|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería de Software | **GUÍA**  No. 03 | **TIEMPO ESTIMADO:**  1h y 20 min. |
| **ASIGNATURA:**  Estructura de datos  NRC: 2967 | **FECHA DE ELABORACION:** 26-01-2020  **SEMESTRE**: septiembre 2019 – febrero 2020 | |
| **TÍTULO:**  Arboles Binarios | **DOCENTE:** Ing. Fernando Solis | |

**OBJETIVO**

Comprender el concepto y utilización de Árboles además de sus diferentes tipos para la realización de un diccionario-traductor.

**INSTRUCCIONES**

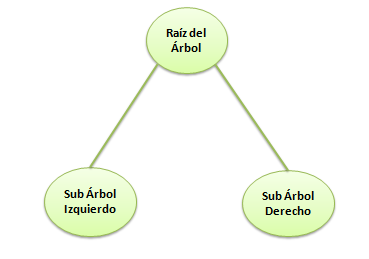
Los árboles a diferencia de las listas son una estructura de datos de no lineal, atendiendo más a una estructura de tipo jerárquico. Los árboles son, sin duda, una de las estructuras de datos no lineales, empleadas en informática, tanto para resolver problemas de hardware como de software. Los árboles de directorios son organizaciones bastante empleadas por cualquier usuario o programador de una computadora. De igual manera cumplen un buen papel en la toma de decisiones, valido como árbol de decisiones.

Definición de árboles

Los árboles binarios son estructuras de datos muy similares a las listas doblemente enlazadas, en el sentido que tienen dos punteros que apuntan a otros elementos, pero no tienen una estructura lógica de tipo lineal o secuencial como aquellas, sino ramificada. Tienen aspecto de árbol, de ahí su nombre.

Un árbol binario es una estructura de datos no lineal en la que cada nodo puede apuntar a uno o máximo a dos nodos. También se suele dar una definición recursiva que indica que es una estructura compuesta por un dato y dos árboles. Esto son definiciones simples. Este tipo de árbol se caracteriza porque tienen un vértice principal y de él se desprende dos ramas. La rama izquierda y la rama derecha a las que también se les conoce como subárboles.

Una representación gráfica de la estructura general de un árbol binario se puede visualizar a continuación.



Árbol binario

Preorden: (raíz, izquierdo, derecho). Para recorrer un árbol binario no vacío en preorden, hay que realizar las siguientes operaciones recursivamente en cada nodo, comenzando con el nodo de raíz:

Visite la raíz

Atraviese el sub-árbol izquierdo

Atraviese el sub-árbol derecho

Inorden: (izquierdo, raíz, derecho). Para recorrer un árbol binario no vacío en inorden (simétrico), hay que realizar las siguientes operaciones recursivamente en cada nodo:

Atraviese el sub-árbol izquierdo

Visite la raíz

Atraviese el sub-árbol derecho

Postorden: (izquierdo, derecho, raíz). Para recorrer un árbol binario no vacío en postorden, hay que realizar las siguientes operaciones recursivamente en cada nodo:

Atraviese el sub-árbol izquierdo

Atraviese el sub-árbol derecho

Visite la raíz

En general, la diferencia entre preorden, inorden y postorden es cuándo se recorre la raíz. En los tres, se recorre primero el sub-árbol izquierdo y luego el derecho.

En preorden, la raíz se recorre antes que los recorridos de los subárboles izquierdo y derecho

En inorden, la raíz se recorre entre los recorridos de los árboles izquierdo y derecho, y

En postorden, la raíz se recorre después de los recorridos por el subárbol izquierdo y el derecho

Preorden (antes), inorden (en medio), postorden (después).

**ACTIVIDADES**

1. **Ubicación de recursos**
2. El grupo esta conformado por Angel Cárdenas, Edison Báez
3. Se utilizo Visual Studio 2019 y Dev C
4. **Planteamiento del problema**

Realizar un proyecto que pueda traducir palabras de ingles al español y viceversa, mostrar en pantalla la traducción y además tener la capacidad de reproducir la pronunciación de la palabra.

1. **Entregable (s)**

Código Menú (DevC++)

/\* \*\*\*\*\* UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE \*\*\*\*\* \*\*

\*\* Estructura de Datos \*\*

\*\* Nombre: Edison Baez Angel Cardenas \*\*

\*\* NRC: 2742 \*\*

\*\* Fecha de realizacion: 25/12/2019 \*\*

\*\* 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 ARRIBA 72

#define ABAJO 80

#define ENTER 13

#define F1 59

using namespace std;

void portada();

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, 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] =" \*\*\*\*\*\* 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<<"\nProgramar ";

ResetConsoleColour(Attributes);

cout<<"\n CodigoQR \n Codigo de Barras \n Sello Pixeleado \n Encriptar \n Desencriptar \n Salir "<<endl;

}

// Remarca el metodo dos en la consola

void dos(WORD Attributes){

cout << endl;

cout<<"\nPrograma \n ";

SetConsoleColour(&Attributes, FOREGROUND\_BLUE | BACKGROUND\_INTENSITY | BACKGROUND\_GREEN);

cout<<"CodigoQR ";

ResetConsoleColour(Attributes);

cout<<"\n Codigo de Barras \n Sello Pixeleado \n Encriptar \n Desencriptar \n Salir"<<endl;

}

// Remarca el metodo tres en la consola

void tres(WORD Attributes){

cout << endl;

cout<<"\nProgramar \n CodigoQR \n ";

SetConsoleColour(&Attributes, FOREGROUND\_BLUE | BACKGROUND\_INTENSITY | BACKGROUND\_GREEN);

cout<<"Codigo de Barras";

ResetConsoleColour(Attributes);

cout<<"\n Sello Pixeleado \n Encriptar \n Desencriptar \n Salir";

}

// Remarca el metodo cuatro en la consola

void cuatro(WORD Attributes){

cout << endl;

cout<<"\nPrograma \n CodigoQR \n Codigo de Barras \n ";

SetConsoleColour(&Attributes, FOREGROUND\_BLUE | BACKGROUND\_INTENSITY | BACKGROUND\_GREEN);

cout<<"Sello Pixeleado";

ResetConsoleColour(Attributes);

cout << "\n Encriptar \n Desencriptar \n Salir";

}

// Remarca el metodo cinco en la consola

void cinco(WORD Attributes){

cout << endl;

cout<<"\nPrograma \n CodigoQR \n Codigo de Barras \n Sello Pixeleado \n";

SetConsoleColour(&Attributes, FOREGROUND\_BLUE | BACKGROUND\_INTENSITY | BACKGROUND\_GREEN);

cout<<"Encriptar";

ResetConsoleColour(Attributes);

cout << " \n Desencriptar \n Salir";

}

// Remarca el metodo seis en la consola

void seis(WORD Attributes){

cout << endl;

cout<<"\nPrograma \n CodigoQR \n Codigo de Barras \n Sello Pixeleado \n Encriptar \n ";

SetConsoleColour(&Attributes, FOREGROUND\_BLUE | BACKGROUND\_INTENSITY | BACKGROUND\_GREEN);

cout<<"Desencriptar";

ResetConsoleColour(Attributes);

cout << "\n Salir";

}

// Remarca el metodo siete en la consola

void siete(WORD Attributes){

cout << endl;

cout<<"\nPrograma \n CodigoQR \n Codigo de Barras \n Sello Pixeleado \n Encriptar \n Desencriptar \n";

SetConsoleColour(&Attributes, FOREGROUND\_BLUE | BACKGROUND\_INTENSITY | BACKGROUND\_GREEN);

cout<<"Salir";

ResetConsoleColour(Attributes);

}

void portada()

{

system ("color e" );

cout<<"\t\t\t\t\tBIENVENIDOS A NUESTRO PROGRAMA"<<endl;

