

**“Entregable 05. Implementación de operaciones II: Consultas, parte 1”**

**Juan Antonio Ramírez Aguilar**

**(212482507)**

**Seminario de Solución de Problemas de Estructuras de Datos II**

**Mtro. Alfredo Gutiérrez Hernández**

**Clave: I5889 Sección: D13**

**15/09/2023**

**Introducción**

En el proceso de desarrollo de un sistema de gestión de consultas médicas, me enfrenté al desafío de importar datos desde un archivo de respaldo. Este archivo contenía información crucial sobre consultas médicas, incluyendo detalles sobre pacientes, médicos, diagnósticos y medicamentos. Sin embargo, el formato del archivo no siempre coincidía con la estructura esperada, lo que planteó un desafío significativo en el proceso de importación y procesamiento de datos.

Para abordar este problema, implementé una serie de estrategias y modificaciones en el código para manejar escenarios donde el formato de los datos era inesperado. Esto incluyó la creación de lógica adicional para validar y adaptar la entrada según las circunstancias. Además, se introdujeron técnicas para evitar fallos inesperados y garantizar la integridad de los datos importados.

A lo largo de este proceso, se realizaron pruebas exhaustivas para verificar la robustez de las soluciones implementadas y se llevaron a cabo ajustes según los resultados obtenidos. A continuación, se presentan los detalles específicos de cómo se abordó cada uno de los desafíos y los resultados alcanzados.

**Implementación**

Para este programa adapte una de las funciones de la clase archivoConsultas para poder cargar los datos del archivo a una lista donde se pueda almacenar dentro del programa.

class ArchivoConsulta {

    private:

        std::fstream archivo;

        std::fstream indicePorNombre; // int, int

        Lista<indexDuple<int>> indexadoPorNombre;

        void reIndex();

        template <class T>

        Lista<indexDuple<T>> fileToList(std::fstream, Lista<indexDuple<T>>);

        template <class T>

        Lista<indexDuple<T>> listToFile(const Lista<indexDuple<T>>&, std::fstream&);

        template <class T>

        int getIndex(const Lista<T>&, const T&);

    public:

        ArchivoConsulta();

        ~ArchivoConsulta();

        void addData(const Consulta&);

        void addData(const Lista<Consulta>&);

        void delData(const int&);

        int findData(const Consulta&);

        void clear();

        void compress();

        Lista<Consulta> importBackup(std::string&); // Se uso esta funcion.

        void exportBackup(const std::string&);

};

Esta función se implente de la sifuiente manera:

Lista<Consulta> ArchivoConsulta::importBackup(std::string& nombreArchivo) {

    std::ifstream archivoEntrada;

    archivoEntrada.open(nombreArchivo, std::ios::in);

    Lista<Consulta> miLista;

    if (!archivoEntrada) {

        throw std::runtime\_error("Error al abrir el archivo.");

    }

    std::string miCampo;

    while(!archivoEntrada.eof()) {

        std::getline(archivoEntrada, miCampo, '#');

        if (miCampo.empty()) { continue; }

        // Se instancian las clases necesarias

        Nombre miNombre;

        Fecha miFecha;

        Hora miHora;

        Domicilio miDomicilio;

        Paciente miPaciente;

        Medico miMedico;

        Medicamento miMedicamento;

        Diagnostico miDiagnostico;

        Consulta miConsulta;

        std::string codigo;

        std::stringstream flujoDeRegistro(miCampo);

        // Toma la fecha de la consulta

        flujoDeRegistro >> miFecha;

        miConsulta.setFecha(miFecha);

        // Toma la hora de la consulta

        flujoDeRegistro >> miHora;

        miConsulta.setHora(miHora);

        // Toma el codigo de la consulta

        std::getline(flujoDeRegistro, miCampo, '\*');

        miConsulta.setCodigo(miCampo);

        // Toma el Diagnostico de la consulta

        flujoDeRegistro >> miDiagnostico;

        miConsulta.setDiagnostico(miDiagnostico);

