

**“Entregable 06. Implementación de operaciones III, Implementación de interfaces de usuario I”**

**Juan Antonio Ramírez Aguilar**

**(212482507)**

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

**Mtro. Alfredo Gutiérrez Hernández**

**Clave: I5889 Sección: D13**

**01/10/2023**

**Introduccion**

Para esta actividad se me pidio que integrara el archivoMedico junto con la interfaz gráfica al programa, de tal manera que ya funcioné cuando queramos modificarlos registros del archivo.

**archivoMedico.h**

En archivo Medico se encuentran todas las definiciones que se van a utilizar para manejar los archivos que saldrán de archivo de respaldo.

#ifndef ARCHIVOMEDICO\_H

#define ARCHIVOMEDICO\_H

#include <fstream>

#include <string>

#include "Lista.h"

#include "Medico.h"

#include "Nombre.h"

#include "indexDuple.h"

class ArchivoMedico {

    private:

        std::fstream archivo;

        std::fstream archivoPorNombre;

        std::fstream archivoPorCodigo;

        std::string nameFileMedico;

        std::string indexNameFileMedico;

        std::string indexCodeFileMedico;

        Lista<indexDuple<std::string>> indexadoPorCodigo;

        Lista<indexDuple<Nombre>> indexadoPorNombre;

        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:

        ArchivoMedico();

        ~ArchivoMedico();

        void addData(const Medico&);

        void addData(Lista<Medico>&);

        Medico getData(const int&);

        void delData(const int&);

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

        int findData(const Nombre&) const;

        Lista<Medico> toList();

        void clear();

        void compress();

        void importBackup();

        void exportBackup(const std::string&);

};

#endif

**archivoMedico.cpp**

En archivoMedico.cpp, están las implementaciones de archivoMedico.h, todas estas funciones son las que se encargan de comunicarse con los datos de los archivos.

#include "ArchivoMedico.h"

#include <fstream>

#include <cstdlib>

#include <sstream>

#include "Nombre.h"

#include "Hora.h"

#include "Medico.h"

using namespace std;

// Constructor

ArchivoMedico::ArchivoMedico() : nameFileMedico("archivoMedicos.txt"), indexNameFileMedico("archivoMedicosNombre.txt"),

indexCodeFileMedico("archivoMedicosCodigo.txt") {}

void ArchivoMedico::reIndex() {

    string myString;

    int myIndex;

    Medico miMedico;

    indexDuple<> miTupleCode;

    indexDuple<Nombre> miTupleName;

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

    if (!archivo.is\_open()) {

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

    }

    indexadoPorCodigo.anular();

    indexadoPorNombre.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 >> miMedico;

            miTupleCode.setIndex(myIndex);

            miTupleCode.setData(miMedico.getCodigo());

            miTupleName.setIndex(myIndex);

            miTupleName.setData(miMedico.getNombre());

            indexadoPorCodigo.insertar(miTupleCode);

            indexadoPorNombre.insertar(miTupleName);

        }

    }

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

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

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

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

        }

        listToFile(indexadoPorCodigo, archivoPorCodigo);

        archivoPorCodigo.close();

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

        archivoPorNombre.open(indexNameFileMedico, ios\_base::out | ios\_base::trunc);

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

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

        }

        listToFile(indexadoPorNombre, archivoPorNombre);

        archivoPorNombre.close();

        archivo.close();

}

// Destructor

ArchivoMedico::~ArchivoMedico() {

    if (archivo.is\_open()) {

        archivo.close();

    }

    if (archivoPorNombre.is\_open()) {

        archivoPorNombre.close();

    }

    if (archivoPorCodigo.is\_open()) {

        archivoPorCodigo.close();

    }

}

template<class T>

Lista<T> ArchivoMedico::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 ArchivoMedico::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 ArchivoMedico::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 ArchivoMedico::addData(const Medico & medicoToAdd) {

    /\*archivoNombre.open(NombreArchivo, ios\_base::out | ios\_base::app);

    if (!archivoNombre.is\_open()) {

        throw ios\_base::failure("No se pudo abrir el archivo para esrcitura.");

    }

    archivoNombre << "1\*" << medicoToAdd << "#";

    archivoNombre.close();\*/

    Lista<Medico> myList;

    myList.insertar(medicoToAdd);

    addData(myList);

