Proyecto fase 2

Programación 3
Ingeniería en sistemas
Universidad Mariano Gálvez de Guatemala, Sede Portales

9989 21 13256 – Erwin Adolfo Roldán Hernández
9989 21 392 – Javier Alejandro Avalos Galindo

Contenido

Proyecto fase 1

Creación de una aplicación en c++ que permita realizar la lectura de un archivo, que contenga registros estructurados de palabras en español y sus traducciones a italiano, francés, alemán e inglés.

El programa debe de poder cargar la información desde un archivo,
El programa debe de permitir agregar nuevos nodos con las palabras y sus traducciones,
El programa debe de permitir eliminar nodos del árbol.
Debe de tener la capacidad de reproducir el audio de las palabras traducidas.

Definición de librerías

```
1 #include <iostream>
2 #include <sstream>
3 #include <fstream>
4 #include <conio.h>
5 #include <windows.h>
6 #include <cstdio>
7 #include <MMsystem.h>
8 #include<stdio.h>
9 #include "colors.h"
```

Iostream

Se definió para poder utilizar datos de entrada y salida ya que el usuario debe ingresar palabras en español y debe obtener la palabra en la traducción seleccionada.

Sstream

Se utilizó está librería para poder trabajar flujos de datos en forma de cadenas de caracteres.

Fstream

La utilizamos para poder realizar operaciones de entrada y salida en archivos ya que las palabras fueron almacenadas en un documento csv.

Conio.h

Se utilizó para poder leer caracteres del teclado y poder y generar o modificar texto en pantalla.

Windows.h

Es necesario interactuar con el sistema operativo, esta librería nos da la capacidad de interactuar con la base de datos antes mencionada y también con los archivos de audio de cada una de las palabras.

Cstdio

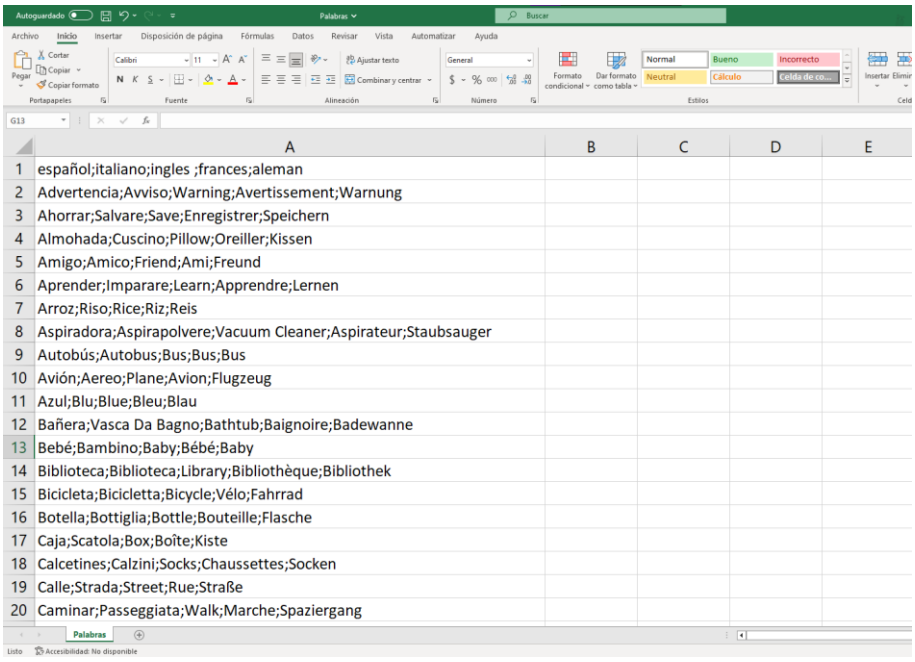
Esta librería nos permite interactuar con archivos, como abrirlos, cerrarlos o incluso modificarlos agregando o eliminando datos.

MMsystem.h

Seleccionamos esta librería para poder tener la capacidad de reproducir archivos en nuestro programa ya que se nos solicita reproducir un audio de la palabra traducida.

