|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería en Software | **GUÍA**  No. 01 | **TIEMPO ESTIMADO:** |
| **ASIGNATURA:**  Estructura de datos  NRC: 2967 | **FECHA DE ELABORACION:** 27-01-2020  **SEMESTRE**: septiembre 2019 – enero 2020 | |
| **TÍTULO:**  Proyecto Segundo Parcial (Tetris) | **DOCENTE:** Ing. Fernando Solís | |

**OBJETIVO**

Aplicar el conocimiento de las estructuras de datos “Listas circulares doblemente enlazadas” en el desarrollo de un Tetris.

**INSTRUCCIONES**

1. Utilice como material principal, las principales herramientas para desarrollo en C++
2. Utilice información consultada en Internet y conocimiento adquirido en clase.

**ACTIVIDADES**

1. **Ubicación de recursos**
2. Formar grupos de máximo 2 personas por computador
3. Realizar el programa en cualquier IDE para C++
4. **Planteamiento del problema**

**p.e. Ejercicio No 1**

Desarrollar un programa que aplique los conocimientos aprendidos en el segundo parcial, principalmente “Listas Circulares Doblemente enlazadas” y recursividad. Utilizar memoria dinámica. Se deben eliminar los cuadrados del mismo color si se juntan.

1. **Marco Teórico**

El Tetris, uno de los juegos de computadora más populares del mundo a lo largo de la historia, fue creado el 6 de junio de 1985 por el inventor Alexei Pajitnov, quien entonces trabajaba en la Academia Soviética de la Ciencia, en Moscú. Su juego, que consistía básicamente en ir encajando piezas de diferentes formas y tamaños que caen desde la parte superior de la pantalla para completar un muro sin dejar huecos, fue lanzado al mercado durante la Perestroika, el plan diseñado por Mijaíl Gorbachov para relanzar la economía soviética. El juego original era en una pantalla en blanco y negro, sin sonido. El campo estaba lleno de asteriscos, y los cuadrados se hacían con dos corchetes enfrentados. Se dice que, aunque fue un éxito rotundo (se vendieron más de 40 millones de copias), Pajitnov no recibió beneficios económicos de su venta; en la actualidad, vive en Estados Unidos y trabaja desarrollando nuevas versiones de su juego para la firma Nintendo.

En el 2007 fue nombrado el segundo mejor juego de todos los tiempos, sólo detrás de Súper Mario Bros. **Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**¿Como resolverlo con listas circulares doblemente enlazadas?**

Se ha pedido que los datos generados se introduzcan en una lista circular doblemente enlazada la misma que verificara si el dato ingresado es igual al anterior o siguiente de esta lista para así proceder a eliminar hasta que no quede ningún valor.

Se ha utilizado una función recursiva que verifique la lista hasta que no pueda hacer más eliminaciones, haciendo este problema eficiente.

1. **Código del proyecto**

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

\* 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 TRAFORMACIONSTRING\_H\_INCLUDED

#define TRAFORMACIONSTRING\_H\_INCLUDED

#include<string>

#include<stdlib.h>

#include <sstream>

#include<iostream>

**using** **namespace** std**;**

char**\***trasformarStringChar**(**string str**)**

**{**

char **\***c**=** **new** char**(**str**.**length**());**

c**=(**char**\*)**str**.**c\_str**();**

**return** c**;**

**}**

int**\***arregloEntero**(**string str**)**

**{**

int **\***e **=** **new** int**(**str**.**length**());**

**for(**int i**=**0**;** i**<**str**.**length**();**i**++)**

**{**

**\*(**e**+**i**)=(**int**)**str**.**at**(**i**)-**48**;**

**}**

**return** e**;**

**}**

int stringAint**(**string str**)**

**{**

**return** atoi**(**str**.**c\_str**());**

**}**

template **<**class Dato**>**

string arreglosAstring**(**Dato **\***arrayDato**,**int tam**)**

**{**

ostringstream aux**;**

**for(**int i**=**0**;**i**<**tam**;**i**++)**

**{**

aux**<<\*(**arrayDato**+**i**);**

**}**

**return** aux**.**str**();**

**}**

#endif // TRAFORMACIONSTRING\_H\_INCLUDED

#include <iostream>

#include "ListaCircularDoble.h"