        // Toma el Medico de la consulta;

        flujoDeRegistro >> miMedico;

        miConsulta.setMedico(miMedico);

        // Toma el Paciente de la consulta;

        flujoDeRegistro >> miPaciente;

        miConsulta.setPaciente(miPaciente);

        // Toma el Medicamento de la consulta;

        flujoDeRegistro >> miMedicamento;

        miConsulta.setMedicamento1(miMedicamento);

        flujoDeRegistro >> miMedicamento;

        miConsulta.setMedicamento2(miMedicamento);

        flujoDeRegistro >> miMedicamento;

        miConsulta.setMedicamento3(miMedicamento);

        // Se ingresa en la lista

        miLista.insertar(miConsulta);

    }

    archivoEntrada.close();

    return miLista;

}

La función toma el nombre del archivo y después trata de abrirlo, cuando lo abre, entonces empieza a integrar cada una de las partes de la consulta para poder luego ingresarlas a una lista que va a retornar después.

Una vez integrado la lista y devuelto, cree una clase para el menú de las consultas con las cuales puedo interactuar con la lista:

#ifndef MENUCONSULTAS\_H

#define MENUCONSULTAS\_H

#include "Consulta.h"

#include "ArchivoConsulta.h"

#include "Lista.h"

#include <string>

class MenuConsultas {

    private:

        std::string opc;

        bool existeImporte;

        Lista<Consulta> consultas;

        ArchivoConsulta archivo;

    public:

        MenuConsultas();

        ~MenuConsultas();

        void setOpc(std::string valorOpc);

        void setExisteImporte(bool valorBool);

        void setLista(Lista<Consulta> list);

        std::string getOpc();

        bool getExisteImporte();

        Lista<Consulta> getConsultas();

        void menu();

        void buscarRegistro();

        void importar();

        void imprimirTodo();

        void salir();

};

#endif

Implementación de la clase:

#include "menuConsultas.h"

#include "StandarLibrary.h"

#include "Colores.h"

#ifdef \_WIN32

#define CLEAR "cls"

#else

#define CLEAR "clear"

#endif

#define ALTURA\_BORDE 100

#define ANCHURA\_BORDE 150

#define ANCHURA\_PANTALLA 1366

#define ALTURA\_PANTALLA 768

const std::string Titulo = "Sistema Integral de Registros Medicos";

const std::string Subtitulo = "Menu de Consultas";

MenuConsultas::MenuConsultas() : opc("x"), existeImporte(false) {}

MenuConsultas::~MenuConsultas() {}

void MenuConsultas::setOpc(std::string valorOpc) { opc = valorOpc; }

void MenuConsultas::setExisteImporte(bool valorBool) { existeImporte = valorBool; }

void MenuConsultas::setLista(Lista<Consulta> list) { consultas = list; }

std::string MenuConsultas::getOpc() { return opc; }

bool MenuConsultas::getExisteImporte() { return existeImporte; }

Lista<Consulta> MenuConsultas::getConsultas() { return consultas; }

void MenuConsultas::menu() {

    std::string opciones;

    AltEnter();

    do {

        system(CLEAR);

        std::cout<<ARB; setborder(ALTURA\_BORDE, ANCHURA\_BORDE); std::cout<<RTNC;

        gotoxy(((ANCHURA\_BORDE - int(Titulo.length())) / 2), 3);

        std::cout << VB << Titulo <<RTNC;

        gotoxy(((ANCHURA\_BORDE - int(Subtitulo.length())) / 2), 4);

        std::cout << ARB << Subtitulo <<RTNC;

        gotoxy(3, 7);

        std::cout <<GB; std::cout << "Opciones del Menu:";

        gotoxy(3, 9);

        std::cout <<GB; std::cout << "Buscar una consulta por indice. [ " <<RF<< "A" <<GB " ]";

        gotoxy(3, 10);

