|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería de Software | **GUÍA**  No. 02 | **TIEMPO ESTIMADO:**  1 mes |
| **ASIGNATURA:**  Estructura de datos  NRC: 2967 | **FECHA DE ELABORACION:** 23-10-2019  **SEMESTRE**: septiembre 2019 – febrero 2020 | |
| **TÍTULO:**  Backtracking | **DOCENTE:** Ing. Fernando Solis | |

**OBJETIVO**

Comprender el uso de e implementación tanto en teoría como en código de listas simples, listas dobles, listas doblemente circulares, para crear un pequeño juego con listas doblemente circulares.

**INSTRUCCIONES**

La lista circular doble es una especie de lista enlazada “doblemente enlazada”, pero que posee una característica adicional para el desplazamiento dentro de la lista, “ésta no tiene fin” y tiene 2 apuntadores a si misma.

Para que la lista sea sin fin, el puntero siguiente del último elemento apuntará hacia el 1er elemento y el puntero anterior del primer elemento apuntara hacia el ultimo elemento de la lista en lugar de apuntar al valor NULL, como hemos visto en el caso de listas enlazadas simples o doblemente enlazadas.

En las listas circulares dobles, nunca se llega a una posición en la que ya no sea posible desplazarse.

Cuando se llegue al último elemento, el desplazamiento volverá a comenzar desde el primer elemento.

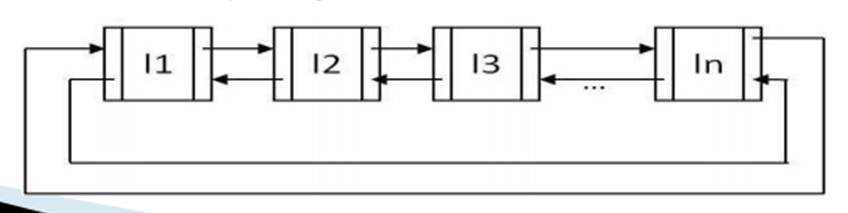
Los operaciones básicas de una lista circular doble son:

· Insertar: inserta un nodo con dato x en la lista, pudiendo realizarse esta inserción al principio o final de la lista o bien en orden.

· Eliminar: elimina un nodo de la lista, puede ser según la posición o por el dato.

· Buscar: busca un elemento en la lista.

· Localizar: obtiene la posición del nodo en la lista.

·Imprimir: imprime los elementos de la lista.

Ejemplo de lista enlazada doble circular

Un ejemplo donde se almacenan números y vemos como la estructura donde el último

**ACTIVIDADES**

1. **Ubicación de recursos**
2. El grupo está conformado por Angel Cárdenas y Edison Báez.
3. Se utilizó DEV C++ para la realización.
4. **Planteamiento del problema**

Realizar un juego, un tetris específicamente, con la ayuda de las listas enlazadas doblemente circulares.

1. **Entregable (s)**

**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++) { b[i] = a[i]; a[i].y += 1; }

if (!check())

{

for (int i = 0; i < 4; i++) field[b[i].y][b[i].x] = colorNum;

colorNum = 1 + 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];

}

k--;

}

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;

}

**Aplicativo.cpp**

#include "ClaseNodo.h"

#include "iostream"

int main()

{

Nodo Obj;

Nodo \*Cabecera;

Cabecera=NULL;

int num;

int opc=0;

do

{

system("cls");

printf("\n\t\t\t\t UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE\n");

printf("\t\t\t\t\t\tESTRUCTURA DE DATOS\n");

printf("\nMENU\n");

printf("====\n");

std::cout<<"1.-Insertar\n";

std::cout<<"2.-Verificar si la lista esta vacia\n";

std::cout<<"3.-Imprimir lista\n";

std::cout<<"4.-Salir\n";

std::cout<<"Ingrese la opcion: ";

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);

}

else{

std::cout<<"La lista no esta vacia y es: \n";

Obj.Imprimir(Cabecera);

}

system("pause");

break;

case 3:

printf("\nLos datos ingresados son: \n");

Obj.Imprimir(Cabecera);

system("pause");

break;

case 4:

system("pause");