#include "Nodo.h"

#include "PersonalLibrary.h"

void ListaCircularDoble**::**ingresoInicio**(**int obj**)**

**{**

Nodo **\***p **=** **new** Nodo**();**

p**->**setDatos**(**obj**);**

**if** **(**primero **==** **NULL)**

**{**

p**->**setSiguiente**(**p**);**

p**->**setAnterior**(**p**);**

primero **=** p**;**

**}**

**else**

**{**

Nodo**\*** ultimo **=** primero**->**getAnterior**();**

p**->**setSiguiente**(**primero**);**

p**->**setAnterior**(**ultimo**);**

primero**->**setAnterior**(**p**);**

ultimo**->**setSiguiente**(**p**);**

primero **=** p**;**

**}**

//cout<<"Ingresado al inicio"<<endl;

**}**

void ListaCircularDoble**::**ingresoFinal**(**int obj**)**

**{**

Nodo **\***p **=** **new** Nodo**();**

p**->**setDatos**(**obj**);**

**if** **(**primero **==** **NULL)**

**{**

ingresoInicio**(**obj**);**

**}**

**else**

**{**

Nodo**\*** ultimo **=** primero**->**getAnterior**();**

p**->**setSiguiente**(**primero**);**

p**->**setAnterior**(**ultimo**);**

primero**->**setAnterior**(**p**);**

ultimo**->**setSiguiente**(**p**);**

// cout<<"Ingresado al final"<<endl;

**}**

**}**

void ListaCircularDoble**::**imprimirDatos**()**

**{**

int x**=**8**,**y**;**

Nodo**\*** Aux**=new** Nodo**();**

Aux**=**primero**;**

**do{**

y**=**5**;**

**switch(**Aux**->**getDatos**())**

**{**

**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%c \n"**,**220**,**219**);**gotoxy**(**x**,**y**++);**

printf**(**"%c%c%c%c%c"**,**219**,**220**,**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**,**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**);**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%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**->**getSiguiente**();**

**}while(**Aux**!=**primero**);**

**}**

int ListaCircularDoble**::**tamLista**()**

**{**

int i**=**0**;**

**if(!**vacia**())**

**{**

Nodo **\***p **=** primero**;**

**do**

**{**

cout **<<** p**->**getDatos**()<<**" "**;**

p **=** p**->**getSiguiente**();**

i**++;**

**}**

**while** **(**p **!=** primero**);**

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

**}else{**

cout**<<**"Lista vacia"**<<**endl**;**

**}**

**return** i**;**

**}**

ListaCircularDoble**::**ListaCircularDoble**(){**

primero **=** **NULL;**

**}**

void ListaCircularDoble**::**destruir**(){**

Nodo **\***p **=** primero**->**getSiguiente**();**

Nodo **\***actual**;**

**while(** p **!=** primero**){**

actual **=** p**;**

p **=** p**->**getSiguiente**();**

**delete** actual**;**

**}**

**delete** primero**;**

**}**

void ListaCircularDoble**::**setPrimero**(**Nodo **\***newPrimero**){**

primero **=** newPrimero**;**

**}**

Nodo**\*** ListaCircularDoble**::**getPrimero**(){**

**return** primero**;**

**}**

bool ListaCircularDoble**::**vacia**()**

**{**

**if** **(**primero **==** **NULL)**

**return** **true;**

**else**

**return** **false;**

**}**

int ListaCircularDoble**::**cantidad**()**

**{**

int cant **=** 0**;**

**if** **(!**vacia**())**

**{**

Nodo**\*** reco **=** primero**;**

**do** **{**

cant**++;**

reco **=** reco**->**getSiguiente**();**

**}while** **(**reco **!=** primero**);**

**}**

**return** cant**;**

**}**

void ListaCircularDoble**::**borrarPosicion**(**int pos**)**

**{**

**if** **(**pos **<=** cantidad**())**

**{**

**if** **(**pos **==** 1**)**

**{**

**if** **(**cantidad**()** **==** 1**)**

**{**

**delete** primero**;**

primero **=** **NULL;**

//cout<<"Elemento eliminado en la posicion "<<pos<<endl;

