

**“Entregable 08. Implementación de operaciones V, Implementación de interfaces de usuario III”**

**Juan Antonio Ramírez Aguilar**

**(212482507)**

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

**Mtro. Alfredo Gutiérrez Hernández**

**Clave: I5889 Sección: D13**

**Fecha de Elaboración: 02/10/2023**

### Introducción

En este entregable implemente el archivo para medicamentos, así como su menú propio, al crearse el archivo principal, también se crea un archivo secundario que guarda los índices dentro del archivo de los medicamentos tomando en cuenta el código de cada uno de ellos.

El menú es exactamente igual a los anteriores con las diferencias de que al solo manejar código y nombre del medicamento; es más sencillo.

La función agregar, eliminar y buscar están funcionando de manera correcta y sin ningún error aparente.

### Código del Programa

**archivoMedicamento.h**

#ifndef ARCHIVOMEDICAMENTO\_H

#define ARCHIVOMEDICAMENTO\_H

#include <fstream>

#include <string>

#include <sstream>

#include "Lista.h"

#include "Medicamento.h"

#include "indexDuple.h"

class archivoMedicamento {

    private:

        std::fstream archivo;

        std::fstream archivoPorCodigo;

        std::string nameFileMedicamento;

        std::string indexCodeFileMedicamento;

        Lista<indexDuple<>> indexadoPorCodigo;

        void reIndex();

        template <class T>

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

        template <class T>

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

        template <class T>

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

    public:

        archivoMedicamento();

        ~archivoMedicamento();

        void addData(const Medicamento&);

        void addData(Lista<Medicamento>&);

        Medicamento getData(const int&);

        void delData(const int&);

        int findData(const std::string&) const;

        Lista<Medicamento> toList();

        void clear();

        void compress();

        void importBackup();

        void exportBackup(const std::string&);

};

#endif // ARCHIVOMEDICAMENTO\_H

**archivoMedicamento.cpp**

#include "archivoMedicamento.h"

archivoMedicamento::archivoMedicamento() : nameFileMedicamento("archivoMedicamentos.txt"),

                                           indexCodeFileMedicamento("archivoMedicamentoCodigo.txt") {}

archivoMedicamento::~archivoMedicamento() {

    if (archivo.is\_open()) {

        archivo.close();

    }

    if (archivoPorCodigo.is\_open()) {

        archivoPorCodigo.close();

    }

}

void archivoMedicamento::reIndex() {

    string myString;

    int myIndex;

    Medicamento miMedicamento;

    indexDuple<> miTupleCodigo;

    archivo.open(nameFileMedicamento, ios\_base::in);

    if (!archivo.is\_open()) {

        throw ios\_base::failure("Error al cargar el archivo en archivoMedicamento::reIndex()");

    }

    indexadoPorCodigo.anular();

    while (!archivo.eof()) {

        myIndex = int(archivo.tellg());

        getline(archivo, myString, '#');

        if (myString.empty()) {

            continue;

        }

        stringstream myStrString(myString);

        getline(myStrString, myString, '\*');

        if (myString == "1") {

            myStrString >> miMedicamento;

            miTupleCodigo.setIndex(myIndex);

            miTupleCodigo.setData(miMedicamento.getCodigo());

            indexadoPorCodigo.insertar(miTupleCodigo);

        }

    }

        // Se pasa la lista de Tuplas "Codigo" a su archivo.

        archivoPorCodigo.open(indexCodeFileMedicamento, ios\_base::out | ios\_base::trunc);

        if (!archivoPorCodigo.is\_open()) {

            throw ios\_base::failure("Error al cargar el archivo en archivoMedicamento::reIndex()");

        }

        listToFile(indexadoPorCodigo, archivoPorCodigo);

        archivoPorCodigo.close();

        archivo.close();

}

template<class T>

Lista<T> archivoMedicamento::fileToList(std::fstream& myFStream, Lista<T> miLista) {

    string myStr;

    T data;

    while (!myFStream.eof()) {

        getline(myFStream, myStr, '#');

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

        stringstream myStringStream(myStr);

        getline(myStringStream, myStr, '\*');

        if (myStr == "1") {

            myStringStream >> data;

            miLista.insertar(data);

        }

    }

    return miLista;

}

template<class T>

bool archivoMedicamento::listToFile(const Lista<T>& myLista, std::fstream & myFstream) {

    clear();

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

        myFstream << "1\*" << myLista.indice(i) << "#";

    }

    return true;