Base de datos

Para poder almacenar el listado de palabras realizamos un archivo csv donde se utilizó una columna, en cada celda se ingresó inicialmente la palabra en español seguido de sus traducciones en italiano, inglés, francés y alemán. Cada una de estas palabras está separada por un punto y coma “;” para que el programa pueda interpretar que son datos relacionados más no son el mismo. Se ingresaron 100 palabras en español con sus respectivas traducciones haciendo un total de 500 palabras almacenadas dentro de la base de datos.



	A	B	C	D	E
1	español;italiano;ingles ;frances;aleman				
2	Advertencia;Avviso;Warning;Avertissement;Warnung				
3	Ahorrar;Salvare;Save;Enregistrer;Speichern				
4	Almohada;Cuscino;Pillow;Oreiller;Kissen				
5	Amigo;Amico;Friend;Ami;Freund				
6	Aprender;Imparare;Learn;Apprendre;Lernen				
7	Arroz;Riso;Rice;Riz;Reis				
8	Aspiradora;Aspirapolvere;Vacuum Cleaner;Aspirateur;Staubsauger				
9	Autobús;Autobus;Bus;Bus;Bus				
10	Avión;Aereo;Plane;Avion;Flugzeug				
11	Azul;Blu;Blue;Bleu;Blau				
12	Bañera;Vasca Da Bagno;Bathtub;Baignoire;Badewanne				
13	Bebé;Bambino;Baby;Bébé;Baby				
14	Biblioteca;Biblioteca;Library;Bibliothèque;Bibliothek				
15	Bicicleta;Bicicletta;Bicycle;Vélo;Fahrrad				
16	Botella;Bottiglia;Bottle;Bouteille;Flasche				
17	Caja;Scatola;Box;Boîte;Kiste				
18	Calcetines;Calzini;Socks;Chaussettes;Socken				
19	Calle;Strada;Street;Rue;Straße				
20	Caminar;Passeggiata;Walk;Marche;Spaziergang				

Carga de datos

Lo primero que debemos hacer al momento de trabajar con una base de datos es cargar los datos que ya se encuentran en la misma. En este caso nuestra base de datos es un archivo CSV, por lo tanto, debemos crear una función “CargarDatos” que se encargue de obtener todos los datos que tenemos en la misma y guardarlos dentro de la estructura de árboles que a continuación se detallará.

Es importante saber que para cada una de las inserciones dentro de la BD se debe crear un nuevo nodo, el cual será colocado de acuerdo a las propias características del dato que estemos ingresando. La función “nuevoNodo” e “insertarNodo” realizaran estas acciones en nuestra estructura de datos.

```
void CargarDatos() {
    ifstream archivo(NOMBRE_ARCHIVO);
    string linea;
    char delimitador = ';';

    // Declarar el árbol AVL
    Nodo* arbolAVL = NULL;

    while (getline(archivo, linea)) {
        stringstream stream(linea); // Convertir la cadena a un stream
        string espanol, italiano, ingles, frances, aleman;
        // Extraer todos los valores de esa fila
        getline(stream, espanol, delimitador);
        getline(stream, italiano, delimitador);
        getline(stream, ingles, delimitador);
        getline(stream, frances, delimitador);
        getline(stream, aleman, delimitador);

        // Crear un nuevo nodo para la palabra
        Nodo* nuevoNodo = new Nodo;
        nuevoNodo->palabra = espanol;
        nuevoNodo->izquierdo = NULL;
        nuevoNodo->derecho = NULL;

        // Insertar el nuevo nodo en el árbol AVL
        arbolAVL = insertarNodo(arbolAVL,Palabra);
    }
}
```

Por ultimo, en nuestro main obtenemos la lista de palabras que tenemos almacenadas en nuestra BD

```
int main() {
    Nodo* arbolAVL = NULL;

    // Cargar datos en el árbol AVL
    CargarDatos();

    // Insertar palabras en el árbol AVL
    for (int i = 0; i < x; i++) {
        arbolAVL = insertarNodo(arbolAVL, Pal[i].PalEs);
    }

    // Realizar recorrido inorden del árbol AVL
    cout << "Recorrido inorden del arbol AVL: ";
    recorridoInorden(arbolAVL);
    cout << endl;
```