        std::cout <<GB; std::cout << "Imprimir todos los datos.       [ " <<RF<< "B" <<GB " ]";

        gotoxy(3, 11);

        std::cout <<GB; std::cout << "Importar archivo.               [ " <<RF<< "C" <<GB " ]";

        gotoxy(3, 12);

        std::cout <<GB; std::cout << "Salir del programa.             [ " <<RF<< "X" <<GB " ]";

        gotoxy(3, 14);

        std::cout <<GB << "Tu opcion ==> ";

        gotoxy(18, 14);

        std::cout <<RF; std::getline(std::cin, opciones); setOpc(opciones); std::cout <<RTNC;

        if (opc == "A" or opc == "a") { buscarRegistro(); }

        else if (opc == "B" or opc == "b") { imprimirTodo(); }

        else if (opc == "C" or opc == "c") { importar(); }

        else if (opc == "X" or opc == "x") { salir(); }

        else { gotoxy(3, 16); std::cout <<GB; std::cout << "La opcion: \"" <<RB<< opc <<GB<< "\" no es una opcion valida..."<<RTNC; pausa(); }

    } while (opc != "X" and opc != "x");

}

void MenuConsultas::buscarRegistro() {

    if (!existeImporte) {

        gotoxy(3, 16);

        std::cout <<RB; std::cout << "Primero importa un archivo..."<<RTNC; pausa();

    } else {

        const std::string subtiImport = "Consulta por indice";

        int indice = -1;

        while (indice < 0 or indice >= consultas.length()) {

            system(CLEAR);

            std::cout<<ARB; setborder(ALTURA\_BORDE, ANCHURA\_BORDE); std::cout<<RTNC;

            gotoxy(((ANCHURA\_BORDE - int(Titulo.length())) / 2), 3);

            std::cout << VB << Titulo <<RTNC;

            gotoxy(((ANCHURA\_BORDE - int(Subtitulo.length())) / 2), 4);

            std::cout << ARB << Subtitulo <<RTNC;

            gotoxy(((ANCHURA\_BORDE - int(subtiImport.length())) / 2), 5);

            std::cout << AB << subtiImport <<RTNC;

            gotoxy(3, 7);

            std::cout <<GB; std::cout << "Dame el indice del registro que quiereas ver."<<RTNC;

            gotoxy(3, 8);

            std::cout <<GB; std::cout << "Indice del registro ==>  "; std::cout<<RTNC;

            gotoxy(29, 8);

            std::cout <<RF; std::cin >> indice; std::cout <<RTNC;

            if (indice < 0 or indice >= consultas.length()) {

                gotoxy(3, 10);

                std::cout <<RB; std::cout << "Indice fuera de Rango..."<<RTNC; pausa();

            } else {

                gotoxy(3, 10);

                std::cout <<GB; std::cout << "Registro Encontrado."<<RTNC;

                gotoxy(3, 11);

                std::cout <<VB<< indice << ").\_ " <<RTNC<<std::endl;

                gotoxy(3, 12);

                std::cout <<AQB<< "Codigo de Consulta: " << GB << consultas.indice(indice).getCodigo() <<RTNC<<std::endl;

                gotoxy(3, 13);

                std::cout <<AF<< "Nombre Paciente: " << GB << consultas.indice(indice).getPaciente().getNombre().toString() <<RTNC<<std::endl;

                gotoxy(3, 14);

                std::cout <<ARF<< "Nombre del Medico: " << GB << consultas.indice(indice).getMedico().getNombre().toString() <<RTNC<<std::endl<<std::endl;

                gotoxy(3, 15);

                std::cout <<GB<< std::endl << "Presiona Enter para continuar..."; std::cout<<RTNC; pausa();