**}**

**else**

**{**

Nodo**\*** bor **=** primero**;**

Nodo**\*** ultimo **=** primero**->**getAnterior**();**

primero **=** primero**->**getSiguiente**();**

ultimo**->**setSiguiente**(**primero**);**

primero**->**setAnterior**(**ultimo**);**

**delete** bor**;**

// cout<<"Elemento eliminado en la posicion "<<pos<<endl;

**}**

**}**

**else**

**{**

Nodo**\*** reco **=** primero**;**

**for** **(**int f **=** 1**;** f **<=** pos **-** 1**;** f**++){**

reco **=** reco**->**getSiguiente**();**

**}**

Nodo**\*** bor **=** reco**;**

Nodo**\*** anterior **=** reco**->**getAnterior**();**

reco **=** reco**->**getSiguiente**();**

anterior**->**setSiguiente**(**reco**);**

reco**->**setAnterior**(**anterior**);**

**delete** bor**;**

//cout<<"Elemento eliminado en la posicion "<<pos<<endl;

**}**

**}**

**}**

void ListaCircularDoble**::**recursivo**(**Nodo **\***p**,** int posicion**,** bool bandera**,**int**&**puntos**){**

p **=** primero**;**

posicion **=** 1**;**

**do**

**{**

**if(**cantidad**()** **!=** 1**)**

**{**

**if(**primero **==** **NULL)**

**{**

bandera **=** **false;**

**break;**

**}**

**else**

**{**

**if(**p**->**getDatos**()** **==** p**->**getSiguiente**()->**getDatos**())**

**{**

borrarPosicion**(**posicion**);**

borrarPosicion**(**posicion**);**

puntos**=**puntos**+**25**;**

bandera **=** **true;**

**break;**

**}**

**if(**p**->**getDatos**()** **==** p**->**getAnterior**()->**getDatos**())**

**{**

borrarPosicion**(**posicion**);**

**if(**posicion **==** 1**)**

**{**

borrarPosicion**(**cantidad**());**

**}**

**else**

**{**

borrarPosicion**(**posicion**-**1**);**

**}**

puntos**=**puntos**+**25**;**

bandera **=** **true;**

**break;**

**}**

bandera **=** **false;**

**}**

bandera **=** **false;**

**}**

bandera **=** **false;**

posicion**++;**

p **=** p**->**getSiguiente**();**

**}**

**while(**p **!=** primero**);**

**if(!**bandera**){**

**return;**

**}else{**

recursivo**(**p**,**posicion**,**bandera**,**puntos**);**

**}**

**}**

void ListaCircularDoble**::**borrarIguales**(**int **&**puntos**)**

**{**

Nodo **\***p **=** **new** Nodo**();**

p **=** primero**;**

int posicion **=** 1**;**

recursivo**(**p**,**posicion**,false,**puntos**);**

**}**

void ListaCircularDoble**::**insertarPosicion**(**int obj**,** int pos**)**

**{**

**if** **(**pos **<=** cantidad**())**

**{**

**if** **(**pos **==** 1**)**

**{**

ingresoInicio**(**obj**);**

**}**

**else**

**{**

Nodo**\*** reco **=** primero**;**

**for** **(**int f **=** 1**;** f **<** pos **-** 1**;** f**++){**

reco **=** reco**->**getSiguiente**();**

**}**

Nodo**\*** nuevo **=** **new** Nodo**();**

nuevo**->**setDatos**(**obj**);**

nuevo**->**setSiguiente**(**reco**->**getSiguiente**());**

nuevo**->**setAnterior**(**reco**);**

reco**->**getSiguiente**()->**setAnterior**(**nuevo**);**

reco**->**setSiguiente**(**nuevo**);**

**}**

**}else{**

**if(**pos**>=**cantidad**()){**

ingresoFinal**(**obj**);**

**}**

**}**

**}**

int ListaCircularDoble**::**buscarPosicion**(**int posicion**)**

**{**

int cont **=** 1**;**

Nodo **\***p **=** primero**;**

**do{**

**if(**cont **==** posicion**){**

**return** p**->**getDatos**();**

**}else{**

p **=** p**->**getSiguiente**();**

**}**

cont**++;**

**}while(**p **!=** primero**);**

**return** 0**;**

**}**

#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

#include "Nodo.h"