}

template<class T>

int archivoMedicamento::getIndex(const Lista<indexDuple<T>>& miLista, const T& e) const {

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

        if (miLista.indice(i).getData() == e) {

            return miLista.indice(i).getIndex();

        }

    }

    return -1;

}

void archivoMedicamento::addData(const Medicamento & medicamentoToAdd) {

    Lista<Medicamento> myList;

    myList.insertar(medicamentoToAdd);

    addData(myList);

}

void archivoMedicamento::addData(Lista<Medicamento>& myList) {

    archivo.open(nameFileMedicamento, ios\_base::out | ios\_base::app);

    if (!archivo.is\_open()) {

        throw ios\_base::failure("Error al cargar el archivo en archivoMedicamento::addData()");

    }

    listToFile(myList, archivo);

    archivo.close();

    reIndex();

}

Medicamento archivoMedicamento::getData(const int & idx) {

    string myString;

    Medicamento miMedicamento;

    archivo.open(nameFileMedicamento);

    if (!archivo.is\_open()) {

        throw ios\_base::failure("Error al cargar el archivo en archivoMedicamento::getData()");

    }

    archivo.seekg(idx);

    getline(archivo, myString, '#');

    if (myString.empty()) {

        throw ios\_base::failure("Dato inexistente.");

    }

    stringstream myStrString(myString);

    getline(myStrString, myString, '\*');

    if (myString == "0") {

        throw ios\_base::failure("Dato borrado.");

    }

    myStrString >> miMedicamento;

    archivo.close();

    return miMedicamento;

}

void archivoMedicamento::delData(const int& idx) {

    archivo.open(nameFileMedicamento, ios\_base::in | ios\_base::out);

    if (!archivo.is\_open()) {

        throw ios\_base::failure("Error al cargar el archivo en archivoMedicamento::delData()");

    }

    archivo.seekp(idx);

    archivo << "0";

    archivo.close();

    reIndex();

}

int archivoMedicamento::findData(const string& code) const {

    return getIndex(indexadoPorCodigo, code);

}

Lista<Medicamento> archivoMedicamento::toList() {

    archivo.open(nameFileMedicamento, ios\_base::in);

    if (!archivo.is\_open()) {

        throw ios\_base::failure("Error al cargar el archivo en archivoMedicamento::toList()");

    }

    Lista<Medicamento> listaAux;

    listaAux = fileToList(archivo, listaAux);

    archivo.close();

    return listaAux;

}

void archivoMedicamento::clear() {

    remove(nameFileMedicamento.c\_str());

}

void archivoMedicamento::compress() {

    string tempFileName = "temp.file";

    // Se renombra el archivo "archivoMedicos" a "temp.file".

    rename(nameFileMedicamento.c\_str(), tempFileName.c\_str());

    // Se abre el archivo original en modo escritura.

    archivo.open(nameFileMedicamento, ios\_base::out | ios\_base::trunc);

    // Se abre el archivo temporal en modo lectura.

    ifstream tempFile(tempFileName.c\_str());

    // Revisa si ninguno de los archivos se abrió correctamente.

    if (!tempFile.is\_open() or !archivo.is\_open()) {

        throw ios\_base::failure("Error al abrir el archivo.");

    }

    string myString;

    // Bucle para obtener todos los registros del archivo temporal.

    while (getline(tempFile, myString, '#')) {

        // Si el registro tiene un 0 al inicio, no se toma en cuenta.

        if (myString.empty() or myString[0] == '0') {

            continue;

        }

        // Se vuelve a escribir el dato que sí es válido en el archivo original.

        archivo << myString << "#";

    }

    // Se cierran ambos archivos.

    tempFile.close();

    archivo.close();

    // Se elimina el archivo temporal.

    remove(tempFileName.c\_str());

    // Se hace un reIndex del archivo.

    reIndex();

}

void archivoMedicamento::importBackup() {

    reIndex();

}

**menuMedicamento.h**

#ifndef MENUMEDICAMENTO\_H

#define MENUMEDICAMENTO\_H

#include <iostream>

#include <string>

#include "archivoMedicamento.h"

class menuMedicamento {

    private:

        std::string opc;

        archivoMedicamento miArchivoMedicamento;

    public:

        menuMedicamento();

        ~menuMedicamento();

        void setOpc(std::string valorOpc);

        std::string getOpc();

        void menu();

        void agregar();

        void eliminar();

        void buscar();

        void importarArchivo();

};

