << "Se debe ingresar solo letras\n";
            is_valid = false;
            break;
        }
    }
}
}catch(exception e){
    cout<<"error";
}
if (is_valid){
    res = texto.c_str();
    break;
}
}
return res;
}
float ingresarFlotante(char *msg){
    float valor;
    string numero;
    while (1)
    {
        bool is_valid = true;
        cout << msg << endl;
        cin >> numero;
        try{
            for (size_t i = 0; i < numero.length(); i++) {
                if (!isdigit(numero[i])) {
                    if(!(numero[i]=='.'))
                        cout << "Se debe ingresar numeros\n";
                    is_valid = false;
                    break;
                }
            }
        }catch(exception e){
            cout<<"error";
        }

        if (is_valid){
            stringstream geek(numero);
            geek>>valor;
            break;
        }
    }
    return valor;
}

int ingresarEntero(char *msg){
    int valor;
    string numero;
    char *res;

    while (1)
    {
        bool is_valid = true;
        cout << msg << endl;

```

```

        cin >> numero;
    try{
        for (size_t i = 0; i < numero.length(); i++) {

            if (!isdigit(numero[i])) {

                cout << "Se debe ingresar numeros\n";
                is_valid = false;
                break;

            }
        }
        }catch(exception e){
            cout<<"error";
        }

        if (is_valid){
            res = (char *)numero.c_str();
            valor=atoi(res);
            /*stringstream geek(numero);
            geek>>valor;*/
            break;
        }
    }
    return valor;
}

bool validarCedula(int cedula){
    int ced,pares,impares,total,dec=0;
    int primerDigito;
    int segundoDigito;
    int tercerDigito;
    int cuartoDigito;
    int quintoDigito;
    int sextoDigito;
    int septimoDigito;
    int octavoDigito;
    int novenoDigito;
    int decimoDigito;
    int k;

//cout<<"Ingrese su cedula: "<<endl;
//cin>>cedula;

ced=cedula;

primerDigito= cedula / 1000000000;
cedula=cedula-(primerDigito * 1000000000);
segundoDigito= cedula / 100000000;
cedula=cedula-(segundoDigito * 100000000);
tercerDigito= cedula / 10000000;
cedula=cedula-(tercerDigito * 10000000);
cuartoDigito= cedula / 1000000;
cedula=cedula-(cuartoDigito * 1000000);
quintoDigito= cedula / 100000;
cedula=cedula-(quintoDigito * 100000);
sextodigito= cedula / 10000;

```

```

cedula=cedula-(sextodigito * 10000);
septimodigito= cedula / 1000;
cedula=cedula-(septimodigito * 1000);
octavodigito= cedula / 100;
cedula=cedula-(octavodigito * 100);
novenodigito= cedula / 10;
cedula=cedula-(novenodigito * 10);
decimodigito= cedula / 1;
cedula=cedula-(decimodigito * 1);

if (cedula>2400000000){
    cout<<"Numero de cedula invalido."<<endl;
    return false;
}else{

    pares= segundoDigito + cuartoDigito + sextodigito + octavodigito;

    primerDigito= primerDigito * 2;
    if (primerDigito > 9){
        primerDigito= primerDigito % 10 + primerDigito / 10;
    }

    tercerDigito= tercerDigito * 2;
    if (tercerDigito > 9){
        tercerDigito= tercerDigito % 10 + tercerDigito / 10;
    }

    quintoDigito= quintoDigito * 2;
    if (quintoDigito > 9){
        quintoDigito= quintoDigito % 10 + quintoDigito / 10;
    }

    septimodigito= septimodigito * 2;
    if (septimodigito > 9){
        septimodigito= septimodigito % 10 + septimodigito / 10;
    }

    novenodigito= novenodigito * 2;
    if (novenodigito > 9){
        novenodigito= novenodigito % 10 + novenodigito / 10;
    }

    impares= primerDigito + tercerDigito + quintoDigito + septimodigito +
    novenodigito;

    total=pares+impares;

    while (dec-total != decimodigito && dec < total + 10){
        dec=dec+10;
    }

    if (dec-total == decimodigito){
        return true;
    }else {
        return false;
    }
}

```

```

        }
    }
}

● JuegoPalabras.cpp: Función principal que dirige nuestro aplicativo.

#include <iostream>
#include "Lista.cpp"
#include "ingreso.h"
using namespace std;

void presentacion();
void perdida();
string pedirFrase();

int main(){
    Lista historia;
    bool flag;
    bool juego=true;
    string frase="";
    string ultPalabra="";

    presentacion();

    historia.ingresarDatosFinal(pedirFrase());
    cout<<"palabra ingresada correctamente"<<endl;
    system("pause");

    do{
        system("cls");
        fflush(stdin);

        frase = ingresarSoloTexto("Ingresa la/las palabra(s) anteriores y
una palabra nueva: ");

        flag = historia.verificarCadena(frase);

        if(flag){
            cout<<"BIEN HECHO :D !"<<endl;
            ultPalabra = historia.obtenerUltimaPalabra(frase);
            historia.ingresarDatosFinal(ultPalabra);
            historia.imprimir();
            juego = true;
            ultPalabra.clear();
            system("pause");
        }
        else
        {
            cout << "Error te equivocaste!!" << endl;
            historia.imprimir();
            perdida();
            juego = false;
            historia.~Lista();
        }
    }while(juego);
    system("pause");
}

```

```

        return 0;
    }

void presentacion()
{
    cout<<"***** BIENVENIDO AL JUEGO DE LA HISTORIA *****"
*<<endl;
    cout<<"***** PERDISTE ----- FIN DE LA HISTORIA *****"
*<<endl;
    cout<<"*****"
*<<endl;
}

void perdida()
{
    cout<<"*****"
*<<endl;
    cout<<"***** PERDISTE ----- FIN DE LA HISTORIA *****"
*<<endl;
    cout<<"*****"
*<<endl;
}

string pedirFrase(){
    string frase="";
    frase = ingresarSoloTexto("Ingresa nueva palabra: ");
    return frase;
}

```

- En la clase "Lista.h" podemos observar la declaración de atributos y métodos a usarse y la clase Nodo.

```
#include <iostream>
```

```

using namespace std;

class Nodo
{
private:
    string palabra;
    Nodo *siguiente;

    friend class Lista;
public:
    void setPalabra(string newPalabra){
        palabra=newPalabra;
    }
    string getPalabra(){
        return palabra;
    }
};

class Lista

```

```

{
    private:
        Nodo *primero;
    public:
        Lista();
        void ingresarDatoFinal(string palabra);
        void borrarPrimerElemento();
        bool verificarCadena(string palabra);
        string obtenerUltimaPalabra(string frase);
        ~Lista();
        Nodo* getPrimero();
        void imprimir();
};

● En la clase "Lista.cpp" se encuentra los procesos de cada método declarado en "Lista.h"
- Nodo* Lista::getPrimero(): Método get para obtener el primer Nodo.
- void Lista::ingresarDatoFinal(string palabra): Método de listas simples para el ingreso por cola de una palabra
- void Lista::borrarPrimerElemento(): Método para borrar el primer elemento de la lista simple.
- bool Lista::verificarCadena(string frase): Verifica si la oración digitada por usuario pertenece a cada elemento de la lista.
- string Lista::obtenerUltimaPalabra(string frase): retorna el valor de la ultima palabra digitada en la oración del ultimo participante para posteriormente agregarla a la lista.
- void Lista::imprimir(): Imprime los elementos de la lista.
- Lista::~Lista(): Destructor de la clase Lista.
- Lista::Lista(): Constructor de la clase

#include "Lista.h"
#include <string.h>

using namespace std;

Lista::Lista()
{
    primero=NULL;
}

Nodo* Lista::getPrimero()
{
    return primero;
}

void Lista::ingresarDatoFinal(string palabra)
{
    Nodo *temp = new Nodo;
    Nodo *aux = primero;
    temp->setPalabra(palabra);
    temp->siguiente=NULL;

    if(aux==NULL)
    {
        primero= temp;
    }
    else
    {
        while(siguiente!=NULL)
        {
            aux=siguiente;
            siguiente=siguiente->siguiente;
        }
        aux->siguiente=temp;
    }
}

```

```

        }
    else
    {
        while(aux->siguiente != NULL)
        {
            aux=aux->siguiente;
        }
        aux->siguiente=temp;
    }

}

void Lista::borrarPrimerElemento()
{
    Nodo *aux = primero;

    if(aux == NULL)
    {
        cout << "No existen elementos para borrar" << endl;
    }
    else
    {
        Nodo *temp = primero->siguiente;
        delete primero;

        primero=temp;
    }
}

bool Lista::verificarCadena(string frase){
    Nodo* recorredor= primero;

    while(recorredor->siguiente != NULL)
    {
        int posicion = frase.find(" ");
        string palabra = frase.substr(0,posicion);

        if(!(recorredor->getPalabra().compare(palabra) == 0))
        {
            return false;
        }
        else
        {
            recorredor = recorredor->siguiente;
        }

        frase = frase.substr(posicion+1,frase.length()-1);
    }

    return true;
}

string Lista::obtenerUltimaPalabra(string frase)

```

```

{
    string ultPalabra="";

    // Returns first token
    char *token = strtok((char*)frase.c_str(), " ");

    while (token != NULL)
    {
        ultPalabra = token;
        token = strtok(NULL, " ");
    }

    return ultPalabra;
}

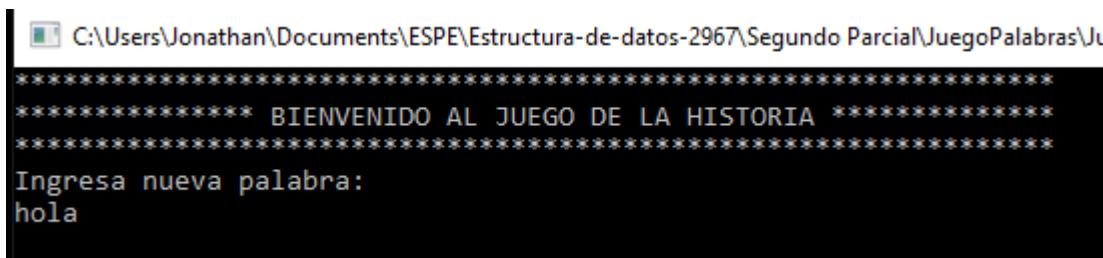
void Lista::imprimir()
{
    Nodo *temp = primero;

    while(temp!=NULL)
    {
        cout<<temp->getPalabra() <<, " <<endl;
        temp = temp->siguiente;
    }
    delete temp;
}

Lista::~Lista()
{
    while(primerоР!=NULL) {
        borrarPrimerElemento();
    }
}

```

Ejecución del aplicativo

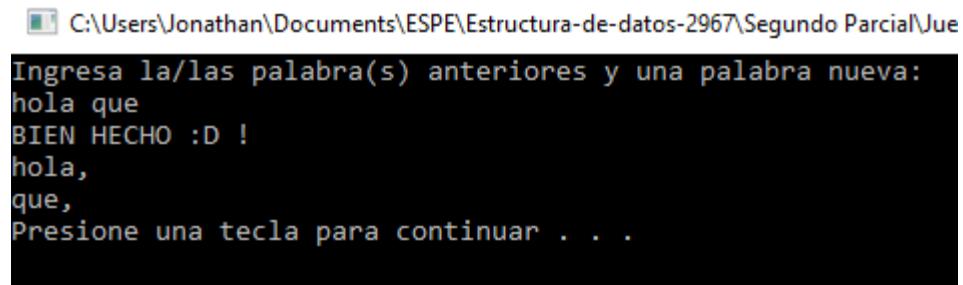


```

C:\Users\Jonathan\Documents\ESPE\Estructura-de-datos-2967\Segundo Parcial\JuegoPalabras\Ju
*****
***** BIENVENIDO AL JUEGO DE LA HISTORIA *****
*****
Ingresa nueva palabra:
hola

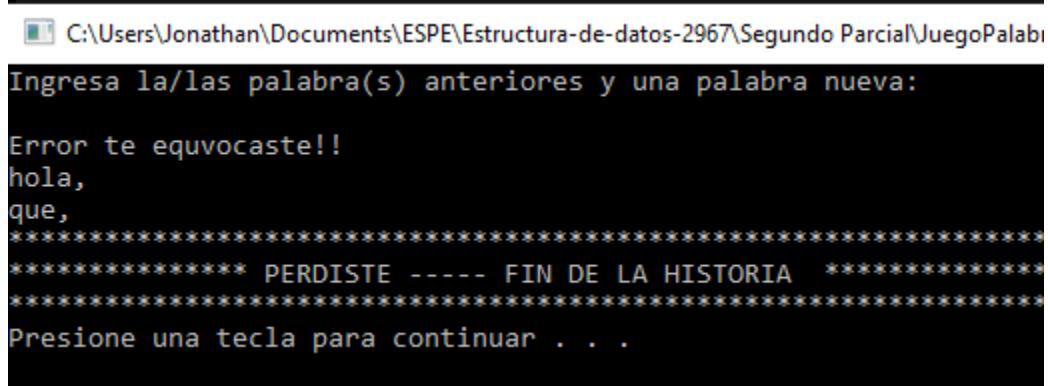
```

Figura 2: Ingreso de palabra nueva por usuario.



```
C:\Users\Jonathan\Documents\ESPE\Estructura-de-datos-2967\Segundo Parcial\Jue
Ingresa la/las palabra(s) anteriores y una palabra nueva:
hola que
BIEN HECHO :D !
hola,
que,
Presione una tecla para continuar . . .
```

Figura 3: Verificación si la oración digitada + una palabra nueva está correcta.



```
C:\Users\Jonathan\Documents\ESPE\Estructura-de-datos-2967\Segundo Parcial\JuegoPalabi
Ingresa la/las palabra(s) anteriores y una palabra nueva:
Error te equivocaste!!
hola,
que,
*****
***** PERDISTE ----- FIN DE LA HISTORIA *****
*****
Presione una tecla para continuar . . .
```

Figura 4: Verificación si la oración digitada + una palabra nueva está incorrecta.

Números primos

Descripción

El programa consiste en generar todos los números primos en forma constante y que de la misma manera se presenten en consola, el cual solo se detendrá si pulsamos la

tecla “ENTER” una vez detenido la aplicación se guardara en una lista.

Objetivo

Crear una función recursiva la cual se ejecute constantemente para ir generando los números primos y una vez terminada la ejecución se guarde en una lista posteriormente te crea un archivo (.txt) el cual contenga esta solución.

Modelo

Nodo	
- numero : int	
- siguiente : Nodo*	
+ <<Constructor>> Nodo ()	
+ <<Constructor>> Nodo (int newNumero, Nodo* newSiguiente)	
+ <<Destructor>> ~Nodo ()	
+ <<Getter>> getNumero () : int	
+ <<Setter>> setNumero (int newNumero) : void	
+ <<Getter>> getSiguiente () : Nodo*	
+ <<Setter>> setSiguiente (Nodo* newSiguiente) : void	

Lista	
- siguiente : Nodo*	
+ InsertarInicio (int numero) : void	
+ InsertarFinal (int numero) : void	
+ Imprimir () : void	
+ vacio () : bool	

NumeroPrimo	
+ primo (int numero, int i) : bool	

Código

Lista.cpp

- En la clase "Lista.cpp" podemos observar la declaración de atributos y métodos a usarse y la clase Nodo.

```
#include<iostream>
#include "Lista.h"

void Lista::InsertarInicio(int numero)
{
    if(vacio()){
        siguiente = new Nodo(numero, NULL);
    }else{
        Nodo *aux = new Nodo();
        aux->setNumero(numero);
        aux->setSiguiente(siguiente);
        siguiente=aux;
    }
}
void Lista::InsertarFinal(int numero)
{
    if(vacio()){
        siguiente = new Nodo(numero, NULL);
    }else{
        Nodo *aux1 = new Nodo();
        aux1=siguiente;
        Nodo *aux2=new Nodo();
        while((aux1)!=NULL){
            aux2=aux1;
            aux1=aux1->getSiguiente();
        }
        Nodo *aux3=new Nodo(numero,NULL);
        aux2->setSiguiente(aux3);

    }
}
```

```

void Lista::Imprimir(void)
{
    Nodo *aux = new Nodo();
    aux = siguiente;
    int cont=1;
    int salto=10;
    if(!vacio()){

        while(aux != NULL){
            std::cout<<aux->getNumero()<<"\n";
            aux = aux->getSiguiente();
            cont++;
        }
    }else {
        std::cout << "Lista vacia" << std::endl;
    }
}

int Lista::vacio(void)
{
    if(siguiente == NULL)
        return true;
    else
        return false;
}

```

● En la “Lista.h” están declaradas todas las funciones a utilizar

```

#ifndef __NumeroPrimo2_Lista_h
#define __NumeroPrimo2_Lista_h

#include "Nodo.h"

class Lista
{
public:
    void InsertarInicio(int numero);
    void InsertarFinal(int numero);
    void Imprimir(void);
    int vacio(void);

protected:
private:
    Nodo *siguiente;
};


```

Nodo.cpp

- Se crean los nodos de la lista

```

#include "Nodo.h"

Nodo::Nodo()
{

```

```

}

Nodo::~Nodo()
{
    // TODO : implement
}
Nodo *Nodo::getSiguiente(void)
{
    return siguiente;
}

void Nodo::setSiguiente(Nodo *newSiguiente)
{
    siguiente = newSiguiente;
}
Nodo::Nodo(int newNumero, Nodo *newSiguiente)
{
    numero=newNumero;
    siguiente=newSiguiente;
}
int Nodo::getNumero(void)
{
    return numero;
}
void Nodo::setNumero(int newNumero)
{
    numero = newNumero;
}
Nodo.h
● Se declaran las funciones de la clase nodo que se van a utilizar
#ifndef __NumeroPrimo2_Nodo_h
#define __NumeroPrimo2_Nodo_h

class Nodo
{
public:
    Nodo();
    Nodo(int newNumero, Nodo *newSiguiente);
    ~Nodo();
    Nodo *getSiguiente(void);
    void setSiguiente(Nodo *newSiguiente);
    int getNumero(void);
    void setNumero(int newNumero);
protected:
private:
    int numero;
    Nodo *siguiente;
};

NumeroPrimo.cpp
● Se crea la función recursiva que será utilizada para generar los
números primos
#include "NumeroPrimo.h"
#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <pthread.h>
#include <conio.h>
bool NumeroPrimo::primo(int numero, int i)

```

```

{
    if(numero<=i){
        return true;
    }
    if(numero%i==0){
        return false;
    }else{
        return primo(numero,i+1);
    }
}

NumeroPrimo.h
● Se declara la función recursiva para ser utilizada
#ifndef __NumerosPrimos2_NumeroPrimo_h
#define __NumerosPrimos2_NumeroPrimo_h

#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <pthread.h>
#include <conio.h>

class NumeroPrimo
{
public:
    bool primo(int numero, int i);
protected:
private:
};

Main.cpp
● La clase donde se llama a todas las funciones para que el programa
funcione de manera correcta
#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <pthread.h>
#include <conio.h>
#include "NumeroPrimo.h"
#include "Lista.h"

using namespace std;

//Funciones Hilos
void *primoInfinito(void *dato);
void *pararPrimos(void *dato);

bool bandera = true;
Lista *listaNumerosPrimo= new Lista();

int main()
{
    cout<<"Numeros Primos"<<endl;
    pthread_t procesol;
    pthread_t proceso2;
    pthread_create(&procesol,NULL,&primoInfinito,NULL);
    pthread_create(&proceso2,NULL,&pararPrimos,NULL);
    pthread_join(procesol,NULL);
}

```

```

    pthread_join(proceso2,NULL) ;

cout<<"Imprecision lista:"<<endl;
    listaNumerosPrimo->Imprimir();
    return 0;
}

void *primoInfinito(void *dato){
    NumeroPrimo validarPrimo;
    int numero=2;
    fstream enter;

        enter.open("SolucionPrimos.txt",fstream::out); //para leer in,
para salir es out escribir
        enter<<"\t\tUniversidad de las Fuerzas Armadas ESPE\nEstructura
de datos\nNRC: 2967\nDocente: Ing. Fernando Solis\n\nSolucion Numeros
Primo"<<endl;
        while (bandera) {
            if(validarPrimo.primo(numero,2)){
                cout << numero<<endl;
                enter<<numero<<endl;
                listaNumerosPrimo->InsertarFinal(numero);
            }
            numero++;
            _sleep(40);
        }
        enter.close();
}
void *pararPrimos(void *dato){
    char salida;
    salida = getch();
    if(salida==13){
        bandera = false;
    }
}

```

Ejecución del aplicativo

```
Numeros Primos  
2  
3  
5  
7  
11  
13  
17  
19  
23  
29  
31  
37  
41  
43  
47  
53  
59
```

Figura 1 Se ve cómo se generan los números primos ne consola

```
Imprecision lista:  
2  
3  
5  
7  
11  
13  
17  
19  
23  
29  
31  
37  
41  
43  
47  
53  
59  
  
-----  
Process exited after 2.566 seconds with return value 0  
Presione una tecla para continuar . . .
```

Figura 2 Se presenta la lista que generó el programa con los números primos

```
Archivo Edición Formato Ver Ayuda
Universidad de las Fuerzas Armadas ESPE
Estructura de datos
NRC: 2967
Docente: Ing. Fernando Solis

Solucion Numeros Primo
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
```

Figura 3. El archivo (.txt) de la solución

Biblioteca

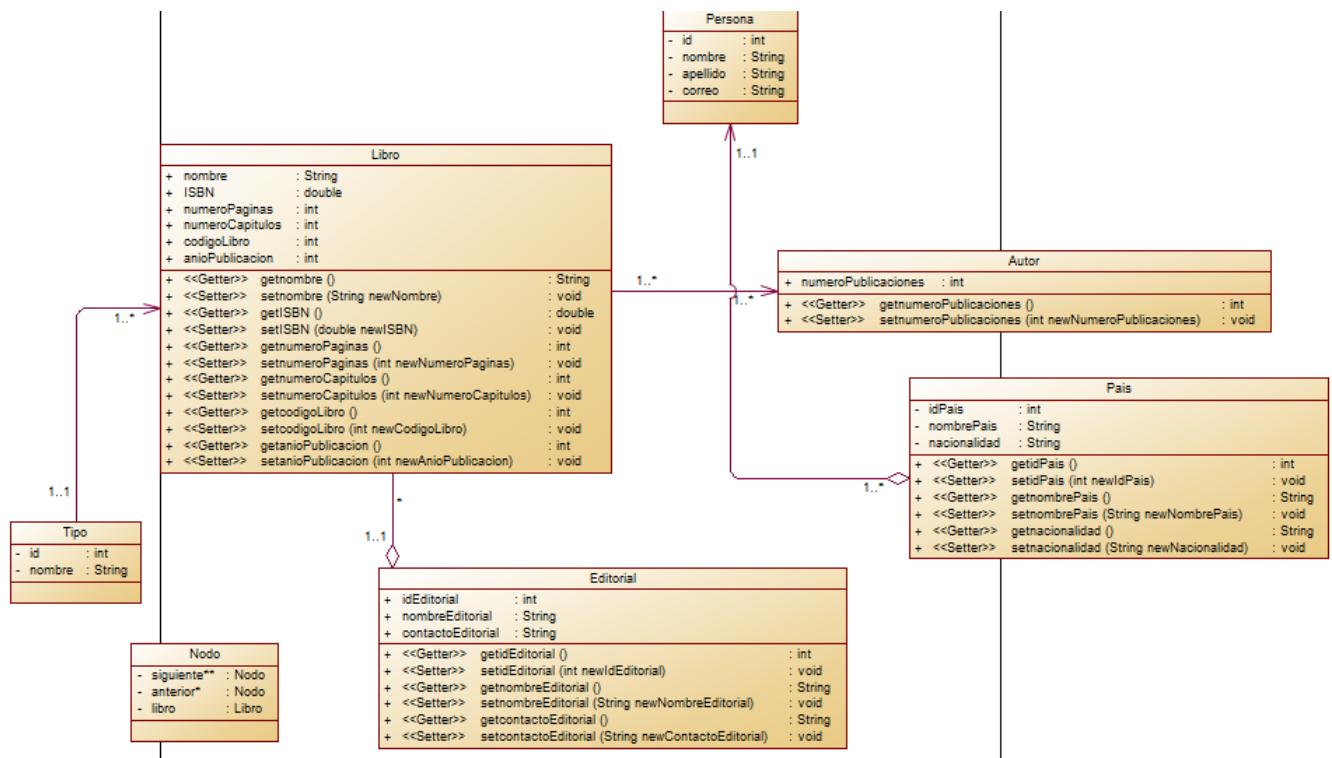
Descripción

El programa consiste en el ingreso respectivo de libros en una biblioteca virtual en la cual se deben ingresar el nombre del libro, el nombre y la identificación del autor, el correo, el país del autor la nacionalidad y los datos respectivos del editorial y estos datos son guardados en el sistema.

Objetivo

Construir una lista doblemente enlazada para poder guardar los datos de una simulación de biblioteca ya sea en el principio, final o entre de la lista y también permite modificar la lista.

Modelo



Código

```

Autor.h
#include "Persona.h"
class Autor
{
public:
    Autor();
    ~Autor();
    Persona getPersona(void);
    void setPersona(Persona newPersona);
    int getNumPublicacion(void);
    void setNumPublicacion(int newNumPublicacion);
protected:
private:
    Persona persona;
    int numPublicacion;
};

Autor.cpp
#include "Autor.h"
Autor::Autor()
{
}
Autor::~Autor()
{
    // TODO : implement
}
  
```

```

}

Persona Autor::getPersona(void)
{
    return persona;
}
void Autor::setPersona(Persona newPersona)
{
    persona = newPersona;
}
int Autor::getNumPublicacion(void)
{
    return numPublicacion;
}
void Autor::setNumPublicacion(int newNumPublicacion)
{
    numPublicacion = newNumPublicacion;
}
Editorial.h
#include<iostream>
#ifndef __BIBLIOTECA_LISTA_DOBLE2_Editorial_h
#define __BIBLIOTECA_LISTA_DOBLE2_Editorial_h
class Editorial
{
public:
    Editorial();
    ~Editorial();
    int getIdEditorial(void);
    void setIdEditorial(int newIdEditorial);
    std::string getNombreEditorial(void);
    void setNombreEditorial(std::string newNombreEditorial);
    std::string getContactoEditorial(void);
    void setContactoEditorial(std::string newContactoEditorial);
protected:
private:
    int idEditorial;
    std::string nombreEditorial;
    std::string contactoEditorial;
};

Editorial.cpp
#include "Editorial.h"
Editorial::Editorial()
{
}
Editorial::~Editorial()
{
    // TODO : implement
}
int Editorial::getIdEditorial(void)
{
    return idEditorial;
}
void Editorial::setIdEditorial(int newIdEditorial)
{
    idEditorial = newIdEditorial;
}
std::string Editorial::getNombreEditorial(void)

```

```

{
    return nombreEditorial;
}
void Editorial::setNombreEditorial(std::string newNombreEditorial)
{
    nombreEditorial = newNombreEditorial;
}
std::string Editorial::getContactoEditorial(void)
{
    return contactoEditorial;
}
void Editorial::setContactoEditorial(std::string newContactoEditorial)
{
    contactoEditorial = newContactoEditorial;
}
Libro.h
#include "Editorial.h"
Editorial::Editorial()
{
}
Editorial::~Editorial()
{
    // TODO : implement
}
int Editorial::getIdEditorial(void)
{
    return idEditorial;
}
void Editorial::setIdEditorial(int newIdEditorial)
{
    idEditorial = newIdEditorial;
}
std::string Editorial::getNombreEditorial(void)
{
    return nombreEditorial;
}
void Editorial::setNombreEditorial(std::string newNombreEditorial)
{
    nombreEditorial = newNombreEditorial;
}
std::string Editorial::getContactoEditorial(void)
{
    return contactoEditorial;
}
void Editorial::setContactoEditorial(std::string newContactoEditorial)
{
    contactoEditorial = newContactoEditorial;
}
Libro.cpp
#include "Libro.h"
Libro::Libro()
{
}
Libro::~Libro()
{
    // TODO : implement
}

```

```

int Libro::getIdLibro(void)
{
    return idLibro;
}
void Libro::setIdLibro(int newIdLibro)
{
    idLibro = newIdLibro;
}
std::string Libro::getNombreLibro(void)
{
    return nombreLibro;
}
void Libro::setNombreLibro(std::string newNombreLibro)
{
    nombreLibro = newNombreLibro;
}
Autor Libro::getAutor(void)
{
    return autor;
}
void Libro::setAutor(Autor newAutor)
{
    autor = newAutor;
}
Pais Libro::getPais(void)
{
    return pais;
}
void Libro::setPais(Pais newPais)
{
    pais = newPais;
}
Editorial Libro::getEditorial(void)
{
    return editorial;
}
void Libro::setEditorial(Editorial newEditorial)
{
    editorial = newEditorial;
}
ListaDobleCircular.h
#include "Nodo.h"
#include "Libro.h"
class ListaDobleCircular
{
public:
    bool vacia(void);
    void insertarInicio(Libro newLibro);
    void insertarFinal(Libro newLibro);
    void insertarPosicion(int newPosition, Libro newLibro);
    int buscarPosicionLibroId(int idLibro);
    void modificarPorId(int idLibro, Libro libro);
    void eliminar(int id);
    void imprimir(void);
    void imprimirNodo(Nodo* nodo);
    int tamanoLista(void);
    void crearTxt(Nodo* nodo);

```

```

protected:
private:
    Nodo* primero;
    Nodo* ultimo;

};

listaDobleCircular.cpp
#include "ListaDobleCircular.h"
#include<stdlib.h>
#include <iostream>
bool ListaDobleCircular::vacia(void)
{
    if(primer==NULL && ultimo==NULL) {
        return true;
    }else{
        return false;
    }
}
void ListaDobleCircular::insertarInicio(Libro newLibro)
{
    Nodo* nuevo=new Nodo();
    nuevo->setLibro(newLibro);
    if(vacia()){
        primero=nuevo;
        ultimo=nuevo;
        primero->setSiguiente(primer);
        primero->setAnterior(ultimo);
    }else{
        primero->setAnterior(nuevo);
        nuevo->setAnterior(ultimo);
        nuevo->setSiguiente(primer);
        primero=nuevo;
        ultimo->setSiguiente(primer);
    }
    crearTxt(nuevo);
}
void ListaDobleCircular::insertarFinal(Libro newLibro)
{
    Nodo* nuevo=new Nodo();
    nuevo->setLibro(newLibro);
    if(vacia()){
        primero=nuevo;
        ultimo=nuevo;
        primero->setSiguiente(primer);
        primero->setAnterior(ultimo);
    }else{
        ultimo->setSiguiente(nuevo);
        nuevo->setAnterior(ultimo);
        nuevo->setSiguiente(primer);
        ultimo=nuevo;
        primero->setAnterior(ultimo);
    }
    crearTxt(nuevo);
}
void ListaDobleCircular::insertarPosicion(int newPosicion, Libro newLibro)
{
    Nodo* aux =primer;

```

```

int cont=0;
if(tamanoLista()>=newPosicion){
    while(cont!=newPosicion){
        aux=aux->getSiguiente();
        cont++;
    }
    Nodo* aux2=aux->getAnterior();
    Nodo* nuevo=new Nodo();
    nuevo->setLibro(newLibro);
    aux2->setSiguiente(nuevo);
    nuevo->setAnterior(aux2);
    nuevo->setSiguiente(aux);
    aux->setAnterior(nuevo);
    crearTxt(nuevo);
} else{
    std::cout<<"LA POSICION ES MAYOR AL TAMAÑO DE LA LISTA\n";
}
}

int ListaDobleCircular::buscarPosicionLibroId(int idLibro)
{
    int contador=0;
    if(!vacia()){
        Nodo* aux=primero;
        do{
            if(idLibro==aux->getLibro().getIdLibro()){
                return contador;
            }
            aux=aux->getSiguiente();
            contador++;
        }while(aux!=primero);
    }
    return -1;
}
void ListaDobleCircular::modificarPorId(int idLibro,Libro libro)
{
    Nodo* aux =primero;
    Nodo* modificar = new Nodo();
    modificar->setLibro(libro);
    int posicion=buscarPosicionLibroId(idLibro);
    int cont=0;
    bool bandera=true;
    if(posicion!=-1 && !vacia()){
        while(cont!=posicion){
            aux=aux->getSiguiente();
            cont++;
        }
    }else{
        bandera=false;
    }
    if(bandera){
        if(cont==0){/** CUANDO EN LA LISTA SOLO EXISTA UN NODO EN LA
LISTA */
            primero=NULL;
            ultimo=NULL;
            insertarInicio(libro);
        }else{
            Nodo* aux2=aux->getAnterior();

```

```

        Nodo* aux3=aux->getSiguiente();
        aux2->setSiguiente(modificar);
        modificar->setAnterior(aux2);
        modificar->setSiguiente(aux3);
        aux3->setAnterior(modificar);
    }
    std::cout<<"LIBRO MODIFICADO\n";
    imprimirNodo(modificar);
    free(aux);
} else{
    std::cout<<"NO EXISTE ID PARA MODIFICAR\n";
}
}

void ListaDobleCircular::eliminar(int idLibro)
{
    Nodo* aux =primero;
    int posicion=buscarPosicionLibroId(idLibro);
    int cont=0;
    bool bandera=true;
    if(posicion!=-1 && !vacia()){
        while(cont!=posicion){
            aux=aux->getSiguiente();
            cont++;
        }
    } else{
        bandera=false;
    }
    if(bandera){
        if(cont==0){/** CUANDO EN LA LISTA SOLO EXISTA UN NODO EN LA
LISTA */
            primero=NULL;
            ultimo=NULL;
        }
        if(cont==1){/** CUANDO EN LA LISTA SOLO EXISTAN DOS NODOS EN LA
LISTA*/
            primero=aux->getAnterior();
            ultimo=aux->getAnterior();
            primero->setSiguiente(ultimo);
            primero->setAnterior(ultimo);
        } else{/** CUANDO EN LA LISTA EXISTAN MAS DE DOS NODO EN LA
LISTA*/
            Nodo* aux2=aux->getAnterior();
            Nodo* aux3=aux->getSiguiente();
            aux2->setSiguiente(aux3);
            aux3->setAnterior(aux2);
        }
    std::cout<<"LIBRO ELIMINADO\n";
    imprimirNodo(aux);
    free(aux);
} else{
    std::cout<<"NO EXISTE ID PARA ELIMINAR\n";
}
}

void ListaDobleCircular::imprimir(void)
{
    Nodo* aux=primero;
    if(!vacia()){

```

```

    do{

        std::cout<<"//////////////////////////////\n";
        //////////////////<<std::endl;
        std::cout<<"    IDENTI LIBRO: "<<aux-
>getLibro().getIdLibro()<<std::endl;
        std::cout<<"    NOMBRE LIBRO: "<<aux-
>getLibro().getNombreLibro()<<std::endl;
        std::cout<<"    IDENTI AUTOR: "<<aux-
>getLibro().getAutor().getPersona().getId()<<std::endl;
        std::cout<<"    NOMBRE AUTOR: "<<aux-
>getLibro().getAutor().getPersona().getNombre();
        std::cout<<"    "<<aux-
>getLibro().getAutor().getPersona().getApellido()<<std::endl;
        std::cout<<"    CORREO AUTOR: "<<aux-
>getLibro().getAutor().getPersona().getCorreo()<<std::endl;
        std::cout<<"    IDENTI. PAIS : "<<aux-
>getLibro().getPais().getIdPais()<<std::endl;
        std::cout<<"    NOMBRE PAIS : "<<aux-
>getLibro().getPais().getNombrePais()<<std::endl;
        std::cout<<"    NACIONALIDAD : "<<aux-
>getLibro().getPais().getNacionalidad()<<std::endl;
        std::cout<<"    NUM. PUBLICA.: "<<aux-
>getLibro().getAutor().getNumPublicacion()<<std::endl;
        std::cout<<"    IDENTI EDITORIAL: "<<aux-
>getLibro().getEditorial().getIdEditorial()<<std::endl;
        std::cout<<"    NOMBRE EDITORIAL: "<<aux-
>getLibro().getEditorial().getNombreEditorial()<<std::endl;
        std::cout<<"    CONTAC EDITORIAL: "<<aux-
>getLibro().getEditorial().getContactoEditorial()<<std::endl;

        std::cout<<"/////////////////////////////\n";
        //////////////////<<std::endl<<std::endl;
        aux=>getSiguiente();
    }while(aux!=primero);
    }else{
        std::cout<<"LISTA DOBLE VACIA\n";
    }
}

void ListaDobleCircular::imprimirNodo(Nodo* nodo)
{
    std::cout<<"/////////////////////////////\n";
    //////////////////<<std::endl;
    std::cout<<"    IDENTI LIBRO: "<<nodo-
>getLibro().getIdLibro()<<std::endl;
    std::cout<<"    NOMBRE LIBRO: "<<nodo-
>getLibro().getNombreLibro()<<std::endl;
    std::cout<<"    IDENTI AUTOR: "<<nodo-
>getLibro().getAutor().getPersona().getId()<<std::endl;
    std::cout<<"    NOMBRE AUTOR: "<<nodo-
>getLibro().getAutor().getPersona().getNombre();
    std::cout<<"    "<<nodo-
>getLibro().getAutor().getPersona().getApellido()<<std::endl;
    std::cout<<"    CORREO AUTOR: "<<nodo-
>getLibro().getAutor().getPersona().getCorreo()<<std::endl;
    std::cout<<"    IDENTI. PAIS : "<<nodo-
>getLibro().getPais().getIdPais()<<std::endl;

```