}

void ArchivoMedico::addData(Lista<Medico>& myList) {

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

    if (!archivo.is\_open()) {

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

    }

    listToFile(myList, archivo);

    archivo.close();

    reIndex();

}

Medico ArchivoMedico::getData(const int & idx) {

    string myString;

    Medico miMedico;

    archivoPorNombre.open(nameFileMedico);

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

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

    }

    archivoPorNombre.seekg(idx);

    getline(archivoPorNombre, 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 >> miMedico;

    return miMedico;

}

/\*void ArchivoMedico::delData(const int& idx) {

    archivo.open(nameFileMedico, ios\_base::out);

    if (!archivo.is\_open()) {

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

    }

    archivo.seekp(idx);

    archivo << "0";

    archivo.close();

    reIndex();

}\*/

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

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

    if (!archivo.is\_open()) {

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

    }

    archivo.seekp(idx);

    archivo << "0";

    archivo.close();

    reIndex();

}

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

    return getIndex(indexadoPorCodigo, code);

}

int ArchivoMedico::findData(const Nombre& e) const {

    return getIndex(indexadoPorNombre, e);

}

Lista<Medico> ArchivoMedico::toList() {

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

    if (!archivo.is\_open()) {

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

    }

    Lista<Medico> listaAux;

    listaAux = fileToList(archivo, listaAux);

    archivo.close();

    return listaAux;

}

void ArchivoMedico::clear() {

    remove(nameFileMedico.c\_str());

}

void ArchivoMedico::compress() {

    string tempFileName = "temp.file";

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

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

    // Se abre el archivo original en modo escritura.

    archivo.open(nameFileMedico, 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 ArchivoMedico::importBackup() {

    reIndex();

}

**menuMedicos.h**

menúMedicos es la que se encarga de comunicar a archivoMedico con el usuario, aquí se llevan a cabo las operaciones de menú, agregar, eliminar, buscar e importar el archivo de médicos, además se instancia como privado un objeto de tipo ArchivoMedico y una variable para operaciones.

#ifndef MENUMEDICOS\_H

#define MENUMEDICOS\_H

#include "ArchivoMedico.h"

class menuMedicos {

    private:

        std::string opc;

        ArchivoMedico miArchivoMedico;

    public:

        menuMedicos();

        ~menuMedicos();

        void setOpc(std::string valorOpc);

        std::string getOpc();

        void menu();

        void agregar();

        void eliminar();

        void buscar();

        void importarArchivo();

};

#endif

**menuMedicos.cpp**

Aquí están las implementaciones del .h, como podemos ver se despliega un menú para escoger cada una de las opciones de las operaciones.

#include "menuMedicos.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 Medicos";

menuMedicos::menuMedicos() : opc("") {}

menuMedicos::~menuMedicos() {}

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

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

void menuMedicos::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 Medico.         [ " <<RF<< "A" <<GB " ]";

        gotoxy(3, 10);

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

        gotoxy(3, 11);

        std::cout <<GB; std::cout << "Buscar un Medico.          [ " <<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 menuMedicos::agregar() {

    Medico miMedico;

    Nombre MiNombre;

    Hora miHora;

    std::string miString;

    int auxHora;

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

    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 medico nuevo al archivo." <<RTNC;

    gotoxy(3, 9);

    std::cout << GF << "Dame la Cedula del medico:" <<RTNC;

    gotoxy(40, 9);

    fflush(stdin);

    std::getline(cin, miString);

    miMedico.setCedula(miString);

    gotoxy(3, 11);

    std::cout << GF << "Dame el Apellido del Medico:" <<RTNC;

    fflush(stdin);

    gotoxy(40, 11);

    std::getline(cin, miString);

    MiNombre.setApellido(miString);

    gotoxy(3, 13);

    std::cout << GF << "Dame el Nombre del Medico:" <<RTNC;

    gotoxy(40, 13);

    fflush(stdin);

    std::getline(cin, miString);