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

cout<<"\n\t\t||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||";

cout<<"\n\t\t|||| ||||";

cout<<"\n\t\t|||| UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE ||||";

cout<<"\n\t\t|||| ESTRUCTURA DE DATOS ||||";

cout<<"\n\t\t|||| INTEGRANTES: ANGEL CARDENAS - EDISON BAEZ ||||";

cout<<"\n\t\t|||| NRC: 2967 ||||";

cout<<"\n\t\t|||| ||||";

cout<<"\n\t\t||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\n\n\n\n\n";

cout<<"\n";

system("PAUSE");

}

// funcion principal

int main()

{

system("color b");

// declaracoion de variables

portada();

WORD Attributes=0;

pthread\_t thread1;

pthread\_create(&thread1, NULL, marquesina, NULL);

int met,tecla,opc=0;

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

cout<<"\t Seleccione PORFAVOR \t\t\t\t\t\t \nPulse F1 para AYUDA"<<endl;

cout << endl;

cout<<"\nPrograma \n CodigoQR \n Codigo de Barras \n Sello Pixeleado \n Salir "<<endl;

do{

// lee la tecla que se pulsa

do{

tecla=getch();

}while(tecla!=ARRIBA && tecla!=ABAJO && tecla!=ENTER && tecla!=F1);

system("cls");

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

cout<<"\t Seleccione PORFAVOR \t\t\t\t \npulse 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 ABAJO:

opc++;

if(opc==10){

opc=1;

}

break;

case ARRIBA:

opc--;

if(opc<=0){

opc=9;

}

break;

case ENTER:

met=opc;

break;

// en caso de presionar la tecla f1 se abre una ventana de ayuda

case F1:

system("Ayuda.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;

case 6:

seis(Attributes);

break;

case 7:

siete(Attributes);

break;

}

// Cuando se pulsa enter entra al metodo que se encuentre seleccionado

switch(met){

// ejecuta jugar

case 1:

SuspendThread(&thread1);

system("ProyectoDiccionario.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 pixeleado

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;

case 5:

system("cls");

char original[FILENAME\_MAX], encriptado[FILENAME\_MAX];

char c, clave;

FILE \*entrada, \*salida;

unsigned temp;

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

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

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");

opc=0;

met=0;;

break;

case 6:

system("cls");

char secundario[FILENAME\_MAX], desencriptado[FILENAME\_MAX];

char 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");

break;

}

}while(met!=7);

// Termina el programa

system("cls");

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

cout<<"Gracias por ver el proyecto!!!"<<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 ( \_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;

}

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("sello.bmp", hConWnd, 123, 150, 150, 0, 0);

Sleep(5000);

pthread\_mutex\_unlock(&ptmutex1);

}

Código de ejecutable Diccionario (Visual Studio 2019)

**Aplicativo.cpp**

#include <string.h>

#include <errno.h>

#include <conio.h>

#include <ctype.h>

#include <stdio.h>

#include <stdlib.h>

#include<fstream>

#include<iostream>

#include "Clases.h"

#include<Windows.h>

//-------------------------------------------------------------------------------------

// Estructura del Nodo

//-------------------------------------------------------------------------------------

struct nodos {

char ingles[50];

char espanol[50];

struct nodos\* der;

struct nodos\* izq;

int Fe;

};

//-------------------------------------------------------------------------------------

// --------------------

//-------------------------------------------------------------------------------------

typedef struct nodos Arbol;

typedef Arbol\* ARBOL;

int listo;

char in[50], es[50];

void menu\_agregar();

void menu\_ingles();

void menu\_espanol();

void menu\_borrar();

void menu\_listar();

void agregar(ARBOL\* nodoarb);

void buscar\_ingles(ARBOL nodoarb);

void buscar\_espanol(ARBOL nodoarb);

void listar(ARBOL nodoarb);

void rotacionddsimple(ARBOL\* nodoarb);

void rotaciondidoble(ARBOL\* nodoarb);

void rotaciondd2(ARBOL\* nodoarb);

void rotacionii2(ARBOL\* nodoarb);

void rotacioniisimple(ARBOL\* nodoarb);

void rotacioniddoble(ARBOL\* nodoarb);

void menor(ARBOL nodoarb);

void actualizarbi(ARBOL\* nodoarb, int\* ban);

void actualizarbd(ARBOL\* nodoarb, int\* ban);

void eliminar(ARBOL\* nodoarb, int\* ban);

void borrar(ARBOL\* nodoarb, int\* ban);

void main()

{

int ban = 1;

ARBOL raiz = NULL;

char x[20],y[20]="miespe";

char sonido[20],sonidoi[20];

char mp3[10]=".wav";

int x1,y1,cl[9],inc,i=0, aux,m;

char op,opcion,u[30],c[30];

inc=0;

Acceso archivo(u,c);

archivo.creararchivo();

do{

system("cls");

fflush(stdin);

archivo.consola();

printf\_s("\n\n\n\t\t\t\t\t\t USUARIO: ");

gets\_s(u);

Acceso archivou(u,c);

x1=archivou.validarusuario(archivou);

}while(x1!=-1);

do{

system("cls");

archivo.consola();

printf\_s("\n\n\n\t\t\t\t\t\t USUARIO: ");

printf\_s("%s",u);

printf\_s("\n\n\t\t\t\t\t\t CLAVE: ");

while((\*cl = \_getch()) != 13){ //Validar tecla borrar

if(\*cl == 8){

printf("\b \b");

inc--;

}

else{

putchar('\*');

c[inc++] = \*cl;

}

}

c[inc] = '\0';

inc = 0;

Acceso archivoc(u,c);

x1=archivoc.validarclave(archivoc);

}while(x1!=-1);

system("cls");

Opciones obj;

obj.Palabras();

obj.Agregar();

obj.Traducir();

obj.Backup();

obj.Ayuda();

obj.Salir();

y1=mover();// llama a func mover --> retorna valor de la posicion en dond se encuentra l cursor

do{

if(y1>=0&&y1<=7){

system("cls");

menu\_listar();

listar(raiz);

\_getch();

}

if(y1>=11&&y1<=17){

system("cls");

std::cout<<"\n\t\t\t !!Las palabras solo pueden ser agregadas por el administrador!!";

SetConsoleTextAttribute(color,11);

obj.consola();

fflush(stdin);

printf("\t\t\t\t\t Administrador: ");

while((\*cl = \_getch()) != 13){ //Validar tecla borrar

if(\*cl == 8){

printf("\b \b");

inc--;

}

else{

putchar('\*');

x[inc++] = \*cl;

}

}

x[inc] = '\0';

inc = 0;

if(strcmp(x,y)!= 0||strcmp(x,y)==1){

SetConsoleTextAttribute(color,13);

printf("\n\t\t\t\t !!Acceso Denegado. Contrase%ca incorrecta!! ",164);

getch();

}

else{

do{

SetConsoleTextAttribute(color,15);

system("cls");

obj.consola();

std::cout << "Introduzca Palabra en Ingles : ";

std::cin >> in;

std::cout << "Introduzca Palabra en Espanol : ";

std::cin >> es;

agregar(&raiz);

std::cout<<"\n\n\t\t\t\t\t\tDesea ingresar mas datos?? 's'.\n\t\t\t\t\t\tCaso contrario cualquier tecla.";

std::cin >> op;

}while(op=='s' || op=='S');

}

}

if(y1>=21&&y1<=28){

system("cls");

obj.consola();

do{

do{

m=0;

fflush(stdin);

system("cls");

system("cls");

SetConsoleTextAttribute(color,15);

obj.consola();

SetConsoleTextAttribute(color,15);

printf("\nMENU\n");

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

SetConsoleTextAttribute(color,14);

printf("1.-English-Spanish\n",164,130);

SetConsoleTextAttribute(color,10);

printf("2.-Espa%col-Ingl%cs\n");