```

        std::cout<<" NOMBRE PAIS : "<<nodo-
>getLibro().getPais().getNombrePais()<<std::endl;
        std::cout<<" NACIONALIDAD : "<<nodo-
>getLibro().getPais().getNacionalidad()<<std::endl;
        std::cout<<" NUM. PUBLICA.: "<<nodo-
>getLibro().getAutor().getNumPublicacion()<<std::endl;
        std::cout<<" IDENTI EDITORIAL: "<<nodo-
>getLibro().getEditorial().getIdEditorial()<<std::endl;
        std::cout<<" NOMBRE EDITORIAL: "<<nodo-
>getLibro().getEditorial().getNombreEditorial()<<std::endl;
        std::cout<<" CONTAC EDITORIAL: "<<nodo-
>getLibro().getEditorial().getContactoEditorial()<<std::endl;
        std::cout<<"/////////////////////////////"<<std::endl<<std::endl;
}
int ListaDobleCircular::tamanioLista(void)
{
    int cont=0;
    if(!vacia()){
        Nodo* aux=primero;
        do{
            aux=aux->getSiguiente();
            cont++;
        }while(aux!=primero);
    }
    return cont;
}
void ListaDobleCircular::crearTxt(Nodo* nodo)
{
    std::fstream enter;
    enter.open("BIBLIOTECA.txt", std::fstream::app);
    enter<<"/////////////////////////////"<<std::endl;
    enter<<" IDENTI LIBRO: "<<nodo->getLibro().getIdLibro()<<std::endl;
    enter<<" NOMBRE LIBRO: "<<nodo-
>getLibro().getNombreLibro()<<std::endl;
    enter<<" IDENTI AUTOR: "<<nodo-
>getLibro().getAutor().getPersona().getId()<<std::endl;
    enter<<" NOMBRE AUTOR: "<<nodo-
>getLibro().getAutor().getPersona().getNombre() ;
    enter<<" "<<nodo-
>getLibro().getAutor().getPersona().getApellido()<<std::endl;
    enter<<" CORREO AUTOR: "<<nodo-
>getLibro().getAutor().getPersona().getCorreo()<<std::endl;
    enter<<" IDENTI. PAIS : "<<nodo-
>getLibro().getPais().getIdPais()<<std::endl;
    enter<<" NOMBRE PAIS : "<<nodo-
>getLibro().getPais().getNombrePais()<<std::endl;
    enter<<" NACIONALIDAD : "<<nodo-
>getLibro().getPais().getNacionalidad()<<std::endl;
    enter<<" NUM. PUBLICA.: "<<nodo-
>getLibro().getAutor().getNumPublicacion()<<std::endl;
    enter<<" IDENTI EDITORIAL: "<<nodo-
>getLibro().getEditorial().getIdEditorial()<<std::endl;
    enter<<" NOMBRE EDITORIAL: "<<nodo-
>getLibro().getEditorial().getNombreEditorial()<<std::endl;

```

```

        enter<<"  CONTAC EDITORIAL: "<<nodo-
>getLibro().getEditorial().getContactoEditorial()<<std::endl;
        enter<<"///////////////////////////////"<<std::endl;
////////////////"<<std::endl;
}

Nodo.h

#include "Libro.h"
class Nodo
{
public:
    Nodo();
    ~Nodo();
    Libro getLibro(void);
    void setLibro(Libro newLibro);
    Nodo* getAnterior(void);
    void setAnterior(Nodo* newAnterior);
    Nodo* getSiguiente(void);
    void setSiguiente(Nodo* newSiguiente);
protected:
private:
    Libro libro;
    Nodo* anterior;
    Nodo* siguiente;
};

Nodo.cpp
#include "Nodo.h"
Nodo::Nodo()
{
}
Nodo::~Nodo()
{
    // TODO : implement
}
Libro Nodo::getLibro(void)
{
    return libro;
}
void Nodo::setLibro(Libro newLibro)
{
    libro = newLibro;
}
Nodo* Nodo::getAnterior(void)
{
    return anterior;
}
void Nodo::setAnterior(Nodo* newAnterior)
{
    anterior = newAnterior;
}
Nodo* Nodo::getSiguiente(void)
{
    return siguiente;
}

```

```

void Nodo::setSiguiente(Nodo* newSiguiente)
{
    siguiente = newSiguiente;
}
Pais.h
#include<iostream>
class Pais
{
public:
    Pais();
    ~Pais();
    int getIdPais(void);
    void setIdPais(int newIdPais);
    std::string getNombrePais(void);
    void setNombrePais(std::string newNombrePais);
    std::string getNacionalidad(void);
    void setNacionalidad(std::string newNacionalidad);
protected:
private:
    int idPais;
    std::string nombrePais;
    std::string nacionalidad;
};

Pais.cpp
#include "Pais.h"
Pais::Pais()
{
}
Pais::~Pais()
{
    // TODO : implement
}
int Pais::getIdPais(void)
{
    return idPais;
}
void Pais::setIdPais(int newIdPais)
{
    idPais = newIdPais;
}
std::string Pais::getNombrePais(void)
{
    return nombrePais;
}
void Pais::setNombrePais(std::string newNombrePais)
{
    nombrePais = newNombrePais;
}
std::string Pais::getNacionalidad(void)
{
    return nacionalidad;
}
void Pais::setNacionalidad(std::string newNacionalidad)
{
    nacionalidad = newNacionalidad;
}

```

```

Persona.h
#include<iostream>
class Persona
{
public:
    Persona();
    ~Persona();
    std::string getId(void);
    void setId(std::string newId);
    std::string getNombre(void);
    void setNombre(std::string newNombre);
    std::string getApellido(void);
    void setApellido(std::string newApellido);
    std::string getCorreo(void);
    void setCorreo(std::string newCorreo);
protected:
private:
    std::string id;
    std::string nombre;
    std::string apellido;
    std::string correo;
};

Persona.cpp
#include "Persona.h"
Persona::Persona()
{
}
Persona::~Persona()
{
    // TODO : implement
}
std::string Persona::getId(void)
{
    return id;
}
void Persona::setId(std::string newId)
{
    id = newId;
}
std::string Persona::getNombre(void)
{
    return nombre;
}
void Persona::setNombre(std::string newNombre)
{
    nombre = newNombre;
}
std::string Persona::getApellido(void)
{
    return apellido;
}
void Persona::setApellido(std::string newApellido)
{
    apellido = newApellido;
}
std::string Persona::getCorreo(void)

```

```

{
    return correo;
}
void Persona::setCorreo(std::string newCorreo)
{
    correo = newCorreo;
}
Ingreso.h
#include<stdio.h>
class Ingreso {
public:
    char * ingresoCadena(char *mensage);
    char* ingresoNumero(char *mensage);
    char * ingresoCorreo(char *mensage);
};

char * Ingreso::ingresoCadena(char * mensage) {
printf("%s ",mensage);
bool flag=true;
char caracter[]="";
char *lectura=(char*)malloc(sizeof(char));
int i=0;
while(flag){
    caracter[0]=getch();
    if((caracter[0]>=65 && caracter[0]<=90) || (caracter[0]>=97 &&
caracter[0]<=122) || caracter[0]==32)
    {
        std::cout<<caracter[0];
        *(lectura+i)=caracter[0];
        i++;
    }
    else
    {
        if(caracter[0]==13)
        {
            flag=false;
        }
        else
        {
            if(caracter[0]==8 && i>0)
            {
                *(lectura+i-1)='\0';
                std::cout<< "\b \b";
                i--;
            }
        }
    }
}
*(lectura+i)='\0';
return lectura;
}
char * Ingreso::ingresoNumero(char * mensage) {
printf("%s ",mensage);
bool flag=true;
char caracter[]="";
char *lectura=(char*)malloc(sizeof(char));
int i=0;
while(flag){

```

```

caracter[0]= getch();
if((caracter[0]>=48 && caracter[0]<=57))
{
    std::cout<<caracter[0];
    *(lectura+i)=caracter[0];
    i++;
}
else
{
    if(caracter[0]==13)
    {
        flag=false;
    }
    else
    {
        if(caracter[0]==8 && i>0)
        {
            *(lectura+i-1)='\0';
            std::cout<< "\b \b";
            i--;
        }
    }
}
*(lectura+i)='\0';
return lectura;
}
char * Ingreso::ingresoCorreo(char * mensaje) {
printf("%s ",mensaje);
bool flag=true;
char caracter[]="";
char *lectura=(char*)malloc(sizeof(char));
bool validar=true;
int i=0;
while(flag){
    caracter[0]= getch();
    if((caracter[0]>=48 && caracter[0]<=57) || (caracter[0]>=97 &&
caracter[0]<=122) || caracter[0]==64 || caracter[0]==46 || caracter[0]==95)
    {
        std::cout<<caracter[0];
        *(lectura+i)=caracter[0];
        i++;
        if(caracter[0]==64){
            validar=false;
        }
    }
    else
    {
        if(caracter[0]==13)
        {
            flag=false;
        }
        else
        {
            if(caracter[0]==8 && i>0)
            {
                *(lectura+i-1)='\0';
            }
        }
    }
}
*(lectura+i)='\0';
return lectura;
}

```

```

        std::cout<< "\b \b";
        i--;
    }
}
}
}
*(lectura+i)='\0';
if(validar){
    printf("\nCORREO NO VALIDO VUELVA A INGRESAR\n");
    free(lectura);
    lectura=ingresoCorreo(mensaje);
}
return lectura;
}

Principal
#include <iostream>
#include <windows.h>
#include <conio.h>
#include <stdlib.h>
#include <fstream>
#include "Ingreso.h"
#include "ListaDobleCircular.h"
#include "Libro.h"
#include "Autor.h"
#include "Persona.h"
#include "Pais.h"
#include "Editorial.h"
#define ARRIBA    72
#define ABAJO     80
#define ENTER     13

void gotoxy(short posicionx, short posiciony);
void menu();
void biblioteca(int opcion,ListaDobleCircular *lista,int &idLibro);
Libro ingresarLibro(int &id);
Autor ingresarAutor();
Persona ingresarPersona();
Pais ingresarPais();
Editorial ingresarEditorial();
using namespace std;
int main()
{
    fstream enter;
    ListaDobleCircular *lista=new ListaDobleCircular();
    int posX = 4;
    int posY = 6;
    int posMaxY=12;
    bool bandera = true;
    int idLibro=1;
    enter.open("BIBLIOTECA.txt", fstream::out);
    enter << "Lista Dobles" << endl;
    enter << "Lista de libros" << endl;

    do{
        system("cls");
        std::cout << "\n\t BIBLIOTECA \n\tLISTAS DOBLES";

```

```

menu();
gotoxy(posX, posY);
cout << "=>";
switch (_getch())
{
case ARRIBA:
    posY--;
    if (posY < 6) {
        posY = posMaxY;
    }
    break;
case ABAJO:
    posY++;
    if (posY > posMaxY) {
        posY = 6;
    }
    break;
case ENTER:
    if (posY == posMaxY) {
        bandera = false;
    }
    else {
        biblioteca(posY, lista, idLibro);
    }
    break;
}
} while (bandera);
enter.close();

return 0;
}
void gotoxy(short posicionx, short posiciony) {
COORD coordenadaPosicion = { posicionx, posiciony };
SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),
coordenadaPosicion);
}

void menu() {
int posX = 7;
int posY = 6;
gotoxy(posX, posY++); cout << "INGRESAR INICIO";
gotoxy(posX, posY++); cout << "INGRESAR FINAL";
gotoxy(posX, posY++); cout << "INGRESAR POSICION";
gotoxy(posX, posY++); cout << "MODIFICAR POR ID";
gotoxy(posX, posY++); cout << "ELIMINAR POR ID";
gotoxy(posX, posY++); cout << "IMPRIMIR";
gotoxy(posX, posY++); cout << "SALIR";
}
void biblioteca(int opcion, ListaDobleCircular* lista, int &idLibro) {
    Ingreso leer;
    bool bandera;
system("cls");
int idModificar=0;
int aux;
switch (opcion)
{
case 6:

```

```

//INGRESO AL INICIO
cout << "\n\tINGRESO AL INICIO\n\n";
lista->insertarInicio(ingresarLibro(idLibro));
break;
case 7:
//INGRESO AL FINAL
cout << "\n\tINGRESO AL FINAL\n\n";
lista->insertarFinal(ingresarLibro(idLibro));
break;
case 8:
//INGRESO AL POSICION
cout << "\n\tINGRESO POR POSICION\n\n";
lista->insertarPosicion(1,ingresarLibro(idLibro));
break;
case 9:
//MODIFICAR
cout << "\n\tMODIFICAR POR ID\n\n";
do{
    bandera=false;
    idModificar=atoi(leer.ingresoNumero(" Ingrese el ID:"));
    if(lista->buscarPosicionLibroId(idModificar)==-1){
        cout<<" ID NO ENCONTRADO\n";
        cout<<" ENTER para ingresar o ESC para salir\n";
        switch(getch()){
            case 13:
                bandera=true;
                break;
        }
    }
}while(bandera);
aux=idModificar;
lista->modificarPorId(aux,ingresarLibro(idModificar));
break;
case 10:
//ELIMINAR
cout << "\n\tELIMINAR POR ID\n\n";
do{
    bandera=false;
    idModificar=atoi(leer.ingresoNumero(" Ingrese el ID:"));
    if(lista->buscarPosicionLibroId(idModificar)==-1){
        cout<<"\n ID NO ENCONTRADO\n";
        cout<<" ENTER para ingresar o ESC para salir\n";
        switch(getch()){
            case 13:
                bandera=true;
                break;
        }
    }
}while(bandera);
lista->eliminar(idModificar);
break;
case 11:
//IMPRIMIR
cout << "\n\tIMPRESION DE DATOS DE LA LISTA\n\n";
lista->imprimir();
break;
default:

```

```

        break;
    }
    system("pause");
}
Libro ingresarLibro(int &id){
    Ingreso leer;
    Libro libro;
    cout<<"\nINGRESE LOS DATOS\n";
    cout<<" ID del Libro: "<<id<<endl;
    libro.setIdLibro(id);
    libro.setNombreLibro(leer.ingresoCadena(" Nombre del
Libro:"));cout<<endl;
    Autor autor=ingresarAutor();
    Editorial editorial=ingresarEditorial();
    Pais pais=ingresarPais();
    libro.setAutor(autor);
    libro.setEditorial(editorial);
    libro.setPais(pais);
    id++;
    return libro;
}
Autor ingresarAutor(){
    Autor autor;
    Ingreso leer;
    cout<<"\nDATOS AUTOR\n";
    Persona persona=ingresarPersona();
    autor.setPersona(persona);
    autor.setNumPublicacion(atoi(leer.ingresoNumero(" Num
Publicacion:")));cout<<endl;
    return autor;
}
Persona ingresarPersona(){
    Persona persona;
    Ingreso leer;
    persona.setId(leer.ingresoNumero(" Cedula:"));cout<<endl;
    persona.setNombre(leer.ingresoCadena(" Nombre:"));cout<<endl;
    persona.setApellido(leer.ingresoCadena(" Apellido:"));cout<<endl;
    persona.setCorreo(leer.ingresoCorreo(" Correo:"));cout<<endl;
    return persona;
}
Pais ingresarPais(){
    Pais pais;
    Ingreso leer;
    cout<<"\nDATOS DEL PAIS\n";
    pais.setIdPais(atoi(leer.ingresoNumero(" Id del Pais:")));cout<<endl;
    pais.setNombrePais(leer.ingresoCadena(" Nombre del
Pais:"));cout<<endl;
    pais.setNacionalidad(leer.ingresoCadena(" Nacionalidad:
"));cout<<endl;
    return pais;
}
Editorial ingresarEditorial(){
    Editorial editorial;
    Ingreso leer;
    cout<<"\nDATOS DE LA EDITORIAL\n";
    editorial.setIdEditorial(atoi(leer.ingresoNumero(" ID
Editorial")));cout<<endl;

```

```

        editorial.setNombreEditorial(leer.ingresoCadena("
Nombre:"));cout<<endl;
        editorial.setContactoEditorial(leer.ingresoNumero("
Telefono:"));cout<<endl;
    return editorial;
}

```

Ejecución

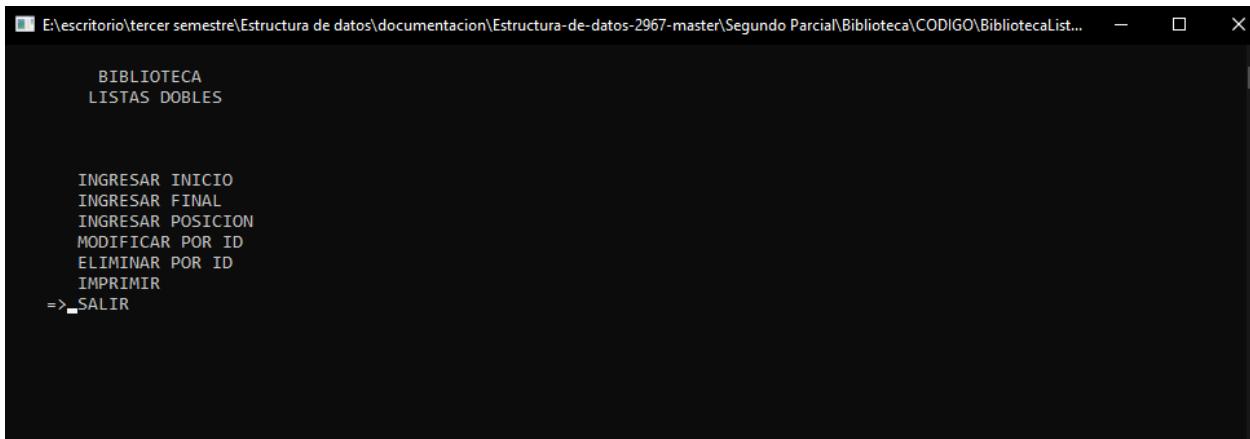


Figura 1. menú de inicio

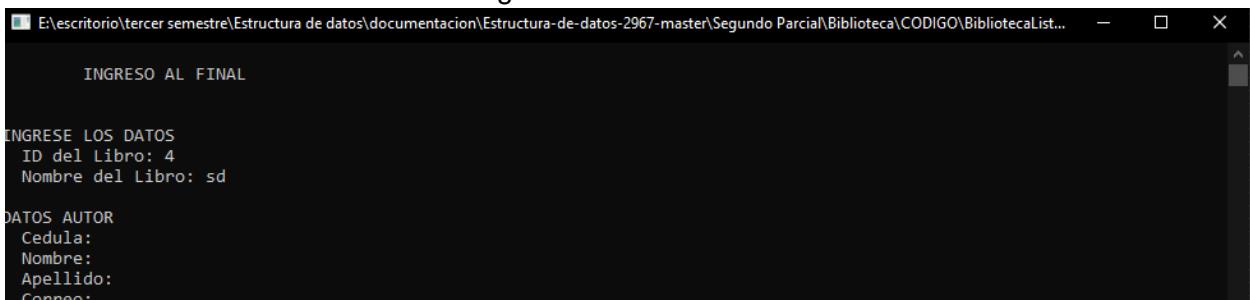


Figura 2. Ingreso de datos

```

E:\escritorio\tercer semestre\Estructura de datos\documentacion\Estructura-de-datos-2967-master\Segundo Parcial\Biblioteca\CODIGO\BibliotecaList...
INGRESO AL INICIO

INGRESE LOS DATOS
ID del Libro: 3
Nombre del Libro: Wario

DATOS AUTOR
Cedula: 1725454463
Nombre: Mario
Apellido: Majak
Correo: hut@hotmail.com
Num Publicacion: 3

DATOS DE LA EDITORIAL
ID Editorial 21
Nombre: Ojsayo
Telefono: 23995674

DATOS DEL PAIS
Id del Pais: 3
Nombre del Pais: Ecuador
Nacionalidad: ecuatoriana

```

Figura 3. Impresión de datos

Sección de proyectos segundo parcial

Descripción general

Los aplicativos tienen como objetivos desplazar números o figuras de forma vertical con semejanza al juego tetris, estos a su vez contienen información numérica ya que cada vez que estos colisionan se eliminaran y la información que guardan se imprimen como una lista doblemente enlazada circular.

Grupo Picado-Naula

Modelado



Codigo

```
Clase Main

#include "PersonalLibrary.h"
#include "ListaDoble.h"

void encriptarArchivo() {
    FILE *encriptame;
    FILE *encriptado;

    system("cls");
    printf("Importante:\n");
    printf(" * El archivo a ser encriptado debe llamarse
'encriptame.txt'.\n");
    printf(" * El archivo 'encriptame.txt' debe estar en la raiz del
ejecutable.\n");
    printf("Pulse Enter para proceder con el proceso de encriptacion.\n");

    getchar();

    encriptame = fopen("encriptame.txt", "r");
    encriptado = fopen("encriptado.txc", "w");

    while (!feof(encriptame)) {
        fputc(fgetc(encriptame) + 3, encriptado);
    }

    fclose(encriptame);
    fclose(encriptado);

    printf("Listo. Pulsa Enter para salir.\n");

    getchar();
    exit(0);
}

void desencriptarArchivo() {
    FILE *encriptado;
    FILE *desencriptado;

    system("cls");
    printf("Importante:\n");
    printf(" * El archivo a ser desencriptado debe llamarse
'encriptado.txc'.\n");
    printf(" * El archivo 'encriptado.txc' debe estar en la raiz del
ejecutable.\n");
    printf("Pulse Enter para proceder con el proceso de desencriptacion.\n");

    getchar();

    encriptado = fopen("encriptado.txc", "r");
    desencriptado = fopen("desencriptado.txt", "w");

    while (!feof(encriptado)) {
```



```

        ShellExecute(NULL, TEXT("open"),
TEXT("Extras\\WinAppMSAgentsManagementQR.exe"), NULL, NULL, SW_SHOWNORMAL);
        system("start Extras\\CreateCodigoQR.jar");
        break;
    case 2:
        flagGame = false;
        break;
    }
    if (opcionJuego != 4) {
        printf("Presione cualquier tecla para volver al menu
. . .");
        getch();
    }
} while (flagGame);
break;
case 2:
/*
    system("cls");
    system("java -jar pixel.jar");
    system("macara.jpg");
    system("image_pixelated.jpg");
    cout<<"Imagen Pixeleadada\n";
    system("Pause");
    system("cls");*/
}

ShellExecute(NULL, TEXT("open"),
TEXT("Extras\\WinAppMSAgentsManagementImage.exe"), NULL, NULL,
SW_SHOWNORMAL);
    ShellExecute(NULL, TEXT("open"), TEXT("Extras\\image.exe"),
NULL, NULL, SW_SHOWNORMAL);
    break;
case 3:
    ShellExecute(NULL, TEXT("open"),
TEXT("Extras\\WinAppMSAgentsManagementPDF.exe"), NULL, NULL, SW_SHOWNORMAL);
    system("start Extras\\CreatePDF.jar");
    break;
case 4:
    ShellExecute(NULL, TEXT("open"),
TEXT("Extras\\WinAppMSAgentsManagementQR.exe"), NULL, NULL, SW_SHOWNORMAL);
    system("start Extras\\CreateCodigoQR.jar");
    break;
case 5:
    ShellExecute(NULL, TEXT("open"),
NULL, NULL, SW_SHOWNORMAL);
    break;
case 6:
    encrip();
    break;
case 7:
    ShellExecute(NULL, TEXT("open"),
TEXT("Extras\\WinAppMSAgentsManagementExit.exe"), NULL, NULL, SW_SHOWNORMAL);
    color(13);
    printf("\t\t%c Gracias por Jugar %c%c\n\n", 245, 245, 245,
245);
    color(15);
    flag = false;
    break;

```

```

        }
    } while (flag);
    delete(ObjJuego);
}

Clase ListaDoble

#include "Nodo.h"

class ListaDoble {
private:
    Nodo *lista;
    char jugador[50];
    int contadorNodo;
    int puntaje;

public:

    ListaDoble() {
        lista = NULL;
        contadorNodo = 0;
        puntaje = 0;
    }
    void insertarAlFinal(int);
    void impresionArchivos();
    void impresion();
    void impresionLista();
    void generarLista();
    void listaTetris(int, int);
    void juegoTetris(int);
    int posicionAIIndice(int x);
    void deleteNumber(int, int);
    void siguienteNumero(int, Nodo *);
    void deleteNodo(Nodo *Actual);
    void guardar();

};

void ListaDoble::impresionArchivos() {
    Nodo* Aux = new Nodo();
    FILE *archivoPDF, *archivoQR;
    archivoQR = fopen("C:\\\\JuegoSnake\\\\PuntajeJugador.txt", "w");
    archivoPDF = fopen("C:\\\\JuegoSnake\\\\Puntajes.txt", "a+");

    Aux = lista;
    printf("Jugador: %s.\n\n", jugador);
    fprintf(archivoPDF, "Jugador: %s\n\n", jugador);
    fprintf(archivoQR, "Jugador: %s.\n\n", jugador);
    if (Aux == NULL) {
        printf("No Acerto Ningun Numero.\n\n");
        fprintf(archivoPDF, "No Acerto Ningun Numero.\n\n");
        fprintf(archivoQR, "No Acerto Ningun Numero.\n\n");
    } else {
        printf("\t\tLista de Numeros Acertados.\n\n ==> ");
        fprintf(archivoPDF, "\t\tNumeros Acertados.\n\n ==> ");
        fprintf(archivoQR, "\t\tNumeros Acertados.\n\n ==> ");
        while (Aux != NULL) {

```

```

        printf("%d, ", Aux->getNumero());
        fprintf(archivoPDF, "%d, ", Aux->getNumero());
        fprintf(archivoQR, "%d, ", Aux->getNumero());
        Aux = Aux->getSiguienteDireccion();
    }
    printf("\b\b. \n\nPUNTAJE: %d\n\n", puntaje);
    fprintf(archivoPDF, "\b\b.b. \n\nPUNTAJE: %d\n\n-----");
-----\n", puntaje);
    fprintf(archivoQR, "\b\b.b. \n\nPUNTAJE: %d", puntaje);
}
fclose(archivoPDF);
fclose(archivoQR);
}

void ListaDoble::impresion() {
Nodo* Aux = new Nodo();

Aux = lista;
printf("Jugador: %s.\n\n", jugador);

if (Aux == NULL)
    printf("No Acerto Ningun Numero.\n\n");
else {
    printf("\t\tLista de Numeros Acertados\n\n ==> ");
    while (Aux != NULL) {
        printf("%d, ", Aux->getNumero());
        Aux = Aux->getSiguienteDireccion();
    }
    printf("\b\b. \n\nPUNTAJE: %d\n\n", puntaje);
}
}

void ListaDoble::impresionLista() {
int x = 8, y;
Nodo* Aux = new Nodo();
Aux = lista;

while (Aux != NULL) {
    y = 34;
    switch (Aux->getNumero()) {
        case 0:
            color(1);
            gotoxy(x, y++);
            printf(" %c%c%c \n", 219, 219, 219);
            gotoxy(x, y++);
            printf("%c %c\n", 219, 219);
            gotoxy(x, y++);
            printf("%c %c\n", 219, 219);
            gotoxy(x, y++);
            printf("%c %c\n", 219, 219);
            gotoxy(x, y++);
            printf(" %c%c%c ", 219, 219, 219);
            break;

        case 1:
            color(2);
            gotoxy(x, y++);
    }
}
}

```

```

        printf("%c%c\n", 219, 219);
        gotoxy(x, y++);
        printf("%c %c\n", 219, 219);
        gotoxy(x, y++);
        printf(" %c\n", 219);
        gotoxy(x, y++);
        printf(" %c\n", 219);
        gotoxy(x, y++);
        printf("%c%c%c%c%c", 220, 220, 219, 220, 220, 220, 220);
        gotoxy(x, y++);
        break;

case 2:
    color(3);
    gotoxy(x, y++);
    printf(" %c%c%c\n", 219, 219, 219);
    gotoxy(x, y++);
    printf("%c %c\n", 219, 219);
    gotoxy(x, y++);
    printf(" %c\n", 219);
    gotoxy(x, y++);
    printf(" %c\n", 219);
    gotoxy(x, y++);
    printf("%c%c\n", 220, 219);
    gotoxy(x, y++);
    printf("%c%c%c%c%c", 219, 220, 220, 220, 220, 220);
    gotoxy(x, y++);
    break;

case 3:
    color(4);
    gotoxy(x, y++);
    printf(" %c%c%c\n", 219, 219, 219);
    gotoxy(x, y++);
    printf("%c %c\n", 219, 219);
    gotoxy(x, y++);
    printf(" %c\n", 219);
    gotoxy(x, y++);
    printf("%c %c\n", 219, 219);
    gotoxy(x, y++);
    printf(" %c%c%c ", 219, 219, 219);
    gotoxy(x, y++);
    break;

case 4:
    color(6);
    gotoxy(x, y++);
    printf(" %c%c\n", 219, 219);
    gotoxy(x, y++);
    printf(" %c %c\n", 219, 219);
    gotoxy(x, y++);
    printf(" %c %c\n", 219, 219);
    gotoxy(x, y++);
    printf("%c%c%c%c%c\n", 219, 220, 220, 220, 219);
    gotoxy(x, y++);
    printf(" %c\n", 219);
    gotoxy(x, y++);

```

```

break;

case 5:
    color(7);
    gotoxy(x, y++);
    printf("%c%c%c \n", 219, 219, 219);
    gotoxy(x, y++);
    printf("%c \n", 219);
    gotoxy(x, y++);
    printf("%c%c%c \n", 219, 219, 220);
    gotoxy(x, y++);
    printf("%c \n", 219);
    gotoxy(x, y++);
    printf("%c%c " , 219, 219);
    gotoxy(x, y++);
    break;

case 6:
    color(12);
    gotoxy(x, y++);
    printf("%c%c \n", 220, 219);
    gotoxy(x, y++);
    printf("%c \n", 219);
    gotoxy(x, y++);
    printf("%c%c%c%c\n", 219, 220, 220, 220);
    gotoxy(x, y++);
    printf("%c %c\n", 219, 219);
    gotoxy(x, y++);
    printf("%c%c%c ", 219, 219, 219);
    gotoxy(x, y++);
    break;

case 7:
    color(9);
    gotoxy(x, y++);
    printf("%c%c%c%c%c\n", 220, 220, 220, 220, 220);
    gotoxy(x, y++);
    printf("%c\n", 219);
    gotoxy(x, y++);
    printf("%c\n", 219);
    gotoxy(x, y++);
    printf("%c\n", 219);
    gotoxy(x, y++);
    printf("%c ", 219);
    gotoxy(x, y++);
    break;

case 8:
    color(10);
    gotoxy(x, y++);
    printf("%c%c%c \n", 219, 219, 219);
    gotoxy(x, y++);
    printf("%c %c\n", 219, 219);
    gotoxy(x, y++);
    printf("%c%c%c \n", 219, 219, 219);
    gotoxy(x, y++);
    printf("%c %c\n", 219, 219);
    break;

```

```

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

    case 9:
        color(11);
        gotoxy(x, y++);
        printf(" %c%c%c \n", 219, 219, 219);
        gotoxy(x, y++);
        printf("%c %c\n", 219, 219);
        gotoxy(x, y++);
        printf("%c%c%c%c\n", 219, 220, 220, 220, 219);
        gotoxy(x, y++);
        printf(" %c\n", 219);
        gotoxy(x, y++);
        printf(" %c%c%c ", 220, 220, 219);
        gotoxy(x, y++);
        break;
    }
    x += 6;
    Aux = Aux->getSiguienteDireccion();
}
}

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++;
}

//ANALIZAR
void ListaDoble::generarLista() {
    Nodo* Aux = new Nodo();
    int numero, i;
    int selec =1;
    srand(time(NULL));
    for (i = 0; i < 4 + selec; i++) {
        numero = rand() % 10;
        insertarAlFinal(numero);
    }
}

```

```

}

int aleatorio(int caso) {
    if (caso == 1)
        return rand() % 10;
    else
        if (caso == 2)
            return rand() % (39 - 4 + 1) + 4;
    else
        return rand() % (99 - 2 + 1) + 2;
}

void margen() {
    color(121);
    for (int i = 2; i < 100; i++) {
        //PARTE SUPERIOR
        gotoxy(i, 3);
        printf("%c", 178);
        //PARTE INFERIOR
        gotoxy(i, 40);
        printf("%c", 178);
    }
    for (int j = 4; j < 40; j++) {
        //PARTE IZQUIERDA
        gotoxy(2, j);
        printf("%c", 178);
        //PARTE DERECHA
        gotoxy(100, j);
        printf("%c", 178);
    }
    //ESQUINAS
    gotoxy(2, 3);
    printf("%c", 178);
    gotoxy(2, 40);
    printf("%c", 178);
    gotoxy(100, 3);
    printf("%c", 178);
    gotoxy(100, 40);
    printf("%c", 178);
    color(15);
}

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 - 8) / 6;
}

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);
    }
}

void ListaDoble::siguienteNumero(int numero, Nodo *aux) {
    if (aux == NULL || (aux->getNumero() != (numero + 1))) {
        return;
    } else {
        if (aux->getNumero() == (numero + 1)) {
            punaje++;
            deleteNodo(aux);
        }
    }
}

void ListaDoble::deleteNumber(int indice, int numero) {
    int cont = 0;
    Nodo* Actual = new Nodo();

```



```

        x = 34;
        y = 4;
    }
    impresionLista();
    gotoxy(x, y);
    numbers(numero, x, y);
    color(15);
    tecla = getch();
    if (contadorNodo == 13)
        tecla = 's';
    switch (tecla) {
        case TECLA_ABAJO:
            y++;
            if (y >= 28) {
                deleteNumber(posicionAIndice(x), numero);
                y = 4;
                primera = true;
            }
            break;
        case TECLA_DERECHA:
            x += 6;
            if (x >= 92)
                x = 8;
            break;
        case TECLA_IZQUIERDA:
            x -= 6;
            if (x <= 2)
                x = 92;
            break;
        case 's': case 'S':
            gotoxy(3, 43);
            printf("Presiona una tecla para continuar.");
            getch();
            tecla = TECLA_ENTER;
            break;
    }
    if (contadorNodo == 13)
        tecla = TECLA_ENTER;
} while (tecla != TECLA_ENTER);
system("cls");
impresionArchivos();
}

```

Ejecucion



Figura 1: Inicio de juego desplazando números verticalmente.



Figura 2: Colisión de números semejantes.



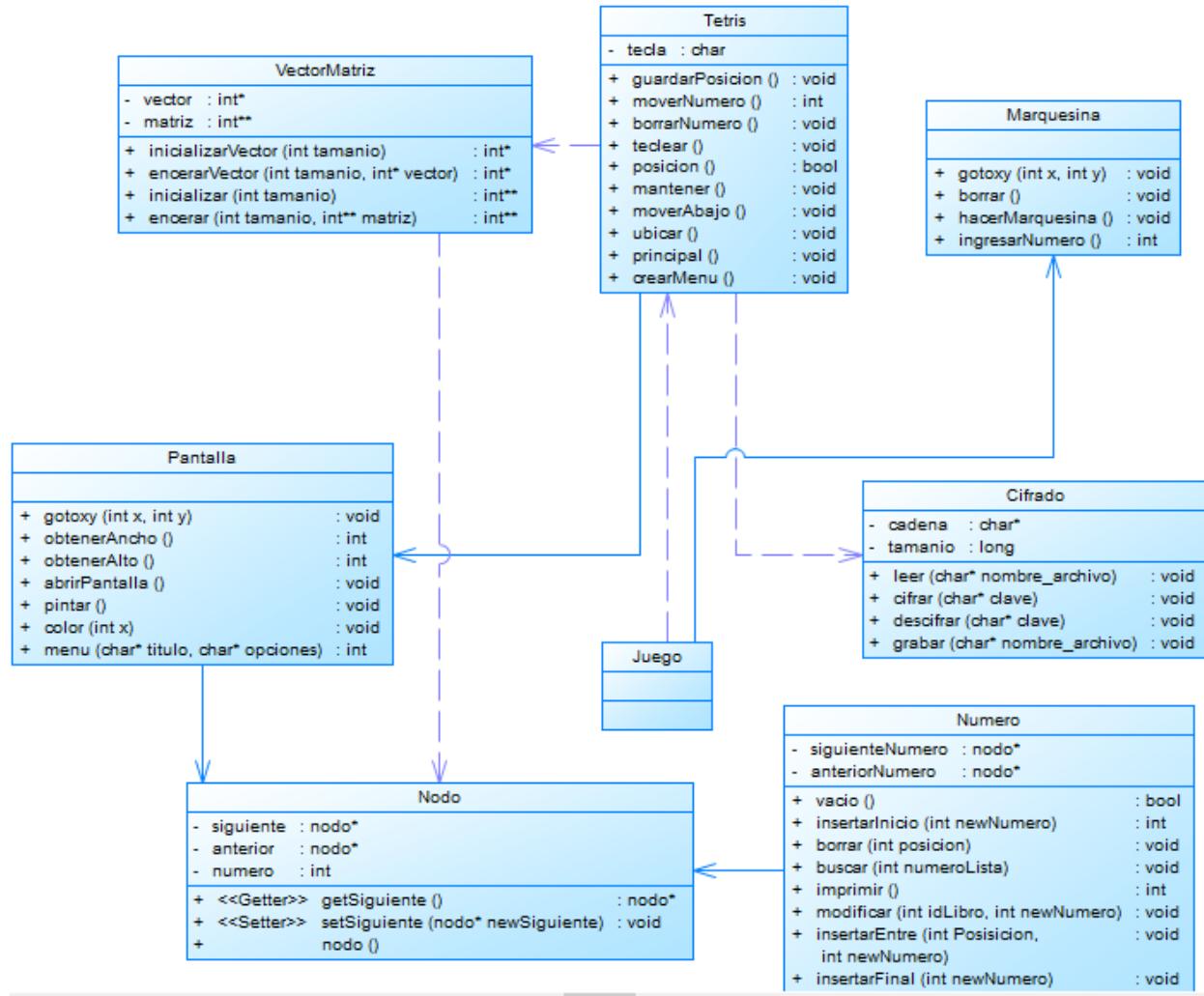
Figura 3: Una vez que colisionaron los números estos se eliminan de la lista inferior.