Lectura de las palabras almacenadas

Estructura para recorrer archivo

Se define una estructura de datos de árbol AVL, los mismo se realiza para que la búsqueda de las palabras sea mas eficiente y no tenga que recorrer todas las posibilidades dentro de las palabras del mismo idioma.

```
struct Nodo {
    string palabra;
    Nodo* izquierdo;
    Nodo* derecho;
    int altura;
};
```

Se realizan distintas funciones para controlar el árbol y poder trabajarlo como AVL, la creación de nodos y la rotación de estos es importante dentro de esta estructura de datos, debido a que los arboles AVL se caracterizan en la velocidad de búsqueda que tienen. Es una de las estructuras de datos más versátiles para trabajar.

```
// Función para obtener la altura de un nodo
int obtenerAltura(Nodo* nodo) {
    if (nodo == NULL)
        return 0;
    return nodo->altura;
}

// Función para obtener el máximo de dos enteros
int maximo(int a, int b) {
    return (a > b) ? a : b;
}

// Función para crear un nuevo nodo
Nodo* crearNodo(string palabra) {
    Nodo* nodo = new Nodo();
    nodo->palabra = palabra;
    nodo->izquierdo = NULL;
    nodo->derecho = NULL;
    nodo->altura = 1;
    return nodo;
}
```

```
// Función para rotar a la derecha un subárbol a partir del nodo y devolver el nuevo nodo raíz
Nodo* rotacionDerecha(Nodo* nodo) {
    Nodo* nodoIzquierdo = nodo->izquierdo;
    Nodo* subArbol = nodoIzquierdo->derecho;

    nodoIzquierdo->derecho = nodo;
    nodo->izquierdo = subArbol;

    nodo->altura = maximo(obtenerAltura(nodo->izquierdo), obtenerAltura(nodo->derecho)) + 1;
    nodoIzquierdo->altura = maximo(obtenerAltura(nodoIzquierdo->izquierdo), obtenerAltura(nodoIzquierdo->derecho)) + 1

    return nodoIzquierdo;
}

// Función para rotar a la izquierda un subárbol a partir del nodo y devolver el nuevo nodo raíz
Nodo* rotacionIzquierda(Nodo* nodo) {
    Nodo* nodoDerecho = nodo->derecho;
    Nodo* subArbol = nodoDerecho->izquierdo;

    nodoDerecho->izquierdo = nodo;
    nodo->derecho = subArbol;

    nodo->altura = maximo(obtenerAltura(nodo->izquierdo), obtenerAltura(nodo->derecho)) + 1;
    nodoDerecho->altura = maximo(obtenerAltura(nodoDerecho->izquierdo), obtenerAltura(nodoDerecho->derecho)) + 1;

    return nodoDerecho;
}
```

Además, se crean condiciones al momento de realizar la insecion de cualquier nodo dentro del árbol, estas condiciones nos ayudan a determinar si el árbol se encuentra o no balanceado en todos los posibles nodos con sus hijos.

```
// Caso de desequilibrio izquierda-izquierda
if (factorEquilibrio > 1 && palabra < raiz->izquierdo->palabra) {
    return rotacionDerecha(raiz);
}

// Caso de desequilibrio derecha-derecha
if (factorEquilibrio < -1 && palabra > raiz->derecho->palabra) {
    return rotacionIzquierda(raiz);
}

// Caso de desequilibrio izquierda-derecha
if (factorEquilibrio > 1 && palabra > raiz->izquierdo->palabra) {
    raiz->izquierdo = rotacionIzquierda(raiz->izquierdo);
    return rotacionDerecha(raiz);
}

// Caso de desequilibrio derecha-izquierda
if (factorEquilibrio < -1 && palabra < raiz->derecho->palabra) {
    raiz->derecho = rotacionDerecha(raiz->derecho);
    return rotacionIzquierda(raiz);
}
```