#endif // MENUMEDICAMENTO\_H

**menuMedicamento.cpp**

#include "menuMedicamento.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 Medicamentos";

menuMedicamento::menuMedicamento() {}

menuMedicamento::~menuMedicamento() {}

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

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

void menuMedicamento::menu() {

    std::string opciones;

    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 << "Agregar un Medicamento.        [ " <<RF<< "A" <<GB " ]";

        gotoxy(3, 10);

        std::cout <<GB; std::cout << "Elimnar un Medicamento.        [ " <<RF<< "B" <<GB " ]";

        gotoxy(3, 11);

        std::cout <<GB; std::cout << "Buscar un Medicamento.         [ " <<RF<< "C" <<GB " ]";

        gotoxy(3, 12);

        std::cout <<GB; std::cout << "Regresar al menu anterior.     [ " <<RF<< "X" <<GB " ]";

        gotoxy(3, 14);

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

        gotoxy(18, 14);

        fflush(stdin);

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

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

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

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

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

        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 menuMedicamento::agregar() {

    Medicamento nuevoMedicamento;

    std::string auxString;

    const std::string subtiImport = "Agregar un Medicamento";

    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 << GF << "Ingresar un Medicamento nuevo al archivo." <<RTNC;

    gotoxy(3, 9);

    std::cout << GF << "Dame el codigo del medicamento:" <<RTNC;

    gotoxy(35, 9);

    fflush(stdin);

    std::getline(cin, auxString);

    nuevoMedicamento.setCodigo(auxString);

    gotoxy(3, 11);

    std::cout << GF << "Dame el nombre del medicamento:" <<RTNC;

    fflush(stdin);

    gotoxy(35, 11);

    std::getline(cin, auxString);

    nuevoMedicamento.setNombre(auxString);

    miArchivoMedicamento.addData(nuevoMedicamento);

    gotoxy(3, 13);

    std::cout << VB << "Medicamento Registrado en el archivo..." <<RTNC; pausa();

}

void menuMedicamento::eliminar() {

    std::string auxCodigo;

    system(CLEAR);

    const std::string subtiImport = "Eliminar un Medicamento";

    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 <<GF<< "Ingresa el codigo del medicamento a buscar:";

    gotoxy(47, 7);

    fflush(stdin);

    getline(std::cin, auxCodigo);

    int idx = miArchivoMedicamento.findData(auxCodigo);

    if (idx != -1) {

        string opcBorrar;

        Medicamento showMedicamento = miArchivoMedicamento.getData(idx);

        gotoxy(3, 11);

        std::cout << "Medicamento: " << std::endl;

        gotoxy(3, 12);

        std::cout << AF << showMedicamento.toString();

        gotoxy(3, 15);

        std::cout << ARB << "Seguro que lo quieres borrar?" << RTNC <<std::endl;

        gotoxy(3, 16);

        std::cout << GB << "Si.  [ " <<RF<< "Y" <<GB " ]";

        gotoxy(3, 17);

        std::cout << GB << "No.  [ " <<RF<< "N" <<GB " ]";

        gotoxy(3, 18);

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

        gotoxy(18, 18);

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

        if (opcBorrar == "Y" or opcBorrar == "y") {

            miArchivoMedicamento.delData(idx);

            miArchivoMedicamento.compress();

            gotoxy(3, 20);

            std::cout <<VB<< "Medicamento eliminado correctamente."<<RTNC; pausa();

        } else if(opcBorrar == "N" or opcBorrar == "n" or opcBorrar != "Y" or opcBorrar != "y") {

            gotoxy(3, 20);

            std::cout << RB << "Opcion invalida o elegiste \"N\", Saliendo de eliminar medicamento..." <<RTNC; timeStop(1000);

        }

    } else {

        gotoxy(3, 11);

        std::cout <<RB<< "No se encontró un medicamento con ese codigo."<<RTNC; pausa();

    }

}

void menuMedicamento::buscar() {

    system(CLEAR);

    const std::string subtiImport = "Buscar un Medicamento";

    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;

    std::string auxCodigo;

    gotoxy(3, 7);

    std::cout <<GF<< "Ingresa el codigo del medicamento:";

    gotoxy(38, 7);

    fflush(stdin);

    getline(std::cin, auxCodigo);

    int idx = miArchivoMedicamento.findData(auxCodigo);

    if (idx != -1) {

        Medicamento MedicamentoAux = miArchivoMedicamento.getData(idx);

        // Imprime el Codigo del Medicamento

        gotoxy(3, 11);

        std::cout << ARB << "Codigo del medicamento:";

        gotoxy(27, 11);

        std::cout << VF << MedicamentoAux.getCodigo();