```
Jugador: Japs.  
Lista de Numeros Acertados.  
==> 6, 0, 6, 9, 7.  
PUNTAJE: 2  
Presione cualquier tecla para volver al menu . . .
```

Figura 4: Presentación de los resultados una vez terminada la ejecución.

Grupo Duy y Puco

Modelado



Código

- `Juego.cpp (Main)`

```

#include "Tetris.cpp"
#include <conio.h>

int main() {
    pthread_t pthread1, pthread2, pthread3;
    archivo.open("solucion.txt",ios::out);
    if (pthread_create(&pthread1, NULL, funcion1, NULL) ) {
        cout << ("Error creando el hilo.");
        abort();
    }
}

```

```

    }
    if (pthread_create(&pthread2, NULL, funcion2, NULL) ) {
        printf("Error creando el hilo.");
        abort();
    }
    if (pthread_create(&pthread3, NULL, funcion3, NULL) ) {
        printf("Error creando el hilo.");
        abort();
    }
    if (pthread_join(pthread1, NULL)) {
        cout << ("Error creando el hilo.");
        abort();
    }
    if (pthread_join(pthread2, NULL)) {
        cout << ("Error creando el hilo.");
        abort();
    }
    if (pthread_join(pthread3, NULL)) {
        cout << ("Error creando el hilo.");
        abort();
    }
    return 0;
}

```

- Cifrado.h

```

class Cifrado {
private:
    char *cadena;
    long tamanio;
public:
    ~Cifrado();
    void leer( const char* );
    void cifrar( const char* );
    void descifrar( const char* );
    void grabar( const char* );
};

```

- Cifrado.cpp

```

#include "Cifrado.h"
#include<iostream>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
using namespace std;

void Cifrado::leer(const char* nombre_archivo) {
    FILE *arl;
    arl = fopen( nombre_archivo,"rb" );
    if( !arl )exit(1);
    fseek( arl,0,2 );
    tamanio = ftell( arl );
    fseek( arl,0,0 );
    cadena = new char[tamanio];
    if( !fread(cadena,tamanio,1,arl) )exit(1);
    fclose( arl );
}

```

```

}

void Cifrado::cifrar(const char *clave) {
    int x=0,y=0;
    while( cadena[x]!=0 ) {
        cadena[x]+=clave[y];
        x++;
        if( clave[y+1]==0 )
            y=0;
        else
            y++;
    }
}

void Cifrado::descifrar(const char *clave) {
    int x=0,y=0;
    while( cadena[x]!=0 ) {
        cadena[x]-=clave[y];
        x++;
        if( clave[y+1]==0 )
            y=0;
        else
            y++;
    }
}

void Cifrado::grabar( const char *nombre_archivo ) {
    FILE *arl;
    arl = fopen( nombre_archivo,"rb+" );
    if( !arl )exit(1);
    if( !fwrite(cadena,tamanio,1,arl) )exit(1);
    fclose( arl );
}

Cifrado::~Cifrado() {
    delete[] cadena;
}

•     ingreso.h

#include <iostream>
#include <stdlib.h>
#include <string.h>

using namespace std;

class Ingreso {
public:
    int ingresarEntero();
    double ingresarDouble();
    float ingresarFloat();
    string ingresarLetra();
    bool validar(string);
    bool validarEntero(string);
    bool validarLetra(string);
};


```

```

int Ingreso::ingresarEntero() {
    string numero;
    bool valido = false;
    while(!valido) {
        try {
            cin >> numero;
            valido = validarEntero(numero);
            if(!valido) {
                throw numero;
            }
        }catch(string e) {
            cout << "El numero " << e << " no es valido" << endl;
        }
    }
    return atoi(numero.c_str());
}

double Ingreso::ingresarDouble() {
    string numero;
    bool valido = false;
    while(!valido) {
        try {
            cin >> numero;
            valido = validar(numero);
            if(!valido) {
                throw numero;
            }
        }catch(string e) {
            cout << "El numero " << e << " no es valido" << endl;
        }
    }
    return atof(numero.c_str());
}

float Ingreso::ingresarFloat() {
    string numero;
    bool valido = false;
    while(!valido) {
        try {
            cin >> numero;
            valido = validar(numero);
            if(!valido) {
                throw numero;
            }
        }catch(string e) {
            cout << "El numero " << e << " no es valido" << endl;
        }
    }
    return atof(numero.c_str());
}

string Ingreso::ingresarLetra() {
    string palabra;
    bool valido = false;
    while(!valido) {
        try {
            cin >> palabra;

```

```

        valido = validarLetra(palabra);
        if(!valido) {
            throw palabra;
        }
    }catch(string e) {
        cout << "La palabra " << e << " no es valida" <<
endl;
    }
}
return palabra;
}

bool Ingreso::validar(string numero) {
    int inicio = 0;
    if(numero.length() == 0) {
        return 0;
    }
    if(numero[0] == '+' || numero[0] == '-') {
        inicio = 1;
        if(numero.length() == 1) {
            return 0;
        }
    }
    for(int i=inicio; i<numero.length(); i++) {
        if(!isdigit(numero[i]) && numero[i] != '.') {
            return 0;
        }
    }
    return 1;
}

bool Ingreso::validarEntero(string numero) {
    int inicio = 0;
    if(numero.length() == 0) {
        return 0;
    }
    if(numero[0] == '+' || numero[0] == '-') {
        inicio = 1;
        if(numero.length() == 1) {
            return 0;
        }
    }
    for(int i=inicio; i<numero.length(); i++) {
        if(!isdigit(numero[i])) {
            return 0;
        }
    }
    return 1;
}

bool Ingreso::validarLetra(string palabra) {
    char c;
    for(int i=0; i<palabra.size(); i++) {
        c=palabra[i];
        if(isalpha(c) == 0) {
            if(isspace(c) == 0) {
                return 0;
            }
        }
    }
}

```

```

        }
    }
    return 1;
}

● Marquesina.h

#include "metrics.cpp"
#include "pdf.cpp"
#include <windows.h>

typedef void (*DemoFunction)(PDF &);

class Marquesina {
public:
    void gotoxy(int,int);
    void hacerMarquesina();
    void borrar();
};

● Marquesina.cpp

#include "Marquesina.h"
#include "Pantalla.cpp"
#include <iostream>

using namespace std;

int i,j;

void Marquesina::borrar() {
    Pantalla pantalla;
    if(i<=50) {
        pantalla.gotoxy(i,1);
        cout<< " ";
    } else {
        pantalla.gotoxy(j+5,1);
        cout<< "      ";
    }
}

void Marquesina::hacerMarquesina () {
    Pantalla pantalla;
    for (i=1; i<=50; i++) {
        pantalla.gotoxy(i,1);
        cout<<"Tetris";
        Sleep(100);
        borrar();
    }
    for (j=50; j>=1; j--) {
        pantalla.gotoxy(j,1);
        cout<<"Tetris";
        Sleep (100);
        borrar();
    }
}

```

```

}

void generaPDF(PDF &p) {
    p.setFont(PDF::HELVETICA, 12);
    string filename = "solucion.txt";
    ifstream file(filename.c_str());
    if(!file) {
        cout << "Error no se puede abrir el archivo: " << filename <<
endl;
    }
    string linea, archivo;
    while(getline(file, linea)) {
        archivo += linea + "\n";
    }
    vector<string> sv = p.wrapText(archivo, 300, true);
    int n = sv.size();
    for(int i = 0; i < n; i++) {
        p.setFont(PDF::HELVETICA, 12);
        p.showTextXY(sv[i], 100, 745-20 * i);
    }
}

void muestraPDF() {
    DemoFunction functions = generaPDF;
    int n = sizeof(functions) / sizeof(functions);
    for(int i = 0; i < n; i++) {
        ostringstream out;
        out << "solucion.pdf";
        string fileName = out.str();
        PDF pdf;
        functions(pdf);
        string errMsg;
        if(!pdf.writeToFile(fileName, errMsg)) {
            cout << errMsg << endl;
        } else {
            cout << endl << endl << endl << endl << "Se genero el PDF" <<
endl;
        }
        cout << endl;
    }
}

```

- Nodo.h

```

#ifndef __UML_Class_Diagram_2_Nodo_h
#define __UML_Class_Diagram_2_Nodo_h

class Nodo
{
public:
    Nodo* getSiguiente(void);
    void setsiguiente(Nodo* newSiguiente);
    Nodo* getAnterior(void);
    void setanterior(Nodo* newAnterior);
    Nodo();

protected:

```

```

private:
    Nodo *siguiente;
    Nodo *anterior;
    int numero;

friend class Numero;
};

#endif

• Nodo.cpp

#include "Nodo.h"

Nodo* Nodo::getSiguiente(void)
{
    return siguiente;
}

void Nodo::setsiguiente(Nodo* newSiguiente)
{
    siguiente = newSiguiente;
}

Nodo* Nodo::getAnterior(void)
{
    return anterior;
}

void Nodo::setanterior(Nodo* newAnterior)
{
    anterior = newAnterior;
}

Nodo::Nodo()
{
}

• Numero.h

#ifndef __UML_Class_Diagram_2_Biblioteca_h
#define __UML_Class_Diagram_2_Biblioteca_h

#include "Nodo.h"

int contador = 0;

class Numero
{
public:
    bool vacio();
    int tamano();
    void insertarInicio(int);
    void borrar(int,int);
    void imprimir(void);
    void insertarEntre(int,int);
    void insertarFinal(int);
};

```

```

protected:
private:
    Nodo *siguienteNumero;
    Nodo *anteriorNumero;

friend class Nodo;
};

#endif

●     Nodo.cpp

#include "Numero.h"
#include "Nodo.cpp"
#include <time.h>
#include <iostream>

using namespace std;

void Numero::insertarInicio(int newNumero)
{
    Nodo *auxiliar = new Nodo();
    auxiliar->numero = newNumero;
    if(siguienteNumero == NULL) {
        auxiliar->siguiente = auxiliar;
        auxiliar->anterior = auxiliar;
        siguienteNumero = auxiliar;
    }
    else {
        anteriorNumero = siguienteNumero->anterior;
        auxiliar->siguiente = siguienteNumero;
        auxiliar->anterior = anteriorNumero;
        siguienteNumero->anterior = auxiliar;
        anteriorNumero->siguiente = auxiliar;
        siguienteNumero = auxiliar;
    }
}

void Numero::borrar(int posicion,int number)
{
    if (posicion <= tamanio()) {
        if (posicion == 1) {
            if (tamanio() == 1) {
                delete siguienteNumero;
                siguienteNumero = NULL;
            }
            else {
                Nodo *bor = siguienteNumero;
                anteriorNumero = siguienteNumero->anterior;
                siguienteNumero = siguienteNumero->siguiente;
                anteriorNumero->siguiente = siguienteNumero;
                siguienteNumero->anterior = anteriorNumero;
                delete bor;
            }
        }
        Nodo *auxiliar = siguienteNumero;
        if(auxiliar->numero == number) {

```

```

        for (int f = 1; f <= posicion-1; f++) {
            auxiliar = auxiliar->siguiente;
        }
        Nodo *bor = auxiliar;
        anteriorNumero = auxiliar->anterior;
        auxiliar = auxiliar->siguiente;
        anteriorNumero->siguiente = auxiliar;
        auxiliar->anterior = anteriorNumero;
        delete bor;
    }
}

void Numero::imprimir(void)
{
    if (!vacio()) {
        Nodo *auxiliar = siguienteNumero;
        do {
            cout << auxiliar->numero << " ";
            auxiliar = auxiliar->siguiente;
        } while (auxiliar != siguienteNumero);
    }
}

void Numero::insertarEntre(int posicion, int newNumero)
{
    Nodo *auxiliar = new Nodo;
    auxiliar->numero = newNumero;
    auxiliar->siguiente = auxiliar;
    auxiliar->anterior= auxiliar;
    if(vacio()) {
        siguienteNumero = auxiliar;
        anteriorNumero = auxiliar;
    }else {
        if(posicion == 1) {
            insertarInicio(newNumero);
        }else if(posicion == (tamanio()+1)){
            insertarFinal(newNumero);
        }else if(posicion > 1 && posicion < (tamanio()+1)) {
            Nodo *temporal;
            temporal = siguienteNumero;
            for(int i=1; i<posicion; i++) {
                temporal = temporal->siguiente;
            }
            temporal->anterior->siguiente = auxiliar;
            auxiliar->siguiente = temporal;
            auxiliar->anterior = temporal->anterior;
            temporal->anterior = auxiliar;
        }
        else {
            cout << "Error, posicion invalida" << endl;
        }
    }
}

void Numero::insertarFinal(int newNumero)
{
}

```

```

Nodo *auxiliar = new Nodo();
auxiliar->numero = newNumero;
if (siguienteNumero == NULL) {
    auxiliar->siguiente = auxiliar;
    auxiliar->anterior = auxiliar;
    siguienteNumero = auxiliar;
}
else {
    anteriorNumero = siguienteNumero->anterior;
    auxiliar->siguiente = siguienteNumero;
    auxiliar->anterior = anteriorNumero;
    siguienteNumero->anterior = auxiliar;
    anteriorNumero->siguiente = auxiliar;
}
}

bool Numero::vacio() {
    if (siguienteNumero == NULL)
        return true;
    else
        return false;
}

int Numero::tamanio() {
    if (!vacio()) {
        Nodo *auxiliar = siguienteNumero;
        do {
            contador++;
            auxiliar = auxiliar->siguiente;
        } while (auxiliar != siguienteNumero);
    }
    return contador;
}

```

- Pantalla.h

```

#include <iostream>
#include <Windows.h>

using namespace std;

class Pantalla {
public:
    void gotoxy(int,int);
    int obtenerAncho();
    int obtenerAlto();
    void abrirPantalla();
    void pintar();
    void color(int);
    int menu(const char*,const char* [],int);
};

```

- Pantalla.cpp

```

#include "Pantalla.h"
#include <conio.h>

#define TECLA_ARRIBA 72

```

```

#define TECLA_ABAJO 80
#define TECLA_DERECHA 77
#define TECLA_IZQUIERDA 75
#define TECLA_ENTER 13

void Pantalla::gotoxy(int x, int y) {
    HANDLE hCon;
    COORD dwPos;
    dwPos.X = x;
    dwPos.Y = y;
    hCon = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleCursorPosition(hCon,dwPos);
}

int Pantalla::obtenerAncho() {
    int ancho;
    ancho = GetSystemMetrics(SM_CXSCREEN);
    return ancho;
}

int Pantalla::obtenerAlto() {
    int alto;
    alto = GetSystemMetrics(SM_CYSCREEN);
    return alto;
}

void Pantalla::abrirPantalla() {
    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);
}

void Pantalla::pintar() {
    for(int i=0; i<167; i++) {
        gotoxy(i,47);
        cout << "-";
    }
    for(int i=0; i<47; i++) {
        gotoxy(0,i);
        cout << "|";
        gotoxy(167,i);
        cout << "|";
    }
}

void Pantalla::color(int x) {
    SetConsoleTextAttribute(GetStdHandle (STD_OUTPUT_HANDLE),x);
}

int Pantalla::menu(const char *titulo,const char*opciones[], int numero) {
    Pantalla pantalla;
    bool repite=true;
    int seleccionar=1,tecla;
    int i;
    system("cls");
    do {
        pantalla.gotoxy(10,2);

```

```

        pantalla.color(1);
        printf("=====");
        pantalla.color(2);
        printf("\n\t\t\t\t%s\t\t\t\t\n",titulo);
        pantalla.color(3);
        for(i=0;i<numero;i++)
            printf("\t\t%s\n",opciones[i]);
        pantalla.gotoxy(16,4+seleccionar);
        pantalla.color(250);
        printf("%s",opciones[seleccionar-1]);
        pantalla.color(1);
        pantalla.gotoxy(10,15);
        printf("=====");
        pantalla.color(5);
        do {
            tecla=getch();
        }while(tecla!=TECLA_ARRIBA && tecla!=TECLA_ABAJO && tecla!=
TECLA_ENTER);
        switch(tecla) {
            case TECLA_ARRIBA: {
                seleccionar--;
                if(seleccionar==0)
                    seleccionar=numero;
            }
            break;
            case TECLA_ABAJO: {
                seleccionar++;
                if(seleccionar==numero+1)
                    seleccionar=1;
            }
            break;
            case TECLA_ENTER:
                repite=false;
                break;
        }
    }while(repite);
    pantalla.color(7);
    return seleccionar;
}

```

- Tetris.h

```
#include <windows.h>
#include <conio.h>
#include <fstream>

#define IZQUIERDA 75
#define DERECHA 77
#define ESC 27

using namespace std;

pthread_mutex_t ptmutex1;
int n = 1;
ofstream archivo;
int auxiliar = 1;
int aumenta = 1;
```

```

class Tetris {
private:
    char tecla;
public:
    void guardarPosicion();
    int moverNumero();
    void borrarNumero();
    void teclear();
    void mantener();
    void moverAbajo();
    void ubicar(int);
    void principal();
    void crearMenu();
};

● Tetris.cpp

#include "Marquesina.cpp"
#include "Tetris.h"
#include "VectorMatriz.cpp"
#include "Numero.cpp"
#include "Cifrado.cpp"

int numero = (rand()%9)+1;
int mantenerY = 47;
static HWND hConWnd;
HWND BCX_Bitmap(char*, HWND = 0, int = 0, int = 0, int = 0, int = 0,
int = 0, int = 0, int = 0);
int x = 1;
int y = 20;
HWND GetConsoleWndHandle(void);
Numero *lista = new Numero();

int **cuerpo;
void Tetris::guardarPosicion() {
    *(*cuerpo+n)+0) = x;
    *(*cuerpo+n)+1) = y;
    n++;
    if(n == tamano) {
        n = 1;
    }
}

int tamano = 2;
int cont = 0;
int Tetris::moverNumero() {
    Pantalla pantalla;
    for(int i=1; i<tamano; i++) {
        pantalla.gotoxy(*(*cuerpo+i)+0), *(*cuerpo+i)+1));
        cout << numero;
    }
    return numero;
}

void Tetris::borrarNumero() {
    Pantalla pantalla;

```

```

        pantalla.gotoxy(*(*cuerpo+n)+0), *(*cuerpo+n)+1));
        cout << " ";
    }

int direccion = 3;
void Tetris::teclear() {
    if(kbhit()) {
        tecla = getch();
        switch(tecla) {
            case DERECHA:
                if(direccion != 4) {
                    direccion=3;
                }
                break;
            case IZQUIERDA:
                if(direccion != 3) {
                    direccion=4;
                }
                break;
        }
    }
}

void Tetris::mantener() {
    Pantalla pantalla;
    pantalla.gotoxy(0,0);
}

void Tetris::moverAbajo() {
    if(direccion != 1) {
        direccion=2;
    }
}

void Tetris::ubicar(int auxiliarY) {
    Pantalla pantalla;
    auxiliarY = mantenerY;
    pantalla.pintar();
    cont++;
    if(y == auxiliarY) {
        pantalla.gotoxy(x,y-1);
        cout << numero;
        archivo << numero << endl;
        pantalla.gotoxy(1,1);
        cout << "La lista es: " << endl;
        pantalla.gotoxy(1,2);
        lista->insertarEntre(x,numero);
        lista->imprimir();
        numero = (rand()%9)+1;
        numero = moverNumero();
        y = 20;
        auxiliar++;
        if(auxiliar == 167) {
            auxiliar = 1;
            aumenta++;
            mantenerY--;
        }
    }
}

```

```

        }
        if(x == 0) {
            x = 1;
        }
        if(x == 167) {
            x = 166;
        }
    }

void Tetris::principal() {
    Pantalla pantalla;
    VectorMatriz generar;
    cuerpo = generar.inicializar(100);
    cuerpo = generar.encerar(100,cuerpo);
    pantalla.pintar();
    while(tecla != ESC) {
        borrarNumero();
        guardarPosicion();
        moverNumero();
        moverAbajo();
        teclear();
        ubicar(mantenerY);
        if(direccion == 2) {
            y++;
            Sleep(50);
        }
        if(direccion == 3) {
            x++;
            Sleep(50);
        }
        if(direccion == 4) {
            x--;
            Sleep(50);
        }
        Sleep(50);
    }
    pantalla.pintar();
}

void* funcion1(void *arg) {
    Tetris tetris;
    pthread_mutex_lock(&ptmutex1);
    Sleep(50);
    tetris.crearMenu();
    pthread_mutex_unlock(&ptmutex1);
    return NULL;
}

void* funcion2(void *arg) {
    Marquesina marquesina;
    pthread_mutex_lock(&ptmutex1);
    marquesina.hacerMarquesina();
    Sleep(5000);
    pthread_mutex_unlock(&ptmutex1);
    return NULL;
}

```

```

void* funcion3(void *arg) {
    pthread_mutex_lock(&ptmutex1);
    hConWnd = GetConsoleWindow();
    Sleep(1000);
    BCX_Bitmap("logo.bmp", hConWnd, 123, 600, 400, 0, 0);
    Sleep(8000);
    pthread_mutex_unlock(&ptmutex1);
    return NULL;
}

void Tetris::crearMenu() {
    Cifrado cifrar; //objeto para manejar el proceso
    char *nombreArchivo = (char*)malloc(30*sizeof(char)); //direccion del archivo
    char *claveArchivo = (char*)malloc(30*sizeof(char));
    *(claveArchivo+0)=0;
    Pantalla pantalla;
    pantalla.abrirPantalla();
    int opcion;
    const char *opciones[]={"1. Jugar","2. Abrir QR","3. Abrir codigo de barras","4. Abrir Ayuda","5. Seleccionar Archivo","6. Encriptar","7. Desencriptar",
                           "8. Salir"};
    opcion=pantalla.menu("TETRIS",opciones,8);
    bool bandera=false;
    do {
        switch(opcion) {
            case 1: {
                system("cls");
                principal();
                muestraPDF();
                opcion=pantalla.menu("TETRIS",opciones,8);
                archivo.close();
                break;
            }
            case 2: {
                system("cls");
                system("QR.png");
                opcion=pantalla.menu("TETRIS",opciones,8);
                break;
            }
            case 3: {
                system("cls");
                system("barra.jpeg");
                opcion=pantalla.menu("TETRIS",opciones,8);
                break;
            }
            case 4: {
                system("cls");
                system("tetris1.chm");
                opcion=pantalla.menu("TETRIS",opciones,8);
                break;
            }
            case 5: {
                system("cls");
                cout << "Nombre del archivo: ";
                cin >> nombreArchivo;
            }
        }
    } while(bandera==false);
}

```

```

        //si la cadena esta vacia no hace nada
        if(*(nombreArchivo+0)==0 )break;
        cifrar.leer(nombreArchivo);
        opcion=pantalla.menu("TETRIS",opciones,8);
        break;

    case 6:
        system("cls");
        //si la cadena esta vacia no hace nada
        if(*(nombreArchivo+0)==0 )break;
        cout << "Ingrese la clave: ";
        cin >> claveArchivo;
        //si la clave esta vacia no hace nada
        if(*(claveArchivo+0)==0 )break;
        cifrar.cifrar(claveArchivo ); //cifra
        cifrar.grabar(nombreArchivo ); //guarda
        opcion=pantalla.menu("TETRIS",opciones,8);
        break;

    case 7:
        system("cls");
        //si la cadena esta vacia no hace nada
        if(*(nombreArchivo)==0 )break;
        cout << "Ingrese la clave: ";
        cin >> claveArchivo;
        //si la clave esta vacia no hace nada
        if(*(claveArchivo)==0 )break;
        cifrar.descifrar(claveArchivo ); //descifra
        cifrar.grabar(nombreArchivo ); //guarda
        opcion=pantalla.menu("TETRIS",opciones,8);
        break;

    case 8:
        exit(0);
        break;

    default:
        system("cls");
        cout << endl << endl << endl << endl << "Opcion no
valida.\a\n";
        break;
    }
}while(bandera!=true);
}

HWND BCX_Bitmap(char* Text, HWND hWnd, int id, int X, int Y, int W, int H,
int Res, int Style, int Exstyle)
{
    HWND A;
    HBITMAP hBitmap;

    // set default style
    if (!Style) Style = WS_CLIPSIBLINGS | WS_CHILD | WS_VISIBLE | SS_BITMAP
| WS_TABSTOP;

    // form for the image

```

```

A = CreateWindowEx(Exstyle, "static", NULL, Style, X, Y, 0, 0, hWnd,
(HMENU)id, GetModuleHandle(0), NULL);

    // Text contains filename
    hBitmap = (HBITMAP)LoadImage(0, Text, IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE | LR_CREATEDIBSECTION);

    // auto-adjust width and height
    if (W || H) hBitmap = (HBITMAP)CopyImage(hBitmap, IMAGE_BITMAP, W, H,
LR_COPYRETURNORG);
    SendMessage(A, (UINT)STM_SETIMAGE, (WPARAM)IMAGE_BITMAP,
(LPARAM)hBitmap);
    if (W || H) SetWindowPos(A, HWND_TOP, X, Y, W, H, SWP_DRAWFRAME);
    return A;
}

HWND GetConsoleWndHandle(void)
{
    HWND hConWnd;
    OSVERSIONINFO os;
    char szTempTitle[64], szClassName[128], szOriginalTitle[1024];

    os.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
    GetVersionEx(&os);
    // may not work on WIN9x
    if (os.dwPlatformId == VER_PLATFORM_WIN32s) return 0;

    GetConsoleTitle(szOriginalTitle, sizeof(szOriginalTitle));
    sprintf(szTempTitle, "%u - %u", GetTickCount(), GetCurrentProcessId());
    SetConsoleTitle(szTempTitle);
    Sleep(60);
    // handle for NT and XP
    hConWnd = FindWindow(NULL, szTempTitle);
    SetConsoleTitle(szOriginalTitle);

    // may not work on WIN9x
    if (os.dwPlatformId == VER_PLATFORM_WIN32_WINDOWS)
    {
        hConWnd = GetWindow(hConWnd, GW_CHILD);
        if (hConWnd == NULL) return 0;
        GetClassName(hConWnd, szClassName, sizeof(szClassName));
        // while (_stricmp( szClassName, "ttyGrab" ) != 0 )
        while (strcmp(szClassName, "ttyGrab") != 0)
        {
            hConWnd = GetNextWindow(hConWnd, GW_HWNDNEXT);
            if (hConWnd == NULL) return 0;
            GetClassName(hConWnd, szClassName, sizeof(szClassName));
        }
    }
    return hConWnd;
}

● VectorMatriz.h

class VectorMatriz {
    private:
        int *vector;

```

```

        int **matriz;
public:
    int* inicializarVector(int);
    int* encerarVector(int,int*);
    int** inicializar(int);
    int** encerar(int,int**);
};

● VectorMatriz.cpp

#include "VectorMatriz.h"
#include <stdlib.h>

int* VectorMatriz::inicializarVector(int tamano) {
    vector = (int*)malloc(tamano *sizeof(int));
    return vector;
}

int* VectorMatriz::encerarVector(int tamano, int *vector) {
    for(int i=0; i<tamano; i++) {
        *(vector+i) = 0;
    }
    return vector;
}

int** VectorMatriz::inicializar(int tamano) {
    matriz = (int**)malloc(sizeof(int *)*tamano);
    for(int i=0; i<tamano; i++) {
        *(matriz+i) = (int*)malloc(sizeof(int)*tamano);
    }
    return matriz;
}

int** VectorMatriz::encerar(int tamano, int **matriz) {
    for(int i=0; i<tamano; i++) {
        for(int j=0; j<tamano; j++) {
            *(*(matriz+i)+j) = 0;
        }
    }
    return matriz;
}

```

Ejecución

La siguiente figura demuestra la ejecución del programa de Tetris, el cual consiste en ir generando números aleatorios los cuales al llegar al final de la pantalla se van almacenando en una lista circular doblemente enlazada, tal como se muestra en la parte superior izquierda en donde en cada iteración la lista se vuelve a actualizar con el dato agregado, para salir del juego es necesario que el usuario presione ESC.

```

La lista es:
6 8 6 5 2 4 8 9 1 9

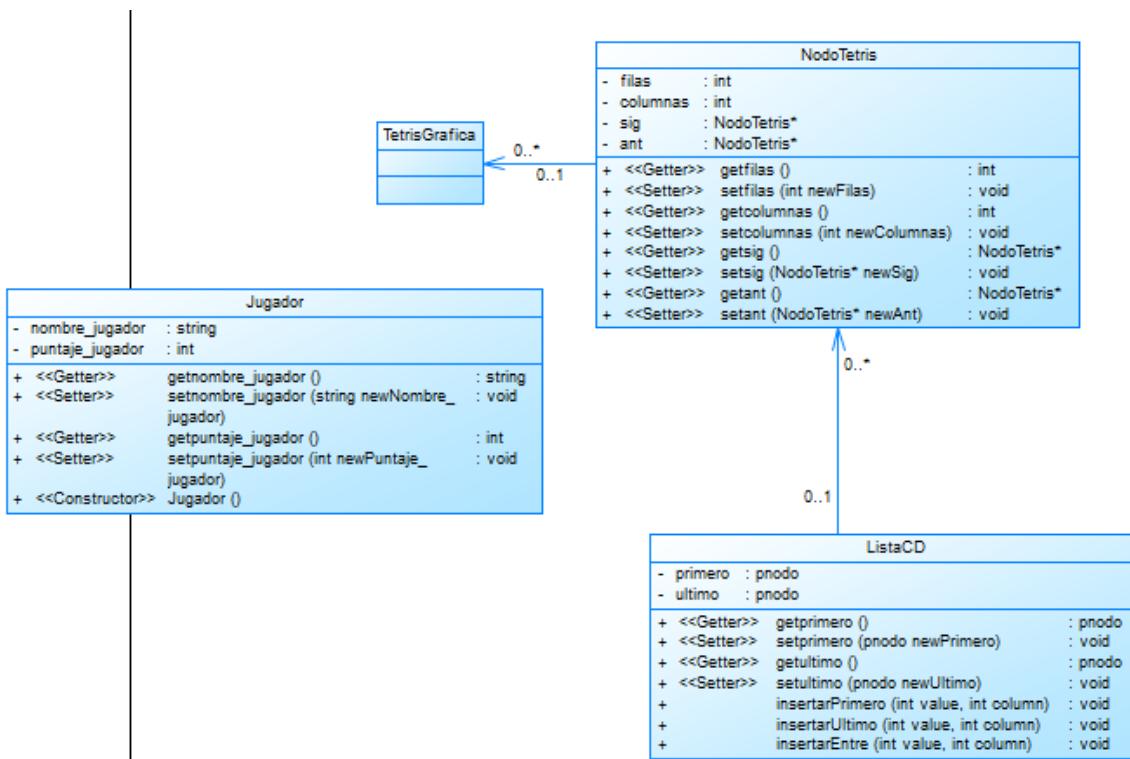
3

68952481

```

Grupo Galarza-Zurita

Modelado:



Codigo:

```
● Jugador.h
#include <iostream>
#include <stdio.h>

class Jugador{
    private:
        string nombre_jugador;
        int puntaje_jugador;
    public:
        Jugador(string, int);
        Jugador();
        string getNombre();
        void setNombre(string);
        int getPuntaje();
        void setPuntaje(int);
};

Jugador::Jugador(string name, int points){
    nombre_jugador=name;
    puntaje_jugador=points;
}

Jugador::Jugador(){
}

string Jugador::getNombre(){
    return nombre_jugador;
}

void Jugador::setNombre(string name){
    nombre_jugador=name;
}

int Jugador::getPuntaje(){
    return puntaje_jugador;
}

void Jugador::setPuntaje(int points){
    puntaje_jugador=points;
}

● ListaCD.h
#include <iostream>
#include <stdio.h>
#include <string>
#include <fstream>
#include "NodoTetris.h"

using namespace std;

typedef NodoTetris* pnodo;

class ListaCD{
    private:
```

```

        pnodo primero;
        pnodo ultimo;
    public:
        ListaCD();
        void insertarPrimero(int, int);
        void insertarUltimo(int, int);
        void insertarEntre(int, int);
        int chequeoEliminar(int);
        void mostrar();
        int numeroFila();
        bool listaVacia();
        int numeroElementos();
        void unirLista();
        pnodo returnNodo(int);
        void generarArchivo();
    };

ListaCD::ListaCD(){
    primero=NULL;
    ultimo=NULL;
}

bool ListaCD::listaVacia(){
    return primero == NULL;
}

void ListaCD::insertarPrimero(int value, int column){
    pnodo anterior;
    pnodo siguiente;
    if(listaVacia()) {
        primero = new NodoTetris(value,1,column);
        primero->sig=primero;
        primero->ant=primero;
        ultimo=primero;
    } else {
        if(ultimo==primero){
            anterior = primero;
            column=anterior->getColumnas()-1;
            if(column==0){
                column=1;
                anterior->setColumnas(2);
            }
            primero = new NodoTetris(value,numeroFila(),column);
            primero->sig=anterior;
            anterior->ant=primero;
            anterior->sig=primero;
            ultimo=anterior;
        }
        else{
            anterior = primero;
            column=anterior->getColumnas()-1;
            if(column==0){
                column=1;
            }
            primero = new NodoTetris(value,numeroFila(),column);
            primero->sig=anterior;
            anterior->ant=primero;
        }
    }
}

```

```

        ultimo->sig=primero;
        pnode aux=primero->sig;
        while(aux!=primero){
            aux->setColumnas(aux->ant->getColumnas()+1);
            aux=aux->sig;
        }
    }
}

void ListaCD::insertarUltimo(int value, int column){
    pnode temp;
    if(listaVacia()) {
        primero = new NodoTetris(value,1,column);
        primero->sig=primero;
        primero->ant=primero;
        ultimo=primero;
    } else {
        if(ultimo==primero){
            temp = new NodoTetris(value,numeroFila(),primero->getColumnas()+1);
            primero->sig=temp;
            temp->sig=primero;
            temp->ant=primero;
            ultimo=temp;
        }
        else{
            temp = new NodoTetris(value,numeroFila(),ultimo-
>getColumnas()+1);
            ultimo->sig=temp;
            temp->ant=ultimo;
            temp->sig=primero;
            ultimo=temp;
        }
    }
}

void ListaCD::insertarEntre(int value, int column){
    pnode aux;
    pnode sig;
    pnode control;
    aux=primero;
    if(listaVacia()){
        insertarPrimero(value,column);
    }else{
        if(column<=primero->getColumnas() && numeroFila()==primero->getFilas()){
            insertarPrimero(value,column);
        }
        else if(column>ultimo->getColumnas() && numeroFila()==ultimo-
>getFilas()){
            insertarUltimo(value,column);
        }
        else{
            if(numeroFila()!=primero->getFilas()){
                if(ultimo->getFilas()!=numeroFila()){
                    pnode temp=new NodoTetris(value,numeroFila(),column);
                    ultimo->sig=temp;
                    temp->ant=ultimo;
                }
            }
        }
    }
}

```

```

        temp->sig=primero;
        ultimo=temp;
    }else{
        while(aux->getFilas()!=numeroFila()){
            aux=aux->sig;
        }
        control=aux;
        if(column<=aux->getColumnas()){
            int x= aux->getColumnas()-1;
            if(x<=0){
                x=1;
            }
            pnodo temp=new
NodoTetris(value,numeroFila(),x);
            sig=aux->ant;
            sig->sig=temp;
            temp->ant=sig;
            temp->sig=aux;
            aux->ant=temp;
            aux=sig;
            temp=temp->sig;
            while(temp!=primero){
                temp->setColumnas(temp->ant-
>getColumnas()+1);
                temp=temp->sig;
            }
            if(ultimo->getColumnas()>10){
                aux->setColumnas(aux->getColumnas()-1);
                aux=aux->sig;
                while(aux!=primero){
                    aux->setColumnas(aux->getColumnas()-1);
                    aux=aux->sig;
                }
            }
        }else if(column>aux->getColumnas()){
            while(aux->getColumnas()+1!=column){
                aux=aux->sig;
            }
            if(aux->sig==primero){
                pnodo temp=new
NodoTetris(value,numeroFila(),ultimo->getColumnas()+1);
                ultimo->sig=temp;
                temp->ant=ultimo;
                temp->sig=primero;
                ultimo=temp;
                aux=control;
                if(ultimo->getColumnas()>10){
                    aux->setColumnas(aux->getColumnas()-1);
                    aux=aux->sig;
                    while(aux!=primero){
                        aux->setColumnas(aux->getColumnas()-1);
                        aux=aux->sig;
                    }
                    aux=control;
                    if(ultimo->getColumnas()>10){
                        aux->setColumnas(aux->getColumnas()-1);
                        aux=aux->sig;
                    }
                }
            }
        }
    }
}

```



```

        }else{
            while(aux->getColumnas () !=column+1) {
                aux=aux->sig;
            }
            pnodo temp=new
NodoTetris (value,numeroFila (),aux->ant->getColumnas ()+1);
            sig=aux->ant;
            sig->sig=temp;
            temp->ant=sig;
            temp->sig=aux;
            aux->ant=temp;
            aux=primero->sig;
            while(aux!=primero){
                aux->setColumnas (aux->ant-
>getColumnas ()+1);
                aux=aux->sig;
            }
            if(ultimo->getColumnas ()>10){
                aux->setColumnas (aux->getColumnas ()-1);
                aux=aux->sig;
                while(aux!=primero){
                    aux->setColumnas (aux->getColumnas ()-1);
                    aux=aux->sig;
                }
            }
        }
    }
void ListaCD::mostrar(){
    if(listaVacia()){
    }else{
        pnodo aux;
        aux = primero;
        int idC=aux->getColumnas ();
        int idF=aux->getFilas ();
        do{
            aux->imprimir ();
            aux=aux->sig;
        }while(aux->getColumnas () !=idC||aux->getFilas () !=idF);
        cout<<endl<<endl;
    }
}

int ListaCD::numeroElementos(){
pnodo aux;
aux = primero;
int idC=aux->getColumnas ();
int idF=aux->getFilas ();
int i=0;
if(primer==NULL){
    return 0;
}
if(primer==ultimo){
    return 1;
}
do{

```

```

        i++;
        aux=aux->sig;
    }while(aux->getColumnas()!=idC||aux->getFilas()!=idF);
    return i;
}

int ListaCD::chequeoEliminar(int puntaje){
    if(!listaVacia()){
        unirLista();
        int p=1;
        pnodo aux=primero;
        pnodo temp;
        if(numeroElementos()>=2){
            if(numeroElementos()==2){
                if(aux->getValorTetris()==aux->sig->getValorTetris()){
                    delete primero;
                    delete ultimo;
                    primero=NULL;
                    ultimo=NULL;
                }
            }
            else{
                while(aux!=ultimo){
                    temp=aux->sig;
                    if(aux->getValorTetris()==temp->getValorTetris() && aux->getFilas()==temp->getFilas()){
                        if(aux==primero){
                            primero=aux->sig->sig;
                            ultimo->sig=primero;
                            pnodo aux1=aux;
                            pnodo aux2=aux->sig;
                            aux=aux->sig->sig;
                            delete aux1;
                            delete aux2;
                        }
                        else{
                            if(aux->sig==ultimo){
                                ultimo=aux->ant;
                            }
                            aux->ant->sig=temp->sig;
                            temp->sig->ant=aux->ant;
                            pnodo aux1=aux;
                            pnodo aux2=aux->sig;
                            aux=aux->sig->sig;
                            delete aux1;
                            delete aux2;
                        }
                    }
                    unirLista();
                    puntaje+=100;
                    chequeoEliminar(puntaje);
                }
            }
            aux=aux->sig;
        }
        aux=primero;
        while(aux!=ultimo){
            temp=aux->sig;
            while(temp!=ultimo){

```

