|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería de Software | **GUÍA**  No. 01 | **TIEMPO ESTIMADO:**  1 mes |
| **ASIGNATURA:**  Estructura de datos  NRC: 2967 | **FECHA DE ELABORACION:** 23-10-2019  **SEMESTRE**: septiembre 2019 – febrero 2020 | |
| **TÍTULO:**  Traductor - Árboles | **DOCENTE:** Ing. Fernando Solís | |

**OBJETIVO**

Utilizar Árboles para realizar un proyecto el que tenga códigos u algoritmos más eficientes.

**INSTRUCCIONES**

En ciencias de la computación y en informática, un árbol es un tipo abstracto de datos (TAD) ampliamente usado que imita la estructura jerárquica de un árbol, con un valor en la raíz y subárboles con un nodo padre, representado como un conjunto de nodos enlazados.

Una estructura de datos de árbol se puede definir de forma recursiva (localmente) como una colección de nodos (a partir de un nodo raíz), donde cada nodo es una estructura de datos con un valor, junto con una lista de referencias a los nodos (los hijos) , con la condición de que ninguna referencia esté duplicada ni que ningún nodo apunte a la raíz.

Alternativamente, un árbol se puede definir de manera abstracta en su conjunto como un árbol ordenado, con un valor asignado a cada nodo. Ambas perspectivas son útiles: mientras que un árbol puede ser analizado matemáticamente, realmente es representado como una estructura de datos en la que se trabaja con cada nodo por separado (en lugar de como una lista de nodos y una lista de adyacencia entre nodos, como un grafo). Mirando a un árbol como conjunto, se puede hablar del nodo padre de un nodo dado, pero en general se habla de una estructura de datos de un nodo dado que sólo contiene la lista de sus hijos sin referencia a su padre (si lo hay).

**ACTIVIDADES**

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

Elaborar un algoritmo mediante el uso de Árboles que muestre la traducción en español e ingles de palabras almacenadas en un archivo.

1. **Entregable (s)**

**Clase Arbol.h**

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

\* Module: Convinacion.h

\* Author: Luis Carvajal, Elian Llorente

\* Modified: 12 de ener de 2020 1:04:26

\* Purpose: Declaration of the class Arbol.h

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

#include <stdio.h>

#include <iostream>

#include <string>

using namespace std;

class Arbol

{

private:

int codigo;

string palabra;

public:

typedef Arbol \*arbol;

Arbol \*izq;

Arbol \*der;

void crearNodo(string, int, arbol&);

void insertar(arbol&, int, string);

void preOrden(arbol);

void postOrden(arbol);

void enOrden(arbol);

void verArbol(arbol, int);

string Buscar(arbol, int);

int Buscar(arbol, string);

string Traducir(arbol, arbol, string);

Arbol();

~Arbol();

};

**Clase Arbol.cpp**

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

\* Module: Convinacion.h

\* Author: Luis Carvajal, Elian Llorente

\* Modified: 12 de ener de 2020 1:04:26

\* Purpose: Declaration of the class Arbol.cpp

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

#include "Arbol.h"

Arbol::Arbol()

{

}

Arbol::~Arbol()

{

}

void Arbol::crearNodo(string palabra, int x, arbol& arbo)

{

arbol nuevoNodo = new(Arbol);

nuevoNodo->palabra = palabra;

nuevoNodo->codigo = x;

nuevoNodo->izq = NULL;

nuevoNodo->der = NULL;

arbo = nuevoNodo;

}

void Arbol::insertar(arbol &arbo, int x, string palabra)

{

if (arbo == NULL)

{

crearNodo(palabra, x, arbo);

}

else if (x < arbo->codigo)

insertar(arbo->izq, x, palabra);

else if (x > arbo->codigo)

insertar(arbo->der, x, palabra);

else if (x == arbo->codigo)

insertar(arbo->izq, x, palabra);

}

void Arbol::preOrden(arbol arbo)

{

if (arbo != NULL)

{

cout << arbo->codigo << " ";

preOrden(arbo->izq);

preOrden(arbo->der);

}

}

void Arbol::enOrden(arbol arbo)

{

if (arbo != NULL)

{

enOrden(arbo->izq);

cout << arbo->codigo << " ";

enOrden(arbo->der);

}

}

void Arbol::postOrden(arbol arbo)

{

if (arbo != NULL)

{

postOrden(arbo->izq);

postOrden(arbo->der);

cout << arbo->codigo << " ";