SetConsoleTextAttribute(color,7);

std::cout<<"3.-Salir\n";

SetConsoleTextAttribute(color,15);

cout<< "\n-->Ingrese la opcion a elegir: ";

gets\_s(c);

try{

int sw,i=0,k;

sw=0;

k=strlen(c);

while(i<k && sw==0){

if(isdigit(c[i])!=0||c[i]=='.')

i++;

else

throw(0);

}

}

catch(...){

cout<<"ERROR!!!\n\n";

m=1;

}

}while(m!=0);

aux=atoi(c);

switch(aux){

case 1:

std::cout << "Que Palabra en Ingles desea Buscar? : ";

std::cin >> in;

listo = 0;

buscar\_ingles(raiz);

if (listo == 0)

std::cout << "Elemento no encontrado!\n";

else {

std::cout << "\n";

strcpy(sonidoi, strcat(in, mp3));

system(sonidoi);

}

\_getch();

listo = 0;

system("pause");

break;

case 2:

std::cout << "Que Palabra en Espanol desea Buscar? : ";

std::cin >> es;

listo = 0;

buscar\_espanol(raiz);

if (listo == 0)

std::cout << "Elemento no encontrado!\n";

else {

std::cout << "\n";

strcpy(sonido, strcat(es, mp3));

system(sonido);

}

\_getch();

listo = 0;

system("pause");

break;

case 3:

system("pause");

break;

}

}while(aux!=3);

}

if(y1>=32&&y1<=43){

do{

Acceso obj1;

do{

m=0;

system("cls");

SetConsoleTextAttribute(color,15);

obj.consola();

SetConsoleTextAttribute(color,15);

printf("\nMENU\n");

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

SetConsoleTextAttribute(color,14);

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

SetConsoleTextAttribute(color,10);

std::cout<<"2.-Desencriptar\n";

SetConsoleTextAttribute(color,7);

std::cout<<"3.-Salir\n";

SetConsoleTextAttribute(color,15);

std::cout<<"\nIngrese la opcion: ";

gets\_s(c);

try{

int sw,i=0,k;

sw=0;

k=strlen(c);

while(i<k && sw==0){

if(isdigit(c[i])!=0||c[i]=='.')

i++;

else

throw(0);

}

}

catch(...){

cout<<"ERROR!!!\n\n";

m=1;

}

}while(m!=0);

aux=atoi(c);

switch(aux)

{

case 1:

obj1.encriptarclave();

system("encriptado.txt");

system("pause");

break;

case 2:

obj1.desencriptarclave();

system("desencriptado.txt");

system("pause");

break;

case 3:

system("pause");

break;

}

}while(aux!=3);

}

if(y1>=46&&y1<=50){

}

system("cls");

obj.Palabras();

obj.Agregar();

obj.Traducir();

obj.Backup();

obj.Ayuda();

obj.Salir();

y1=mover();

}while(y1!=54&&y1!=55&&y1!=56&&y1!=57&&y1!=58);

SetConsoleTextAttribute(color,14);

printf("\n\n\n\n\t\t\t\t\t\t !!! FIN DE PROGRAMA !!!\n\n\t\t\t\t\t\t");

}

void menu\_agregar()

{

system("cls");

std::cout << "--------------------------------------------------------------------------------\n";

std::cout << " \n";

std::cout << " AGREGAR PALABRAS AL TRADUCTOR \n";

std::cout << " \n";

std::cout << "--------------------------------------------------------------------------------\n";

std::cout << " \n";

std::cout << " \n";

std::cout << " \n";

std::cout << " \n";

}

//-------------------------------------------------------------------------------------

// Funcion Menu Buscar Ingles

//-------------------------------------------------------------------------------------

void menu\_ingles()

{

system("cls");

std::cout << "--------------------------------------------------------------------------------\n";

std::cout << " \n";

std::cout << " BUSCAR PALABRA EN INGLES \n";

std::cout << " \n";

std::cout << "--------------------------------------------------------------------------------\n";

std::cout << " \n";

std::cout << " \n";

std::cout << " \n";

std::cout << " \n";

}

//-------------------------------------------------------------------------------------

// Funcion Menu Buscar Espanol

//-------------------------------------------------------------------------------------

void menu\_espanol()

{

system("cls");

std::cout << "--------------------------------------------------------------------------------\n";

std::cout << " \n";

std::cout << " BUSCAR PALABRA EN ESPANOL \n";

std::cout << " \n";

std::cout << "--------------------------------------------------------------------------------\n";

std::cout << " \n";

std::cout << " \n";

std::cout << " \n";

std::cout << " \n";

}

//-------------------------------------------------------------------------------------

// Funcion Menu Borrar

//-------------------------------------------------------------------------------------

void menu\_borrar()

{

system("cls");

std::cout << "--------------------------------------------------------------------------------\n";

std::cout << " \n";

std::cout << " BORRAR PALABRA DEL DICCIONARIO \n";

std::cout << " \n";

std::cout << "--------------------------------------------------------------------------------\n";

std::cout << " \n";

std::cout << " \n";

std::cout << " \n";

std::cout << " \n";

}

//-------------------------------------------------------------------------------------

// Funcion Menu Listar

//-------------------------------------------------------------------------------------

void menu\_listar()

{

system("cls");

std::cout << "--------------------------------------------------------------------------------\n";

std::cout << " \n";

std::cout << " LISTAR PALABRAS DEL TRADUCTOR \n";

std::cout << " \n";

std::cout << "--------------------------------------------------------------------------------\n";

std::cout << " \n";

}

//-------------------------------------------------------------------------------------

// Funcion Agregar palabras al Diccionario

//-------------------------------------------------------------------------------------

void agregar(ARBOL\* nodoarb)

{

int det;

if (\*nodoarb == NULL) {

\*nodoarb = new(Arbol);

if (\*nodoarb != NULL) {

strcpy\_s((\*nodoarb)->ingles, in);

strcpy\_s((\*nodoarb)->espanol, es);

(\*nodoarb)->der = NULL;

(\*nodoarb)->izq = NULL;

}

else {

system("cls");

std::cout << "No hay memoria suficiente!";

}

}

else

{

det = strcmp(in, (\*nodoarb)->ingles);

if (det < 0)

agregar(&(\*nodoarb)->izq);

else

if (det > 0)

agregar(&((\*nodoarb)->der));

else

{

std::cout << "¡¡¡ Dato duplicado !!!";

}

}

}

//-------------------------------------------------------------------------------------

// Funcion Buscar Ingles

//-------------------------------------------------------------------------------------

void buscar\_ingles(ARBOL nodoarb)

{

if (nodoarb != NULL) {

buscar\_ingles(nodoarb->izq);

if (strcmp(in, nodoarb->ingles) == 0)

{

std::cout;

std::cout;

std::cout;

std::cout << " " << nodoarb->ingles << " -> " << nodoarb->espanol;

listo = 1;

return;

}

buscar\_ingles(nodoarb->der);

}

}

//-------------------------------------------------------------------------------------

// Funcion Buscar Ingles

//-------------------------------------------------------------------------------------

void buscar\_espanol(ARBOL nodoarb)

{

if (nodoarb != NULL) {

buscar\_espanol(nodoarb->izq);

if (strcmp(es, nodoarb->espanol) == 0)

{

std::cout;

std::cout;

std::cout;

std::cout << " " << nodoarb->espanol << " -> " << nodoarb->ingles;

listo = 1;

return;

}

buscar\_espanol(nodoarb->der);

}

}

//-------------------------------------------------------------------------------------

// Funcion listar Palabras

//-------------------------------------------------------------------------------------

void listar(ARBOL nodoarb)

{

if (nodoarb != NULL) {

listar(nodoarb->izq);

std::cout << " " << nodoarb->ingles << " -> " << nodoarb->espanol<<"\n";

listar(nodoarb->der);

}

}

//-------------------------------------------------------------------------------------

