Estructuras de Datos

Instituto Tecnológico de Costa Rica

I Semestre 2019

Tarea Programada 2:
Registro de Personas

Integrantes:

- **Danny Piedra Acuña 2016210168**
- **Gerald Calvo Portilla 2018084184**

Fecha de Entrega: Viernes 10 de mayo, 2019

ÍNDICE

INTRODUCCIÓN

La gestión y manipulación de los datos para cualquier sistema de información es una parte fundamental puesto que nos permite crear servicios informáticos funcionales, confiables y revolucionarios. Las bases de datos en general han surgido desde los orígenes más nuevos de la computación, desde la arquitectura del CPU de Programa Almacenado propuesta por el computólogo matemático Jon Von Neumann al mezclar las instrucciones y programas en un mismo sector de memoria. Desde ese momento se ha tenido presente el concepto de preservar información útil para el usuario, otros sub sistemas informáticos, etc. Esto para ser utilizada posteriormente con múltiples fines.

Sin embargo, a través de los años este concepto ha estado siendo revolucionado con la invención de algoritmos que edifican estructuras complejas de datos. Con el propósito de crear en este conjunto de datos, un mecanismo de acceso y modificación estratégico.

Por esta razón, existen estructuras que datos que permiten organizar la información en memoria de disco duro como las de estructura ramal como lo son los Árboles. Y por otro lado, las estructuras de datos que almacenan la información en la memoria del programa en ejecución que permiten clasificarlas en función de sus atributos.

Por esta razón, es conveniente aplicar algoritmos como las tablas Hash, que nos permiten almacenar todo un conjunto de registros. Este tipo de estructura de datos es muy particular puesto que su definición por sí misma es ampliamente diversa y compleja en cuanto a sus propiedades matemáticas y aplicaciones criptográficas.

No solamente consideramos este tipo de algoritmo de amplia utilidad porque nos permite encapsular y ocultar información deseada de manera ordenada y confiable ya que solamente el conservador de la llave puede accesar al dato realmente. Y esta llave se genera aplicando una cantidad finita de algoritmos que parten del valor del dato en sí mismo.

Sino también el reto en la mayoría de los casos no consiste en la programación realmente, sino más bien en construir un algoritmo matemático difícilmente

predecible que evite en su mejor intento la retención de la información y esta sea facilitada exclusivamente al portador de la llave, y únicamente la llave.

Considerando todos estos factores de alto menester, se busca con este proyecto crear una representación real bajo un concepto muy básico de una Base de Datos que almacene una cantidad de registros con información de personas como lo son la Cédula, Nombre, Apellido, Apellido 2 y Fecha de Nacimiento.

También se pretende que la aplicación encargada de gestionar la Base de Datos cargue los archivos de la memoria del disco duro en formato Comma-separated values (.csv) en la memoria del programa y las almacene ahí temporalmente a través del algoritmo de las Tablas de Hash.

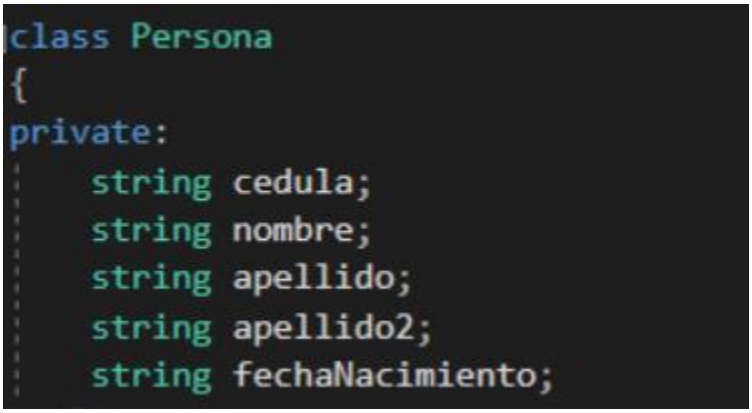
Toda la información respectiva en cuanto a la construcción y el modelado de la aplicación a través de sus clases se presentará en detalle en el siguiente apartado de la documentación del proyecto. Así como la explicación del algoritmo de la función Hash utilizada para gestionar los registros y aplicar las operaciones elementales a una base datos, conocidas como CRUD (Create Read Upate y Delete).