```

        if(aux->getColumnas ()==temp-
>getColumnas () &&aux->getFilas ()+1==temp->getFilas ()) {
                if(aux->getValorTetris ()==temp-
>getValorTetris ()) {
                        if(aux==primero) {
                                if(temp==ultimo) {
                                        pnodo aux1=primero;
                                        pnodo aux2=ultimo;
                                        primero=aux->sig;
                                        ultimo=temp->ant;
                                        ultimo->sig=primero;
                                        temp=ultimo->ant;
                                        aux=primero;
                                        delete aux1;
                                        delete aux2;
                                } else{
                                        pnodo aux1=primero;
                                        pnodo aux2=temp;
                                        primero=aux->sig;
                                        ultimo->sig=primero;
                                        temp->ant->sig=temp->sig;
                                        temp->sig->ant=temp->ant;
                                        temp=temp->ant;
                                        aux=primero;
                                        delete aux1;
                                        delete aux2;
                                }
                        } else{
                                if(temp==ultimo) {
                                        pnodo aux1=aux;
                                        pnodo aux2=ultimo;
                                        ultimo=temp->ant;
                                        ultimo->sig=primero;
                                        aux->ant->sig=aux->sig;
                                        aux->sig->ant=aux->ant;
                                        aux=aux->ant;
                                        temp=ultimo->ant;
                                        delete aux1;
                                        delete aux2;
                                } else{
                                        pnodo aux1=aux;
                                        pnodo aux2=temp;
                                        temp->ant->sig=temp-
                                                temp->sig->ant=temp-
                                                aux->sig->ant=aux->ant;
                                                aux->ant->sig=aux->sig;
                                                aux=aux->ant;
                                                temp=temp->ant;
                                                delete aux1;
                                                delete aux2;
                                }
                        }
                }
        }
        unirLista ();
        puntaje+=100;
        chequeoEliminar (puntaje);

```

```

        }
        temp=temp->sig;
    }
    aux=aux->sig;
}
}

int ListaCD::numeroFila(){
    return (numeroElementos()/10+1);
}

/*void ListaCD::ordenarLista(){
    if(!listaVacia()){
        pnode control=primero;
        while(control->getFilas()!=1) {
            primero=control->sig;
            ultimo->sig=control;
            control->ant=ultimo;
            control->sig=primero;
            ultimo=control;
            control=primero;
        }
        int i=1, f=1, row, ordenamiento=numeroElementos();
        bool flag=true, flag2;
        int c=numeroElementos()%10;
        if(c!=0){
            row=numeroFila();
        }else{
            row=numeroElementos()/10;
        }
        while(row>0) {
            pnode aux=control;
            if(ordenamiento>=10){
                if(flag==true){
                    if(aux->getColumnas()!=1){
                        aux->setColumnas(1);
                    }
                    flag=false;
                }
            }
            while(i!=10) {
                if(aux->getFilas()==f) {
                    if(aux->sig->getColumnas()!=aux-
>getColumnas()+1&&aux->sig->getFilas()==aux->getFilas()) {
                        aux->sig->setColumnas(aux->getColumnas()+1);
                    }
                    i++;
                    aux=aux->sig;
                }
                else{
                    aux->ant->sig=aux->sig;
                    aux->sig->ant=aux->ant;
                }
            }
        }
    }
}

```

```

        ultimo->sig=aux;
        aux->ant=ultimo;
        aux->sig=primero;
        ultimo=aux;
        aux=control;
        i=1;
    }
    if(i==10) {
        pnodo temp=aux->sig;
        if(temp->getColumnas ()>temp->sig-
>getColumnas () &&temp->getFilas ()==temp->sig->getFilas ()) {
            temp->ant->sig=temp->sig;
            temp->sig->ant=temp->ant;
            ultimo->sig=temp;
            temp->ant=ultimo;
            temp->sig=primero;
            ultimo=temp;
        }
        row--;
        control=aux->sig;
        flag=true;
        f++;
        ordenamiento-=10;
    }
    if(aux->sig==primero | |ordenamiento==1) {
        row--;
        i=10;
    }
}
pnodo aux=primero;
while(aux!=ultimo) {
    if(aux->sig->getColumnas ()>10) {
        aux->sig->setColumnas (aux->sig->getColumnas ()-10);
    }
    aux=aux->sig;
}
pnodo ayuda;
aux=primero;
while(aux!=ultimo) {
    if(aux->getColumnas ()>aux->sig->getColumnas () &&aux->getFilas ()==aux-
>sig->getFilas ()) {
        aux->ant->sig=aux->sig;
        aux->sig->ant=aux->ant;
        ayuda=aux->sig;
        while(ayuda->sig->getFilas ()==aux->getFilas ()) {
            ayuda=ayuda->sig;
        }
        ayuda->sig->ant=aux;
        aux->sig=ayuda->sig;
        ayuda->sig=aux;
        aux->ant=ayuda;
    }
    aux=aux->sig;
}
}

```

```

} */

void ListaCD::unirLista() {
    mostrar();
    if(!listaVacia()){
        if(numeroElementos()<10){
            pnodo aux=primero->sig;
            while(aux!=primero){
                aux->setColumnas(aux->ant->getColumnas()+1);
                aux->setFilas(1);
                aux=aux->sig;
            }
        }else{
            pnodo control=primero;
            pnodo aux=primero;
            int filas=1;
            int help=numeroElementos();
            while(help>10){
                aux->setColumnas(1);
                for(int i=1;i<=10;i++){
                    control=control->sig;
                }
                aux=aux->sig;
                while(aux!=control){
                    int p=aux->ant->getColumnas();
                    if(p>=10){
                        p=0;
                    }
                    aux->setColumnas(p+1);
                    aux->setFilas(filas);
                    aux=aux->sig;
                }
                filas++;
                aux=control;
                help-=10;
                if(help>=10){
                    aux->setColumnas(1);
                }
            }
            aux=aux->sig;
            while(aux!=primero){
                aux->setColumnas(aux->ant->getColumnas()+1);
                aux=aux->sig;
            }
        }
    }
}

pnodo ListaCD::returnNodo(int i){
    pnodo aux=primero;
    if(i==1){
        return primero;
    }
    for(int x=1;x<i;x++){
        aux=aux->sig;
    }
    return aux;
}

```

```

}

void ListaCD::generarArchivo(){
    pnodo aux=primero;
    ofstream file;
    file.open("Lista.txt");
    file<<"Lista Jugadores"<<"\n";

    do{
        file<<"Valor: "<<aux->getValorTetris()<<" Filas: "<<aux-
>getFilas()<<" Columnas: "<<aux->getColumnas()<<"\n";
        aux=aux->sig;
    }while(aux!=primero);
    file.close();
}

● NodoTetris.h
#include <iostream>
#include <stdio.h>

using namespace std;

class NodoTetris{
private:
    int valorTetris;
    int filas;
    int columnas;
    NodoTetris* sig;
    NodoTetris* ant;
public:
    NodoTetris(int,int,int);
    int getValorTetris();
    void setValorTetris(int);
    int getColumnas();
    void setColumnas(int);
    int getFilas();
    void setFilas(int);
    NodoTetris* getSiguiente();
    void setSiguiente(NodoTetris* );
    NodoTetris* getAnterior();
    void setAnterior(NodoTetris* );
    void imprimir();

    friend class ListaCD;
};

NodoTetris::NodoTetris(int value, int row, int column){
    valorTetris=value;
    filas=row;
    columnas=column;
    sig=NULL;
    ant=NULL;
}

int NodoTetris::getValorTetris(){
    return valorTetris;
}

```

```

void NodoTetris::setValorTetris(int value) {
    valorTetris=value;
}

int NodoTetris::getFilas () {
    return filas;
}

void NodoTetris::setFilas(int row) {
    filas=row;
}

int NodoTetris::getColumnas () {
    return columna;
}

void NodoTetris::setColumnas(int column) {
    columna=column;
}

NodoTetris* NodoTetris::getAnterior () {
    return ant;
}

void NodoTetris::setAnterior (NodoTetris* before) {
    ant=before;
}

NodoTetris* NodoTetris::getSiguiente () {
    return sig;
}

void NodoTetris::setSiguiente (NodoTetris* after) {
    sig=after;
}

void NodoTetris::imprimir () {
    cout<<valorTetris;
    cout<<" "<<filas;
    cout<<" "<<columna<<endl;
}

● TetrisGrafica.h
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <winbgim.h>
#include <windows.h>
#include <string>
#include <mmsystem.h>
#include <sstream>
#include "ListaCD.h"

class TetrisGrafica{
public:

```

```

        int grafica();
};

int TetrisGrafica::grafica(){
//    char soundfile[] = "C:/Users/JORGE GALARZA/Desktop/Proyecto/Proyecto
Segundo Parcial/Codigo/Tetris99.wav";
//    PlaySound((LPCSTR)soundfile, NULL, SND_FILENAME | SND_ASYNC );
    ListaCD list;
    srand(time(NULL));
    int y=1,j=550, filas=0, puntaje=0;
    initwindow(1300,650);
    rectangle(25,25,1275,625);
    settextstyle(8, 0, 4);
    setfillstyle(1, 0);
    char c;
    char* cN=new char[1];
    int x=rand()%10;
    string num;
    stringstream out;
    int posx=600, posy=50;
    sprintf(cN,"%d",x);
    while(c!=27||list.numeroFila()>=10){
        while(!kbhit()){
            setcolor(x+1);
            outtextxy(posx,posy,cN);
            Sleep(250);
            setcolor(15);
            rectangle(posx-5,posy-5,posx+20,posy+40);
            setcolor(0);
            floodfill(posx,posy,15);
            rectangle(posx-5,posy-5,posx+20,posy+40);
            setcolor(3);
            posy+=50;
            if(posy>(550-(filas*50))){
                int column=posx/120;
                list.insertarEntre(x,column);
//                list.mostrar();
                if(list.numeroElementos()>=2){
                    puntaje=list.chequeoEliminar(puntaje);
//                    list.mostrar();
                }
                filas=list.numeroFila();
                posy=50;
                posx=600;
                free(cN);
                cN=new char[1];
                x=rand()%10;
                sprintf(cN,"%d",x);
                setcolor(0);
                floodfill(600,325,15);
                for(int i=1;i<=list.numeroElementos();i++){
                    pnodo aux=list.returnNodo(i);
                    char* temp = new char[1];
                    sprintf(temp,"%d",aux->getValorTetris());
                    setcolor(aux->getValorTetris()+1);
                    outtextxy(aux->getColumnas()*120,550-((aux-
>getFilas()-1)*50),temp);
                }
            }
        }
    }
}

```

```

                free(temp);
            }
        }
    }
    c=getch();
    switch(c){
        case 75:
            if(posx==120)
                posx=120;
            else
                posx-=120;
            break;
        case 77:
            if(posx==1200)
                posx=1200;
            else
                posx+=120;
            break;
    }
}
PlaySound(NULL, NULL, 0);
list.generarArchivo();
return puntaje;
}
● menu.cpp
/*
***** UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE *****
** Estructura de Datos
** Nombre: Jorge Galarza - Kevin Zurita
** NRC: 2742
**
** Fecha de realizacion: 01/12/2019
** Fecha de modificacion: 12/12/2019
** Ing. Fernando Solis
*/
#include <iostream>
#include <winbgim.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>
#include <fstream>
#define TECLA_DERECHA 77
#define TECLA_IZQUIERDA 75
#define ENTER 13
#define F1 59
void* selloEspe(void * );
static HWND hConWnd;
HWND BCX_Bitmap(char*, HWND = 0, int = 0, int = 0, int = 0, int = 0,
int = 0, int = 0, int = 0);
HWND GetConsoleWndHandle(void);

using namespace std;
// Funcion gotoxy

```

```

void gotoxy(int x,int y){
    HANDLE hcon;
    hcon = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD dwPos;
    dwPos.X = x;
    dwPos.Y= y;
    SetConsoleCursorPosition(hcon,dwPos);
}

// Marquesina
void* marquesina(void *data){

    int a, b, c, n=0,letra=39,pos=1,cont=39,aux,cont1=39,auxg;
    char t[50] ="TETRIS: Proyecto Estructura de Datos ",auxt[39]=" ";

    do{
        for (a=0;a<42;a++) {
            aux=pos;
            for(b=39;b>cont;b--) {
                gotoxy(pos,1);
                cout<<t[b];
                pos--;
            }
            aux++;
            pos=aux;
            cont--;
            Sleep (75);
            if(a==40){
                break;
            }
        }
        for(a=3;a<70;a++) {
            gotoxy(a-1,1);
            cout<<" ";
            gotoxy(a,1);
            cout<<t;
            Sleep (75);
        }

        pos=70;
        auxg=69;
        for (a=0;a<41;a++) {
            gotoxy(auxg,1);
            cout<<" ";
            aux=pos;
            for(b=0;b<=cont1;b++) {
                gotoxy(pos,1);
                cout<<t[b];
                pos++;
            }
            cont1--;
            aux++;
            pos=aux;
            auxg++;
            Sleep (75);
        }
        cont1=39;
    }
}

```

```

        letra=39;
        pos=1;
        cont=39;
        c=0;
    }while (c==1);

}

// Cambia el color de la letras y dondo de la consola
void SetConsoleColour(WORD* Attributes, DWORD Colour){
    CONSOLE_SCREEN_BUFFER_INFO Info;
    HANDLE hStdout = GetStdHandle(STD_OUTPUT_HANDLE);
    GetConsoleScreenBufferInfo(hStdout, &Info);
    *Attributes = Info.wAttributes;
    SetConsoleTextAttribute(hStdout, Colour);
}

// 
void ResetConsoleColour(WORD Attributes){
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), Attributes);
}

// Remarca el metodo uno en la consola
void uno (WORD Attributes){
    cout << endl;
    SetConsoleColour(&Attributes, FOREGROUND_BLUE | BACKGROUND_INTENSITY | BACKGROUND_GREEN);
    cout<<"\tJugar ";
    ResetConsoleColour(Attributes);
    cout<<"\t CódigoQR \t Código de Barras \t Sello Pixepleado \t Salir
"<<endl;
}

// Remarca el metodo dos en la consola
void dos(WORD Attributes){
    cout << endl;
    cout<<"\tJugar \t ";
    SetConsoleColour(&Attributes, FOREGROUND_BLUE | BACKGROUND_INTENSITY | BACKGROUND_GREEN);
    cout<<"CódigoQR ";
    ResetConsoleColour(Attributes);
    cout<<"\t Código de Barras \t Sello Pixepleado \t Salir"<<endl;
}

// Remarca el metodo tres en la consola
void tres(WORD Attributes){
    cout << endl;
    cout<<"\tJugar \t CódigoQR \t ";
    SetConsoleColour(&Attributes, FOREGROUND_BLUE | BACKGROUND_INTENSITY | BACKGROUND_GREEN);
    cout<<"Código de Barras";
    ResetConsoleColour(Attributes);
    cout<<"\t Sello Pixepleado \t Salir";
}

```

```

// Remarca el metodo cuatro en la consola
void cuatro(WORD Attributes){
    cout << endl;
    cout<<"\tJugar \t CódigoQR \t Código de Barras \t ";
    SetConsoleColour(&Attributes, FOREGROUND_BLUE | BACKGROUND_INTENSITY | BACKGROUND_GREEN);
    cout<<"Sello Pixepleado";
    ResetConsoleColour(Attributes);
    cout << "\t Salir";
}

// Remarca el metodo cinco en la consola
void cinco(WORD Attributes){
    cout << endl;
    cout<<"\tJugar \t CódigoQR \t Código de Barras \t Sello Pixepleado \t";
    SetConsoleColour(&Attributes, FOREGROUND_BLUE | BACKGROUND_INTENSITY | BACKGROUND_GREEN);
    cout<<"Salir";
    ResetConsoleColour(Attributes);
}

// funcion principal
int main()
{
    system("color e");
    // declaracion de variables
    WORD Attributes=0;
    pthread_t thread1;
    pthread_create(&thread1, NULL, marquesina, NULL);
    int met,tecla,opc=0;
    cout<<"\n\n"<<endl;
    cout<<"\t Seleccione el metodo que desea usar \t\t\t\t\t\t Pulse
F1 para AYUDA"<<endl;
    cout << endl;
    cout<<"\tJugar \t CódigoQR \t Código de Barras \t Sello Pixepleado
\t Salir "<<endl;
    do{
        // lee la tecla que se pulsa
        do{
            tecla=getch();
        }while(tecla!=TECLA_DERECHA && tecla!=TECLA_IZQUIERDA && tecla!=ENTER
&& tecla!=F1);
        system("cls");
        cout<<"\n\n"<<endl;
        cout<<"\t Seleccione el metodo que desea usar \t\t\t\t\t\t pulse F1 para
ayuda"<<endl;
        // Dependiendo de la tecla que se pulse se usa un contador o abre la
        // ventana de ayuda
        switch (tecla){
            cout <<"\n\n"<<opc<<endl;
            case TECLA_DERECHA:
                opc++;
                if(opc==6){
                    opc=1;
                }
                break;
            case TECLA_IZQUIERDA:
                opc--;
        }
    }
}

```

```

        if(opc<=0) {
            opc=5;
        }
        break;
    case ENTER:
        met=opc;
        break;
    // en caso de presionar la tecla f1 se abre una ventana de ayuda
    case F1:
        system("AyudaTetris.chm");
        break;
    }

        // Dependiendo del contador entra a un caso en el que pinta el
metodo que se seleccionara
        switch(opc){
            case 1:
                uno(Attributes);
                break;
            case 2:
                dos(Attributes);
                break;
            case 3:
                tres(Attributes);
                break;
            case 4:
                cuatro(Attributes);
                break;
            case 5:
                cinco(Attributes);
                break;
        }

        // Cuando se pulsa enter entra al metodo que se encuentre
seleccionado
        switch(met){
            // ejecuta jugar
            case 1:
                SuspendThread(&thread1);
                system("Tetrisdelamuerte.exe");
                system("cls");
                opc=0;
                met=0;
            break;
            // Ejecuta código QR
            case 2:
                system("CodigoQR.png");
                opc=0;
                met=0;
            break;
            // Ejecuta codigo de barras
            case 3:
                system("CodigodeBarras.png");
                opc=0;
                met=0;
            break;
            // Ejecuta el sello pixepleado
        }
    }
}

```

```

    case 4:
        system("cls");
        void *status;
        pthread_t thread2;
        pthread_create (&thread2 , NULL , selloEspe , (void *) & thread1);
        pthread_join(thread2, &status);
        opc=0;
        met=0;
        break;

    }
    }while(met!=5);
    // Termina el programa
    system("cls");
    cout<<"\n\n"<<endl;
    cout<<"Gracias por Jugar !!!"<<endl;
    system("PAUSE");
    return 0;
}

HWND GetConsoleWndHandle(void)
{
    HWND hConWnd;
    OSVERSIONINFO os;
    char szTempTitle[64], szClassName[128], szOriginalTitle[1024];

    os.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
    GetVersionEx(&os);
    // may not work on WIN9x
    if (os.dwPlatformId == VER_PLATFORM_WIN32s) return 0;

    GetConsoleTitle(szOriginalTitle, sizeof(szOriginalTitle));
    sprintf(szTempTitle, "%u - %u", GetTickCount(), GetCurrentProcessId());
    SetConsoleTitle(szTempTitle);
    Sleep(60);
    // handle for NT and XP
    hConWnd = FindWindow(NULL, szTempTitle);
    SetConsoleTitle(szOriginalTitle);

    // may not work on WIN9x
    if (os.dwPlatformId == VER_PLATFORM_WIN32_WINDOWS)
    {
        hConWnd = GetWindow(hConWnd, GW_CHILD);
        if (hConWnd == NULL) return 0;
        GetClassName(hConWnd, szClassName, sizeof(szClassName));
        // while (_strcmp( szClassName, "ttyGrab" ) != 0 )
        while (strcmp(szClassName, "ttyGrab") != 0)
        {
            hConWnd = GetNextWindow(hConWnd, GW_HWNDNEXT);
            if (hConWnd == NULL) return 0;
            GetClassName(hConWnd, szClassName, sizeof(szClassName));
        }
    }
    return hConWnd;
}

HWND BCX_Bitmap(char* Text, HWND hWnd, int id, int X, int Y, int W, int H,
int Res, int Style, int Exstyle)
{

```

```

    HWND A;
    HBITMAP hBitmap;

    // set default style
    if (!Style) Style = WS_CLIPSIBLINGS | WS_CHILD | WS_VISIBLE | SS_BITMAP
| WS_TABSTOP;
    // form for the image
    A = CreateWindowEx(Exstyle, "static", NULL, Style, X, Y, 0, 0, hWnd,
(HMENU)id, GetModuleHandle(0), NULL);
    // Text contains filename
    hBitmap = (HBITMAP)LoadImage(0, Text, IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE | LR_CREATEDIBSECTION);
    // auto-adjust width and height
    if (W || H) hBitmap = (HBITMAP)CopyImage(hBitmap, IMAGE_BITMAP, W, H,
LR_COPYRETURNORG);
    SendMessage(A, (UINT)STM_SETIMAGE, (WPARAM)IMAGE_BITMAP,
(LPARAM)hBitmap);
    if (W || H) SetWindowPos(A, HWND_TOP, X, Y, W, H, SWP_DRAWFRAME);
    return A;
}
void* selloEspe(void *arg)
{
    pthread_mutex_t ptmutex1;
    pthread_mutex_lock(&ptmutex1);
    hConWnd = GetConsoleWndHandle();
    BCX_Bitmap("logo.bmp", hConWnd, 123, 150, 150, 0, 0);
    Sleep(5000);
    pthread_mutex_unlock(&ptmutex1);
}

● main.cpp
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <winbgim.h>
#include <windows.h>
#include <string>
#include "TetrisGrafica.h"
#include "Jugador.h"
#include <string>
#include <iostream>
#include <fstream>

void cifrado();
void decifrado();

using namespace std;
int main()
{
FILE * pFile;
    Jugador player;
    string nombre;
    cout << "\n\n\n\n\n\n" << endl;
    cout << "Ingrese su nombre" << endl;
    cin >> nombre;
}

```

```

player.setNombre(nombre);
TetrisGrafica tGraf;
int x=tGraf.grafica();
player.setPuntaje(x);
ofstream file;
file.open("Datos.txt");
file <<"Jugador: " <<nombre <<" Puntaje: "<<x;
file.close();
cifrado();
decifrado();

return 0;
}

void cifrado()
{
    char arch_in[]{"Datos.txt"};
    char arch_out[]{"DatosCifrado.txt"};
    char c;
    FILE *ofp,*ifp;
    ifp=fopen(arch_in,"r");
    ofp=fopen(arch_out,"w");
    system("txt2pdf.exe Datos.txt Datospdf.pdf -oao -pdfs60 -pps43 -ptc0 -
width3000 -height2000");
    while((c= fgetc(ifp))!=EOF)
    {
        fputc(c+5-3+7-9+4+2-3,ofp);

    }
    fclose(ofp);
    fclose(ifp);
}
void decifrado()
{
    char arch_in[]{"DatosDecifrado.txt"};
    char arch_out[]{"DatosCifrado.txt"};
    char c;
    FILE *ofp,*ifp;
    ofp=fopen(arch_in,"w");
    ifp=fopen(arch_out,"r");

    while((c= fgetc(ifp))!=EOF)
    {
        fputc(c-5+3-7+9-4-2+3,ofp);

    }
    fclose(ofp);
    fclose(ifp);
}

```

Ejecución

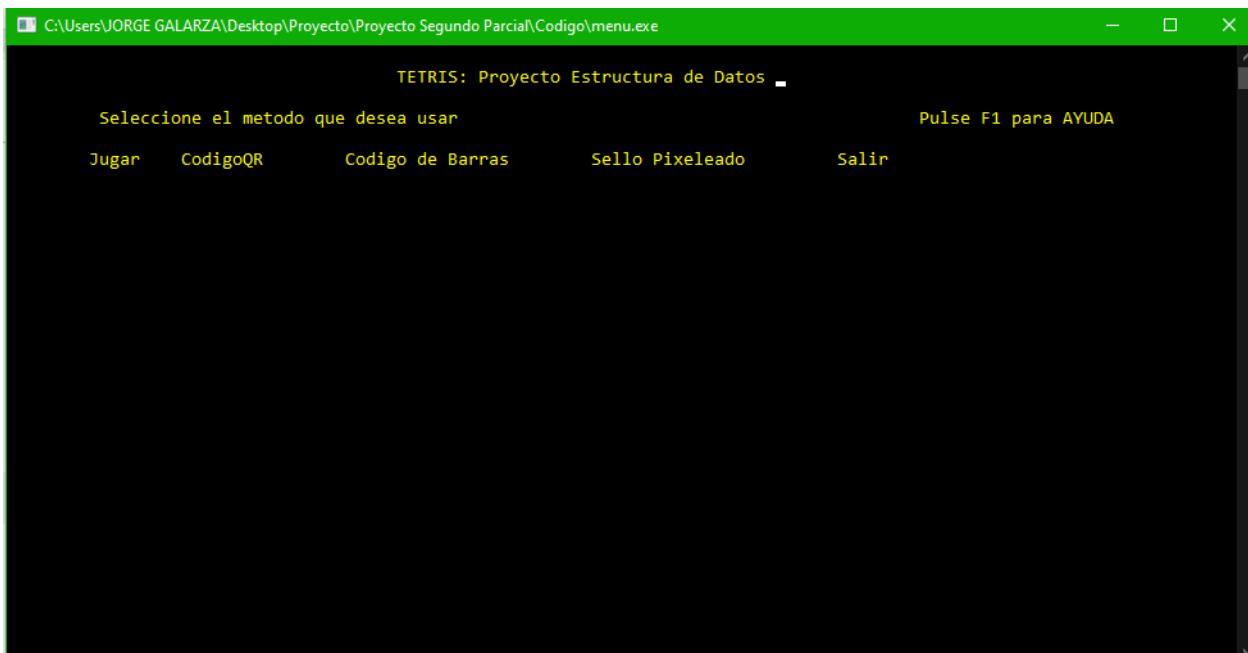


Figura 1: Ejecución del menú

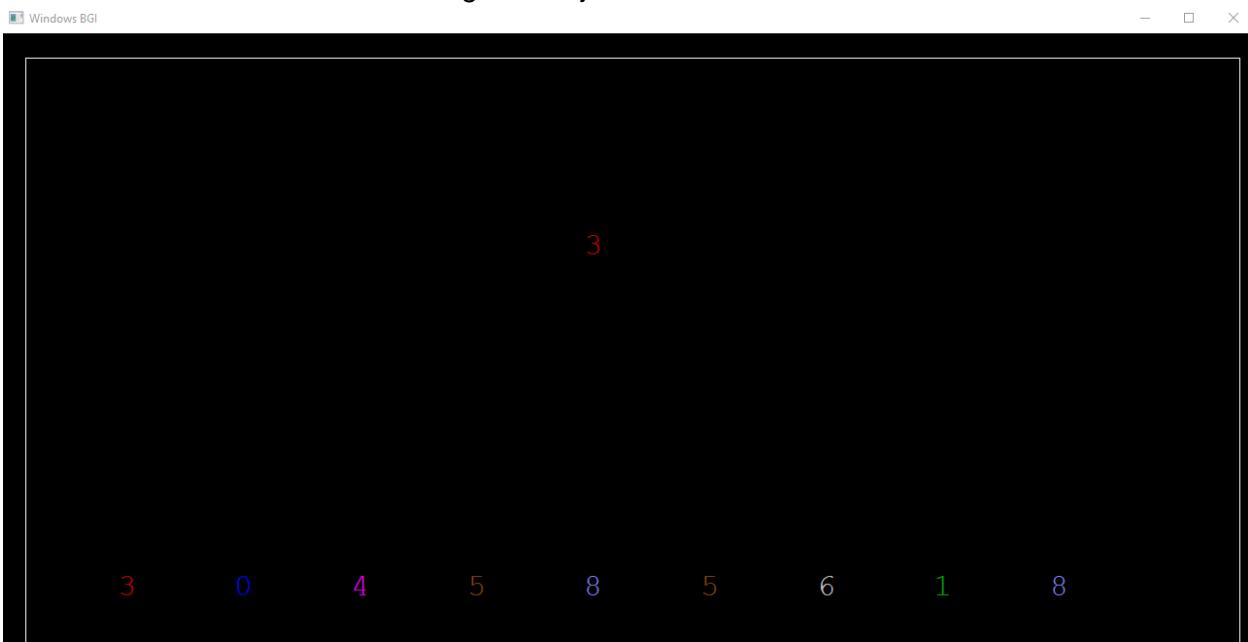


Figura 2: Ejecución del juego (Consola Gráfica)

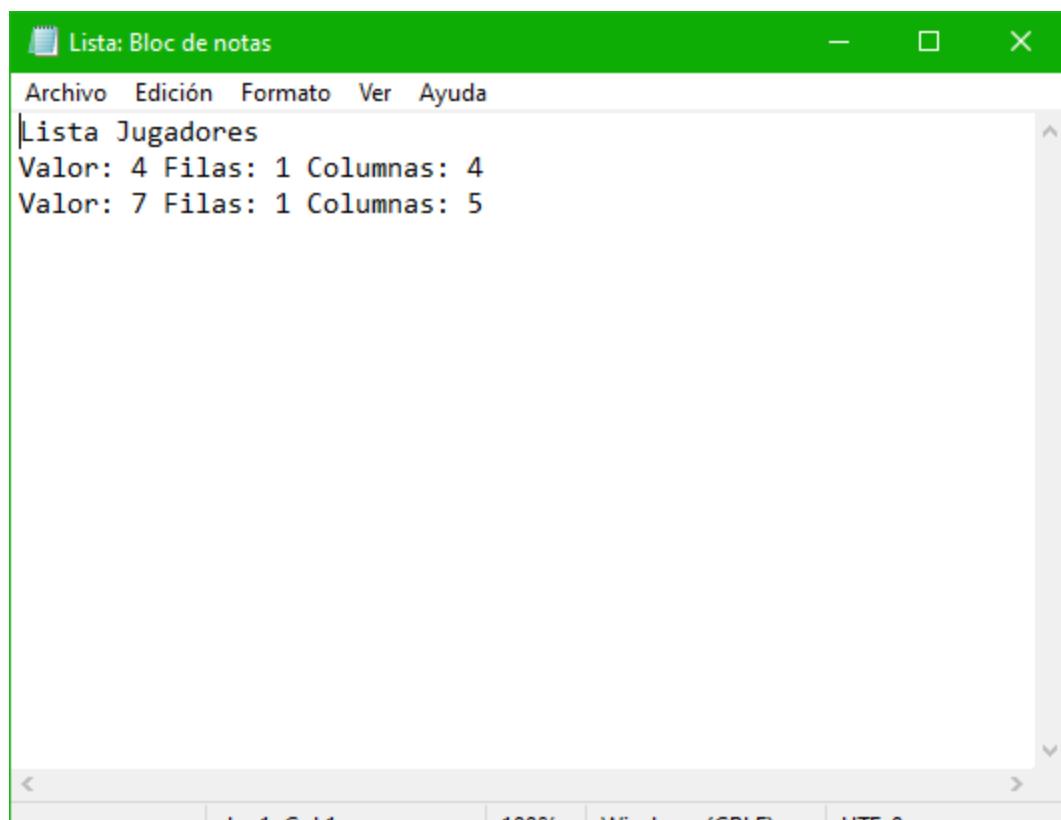
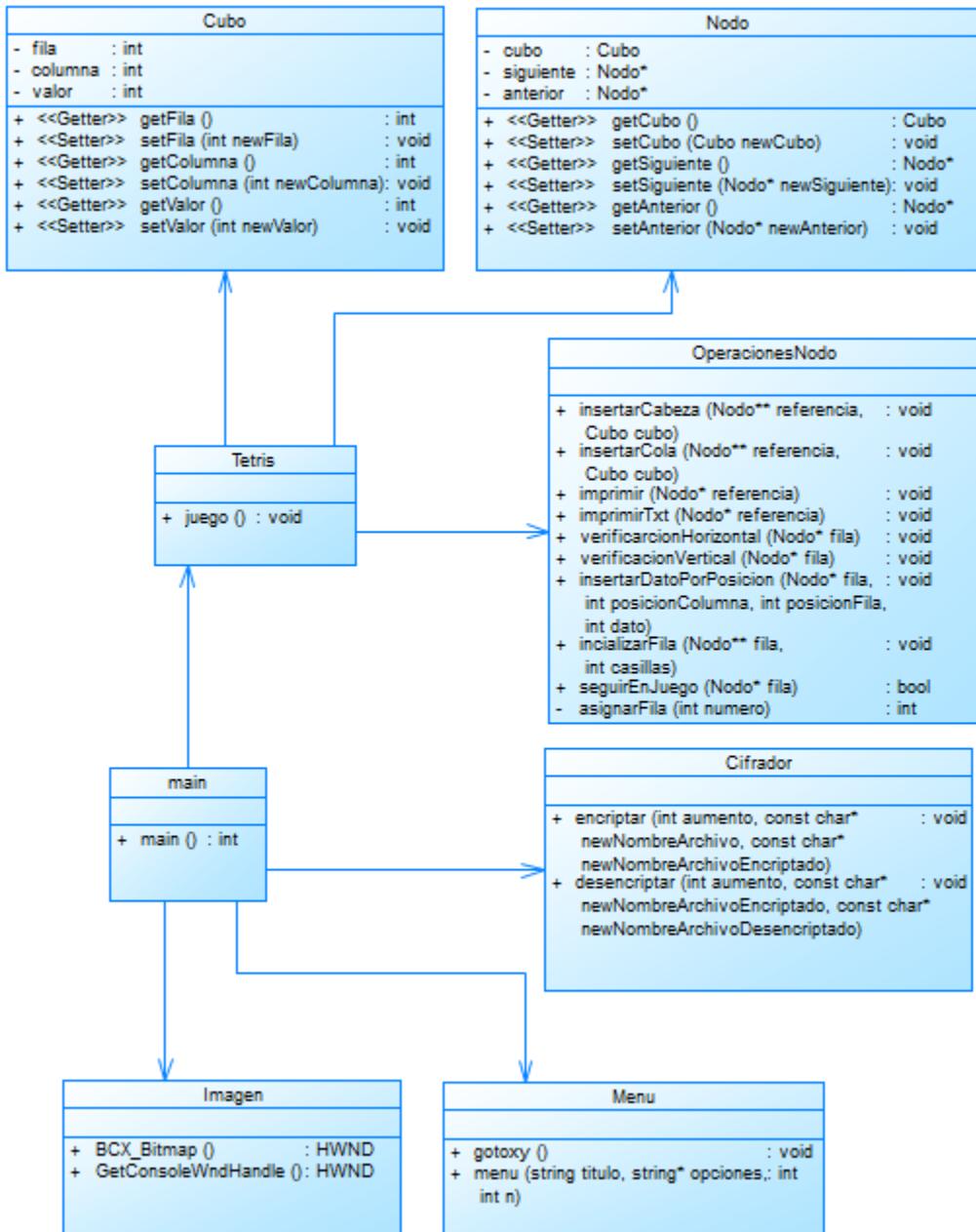


Figura 3: Documento.txt de la última lista.

Grupo Lopez-Zambrano

Modelado



Codigo

```

Clase main
#include "Tetris.h"
#include "Menu.h"
#include "Cifrador.h"

```

```

#include "Imagen.h"
#include "pthread.h"

int main()
{
    Menu* menu = new Menu();
    string titulo = "MENU DE OPCIONES";
    string* opciones = new string[8];
    *(opciones + 0) = "Juego";
    *(opciones + 1) = "Imagen";
    *(opciones + 2) = "Encriptar archivo";
    *(opciones + 3) = "Desencriptar archivo";
    *(opciones + 4) = "Generar barcode";
    *(opciones + 5) = "Generar PDF";
    *(opciones + 6) = "Ayuda";
    *(opciones + 7) = "Salir";

    int numOpciones = 8;
    int opcion;

    do {
        opcion = menu->menu(titulo, opciones, numOpciones);
        system("cls");
        switch (opcion) {
            case 1:
                Tetris tetris;
                tetris.juego();
                break;
            case 2:
                hConWnd = GetConsoleWndHandle();
                if (hConWnd)
                {
                    BCX_Bitmap((char*)"logo.bmp", hConWnd, 123, 1,
1, 0, 0);
                    //system("pause");
                    Sleep(10000);
                }
                system("pause");
                break;
            case 3:
                encriptar(2019, "Soluciones/solucion.txt",
"Soluciones/solucionencriptada.txt");
                break;
            case 4:
                desencriptar(2019,
"Soluciones/solucionencriptada.txt",
"Soluciones/soluciondesencriptada.txt");
                break;
            case 5:
                system("java -jar Generar_barcode.jar");
                break;
            case 6:
                system("java -jar textToPdf.jar");
                break;
            case 7:
                system("Tetris.chm");
        }
    } while (opcion != 7);
}

```

```

        break;
    case 8:
        cout << "Hasta pronto joven" << endl;
        break;
    }
} while (opcion != 8);

delete[] opciones;
return 0;
}

Clase Tetris
#include "Tetris.h"
#include <SFML/Graphics.hpp>
#include "SFML/Audio.hpp"
#include <time.h>
#include <iostream>
#include "OperacionesNodo.h"
#include "Cubo.h"
#include "Nodo.h"
using namespace sf;
using namespace std;

const int limiteAlto = 20;
const int limiteAncho = 10;

struct Point
{
    int x, y;
};

Point* encerarEstruct() {
    Point* temporal = new Point[4];
    for (int i = 0; i < 4; i++) {
        (temporal + i)->x = 0;
        (temporal + i)->y = 0;
    }
    return temporal;
}

Point* figuraBase = encerarEstruct();
Point* figuraPivot = encerarEstruct();

sf::SoundBuffer buffer;

int** encerarField() {
    int** temporal = new int* [limiteAlto];
    for (int i = 0; i < limiteAlto; ++i) {
        *(temporal + i) = new int[limiteAncho];
    }

    for (int i = 0; i < limiteAlto; i++) {
        for (int j = 0; j < limiteAncho; j++) {
            *(*(temporal + i) + j) = 0;
        }
    }
}

```

```

        }

        return temporal;
    }

int** field = encerarField();

bool check()
{
    for (int i = 0; i < 4; i++)
        if ((figuraBase + i)->x < 0 || (figuraBase + i)->x >=
limiteAncho || (figuraBase + i)->y >= limiteAlto) return 0;

        else if (*(*(field + (figuraBase + i))->y) + (figuraBase +
i)->x)) return 0;

    return 1;
};

int* numeros = new int[20];

int posicionColumna;
int posicionFila;
int auxiliarColorNum;
bool revisar;
bool seguirEnJuego = true;

void Tetris::juego()
{
    buffer.loadFromFile("Tetris.wav");

    Sound sound;
    sound.setBuffer(buffer);
    sound.play();
    sound.setLoop(10);
    sound.setVolume(10);
    sound.setAttenuation(10);

    int dimFila = 200;
    OperacionesNodo operaciones;