        // Imprime el Nombre del Medicamento

        gotoxy(3, 12);

        std::cout << ARB << "Nombre del Medicamento:";

        gotoxy(27, 12);

        std::cout << VF << MedicamentoAux.getNombre();

        pausa();

    } else {

        gotoxy(3, 11);

        std::cout <<RB<< "No se encontro un medicamento con ese codigo."<<RTNC; pausa();

    }

}

void menuMedicamento::importarArchivo() {

    miArchivoMedicamento.importBackup();

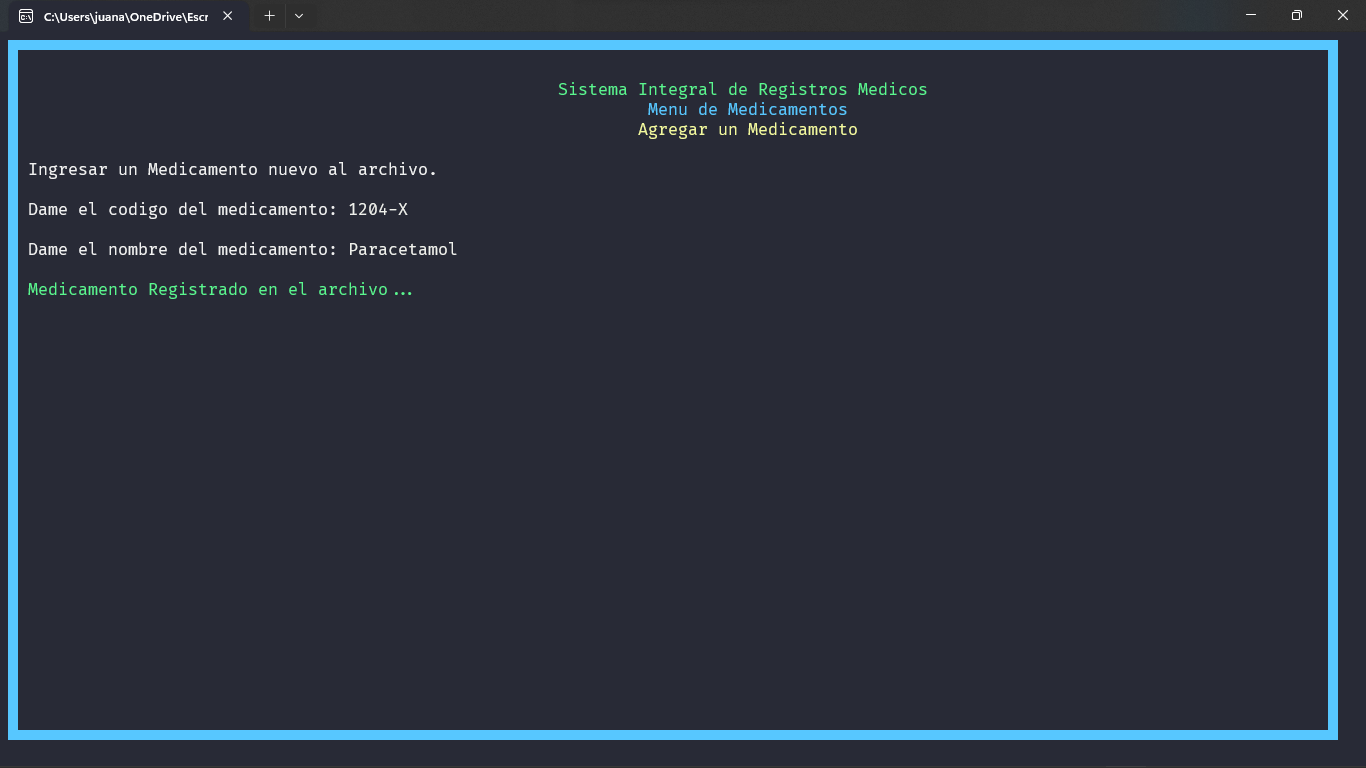
}

### Desarrollo

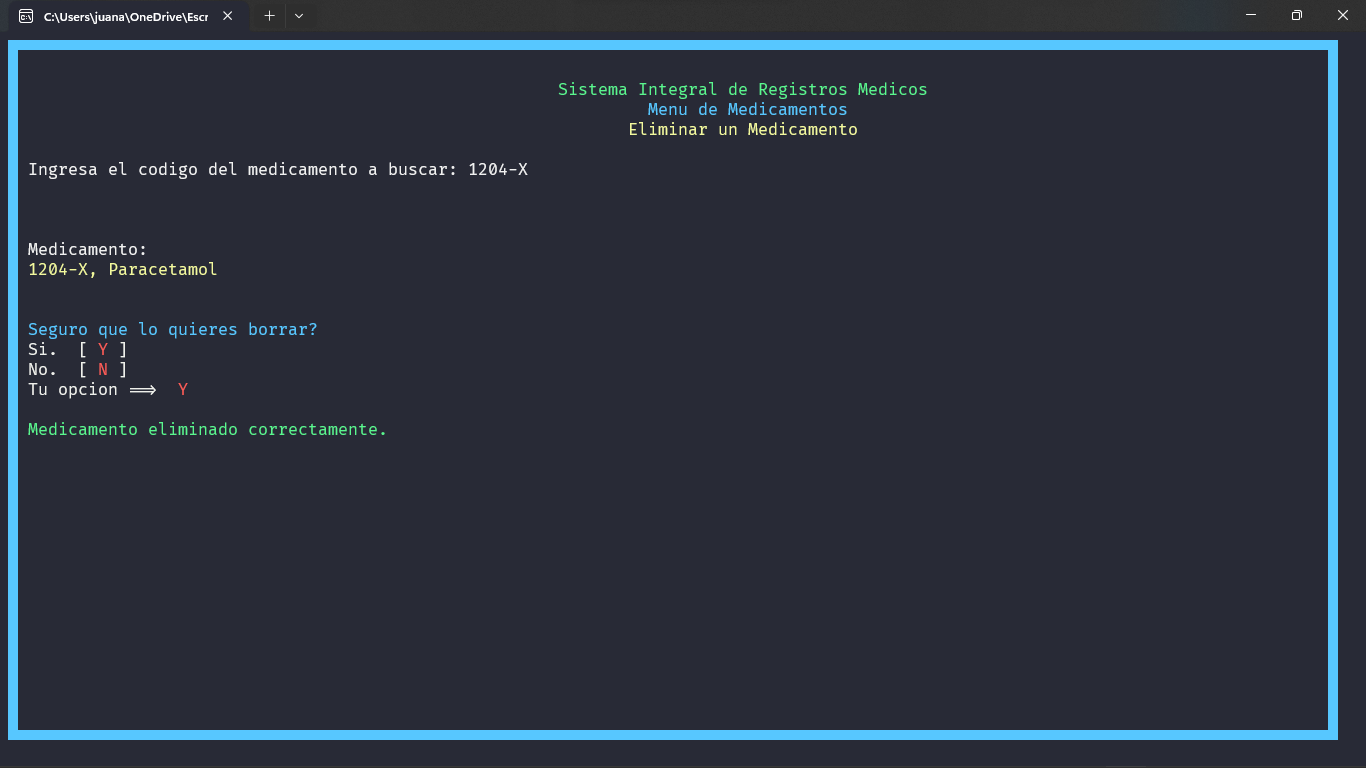
El menú principal consta de 3 operaciones, agregar, eliminar y buscar medicamentos.