Finalmente, posterior a dar una clara introducción de la estrategia utilizada para llevar acabo este proyecto se presentará una lista con los resultados obtenidos y los problemas enfrentados al momento de llevar a cabo la abstracción del problema y la programación de la solución.

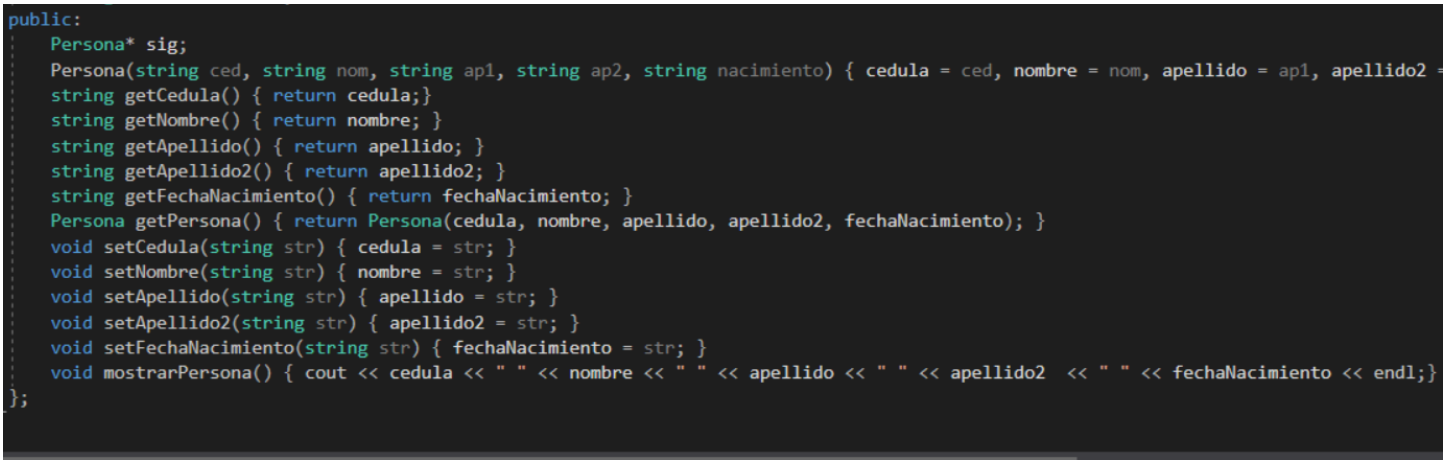
Descripción de las Clases

1) Persona

La primera clase utilizada y la más intuitiva en nuestro programa es la clase Persona. Que nos permite encapsular todas las características de las personas en una unidad. Por tanto, nuestra Base de Datos en memoria del programa estará conservando instancias de la clase Persona.



```
class Persona
{
private:
    string cedula;
    string nombre;
    string apellido;
    string apellido2;
    string fechaNacimiento;
```



```
public:
    Persona* sig;
    Persona(string ced, string nom, string ap1, string ap2, string nacimiento) { cedula = ced, nombre = nom, apellido = ap1, apellido2 =
    string getCedula() { return cedula;}
    string getNombre() { return nombre; }
    string getApellido() { return apellido; }
    string getApellido2() { return apellido2; }
    string getFechaNacimiento() { return fechaNacimiento; }
    Persona getPersona() { return Persona(cedula, nombre, apellido, apellido2, fechaNacimiento); }
    void setCedula(string str) { cedula = str; }
    void setNombre(string str) { nombre = str; }
    void setApellido(string str) { apellido = str; }
    void setApellido2(string str) { apellido2 = str; }
    void setFechaNacimiento(string str) { fechaNacimiento = str; }
    void mostrarPersona() { cout << cedula << " " << nombre << " " << apellido << " " << apellido2  << " " << fechaNacimiento << endl;}
};
```

También nuestra clase procesar tendrá métodos adicionales de suma importancia para otras clases de nuestro proyecto. Todos los métodos, sin embargo, bastante intuitivos y acorde al paradigma de programación orientado a objetos como los setters y getters.

Puesto que esta clase es pequeña y de un solo propósito se lleva a cabo su implementación en un archivo de encabezado.