// Funcion rotacionddsimple

//-------------------------------------------------------------------------------------

void rotacionddsimple(ARBOL\* nodoarb)

{

ARBOL nda = NULL;

nda = (\*nodoarb)->der;

(\*nodoarb)->der = nda->izq;

nda->izq = (\*nodoarb);

nda->Fe = 0;

(\*nodoarb)->Fe = 0;

}

//-------------------------------------------------------------------------------------

// Funcion rotaciondidoble

//-------------------------------------------------------------------------------------

void rotaciondidoble(ARBOL\* nodoarb)

{

ARBOL nda1 = NULL, nda2 = NULL;

nda1 = (\*nodoarb)->der;

nda2 = nda1->der;

nda1->izq = nda2->der;

nda2->der = nda1;

(\*nodoarb)->der = nda2->izq;

nda2->izq = (\*nodoarb);

if (nda2->Fe == 1)

(\*nodoarb)->Fe = -1;

else

(\*nodoarb)->Fe = 0;

if (nda2->Fe == -1)

(\*nodoarb)->Fe = 1;

else

(\*nodoarb)->Fe = 0;

nda2->Fe = 0;

(\*nodoarb) = nda2;

}

//-------------------------------------------------------------------------------------

// Funcion rotaciondd2

//-------------------------------------------------------------------------------------

void rotaciondd2(ARBOL\* nodoarb)

{

ARBOL nda = NULL;

nda = (\*nodoarb)->der;

(\*nodoarb)->der = nda->izq;

nda->izq = (\*nodoarb);

nda->Fe = -1;

(\*nodoarb)->Fe = 1;

(\*nodoarb) = nda;

}

//-------------------------------------------------------------------------------------

// Funcion rotacionii2

//-------------------------------------------------------------------------------------

void rotacionii2(ARBOL\* nodoarb)

{

ARBOL nda = NULL;

nda = (\*nodoarb)->izq;

(\*nodoarb)->izq = nda->der;

nda->der = (\*nodoarb);

nda->Fe = 1;

(\*nodoarb)->Fe = -1;

(\*nodoarb) = nda;

}

//-------------------------------------------------------------------------------------

// Funcion rotacionsimple

//-------------------------------------------------------------------------------------

void rotacioniisimple(ARBOL\* nodoarb)

{

ARBOL nda = NULL;

nda = (\*nodoarb)->izq;

(\*nodoarb)->izq = nda->der;

nda->der = (\*nodoarb);

nda->Fe = 0;

(\*nodoarb)->Fe = 0;

\*nodoarb = nda;

}

//-------------------------------------------------------------------------------------

// Funcion rotacioniddoble

//-------------------------------------------------------------------------------------

void rotacioniddoble(ARBOL\* nodoarb)

{

ARBOL nda1 = NULL, nda2 = NULL;

nda1 = (\*nodoarb)->izq;

nda2 = nda1->der;

nda1->der = nda2->izq;

nda2->izq = nda1;

(\*nodoarb)->izq = nda2->der;

nda2->der = (\*nodoarb);

if (nda2->Fe == 1)

nda1->Fe = -1;

else

nda1->Fe = 0;

if (nda2->Fe == -1)

(\*nodoarb)->Fe = 1;

else

(\*nodoarb)->Fe = 0;

nda2->Fe = 0;

(\*nodoarb) = nda2;

}

//-------------------------------------------------------------------------------------

// Funcion Menor

//-------------------------------------------------------------------------------------

void menor(ARBOL nodoarb)

{

if (nodoarb->izq == NULL)

{

strcpy\_s(in, nodoarb->ingles);

strcpy\_s(es, nodoarb->espanol);

}

else

menor(nodoarb->izq);

}

//-------------------------------------------------------------------------------------

// Funcion Actualizarbi

//-------------------------------------------------------------------------------------

void actualizarbi(ARBOL\* nodoarb, int\* ban)

{

switch ((\*nodoarb)->Fe) {

case -1:

(\*nodoarb)->Fe = 0;

break;

case 0:

(\*nodoarb)->Fe = 1;

\*ban = 0;

break;

case 1:

switch ((\*nodoarb)->der->Fe) {

case 1:

rotacionddsimple(&(\*nodoarb));

break;

case -1:

rotaciondidoble(&(\*nodoarb));

break;

case 0:

rotaciondd2(&(\*nodoarb));

\*ban = 0;

break;

}

break;

}

}

//-------------------------------------------------------------------------------------

// Funcion Actualizarbd

//-------------------------------------------------------------------------------------

void actualizarbd(ARBOL\* nodoarb, int\* ban)

{

switch ((\*nodoarb)->Fe) {

case 1:

(\*nodoarb)->Fe = 0;

break;

case 0:

(\*nodoarb)->Fe = -1;

\*ban = 0;

break;

case -1:

switch ((\*nodoarb)->izq->Fe) {

case -1:

rotacioniisimple(&(\*nodoarb));

break;

case 1:

rotacioniddoble(&(\*nodoarb));

break;

case 0:

rotacionii2(&(\*nodoarb));

\*ban = 0;

break;

}

break;

}

}

//-------------------------------------------------------------------------------------

// Funcion Eliminar

//-------------------------------------------------------------------------------------

void eliminar(ARBOL\* nodoarb, int\* ban)

{

ARBOL aux = NULL;

if ((\*nodoarb)->izq == NULL) {

aux = \*nodoarb;

\*nodoarb = (\*nodoarb)->der;

free(aux);

\*ban = 1;

}

else {

if ((\*nodoarb)->der == NULL) {

aux = \*nodoarb;

\*nodoarb = (\*nodoarb)->izq;

free(aux);

\*ban = 1;

}

else {

menor((\*nodoarb)->der);

borrar(&(\*nodoarb)->der, &(\*ban));

if (\*ban)

actualizarbd(&(\*nodoarb), &(\*ban));

}

}

}

//-------------------------------------------------------------------------------------

// Funcion Borrar

//-------------------------------------------------------------------------------------

void borrar(ARBOL\* nodoarb, int\* ban)

{

if (nodoarb == NULL)

\*ban = 0;

else

if (strcmp((\*nodoarb)->ingles, in) == 0)

eliminar(&(\*nodoarb), &(\*ban));

else

if (strcmp((\*nodoarb)->ingles, in) > 0) {

borrar(&(\*nodoarb)->izq, &(\*ban));

if (\*ban)

actualizarbi(&(\*nodoarb), &(\*ban));

}

else {

borrar(&(\*nodoarb)->der, &(\*ban));

if (\*ban)

actualizarbd(&(\*nodoarb), &(\*ban));

}

}

//-------------------------------------------------------------------------------------

// FIN

//-------------------------------------------------------------------------------------

**Clases.h**

#include <stdio.h>

#include <iostream>

#include<Windows.h>

#include <string>

#include <fstream>

#include <conio.h>

#include <time.h>

using namespace std;

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

typedef std::basic\_ifstream<TCHAR> tifstream;

typedef std::basic\_string<TCHAR> tstring;

//clase para el usuario y la contraseña

class Acceso{

private:

char usuario[30],clave[30];

public:

Acceso() {};

Acceso (char \*usuario,char \*clave);

char \*getUsuario();

void setUsuario(char \*cad);

char \*getClave();

void setClave(char \*cad1);

void creararchivo();

int validarusuario(Acceso obj);

int validarclave(Acceso obj);

void encriptarclave();

void desencriptarclave();

void consola();

};

class Opciones{ //pestañas

public:

void gotoxy(short x, short y);

friend void gotoxy(int x, int y);

friend char tecla(unsigned char pulsacion);

friend int mover();

void Palabras();

void consola();

void Agregar();

void Traducir();

void Backup();

void Ayuda();

void Salir();

};

void gotoxy(int x, int y)//activacion de gotoxy

{

COORD coord;

coord.X = x;

coord.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord);