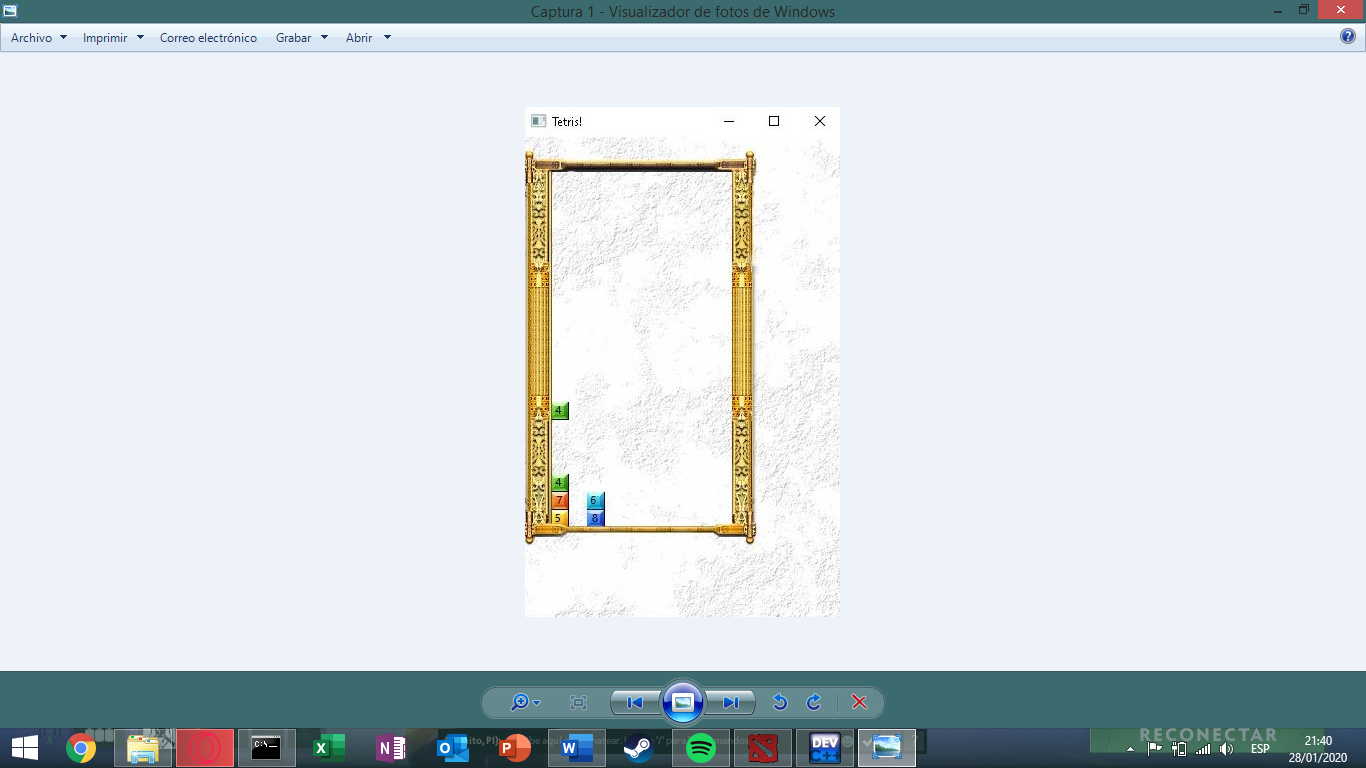
break;