2)Procesar

La clase Procesar se encarga de gestionar todas las funcionalidades auxiliares del programa tales como leer un archivo (.csv), escribir un archivo (.csv), procesar todas las validaciones de las entradas, instanciar el conjunto de objetos tipo Persona en un vector de Personas para poder llegar a cabo la escritura del archivo y manejar el traspaso de archivos a memoria interna con la clase Hash.

```
#include <vector>
#include <string>
#include <fstream>
#include <iostream>
#include <cctype>
#include "Persona.h"

class Procesar
{
private:
    vector<Persona> personas;
    string ruta;
public:
    Procesar() { ruta = ""; };
    void setPersonas(vector<Persona>p);
    void setRuta(string str);
    string getRuta() { return ruta; };
    vector<Persona> getPersonas() { return personas; };
    void cargarArchivo(string ruta);
    void insertarArchivo(string ruta);
    void insertarArchivo();
    bool validarCedula(string fecha);
    bool validarNombre(string nombre);
    bool validarFecha(string fecha);
    bool validarPersona(Persona p);
};
```

La clase cuenta con una cantidad pequeña de métodos, sin embargo, estos son altamente funcionales. Adicionalmente, la clase Procesar se encarga de conservar

en un atributo la última ruta de inserción del archivo (.csv), esto para llevar a cabo la funcionalidad de ‘’Guardar’’ de manera exitosa. Y esto a su vez, fue programado para que en caso que no se haya ‘’guardado un archivo como’’, cuando esta retorne una ruta vacía pregunte al usuario una ruta y la almacene para posteriores salvados de memoria.

3) Hash

Como se mencionó anteriormente su propósito es el de cargar de manera ordenada el archivo o la información insertada a través del formulario de la aplicación a la memoria interna del programa.

```
#include<iostream>
#include <string>
#include "Persona.h"
#include <vector>

using namespace std;

class Hash
{
private:
    static const int tamañoTabla = 20;
    Persona* tabla[tamañoTabla];

public:
    Hash();
    int idx(string key);
    void añadirPersona(Persona p);
    Persona* buscarPersona(string key);
    void modificarPersona(string nuevaCedula, string nuevoNombre, string nuevoAp1, string nuevoAp2, string nuevaFecha);
    int numeroColisiones(int idx);
    bool existeCedula(string cedula);
    void eliminarPersona(string key);
    vector<Persona> devolverPersonas();
    void limpiarHash();
    void imprimirTabla();
};
```

Adicionalmente, la clase Hash cuenta con una serie de métodos auxiliares importantes para el funcionamiento correcto del algoritmo. Como lo son la validación de si existe o no un elemento con determinada llave, buscar una persona en la tabla, modificar una persona, eliminar una persona, limpiar la tabla hash e imprimir la tabla.

Se ha escogido arbitrariamente el valor de 20 como tamaño de la tabla, esto con el propósito de tener un rango de distribución panorámica a través de la tabla y al

mismo tiempo pequeño al punto que intencionalmente genere colisiones, esto con fines académicos y de requerimientos para el proyecto.

Y finalmente el método devolver Personas, que nos carga el contenido de la tabla nuevamente en un vector de Personas, que pueda ser enviado a la clase Procesar para aterrizar este en un archivo (.csv).

El algoritmo utilizado para la función Hash consiste en hacer una suma consecutiva de cada uno de los términos de la llave (cédula), y a este resultado aplicarle la operación módulo con el número de la tabla para que este nos retorne una valor entero en un rango 0 . . . . (N-1), siendo N el tamaño de la tabla.