}

char tecla(unsigned char pulsacion)

{

if (pulsacion == 224) //se hace esta condición por si no se llegase a ingresar flechas para mover

pulsacion = \_getch();

return pulsacion;

}

int mover() // movimiento de flechas de navegacion

{

int i = 0, j = 0, mov = 0;

do

{

mov = tecla(\_getch());

if (mov == 77) //DERECHA

{

if (i == 79)

{

gotoxy(0, j++);

};

gotoxy(i++, j);

}

if (mov == 75) //IZQUIERDA

{

if (i == 0 && j != 0)

{

gotoxy(79, j--);

};

if (i == 0 && j == 0)

{

gotoxy(i, j);

}

gotoxy(i--, j);

}

if (mov == 72)//ARRIBA

{

if (j == 0)

{

//j = 0

gotoxy(i, 0);

//j = j;

}

gotoxy(i, j--);

}

if (mov == 80)

{

gotoxy(i, j++);

}

} while (mov != 13||(i>75||j!=0));

return i;

}

void Opciones::gotoxy(short x, short y)

{

COORD pantalla;

pantalla.X=(short) x;

pantalla.Y=(short) y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),pantalla);

}

////\*

void Opciones::consola(){

printf("\n\n\t\t\t |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\n");//62

printf("\t\t\t |||||||||||||||||||||||DICCIONARIO TRADUCTOR|||||||||||||||||||||\n");

printf("\t\t\t |||||||||||||||||||||||||Espa%col a Ingl%cs|||||||||||||||||||||||\n",164,130);

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

}

void Opciones::Palabras(){// pestañas

SetConsoleTextAttribute(color,10);

gotoxy(0,0);

cout<<"Palabras/"<<endl;

gotoxy(0,1);

printf("%c%c%c%c%c%c%c%c",238,238,238,238,238,238,238,238);

}

void Opciones::Agregar(){

SetConsoleTextAttribute(color,11);

gotoxy(11,0);

cout<<"Agregar/"<<endl;

gotoxy(11,1);

printf("%c%c%c%c%c%c%c",238,238,238,238,238,238,238);

}

void Opciones::Traducir(){

SetConsoleTextAttribute(color,13);

gotoxy(21,0);

cout<<"Traducir/"<<endl;

gotoxy(21,1);

printf("%c%c%c%c%c%c%c%c",238,238,238,238,238,238,238,238);

}

void Opciones::Backup(){

SetConsoleTextAttribute(color,15);

gotoxy(32,0);

cout<<"Encriptacion/"<<endl;

gotoxy(32,1);

printf("%c%c%c%c%c%c%c%c%c%c%c%c",238,238,238,238,238,238,238, 238, 238, 238, 238, 238);

}

void Opciones::Ayuda(){

SetConsoleTextAttribute(color,14);

gotoxy(46,0);

cout<<"Ayuda?/"<<endl;

gotoxy(46,1);

printf("%c%c%c%c%c%c",238,238,238,238,238,238);

}

void Opciones::Salir(){

SetConsoleTextAttribute(color,7);

gotoxy(54,0);

cout<<"Salir/"<<endl;

gotoxy(54,1);

printf("%c%c%c%c%c",238,238,238,238,238);

}

//Constructor de la clase acceso

Acceso::Acceso(char \*u,char \*c){

strcpy(usuario,u);

strcpy(clave,c);

}

//implementacion de los get y set

void Acceso::setUsuario(char \*cad){

strcpy(usuario,cad);

}

char\* Acceso::getUsuario(){

return usuario;

}

void Acceso::setClave(char \*cad1){

strcpy\_s(clave,cad1);

}

char\* Acceso::getClave(){

return clave;

}

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

void Acceso::creararchivo(){

ofstream fs("usuario.txt"); // Crea un fichero de salida

fs << "Edison" << endl; // Enviamos una cadena al fichero de salida

fs << "Angel" << endl;

fs.close(); // Cerrar el fichero, para luego poder abrirlo para lectura:

// Leeremos mediante getline, si lo hiciéramos mediante el operador << sólo leeríamos parte de la cadena

ofstream ft("clave.txt"); // Crea un fichero de salida

ft << "miespe" << endl; // Enviamos una cadena al fichero de salida

ft << "m1espe1" << endl;

ft.close(); // Cerrar el fichero, para luego poder abrirlo para lectura:

}

int Acceso::validarusuario(Acceso obj){

char us1[30],us2[30],us3[30],us4[30],us5[30];

ifstream fe("usuario.txt"); // Abre un fichero de entrada

fe.getline(us1, 30);

fe.getline(us2, 30);

if((strcmp(us1,obj.getUsuario())== 0)||(strcmp(us2,obj.getUsuario())== 0)){

return -1;

}

else{

printf\_s("\n\t\t\t\t\t !!Acceso Denegado. El usuario es incorrecto!! ");

\_getch();

}

}

int Acceso::validarclave(Acceso obj){

char cl1[30],cl2[30];

ifstream fp("clave.txt"); // Abre un fichero de entrada

fp.getline(cl1, 30);

fp.getline(cl2, 30);

if((strcmp(cl1,obj.getClave())==0&&strcmp("Edison",obj.getUsuario())==0)||(strcmp(cl2,obj.getClave())==0&&strcmp("Angel",obj.getUsuario())==0)){

return -1;

}

else{

printf\_s("\n\n\t\t\t\t\t !!Acceso Denegado. Clave incorrecta!! ");

\_getch();

}

}

void Acceso::encriptarclave() {

ifstream salida;

char cl[20];

FILE\* archivo;

char arch[50];

strcpy(arch, "encriptado.txt");

archivo = fopen(arch, "w");

SetConsoleTextAttribute(color, 13);

salida.open("clave.txt");

SetConsoleTextAttribute(color, 11);

salida >> cl;

while (!salida.eof())

{

SetConsoleTextAttribute(color, 15);

for (int i = 0; i < strlen(cl); ++i)

{

//convierte las minuscalas a mayuscula.

\*(cl + i) = toupper(\*(cl + i));

//descarta los digitos

if (isalpha(\*(cl + i)))

{

//verifica los casos especiales X-Y-Z

if (\*(cl + i) > 87)

{

\*(cl + i) = \*(cl + i) - 23;

}

else

{

\*(cl + i) = \*(cl + i) + 3;

}

}

}

fprintf(archivo, "%s\n", cl);

salida >> cl;

SetConsoleTextAttribute(color, 15);

}

cout << endl;

SetConsoleTextAttribute(color, 15);

salida.close();

fclose(archivo);

}

void Acceso::desencriptarclave() {

ifstream salida;

char cl[20];

FILE\* archivo1;

char arch[50];

strcpy(arch, "desencriptado.txt");

archivo1 = fopen(arch, "w");

SetConsoleTextAttribute(color, 13);

salida.open("encriptado.txt");

SetConsoleTextAttribute(color, 11);

salida >> cl;

while (!salida.eof())

{

SetConsoleTextAttribute(color, 15);

for (int i = 0; i < strlen(cl); ++i)

{

\*(cl + i) = tolower(\*(cl + i));

if (isalpha(\*(cl + i)))

{

if (\*(cl + i) < 87)

{

\*(cl + i) = \*(cl + i) + 23;

}

else

{

\*(cl + i) = \*(cl + i) - 3;

}

}

}

fprintf(archivo1, "%s\n", cl);

salida >> cl;

SetConsoleTextAttribute(color, 15);

}

cout << endl;

SetConsoleTextAttribute(color, 15);

salida.close();

fclose(archivo1);

