|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería en Sistemas e informática | **GUÍA**  No. 03 | **TIEMPO ESTIMADO:** |
| **ASIGNATURA:**  Estructura de Datos  NRC: 2969 | **FECHA DE ELABORACION:** dd-mm-aa  **SEMESTRE**: Octubre 2019 – Enero 2020 | |
| **TÍTULO:**  Proyecto Parcial | **DOCENTE: Fernando Solis** | |

**OBJETIVO**

p.e. Aplicar los conocimientos adquiridos en el tercer parcial en un proyecto.

**INSTRUCCIONES**

**p.e.**

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**

**p.e.**

1. Formar grupos de máximo 2 personas
2. Instalar el IDE para C++
3. **Planteamiento del problema**

**p.e. Ejercicio No 1**

Desarrollar un programa que aplique los conocimientos aprendidos en el tercer parcial.

Este proyecto consiste en un juego tipo “snake”, que su comida son números y al comerlos estos se van insertando en un árbol binario, al finalizar el árbol se muestra y se genera un pdf.

**Marco Teórico**

**Árbol Binario**

Es una estructura de datos en la cual cada nodo puede tener un hijo izquierdo y un hijo derecho. No pueden tener más de dos hijos (de ahí el nombre "binario"). Si algún hijo tiene como referencia a null, es decir que no almacena ningún dato, entonces este es llamado un nodo externo. En el caso contrario el hijo es llamado un nodo interno.

**Snake**

El Snake (a veces también llamado la serpiente) es un videojuego lanzado a mediados de la década de los 70 mantenido su popularidad desde entonces, convirtiéndose en un clásico. En 1998, el Snake obtuvo una audiencia masiva tras convertirse en un juego estándar pre-grabado en los teléfonos Nokia.

En el juego, el jugador o usuario controla una larga y delgada criatura, semejante a una serpiente, que vaga alrededor de un plano delimitado, recogiendo alimentos (o algún otro elemento), tratando de evitar golpear a su propia cola o las "paredes" que rodean el área de juego. Cada vez que la serpiente se come un pedazo de comida, la cola crece más, provocando que aumente la dificultad del juego. El usuario controla la dirección de la cabeza de la serpiente (arriba, abajo, izquierda o derecha) y el cuerpo de la serpiente la sigue. Además, el jugador no puede detener el movimiento de la serpiente, mientras que el juego está en marcha.

**Conclusiones**

* Se puede visualizar de manera visual el arbol binario.
* Se logra deducir la forma en la cual se organiza un arbol binario y como se ordenan los numeros insertados.
* Los arboles binarios son estructuras de datos muy avanzadas

**Codigo**

#pragma once

#include <iostream>

using *namespace* std;

/\*--------- Tree Structure ------------\*/

typedef *class* Node {

*public:*

*int* number;

Node\* left, \* right;

}\*ABB;

/\*----------------------------------------\*/

ABB createNode(*int* *x*) {

ABB newNode = new(Node);

newNode->number = x;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

*void* insert(ABB& *tree*, *int* *x*) {

if (tree == NULL) {

tree = createNode(x);

}

else if (x < tree->number)

insert(tree->left, x);

else if (x > tree->number)

insert(tree->right, x);

}

#include <SFML/Graphics.hpp>

#include <SFML//Graphics/Text.hpp>

#include <Windows.h>

#include <time.h>

#include <conio.h>

#include <fstream>

#include "BinaryTree.h"

#include "UtilMenu.h"

#define size 20

#define N 30

#define M 20

#define UP 72

#define DOWN 80

#define ENTER 13

#define ESCAPE 27

#define LEFT 75

#define RIGHT 77

using *namespace* sf;

///// variables ///////

ABB tree = NULL;

*int* dir, num = 4;

*int* auxX = 0; //for printing

*bool* GAMEOVER = false;

//// classes ///

*class* Snake {

*public:*

*int* x, y;

}s[100];

*class* Fruit {

*public:*

*int* x, y;

*int* number; //random from 1 to 50

}f;

//////////////

*void* Tick() {

////// next state /////

for (*int* i = num; i > 0; --i) {

s[i].x = s[i - 1].x; s[i].y = s[i - 1].y;

}

///// Movement /////

if (dir == 0) s[0].y += 1;

if (dir == 1) s[0].x -= 1;

if (dir == 2) s[0].x += 1;

if (dir == 3) s[0].y -= 1;

///// Get fruit /////

if ((s[0].x == f.x) && (s[0].y == f.y)) {

insert(tree, f.number);

num++;

f.x = rand() % N;

f.y = rand() % M;

f.number = 1 + (rand() % 51);

}

///// Out of the screen /////

if (dir == 0 && s[0].y >= M) GAMEOVER = true;

if (dir == 1 && s[0].x < 0) GAMEOVER = true;

if (dir == 2 && s[0].x >= N) GAMEOVER = true;

if (dir == 3 && s[0].y < 0) GAMEOVER = true;

////// collision /////

for (*int* i = 1; i < num; i++) {

if (s[0].x == s[i].x && s[0].y == s[i].y)

GAMEOVER = true;

}

}

*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* showTree(ABB& *tree*, *int* *auxY*) {

if (tree == NULL) {

return;

}

auxX += 5;

showTree(tree->left, auxY + 2);

gotoxy(10 + auxX - auxY, 15 + auxY);

cout << tree->number << endl << endl;

showTree(tree->right, auxY + 2);