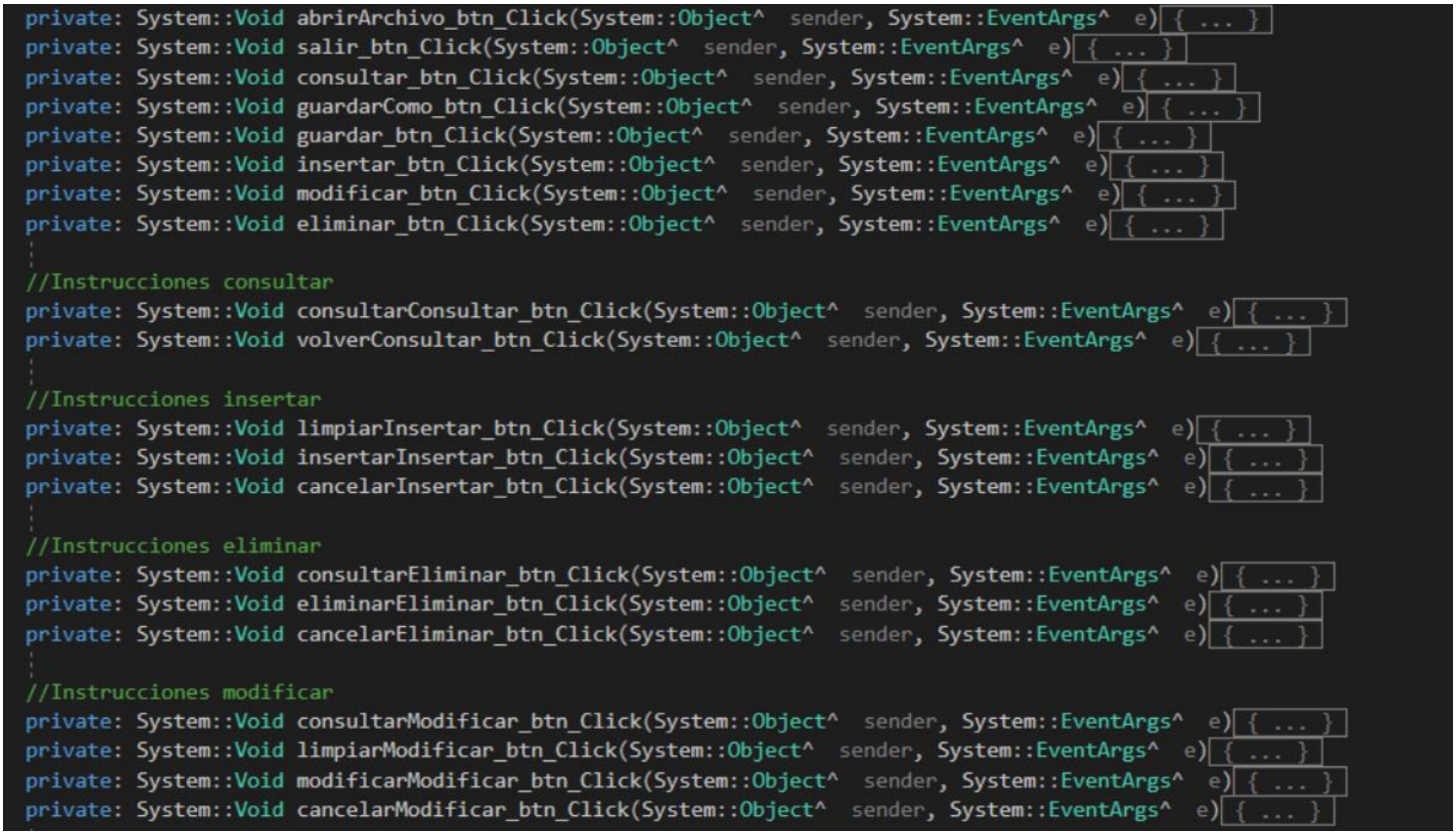
4) MyForm

Finalmente tenemos la clase MyForm, esta quizá es la clase más compleja y grande de todo nuestro programa. Esto dado que implementa los métodos nativos de los componentes gráficos de cada uno de los elementos de la ventana, como los botones, cuadros de texto, etiquetas y visibilidad de estos.

Por esta razón, no comentaremos todo el código referente a esta clase puesto que en su mayoría son métodos genéricos creados de manera automática. Más, sin embargo, si comentaremos la funcionalidad que se le otorga a los botones en relación a nuestras otras clases.

```
private: System::Windows::Forms::Button^  abrirArchivo_btn;
private: Hash* hash;
private: Procesar* procesamiento = new Procesar();
private: bool puedeEliminar = false;
private: bool puedeModificar = false;
private: string* abrirArchivo;
private: string* guardarArchivo;
```

Primeramente, contamos con los atributos de la clase que permite procesar y validar las entradas de los cuadros de texto a través de una instancia de la clase anteriormente descrita. De igual forma, tenemos atributos booleanos que nos indican si podemos eliminar o modificar en determinado momento en función del resultado de la validación de una entrada.



Explicar cada uno de los métodos se considera un ejercicio innecesario. Por esto, buscamos que cada una de las funcionalidades tuviese con un nombre auto descriptivo. Y se adjunta todo el código en la solución de este proyecto para consultar en profundidad cada una de sus funcionalidades.