También, creamos una función para recorrer el árbol como tal.

```
// Función para realizar un recorrido inorden del árbol AVL
void recorridoInorden(Nodo* raiz) {
    if (raiz == NULL)
        return;

    recorridoInorden(raiz->izquierdo);
    cout << raiz->palabra << " ";
    recorridoInorden(raiz->derecho);
}
```

Recorrido de la base de datos

Debido a que ya tenemos las librerías necesarias para poder interactuar con archivos del sistema, abrirlos, leerlos y modificarlos solo debemos crear primero una función para poder ejecutar el archivo el cual se devine como NOMBRE_ARCHIVO después de declarar las librerías y entre comillas se coloca el nombre del archivo csv.

```
#define NOMBRE_ARCHIVO "Palabras.csv"
```

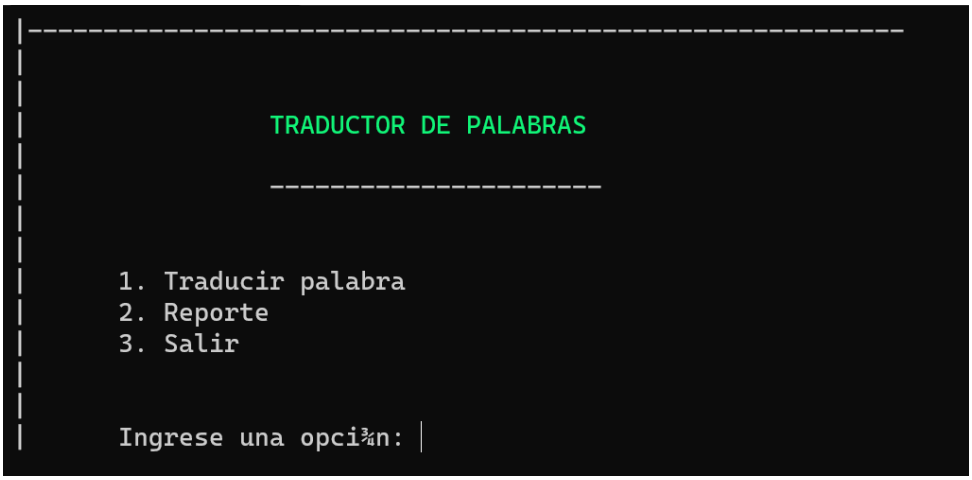
Luego de abrirlo el siguiente paso es poder recorrerlo por lo que utilizamos ifstream para poder acceder al archivo definido dentro de NOMBRE_ARCHIVO, después de esto utilizamos un void **CargarDatos** el cual toma como delimitador el punto y coma ";" para separar los datos que se encuentran el archivo, luego se define a que idioma pertenece cada una de estas palabras ingresándola a un string nombrado con cada uno de los idiomas disponibles dentro del traductor.

Menú principal

Se creo un menú principal en el cual el usuario podrá escoger entre 3 acciones a realizar dentro del programa: traducir, reporte, salir.

```
int menu(){
    int op;
    system("cls");
    cout<<"1= Traducir palabra"<<endl;
    cout<<"2= Reporte"<<endl;
    cout<<"3= Salir"<<endl;
    cin>>op;
    return op;
```

```
int main() {

    CargarDatos();

    do{
        retorno = menu();
        if(retorno==1) {
            do{
                int origen  = idioma_origen();
                int destino = idioma_destino();

                system("cls");

                cout<<"------ Ingrese una palabra ------"<<endl;
                cin>>palabra;

                busqueda(origen, destino, palabra);

                cout<<endl<<endl<<"1 = Traducir una nueva palabra \n2 = Volver al menu principal"<<endl;
                cin>>f;
            } while(f!=2);
        }
        if(retorno==2) {
            do {
                system("cls");
                cout<<"------- Reporte -------"<<endl;
                reporte();
                cout<<"\n\n2: Salir al menú principal"<<endl;
                cin>>f;
            } while(f!=2);
        }
    } while(retorno!=3);
```