            }

        }

    }

}

void MenuConsultas::importar() {

    std::string fileName;

    const std::string subtiImport = "Importar archivo";

    do {

        system(CLEAR);

        std::cout<<ARB; setborder(ALTURA\_BORDE, ANCHURA\_BORDE); std::cout<<RTNC;

        gotoxy(((ANCHURA\_BORDE - int(Titulo.length())) / 2), 3);

        std::cout << VB << Titulo <<RTNC;

        gotoxy(((ANCHURA\_BORDE - int(Subtitulo.length())) / 2), 4);

        std::cout << ARB << Subtitulo <<RTNC;

        gotoxy(((ANCHURA\_BORDE - int(subtiImport.length())) / 2), 5);

        std::cout << AB << subtiImport <<RTNC;

        gotoxy(3, 7);

        std::cout <<GB; std::cout << "Dame el nombre del archivo a importar (o si quieres salir ingresa \"X\").";

        gotoxy(3, 8);

        std::cout <<GB; std::cout << "Nombre del Archivo ==>  "; std::cout<<RTNC;

        gotoxy(28, 8);

        std::cout <<RF; std::getline(std::cin, fileName); std::cout <<RTNC;

        if (fileName != "X" and fileName != "x") {

            gotoxy(3, 9);

            std::cout <<VB; std::cout << "Leyendo el archivo..."; std::cout<<RTNC;

            consultas = archivo.importBackup(fileName);

            gotoxy(3, 10);

            std::cout <<VB; std::cout << "Archivo leido..."; std::cout<<RTNC;

            gotoxy(3, 11);

            std::cout <<GB; std::cout << "Presiona Enter para continuar..."; std::cout<<RTNC; pausa();

            existeImporte = true;

            fileName = "X";

        }

    } while (fileName != "X" and fileName != "x");

}

void MenuConsultas::imprimirTodo() {

    if (!existeImporte) {

        gotoxy(3, 16);

        std::cout <<RB; std::cout << "Primero importa un archivo..."<<RTNC; pausa();

    } else {

        const std::string subtiImport = "Imprimir todo";

        system(CLEAR);

        std::cout<<ARB; setborder(ALTURA\_BORDE, ANCHURA\_BORDE); std::cout<<RTNC;

        gotoxy(((ANCHURA\_BORDE - int(Titulo.length())) / 2), 3);

        std::cout << VB << Titulo <<RTNC;

        gotoxy(((ANCHURA\_BORDE - int(Subtitulo.length())) / 2), 4);

        std::cout << ARB << Subtitulo <<RTNC;

        gotoxy(((ANCHURA\_BORDE - int(subtiImport.length())) / 2), 5);

        std::cout << AB << subtiImport <<RTNC;

        gotoxy(3, 7);

        std::cout <<GB; std::cout << "Se imprimiran todos los datos del programa."<<RTNC;  pausa();

        system(CLEAR);

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

            std::cout <<VB<< i + 1 << ").\_ " <<RTNC<<std::endl;

            std::cout <<AQB<< "Codigo de Consulta: " << GB << consultas.indice(i).getCodigo() <<RTNC<<std::endl;

            std::cout <<AF<< "Nombre Paciente: " << GB << consultas.indice(i).getPaciente().getNombre().toString() <<RTNC<<std::endl;

            std::cout <<ARF<< "Nombre del Medico: " << GB << consultas.indice(i).getMedico().getNombre().toString() <<RTNC<<std::endl<<std::endl;

            std::cout <<GB; std::cout << std::endl << "Presiona Enter para continuar..."; std::cout<<RTNC; pausa();

        }

        std::cout <<GB<< std::endl << "Presiona Enter para continuar..."; std::cout<<RTNC; pausa();

    }

}

void MenuConsultas::salir() {

    gotoxy(3, 16);

    std::cout <<RB; std::cout << "Saliendo del programa..."<<RTNC; timeStop(2000);

}

Como se puede ver utilice dos librerías de mi autoría para darle un poco de diseño a el programa, estos son StandarLibrary.h y Colores.h que todavía no son el producto final, espero poder mejorarlas para implementarlas mejor.