    Nodo* listaBase = NULL;
    Nodo* listaSuperior = NULL;

    operaciones.inicializarFila(&listaBase, dimFila);
    operaciones.inicializarFila(&listaSuperior, dimFila);

    srand(time(0));

    RenderWindow window(VideoMode(320, 480), "Tetris");

    Texture textCubo, textFondo, textMarco, textGameOver;
    textCubo.loadFromFile("images/tiles.png");
    textFondo.loadFromFile("images/background.png");
    textGameOver.loadFromFile("images/gameOver.png");
}

```

```

Sprite s(textCubo), background(textFondo), lose(textGameOver);

int dx = 0;
int colorNum = 1;
float timer = 0;

float auxDelay = 0;
float delay = 0.2 - auxDelay;
int contadorDificultad = 0;

Clock clock;

int x = 20;
for (int i = 0; i < 20; i++) {
    *(numeros + i) = x;
    x--;
}

while (window.isOpen() && seguirEnJuego)
{
    revisar = false;
    float time = clock.getElapsedTime().asSeconds();
    clock.restart();
    timer += time;

    Event e;
    while (window.pollEvent(e))
    {
        if (e.type == Event::Closed)
            window.close();

        if (e.type == Event::KeyPressed)
            if (e.key.code == Keyboard::Left) dx = -1;
            else if (e.key.code == Keyboard::Right) dx = 1;
    }

    if (Keyboard::isKeyPressed(Keyboard::Down)) delay = 0.05;

    //// <- Move -> ////

    for (int i = 0; i < 4; i++)
        *(figuraPivot + i) = *(figuraBase + i);
        (figuraBase + i)->x += dx;
    }

    if (!check())
        for (int i = 0; i < 4; i++)
            *(figuraBase + i) = *(figuraPivot + i);
    }
}

/////////Tick////////

```

```

if (timer > delay)
{
    for (int i = 0; i < 4; i++) {
        *(figuraPivot + i) = *(figuraBase + i);

        (figuraBase + i)->y += 1;
    }

    if (!check())
    {
        for (int i = 0; i < 4; i++) {
            *(*(field + (figuraPivot + i))->y) +
            (figuraPivot + i)->x) = colorNum;
        }

        colorNum = 1 + rand() % 7;
        //coloNum=1;

        for (int i = 0; i < 4; i++)
        {
            (figuraBase + i)->x = 4;
            (figuraBase + i)->y = 1;
        }
    }

    posicionColumna = (posicionColumna + 18) / 18;
    posicionFila = (posicionFila + 18) / 18;
    posicionFila = *(numeros + posicionFila - 1);

    cout << "Fila: " << posicionFila << endl;
    cout << "Columna: " << posicionColumna << endl;
    cout << auxiliarColorNum;

    operaciones.insertarDatosPorPosicion(listaBase,
    posicionFila, posicionColumna, auxiliarColorNum);
    revisar = true;

    cout << endl;

    //aqui reviso las filas
    operaciones.verificacionHorizontal(listaBase);
    operaciones.verificacionVertical(listaBase);
    operaciones.imprimir(listaBase);
    operaciones.imprimirTxt(listaBase);
    contadorDificultad++;

    bool banderaDificultad = false;

    if (auxDelay <= 0.15) {
        banderaDificultad = true;
    }

    if (contadorDificultad == 2 &&
banderaDificultad) {
        auxDelay += 0.01;
        contadorDificultad = 0;
    }
}

```

```

        timer = 0;
    }

/////////check lines/////////
Cubo cuboBase;
Cubo cuboPivot;
class Nodo* temporalBase = listaBase;
class Nodo* temporalPivot = listaBase;
int i_posision, j_posicion;

if (revisar) {

    int i = 0;
    while (i < 10) {
        temporalPivot = temporalPivot->getSiguiente();
        i++;
    }

    while (temporalBase->getSiguiente() != listaBase)
    {
        cuboPivot = temporalPivot->getCubo();
        cuboBase = temporalBase->getCubo();
        i_posision = *(numeros + cuboBase.getFila());

        j_posicion = cuboBase.getColumna() - 1;

        if (cuboBase.getValor() != cuboPivot.getValor()
&& cuboBase.getValor() == -1) {

            int auxFila = i_posision;
            int auxVector;
            for (auxFila; auxFila > 0; auxFila--) {
                *(*(field + auxFila) + j_posicion) =
                *(*(field + auxFila - 1) + j_posicion);
                auxVector = *(numeros + auxFila);

                operaciones.insertarDatoPorPosicion(listaBase, auxVector,
cuboBase.getColumna(), *(*(field + auxFila - 1) + j_posicion));
            }
        }

        if (cuboBase.getValor() == cuboPivot.getValor()
&& cuboBase.getValor() != 0) {
            int auxFila = i_posision;
            int auxVector;
            for (auxFila; auxFila > 2; auxFila--) {
                *(*(field + auxFila) + j_posicion) =
                *(*(field + auxFila - 2) + j_posicion);
                auxVector = *(numeros + auxFila);

                operaciones.insertarDatoPorPosicion(listaBase, auxVector,
cuboBase.getColumna(), *(*(field + auxFila - 2) + j_posicion));
            }
        }
        temporalBase = temporalBase->getSiguiente();
        temporalPivot = temporalPivot->getSiguiente();
    }
}

```

```

        }

        dx = 0; delay = 0.2 - auxDelay;

        //////////////draw///////////
        window.clear(Color::White);
        window.draw(background);

        for (int i = 0; i < limiteAlto; i++)
            for (int j = 0; j < limiteAncho; j++)
            {
                if (*(*(field + i) + j) == 0) continue;
                s.setTextureRect(IntRect(*(*(field + i) + j) *
18, 0, 18, 18));

                s.setPosition(j * 18, i * 18);
                s.move(28, 31); //offset
                window.draw(s);
            }

        for (int i = 0; i < 4; i++)
        {
            posicionColumna = (figuraBase + i)->x * 18;
            posicionFila = (figuraBase + i)->y * 18;

            s.setTextureRect(IntRect(colorNum * 18, 0, 18, 18));
            s.setPosition((figuraBase + i)->x * 18, (figuraBase +
i)->y * 18);

            s.move(28, 31); //offset
            window.draw(s);
        }

        auxiliarColorNum = colorNum;
        window.display();

        seguirEnJuego = operaciones.seguirEnJuego(listaBase);

        if (seguirEnJuego == false) {
            lose.move(0, 100);
            window.draw(lose);
            window.display();
            sound.pause();
            system("pause");
        }
    }

    if (seguirEnJuego == false) {
        cout << "Perdiste Perro";
    }
}

Clase Nodo
#include "OperacionesNodo.h"
#include "Cubo.h"

```

```

#include <iostream>
#include <fstream>

using namespace std;
fstream enter;
int valorClave = -1;
int n = 1;

///////////////////////////////
// Name:      OperacionesNodo::insertarCabeza(Nodo** referencia, Cubo
cubo)
// Purpose:   Implementation of OperacionesNodo::insertarCabeza()
// Parameters:
// - referencia
// - cubo
// Return:    void
///////////////////////////////

void OperacionesNodo::insertarCabeza(Nodo** referencia, Cubo cubo)
{
    if (*referencia == NULL)
    {
        class Nodo* temporal = new Nodo;
        temporal->setCubo(cubo);
        temporal->setAnterior(temporal);
        temporal->setSiguiente(temporal);

        *referencia = temporal;
        return;
    }
    class Nodo* temporal = new Nodo;
    class Nodo* ultimo = (*referencia)->getAnterior();

    temporal->setCubo(cubo);

    temporal->setSiguiente(*referencia);

    temporal->setAnterior(ultimo);

    (*referencia)->setAnterior(temporal);
    ultimo->setSiguiente(temporal);

    *referencia = temporal;
}

///////////////////////////////
// Name:      OperacionesNodo::insertarCola(Nodo** referencia, Cubo
cubo)
// Purpose:   Implementation of OperacionesNodo::insertarCola()
// Parameters:
// - referencia
// - cubo
// Return:    void
///////////////////////////////

void OperacionesNodo::insertarCola(Nodo** referencia, Cubo cubo)

```

```

{
    if (*referencia == NULL)
    {
        class Nodo* temporal = new Nodo;
        temporal->setCubo(cubo);
        temporal->setAnterior(temporal);
        temporal->setSiguiente(temporal);

        *referencia = temporal;
        return;
    }

    Nodo* ultimo = (*referencia)->getAnterior();

    class Nodo* temporal = new Nodo;
    temporal->setCubo(cubo);
    temporal->setSiguiente(*referencia);
    (*referencia)->setAnterior(temporal);
    temporal->setAnterior(ultimo);
    ultimo->setSiguiente(temporal);
}

///////////////////////////////
// Name:          OperacionesNodo::imprimir(Nodo* referencia)
// Purpose:      Implementation of OperacionesNodo::imprimir()
// Parameters:
// - referencia
// Return:       void
///////////////////////////////

void OperacionesNodo::imprimir(Nodo* referencia)
{
    class Nodo* temporal = referencia;
    Cubo cubo;
    int contador = 0;

    while (temporal->getSiguiente() != referencia)
    {
        cubo = temporal->getCubo();
        cout << cubo.getValor() << " ";
        contador++;
        if (contador > 9) {
            cout << endl;
            contador = 0;
        }

        temporal = temporal->getSiguiente();
    }

    cout << endl;
}

void OperacionesNodo::imprimirTxt(Nodo* referencia)
{
    class Nodo* temporal = referencia;
    Cubo cubo;
    int contador = 0;
}

```

```

if (n == 1) {
    enter.open("Soluciones/solucion.txt", fstream::out);
    enter << "Tablero: " << endl << endl;
}

enter << "Intento Numero: " << n << endl;

while (temporal->getSiguiente() != referencia)
{
    cubo = temporal->getCubo();
    enter << cubo.getValor() << " ";
    contador++;
    if (contador > 9) {
        enter << endl;
        contador = 0;
    }

    temporal = temporal->getSiguiente();
}

n++;
enter << endl;
}

////////////////////////////////////////////////////////////////
// Name:          OperacionesNodo::verificacionHorizontal (Nodo* fila)
// Purpose:       Implementation of
OperacionesNodo::verificacionHorizontal()
// Parameters:
// - fila
// Return:        void
////////////////////////////////////////////////////////////////

void OperacionesNodo::verificacionHorizontal(Nodo* fila)
{
    class Nodo* tempInicial = fila;
    class Nodo* tempSiguiente = fila->getSiguiente();
    class Nodo* tempSiguientex2 = tempSiguiente->getSiguiente();

    Cubo cuboInicial;
    Cubo cuboSiguiente;
    Cubo cuboSiguientex2;

    int auxiliar;
    bool bandera;

    while (tempInicial->getSiguiente() != fila)
    {
        bandera = false;
        cuboInicial = tempInicial->getCubo();
        cuboSiguiente = tempSiguiente->getCubo();
        cuboSiguientex2 = tempSiguientex2->getCubo();
        auxiliar = cuboInicial.getValor();
    }
}

```

```

        if (cuboInicial.getValor() ==
cuboSiguiente.getValor() && cuboInicial.getValor() != 0) {
            cuboInicial.setValor(valorClave);
            cuboSiguiente.setValor(valorClave);
            bandera = true;

            tempInicial->setCubo(cuboInicial);
            tempSiguiente->setCubo(cuboSiguiente);
        }

        if (auxiliar == cuboSiguientex2.getValor() && bandera&&auxiliar
!= 0) {
            cuboSiguientex2.setValor(valorClave);

            tempSiguientex2->setCubo(cuboSiguientex2);
        }

        tempInicial = tempInicial->getSigiente();
        tempSiguiente = tempSiguiente->getSigiente();
        tempSiguientex2 = tempSiguientex2->getSigiente();
    }
}

///////////////////////////////
// Name:          OperacionesNodo::verificacionVertical(Nodo* fila, Nodo*
primeraFila)
// Purpose:      Implementation of OperacionesNodo::verificacionVertical()
// Parameters:
// - fila
// - primeraFila
// Return:        void
///////////////////////////////

void OperacionesNodo::verificacionVertical(Nodo* fila)
{
    class Nodo* base = fila;
    class Nodo* pivot = fila;

    int i = 0;
    while (i < 10) {
        pivot = pivot->getSigiente();
        i++;
    }

    Cubo cuboBase;
    Cubo cuboPivot;

    while (base->getSigiente() != fila)
    {
        cuboBase = base->getCubo();
        cuboPivot = pivot->getCubo();

        if (cuboBase.getValor() == cuboPivot.getValor() &&
cuboBase.getValor() != 0 && cuboPivot.getValor() != 0) {
            cuboBase.setValor(valorClave);
            cuboPivot.setValor(valorClave);
            base->setCubo(cuboBase);
        }
    }
}

```

```

                pivot->setCubo(cuboPivot);
            }
            base = base->getSiguiente();
            pivot = pivot->getSiguiente();
        }

        cuboBase = base->getCubo();
        cuboPivot = pivot->getCubo();
        //este if sirve para verificar solo la ultima posicion de la fila
        if (cuboBase.getValor() == cuboPivot.getValor() &&
cuboBase.getValor() != 0 && cuboPivot.getValor() != 0) {
            cuboBase.setValor(valorClave);
            cuboPivot.setValor(valorClave);
            base->setCubo(cuboBase);
            pivot->setCubo(cuboPivot);
        }
    }

///////////////////////////////
// Name:          OperacionesNodo::insertarDatosPorPosicion(Nodo* fila)
// Purpose:       Implementation of
OperacionesNodo::insertarDatosPorPosicion()
// Parameters:
// - fila
// Return:        void
///////////////////////////////

void OperacionesNodo::insertarDatosPorPosicion(Nodo* fila, int
posicionfila,int posicionColumna,int dato)
{
    class Nodo* temporal = fila;
    Cubo cubo;

    while (temporal->getSiguiente() != temporal)
    {
        cubo = temporal->getCubo();
        if (cubo.getColumna() == posicionColumna&&cubo.getFila() ==
posicionfila) {
            cubo.setValor(dato);
            temporal->setCubo(cubo);
            break;
        }
        temporal = temporal->getSiguiente();
    }
}

void OperacionesNodo::inicializarFila(Nodo** lista, int casillas)
{
    int posicionColumna = 0;
    int posicionFila=0;
    int i = 0;
    Cubo cubo;

    while (i<=casillas) {
        cubo.setColumna(posicionColumna+1);

```

```

        posicionFila = asignarFila(i);
        cubo.setFila(posicionFila);

        cubo.setValor(0);

        insertarCola(lista, cubo);
        posicionColumna++;
        i++;

        if (posicionColumna > 9) {
            posicionColumna = 0;
        }
    }

}

bool OperacionesNodo::seguirEnJuego(Nodo* fila)
{
    class Nodo* temp;
    class Nodo* filaTemp=fila;

    Cubo cuboAuxiliar;
    int valor;
    int auxiliar = 180;

    int i = 0;
    while (i < auxiliar) {
        filaTemp = filaTemp->getSiguiente();
        i++;
    }

    while (filaTemp->getSiguiente() !=fila) {
        cuboAuxiliar = filaTemp->getCubo();
        valor = cuboAuxiliar.getValor();
        if (valor != 0) {
            return false;
        }
        filaTemp = filaTemp->getSiguiente();
    }

    return true;
}

int OperacionesNodo::asignarFila(int numero) {

    if (numero < 10) {
        return 1;
    }

    else if (numero < 20) {
        return 2;
    }

    else if (numero < 30) {
        return 3;
    }
}

```

```

else if (numero < 40) {
    return 4;
}

else if (numero < 50) {
    return 5;
}

else if (numero < 60) {
    return 6;
}

else if (numero < 70) {
    return 7;
}

else if (numero < 80) {
    return 8;
}

else if (numero < 90) {
    return 9;
}

else if (numero < 100) {
    return 10;
}

else if (numero < 110) {
    return 11;
}

else if (numero < 120) {
    return 12;
}

else if (numero < 130) {
    return 13;
}

else if (numero < 140) {
    return 14;
}

else if (numero < 150) {
    return 15;
}

else if (numero < 160) {
    return 16;
}

else if (numero < 170) {
    return 17;
}

else if (numero < 180) {

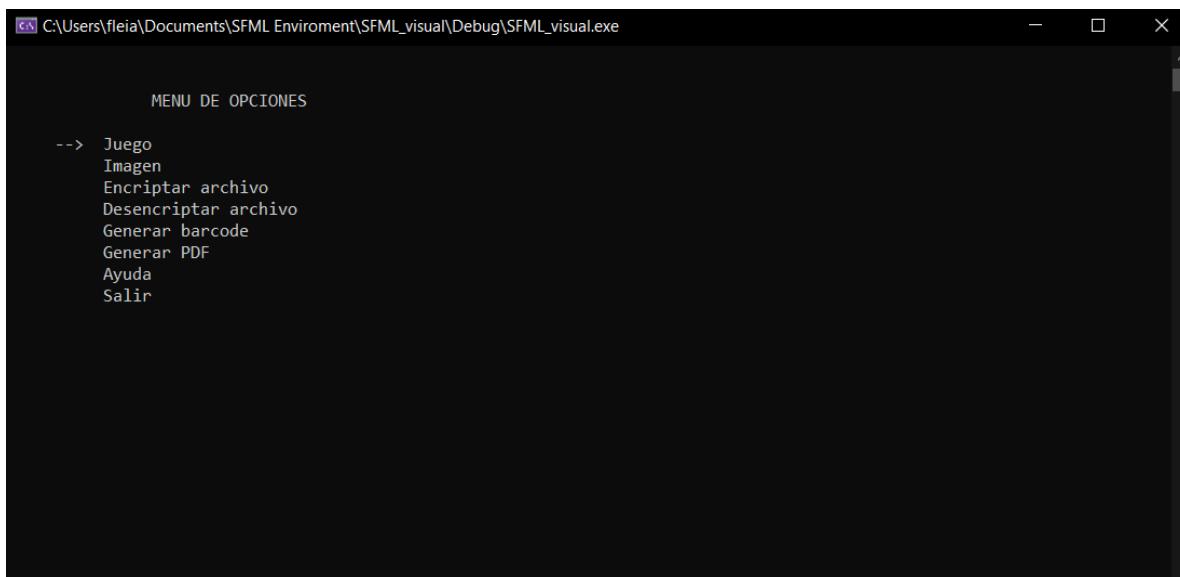
```

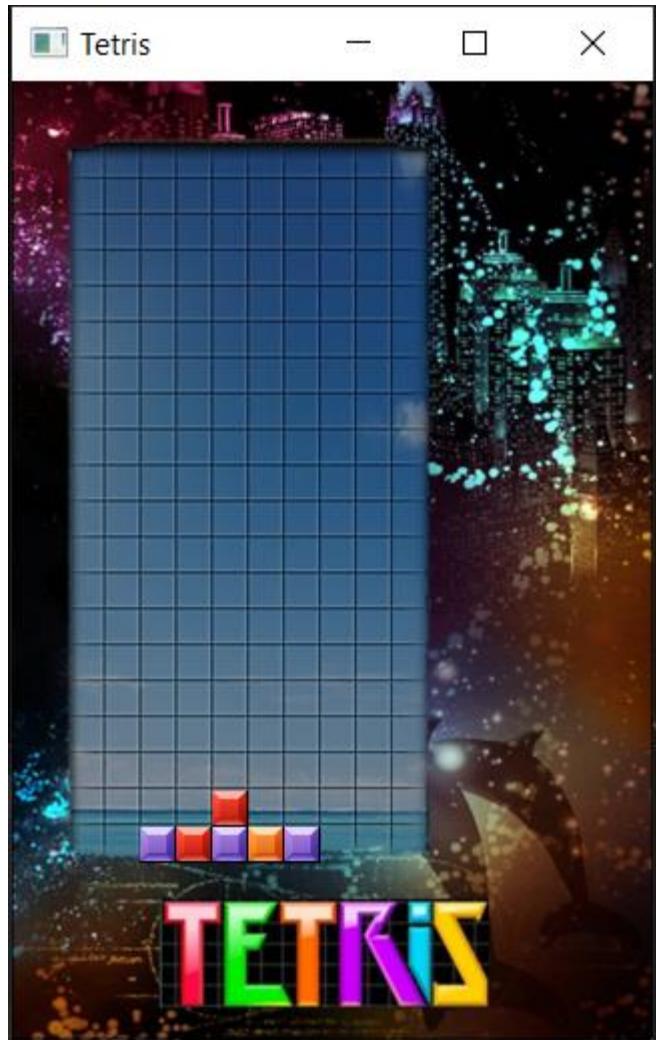
```
        return 18;
    }

    else if (numero < 190) {
        return 19;
    }

    else {
        return 20;
    }
}
```

Ejecución

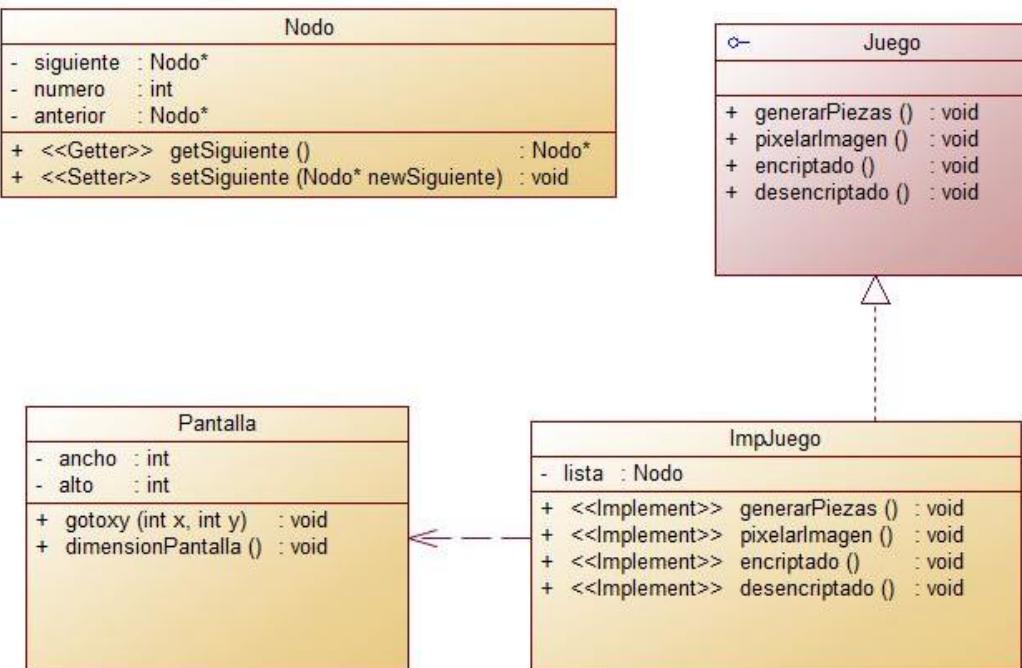






Grupo Bermúdez –Salazar

Modelado



Código

- Aplicativo.cpp

```

#include <iostream>
#include <stdio.h>
#include "ManejoNodo.h"
#include "Nodo.h"
#include "Pantalla.h"
#include <Windows.h>
#include <conio.h>
#include "ImpJuego.h"
#include <fstream>
#define DERECHA 77
#define IZQUIERDA 75
#define ESCAPE 27
/*
    Universidad de las Fuerzas Armadas ESPE
Carrera: Ingeniería de Software
Nombre: Kevin Salazar, Alan Bermudez
NRC:
Fecha de elaboracion:07/10/2019
Fecha de ultima modificacion:07/11/2019
Tetris
*/
using namespace std;

int main() {
    ManejoNodo mn;
    Pantalla pantalla;
    ImpJuego ij;

```

```

string nombre;
bool terminar = false;
int numero;
int siguienteNumero;
int puntuacion = 0;
char tecla;
int x = 20;
int y = 5;
int inicio;
int final;
int posicion = 1;
numero = ij.generarPiezas();
siguienteNumero = ij.generarPiezas();

cout << "Ingrese el nombre del jugador: ";
cin >> nombre;
pantalla.tablero(nombre, numero, puntuacion);
pantalla.gotoxy(x, y);

pantalla.ocultarCursor();

while (!terminar) {
    if (_kbhit()) {
        tecla = _getch();
        if (tecla == '1') {

            system("C:/Users/kevin/Desktop/ProyectoEstructuras/IndianaCroft/IndianaCroft/C++/IndianaCroft/IndianaCroft/AyudaIndianaCroft.chm");

        }

        if (tecla == ESCAPE) {

            terminar = true;
        }
        else if (tecla == DERECHA) {
            pantalla.gotoxy(x, y);
            cout << " ";
            x++;
            posicion++;
        }
        else if (tecla == IZQUIERDA) {
            pantalla.gotoxy(x, y);
            cout << " ";
            x--;
            posicion--;
        }

    }
    Sleep(100);
    pantalla.gotoxy(x, y);
    cout << numero;
    pantalla.gotoxy(x, y);
    y++;
    cout << " ";
}

```

```

pantalla.gotoxy(x, y);
cout << numero;

if (y == 43) {
    if (posicion < 1) {
        mn.insertar(numero, false);
    } else if (posicion > mn.getLongitud()) {
        mn.insertar(numero, true);
    }
    else {
        mn.insertarEnMedio(posicion, numero);
    }
    pantalla.borrarLinea();
    y = 5;
    x = 20;

    numero = siguienteNumero;
    siguienteNumero = ij.generarPiezas();
    pantalla.gotoxy(63, 10);
    cout << siguienteNumero;
    if (mn.getLongitud() > 1) {
        if (mn.borrar()) {
            pantalla.gotoxy(57, 13);
            puntuacion += 5;
            cout << puntuacion;
            pantalla.gotoxy(1, 1);
            cout << mn.getLongitud();
        }
    }
    pantalla.gotoxy(20 - (mn.getLongitud() / 2), 43);
    posicion = (mn.getLongitud() / 2) + 1;
    mn.imprimir();
}

if (mn.getLongitud()==40) {
    system("cls");
    pantalla.gotoxy(20, 40);
    cout << "USTED HA PERDIDO :('<<endl;
    system("pause");
    exit(0);
}
}

mn.imprimir();
fstream doc;
doc.open("Lista.txt", fstream::in);

doc.close();
system("cls");
system("txt2pdf.exe Lista.txt Lista.pdf -oao -pfs60 -pps43 -ptc0 -
width3000 -height2000");

}

●      EncryptDEcript.cpp
#include <iostream>
#include <string.h>

```

```

#include <fstream>

using namespace std;

string encrypt(string key, string message) {

    string newMessage = "";
    char letter;

    for (int i = 0; i < strlen(message.c_str()); i++) {
        letter = message.at(i);
        if (letter == 'a' || letter == 'e' || letter == 'i' ||
letter == 'o' || letter == 'u' || letter == 'A' || letter == 'E' ||
letter == 'I' || letter == 'O' || letter == 'U') {
            newMessage = newMessage + key + letter;
        } else {
            newMessage = newMessage + letter;
        }
    }

    return newMessage;
}

string decrypt(string key, string message) {

    string newMessage = "";
    string auxiliarKey = "";
    char letter;
    for (int i = 0; i < strlen(message.c_str()); i++) {
        if (i < strlen(message.c_str()) - 2) {
            letter = message.at(i + 2);
            auxiliarKey = message.at(i) + message.at(i + 1);
            if (strcmp(key.c_str(), auxiliarKey.c_str()) && (letter == 'a' || letter == 'e' || letter ==
'i' || letter == 'o' || letter == 'u' || letter == 'A' || letter ==
'E' || letter == 'I' || letter == 'O' || letter == 'U')) {
                newMessage = newMessage + message.at(i + 2);
                i = i + 2;
            } else {
                newMessage = newMessage + message.at(i);
            }
        } else {
            newMessage = newMessage + message.at(i);
        }
    }

    return newMessage;
}

int main(){
    ifstream enter;
    enter.open("encriptar.txt",ios::in);
    string clave;

```

```

        string mensaje;
        cout<< "Ingrese la Clave" << endl;
        cin>>clave;

while(!enter.eof()){
    getline(enter,mensaje);

    }

mensaje = encrypt(clave,mensaje);
cout << mensaje << endl;

cout << decrypt(clave,mensaje);

return 0;
}

● Imagen.cpp
#include <allegro.h>

void init();
void deinit();

int main() {
    init();

    BITMAP *imagen;
    imagen=load_bitmap("espe.bmp", NULL);
    draw_sprite(screen,imagen,0,0);
    readkey();
    deinit();
    return 0;
}
END_OF_MAIN()

void init() {
    int depth, res;
    allegro_init();
    depth = desktop_color_depth();
    if (depth == 0) depth = 32;
    set_color_depth(depth);
    res = set_gfx_mode(GFX_AUTODETECT_WINDOWED, 640, 480, 0, 0);
    if (res != 0) {
        allegro_message(allegro_error);
        exit(-1);
    }

    install_timer();
    install_keyboard();
    install_mouse();
    /* add other initializations here */
}

void deinit() {
    clear_keybuf();
    /* add other deinitializations here */
}
● ImpJuego.cpp

```

```
*****
* Module:  ImpJuego.cpp
* Author:  Alan
* Modified: lunes, 18 de noviembre de 2019 23:20:19
* Purpose: Implementation of the class ImpJuego
*****/

#include "ImpJuego.h"
#include <time.h>
#include <stdlib.h>

///////////////////////////////
// Name:      ImpJuego::generarPiezas()
// Purpose:   Implementation of ImpJuego::generarPiezas()
// Return:    void
///////////////////////////////

///////////////////////////////
// Name:      ImpJuego::pixelarImagen()
// Purpose:   Implementation of ImpJuego::pixelarImagen()
// Return:    void
///////////////////////////////

int ImpJuego::generarPiezas()
{
    int numero;
    srand(time(NULL));
    numero = rand() % 9 + 1;
    return numero;
}

void ImpJuego::pixelarImagen(void)
{
    // TODO : implement
}

///////////////////////////////
// Name:      ImpJuego::encriptado()
// Purpose:   Implementation of ImpJuego::encriptado()
// Return:    void
///////////////////////////////

void ImpJuego::encriptado(void)
{
    // TODO : implement
}

///////////////////////////////
// Name:      ImpJuego::desencriptado()
// Purpose:   Implementation of ImpJuego::desencriptado()
// Return:    void
///////////////////////////////

void ImpJuego::desencriptado(void)
```

```

{
    // TODO : implement
}
ImpJuego.h
*****
* Module: ImpJuego.h
* Author: Alan
* Modified: lunes, 18 de noviembre de 2019 23:20:19
* Purpose: Declaration of the class ImpJuego
*****
***** /



#ifndef __Tetris_2_ImpJuego_h
#define __Tetris_2_ImpJuego_h


class ImpJuego
{
public:
    int generarPiezas();
    void pixelarImagen(void);
    void encriptado(void);
    void desencriptado(void);

protected:
private:

};

● ManejoNodo.cpp
#include "ManejoNodo.h"
#include <iostream>
#include <fstream>
#include <stdlib.h>

using namespace std;

void ManejoNodo::insertar(int dato, bool posicion) {
    Nodo* nuevo = new Nodo();
    nuevo->setNumero(dato);
    longitud++;
    if (primero == NULL) {
        primero = nuevo;
        ultimo = nuevo;
        primero->setSiguiente(primero);
        primero->setAnterior(ultimo);
    }
    else if(posicion){
        ultimo->setSiguiente(nuevo);
        nuevo->setAnterior(ultimo);
        nuevo->setSiguiente(primero);
        ultimo = nuevo;
        primero->setAnterior(ultimo);
    }
    else {
        nuevo->setSiguiente(primero);

```

```

        nuevo->setAnterior(ultimo);
        primero->setAnterior(nuevo);
        primero = nuevo;
        ultimo->setSiguiente(primer);
    }
}

void ManejoNodo::insertarEnMedio(int posicion, int dato){
    Nodo* nuevoNodo = new Nodo();
    Nodo* actual = primero;
    nuevoNodo->setNumero(dato);
    for (int i = 1; i <= posicion; i++) {
        actual = actual->getSiguiente();
        if (i+1 == posicion) {
            actual->getSiguiente()->setAnterior(nuevoNodo);
            nuevoNodo->setSiguiente(actual->getSiguiente());
            actual->setSiguiente(nuevoNodo);
            nuevoNodo->setAnterior(actual);
        }
    }
}

void ManejoNodo::imprimir() {
    fstream enter;
    enter.open("Lista.txt", fstream::out);
    enter << "Lista Numeros Tetris" << endl;

    Nodo* actual = new Nodo();
    actual = primero;
    if (primero != NULL) {
        do {
            enter << actual->getNumero() << " ";
            cout << actual->getNumero();
            actual = actual->getSiguiente();
        } while (actual != primero);
    }
}

bool ManejoNodo::borrar()
{
    Nodo* actual = primero->getSiguiente();
    bool continuar = true;
    bool toReturn = false;
    while (continuar) {

        if (primero->getNumero() == ultimo->getNumero()) {
            if (longitud==2) {
                primero = NULL;
                ultimo = NULL;
                continuar = false;
                longitud = longitud - 2;
            }else{
                primero = primero->getSiguiente();
                ultimo = ultimo->getAnterior();
                primero->setAnterior(ultimo);
                ultimo->setSiguiente(primer);
            }
        }
    }
}

```

```

        longitud = longitud - 2;
    }
    toReturn = true;
}
else if (primero->getNumero() == primero->getSiguiente()->getNumero()) {
    primero = primero->getSiguiente()->getSiguiente();
    primero->setAnterior(ultimo);
    ultimo->setSiguiente(primer);
    longitud = longitud - 2;
    toReturn = true;;
}
else if (ultimo->getNumero() == ultimo->getAnterior()->getNumero()) {
    ultimo = ultimo->getAnterior()->getAnterior();
    ultimo->setSiguiente(primer);
    primero->setAnterior(ultimo);
    longitud = longitud - 2;
    toReturn = true;;
}
//Implementar el borrar en medio de la lista
else if (actual->getSiguiente()->getNumero() == actual->getNumero() && actual->getSiguiente() != ultimo) {
    actual->getAnterior()->setSiguiente(actual->getSiguiente()->getSiguiente());
    actual->getSiguiente()->getSiguiente()->setAnterior(actual->getAnterior());
    longitud = longitud - 2;
    toReturn = true;;
}
else if (actual->getSiguiente() == ultimo) {
    primero = actual->getSiguiente()->getSiguiente();
    ultimo = actual->getSiguiente();
    continuar = false;
}
actual = actual->getSiguiente();
}

return toReturn;
}

int ManejoNodo::getLongitud()
{
    return longitud;
}
● ManejoNodo.h
#pragma once
#include "Nodo.h"
#include <iostream>
using namespace std;

class ManejoNodo
{
public:
    void insertar(int dato, bool posicion);
    void insertarEnMedio(int posicion, int dato);

```

```

        void imprimir();
        bool borrar();
        int getLongitud();
private:
    Nodo* primero = NULL;
    Nodo* ultimo = NULL;
    int longitud = 0;
};

Nodo.cpp
//****************************************************************************
 * Module:  Nodo.cpp
 * Author:  Alan
 * Modified: lunes, 18 de noviembre de 2019 23:20:19
 * Purpose: Implementation of the class Nodo
//****************************************************************************

#include "Nodo.h"

///////////////////////////////
// Name:      Nodo::getSiguiente()
// Purpose:   Implementation of Nodo::getSiguiente()
// Return:    Nodo*
///////////////////////////////

Nodo* Nodo::getSiguiente(void)
{
    return siguiente;
}

///////////////////////////////
// Name:      Nodo::setSiguiente(Nodo* newSiguiente)
// Purpose:   Implementation of Nodo::setSiguiente()
// Parameters:
// - newSiguiente
// Return:    void
///////////////////////////////

void Nodo::setSiguiente(Nodo* newSiguiente)
{
    siguiente = newSiguiente;
}

Nodo* Nodo::getAnterior(void)
{
    return anterior;
}

void Nodo::setAnterior(Nodo* newAnterior)
{
    anterior = newAnterior;
}

void Nodo::setNumero(int newNumero)
{
    numero = newNumero;
}

```

```

int Nodo::getNumero(void)
{
    return numero;
}

● Pantalla.cpp
#include "Pantalla.h"
#include <Windows.h>
#include <iostream>
#include <cwchar>
using namespace std;

///////////////////////////////
// Name:      Pantalla::gotoxy(int x, int y)
// Purpose:   Implementation of Pantalla::gotoxy()
// Parameters:
// - x
// - y
// Return:    void
///////////////////////////////

void Pantalla::gotoxy(int x, int y)
{
    HANDLE h_con;
    h_con = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD dwPos;
    dwPos.X = x;
    dwPos.Y = y;
    SetConsoleCursorPosition(h_con, dwPos);
}

///////////////////////////////
// Name:      Pantalla::dimensionPantalla()
// Purpose:   Implementation of Pantalla::dimensionPantalla()
// Return:    void
///////////////////////////////

void Pantalla::dimensionPantalla(void)
{
    int ancho, alto;

    ancho = GetSystemMetrics(SM_CXSCREEN);
    alto = GetSystemMetrics(SM_CYSCREEN);

    cout << "Resolucion de pantalla > " << ancho << "x" << alto << endl;
}

void Pantalla::tablero(string nombre, int numero, int puntuacion)
{
    system("cls");
    const char horizontal = 205;
    const char vertical = 186;
    const char esquinaDerechaS = 187;
    const char esquinaDerechaI = 188;
    const char esquinaIzquierdaS = 201;

```

```

const char esquinaIzquierdaI = 200;
for (int i = 4; i < 40; i++)
{
    gotoxy(i, 4);
    cout << horizontal;
    gotoxy(i, 44);
    cout << horizontal;
}

for (int i = 5; i < 44; i++)
{
    gotoxy(4, i);
    cout << vertical;
    gotoxy(40, i);
    cout << vertical;
}
gotoxy(40, 4);
cout << esquinaDerechaS;
gotoxy(40, 44);
cout << esquinaDerechaI;
gotoxy(4, 4);
cout << esquinaIzquierdas;
gotoxy(4, 44);
cout << esquinaIzquierdaI;
cout << endl;
gotoxy(45, 7);
cout << "Jugador: " << nombre;
gotoxy(45, 10);
cout << "Siguiente numero: " << numero;
gotoxy(45, 13);
cout << "Puntuacion: " << puntuacion;
}

void Pantalla::ocultarCursor()
{
    HANDLE h_con;
    h_con = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_CURSOR_INFO cci;
    cci.dwSize = 28;
    cci.bVisible = FALSE;
    SetConsoleCursorInfo(h_con, &cci);
}

void Pantalla::borrarLinea()
{
    for (int i = 5; i < 40; i++)
    {
        gotoxy(i, 43);
        cout << " ";
    }
}