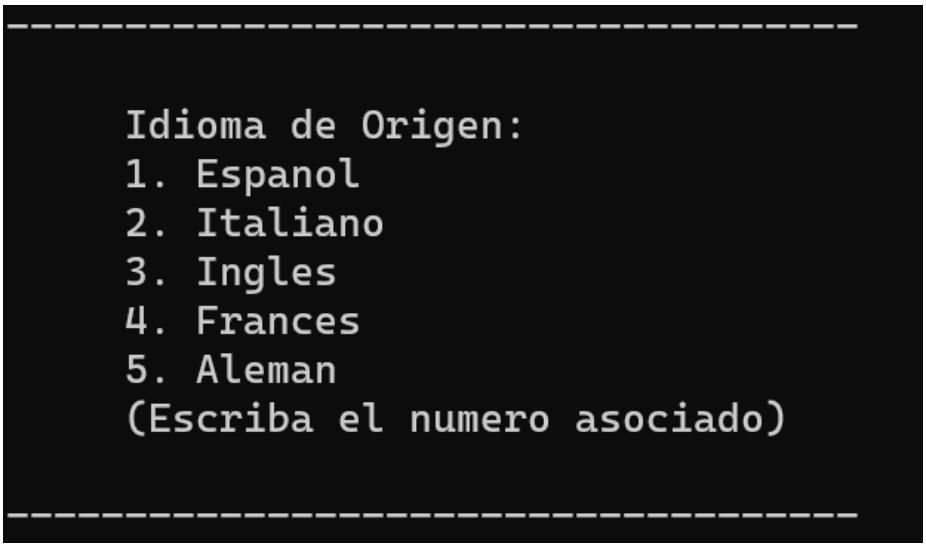


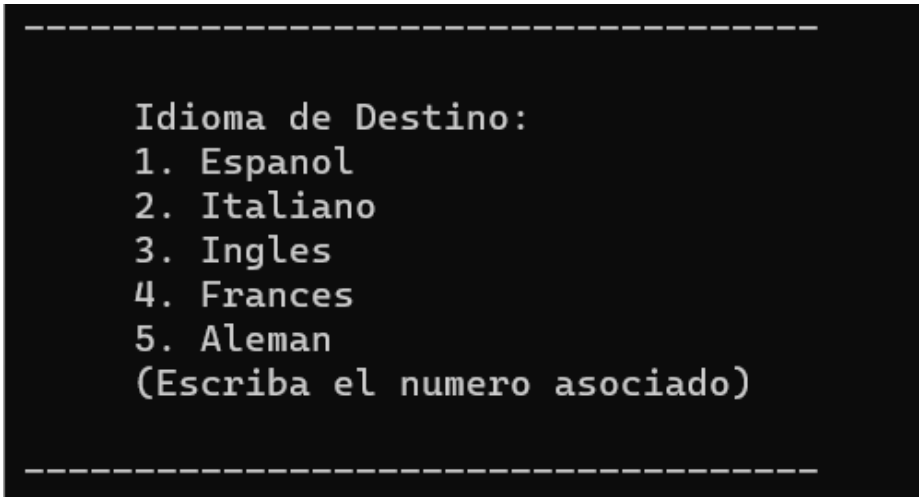
Traducir

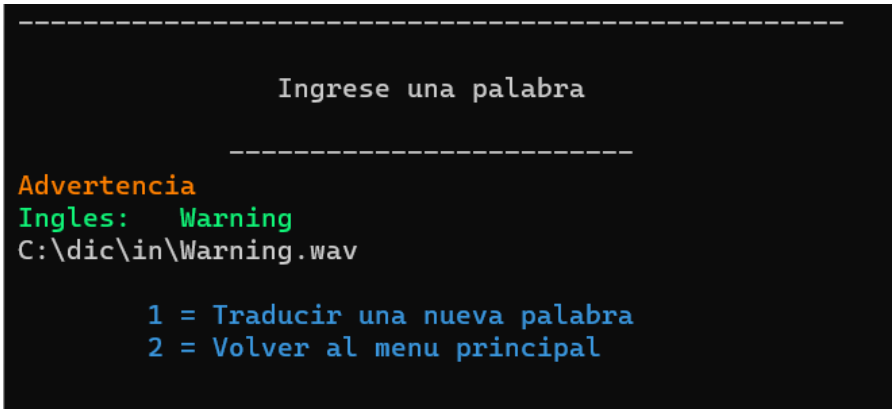
Si el usuario selecciona la opción traducir utilizaremos la función idioma_origen donde se debe indicar en que idioma se está ingresando la palabra, luego con la función idioma_destino el usuario seleccionará a que idioma desea traducir la palabra ingresada.

```
int idioma_origen(){
    system("cls");
    int op;
    cout<<"------\n Idioma de Origen:\n1. Espanol \n2. Italiano \n3. Ingles \n4. Frances \n5. Aleman \n(Escriba el numero asociado) \n------"<<endl;
    cin >> op;
    return op;
}

int idioma_destino(){
    system("cls");
    int op;
    cout<<"------\n Idioma de Destino: \n1. Espanol \n2. Italiano \n3. Ingles \n4. Frances \n5. Aleman \n(Escriba el numero asociado) \n------"<<endl;
    cin >> op;
    return op;
}
```

Luego de hacer la inserción empieza en proceso de búsqueda, primero se hace la validación de cual fue el idioma que seleccionó, luego se utiliza una variable auxiliar llamada encontrado el cual nos ayuda a validar si la palabra existe o no dentro de la base de datos, si esta se encuentra dentro de la base de datos arrojará true, por ultimo se agrega un break para que el programa pueda seguir validando a través de un else if con los demás idiomas disponibles.