**StandarLibrary:**

#ifndef STANDARLIBRARY\_H\_INCLUDED

#define STANDARLIBRARY\_H\_INCLUDED

#include <iostream>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

/\* Pausa del programa portable \*/

void pausa () {

    std::cin.clear();

    std::cin.sync();

    std::cin.ignore();

}

void timeStop(int Milsec) {

    Sleep(Milsec);

}

/\* Alternativa portable a fflush \*/

void limpiaBuffer (void) {

    int c;

    do {

        c = getchar();

    } while (c != '\n' && c != EOF);

}

// Gotoxy

void gotoxy (int x,int y) {

    printf("%c[%d;%df",0x1B,y,x);

}

/\* Marco del programa(Cuadrado) \*/

void setborder (int altura, int anchura) {

    int columnaX, FilaY;

    for (columnaX = 1; columnaX <= anchura; columnaX++) {

        gotoxy(columnaX, 0);

        putchar(char(223));

        gotoxy(columnaX, altura);

        putchar(char(220));

    }

    for (FilaY = 1; FilaY <= altura; FilaY++) {

        gotoxy(0, FilaY);

        putchar(char(219));

        gotoxy(anchura, FilaY);

        putchar(char(219));

    }

}

void AltEnter()

{

    keybd\_event(VK\_MENU,

                0x38,

                0,

                0);

    keybd\_event(VK\_RETURN,

                0x1c,

                0,

                0);

    keybd\_event(VK\_RETURN,

                0x1c,

                KEYEVENTF\_KEYUP,

                0);

    keybd\_event(VK\_MENU,

                0x38,

                KEYEVENTF\_KEYUP,

                0);

    return;

}

#endif // STANDARLIBRARY\_H\_INCLUDED

**Colores:**

#ifndef Colores

#define Colores

#include <stdio.h>

#include <stdlib.h>

/// Todos los colores que hay para cambiar un printf en especifico

#define RTNC "\x1b[0m"

#define NB "\x1b[1;30m"

#define NF "\x1b[0;30m"

#define NEGROFONDO "\x1b[1;30;40m"

#define RB "\x1b[1;31m"

#define RF "\x1b[0;31m"

#define ROJOFONDO "\x1b[1;31;41m"

#define VB "\x1b[1;32m"

#define VF "\x1b[0;32m"

#define VERDEFONDO "\x1b[1;32;42m"

#define AB "\x1b[1;33m"

#define AF "\x1b[0;33m"

#define AMARILLOFONDO "\x1b[1;33;43m"

#define ARB "\x1b[1;34m"

#define ARF "\x1b[0;34m"

#define AZULREYFONDO "\x1b[1;34;44m"

#define MB "\x1b[1;35m"

#define MF "\x1b[0;35m"

#define MORADOFONDO "\x1b[1;35;45m"

#define AQB "\x1b[1;36m"

#define AQF "\x1b[0;36m"

#define AQUAFONDO "\x1b[1;36;46m"

#define GB "\x1b[1;37m"

#define GF "\x1b[0;37m"

#define GRISFONDE "\x1b[1;37;47m"

#endif // Colores

EL main quedaría simplemente asi:

#include <iostream>

#include "menuConsultas.h"

using namespace std;