}

void Acceso::consola(){

SetConsoleTextAttribute(color,10);

printf("\n\n\t %c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c %c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c %c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c %c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c\n", 179,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,177,179 ,179,219,219,219,219,219,219,219,219,219,219,219,219,219,219,177,179, 179,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,177,179, 179,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,177,179);

printf("\t %c%c%c%c%c::::::::::::. %c%c%c%c%c:::::::::::. %c%c%c%c//////////%c%c%c%c%c %c%c%c%c%c::::::::::::.\n",

179,219,219,219,177, 179,219,219,219,177, 179,219,219,219,47,219,219,177,179, 179,219,219,219,177);

printf("\t %c%c%c%c%c``````` %c%c%c%c%c`````````` %c%c%c%c %c%c%c%c %c%c%c%c%c``````` \n",

179,219,219,219,177, 179,219,219,219,177, 179,219,219,219, 219,219,177,179, 179,219,219,219,177);

printf("\t %c%c%c%c%c%c%c%c%c%c%c%c %c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c %c%c%c%c..........-%c%c%c%c %c%c%c%c%c%c%c%c%c%c%c%c \n",

179,219,219,219,219,219,219,219,219,219,177,179, 179,219,219,219,219,219,219,219,219,219,219,219,219,219,219,177,179,

179,219,219,219,219,219,177,179, 179,219,219,219,219,219,219,219,219,219,177,179);

printf("\t %c%c%c%c%c------. `.----------%c%c%c%c %c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c %c%c%c%c%c------. \n",

179,219,219,219,177, 219,219,177,179, 179,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,177,179, 179,219,219,219,177);

printf("\t %c%c%c%c%c-...........` .............%c%c%c%c %c%c%c%c%c%c..........` %c%c%c%c%c-...........`\n",

179,219,219,219,177, 219,219,177,179, 179,219,219,219,177,179, 179,219,219,219,177);

printf("\t %c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c %c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c %c%c%c%c%c%c %c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c\n",

179,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,177,179, 179,219,219,219,219,219,219,219,219,219,219,219,219,219,219,177,179, 179,219,219,219,177,179,

179,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,177,179);

printf("\n\n");

SetConsoleTextAttribute(color,15);

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

printf("\n\t\t\t\t\t Realizado por: Baez Edison, Cardenas Angel");

SetConsoleTextAttribute(color,14);

printf("\n");

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

printf("\t\t\t\t |\_ \_| \_ %c /\_%c | %c| | | |/ \_\_|\_ \_/ \_ %c| \_ %c\n", 92, 92, 92, 92, 92);

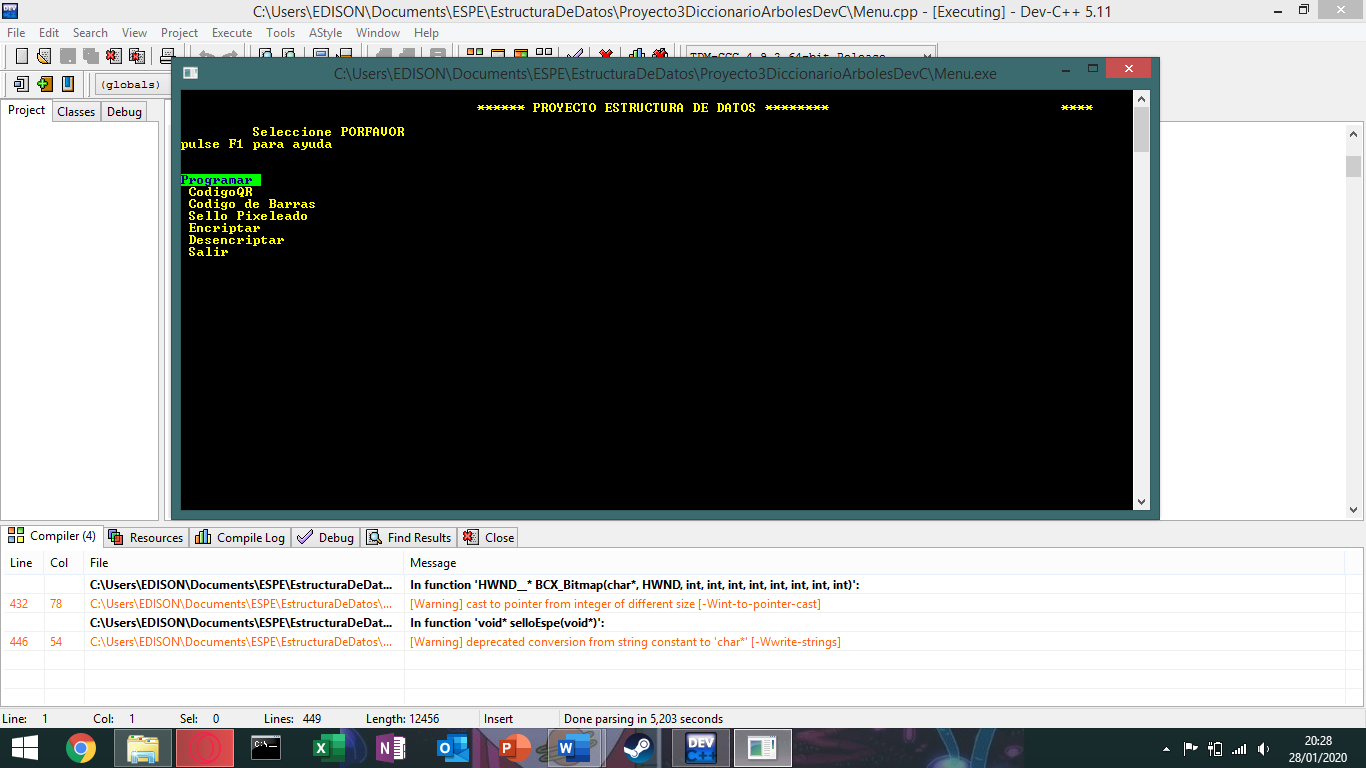
printf("\t\t\t\t | | | / / \_ %c| |) | |\_| | (\_\_ | || (\_) | /\n",92);

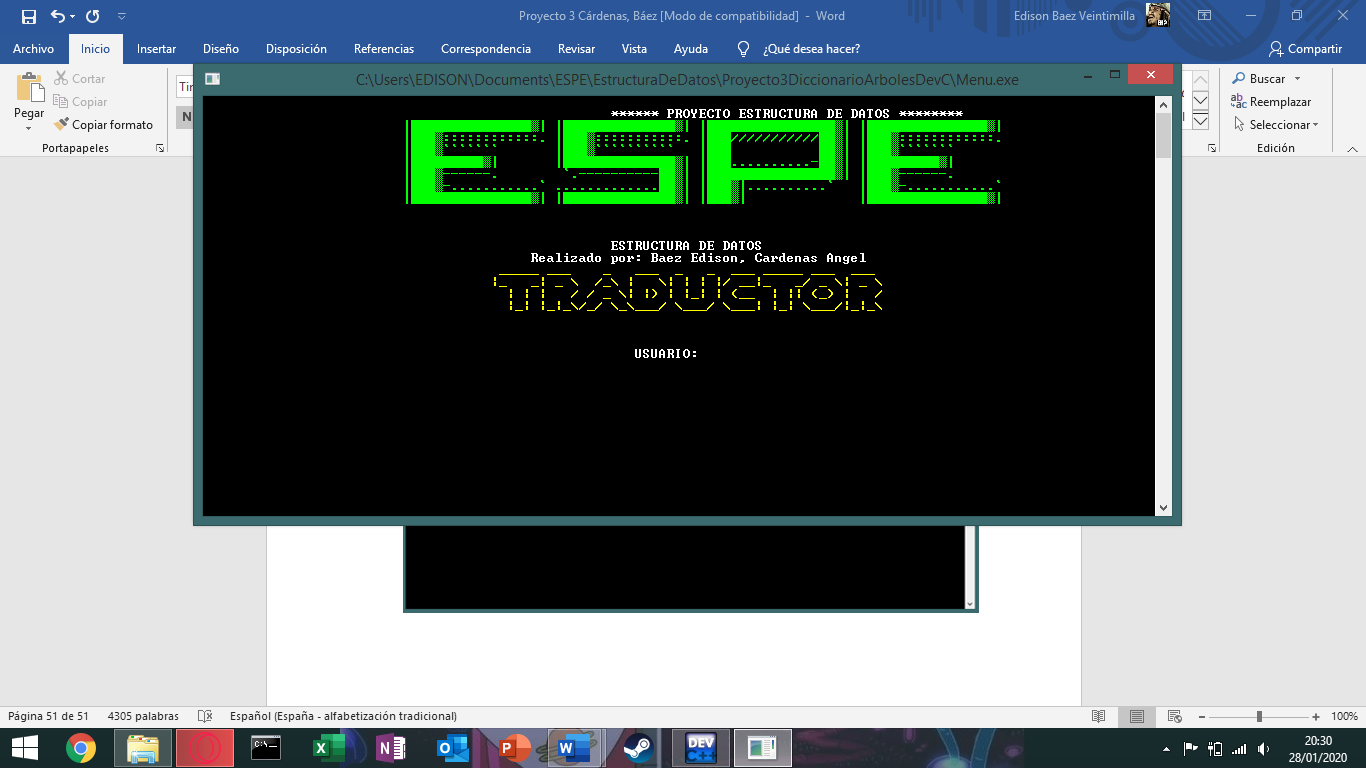
printf("\t\t\t\t |\_| |\_|\_%c/\_/ %c\_%c\_\_\_/ %c\_\_\_/ %c\_\_\_| |\_| %c\_\_\_/|\_|\_%c\n", 92, 92, 92, 92, 92,92,92);