Idioma de Origen:
1. Espanol
2. Italiano
3. Ingles
4. Frances
5. Aleman
(Escriba el numero asociado)


Idioma de Destino:
1. Espanol
2. Italiano
3. Ingles
4. Frances
5. Aleman
(Escriba el numero asociado)

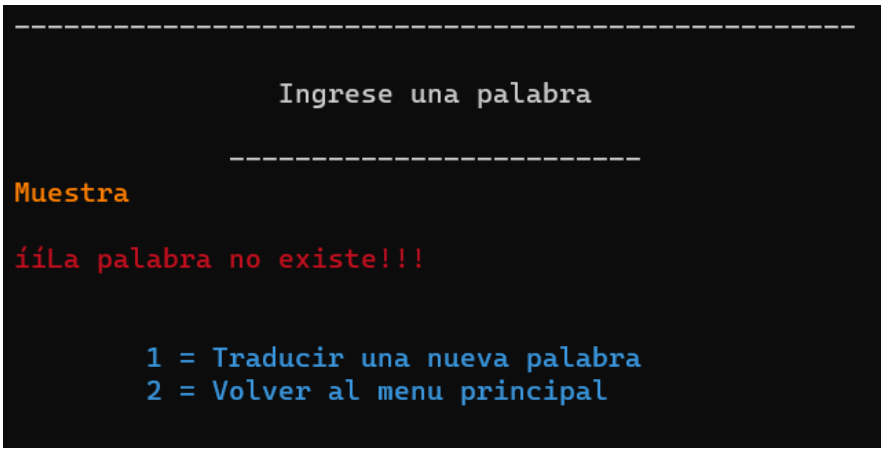

Ingrese una palabra
Advertencia
Ingles: Warning
C:\dic\in\Warning.wav
1 = Traducir una nueva palabra
2 = Volver al menu principal

```
void busqueda(int origen, int destino, string palabra){
    if(origen == 1){
        for(y=0; y<=100; y++) {
            if(palabra==Pal[y].PalEs) {
                encontrado=true;
                break;
            }
        }
    }else if(origen == 2){
        for(y=0; y<=100; y++) {
            if(palabra==Pal[y].PalIt) {
                encontrado=true;
                break;
            }
        }
    }else if(origen == 3){
        for(y=0; y<=100; y++) {
            if(palabra==Pal[y].PalIn) {
                encontrado=true;
                break;
            }
        }
    }else if(origen == 4){
        for(y=0; y<=100; y++) {
            if(palabra==Pal[y].PalFr) {
                encontrado=true;
                break;
            }
        }
    }else if(origen == 5){
        for(y=0; y<=100; y++) {
            if(palabra==Pal[y].PalDe) {
                encontrado=true;
                break;
            }
        }
    }
```