● Pantalla.h

*****
* Module:  Pantalla.h
* Author:  Alan
* Modified: lunes, 18 de noviembre de 2019 23:20:19
*****
```

```

* Purpose: Declaration of the class Pantalla
*****
#endif

#if !defined(_Tetris_2_Pantalla_h)
#define _Tetris_2_Pantalla_h
#include <iostream>

using namespace std;

class Pantalla
{
public:
    void gotoxy(int x, int y);
    void dimensionPantalla(void);
    void tablero(string nombre, int numero,int puntuacion);
    void ocultarCursor();
    void borrarLinea();
protected:
private:
    int ancho;
    int alto;
};


```

Ejecución del aplicativo

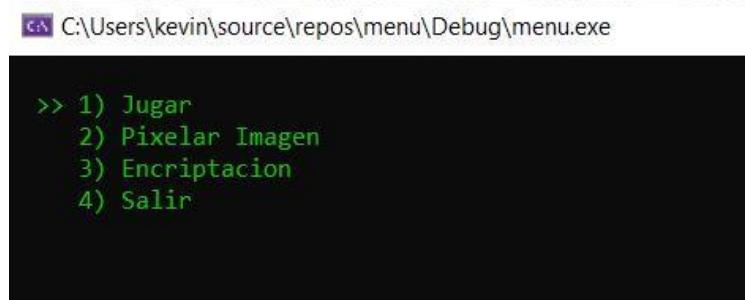


Figura 1 Se muestra el muestra el menú de inicio del juego

```
C:\Users\kevin\Desktop\ProyectoEstructuras\IndianaCrc  
Ingrese el nombre del jugador: kevin
```

Figura 2 Se ingresa el nombre del jugador

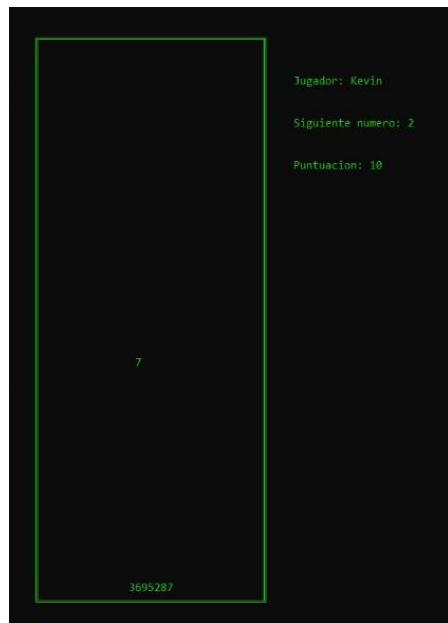
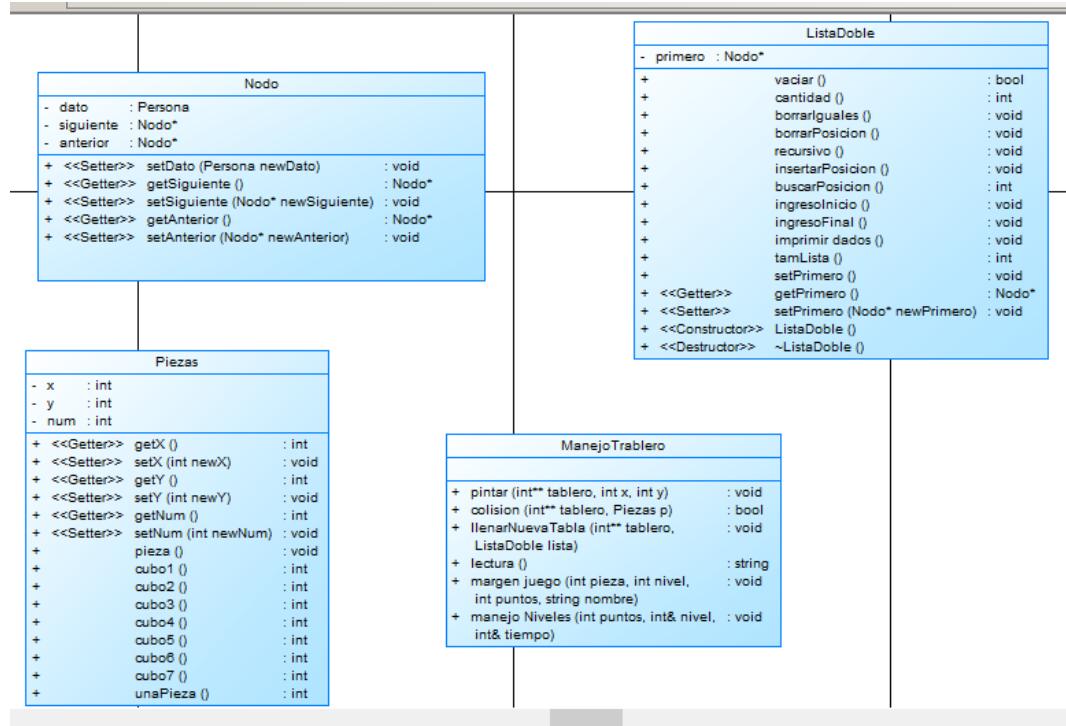


Figura 3 se Ve la interfaz del juego.

Grupo Toapanta–Naranjo

Modelado



Codigo

Codigo del aplicativo

```

*****
*          UNIVERSIDAD DE LAS FUERZAS ARMADAS  *
*                      ESPE                         *
*
*TRABAJO EN GRUPO:
*          NOMBRES:ANTONI TOAPANTA               *
*                      JONNY NARANJO                 *
*MATERIA: ESTRUCTURA DE DATOS                  *
*NRC:2967                                         *
*
*Fecha de Creacion:09/12/2019                   *
*****
#include "miniwin.h"
#include <iostream>
#include<stdlib.h>
#include "Piezas.h"
#include <windows.h>
#include "ManejoMemoria.h"
#include<time.h>
#include "ListaCircularDoble.h"
#include "ManejoTablero.h"
using namespace miniwin;
using namespace std;
int main()

```

```

{
    vredimensiona(900, 800);
    srand(time(NULL));
    int
numPieza,tic=0,t,**tablero,numPiezaS,puntos=0,nivel=1,tiempo=60;
    ListaCircularDoble lista;
    bool salir=true,bandera=true;
    Piezas p;
    tablero=reservar(15,30);
    encesar(tablero,15,30);
    //Inicio de mi lista
    *(*(tablero+13)+1)=3;
    lista.ingresoFinal(3);
    *(*(tablero+13)+2)=2;
    lista.ingresoFinal(2);
    *(*(tablero+13)+3)=1;
    lista.ingresoFinal(1);
    numPieza=1+rand()%(10-1); //Saca un numero aleatorio que da el
numero de mi pieza a ingresar
    //Fin de mi lista
    //Empezara de aqui mi funcion
    do//Por el momento se repite n veces no hay salida toca buscar
    {
        numPiezaS=1+rand()%(10-1);
        int x=10,y=0;
        bool bandera=false ;
        p.setX(x);//cambia la posicion de mi pieza
        p.setY(y);
        agregarPosicion(tablero,x,y,p.unaPieza(numPieza)); //coloca la
pieza en la parte de arriba con coordenadas x y
        t=tecla();

        cout<<endl;
        refresca();
        bandera=true;
        while(y!=13 && bandera==true)
        {
            //sale de mi bucle cuando bandera=pieza toca la otra pieza y
            y sea ==19= tamaño del tablero
            if(t==ESCAPE)
            {
                break;
            }
            if(tic>5) //funcion para que se baje solo las piesas
            {
                tic=0;
                t=ABAJO;
            }
            if(t==DERECHA) //se mueve hacia la derecha
            {
                if(x<=11) //no puede pasar el borde de la pantalla
                {
                    agregarPosicion(tablero,x,y,0);
                    x++;
                }
            }
            else if(t==IZQUIERDA) //izquierda

```

```

{
    if(x>1) // no pude pasar fuera de la pantalla
    {
        agregarPosicion(tablero,x,y,0);
        x--;
    }
}
else if(t==ABAJO) // se mueva para abajo
{
    agregarPosicion(tablero,x,y,0);
    y++;
    agregarPosicion(tablero,x,y,p.unaPieza(numPieza));
    lista.imprimirDatos();
    margenJuego(numPiezaS,nivel,puntos,lectura());
    escribirArchivo(tablero,14,13);
}
if(t!=NINGUNA)//si no resive ninguna tecla
{
    borra(); //borra la anterior para dar una simulacion de
    movimiento
    p.setX(x); //cambio la posiciones
    p.setY(y);
    if(colision(tablero,p)) //comprueba si existe una pieza
    abajo de ella
    {
        margenJuego(numPiezaS,nivel,puntos,lectura());
        pintar(tablero,14,13);
        p.unaPieza(numPieza);
        refresca();
        bandera=false;
    }
    else
    {
        margenJuego(numPiezaS,nivel,puntos,lectura());
        pintar(tablero,14,13);
        p.unaPieza(numPieza);
        lista.imprimirDatos();
        cout<<endl;
        refresca();
    }
}
espera(tiempo);
tic++;
t=tecla();
}
lista.insertarPosicion(numPieza,x);
lista.borrarIguales(puntos);
encerar(tablero,15,30);
llenarNuevaTabla(tablero,lista);
system("cls");
lista.imprimirDatos();
numPieza=numPiezas;
manejoNiveles(puntos,nivel,tiempo);
}
while(t!=ESCAPE);
//terminara aqui

```

```

        system("txt2pdf.exe MovimientoTetris.csv Movimiento.pdf -oao -
pfs60 -pps43 -ptc0 -width3000 -height2000");
        archivoJuego(lectura(),nivel,puntos);
        liberarMemoria(tablero,16);
        mensaje("FIN DEL JUEGO :D\nESTE ES TU LISTA :3");
        vcierra();
        system("cls");
        lista.imprimirDatos();
        refresca();
        return 0;
    }
Código de las clases:
//****************************************************************************
*          UNIVERSIDAD DE LAS FUERZAS ARMADAS
*                      ESPE
*TRABAJO EN GRUPO:
*NOMBRES:ANTONI TOAPANTA
*                JONNY NARANJO
*MATERIA: ESTRUCTURA DE DATOS
*NRC:2967
*Fecha de Creacion:09/12/2019
***** */

#if !defined(__Class_Diagram_1_ListaSimple_h)
#define __Class_Diagram_1_ListaSimple_h

#include <iostream>
#include "Nodo.h"

using namespace std;

class ListaCircularDoble
{
public:
    bool vacia();
    int cantidad();
    void borrarIguales(int&puntos);
    void borrarPosicion(int posicion);
    void recursivo(Nodo *p, int posicion, bool bandera,int&puntos);
    void insertarPosicion(int obj, int pos);
    int buscarPosicion(int posicion);
    void ingresoInicio(int obj);
    void ingresoFinal(int obj);
    void imprimirDatos();
    int tamLista();
    ListaCircularDoble();
    void destruir();
    void setPrimero(Nodo *newPrimero);
    Nodo* getPrimero();
protected:
private:
    Nodo *primero;
};

#endif

*****

```

```

*
*           UNIVERSIDAD DE LAS FUERZAS ARMADAS
*
*           ESPE
*
*TRABAJO EN GRUPO:
*           NOMBRES:ANTONI TOAPANTA
*
*           JONNY NARANJO
*
*MATERIA: ESTRUCTURA DE DATOS
*
*NRC:2967
*
*Fecha de Creacion:09/12/2019
*****
*/



#if !defined(__Class_Diagram_1_ListaSimple_h)
#define __Class_Diagram_1_ListaSimple_h

#include <iostream>
#include "Nodo.h"

using namespace std;

class ListaCircularDoble
{
public:
    bool vacia();
    int cantidad();
    void borrarIguales(int&puntos);
    void borrarPosicion(int posicion);
    void recursivo(Nodo *p, int posicion, bool bandera,int&puntos);
    void insertarPosicion(int obj, int pos);
    int buscarPosicion(int posicion);
    void ingresoInicio(int obj);
    void ingresoFinal(int obj);
    void imprimirDatos();
    int tamLista();
    ListaCircularDoble();
    void destruir();
    void setPrimero(Nodo *newPrimero);
    Nodo* getPrimero();
protected:
private:
    Nodo *primero;
};

#endif

/*
*
*           UNIVERSIDAD DE LAS FUERZAS ARMADAS
*
*           ESPE
*
*TRABAJO EN GRUPO:
*           NOMBRES:ANTONI TOAPANTA
*
*           JONNY NARANJO
*
*MATERIA: ESTRUCTURA DE DATOS
*
*NRC:2967
*
*Fecha de Creacion:09/12/2019
*****
*/



#if !defined(__Class_Diagram_1_Nodo_h)
#define __Class_Diagram_1_Nodo_h

```

```

class Nodo
{
public:
    int getDatos(void);
    void setDatos(int newDatos);
    Nodo *getSiguiente(void);
    void setSiguiente(Nodo* newSiguiente);
    Nodo *getAnterior(void);
    void setAnterior(Nodo* newAnterior);
    Nodo();
protected:
private:
    int datos;
    Nodo *siguiente;
    Nodo *anterior;
};

#endif

/*****************
*      UNIVERSIDAD DE LAS FUERZAS ARMADAS      *
*          ESPE                                     *
*TRABAJO EN GRUPO:                                *
*NOMBRES:ANTONI TOAPANTA                           *
*JONNY NARANJO                                     *
*MATERIA: ESTRUCTURA DE DATOS                      *
*NRC:2967                                         *
*Fecha de Creacion:09/12/2019                      *
*****************/
#include "Nodo.h"
#include <stdlib.h>
///////////////////////////////
// Name:    Nodo::getDatos()
// Purpose: Implementation of Nodo::getDatos()
// Return:  Persona
///////////////////////////////

int Nodo::getDatos(void)
{
    return datos;
}

///////////////////////////////
// Name:    Nodo::setDatos(Persona newDatos)
// Purpose: Implementation of Nodo::setDatos()
// Parameters:
// - newDatos
// Return:  void
///////////////////////////////

void Nodo::setDatos(int newDatos)
{
    datos = newDatos;
}

```

```

////////// Nodo.cpp //////////

// Name:    Nodo::getSiguiente()
// Purpose: Implementation of Nodo::getSiguiente()
// Return:   Nodo *
////////// Nodo.cpp //////////

Nodo * Nodo::getSiguiente(void)
{
    return siguiente;
}

////////// Nodo.cpp //////////

// Name:    Nodo::setSiguiente(Nodo* newSiguiente)
// Purpose: Implementation of Nodo::setSiguiente()
// Parameters:
// - newSiguiente
// Return:  void
////////// Nodo.cpp //////////

void Nodo::setSiguiente(Nodo* newSiguiente)
{
    siguiente = newSiguiente;
}

////////// Nodo.cpp //////////

// Name:    Nodo::setSiguiente(Nodo* newSiguiente)
// Purpose: Implementation of Nodo::setSiguiente()
// Parameters:
// - newSiguiente
// Return:  void
////////// Nodo.cpp //////////

Nodo::Nodo()
{
    datos = NULL;
    siguiente = NULL;
    anterior = NULL;
}
////////// Nodo.cpp //////////

// Name:    Nodo::setSiguiente(Nodo* newSiguiente)
// Purpose: Implementation of Nodo::setSiguiente()
// Parameters:
// - newSiguiente
// Return:  void
////////// Nodo.cpp //////////

void Nodo::setAnterior(Nodo* newAnterior)
{
    anterior = newAnterior;
}
////////// Nodo.cpp //////////

// Name:    Nodo::setSiguiente(Nodo* newSiguiente)
// Purpose: Implementation of Nodo::setSiguiente()
// Parameters:
// - newSiguiente
// Return:  void
////////// Nodo.cpp //////////

Nodo * Nodo::getAnterior(void)
{

```

```

        return anterior;
    }

/*********************************************
*          UNIVERSIDAD DE LAS FUERZAS ARMADAS      *
*          ESPE                                     *
*TRABAJO EN GRUPO:                           *
*          NOMBRES:ANTONI TOAPANTA               *
*          JONNY NARANJO                         *
*MATERIA: ESTRUCTURA DE DATOS                *
*NRC:2967                                     *
*Fecha de Creacion:09/12/2019                 *
********************************************/


#if !defined(__Class_Diagram_1_Piezas_h)
#define __Class_Diagram_1_Piezas_h

class Piezas
{
public:
    int getX(void);
    void setX(int newX);
    int getY(void);
    void setY(int newY);
    void pieza(int x, int y, int col);
    int unaPieza(int pieza);
    int getNum(void);
    void setNum(int n);
    int cubo1(void);
    int cubo2(void);
    int cubo3(void);
    int cubo4(void);
    int cubo5(void);
    int cubo6(void);
    int cubo7(void);
    int cubo8(void);
    int cubo9(void);
    int vacio(void);
protected:
private:
    int x;
    int y;
    int num;
};

#endif

/*********************************************
*          UNIVERSIDAD DE LAS FUERZAS ARMADAS      *
*          ESPE                                     *
*TRABAJO EN GRUPO:                           *
*          NOMBRES:ANTONI TOAPANTA               *
*          JONNY NARANJO                         *
*MATERIA: ESTRUCTURA DE DATOS                *
*NRC:2967                                     *
*Fecha de Creacion:09/12/2019                 *
********************************************/

```

```

#include "Piezas.h"
#include "miniwin.h"
using namespace miniwin;
////////////////////////////////////////////////////////////////
// Name: Piezas::getX()
// Purpose: Implementation of Piezas::getX()
// Return: int
////////////////////////////////////////////////////////////////
void Piezas::pieza(int x, int y, int col)
{
    color(col);
    rectangulo_lleno(1+x*50, 1+y*50, x*50+50, y*50+50);
}
int Piezas::getX(void)
{
    return x;
}

////////////////////////////////////////////////////////////////
// Name: Piezas::setX(int newX)
// Purpose: Implementation of Piezas::setX()
// Parameters:
// - newX
// Return: void
////////////////////////////////////////////////////////////////

void Piezas::setX(int newX)
{
    x = newX;
}

////////////////////////////////////////////////////////////////
// Name: Piezas::getY()
// Purpose: Implementation of Piezas::getY()
// Return: int
////////////////////////////////////////////////////////////////

int Piezas::getY(void)
{
    return y;
}

////////////////////////////////////////////////////////////////
// Name: Piezas::setY(int newY)
// Purpose: Implementation of Piezas::setY()
// Parameters:
// - newY
// Return: void
////////////////////////////////////////////////////////////////

void Piezas::setY(int newY)
{
    y = newY;
}

```

```

// Name:      Piezas::cubo1()
// Purpose:   Implementation of Piezas::cubo1()
// Return:    int
/////////////////////////////// /////////////////////////////////
int Piezas::cubo1(void)
{
    pieza(getX(),getY(),BLANCO);
    return 1;
}

/////////////////////////////// /////////////////////////////////
// Name:      Piezas::cubo2()
// Purpose:   Implementation of Piezas::cubo2()
// Return:    int
/////////////////////////////// /////////////////////////////////

int Piezas::cubo2(void)
{
    pieza(getX(),getY(),ROJO);
    return 2;
}

/////////////////////////////// /////////////////////////////////
// Name:      Piezas::cubo3()
// Purpose:   Implementation of Piezas::cubo3()
// Return:    int
/////////////////////////////// /////////////////////////////////

int Piezas::cubo3(void)
{
    pieza(getX(),getY(),AZUL);
    return 3;
}

/////////////////////////////// /////////////////////////////////
// Name:      Piezas::cubo4()
// Purpose:   Implementation of Piezas::cubo4()
// Return:    int
/////////////////////////////// /////////////////////////////////

int Piezas::cubo4(void)
{
    pieza(getX(),getY(),AMARILLO);
    return 4;
}

/////////////////////////////// /////////////////////////////////
// Name:      Piezas::cubo5()
// Purpose:   Implementation of Piezas::cubo5()
// Return:    int
/////////////////////////////// /////////////////////////////////

int Piezas::cubo5(void)
{
    pieza(getX(),getY(),VERDE);
    return 5;
}

```

```

}

///////////////////////////////
// Name: Piezas::cubo6()
// Purpose: Implementation of Piezas::cubo6()
// Return: int
///////////////////////////////

int Piezas::cubo6(void)
{
    pieza(getX(), getY(), MAGENTA);
    return 6;
}

///////////////////////////////
// Name: Piezas::cubo7()
// Purpose: Implementation of Piezas::cubo7()
// Return: int
///////////////////////////////

int Piezas::cubo7(void)
{
    pieza(getX(), getY(), NEWCOLOR);
    return 7;
}

///////////////////////////////
// Name: Piezas::cubo8()
// Purpose: Implementation of Piezas::cubo8()
// Return: int
///////////////////////////////

int Piezas::cubo8(void)
{
    pieza(getX(), getY(), VERDECLARO);
    return 8;
}

///////////////////////////////
// Name: Piezas::cubo9()
// Purpose: Implementation of Piezas::cubo9()
// Return: int
///////////////////////////////
/

int Piezas::cubo9(void)
{
    pieza(getX(), getY(), CELESTE);
    return 9;
}

///////////////////////////////
// Name: Piezas::cubo9()
// Purpose: Implementation of Piezas::cubo9()
// Return: int
///////////////////////////////
/

int Piezas::vacio(void)

```

```

{
    pieza(getX(),getY(),NEGRO);
    return 0;
}
/////////////////////////////////////////////////////////////////
// Name:      Piezas::cubo9()
// Purpose: Implementation of Piezas::cubo9()
// Return:   int
/////////////////////////////////////////////////////////////////
int Piezas::getNum(void)
{
    return num;
}
/////////////////////////////////////////////////////////////////
// Name:      Piezas::cubo9()
// Purpose: Implementation of Piezas::cubo9()
// Return:   int
/////////////////////////////////////////////////////////////////
void Piezas::setNum(int n)
{
    num=n;
}
/////////////////////////////////////////////////////////////////
// Name:      Piezas::cubo9()
// Purpose: Implementation of Piezas::cubo9()
// Return:   int
/////////////////////////////////////////////////////////////////
int Piezas::unaPieza(int pieza)
{
    switch(pieza)
    {
        case 1:
            setNum(1);
            return cubo1();
            break;
        case 2:
            setNum(2);
            return cubo2();
            break;
        case 3:
            setNum(3);
            return cubo3();
            break;
        case 4:
            setNum(4);
            return cubo4();
            break;
        case 5:
            setNum(5);
            return cubo5();
            break;
        case 6:
            setNum(6);
            return cubo6();
    }
}

```

```

        break;
    case 7:
        setNum(7);
        return cubo7();
        break;
    case 8:
        setNum(8);
        return cubo8();
        break;
    case 9:
        setNum(9);
        return cubo9();
        break;
    case 0:
        setNum(0);
        return vacio();
        break;
    }
}

/*********************************************
*          UNIVERSIDAD DE LAS FUERZAS ARMADAS      *
*                      ESPE                         *
*TRABAJO EN GRUPO:                          *
*NOMBRES:ANTONI TOAPANTA                   *
*JONNY NARANJO                            *
*MATERIA: ESTRUCTURA DE DATOS            *
*NRC:2967                                  *
*Fecha de Creacion:09/12/2019             *
********************************************/


#ifndef MANEJOTABLERO_H_INCLUDED
#define MANEJOTABLERO_H_INCLUDED
#include "miniwin.h"
#include "Piezas.h"
#include<time.h>
#include <string>
#include <fstream>
#include <sstream>
using namespace miniwin;
void pintar(int **tablero,int x, int y)
{
    Piezas p;
    for(int i=0;i<x;i++)
    {
        for(int j=0;j<y;j++)
        {
            p.setX(j);
            p.setY(i);
            p.unaPieza(*(*(tablero+i)+j));
        }
    }
}
bool colision(int **tabla,Piezas p)
{
    if(p.getY()<19)
    {

```

```

        p.setY(p.getY() + 1);
    }
    p.unaPieza(*(*(tabla + p.getY()) + p.getX())));
    if (p.getNum() != 0) {
        return true;
    }
    else{
        return false;
    }
}
void llenarNuevaTabla(int **tabla, ListaCircularDoble lista)
{
    for(int i=0;i<30;i++)
    {
        *(*(tabla+13)+i)=lista.buscarPosicion(i);
    }
}
string lectura()
{
    ifstream archivo;
    string texto;
    archivo.open("Nombre.txt",ios::in);
    if(archivo.fail())
    {
        exit(1);
    }
    else{
        while(!archivo.eof())
        {
            getline(archivo, texto);
        }
    }
    archivo.close();
    return texto;
}
void margenJuego( int pieza, int nivel, int puntos, string nombre)
{
    Piezas p;
    stringstream out,out2;
    out<<puntos;
    out2<<nivel;
    color(AZUL);
    linea(50,0,50,700);
    color(AZUL);
    linea(650,0,650,700);
    color(AZUL);
    linea(50,700,650,700);
    color(BLANCO);
    texto(700,100,"Pieza Siguiente");
    p.setX(15);
    p.setY(3);
    p.unaPieza(pieza);
    color(BLANCO);
    texto(700,250,"Nombre del Jugador:");
    texto(760,300,nombre);
    texto(700,350,"Nivel");
    texto(760,400,out2.str());
}

```

```

        texto(700,450,"Puntos:");
        texto(760,500,out.str());
    }
void manejoNiveles(int puntos,int &nivel,int&tiempo)
{
    if(puntos==100)
    {
        nivel++;
        tiempo=30;
    }
    else if(puntos==200)
    {
        nivel++;
        tiempo=10;
    }
    else if(puntos==300)
    {
        nivel++;
        tiempo=5;
    }
}
#endif // MANEJOTABLERO_H_INCLUDED

```

Menu:

```

#include <iostream>
#include "PersonalLibrary.h"
#include "ingreso.h"
#include <pthread.h>
using namespace std;

```

```

void* moverMarquesina(void *marq){

    int a, b, c, n=0,letra=39,pos=1,cont=39,aux,cont1=39,auxg;
    char t[50] ="GRACIAS POR JUGAR",auxt[39]=" ";

    do{
        for (a=0;a<42;a++){
            aux=pos;
            for(b=39;b>cont;b--){
                gotoxy(pos,1);
                cout<<t[b];
                pos--;
            }
            aux++;
            pos=aux;
            cont--;
            Sleep (75);
            if(a==40){
                break;
            }
        }
        for(a=3;a<70;a++){
            gotoxy(a-1,1);
            cout<<" ";
            gotoxy(a,1);
        }
    }
}
```

```

        cout<<t;
        Sleep (75);
    }

    pos=70;
    auxg=69;
    for (a=0;a<41;a++) {
        gotoxy(auxg,1);
        cout<<" ";
        aux=pos;
        for(b=0;b<=cont1;b++) {
            gotoxy(pos,1);
            cout<<t[b];
            pos++;
        }
        cont1--;
        aux++;
        pos=aux;
        auxg++;
        Sleep (75);
    }
    cont1=39;
    letra=39;
    pos=1;
    cont=39;
    c=0;
}while (c==1);

}

int main()
{
    AltEnter();
    const char *opciones[]={ "1) Jugar", "2) Mostrar Imagen", "3) Mostrar
Ayuda", "4) Codificar", "5) Decodificar", "6) Salir"};
    bool bandera = true;
    string nombre,datos=lectura();
    Ingreso leer;
    do{
        int opcion = menu("Bienvenido",opciones,6);
        switch(opcion){
            case 1:
                system("cls");
                nombre = leer.ingresarString("Ingrese nombre del
jugador: ");
                guardarNombre(nombre);
                system("cls");
                system("Tetris.exe");
                system("pause");
            break;
            case 2:
                system("cls");
                system("imagenExtras.exe");
                system("pause");
            break;
        }
    }
}

```

```

        case 3:
        system("AyudaTetris.chm");
        break;
        case 4:
        system("cls");
        codificar(datos,0);
        archivoC(datos);
        system("pause");
        break;
        case 5:
        system("cls");
        decodificar(datos,0);
        archivoD(datos);
        system("pause");
        break;
        case 6:
        system("cls");
        pthread_create(&thread1,NULL,moverMarquesina,NULL);
        system("pause");
        bandera = false;
        break;
    }
} while(bandera);
return 0;
}
}

```

Ejecución del aplicativo



Fig1)Inicio del aplicativo

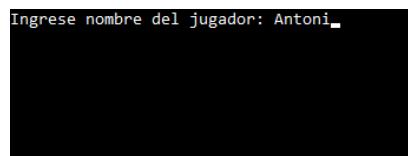


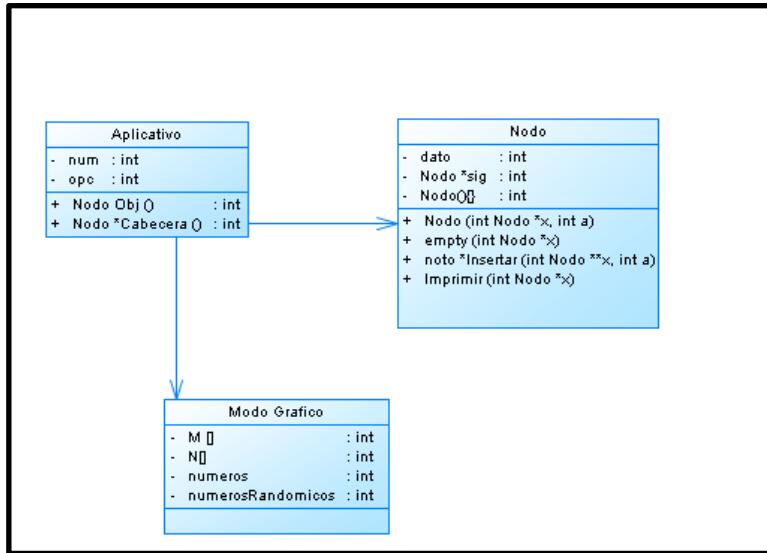
Fig2)Ingreso del nombre del Jugador



Fig3) Inicio del Juego

Grupo Baez_Cardenas

Modelado



Código

Aplicativo.cpp

```

        std::cin>>opc;
        switch(opc)
        {
            case 1:
                std::cout<<"Ingrese el dato:
";
                std::cin>>num;

                Cabecera=Obj.Insertar(&Cabecera,num);
                Obj.Imprimir(Cabecera);
                system("pause");
                break;
            case 2:
                if(Obj.empty(Cabecera))
                {
                    std::cout<<"La
lista esta vacia\n";
                }
                Obj.Imprimir(Cabecera);
                break;
            case 3:
                std::cout<<"La
lista no esta vacia y es: \n";
                Obj.Imprimir(Cabecera);
                system("pause");
                break;
            case 4:
                printf("\nLos datos ingresados
");
                Obj.Imprimir(Cabecera);
                system("pause");
                break;
        }
    }
}

```

ClaseNodo.h

```

#include<iostream>
#include<stdlib.h>
#include<conio.h>
#include<iomanip>
using namespace std;
class Nodo
{
public:
    int dato;
    Nodo *sig;
public:
    Nodo();
    Nodo(Nodo *x,int a);
    bool empty(Nodo *x);

```

```

        Nodo *Insertar(Nodo **x,int a);
        void Imprimir(Nodo *x);
    };
Nodo::Nodo(Nodo *x,int a)
{
    sig=x;
    dato=a;
}
bool Nodo::empty(Nodo *x){
    return x==NULL;
}
Nodo *Nodo::Insertar(Nodo **x,int a)
{
    Nodo *nuevo=new Nodo();
    nuevo->dato=a;
    if(*x == NULL)
        *x = nuevo;
    else{
        nuevo->sig = (*x)->sig;
    }
    (*x)->sig = nuevo;// cerramos la lista circular
    return nuevo;
}

void Nodo::Imprimir(Nodo *x)
{
    Nodo *nuevo=new Nodo();
    nuevo = x;
    do {
        printf("%d -> ", nuevo->dato);
        nuevo = nuevo->sig;
    } while(nuevo != x);
    printf("\n");
}

```

ModoGrafico.cpp

```

#include <SFML/Graphics.hpp>
#include <time.h>
#include <iostream>

using namespace std;
using namespace sf;

const int M = 20;
const int N = 10;

int field[M][N] = { 0 };

struct Point
{
    int x, y;
} a[4], b[4];

int figures[7][4] =
{

```

```

    1,1,1,1, // O
};

int numeros, numerosRandomicos;

bool check()
{
    for (int i = 0; i < 4; i++)
        if (a[i].x < 0 || a[i].x >= N || a[i].y >=
M) return 0;
        else if (field[a[i].y][a[i].x]) return 0;

    return 1;
}

int main()
{
    srand(time(0));

    RenderWindow window(VideoMode(320, 480), "Tetris!");

    Texture t1, t2, t3;
    t1.loadFromFile("images/tiles.png");
    t2.loadFromFile("images/background.png");
    t3.loadFromFile("images/frame.png");

    Sprite s(t1), background(t2), frame(t3);

    int dx = 0; bool rotate = 0; int colorNum = 1;
    float timer = 0, delay = 0.3;

    Clock clock;

    while (window.isOpen())
    {
        float time =
clock.getElapsedTime().asSeconds();
        clock.restart();
        timer += time;

        Event e;
        while (window.pollEvent(e))
        {
            if (e.type == Event::Closed)
                window.close();

            if (e.type ==
Event::KeyPressed)
                if (e.key.code ==
Keyboard::Up) rotate = true;
                else if
(e.key.code == Keyboard::Left) dx = -1;

```

```

        else if
(e.key.code == Keyboard::Right) dx = 1;
}

if
(Keyboard::isKeyPressed(Keyboard::Down)) delay = 0.05;

//// <- Move -> ///
for (int i = 0; i < 4; i++){
    b[i] = a[i];
    a[i].x += dx;
}
if (!check())
    for (int i = 0; i < 4; i++) {
        a[i] = b[i];
    }
/////////Rotate////////
if (rotate)
{
    Point p = a[1]; //center of
rotation
    for (int i = 0; i < 4; i++)
    {
        int x = a[i].y -
p.y;
        int y = a[i].x -
p.x;
        a[i].x = p.x - x;
        a[i].y = p.y + y;
    }
    if (!check()) for (int i = 0;
i < 4; i++) a[i] = b[i];
}

/////////Tick////////
if (timer > delay)
{
    for (int i = 0; i < 4; i++) {
        if (!check())
        {
            for (int i = 0; i
< 4; i++) field[b[i].y][b[i].x] = colorNum;
            rand() % 7;
            int n = rand() %
7;
            for (int i = 0; i
< 4; i++)
            {
                a[i].x
= figures[n][i] % 2;
                a[i].y
= figures[n][i] / 2;
            }
        }
    }
}

```

```

        }

        timer = 0;
    }

/////////check lines/////////
int k = M - 1;
for (int i = M - 1; i > 0; i--)
{
    for (int j = 0; j < N; j++)
    {
        if (field[k+1][j]
== field[i][j]) {

            field[k][j] = NULL;
            field[i][j] = NULL;
        }
        field[k][j] =
field[i][j];
    }
}

dx = 0; rotate = 0; delay = 0.3;

/////////draw/////////
window.clear(Color::White);
window.draw(background);

for (int i = 0; i < M; i++)
    for (int j = 0; j < N; j++)
    {
        if (field[i][j] ==
0) continue;

        s.setTextureRect(IntRect(field[i][j] * 18, 0, 18, 18));
        s.setPosition(j *
18, i * 18);
        s.move(28, 31);
        //offset
        window.draw(s);
    }

    for (int i = 0; i < 4; i++)
    {

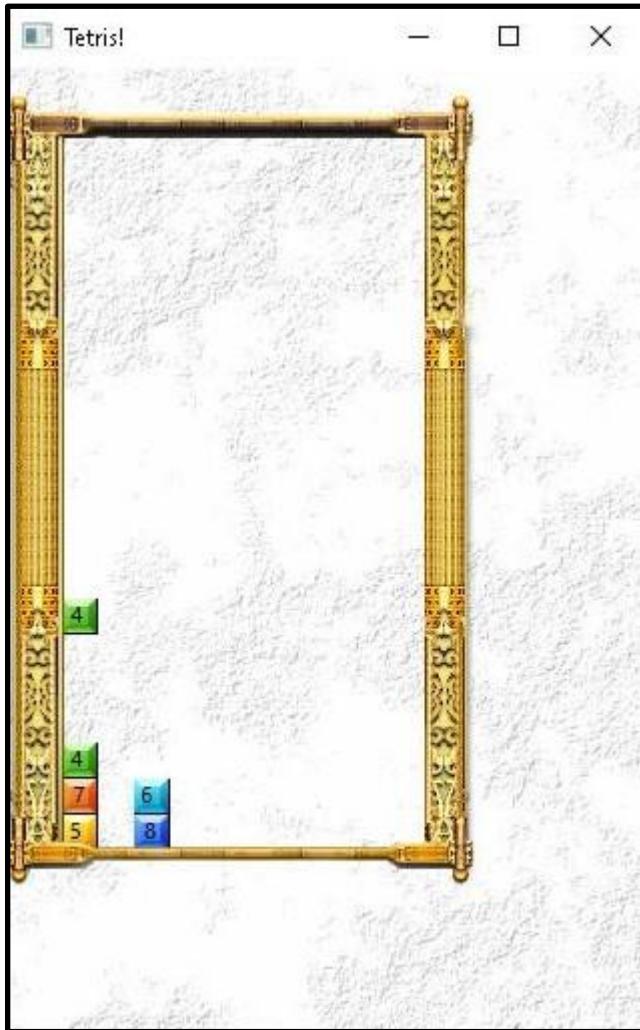
        s.setTextureRect(IntRect(colorNum * 18, 0, 18, 18));
        s.setPosition(a[i].x * 18,
a[i].y * 18);
        s.move(28, 31); //offset
        window.draw(s);
    }
}