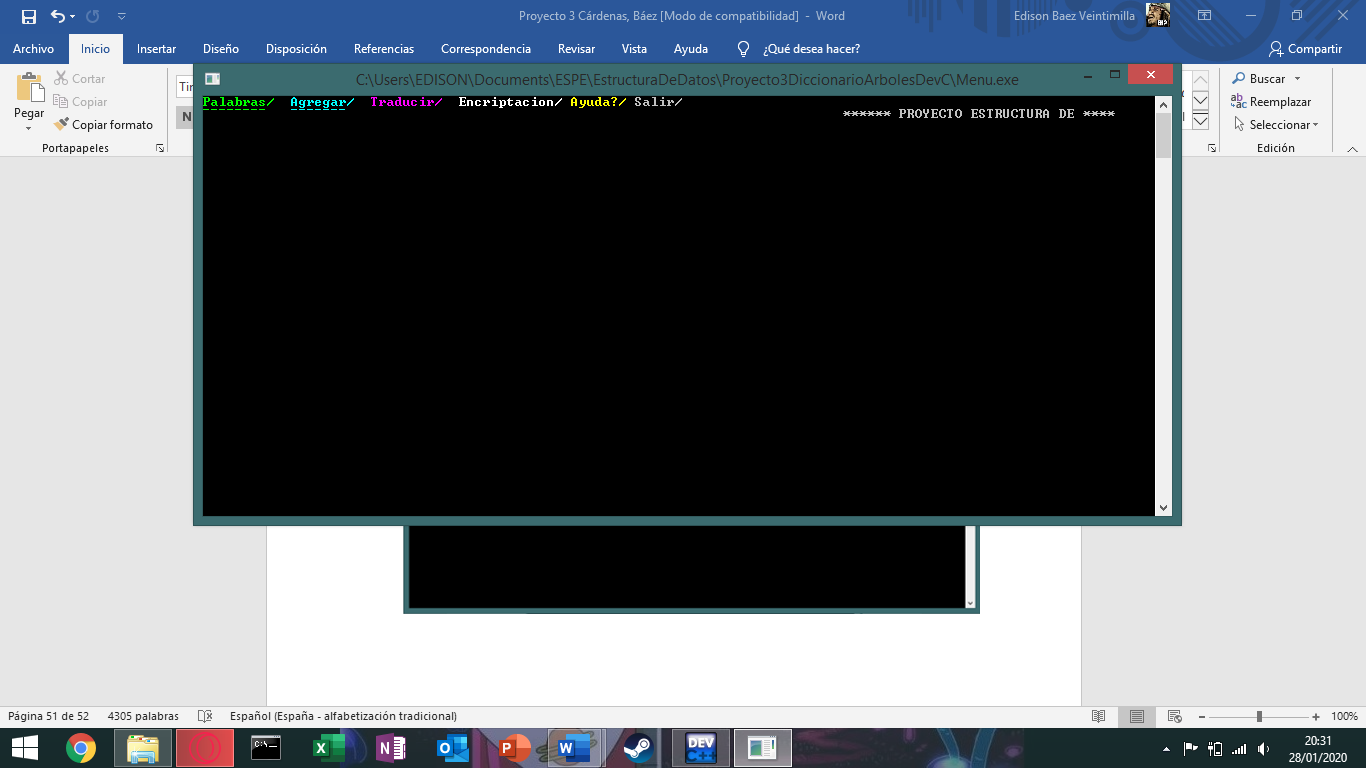
SetConsoleTextAttribute(color,15);