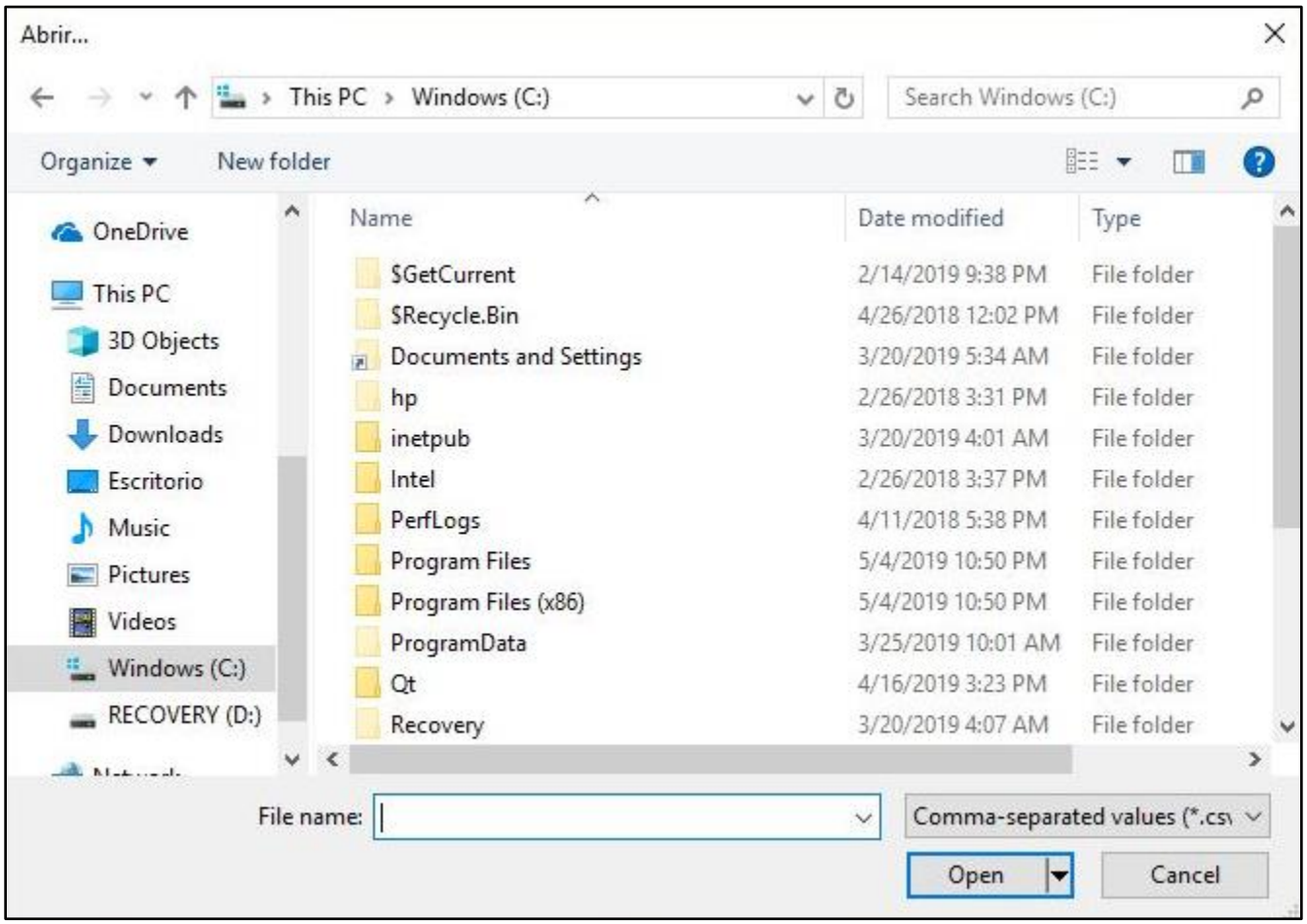
Adicionalmente, añadimos comentarios para segmentar los bloques de código con un propósito en específico como lo es Consultar, Insertar, Eliminar y Modificar.