#include <stdlib.h>

int Nodo**::**getDatos**(**void**)**

**{**

**return** datos**;**

**}**

void Nodo**::**setDatos**(**int newDatos**)**

**{**

datos **=** newDatos**;**

**}**

Nodo **\*** Nodo**::**getSiguiente**(**void**)**

**{**

**return** siguiente**;**

**}**

void Nodo**::**setSiguiente**(**Nodo**\*** newSiguiente**)**

**{**

siguiente **=** newSiguiente**;**

**}**

Nodo**::**Nodo**()**

**{**

datos **=** **NULL;**

siguiente **=** **NULL;**

anterior **=** **NULL;**

**}**

void Nodo**::**setAnterior**(**Nodo**\*** newAnterior**)**

**{**

anterior **=** newAnterior**;**

**}**

Nodo **\*** Nodo**::**getAnterior**(**void**)**

**{**

**return** anterior**;**

**}**

#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

#include "Piezas.h"

#include "miniwin.h"

**using** **namespace** miniwin**;**

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**;**

**}**

void Piezas**::**setX**(**int newX**)**

**{**

x **=** newX**;**

**}**

int Piezas**::**getY**(**void**)**

**{**

**return** y**;**

**}**

void Piezas**::**setY**(**int newY**)**

**{**

y **=** newY**;**

**}**

int Piezas**::**cubo1**(**void**)**

**{**

pieza**(**getX**(),**getY**(),**BLANCO**);**

**return** 1**;**

**}**

int Piezas**::**cubo2**(**void**)**

**{**

pieza**(**getX**(),**getY**(),**ROJO**);**

**return** 2**;**

**}**

int Piezas**::**cubo3**(**void**)**

**{**

pieza**(**getX**(),**getY**(),**AZUL**);**

**return** 3**;**

**}**

int Piezas**::**cubo4**(**void**)**

**{**

pieza**(**getX**(),**getY**(),**AMARILLO**);**

**return** 4**;**

**}**

int Piezas**::**cubo5**(**void**)**

**{**

pieza**(**getX**(),**getY**(),**VERDE**);**

**return** 5**;**

**}**

int Piezas**::**cubo6**(**void**)**

**{**

pieza**(**getX**(),**getY**(),**MAGENTA**);**

**return** 6**;**

**}**

int Piezas**::**cubo7**(**void**)**

**{**

pieza**(**getX**(),**getY**(),**NEWCOLOR**);**

**return** 7**;**

**}**

int Piezas**::**cubo8**(**void**)**

**{**

pieza**(**getX**(),**getY**(),**VERDECLARO**);**

**return** 8**;**

**}**

int Piezas**::**cubo9**(**void**)**

**{**

pieza**(**getX**(),**getY**(),**CELESTE**);**

**return** 9**;**

**}**

int Piezas**::**vacio**(**void**)**

**{**

pieza**(**getX**(),**getY**(),**NEGRO**);**

**return** 0**;**

**}**

int Piezas**::**getNum**(**void**)**

**{**

**return** num**;**

**}**

void Piezas**::**setNum**(**int n**)**

**{**

num**=**n**;**

**}**

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;**

**}**

**}**

#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

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <conio.h>

#include <windows.h>

#include <ctype.h>

#include <string.h>

#include <fstream>

#include <string>

#include <time.h>

#include <pthread.h>

#include <fstream>

**using** **namespace** std**;**

#define TECLA\_ARRIBA 72

#define TECLA\_ABAJO 80

#define TECLA\_DERECHA 77

#define TECLA\_IZQUIERDA 75

#define TECLA\_ENTER 13

void gotoxy**(**int x**,** int y**)**

**{**

HANDLE hCon**;**

hCon**=**GetStdHandle**(**STD\_OUTPUT\_HANDLE**);**

COORD dwPos**;**

dwPos**.**X**=**x**;**

dwPos**.**Y**=**y**;**

SetConsoleCursorPosition**(**hCon**,**dwPos**);**

**}**

void color**(**int x**)**

**{**

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

**}**

int menu**(**const char **\***titulo**,**const char **\***opciones**[],** int numero**)**