}

}

string Arbol::Buscar(arbol arbo, int cod)

{

while (arbo != NULL)

{

if (arbo->codigo == cod)

{

return arbo->palabra;

}

arbo = arbo->der;

}

return "No disponible.";

}

int Arbol::Buscar(arbol arbo, string palabra)

{

while (arbo != NULL)

{

if (arbo->palabra == palabra)

{

return arbo->codigo;

}

arbo = arbo->der;

}

return 0;

}

string Arbol::Traducir(arbol ArbolIn, arbol ArbolEs, string palabraIn)

{

int codPal;

string valCod;

codPal = Buscar(ArbolIn, palabraIn);

valCod = Buscar(ArbolEs, codPal);

return valCod;

}

**Clase Main.cpp**

#include "Arbol.h"

#include <cstdlib>

#include <stdio.h>

#include <time.h>

#include <ctime>

#include <conio.h>

#include <Windows.h>

#include <fstream>

#define ARRIBA 72

#define ABAJO 80

#define ENTER 13

typedef Arbol \*arbol;

Arbol aux;

void llenarArbol(arbol&, arbol&);

void reproducir(string);

void traducir();

void gotoxy(int x, int y);

void Recuadro(const char titulo[]);

int DesplegarMenu(const char titulo[], const char \*opciones[], int n);

int main()

{

traducir();

return 0;

}

void reproducir(string cad) {

std::ofstream audio("reproducir.vbs");

audio << "Dim SAPI" << endl;

audio << "set SAPI = CreateObject (\"sapi.spvoice\")" << endl;

audio << "SAPI.Speak" << "\"" << cad << "\"" << endl;

audio.close();

system("reproducir.vbs");

}

void llenarArbol(arbol &arbolIn, arbol &arbolEs) {

string line;

fstream archivo;

archivo.open("traductor.txt", fstream::in);

ifstream file("traductor.txt");

if (file.is\_open())

{

while (getline(file, line))

{

int id = atoi(line.substr(0, line.find(",")).c\_str());

string words = line.substr(line.find(",") + 1, line.length());

aux.insertar(arbolIn, id, words.substr(0, words.find(",")));

aux.insertar(arbolEs, id, words.substr(words.find(",") + 1, words.length()));

}

file.close();

archivo.close();

}

}

void Recuadro(const char titulo[])

{

system("cls");

system("color D");

printf("\xC9\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xBB\n");

printf("\xBA\t\t \xBA\n");

printf("\xCC\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xB9\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xBA \xBA\n");

printf("\xC8\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xBC\n");

}

void gotoxy(int x, int y)

{

COORD coord;

coord.X = x; coord.Y = y;

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleCursorPosition(hStdOut, coord);

}

int DesplegarMenu(const char titulo[], const char \*opciones[], int n)

{

int opcionSeleccionada = 1;

int tecla;

bool repite = true;

do {

Recuadro(titulo);

gotoxy(5, 3 + opcionSeleccionada); cout << "=>" << endl;

gotoxy(30, 2); cout << titulo;

for (int i = 0; i < n; ++i) {

gotoxy(10, 4 + i); cout << (i + 1) << ") " << opciones[i];

}

do {

tecla = \_getch();

} while (tecla != ARRIBA && tecla != ABAJO && tecla != ENTER);

switch (tecla) {

case ARRIBA:

opcionSeleccionada--;

if (opcionSeleccionada < 1) {

opcionSeleccionada = n;

}

break;

case ABAJO:

opcionSeleccionada++;

if (opcionSeleccionada > n) {

opcionSeleccionada = 1;

}

break;

case ENTER:

repite = false;

break;

}

} while (repite);

return opcionSeleccionada;

}

void traducir() {

string palabra, traduccion;

arbol arbolIn = NULL;

arbol arbolEs = NULL;

bool val;

int opc;

llenarArbol(arbolIn, arbolEs);

do

{

const char\* titulo = "TRADUCTOR DE PALABRAS";

const char\* opciones[] = { "Traducir de Ingles a Espaniol", "Traducir de Espaniol a Ingles", "Ayuda", "Salir" };

opc = DesplegarMenu(titulo, opciones, 4);

switch (opc)

{

case 1:

Recuadro(titulo);

gotoxy(5, 3);

cout << "Ingrese una palabra en ingles: ";

cin >> palabra;

for (int i = 0; i < palabra.length(); i++) {

palabra[i] = tolower(palabra[i]);