Interfaz Gráfica

1) Menú Principal:

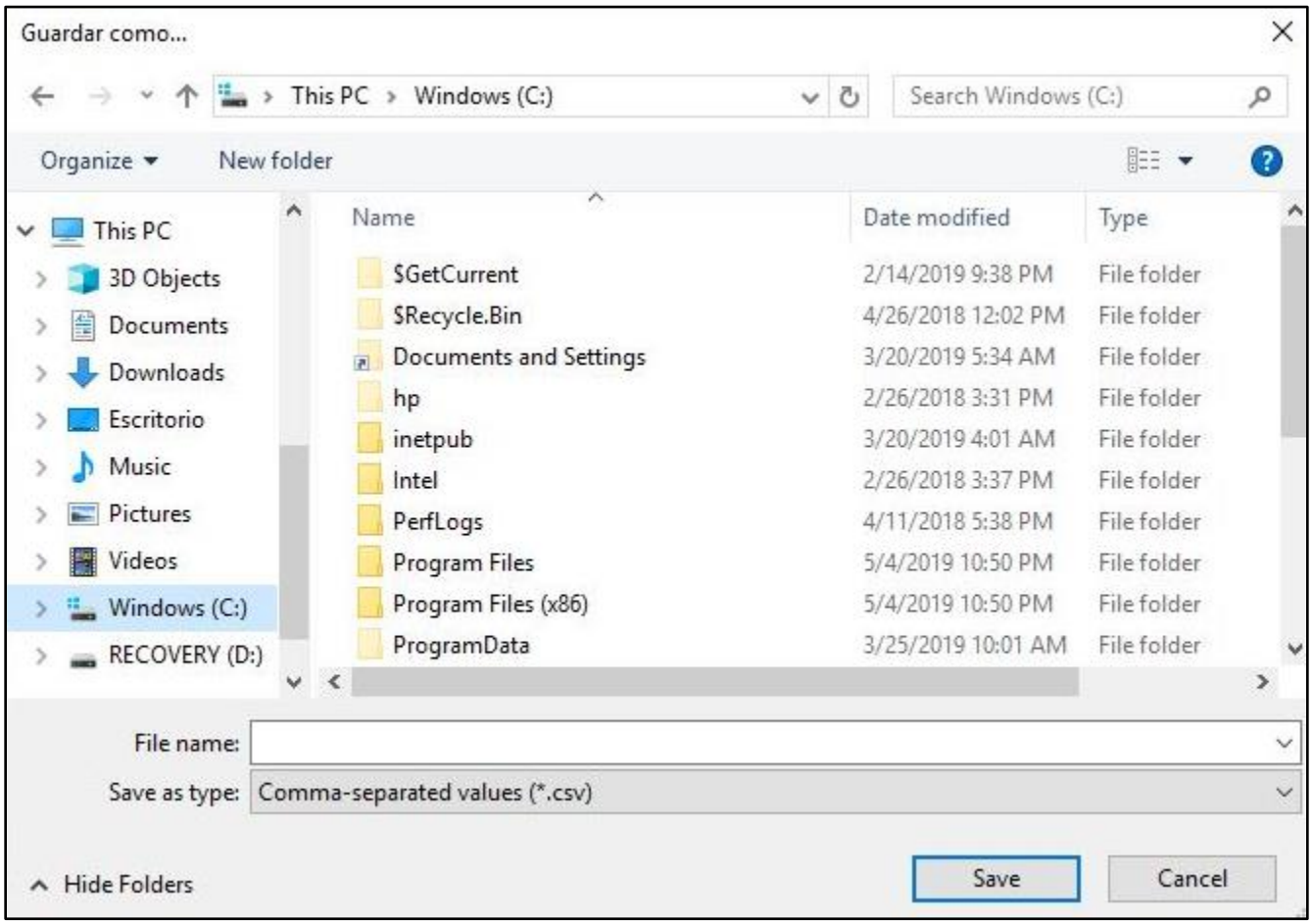
El menú principal consta de los 8 botones para las funciones del Registro de Personas; las cuáles se dividen en 2 partes que son la de Archivo (Abrir, Guardar como, Guardar y Salir) y la de Personas(Consultar, Insertar, Modificar y Eliminar); el menú principal también cuenta con una imagen proporcionada por el profesor que representa el registro de personas.

2) Abrir:



El botón “Abrir” despliega una pestaña que permite al usuario seleccionar el archivo .csv que desee abrir; para esto se usó el “open file dialog” que se encuentra en la librería de Visual Studio, a dicha ventana se le añadió un filtro para que solo permita abrir archivos separados por coma.