**{**

bool repite**=true;**

int seleccionar**=**1**,**tecla**;**

int i**;**

system**(**"cls"**);**

**do**

**{**

gotoxy**(**10**,**2**);**

color**(**158**);**

printf**(**"================================================"**);**

color**(**11**);**

printf**(**"\n\t\t\t%s\t\t\t\n\n"**,**titulo**);**

color**(**15**);** //letras en blanco

**for(**i**=**0**;**i**<**numero**;**i**++)**

printf**(**"\t\t%s\n"**,**opciones**[**i**]);**

gotoxy**(**16**,**4**+**seleccionar**);** //17

color**(**249**);**

//printf("%c %s",175,opciones[seleccionar-1]); //impresion con asterisco

printf**(**"%s"**,**opciones**[**seleccionar**-**1**]);**

color**(**158**);**

gotoxy**(**10**,**15**);**

printf**(**"================================================"**);**

color**(**15**);**

**do**

**{**

tecla**=**getch**();**

**}while(**tecla**!=**TECLA\_ARRIBA **&&** tecla**!=**TECLA\_ABAJO **&&** tecla**!=** TECLA\_ENTER **&&** tecla **!=**00**);**

**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;**

**case** 0**:**

ShellExecute**(NULL,** TEXT**(**"open"**),**TEXT**(**"Extras\\Ayuda.chm"**),NULL,** **NULL,**SW\_SHOWNORMAL**);**

**break;**

**}**

**}while(**repite**);**

color**(**15**);**

**return** seleccionar**;**

**}**

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**);**

**}**

void imprimirTXT**(**char txt**[])**

**{**

string frase**;**

ifstream ficheroEntrada**;**

color**(**10**);**

ficheroEntrada**.**open**(**txt**);**

**while** **(!**ficheroEntrada**.**eof**()** **)**

**{**

getline**(**ficheroEntrada**,** frase**);**

cout **<<** frase **<<** endl**;**

**}**

ficheroEntrada**.**close**();**

**}**

bool validacionCaracter**(**char cadena**[]){**

int i**;**

**for(**i**=**0**;**cadena**[**i**]!=**'\0'**;**i**++)**

**{**

**if((**cadena**[**i**]<**65**||**cadena**[**i**]>**90**)&&(**cadena**[**i**]<**97**||**cadena**[**i**]>**122**)&&**cadena**[**i**]!=**32**&&**cadena**[**i**]!=-**92**&&**cadena**[**i**]!=-**91**)**

**{**

printf**(**"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\nDato Incorrecto !!\nNo puede contener caracteres especiales, ni numeros\nVuelva a ingresar por favor\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n"**);**