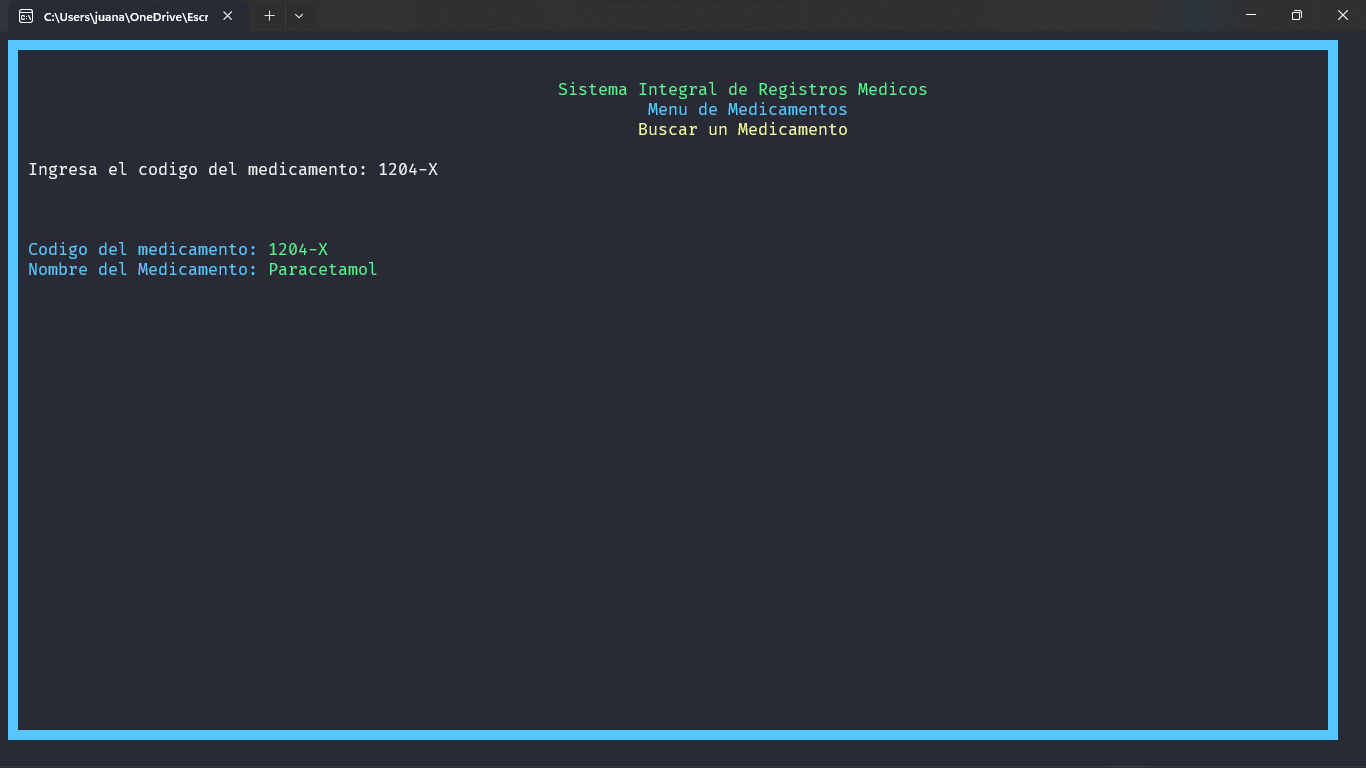
Si se escoge la opción “A”, se ingresa a la opción de ingresar un nuevo medicamento.



Para borrar un medicamento del archivo, basta con ingresar el código del medicamento.



Para buscar un dato es exactamente igual, se ingresa el código del medicamento y se busca en el archivo.



Texto

Descripción generada automáticamenteVamos a agregar unos cuantos registros al archivo, empezaremos agregando 10 registros: Clonazepam, Aspirina, Omeprazol, Levotiroxina, Ramipril, Amlodipino, Atosrvastatina, Salbutamol, Lanzoprazol e Ibuprofeno. Una vez agregado se debería de ver así en el archivo.

Ahora podemos tratar de buscarlos en el programa. Por ejemplo, si buscamos el primer resitro que ingresamos seria así:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteSi buscamos el tercer registro seria así:

El séptimo registro:

Texto

Descripción generada automáticamente

El décimo registro:

Texto

Descripción generada automáticamente

Como se puede observar, no hay ningún problema con los datos ingresados, ni al momento de hacer el ingreso, ni cuando se les busca en el archivo.

Ahora vamos a ver un ejemplo con los registros que ya estaban en el registro.

Imagen que contiene naranja, ciudad, reloj, azul

Descripción generada automáticamente

Si buscamos es registro anterior debería aparecer en el programa:

Texto

Descripción generada automáticamente

Ahora para borrar seria igual, solo buscamos los registros a través de su código. Por ejemplo el ultimo registro de los que ingresamos.

Texto

Descripción generada automáticamente

Y ya no aparece en el archivo:

Imagen que contiene Código QR

Descripción generada automáticamente

Si borramos el registro 6 de los que ingresamos:

Texto

Descripción generada automáticamente

Y lo buscamos en el archivo:

Texto

Descripción generada automáticamente con confianza media

Ya no aparece en el archivo, símbolo de que el programa funciona correctamente Ahora vamos a probar con el primer registro del archivo, el cual es el de la vacuna antirrábica:

Texto

Descripción generada automáticamente

Se busca en el programa y se borra. Y si buscamos en el archivo:



Como se puede ver, ya no se encuentra en el archivo, el programa funciona correctamente.

### Conclusión

Esta parte del programa resulto sumamente sencilla de hacer, es mas sencilla que el archivo paciente y medico por lo simple que es el objeto de medicamento (código y nombre) por lo cual es más fácil de manejar.

El menú también es más fácil de hacer, debido a que no me tengo que preocupar de tantos elementos en la pantalla.