}

//Inglés a español.

traduccion = aux.Traducir(arbolIn, arbolEs, palabra);

gotoxy(5, 5);

cout << "Traduccion: " << traduccion << endl;

reproducir(traduccion);

gotoxy(5, 6);

system("pause");

break;

case 2:

Recuadro(titulo);

gotoxy(5, 3);

cout << "Ingrese una palabra en espaniol: ";

cin >> palabra;

for (int i = 0; i < palabra.length(); i++) {

palabra[i] = tolower(palabra[i]);

}

//Español a inglés.

traduccion = aux.Traducir(arbolEs, arbolIn, palabra);

gotoxy(5, 5);

cout << "Traduccion: " << traduccion << endl;

reproducir(traduccion);

gotoxy(5, 6);

system("pause");

break;

case 3:

Recuadro(titulo);

gotoxy(5, 3);

system("HelpTraductor.chm");

system("pause");

break;

case 4:

Recuadro(titulo);

gotoxy(5, 3);

cout << "Cerrando programa..." << endl;

system("pause");

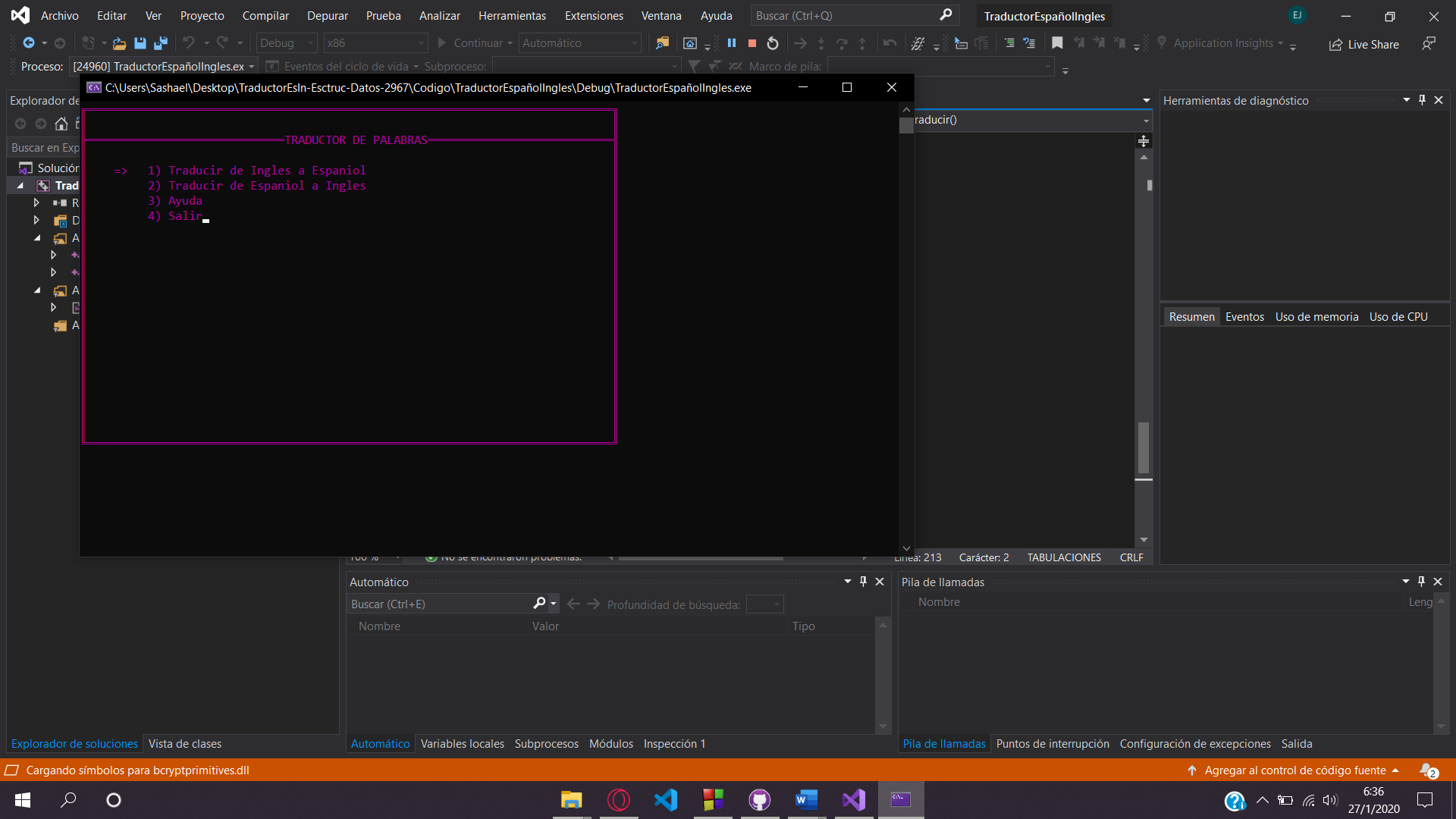
break;