    MiNombre.setNombre(miString);

    miMedico.setNombre(MiNombre);

    gotoxy(3, 15);

    std::cout << GF << "Dame el Codigo del Medico:" <<RTNC;

    gotoxy(40, 15);

    fflush(stdin);

    std::getline(cin, miString);

    miMedico.setCodigo(miString);

    gotoxy(3, 17);

    std::cout << GF << "Dame la hora de entrada del Medico:" <<RTNC;

    gotoxy(3, 18);

    std::cout << GF << "Hora:" <<RTNC;

    gotoxy(15, 18);

    cin >> auxHora;

    miHora.setHora(auxHora);

    gotoxy(3, 19);

    std::cout << GF << "Minutos:" <<RTNC;

    gotoxy(15, 19);

    cin >> auxHora;

    miHora.setMinuto(auxHora);

    miMedico.setHoraEntrada(miHora);

    gotoxy(3, 21);

    std::cout << GF << "Dame la hora de salida del Medico:" <<RTNC;

    gotoxy(3, 22);

    std::cout << GF << "Hora:" <<RTNC;

    gotoxy(15, 22);

    cin >> auxHora;

    miHora.setHora(auxHora);

    gotoxy(3, 23);

    std::cout << GF << "Minutos:" <<RTNC;

    gotoxy(15, 23);

    cin >> auxHora;

    miHora.setMinuto(auxHora);

    miMedico.setHoraSalida(miHora);

    miArchivoMedico.addData(miMedico);

    gotoxy(3, 25);

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

}

void menuMedicos::eliminar() {

    std::string opcElim;

    gotoxy(3, 16);

    std::cout << GB << "Eliminar por Nombre.  [ " <<RF<< "A" <<GB " ]";

    gotoxy(3, 17);

    std::cout << GB << "Eliminar por Codigo.  [ " <<RF<< "B" <<GB " ]";

    gotoxy(3, 18);

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

    gotoxy(3, 19);

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

    gotoxy(18, 19);

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

    if (opcElim == "A" or opcElim == "a") {

        Nombre miNombre;

        system(CLEAR);

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

        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 stringAux;

        gotoxy(3, 7);

        std::cout <<GF<< "Ingresa el apellido del Medico a buscar: ";

        gotoxy(45, 7);

        fflush(stdin);

        getline(std::cin, stringAux);

        miNombre.setNombre(stringAux);

        gotoxy(3, 9);

        std::cout <<GF<< "Ingresa el nombre del Medico a buscar: ";

        gotoxy(43, 9);

        fflush(stdin);

        getline(std::cin, stringAux);

        miNombre.setApellido(stringAux);

        int idx = miArchivoMedico.findData(miNombre);

        if (idx != -1) {

            string opcBorrar;

            Medico showMedico = miArchivoMedico.getData(idx);

            gotoxy(3, 11);

            std::cout << "Medico: " << std::endl;

            gotoxy(3, 12);

            std::cout << AF << showMedico.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") {

                miArchivoMedico.delData(idx);

                miArchivoMedico.compress();

                gotoxy(3, 20);

                std::cout <<VB<< "Medico 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 medico..." <<RTNC; timeStop(1000);

            }

        } else {

            gotoxy(3, 11);

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

        }

    } else if (opcElim == "B" or opcElim == "b") {

        system(CLEAR);

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

        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 codigo;

        gotoxy(3, 7);

        std::cout <<GF<< "Ingresa el codigo del Medico a buscar: ";

        gotoxy(43, 7);

        fflush(stdin);

        getline(std::cin, codigo);

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

        int idx = miArchivoMedico.findData(codigo);

        if (idx != -1) {

            string opcBorrar;

            Medico showMedico = miArchivoMedico.getData(idx);

            gotoxy(3, 9);

            std::cout << "Medico: " << std::endl;

            gotoxy(3, 10);

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

            gotoxy(3, 13);

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

            gotoxy(3, 14);

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

            gotoxy(3, 15);

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

            gotoxy(3, 16);

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

            gotoxy(18, 16);

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

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

                miArchivoMedico.delData(idx);

                miArchivoMedico.compress();

                gotoxy(3, 18);

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

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

                gotoxy(3, 18);

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

            }

        } else {

            gotoxy(3, 9);

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

        }

    } else if (opcElim == "X" or opcElim == "x") {

        return;

    } else {

        gotoxy(3, 21); std::cout <<GB; std::cout << "La opcion: \"" <<RB<< opcElim <<GB<< "\" no es una opcion valida..."<<RTNC; pausa();