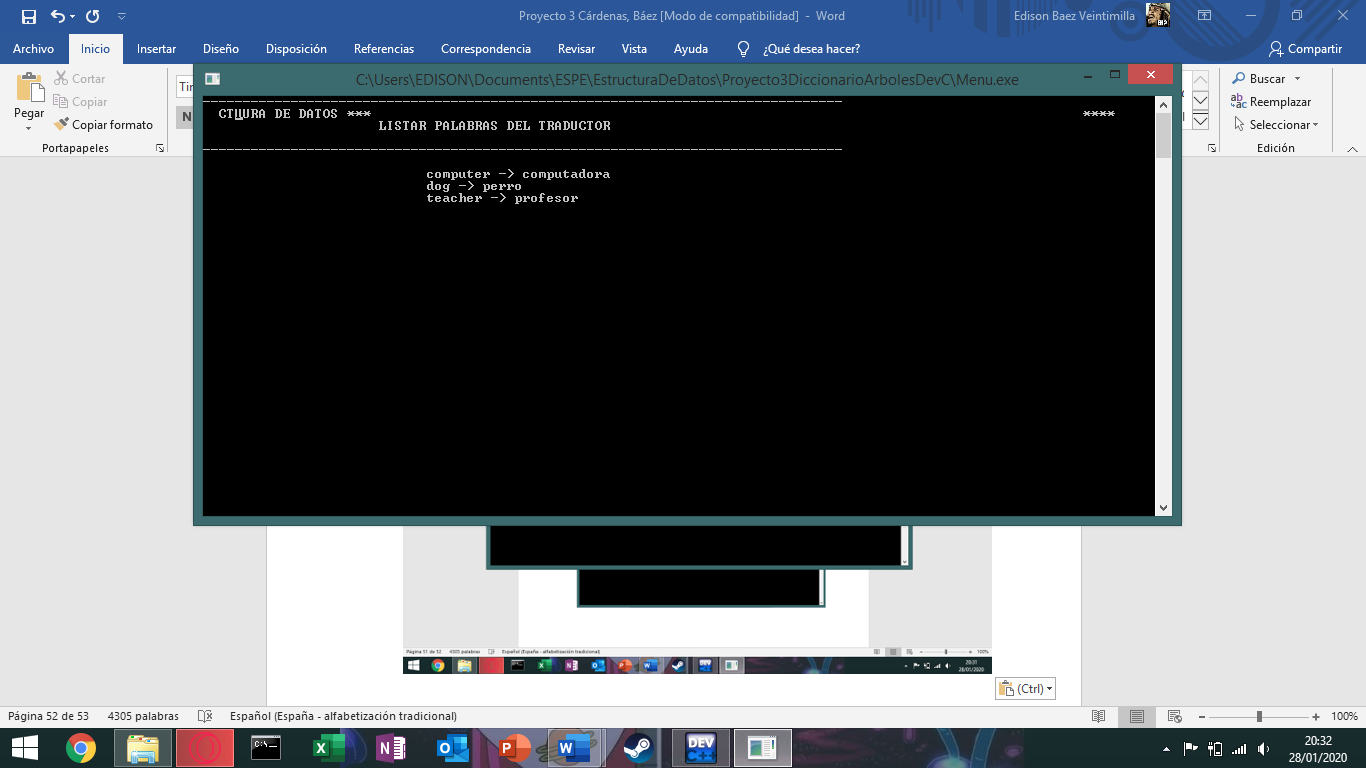
}

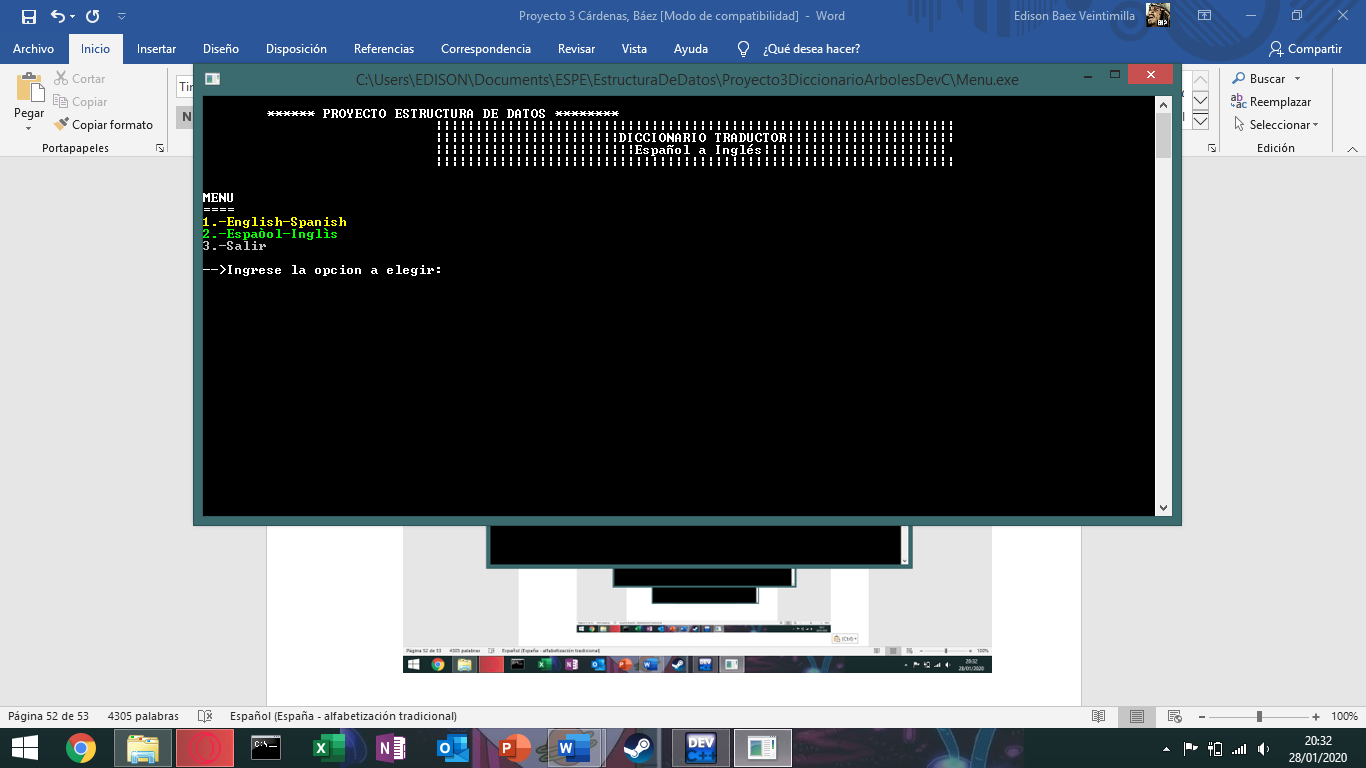
1. **Ejecución**

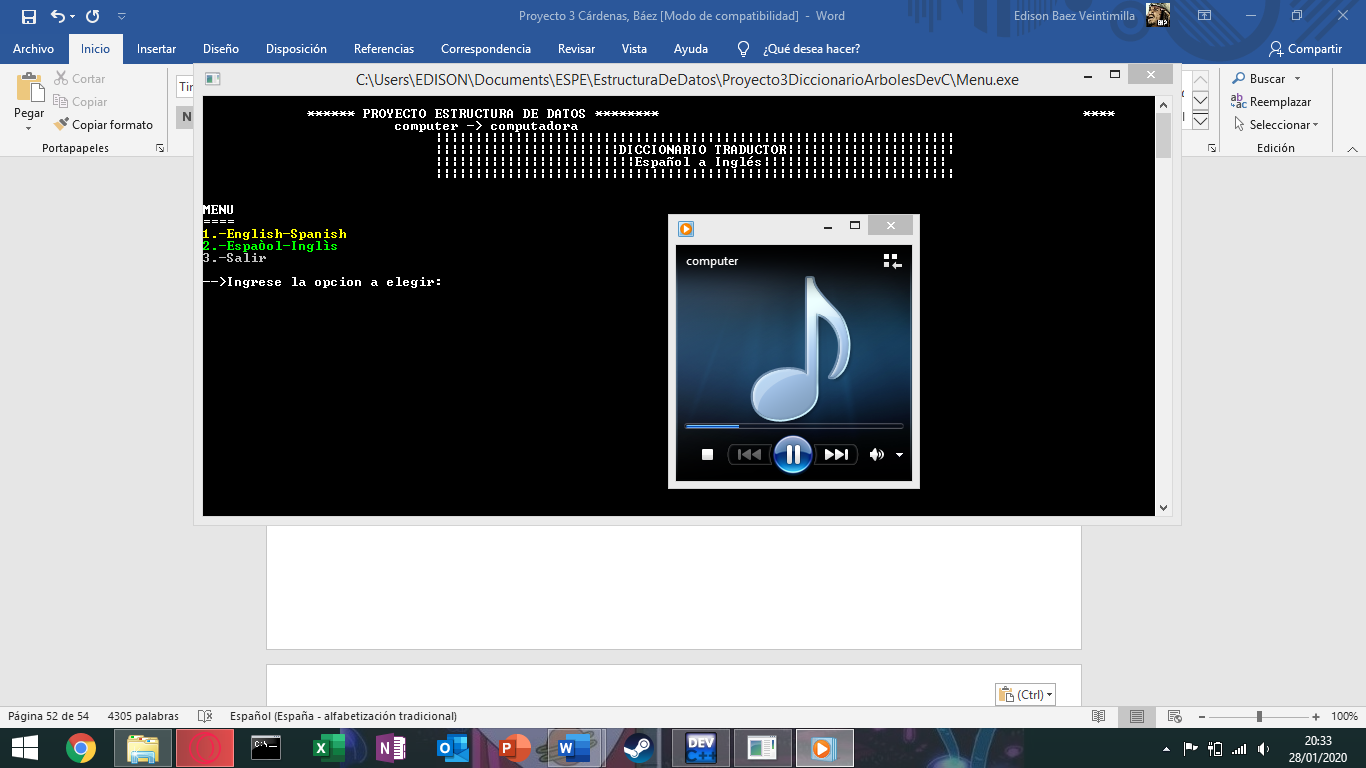












1. **Conclusiones**

Después del respectivo proyecto se pudo comprender de mejor manera el concepto de árboles. Se aprendió a utilizar de manera los algoritmos de árboles y entender la importancia en la estructura de datos.