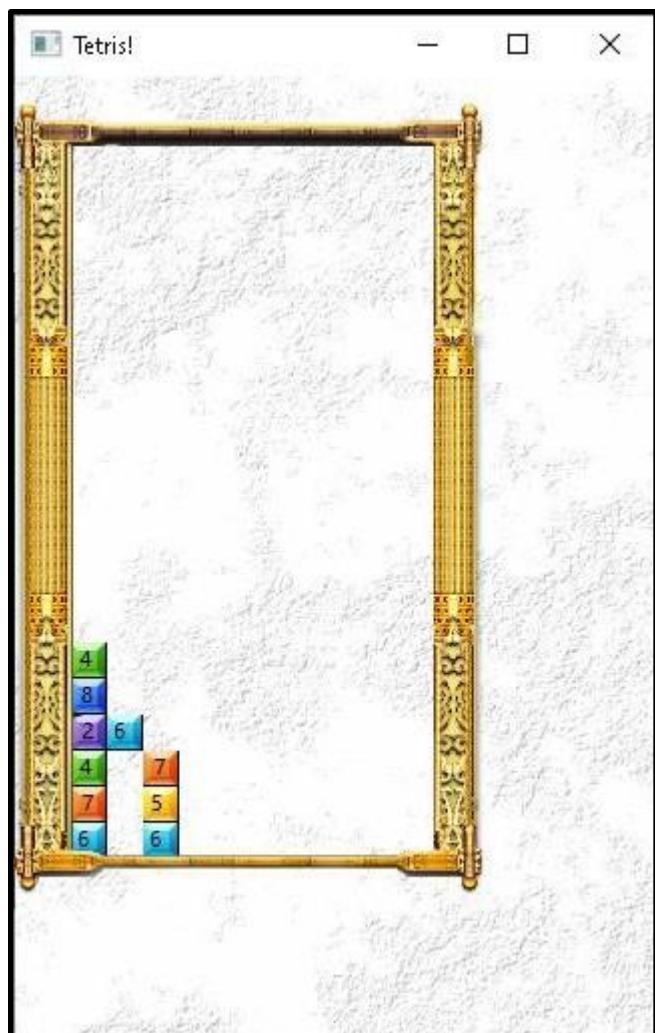
```

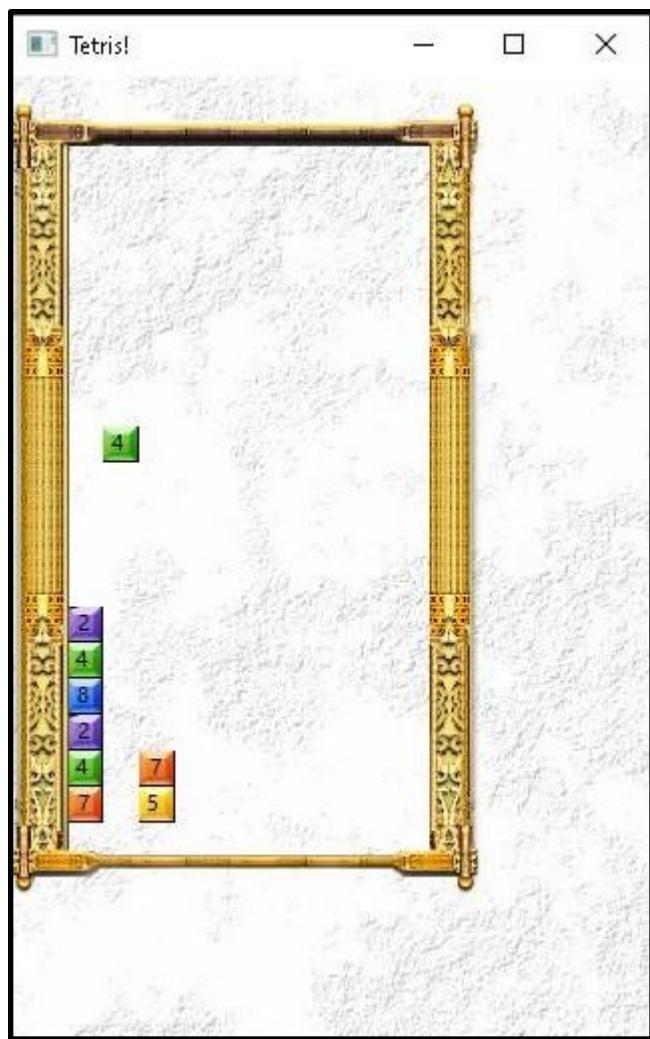
```
        window.draw(frame);
        window.display();
    }

    return 0;
}
```

Ejecución:

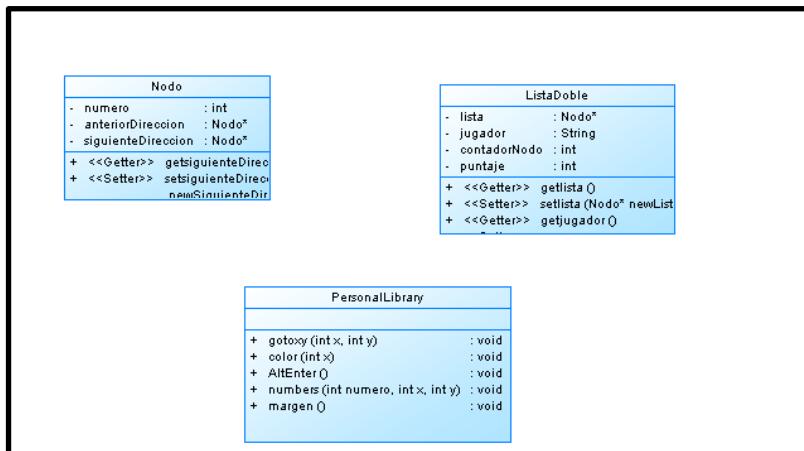






Grupo Carvajal_Llorente

Modelado:



Codigo

ListaDoble.cpp

```
/*
    Universidad de las Fuerzas Armadas ESPE
    Nombre: Luis Carvajal, Elian Llorente
    NRC: 2967
    Fecha creacion: 30/11/2019
    Fecha ultima modificacion: 12/12/2019
    Docente: Ing. Fernando Solis
*/
#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 );
}
```



```

        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=list;
    int cont;

    while(Nuevo!=NULL)
    {
        if(numero==Nuevo->getNumero())
        {
            cont++;
            break;
        }
        Nuevo=Nuevo->getSigienteDireccion();
    }

    switch(posicion)
    {
        case 1:
            if(cont>0)
            {
                Nuevo=Nuevo->getSigienteDireccion();
                Nuevo->setAnteriorDireccion(NULL);
                lista=Nuevo;
                contadorNodo--;
            }
            else
            {

                Aux->setNumero(numero);
                Aux->setSigienteDireccion(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 eliminara
 * @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 coinciden
 * @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=list;
    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->getSigienteDireccion();
    }
    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=list;

```

```

    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;

```

```

                break;
            }
            x+=4; //separacion entre cubos
            Aux=Aux->getSiguienteDireccion();
        }
    }

Nodo.h
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;
}
}

PersonalLibrary.h

#include <stdio.h>
#include <iostream>
#include <stdlib.h>

```

```

#include <conio.h>
#include <windows.h>
#include <cctype.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
            gotoxy(x,y++);

celeste

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
            gotoxy(x,y++);

claro

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);

}

```

Main.cpp

```

#include <iostream>
#include <windows.h>
#include <cconio.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 seleccionar 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 )
    {

```



```

        tecla = getch();
    else
        tecla = ' ';
    menuinicio(tecla);

    Sleep(100);
}

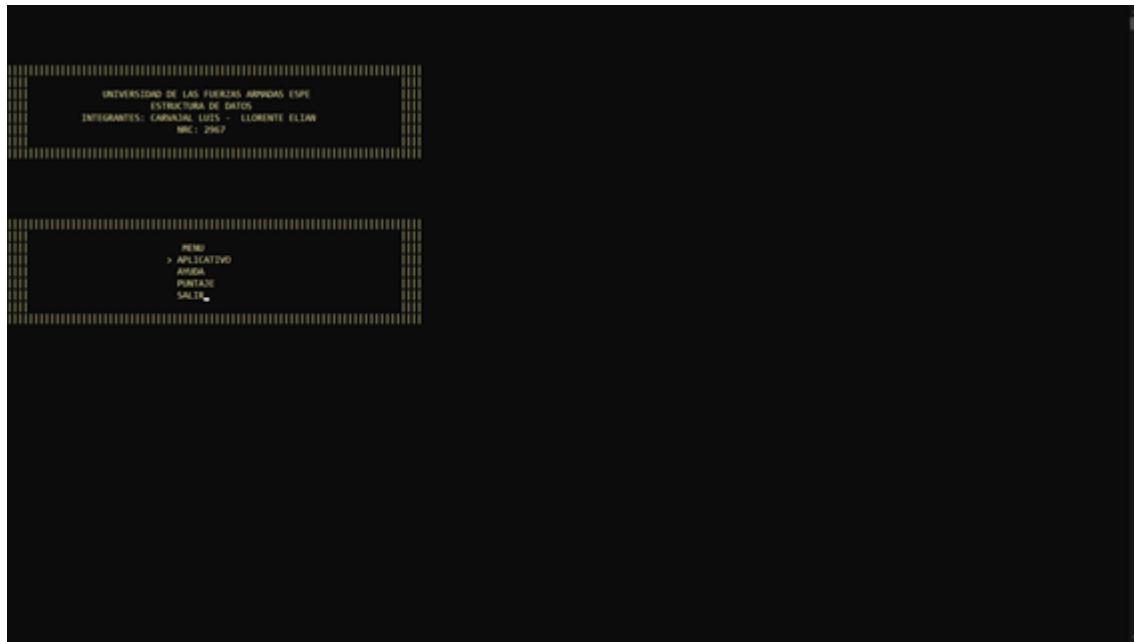
system("pause>null");

return 0;
}
}

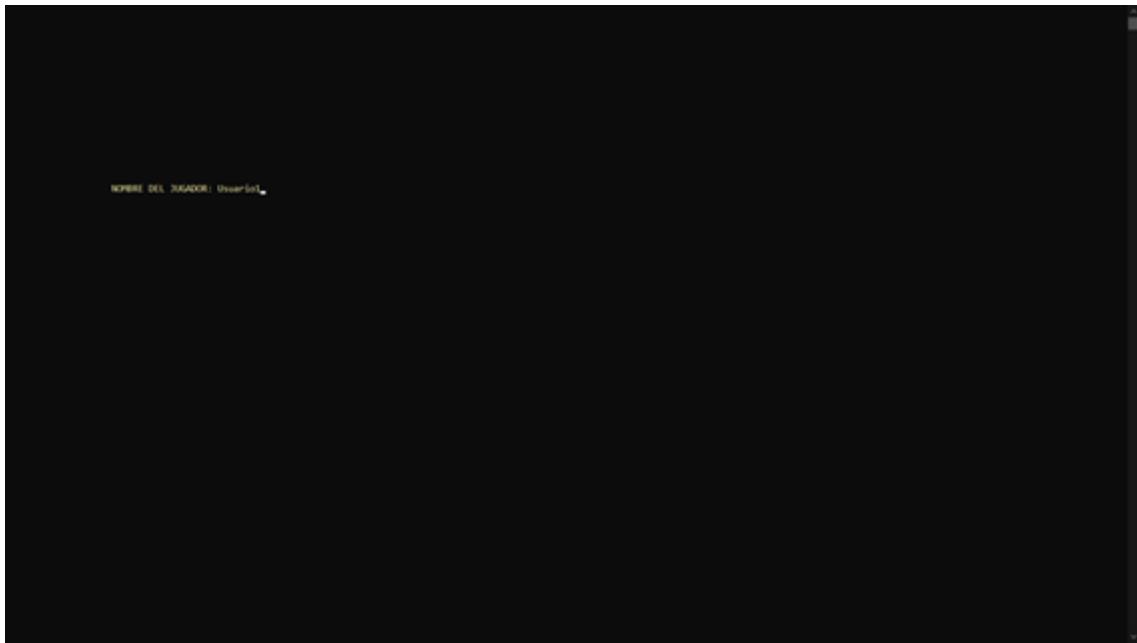
```

Ejecución:

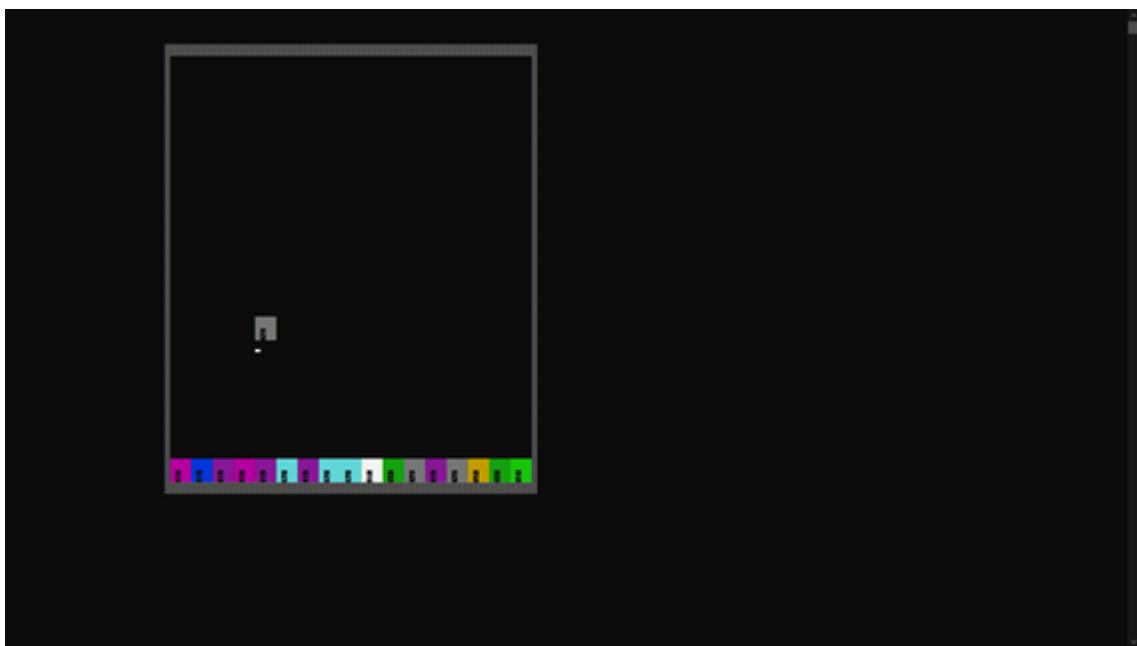
Menú Principal:



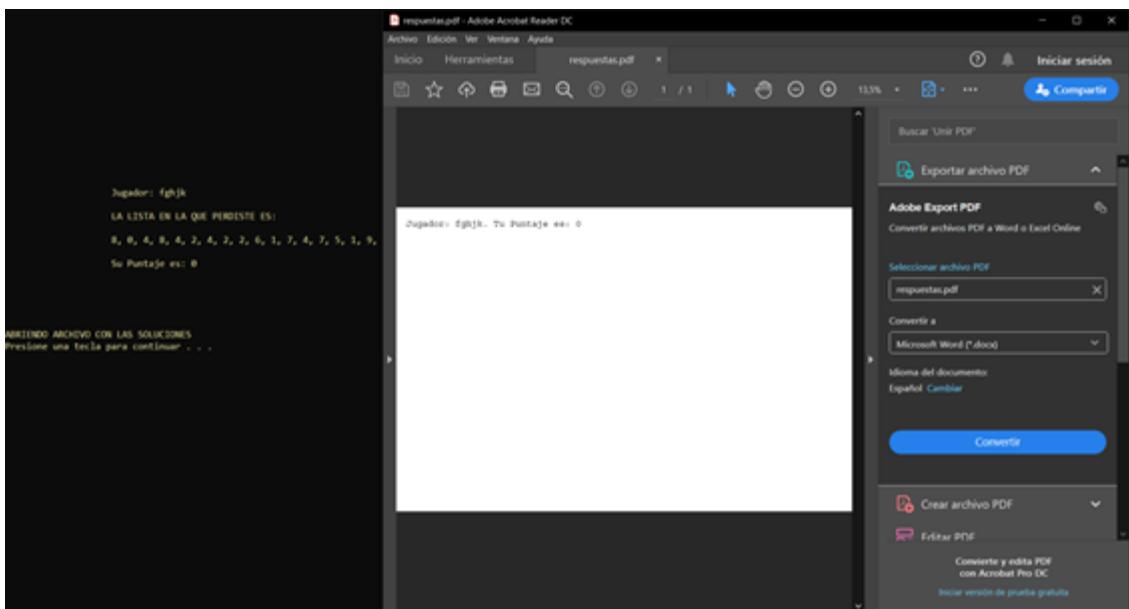
Ingreso del nombre de jugador:



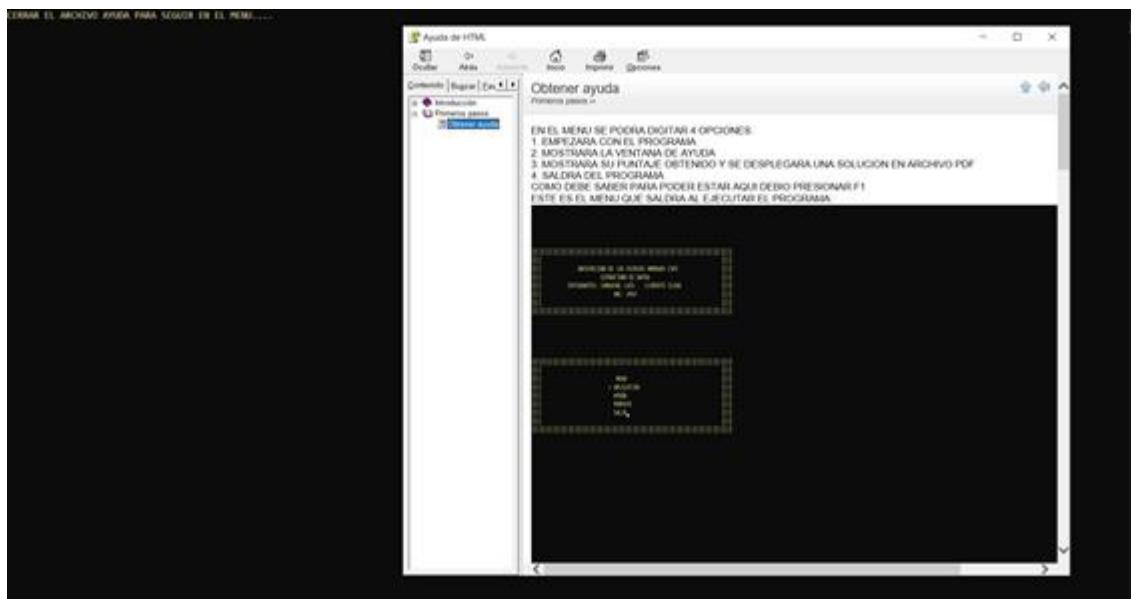
Juego:



Resultados:

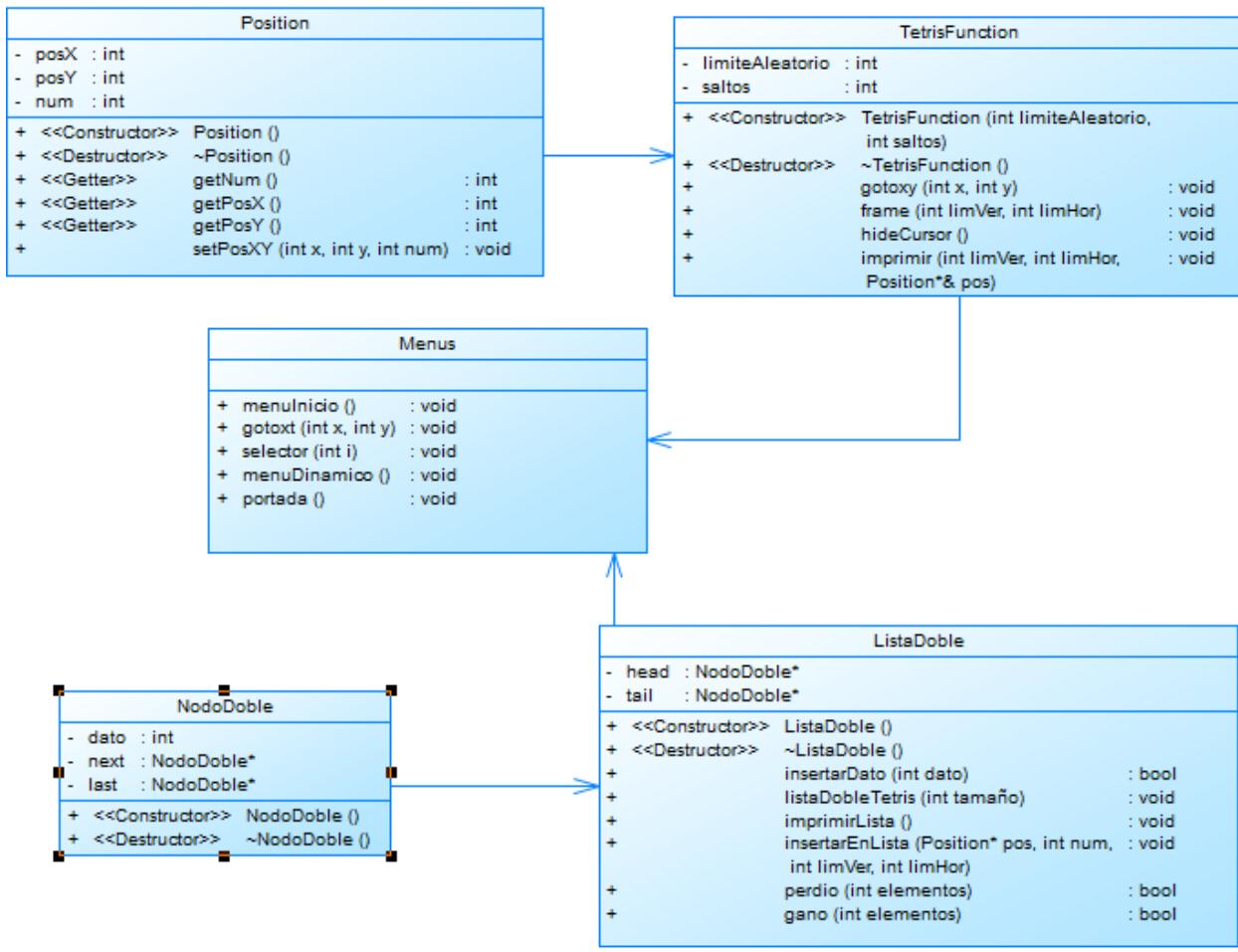


Ayuda al usuario:



Grupo Avila_Zurita

Modelado



Codigo

```

#include "NodoDoble.h"
#include <iostream>
#include <math.h>
#include <fstream>
#include "Position.h"
class ListaDoble
{
private:
    NodoDoble* head;
    NodoDoble* tail;
public:
    //Constructor
    ListaDoble();
    //Destructor
    ~ListaDoble();
    //Funciones
    bool insertarData(int dato);
    void listaDobleTetris(int tamaño);
    void imprimirLista();
    void insertarEnLista(Position* pos, int num,int limVer,int limHor);
    bool perdio(int elementos);
  
```

```

        bool gano(int elementos);
    };

#include "ListaDoble.h"

ListaDoble::ListaDoble()
{
    head = new NodoDoble();
    tail = new NodoDoble();
    head->last = NULL;
    head->next = tail;
    tail->next = NULL;
    tail->last = head;
}

ListaDoble::~ListaDoble()
{
    delete head;
    delete tail;
}

bool ListaDoble::insertarDatos(int dato)
{
    return false;
}

void ListaDoble::listaDobleTetris(int tamaño)
{
    int cont = 0;
    do {
        NodoDoble* nuevoNodo = new NodoDoble();
        nuevoNodo->dato = 0;
        nuevoNodo->last = tail->last;
        nuevoNodo->next = tail;
        tail->last->next = nuevoNodo;
        tail->last = nuevoNodo;
        cont = cont + 1;
    } while (cont != tamaño);
}

void ListaDoble::imprimirLista()
{
    std::fstream enter;
    enter.open("Lista.txt", std::fstream::out);
    int numero;
    NodoDoble* aux = head->next;
    enter << "La lista es:\n";
    while (aux != tail) {

        if (aux->dato == 0 || aux->dato== -1) {
            std::cout << " ";
            enter << " ";
            aux = aux->next;
        }
    }
}

```

```

        else {
            std::cout << aux->dato;
            numero= aux->dato;
            enter << numero;
            aux = aux->next;
        }
    }

void ListaDoble::insertarEnLista(Position* pos, int num, int limVer,int
limHor)
{
    int cont=0;
    if (pos->getPosY() >= limVer-1) {
        NodoDoble* aux = new NodoDoble();
        aux = head->next;
        while (cont != pos->getPosX() - 3) {
            aux = aux->next;
            cont = cont + 1;
        }
        if (cont % 3 == 0 && (aux->dato ==0 || aux->dato == -1)) {
            if (aux->dato == num) {
                aux->dato = -1;
            }
            else {
                aux->dato = num;
            }
        }
        else {
            if (pos->getPosX() - 3 == 0) {
                if (aux->dato == num) {
                    aux->dato = -1;
                    return;
                }
                cont = 0;
                aux = head->next;
                while (aux != tail) {
                    if ((aux->dato == 0 || aux->dato == -1) &&
cont%3==0) {
                        aux->dato = num;
                        return;
                    }
                    else {
                        aux = aux->next;
                        cont = cont + 1;
                    }
                }
            }
            else if (cont == limHor-8) {
                if (aux->dato == num) {
                    aux->dato = -1;
                    return;
                }
                cont = pos->getPosX()-3;
                aux = tail->last;
                while (aux != head) {

```

```

                if ((aux->dato == 0 || aux->dato == -1) && cont
% 3 == 0) {
                    aux->dato = num;
                    return;
                }
                else {
                    aux = aux->last;
                    cont = cont - 1;
                }
            }
        }else if(cont%3==0) {
            if (aux->dato == num) {
                aux->dato = -1;
                return;
            }
            cont = 0;
            aux = head->next;
            while (aux != tail) {
                if ((aux->dato == 0 || aux->dato == -1) && cont
% 3 == 0) {
                    aux->dato = num;
                    return;
                }
                else {
                    aux = aux->next;
                    cont = cont + 1;
                }
            }
            if (aux->dato == num) {
                aux->dato = -1;
                return;
            }
            cont = pos->getPosX() - 3;
            aux = tail->last;
            while (aux != head) {
                if ((aux->dato == 0 || aux->dato == -1) && cont
% 3 == 0) {
                    aux->dato = num;
                    return;
                }
                else {
                    aux = aux->last;
                    cont = cont - 1;
                }
            }
        }
    }

bool ListaDoble::perdio(int elementos)
{
    int cont = 0;
    int cont2 = 0;
    NodoDoble* aux = new NodoDoble();
    aux = head->next;

```

```

        while (aux != tail) {
            if (aux->dato != 0 ) {
                cont = cont + 1;
            }
            if (aux->dato == -1) {
                cont2 = cont2 + 1;
            }
            aux = aux->next;
        }
        if (cont == elementos && cont2==0) {
            return false;
        }
        return true;
    }

bool ListaDoble::gano(int elementos)
{
    int cont = 0;
    int cont2 = 0;
    int cont3 = 0;
    int total = 0;
    NodoDoble* aux = new NodoDoble();
    aux = head->next;
    while (aux != tail) {
        if (aux->dato == -1) {
            cont = cont + 1;
        }
        if (aux->dato == 0 && cont3%3==0) {
            cont2 = cont2 + 1;
        }
        aux = aux->next;
        cont3 = cont3 + 1;
    }
    if (cont2 == elementos) {
        cont2 = cont2 - 1;
    }
    total = cont + cont2;
    if (total == elementos) {
        return false;
    }
    return true;
}

#include <cstddef>
class NodoDoble
{
private:
    int dato;
    NodoDoble* next;
    NodoDoble* last;
public:
    NodoDoble();
    ~NodoDoble();

    friend class ListaDoble;
};

```

```

#include "NodoDoble.h"

NodoDoble::NodoDoble()
{
    dato = NULL;
    next = NULL;
    last = NULL;
}

NodoDoble::~NodoDoble()
{
}

class Position
{
private:
    int posX;
    int posY;
    int num;
public:
    Position();
    ~Position();
    void setPosXY(int X, int Y,int num);
    int getPosX();
    int getPosY();
    int getNum();
};

#include "Position.h"

Position::Position()
{
    this->posX = 0;
    this->posY = 0;
    this->num = 0;
}

Position::~Position()
{
}

void Position::setPosXY(int X, int Y,int num)
{
    this->posX = X;
    this->posY = Y;
    this->num = num;
}

int Position::getPosX()
{
    return this->posX;
}

```

```

int Position::getPosY()
{
    return this->posY;
}

int Position::getNum() {
    return this->num;
}

#include <stdio.h>
#include <windows.h>
#include <iostream>
#include <string.h>
#include <time.h>
#include <conio.h>
#include "Position.h"
#define UP 72
#define DOWN 80
#define LEFT 75
#define RIGHT 77

class TetrisFunction
{
private:
    int limiteAleatorio;
    int saltos;
public:
    //Constructor
    TetrisFunction(int limiteAleatorio,int saltos);
    //Destructor
    ~TetrisFunction();
    void gotoxy(int x, int y);
    void frame(int limVer,int limHoriz);
    void hideCursor();
    void imprimir(int limVer, int limHoriz, Position*& pos);

};

#include "TetrisFunction.h"

TetrisFunction::TetrisFunction(int limiteAleatorio,int saltos)
{
    srand(time(NULL));
    this->limiteAleatorio = limiteAleatorio;
    this->saltos = saltos;
}

TetrisFunction::~TetrisFunction()
{
}

void TetrisFunction::gotoxy(int x, int y) {
    HANDLE hcon;
    hcon = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD dwPos;
    dwPos.X = x;

```

```

        dwPos.Y = y;
        SetConsoleCursorPosition(hcon, dwPos);
    }
void TetrisFunction::frame(int limVer,int limHoriz) {
    hideCursor();
    char a = char(223);
    char b[] = { a };
    std::string c;
    c.assign(b, 1);
    for (int i = 0; i <= limHoriz; i++) {
        gotoxy(i, 0);
        std::cout << c;
    }
    a = char(219);
    char d[] = { a };
    c.assign(d, 1);
    for (int i = 0; i <= limVer; i++) {
        gotoxy(limHoriz, i);
        std::cout << c;
    }
    c.assign(b, 1);
    for(int i = limHoriz; i >= 0; i--) {
        gotoxy(i, limVer);
        std::cout << c;
    }
    c.assign(d, 1);
    for (int i = limVer-1; i >= 0; i--) {
        gotoxy(0, i);
        std::cout << c;
    }
    gotoxy(0, limVer+1);
}
void TetrisFunction::hideCursor()
{
    HANDLE consoleHandle = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_CURSOR_INFO info;
    info.dwSize = 100;
    info.bVisible = FALSE;
    SetConsoleCursorInfo(consoleHandle, &info);
}

void TetrisFunction::imprimir(int limVer,int limHoriz,Position *&pos)
{
    int num = 0;
    int posY = 0;
    int salto = 0;
    system("cls");
    frame(limVer, limHoriz);
    if (_kbhit()) {
        char key = _getch();
        switch (key) {
        case DOWN:
            if (pos->getPosY() >= limVer - 1) {
                num = 1 + rand() % (limiteAleatorio - 1);
                do {
                    posY = (3 + rand() % ((limHoriz - 4) - 3));
                } while (posY % 3 != 0);
            }
        }
    }
}

```

```

        pos->setPosXY(posY, (pos->getPosY() + 1), num);
        gotoxy(pos->getPosX(), pos->getPosY());
        std::cout << pos->getNum();
    }
    else {
        if (pos->getPosY() + 5 >= limVer - 1) {
            num = 1 + rand() % (limiteAleatorio - 1);
            do {
                posY = 3 + rand() % ((limHoriz - 4) - 3);
            } while (posY % 3 != 0);
            pos->setPosXY(posY, (pos->getPosY() + 1), num);
            gotoxy(pos->getPosX(), pos->getPosY());
            std::cout << pos->getNum();
        }
        else {
            pos->setPosXY(pos->getPosX(), (pos->getPosY() +
5), (pos->getNum()));
            gotoxy(pos->getPosX(), pos->getPosY());
            std::cout << pos->getNum();
        }
    }
    break;
}
case LEFT:
    if (pos->getPosX() <= 3) {
        pos->setPosXY(pos->getPosX(), (pos->getPosY() + 1),
(pos->getNum()));
        gotoxy(pos->getPosX(), pos->getPosY());
        std::cout << pos->getNum();
    }
    else {
        if (pos->getPosX() - saltos <= 3) {
            salto = pos->getPosX() - saltos;
            pos->setPosXY(pos->getPosX()-salto, (pos-
>getPosY() + 1), (pos->getNum()));
            gotoxy(pos->getPosX(), pos->getPosY());
            std::cout << pos->getNum();
        }
        else {
            pos->setPosXY(pos->getPosX() - saltos, (pos-
>getPosY() + 1), (pos->getNum()));
            gotoxy(pos->getPosX(), pos->getPosY());
            std::cout << pos->getNum();
        }
    }
    break;
}
case RIGHT:
    if (pos->getPosX() >= limHoriz-5) {
        pos->setPosXY(pos->getPosX(), (pos->getPosY() + 1),
(pos->getNum()));
        gotoxy(pos->getPosX(), pos->getPosY());
        std::cout << pos->getNum();
    }
    else {
        if (pos->getPosX() + saltos >= limHoriz - 5) {
            salto = saltos - ((pos->getPosX() + saltos)-
(limHoriz-5));

```

```

                pos->setPosXY(pos->getPosX() + salto, (pos-
>getPosY() + 1), (pos->getNum()));
                gotoxy(pos->getPosX(), pos->getPosY());
                std::cout << pos->getNum();
            }
            else {
                pos->setPosXY(pos->getPosX() + saltos, (pos-
>getPosY() + 1), (pos->getNum()));
                gotoxy(pos->getPosX(), pos->getPosY());
                std::cout << pos->getNum();
            }
        }
        break;
    case 13:
        gotoxy(pos->getPosX(), pos->getPosY());
        std::cout << pos->getNum();
        gotoxy((limHoriz/2)-7, limVer/2);
        std::cout << "JUEGO PAUSADO";
        gotoxy((limHoriz / 2) - 14, (limVer / 2)+1);
        std::cout << "PRESIONE CUALQUIER TECLA PARA CONTINUAR";
        getch();
        break;
    default:
        if (pos->getPosY() >= limVer - 1) {
            num = 1 + rand() % (limiteAleatorio - 1);
            do {
                posY = 3 + rand() % ((limHoriz - 4) - 3);
            } while (posY % 3 != 0);
            pos->setPosXY(posY, (pos->getPosY() + 1), num);
            gotoxy(pos->getPosX(), pos->getPosY());
            std::cout << pos->getNum();
        }
        else {
            pos->setPosXY(pos->getPosX(), (pos->getPosY() + 1),
(pos->getNum()));
            gotoxy(pos->getPosX(), pos->getPosY());
            std::cout << pos->getNum();
        }
        break;
    }
}
else {
    if (pos->getPosY() == 0) {
        num = 1 + rand() % (limiteAleatorio - 1);
        do {
            posY = (3 + rand() % ((limHoriz - 4) - 3));
        } while (posY % 3 != 0);
        pos->setPosXY(posY, (pos->getPosY() + 1), num);
        gotoxy(pos->getPosX(), pos->getPosY());
        std::cout << pos->getNum();
    }
    else {
        if (pos->getPosY() >= limVer - 1) {
            num = 1 + rand() % (limiteAleatorio - 1);
            do {
                posY = (3 + rand() % ((limHoriz - 4) - 3));

```

```

        } while (posY % 3 != 0);
        pos->setPosXY(posY, (1), num);
        gotoxy(pos->getPosX(), pos->getPosY());
        std::cout << pos->getNum();
    }
    else {
        pos->setPosXY(pos->getPosX(), (pos->getPosY() + 1),
        posY, pos->getPosY());
        std::cout << pos->getNum();
    }
}
}

#include <iostream>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include<fstream>
#include <sstream>
#define ARRIBA 72
#define ABAJO 80
#define ENTER 13

using namespace std;

class Menus{

public:
    void menuInicio(){
system ("color 9F" );
printf("\t\t\t=====MENU=====\n");
gotoxy(5,1);
cout<<"\t\t\t-----MENU-----" << endl;
gotoxy(5,2);
cout<<"\t\t\t\t* Pixelar imagen";
gotoxy(5,3);
cout<<"\t\t\t\t* Generar PDF del resultado";
gotoxy(5,4);
cout<<"\t\t\t\t* Generar QR";
gotoxy(5,5);
cout<<"\t\t\t\t* Mostrar codigo de barras";
gotoxy(5,6);
cout<<"\t\t\t\t* Encriptar";
gotoxy(5,7);
cout<<"\t\t\t\t* Desencriptar";
gotoxy(5,8);
cout<<"\t\t\t\t* Empezar el juego";
gotoxy(5,9);
cout<<"\t\t\t\t* Salir\n";
printf("\t\t\t=====");
}
}

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 Selector(int i)
{
    gotoxy(30,2+i);
}

void menuDinamico(){
    int i = 0;
    ifstream doc;
    char tecla;
    menuInicio();
    Selector(i);
    while(true){
        tecla = getch();
        switch(tecla)
        {
            case ARRIBA:
                i--;
                if(i < 0)
                {
                    i = 9;
                }
                Selector(i);
                break;
            case 59:
                system("cls");
                cout<<"Ayuda"<<endl;
                system("Tetris.chm");
                system("pause");
                system("cls");
                menuDinamico();
                break;
            case ABAJO:
                i++;
                if(i == 0)
                {
                    i = 1;
                }
                if(i == 10)
                {
                    i = 0;
                }
                Selector(i);
                break;
            case ENTER:
                Selector(7);
                switch(i)
                {
                    case 0:
                        system("cls");
                        system("java -jar pixel.jar");
                }
        }
    }
}

```

```

        system("macara.jpg");
        system("image_pixelated.jpg");
        cout<<"Su imagen fue pixelada\n";
        system("Pause");
        system("cls");
        menuDinamico();
        break;
    case 1:
        system("cls");

        doc.open("Lista.txt", fstream::in);
        if(doc.fail()){
            system("cls");
            cout<<"ERROR: El archivo de agenda
no esta creado\n";
            system("pause");
            system("cls");
            menuDinamico();
        }
        else{
            doc.close();
            system("cls");
            system("txt2pdf.exe Lista.txt
Lista.pdf -oao -pfs60 -pps43 -ptc0 -width3000 -height2000");
            cout<<"Archivo generado
exitosamente\n";
            system("pause");
            system("cls");
            menuDinamico();
        }

    case 2:
        system("cls");
        cout<<"Mostrar QR de licencia\n";
        system("Pause");
        system("QRlicencia.jpg");
        system("cls");
        menuDinamico();

    case 3:
        system("cls");
        cout<<"Mostrar codigo de barras de los
autores";
        system("Pause");
        system("Codicodebarras.gif");
        system("cls");
        menuDinamico();

    case 4:
        system("cls");
        char original[FILENAME_MAX],
        enciptado[FILENAME_MAX];

```