3) Guardar Como:



El botón “Guardar como” despliega una pestaña que permite al usuario seleccionar como desea guardar el archivo .csv; para esto se usó el “save file dialog” que se encuentra en la librería de Visual Studio, a dicha ventana se le añadió un filtro para que solo permita guardar archivos separados por coma.

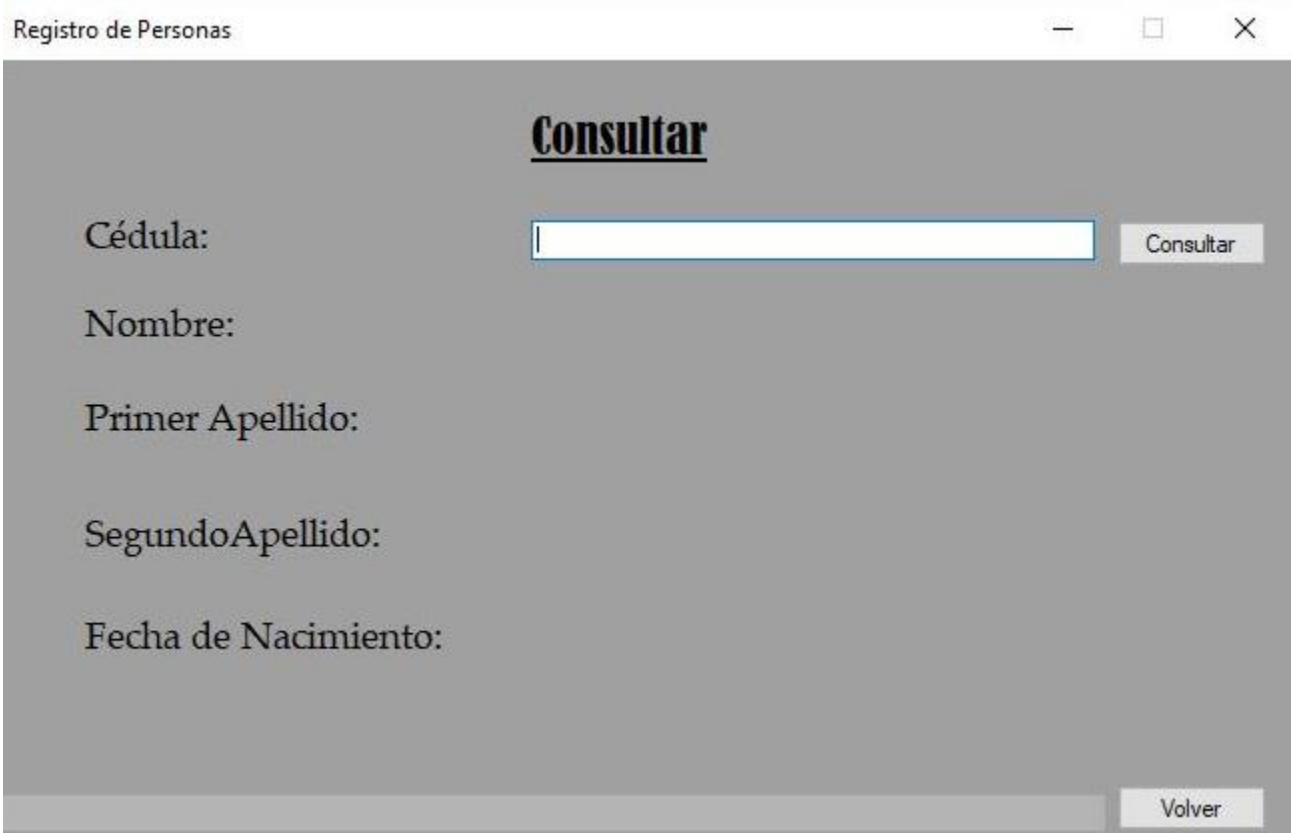
4) Guardar:

El botón “Guardar” del menú principal guarda en disco el registro de personas que se encontraba en memoria; si se utilizó recientemente el botón “Abrir” o el botón “Guardar como” se guardará el archivo con el mismo nombre y ruta que el que se encuentra en memoria recientemente; en caso de que no se hayan usado anteriormente ningún botón se sustituirá esta función por la del botón “Guardar como”.

5) Salir:

El botón “Salir” cierra el programa de Registro de Personas, los datos que se encuentren en memoria y no hayan sido guardados en disco anteriormente se borrarán.