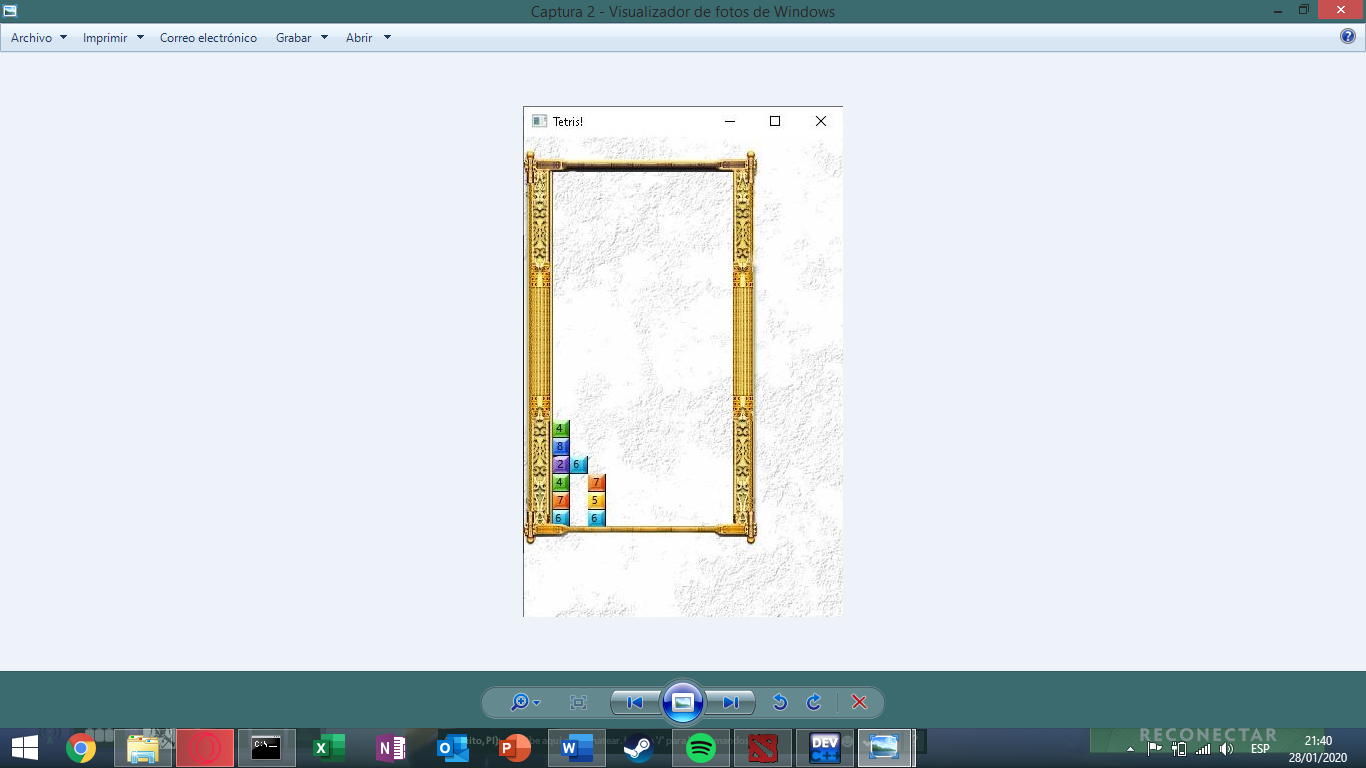
}

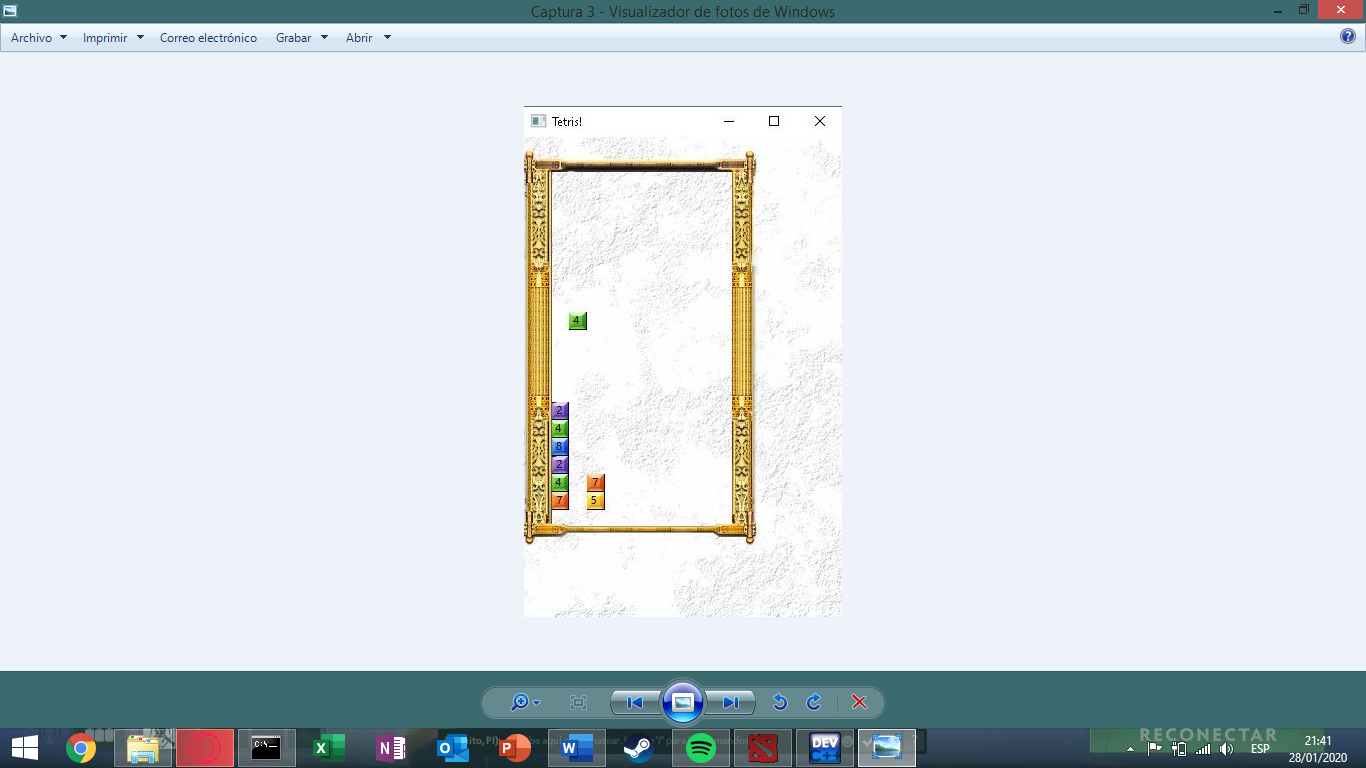
}while(opc!=4);

}

1. **Ejecución**







1. **Conclusiones**

Después del respectivo proyecto se pudo comprender de mejor manera el concepto de listas simples, listas dobles, listas doblemente circulares. Se aprendió a utilizar de manera los algoritmos de listas doblemente circulares y un uso efectivo de la recursividad