```

        puts("Introduzca el nombre del archivo
original:");
scanf("%s", original);

puts("Nombre del archivo encriptado:");
scanf("%s", encriptado);

puts("Introduzca la clave (0-255)");
scanf("%u", &temp);
clave = temp & 0xFFU;

if((entrada = fopen(original, "rb")) ==
NULL) {
    printf("Error al tratar de leer el
archivo %s", original);
    //return EXIT_FAILURE;
} else if ((salida = fopen(encriptado,
"wb")) == NULL) {
    printf("Error al tratar de leer el
archivo %s", encriptado);
    //return EXIT_FAILURE;
}

while ((c = getc(entrada)) != EOF)
putc((c + clave) & 0xFF, salida);

fclose(salida);
fclose(entrada);

system("pause");
system("cls");
menuDinamico();

case 5:
system("cls");
char secundario[FILENAME_MAX],
d, codigo;
FILE *inicio, *fin;
unsigned aux;

puts("Introduzca el nombre del archivo
original:");
scanf("%s", secundario);

puts("Nombre del archivo encriptado:");
scanf("%s", desencriptado);

puts("Introduzca la clave (0-255)");
scanf("%u", &aux);
clave = aux & 0xFFU;

if((inicio = fopen(secundario, "rb")) ==
NULL) {
    printf("Error al tratar de leer el
archivo %s", secundario);
}

```

```

                //return EXIT_FAILURE;
        }else if ((fin = fopen(encriptado, "wb")) == NULL) {
                printf("Error al tratar de leer el
archivo %s", desencriptado);
                //return EXIT_FAILURE;
        }

        while ((d = getc(inicio)) != EOF)
putc((d - codigo) & 0xFF, fin);

fclose(inicio);
fclose(fin);

system("pause");
system("cls");
menuDinamico();

default:
exit(0);
}
}
}

void portada()
{
    system ("color 9F" );
cout<<"\t\t\t\t\t\tBIENVENIDOS A NUESTRO PROGRAMA"<<endl;
cout<<"\n\n";
cout<<"\t\t:::::::::::::::::::      :::::::::::      :::::::::::      ::::::::::::
:::::::::::::::::::      :::::::::::      :::::::::::      ::::::::::::
cout<<"\t\t      :::      ::::      ::::      ::::      ::::      ::::      :::
:::      :::      ::::      ::::      ::::      ::::      ::::      :::
cout<<"\t\t      :::      ::::      ::::      ::::      ::::      ::::      :::
:::      :::      ::::      ::::      ::::      ::::      ::::      :::
cout<<"\t\t      :::      ::::::::::::      ::::      ::::::::::::
:::      ::::::::::::      ::::      ::::::::::::
cout<<"\t\t      :::      ::::      ::::      ::::      ::::      :::
:::      :::      ::::      ::::      ::::      ::::      :::
cout<<"\t\t      :::      ::::      ::::      ::::      ::::      :::
:::      :::      ::::      ::::      ::::      ::::      :::
cout<<"\t\t      :::      ::::::::::::      ::::      ::::::::::::
:::      ::::::::::::      ::::      ::::::::::::
cout<<"\n";
system("PAUSE");
}
};

```

```

int main()
{
    Menus menu;
    menu.portada();
    system("cls");
    menu.menuDinamico();
    return 0;
}

#include <iostream>
#include <stdio.h>
#include "ListaDoble.h"
#include "TetrisFunction.h"
#include "Position.h"
int main()
{
    system("Menu.exe");
    int n = 6;//Numero de elementos a insertar
    int m = 0;
    if (n >= 11) {
        m = 9;
    }
    else {
        m = n;
    }
    int MAXH = 0;
    bool escape=true;
    int cont = 0;
    int MAXV = 0;
    MAXH = 3 * (n - 1) + 8;
    MAXV = (int) (0.60 * MAXH);
    Position* pos = new Position();
    ListaDoble lista = ListaDoble();
    lista.listaDobleTetris(MAXH-7);
    system("color F9");
    TetrisFunction tetris = TetrisFunction(m+1,3);
    tetris.hideCursor();
    tetris.frame(MAXV, MAXH);
    do {
        if (lista.gano(n) == false) {
            escape = false;
        }
        if (lista.perdio(n) == false) {
            escape = false;
        }
        tetris.imprimir(MAXV, MAXH, pos);
        tetris.gotoxy(3, MAXV - 1);
        lista.imprimirLista();
        lista.insertarEnLista(pos, pos->getNum(), MAXV, MAXH);
        Sleep(150);
    } while (escape);

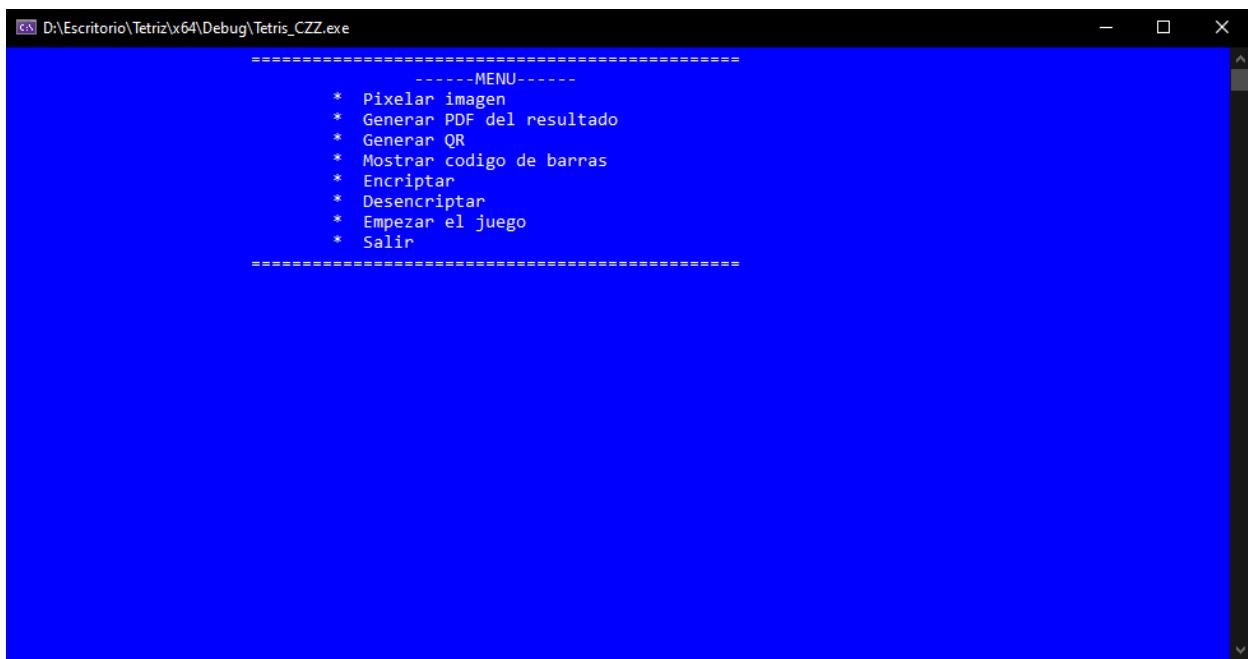
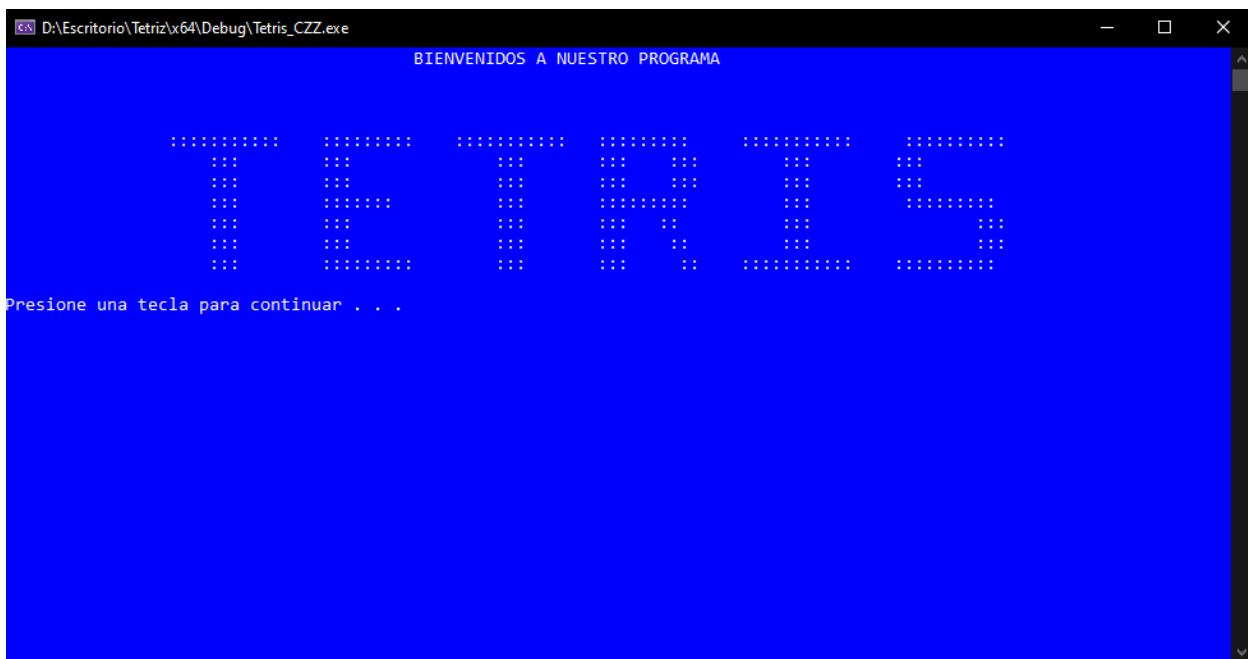
    tetris.gotoxy(0,MAXV+10);

    lista.insertarEnLista(pos, pos->getNum(), MAXV, MAXH);
    system("cls");
    std::cout << "La lista es:\n";
}

```

```
    lista.imprimirLista();
    std::cout << "\n";
    system("pause");
    return 0;
}
```

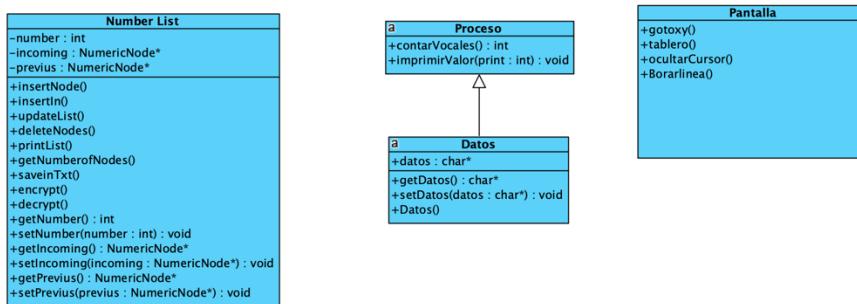
Ejecucion





Grupo Alvarez-Garcia

Modelado



Codigo

```
#pragma once

#include<fstream>
#include <iostream>
#include <windows.h>
static HWND hConWnd;

HWND BCX_Bitmap(std::string, HWND = 0, int = 0, int = 0, int = 0, int = 0,
int = 0, int = 0, int = 0, int = 0);
HWND GetConsoleWndHandle(void);

HWND BCX_Bitmap(std::string Text, HWND hWnd, int id, int X, int Y, int W, int
H, int Res, int Style, int Exstyle)
{
    HWND A;
    HBITMAP hBitmap;

    // set default style
    if (!Style) Style = WS_CLIPSIBLINGS | WS_CHILD | WS_VISIBLE | SS_BITMAP
| WS_TABSTOP;

    // form for the image
    A = CreateWindowEx(Exstyle, "static", NULL, Style, X, Y, 0, 0, hWnd,
(HMENU)id, GetModuleHandle(0), NULL);

    // Text contains filename
    hBitmap = (HBITMAP)LoadImage(0, Text.c_str(), IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE | LR_CREATEDIBSECTION);

    // auto-adjust width and height
```

```

        if (W || H) hBitmap = (HBITMAP)CopyImage(hBitmap, IMAGE_BITMAP, W, H,
LR_COPYRETURNORG);
        SendMessage(A, (UINT)STM_SETIMAGE, (WPARAM)IMAGE_BITMAP,
(LPARAM)hBitmap);
        if (W || H) SetWindowPos(A, HWND_TOP, X, Y, W, H, SWP_DRAWFRAME);
        return A;
    }

// tricking Windows just a little ...
HWND GetConsoleWndHandle(void)
{
    HWND hConWnd;
    OSVERSIONINFO os;
    char szTempTitle[64], szClassName[128], szOriginalTitle[1024];

    os.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
    GetVersionEx(&os);
    // may not work on WIN9x
    if (os.dwPlatformId == VER_PLATFORM_WIN32s) return 0;

    GetConsoleTitle(szOriginalTitle, sizeof(szOriginalTitle));
    sprintf_s(szTempTitle, "%u - %u", GetTickCount(),
GetCurrentProcessId());
    SetConsoleTitle(szTempTitle);
    Sleep(60);
    // handle for NT and XP
    hConWnd = FindWindow(NULL, szTempTitle);
    SetConsoleTitle(szOriginalTitle);

    // may not work on WIN9x
    if (os.dwPlatformId == VER_PLATFORM_WIN32_WINDOWS)
    {
        hConWnd = GetWindow(hConWnd, GW_CHILD);
        if (hConWnd == NULL) return 0;
        GetClassName(hConWnd, szClassName, sizeof(szClassName));
        while (_stricmp( szClassName, "ttyGrab" ) != 0 )
        //while (strcmp(szClassName, "ttyGrab") != 0)
        {
            hConWnd = GetNextWindow(hConWnd, GW_HWNDNEXT);
            if (hConWnd == NULL) return 0;
            GetClassName(hConWnd, szClassName, sizeof(szClassName));
        }
    }
    return hConWnd;
}

#pragma once
#include <time.h>
#include <stdlib.h>

class Impresion {
public:
    int generarPiezas();
};

int Impresion::generarPiezas() {

```

```

        int numero;
        srand(time(NULL));
        numero = rand() % 5 + 1;
        return numero;
    }

#pragma once
#include <iostream>
#include <string>
#include <fstream>

class NumericNode {
private:
    int number;
    NumericNode* incoming;
    NumericNode* previous;

    friend class CircularList;
public:
    NumericNode(int data, NumericNode* nI, NumericNode* nP)
    {
        incoming = nI; previous = nP; number = data;
    }
};

class CircularList {
private:
    NumericNode* first;
    NumericNode* last;

public:
    CircularList() { first = last = nullptr; }
    void insertNode(int data, bool flag);
    void insertIn(int data, int pos);
    void updateCList();
    void deleteNodes(int pos);
    void printCList();
    int getNumberOfNodes();
    void saveInTxt();
    void encrypt();
    void decrypt();
};

#include "NumberCList.h"
#include <string>
#include <fstream>

void CircularList::insertNode(int data, bool flag) {
    NumericNode* primal;
    primal = first;

    if (first == nullptr) {
        first = new NumericNode(data, first, nullptr);
    }
    else if (flag) {
        while (primal->incoming != nullptr) {
            primal = primal->incoming;
    }
}

```

```

        }
        primal->incoming = new NumericNode(data, nullptr, primal);
    }
    else {
        primal = new NumericNode(data, first, nullptr);
        first = primal;;
    }
}

void CircularList::insertIn(int data, int pos) {
    NumericNode* newNode = new NumericNode(data, nullptr, nullptr);
    NumericNode* current;
    current = first;

    for (int i = 1; i < pos; ++i) current = current->incoming;
    newNode->incoming = current->incoming;
    newNode->previous = current;

    if (current->incoming == nullptr) last = newNode;
    else {
        current->incoming->previous = newNode;
        current->incoming = newNode;
    }
}

void CircularList::updateCList() {
    NumericNode* current;
    current = first;
    int pos = 0;
    while (current->incoming != nullptr) {
        if (current->number == (current->incoming)->number) {
            deleteNodes(pos);
        }
        pos++;
        current = current->incoming;
    }
}

void CircularList::deleteNodes(int pos) {
    NumericNode* actual=first;
    int auxPos = 0;
    if (first != nullptr) {
        if (pos == 0) {
            first = first->incoming->incoming;
            first->previous = nullptr;
        }
        else {
            while (pos != auxPos) {
                actual = actual->incoming;
                auxPos++;
            }
            actual->previous->incoming = actual->incoming->incoming;
            actual->incoming->incoming->previous = actual->previous;
        }
    }
}

```

```

void CircularList::printCList() {
    NumericNode* current;
    current = first;

    if (first != nullptr) {
        while (current) {
            std::cout << current->number;
            current = current->incoming;
        }
    }
}

int CircularList::getNumberOfNodes() {
    NumericNode* current;
    current = first;
    int auxCont = 0;
    if (first != nullptr) {
        while (current) {
            auxCont++;
            current = current->incoming;
        }
    }

    return auxCont;
}

void CircularList::saveInTxt() {
    std::fstream _txtArchive;
    _txtArchive.open("Solution.txt", std::fstream::out);
    _txtArchive << "SOLUTION" << std::endl;
    for (int i = 0; i < 7; i++) {
        for (int k = 0; k < getNumberOfNodes(); k++) {
            _txtArchive << "0 ";
        }
        _txtArchive << std::endl;
    }

    NumericNode* current;
    current = first;
    if (first != nullptr) {
        while (current) {
            _txtArchive << current->number << " ";
            current = current->incoming;
        }
    }
}

void CircularList::encrypt() {
    std::fstream _txtArchive;
    _txtArchive.open("Solution.txt", std::fstream::app);
}

void CircularList::decrypt() {
}

```



```

        " || "
        " _____ || \n" <<
        "\n" << endl;
    }

void Pantalla::ocultarCursor()
{
    HANDLE h_con;
    h_con = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_CURSOR_INFO cci;
    cci.dwSize = 28;
    cci.bVisible = FALSE;
    SetConsoleCursorInfo(h_con, &cci);
}

void Pantalla::borrarLinea(int posY)
{
    for (int i = 2; i < 26; i++)
    {
        gotoxy(i, posY);
        cout << " ";
    }
}

#pragma once
#include <iostream>
#include <Windows.h>

enum COLOR {
    // Text foreground colors
    // Standard text colors
    GRAY_TEXT = 8, BLUE_TEXT, GREEN_TEXT,
    TEAL_TEXT, RED_TEXT, PINK_TEXT,
    YELLOW_TEXT, WHITE_TEXT,
    // Faded text colors
    BLACK_TEXT = 0, BLUE_FADE_TEXT, GREEN_FADE_TEXT,
    TEAL_FADE_TEXT, RED_FADE_TEXT, PINK_FADE_TEXT,
    YELLOW_FADE_TEXT, WHITE_FADE_TEXT,
    // Standard text background color
    GRAY_BACKGROUND = GRAY_TEXT << 4, BLUE_BACKGROUND = BLUE_TEXT << 4,
    GREEN_BACKGROUND = GREEN_TEXT << 4, TEAL_BACKGROUND = TEAL_TEXT << 4,
    RED_BACKGROUND = RED_TEXT << 4, PINK_BACKGROUND = PINK_TEXT << 4,
    YELLOW_BACKGROUND = YELLOW_TEXT << 4, WHITE_BACKGROUND = WHITE_TEXT <<
4,
    // Faded text background color
    BLACK_BACKGROUND = BLACK_TEXT << 4, BLUE_FADE_BACKGROUND =
    BLUE_FADE_TEXT << 4,
    GREEN_FADE_BACKGROUND = GREEN_FADE_TEXT << 4, TEAL_FADE_BACKGROUND =
    TEAL_FADE_TEXT << 4,
    RED_FADE_BACKGROUND = RED_FADE_TEXT << 4, PINK_FADE_BACKGROUND =
    PINK_FADE_TEXT << 4,
    YELLOW_FADE_BACKGROUND = YELLOW_FADE_TEXT << 4, WHITE_FADE_BACKGROUND =
    WHITE_FADE_TEXT << 4
};

void menuSelecInformation() {
    system("cls");
    std::cout << "***** NUMERIC TETRIS *****" << std::endl;
}

```

```

        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
GRAY_BACKGROUND | BLUE_TEXT);
        std::cout << " Informacion" " << std::endl;
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
WHITE_FADE_TEXT);
        std::cout << " JUGAR AHORA!" " << std::endl;
        std::cout << " Salir" " << std::endl;
        std::cout << "*****" " << std::endl;
    }

void menuSelecPlayNow() {
    system("cls");
    std::cout << "***** NUMERIC TETRIS *****" << std::endl;
    std::cout << " Informacion" " << std::endl;
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
GRAY_BACKGROUND | BLUE_TEXT);
    std::cout << " JUGAR AHORA!" " << std::endl;
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
WHITE_FADE_TEXT);
    std::cout << " Salir" " << std::endl;
    std::cout << "*****" " << std::endl;
}

void menuSelecExit() {
    system("cls");
    std::cout << "***** NUMERIC TETRIS *****" << std::endl;
    std::cout << " Informacion" " << std::endl;
    std::cout << " JUGAR AHORA!" " << std::endl;
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
GRAY_BACKGROUND | BLUE_TEXT);
    std::cout << " Salir" " << std::endl;
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
WHITE_FADE_TEXT);
    std::cout << "*****" " << std::endl;
}

void utilMenu(int selector) {
    switch (selector) {
    case 0:
        menuSelecInformation();
        break;
    case 1:
        menuSelecPlayNow();
        break;
    case 2:
        menuSelecExit();
        break;
    }
}

/**
 *
*
*                               UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE
*
*                               CARRERA DE INGENIERIA DE SOFTWARE
*
*                               TERCER SEMESTRE
*
*/

```

```

* PROYECTO DEL SEGUNDO PARCIAL
* AUTORES: Sebastian Alvarez
*           Renan Garcia
* NRC:2967
* FECHA DE MODIFICACION: 11/DIC/2019
* FECHA DE ENTREGA: 12/DIC/2019
*
* STATUS: Finalizado
*/
```

```

#include <iostream>
#include <string>
#include <Windows.h>
#include <time.h>
#include <cstdlib>
#include <conio.h>
#include "UtilMenu's.h"
#include "NumberCLList.h"
#include "Imagen.h"
#include "Pantalla.h"
#include "Impresion.h"

#define UP 72
#define DOWN 80
#define ENTER 13
#define ESCAPE 27
#define LEFT 75
#define RIGHT 77

using namespace std;

void VisualMenu(CircularList, int);

void showInformation() {
    std::cout <<
        "
        "
        "
FUERZAS ARMADAS ESPE\n" <<
        "
        "
        "
        "
        "
Renan Garcia" << std::endl;
}

int main() {
    CircularList CList;
    int posY = 0;

    for (;;) {
        VisualMenu(CList, posY);
    }

    return 0;
}
```

UNIVERSIDAD DE LAS
FUERZAS ARMADAS ESPE

Este videojuego fue desarrollado como
proyecto de segundo parcial para la

materia de Estructura de Datos

Lenguaje: C++\n" <<
IDE: Visual Studio Community 2019\n" <<
Libreria grafica: SFML 2.5.1\n\n\n" <<
Desarrollado por: Sebastian Alvarez y
Renan Garcia

```

}

void VisualMenu(CircularList CList, int posY) {
    menuSelectInformation();
    char key;

    while (true) {
        key = _getch();

        switch (key) {
            case UP:
                posY--;
                if (posY < 0) {
                    posY = 2;
                }
                utilMenu(posY);
                break;

            case DOWN:
                posY++;
                if (posY == 0) {
                    posY = 1;
                }
                if (posY > 2) {
                    posY = 0;
                }
                utilMenu(posY);
                break;

            case ENTER:
                Pantalla screen;
                Impresion printer;
                bool GAMEOVER = false;
                int numero, siguienteNumero, x, y, yBarra, posicion;
                char tecla;

                switch (posY) {
                    case 0:
                        /*Show information about the game*/
                        showInformation();
                        hConWnd = GetConsoleWindow();
                        if (hConWnd) {
                            BCX_Bitmap("images/logo.bmp", hConWnd, 123, 1,
1, 0, 0);
                            Sleep(3000);
                            system("pause");
                        }

                        system("pause");
                        break;

                    case 1:
                        /*PLAY NOW!*/
                        std::cout << "COMENZANDO..." << std::endl;
                        Sleep(2000);
                        system("cls");
                        system("color F3");
                }
        }
    }
}

```

```

screen.tablero();
srand(time(NULL));
x = 13, y = 1;
yBarra = 29, posicion = 13;
numero = printer.generarPiezas();
siguienteNumero = printer.generarPiezas();
screen.gotoxy(x, y);
screen.ocultarCursor();
CList.insertNode(0, true);

while (!GAMEOVER) {
    Sleep(60);

    screen.borrarLinea(29);
    screen.gotoxy(x, y);
    cout << numero;
    screen.gotoxy(x, y);
    cout << " ";
    y++;
    screen.gotoxy(x, y);
    cout << numero;
    screen.gotoxy(posicion, yBarra);
    CList.printCList();

    if (y == 29) {
        if (x < posicion) {
            CList.insertNode(numero, false);
            posicion--;
        }
        else if (x > posicion + CList.getNumberOfNodes() - 1) {
            CList.insertNode(numero, true);
        }
        else if (x >= posicion && x <= posicion + CList.getNumberOfNodes() - 1) {
            CList.insertIn(numero, x - posicion);
        }
        if (CList.getNumberOfNodes() == 26) {
            GAMEOVER = true;
        }
        CList.updateCList();
        CList.saveInTxt();

        y = 1;
        x = 13;
        numero = siguienteNumero;
        siguienteNumero =
    }

    printer.generarPiezas();
    screen.gotoxy(x, y);
    cout << siguienteNumero;
}

if (_kbhit()) {
    tecla = _getch();
    if (tecla == ESCAPE) {
        GAMEOVER = true;
}

```

```

        }
        else if (tecla == RIGHT && x != 25) {
            screen.borrarLinea(y);
            x++;
        }
        else if (tecla == LEFT && x != 0) {
            screen.borrarLinea(y);
            x--;
        }
    }
}

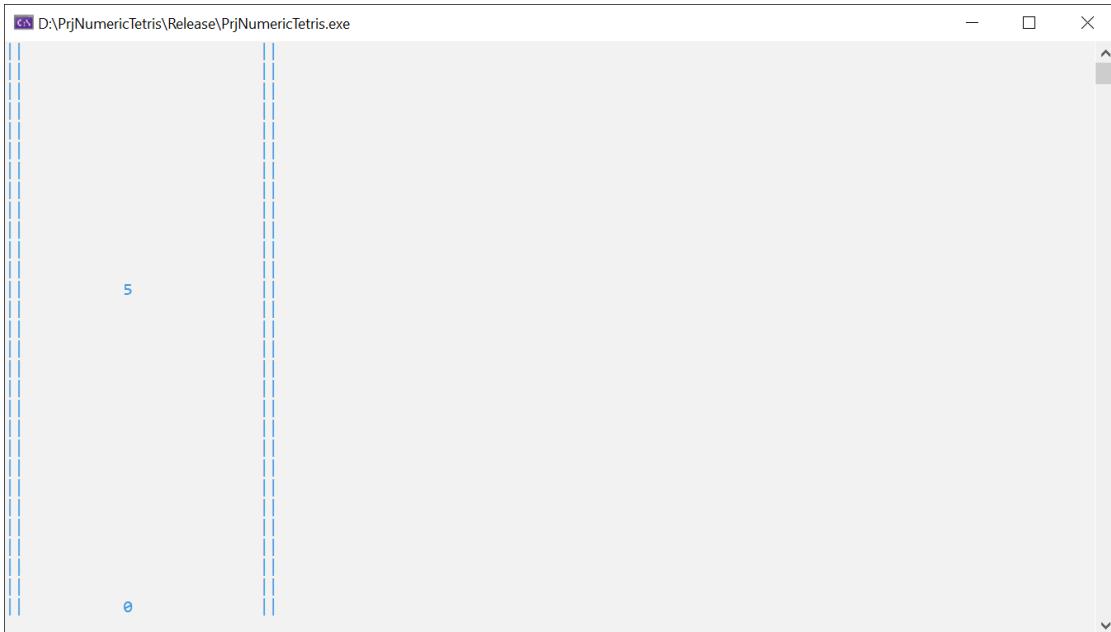
/*PDF creation*/
system("java -jar dist/textToPdf.jar");

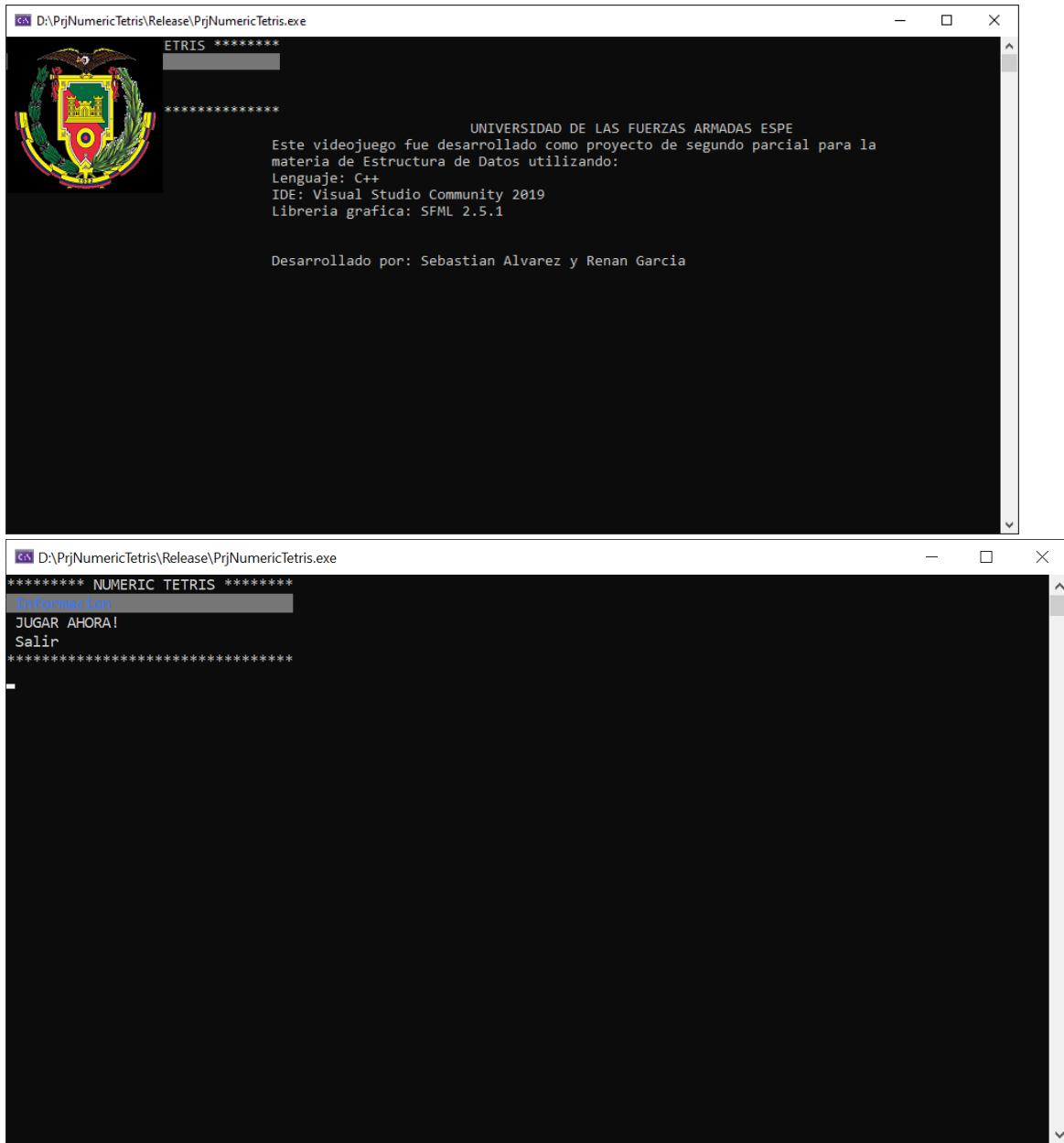
system("pause");
break;

case 2:
    /*Exit*/
    std::cout << "Gracias por jugar :)" << std::endl;
    exit(0);
}
utilMenu(posY);
break;
}
}
}

```

Ejecucion





Arbol Binario

se va ordenando según ingresen los números esto dependerá del valor de la raíz se comprobando con la raíz los números que ingresan y se van a la izquierda si es menor a la raíz y a la derecha si es mayor que la raíz.

Código:

```
#include <iostream>
#include <stdlib.h>
#include <windows.h>

using namespace std;

/*----- Estructura del arbol -----*/
typedef class nodo {
```

```

public:
    int nro;
    nodo* izq, * der;
}*ABB;

int numNodos = 0; // numero de nodos del arbol ABB
int numK = 0, k;      // nodos menores que un numero k ingresado

/* ----- Estructura de la cola -----*/
class nodoCola {
public:
    ABB ptr;
    nodoCola* sgte;
};

class cola {
public:
    nodoCola* delante;
    nodoCola* atras;
};

void inicializaCola(cola& q) {
    q.delante = NULL;
    q.atras = NULL;
}

void encola(cola& q, ABB n) {
    nodoCola* p;
    p = new nodoCola();
    p->ptr = n;
    p->sgte = NULL;
    if (q.delante == NULL)
        q.delante = p;
    else
        (q.atras)->sgte = p;
    q.atras = p;
}

ABB desencola(struct cola& q) {
    nodoCola* p;
    p = q.delante;
    ABB n = p->ptr;
    q.delante = (q.delante)->sgte;
    delete(p);
    return n;
}

ABB crearNodo(int x) {
    ABB nuevoNodo = new(nodo);
    nuevoNodo->nro = x;
    nuevoNodo->izq = NULL;
    nuevoNodo->der = NULL;
}

```

```

        return nuevoNodo;
    }

void insertar(ABB& arbol, int x) {
    if (arbol == NULL) {
        arbol = crearNodo(x);
        cout << "\n\t Insertado..!" << endl << endl;
    }
    else if (x < arbol->nro)
        insertar(arbol->izq, x);
    else if (x > arbol->nro)
        insertar(arbol->der, x);
}

void preOrden(ABB arbol) {
    if (arbol != NULL) {
        cout << arbol->nro << " ";
        preOrden(arbol->izq);
        preOrden(arbol->der);
    }
}

void enOrden(ABB arbol) {
    if (arbol != NULL) {
        enOrden(arbol->izq);
        cout << arbol->nro << " ";
        enOrden(arbol->der);
    }
}

void postOrden(ABB arbol) {
    if (arbol != NULL) {
        enOrden(arbol->izq);
        enOrden(arbol->der);
        cout << arbol->nro << " ";
    }
}

void gotoxy(int x, int y) {
    HANDLE hCon;
    COORD dwPos;

    dwPos.X = x;
    dwPos.Y = y;
    hCon = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleCursorPosition(hCon, dwPos);
}

void impresionArbol(ABB arbol, int auxY) {
    if (arbol == NULL)
        return;
}

```

```

auxX += 5;
impresionArbol(arbol->izq, auxY + 2);
gotoxy(10 + auxX - auxY, 15 + auxY);
cout << arbol->nro << endl << endl;
impresionArbol(arbol->der, auxY + 2);
}

bool busquedaRec(ABB arbol, int dato) {
    int r = 0; // 0 indica que no lo encontro

    if (arbol == NULL)
        return r;

    if (dato < arbol->nro)
        r = busquedaRec(arbol->izq, dato);

    else if (dato > arbol->nro)
        r = busquedaRec(arbol->der, dato);

    else
        r = 1; // son iguales, lo encontro

    return r;
}

ABB unirABB(ABB izq, ABB der) {
    if (izq == NULL) return der;
    if (der == NULL) return izq;

    ABB centro = unirABB(izq->der, der->izq);
    izq->der = centro;
    der->izq = izq;
    return der;
}

void elimina(ABB& arbol, int x) {
    if (arbol == NULL) return;

    if (x < arbol->nro)
        elimina(arbol->izq, x);
    else if (x > arbol->nro)
        elimina(arbol->der, x);

    else {
        ABB p = arbol;
        arbol = unirABB(arbol->izq, arbol->der);
        delete p;
    }
}

void menu() {
    //system("cls");
}

```

```

cout << "\n\t\t ..[ ARBOL BINARIO DE BUSQUEDA ].. \n\n";
cout << "\t [1] Insertar elemento \n";
cout << "\t [2] Mostrar arbol \n";
cout << "\t [3] Recorridos de profundidad \n";
cout << "\t [4] Buscar elemento \n";
cout << "\t [5] Eliminar elemento \n";
cout << "\t [6] SALIR \n";

cout << "\n\t Ingrese opcion : ";
}

void menu2() {
    //system("cls"); // para limpiar pantalla
    cout << endl;
    cout << "\t [1] En Orden \n";
    cout << "\t [2] Pre Orden \n";
    cout << "\t [3] Post Orden \n";
    cout << "\n\t Opcion : ";
}

int main() {
    ABB arbol = NULL;
    int x;
    int op, op2;

    system("color f9"); // poner color a la consola
    do
    {
        system("cls");
        menu(); cin >> op;
        cout << endl;

        switch (op)
        {
        case 1:
            cout << " Ingrese valor : "; cin >> x;
            insertar(arbol, x);
            system("pause");
            break;

        case 2:
            impresionArbol(arbol, 0);
            system("pause");
            break;

        case 3:
            menu2(); cin >> op2;

            if (arbol != NULL)
            {
                switch (op2)
                {

```

```

        case 1:
            enOrden(arbol); break;
        case 2:
            preOrden(arbol); break;
        case 3:
            postOrden(arbol); break;
        }
    }
    else
        cout << "\n\tArbol vacio..!";
    system("pause");
    break;

case 4:
    bool band;

    cout << " Valor a buscar: "; cin >> x;

    band = busquedaRec(arbol, x);

    if (band == 1)
        cout << "\n\tEncontrado...";
    else
        cout << "\n\tNo encontrado...";
    system("pause");
    break;

case 5:
    cout << " Valor a eliminar: "; cin >> x;
    elimina(arbol, x);
    cout << "\n\tEliminado..!";
    system("pause");
    break;
case 6:
    break;
default:
    cout << "Ingrese una opcion correcta" << endl;
    system("pause");
    break;
}

cout << "\n\n\n";
//system("pause"); // hacer pausa y presionar una tecla para continuar
} while (op != 6);

cout << "Gracias por utilizar :)" << endl;
exit(1);
return (0);
}

```