**return** **true;**

**break;**

**}**

**}**

**return** **false;**

**}**

void guardarNombre**(**string nombre**)**

**{**

ofstream arch**;**

arch**.**open**(**"Nombre.txt"**,**ios**::**out**);**

arch**<<**nombre**;**

arch**.**close**();**

**}**

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

**{**

**switch(**numero**){**

**case** 0**:**

color**(**1**);**

gotoxy**(**x**,**y**++);**

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

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

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

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

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

**break;**

**case** 1**:**

color**(**2**);**

gotoxy**(**x**,**y**++);**

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

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

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

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

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

**break;**

**case** 2**:**

color**(**3**);**

gotoxy**(**x**,**y**++);**

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

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

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

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

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

**break;**

**case** 3**:**

color**(**4**);**

gotoxy**(**x**,**y**++);**

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

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

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

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

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

**break;**

**case** 4**:**

color**(**6**);**

gotoxy**(**x**,**y**++);**

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

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

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

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

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

**break;**

**case** 5**:**

color**(**7**);**

gotoxy**(**x**,**y**++);**

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

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

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

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

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

**break;**

**case** 6**:**

color**(**12**);**

gotoxy**(**x**,**y**++);**

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

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

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

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

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

**break;**

**case** 7**:**

color**(**9**);**

gotoxy**(**x**,**y**++);**

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

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

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

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

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

**break;**

**case** 8**:**

color**(**10**);**

gotoxy**(**x**,**y**++);**

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

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

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

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

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

**break;**

**case** 9**:**

color**(**11**);**

gotoxy**(**x**,**y**++);**

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

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

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

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

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

**break;**

**}**

**}**

string lectura**()**

**{**

ifstream archivo**;**

string texto**;**

archivo**.**open**(**"Datos.txt"**,**ios**::**in**);**

**if(**archivo**.**fail**())**

**{**

exit**(**1**);**

**}**

**else{**

**while(!**archivo**.**eof**())**

**{**

getline**(**archivo**,**texto**);**

**}**

**}**

archivo**.**close**();**

**return** texto**;**

**}**

void codificar**(**string **&**nombre**,**int i**)**

**{**

**do**

**{**

**if(**i**%**2**==**0**)**

**{**

nombre**.**at**(**i**)=(**nombre**.**at**(**i**)+**6**)+**nombre**.**length**();**

**}**

**else{**

nombre**.**at**(**i**)=(**nombre**.**at**(**i**)+**3**)-**nombre**.**length**();**

**}**

i**++;**

**}while(**i**!=**nombre**.**length**());**

**}**

void decodificar**(**string **&**nombre**,**int i**)**

**{**

**do**

**{**

**if(**i**%**2**==**0**)**

**{**

nombre**.**at**(**i**)=(**nombre**.**at**(**i**)-**6**)-**nombre**.**length**();**

**}**

**else{**

nombre**.**at**(**i**)=(**nombre**.**at**(**i**)-**3**)+**nombre**.**length**();**

**}**

i**++;**

**}while(**i**!=**nombre**.**length**());**

**}**

void archivoC**(**string nombre**)**

**{**

ofstream archivo**;**

archivo**.**open**(**"codificar.txt"**,**ios**::**out**);**

**if(**archivo**.**fail**())**

**{**

cout**<<**"No se pudo Abrir"**<<**endl**;**

exit**(**1**);**

**}**

**else{**

archivo**<<**nombre**;**

**}**

archivo**.**close**();**

**}**

void archivoD**(**string nombre**)**

**{**

ofstream archivo**;**

archivo**.**open**(**"decodificar.txt"**,**ios**::**out**);**

**if(**archivo**.**fail**())**

**{**

cout**<<**"No se pudo Abrir"**<<**endl**;**

exit**(**1**);**

**}**

**else{**

archivo**<<**nombre**;**

**}**

archivo**.**close**();**

**}**

#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

#ifndef MANEJOMEMORIA\_H\_INCLUDED

#define MANEJOMEMORIA\_H\_INCLUDED

#include<iostream>

#include <conio.h>

#include<time.h>

#include <stdio.h>

#include<fstream>

**using** **namespace** std**;**

int **\*\***reservar**(**int fila**,**int colum**)**

**{**

int **\*\***m**,**j**;**

m**=new** int**\*[**fila**];**//reservando memoria de filas

**for(**int i**=**0**;**i**<**fila**;**i**++)**

**{**

m**[**i**]=new** int**[**colum**];**//reservar columnas

**}**

**return** m**;**

**}**

void agregar**(**int **\*\***matriz**,**int tam**)**

**{**

**for(**int i**=**0**;**i**<**tam**;**i**++)**

**{**

**for(**int j**=**0**;**j**<**tam**;**j**++)**

**{**

cout**<<**i**+**1**<<**" "**<<**j**+**1**<<**endl**;**

cin**>>\*(\*(**matriz**+**i**)+**j**);**

**}**

**}**

**}**

void mostrar**(**int**\*\***matriz**,**int fila**,**int columna**)**

**{**

**for(**int i**=**0**;**i**<**fila**;**i**++)**

**{**

**for(**int j**=**0**;**j**<**columna**;**j**++)**

**{**

cout**<<\*(\*(**matriz**+**i**)+**j**);**

**}**

cout**<<**endl**;**

**}**

**}**

void liberarMemoria**(**int **\*\***matriz**,**int columna**)**

**{**

**for(**int i**=**0**;**i**<**columna**;**i**++)**

**{**

**delete[]** matriz**[**i**];**

**}**

**delete** **[]**matriz**;**

**}**

void encerar**(**int **\*\*** m**,**int fila**,** int columna**)**

**{**

**for(**int i**=**0**;**i**<**fila**;**i**++)**

**{**

**for(**int j**=**0**;**j**<**columna**;**j**++)**

**{**

**\*(\*(**m**+**i**)+**j**)=**0**;**

**}**

**}**

**}**

void agregarPosicion**(**int**\*\***matriz**,**int x**,** int y**,**int valor**)**