6) Consultar:



El botón “Consultar” del menú principal despliega una pantalla que permitirá al usuario consultar la información de una Persona con solo insertar la cédula. Esta pantalla al presionar el botón con la etiqueta “Consultar” despliega la información deseada y en la parte inferior donde se encuentra una zona de mensajes despliega las colisiones o el error que se cometió al escribir la cédula. El botón con la etiqueta “Volver” devuelve al usuario al menú principal.

7) Insertar:

El botón “Insertar” del menú principal despliega una pestaña donde el usuario podrá agregar una Persona nueva al registro, para ello se deben llenar los campos y presionar el botón con la etiqueta “Insertar” el cual se encargará de validar los datos y en la parte inferior donde se encuentra la zona de mensajes desplegará en qué posición insertó la nueva Persona y las colisiones que se ocasionaron o en caso de que las validaciones resulten en error notificará al usuario el error cometido, en caso de que el usuario desee limpiar la ventana el botón con la etiqueta “Limpiar” devolverá a sus valores por defecto todas las entradas y la zona de mensajes. El botón con la etiqueta “Cancelar” devuelve al usuario al menú principal.

8) Modificar:

El botón “Modificar” del menú principal despliega una ventana que permite al usuario modificar los datos de alguna Persona, para esto se le solicita una cédula y tras presionar el botón con la etiqueta “Consultar” el cuál verificará que la cédula se encuentre en el registro y activará los demás campos, permitiéndole así al usuario que rellene los registros con la información deseada, el botón con la etiqueta “Limpiar” devuelve a sus valores iniciales todos los elementos de la ventana. El botón con la etiqueta “Modificar” verifica que los datos introducidos por el usuario sean correctos y modifica los datos de Persona; dependiendo del resultado de las verificaciones le notificará al usuario el resultado de la operación mediante la zona de mensajes. El botón con la etiqueta “Cancelar” devuelve al usuario al menú principal.

9) Eliminar:

El botón “Eliminar” del menú principal despliega una ventana que permite al usuario eliminar una Persona del registro; para ello se le solicita una cédula que al ingresarla y presionar el botón con la etiqueta “Consultar” desplegará la información de la Persona con dicha cédula, en caso de que el usuario desee continuar con la eliminación de dicha Persona se debe presionar el botón con la etiqueta “Eliminar” el cuál ejecutará la acción y lo notificará en la zona de mensajes situada en la parte inferior de la ventana. El botón con la etiqueta “Cancelar” devuelve al usuario al menú principal.

Resultados Obtenidos y Problemas Enfrentados

A. El programa cumple con los objetivos propuestos por el profesor, pues mantiene una base de datos de personas, usando una interfaz gráfica clásica proporcionada por la librería de Visual Studio, además de utilizar archivos de texto .csv con los que se guarda la base de datos en disco y mientras se mantiene en memoria los datos son administrados mediante una tabla de hash.
B. El programa cumple con las funciones CRUD solicitadas por el profesor además de las opciones de guardar y abrir archivos para permitirle al usuario guardar en disco su registro de personas.
C. En cuanto a los problemas enfrentados se encuentran los problemas con la librería de Visual Studio para crear “Windows Forms”, pues posee varios bugs que obligan al programador a insertar propiedades y código por defecto, en lo que no se poseía conocimiento alguno. Además de los problemas con las ventanas de “Abrir” y “Guardar como” que originan problemas de “nullpointer” con el guardado de la ruta.

Conclusiones

A. Se concluye que los objetivos propuestos para el proyecto n° 2 de Estructuras de Datos se alcanzaron de forma exitosa, pues el programa permite al usuario tener un registro de personas con todas las opciones CRUD y de apertura y guardado de archivos.
B. Se concluye que los estudiantes aprendieron a implementar una interfaz gráfica de tipo clásico con la librería que proporciona Visual Studio.
C. Se concluye que los estudiantes ampliaron su conocimiento del uso de archivos de texto en el lenguaje de programación C++, pues la base de datos de personas se lee y guarda en archivos de texto .csv.
D. Se concluye que los estudiantes ampliaron su conocimiento en el uso de tablas de hash pues los datos almacenados en el registro de personas se administran mediante una tabla de hash.
E. Se concluye que los estudiantes fueron capaces de coordinar el desarrollo del proyecto mediante el controlador de versiones GitHub con su extensión para Visual Studio.