}

*int* main() {

*int* posY = 0;

menuSelecInformation();

*char* key;

while (true) {

key = \_getch();

switch (key) {

case UP:

posY--;

if (posY < 0) {

posY = 3;

}

utilMenu(posY);

break;

case DOWN:

posY++;

if (posY == 0) {

posY = 1;

}

if (posY > 3) {

posY = 0;

}

utilMenu(posY);

break;

case ENTER:

srand(time(0));

RenderWindow window(VideoMode(size \* N, size \* M), "Snake Game!");

window.setActive(false);

window.setVisible(false);

sf::Font font;

font.loadFromFile("font/arial\_narrow\_7.ttf");

sf::Text text;

sf::String string;

Texture t1, t2, t3, gameover;

t1.loadFromFile("images/white.png");

t2.loadFromFile("images/red.png");

t3.loadFromFile("images/green.png");

gameover.loadFromFile("images/gameover\_1.jpg");

Sprite sprite1(t1);

Sprite sprite2(t2);

Sprite sprite3(t3);

Sprite spriteGO(gameover);

Clock clock;

*float* timer = 0, delay = 0.1, time;

f.x = rand() % N;

f.y = rand() % M;

f.number = 1 + (rand() % 51);

*char* tecla;

Event e;

switch (posY) {

case 0:

/\*Show information about the game\*/

system("pause");

break;

case 1:

/\*PLAY NOW!\*/

window.setActive(true);

window.setVisible(true);

while (window.isOpen() && !GAMEOVER) {

time = clock.getElapsedTime().asSeconds();

clock.restart();

timer += time;

while (window.pollEvent(e)) {

if (e.type == Event::Closed)

window.close();

}

if (Keyboard::isKeyPressed(Keyboard::Left)) dir = 1;

if (Keyboard::isKeyPressed(Keyboard::Right)) dir = 2;

if (Keyboard::isKeyPressed(Keyboard::Up)) dir = 3;

if (Keyboard::isKeyPressed(Keyboard::Down)) dir = 0;

if (timer > delay) { timer = 0; Tick(); }

////// draw ///////

window.clear();

///// chess //////

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

for (*int* j = 0; j < M; j++) {

sprite1.setPosition(i \* size, j \* size); window.draw(sprite1);

}

///// snake /////

for (*int* i = 0; i < num; i++) {

sprite2.setPosition(s[i].x \* size, s[i].y \* size); window.draw(sprite2);

}

///// fruit with number/////

sprite3.setPosition(f.x \* size, f.y \* size); window.draw(sprite3);

string = to\_string(f.number);

text.setString(string);

text.setFont(font);

text.setCharacterSize(12);

text.setFillColor(Color::Black);

text.setPosition(f.x \* size, f.y \* size); window.draw(text);

window.display();

}

window.clear();

window.draw(spriteGO);

window.display();

Sleep(5000);

window.close();

/\*PDF creation

system("java -jar dist/textToPdf.jar");\*/

system("pause");

break;

case 2:

/\*Binary tree\*/

system("cls");

cout << "\t\t\tARBOL RESULTANTE" << endl;

auxX = 0;

showTree(tree, 0);

\_getch();

break;

case 3:

/\*Exit\*/

std::cout << "Gracias por jugar :)" << std::endl;

exit(0);

}

utilMenu(posY);

break;

}

}

return 0;

}

#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 << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

std::cout << "| |" << std::endl;

std::cout << "| SNAKETREE GAME |" << std::endl;

std::cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

std::cout << "| |" << 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 << "| Mostrar arbol binario |" << std::endl;

std::cout << "| Salir |" << std::endl;

std::cout << "| |" << std::endl;

std::cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

}

*void* menuSelecPlayNow() {

system("cls");

std::cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

std::cout << "| |" << std::endl;

std::cout << "| SNAKETREE GAME |" << std::endl;

std::cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

std::cout << "| |" << 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 << "| Mostrar arbol binario |" << std::endl;

std::cout << "| Salir |" << std::endl;

std::cout << "| |" << std::endl;

std::cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

}

*void* menuSelecShowTree() {

system("cls");

std::cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

std::cout << "| |" << std::endl;

std::cout << "| SNAKETREE GAME |" << std::endl;

std::cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

std::cout << "| |" << std::endl;

std::cout << "| Informacion |" << std::endl;

std::cout << "| JUGAR AHORA! |" << std::endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), GRAY\_BACKGROUND | BLUE\_TEXT);

std::cout << "| Mostrar arbol binario |" << std::endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), WHITE\_FADE\_TEXT);

std::cout << "| Salir |" << std::endl;

std::cout << "| |" << std::endl;

std::cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

}

*void* menuSelecExit() {

system("cls");

std::cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

std::cout << "| |" << std::endl;

std::cout << "| SNAKETREE GAME |" << std::endl;

std::cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

std::cout << "| |" << std::endl;

std::cout << "| Informacion |" << std::endl;

std::cout << "| JUGAR AHORA! |" << std::endl;

std::cout << "| Mostrar arbol binario |" << 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;

std::cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << std::endl;

}

*void* utilMenu(*int* *selector*) {

switch (selector) {

case 0:

menuSelecInformation();

break;

case 1:

menuSelecPlayNow();

break;

case 2:

menuSelecShowTree();

break;

case 3:

menuSelecExit();

break;

}

}