**{**

**\*(\*(**matriz**+**y**)+**x**)=**valor**;**

**}**

void escribirArchivo**(**int **\*\***guardar**,**int x **,** int y**)**

**{**

ofstream archivo**;**

archivo**.**open**(**"MovimientoTetris.csv"**,**ios**::**out **|** ios**::**app**);**

**if(**archivo**.**fail**())**

**{**

cout**<<**"No se pudo Abrir"**<<**endl**;**

exit**(**1**);**

**}**

**else{**

**for(**int i**=**0**;**i**<**x**;**i**++)**

**{**

**for(**int j**=**0**;**j**<**y**;**j**++)**

**{**

archivo**<<\*(\*(**guardar**+**i**)+**j**)<<**" \t"**;**

**}**

archivo**<<**" \n"**;**

archivo**<<**" \n"**;**

**}**

**}**

archivo**<<**" \n"**;**

archivo**<<**" \n"**;**

archivo**<<**" \n"**;**

archivo**<<**" \n"**;**

archivo**.**close**();**

**}**

void archivoJuego**(**string nombre**,**int nivel**,**int puntos**)**

**{**

ofstream archivo**;**

archivo**.**open**(**"Datos.txt"**,**ios**::**out**);**

**if(**archivo**.**fail**())**

**{**

cout**<<**"No se pudo Abrir"**<<**endl**;**

exit**(**1**);**

**}**

**else{**

archivo**<<**nombre**<<**" "**<<**nivel**<<**" "**<<**" "**<<**puntos**;**

**}**

archivo**.**close**();**

**}**

#endif // MANEJOMEMORIA\_H\_INCLUDED

#include <stdio.h>

#include <string.h>

#include <windows.h> // Win32Api Header File

#define \_CRT\_SECURE\_NO\_DEPRECATE

#include <stdio.h>

#include <iostream>

#include <string>

#include<fstream>

#include <iostream>

#include <conio.h>

#include <string>

#include <locale.h>

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <time.h>

#include <ctype.h>

#define KEY\_ESC 27

**using** **namespace** std**;**

static HWND hConWnd**;**

HWND BCX\_Bitmap**(**char**\*,** HWND **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**);**

HWND GetConsoleWndHandle**(**void**);**

int main**()**

**{**

hConWnd **=** GetConsoleWndHandle**();**

**if** **(**hConWnd**)**

**{**

// select a bitmap file you have or use one of the files in the Windows folder

// filename, handle, ID, ulcX, ulcY, width, height 0,0 auto-adjusts

BCX\_Bitmap**(**"LOGO.bmp"**,** hConWnd**,** 235**,** 6**,** 6**);**

Sleep**(**1000**)** **;** // wait

**}**

**return** 0**;**

**}**

// draw the bitmap

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**;**

**}**

// 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**(**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**;**

**}**

#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**]** **==** ' '**){**

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**;**

**}**

#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**);**

encerar**(**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 cordenadas x y

t**=**tecla**();**

cout**<<**endl**;**

refresca**();**

bandera**=true;**

**while(**y**!=**13 **&&** bandera**==true)**

**{**

//sale de mi bucle cuado bandera=pieza toca la otra pieza y y sea ==19= tama񯠤el 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 patalla

**{**

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**;**

**}**

#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**;**

**}**

1. **Conclusiones**

Las listas circulares doblemente enlazadas han sido de gran ayuda para el proyecto pues es más fácil comparar tanto para el nodo anterior como al siguiente. Así mismo la recursividad sigue siendo una forma de resolver este problema de una forma más sencilla.