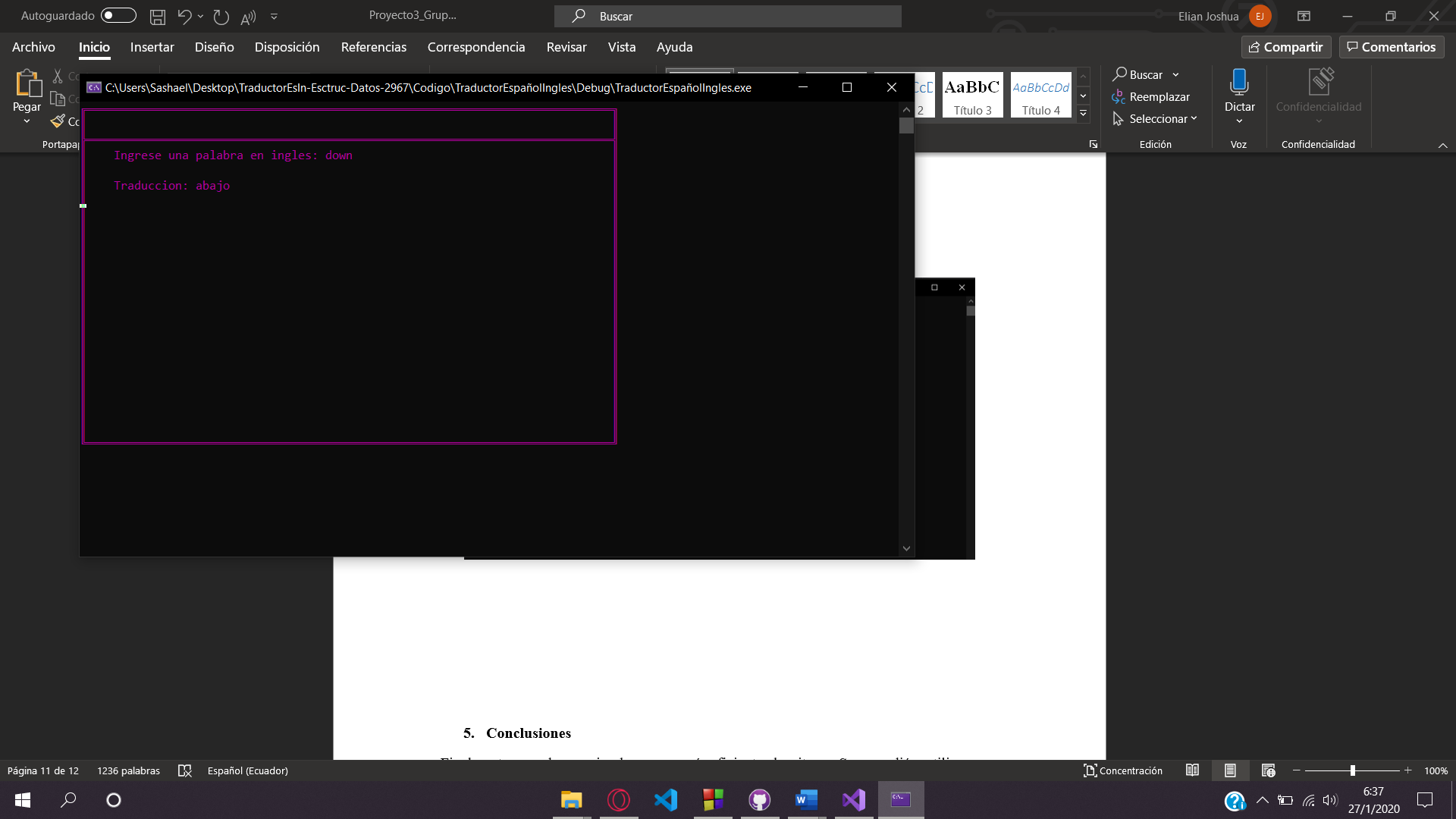
}

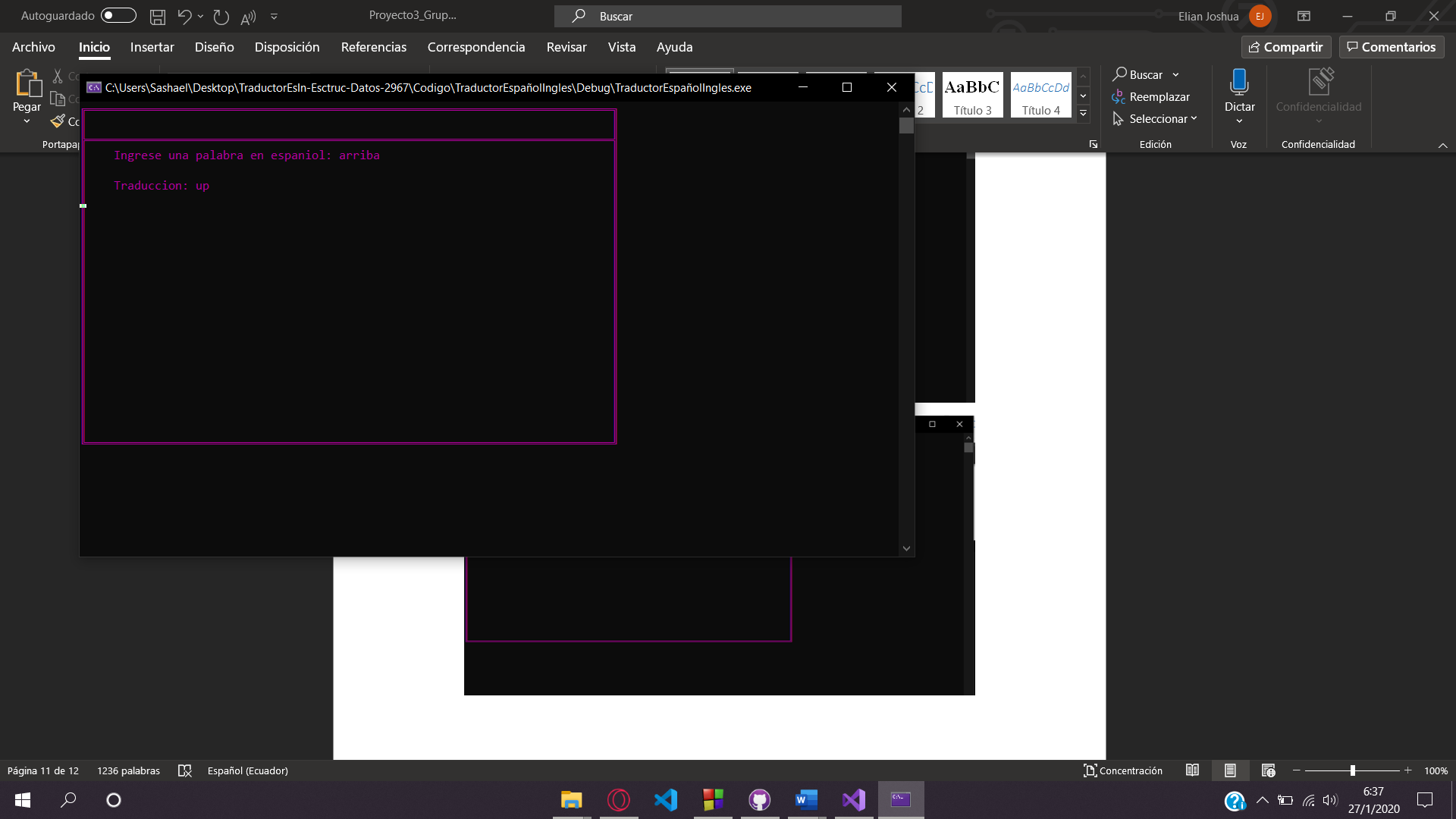
} while (opc != 4);

}

1. **Ejecución**







1. **Conclusiones**

Finalmente se pudo manejar de manera más eficiente algoritmos. Se aprendió a utilizar algoritmos que incluyen árboles binarios y entender de mejor manera como funcionan estos.

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

DOCENTE RESPONSABLE COORDINADOR DE ÁREA

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