Ahora que ya pudimos recorrer los datos y validar si existe o no el siguiente paso es mostrarlos y reproducir el audio por lo cual de la variable “encontrado” el cual si arroja un true realizará un proceso en el cual primero se evalúa en que idioma fue encontrado, luego un cout mostrando la palabra en el idioma solicitado por el usuario y por último se ejecuta el string ruta el cual mandará a llamar el audio de la palabra a través de su localización en la carpeta del sistema operativo seguido del formato del mismo, en este caso wav, y de esta forma es como logramos mostrar y reproducir la palabra traducida.

```
if(encontrado == true){
    if(destino == 1){

        cout << "Espanol:  " << Pal[y].PalEs << endl;
        string ruta = "C:\\dic\\es\\"; ruta = ruta.append(Pal[y].PalEs); ruta = ruta.append(".wav");
        ruta_lpcstr = ruta.c_str();
        cout << ruta;
        encontrado=false;

    }else if(destino == 2){

        cout << "Italiano: " << Pal[y].PalIt << endl;
        string ruta = "C:\\dic\\it\\"; ruta = ruta.append(Pal[y].PalIt); ruta = ruta.append(".wav");
        ruta_lpcstr = ruta.c_str();
        cout << ruta;
        encontrado=false;

    }else if(destino == 3){

        cout << "Ingles:   " << Pal[y].PalIn << endl;
        string ruta = "C:\\dic\\in\\"; ruta = ruta.append(Pal[y].PalIn); ruta = ruta.append(".wav");
        ruta_lpcstr = ruta.c_str();
        cout << ruta;
        encontrado=false;

    }else if(destino == 4){

        cout << "Frances:  " << Pal[y].PalFr << endl;
        string ruta = "C:\\dic\\fr\\"; ruta = ruta.append(Pal[y].PalFr); ruta = ruta.append(".wav");
        ruta_lpcstr = ruta.c_str();
        cout << ruta;
        encontrado=false;

    }else if(destino == 5){

        cout << "Aleman:   " << Pal[y].PalDe << endl;
        string ruta = "C:\\dic\\de\\"; ruta = ruta.append(Pal[y].PalDe); ruta = ruta.append(".wav");
```

En caso de que encontrado sea igual a false simplemente se mostrará un mensaje en pantalla indicando que la palabra no existe.



Reporte

En el caso de que el cliente solicite un reporte de las palabras existentes se generara un recorrido a través de un for ya que conocemos la cantidad de datos que están manejando y a través de un cout mostraremos las palabras que están en cada idioma.

```
void reporte(){
    for(x=0; x<=100; x++) {
        // Imprimir
        cout << "==================" << endl;
        cout << "Espanol:  " << Pal[x].PalEs << endl;
        cout << "Italiano: " << Pal[x].PalIt << endl;
        cout << "Ingles:   " << Pal[x].PalIn << endl;
        cout << "Frances:  " << Pal[x].PalFr << endl;
        cout << "Aleman:   " << Pal[x].PalDe << endl;
    }
}
```

```
----------------------
               Espanol :   Advertencia
               Italiano:   Avviso
               Ingles  :   Warning
               Frances :   Avertissement
               Aleman  :   Warnung
----------------------
----------------------
               Espanol :   Ahorrar
               Italiano:   Salvare
               Ingles  :   Save
               Frances :   Enregistrer
               Aleman  :   Speichern
----------------------
----------------------
               Espanol :   Almohada
               Italiano:   Cuscino
               Ingles  :   Pillow
               Frances :   Oreiller
               Aleman  :   Kissen
----------------------
```