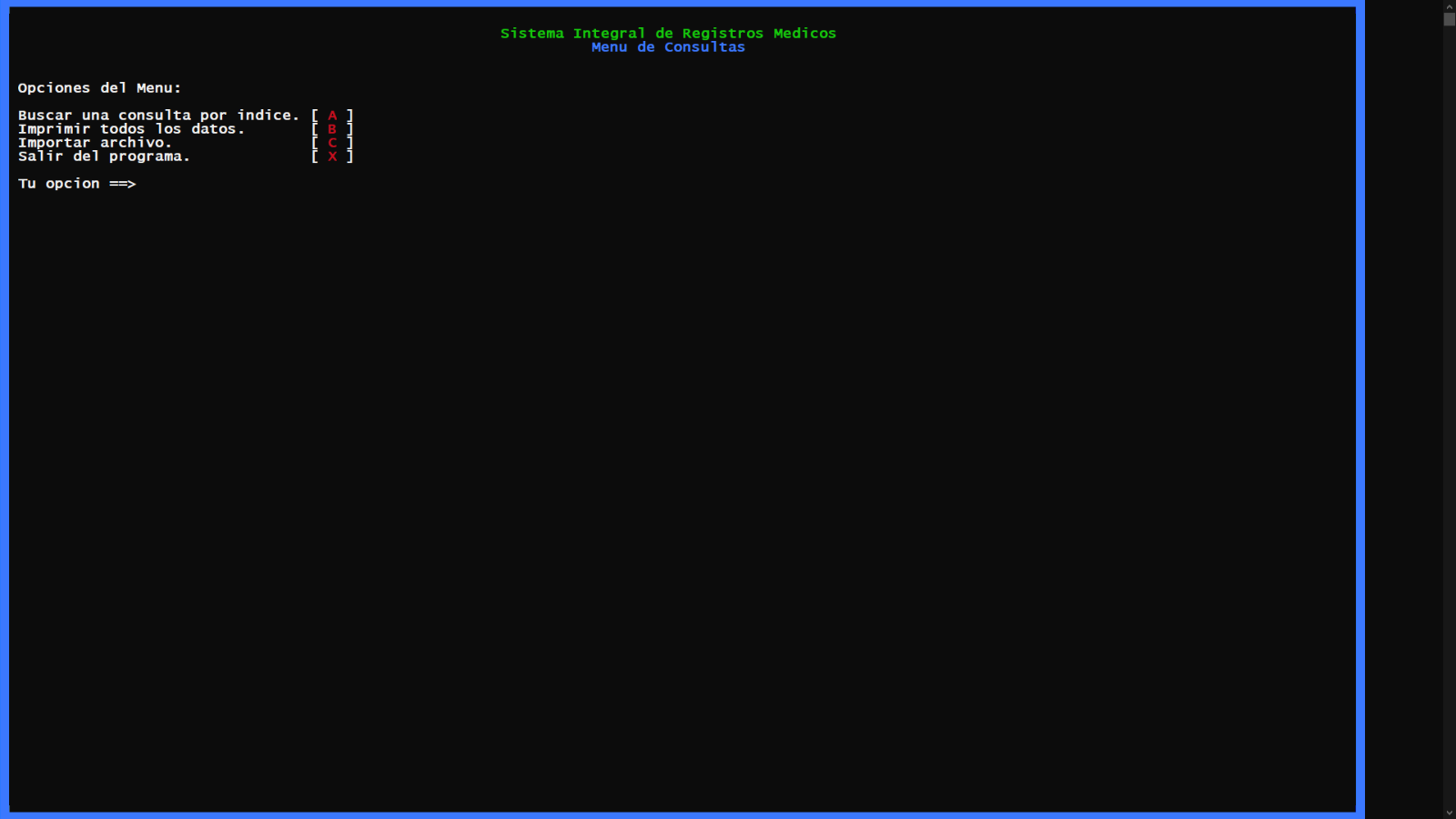
int main () {

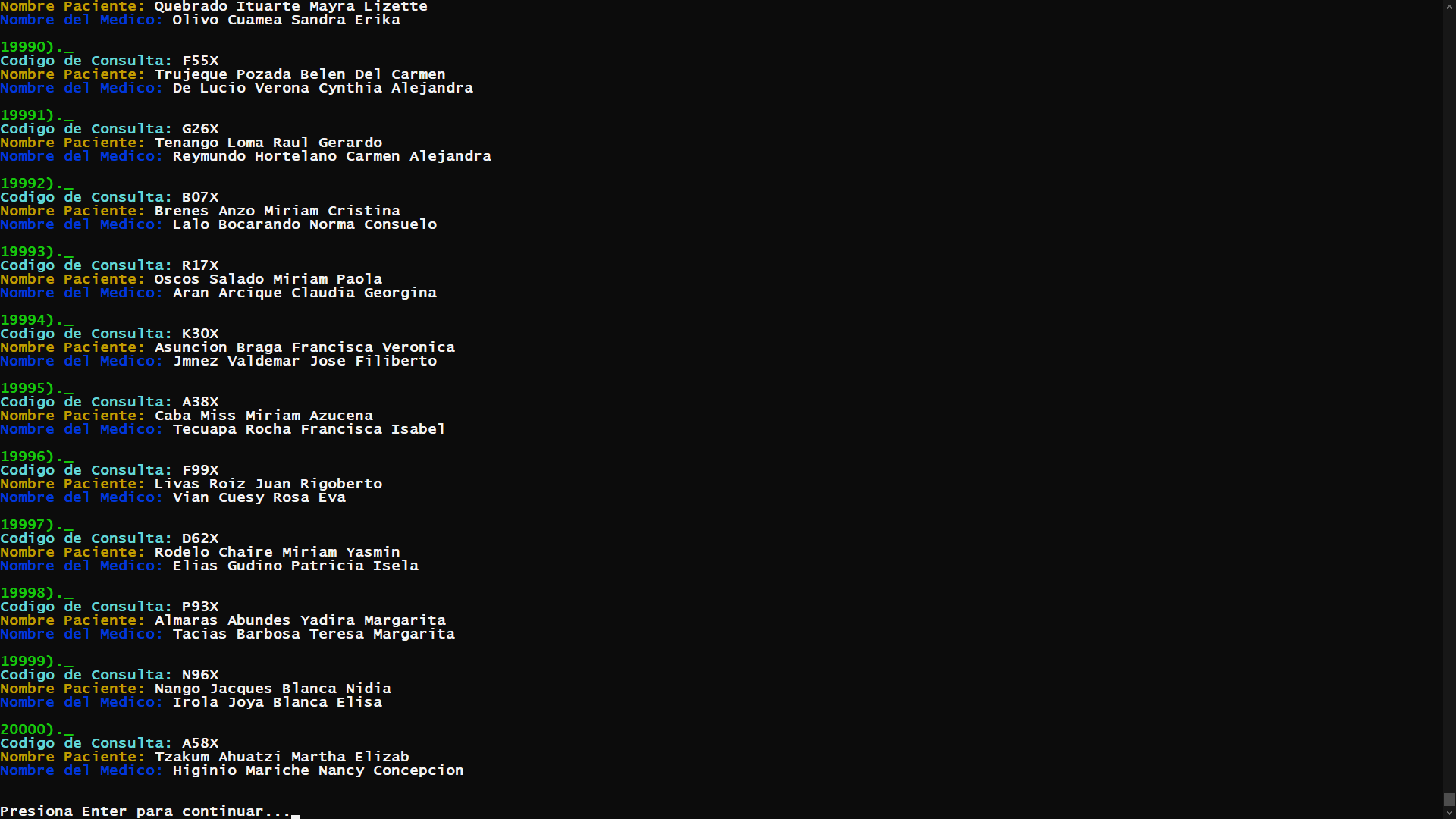
    MenuConsultas menuConsultas;

    menuConsultas.menu();

    return 0;

}

**Resultado:**



**Conclusiones:**

Durante este proyecto, enfrentamos desafíos al importar datos de un archivo de respaldo para un sistema de gestión de consultas médicas. Al abordar estos desafíos, adaptamos y refinamos el código para manejar diferentes formatos de datos y situaciones inesperadas.

Implementamos soluciones específicas, como validaciones adicionales y ajustes de datos, para garantizar una importación confiable. Realizamos pruebas exhaustivas para asegurarnos de que el sistema maneje una amplia gama de situaciones de datos.