    }

}

void menuMedicos::buscar() {

    std::string opcBuscar;

    gotoxy(3, 16);

    std::cout << GB << "Buscar por Nombre.  [ " <<RF<< "A" <<GB " ]";

    gotoxy(3, 17);

    std::cout << GB << "Buscar por Codigo.  [ " <<RF<< "B" <<GB " ]";

    gotoxy(3, 18);

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

    gotoxy(3, 19);

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

    gotoxy(18, 19);

    fflush(stdin);

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

    if (opcBuscar == "A" or opcBuscar == "a") {

        system(CLEAR);

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

        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 stringAux;

        Nombre nombreAux;

        gotoxy(3, 7);

        std::cout <<GF<< "Ingresa el apellido del Medico a buscar: ";

        gotoxy(45, 7);

        fflush(stdin);

        getline(std::cin, stringAux);

        nombreAux.setApellido(stringAux);

        gotoxy(3, 9);

        std::cout <<GF<< "Ingresa el nombre del Medico a buscar: ";

        gotoxy(43, 9);

        fflush(stdin);

        getline(std::cin, stringAux);

        nombreAux.setNombre(stringAux);

        int idx = miArchivoMedico.findData(nombreAux);

        if (idx != -1) {

            Medico medicoAux = miArchivoMedico.getData(idx);

            // Imprime el Nombre del Medico

            gotoxy(3, 11);

            cout << ARB << "Nombre del Medico: ";

            gotoxy(23, 11);

            cout << VF << medicoAux.getNombre().toString();

            // Imprime el Codigo del Medico

            gotoxy(3, 12);

            cout << ARB << "Codigo del Medico: ";

            gotoxy(23, 12);

            cout << VF << medicoAux.getCodigo();

            // Imprime la Cedula del Medico

            gotoxy(3, 13);

            cout << ARB << "Cedula del Medico: ";

            gotoxy(23, 13);

            cout << VF << medicoAux.getCedula();

            // Imprime la Hora de Entrada del Medico

            gotoxy(3, 14);

            cout << ARB << "Hora de Entrada del Medico: ";

            gotoxy(31, 14);

            cout << VF << medicoAux.getHoraEntrada().toString();

            // Imprime la Hora de Salida del Medico

            gotoxy(3, 15);

            cout << ARB << "Hora de Salida del Medico: ";

            gotoxy(31, 15);

            cout << VF << medicoAux.getHoraSalida().toString();

            pausa();

        } else {

            gotoxy(3, 11);

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

        }

    } else if (opcBuscar == "B" or opcBuscar == "b") {

        system(CLEAR);

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

        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 stringAux;

        gotoxy(3, 7);

        std::cout <<GF<< "Ingresa el codigo del Medico a buscar: ";

        gotoxy(45, 7);

        fflush(stdin);

        getline(std::cin, stringAux);

        int idx = miArchivoMedico.findData(stringAux);

        if (idx != -1) {

            Medico medicoAux = miArchivoMedico.getData(idx);

            // Imprime el Nombre del Medico

            gotoxy(3, 9);

            cout << ARB << "Nombre del Medico: ";

            gotoxy(23, 9);

            cout << VF << medicoAux.getNombre().toString();

            // Imprime el Codigo del Medico

            gotoxy(3, 10);

            cout << ARB << "Codigo del Medico: ";

            gotoxy(23, 10);

            cout << VF << medicoAux.getCodigo();

            // Imprime la Cedula del Medico

            gotoxy(3, 11);

            cout << ARB << "Cedula del Medico: ";

            gotoxy(23, 11);

            cout << VF << medicoAux.getCedula();

            // Imprime la Hora de Entrada del Medico

            gotoxy(3, 12);

            cout << ARB << "Hora de Entrada del Medico: ";

            gotoxy(31, 12);

            cout << VF << medicoAux.getHoraEntrada().toString();

            // Imprime la Hora de Salida del Medico

            gotoxy(3, 13);

            cout << ARB << "Hora de Salida del Medico: ";

            gotoxy(31, 13);

            cout << VF << medicoAux.getHoraSalida().toString();

            pausa();

        } else {

            gotoxy(3, 9);

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

        }

    } else if (opcBuscar == "X" or opcBuscar == "x") {

        return;

    } else {

        gotoxy(3, 21); std::cout <<GB; std::cout << "La opcion: \"" <<RB<< opcBuscar <<GB<< "\" no es una opcion valida..."<<RTNC; pausa();

    }

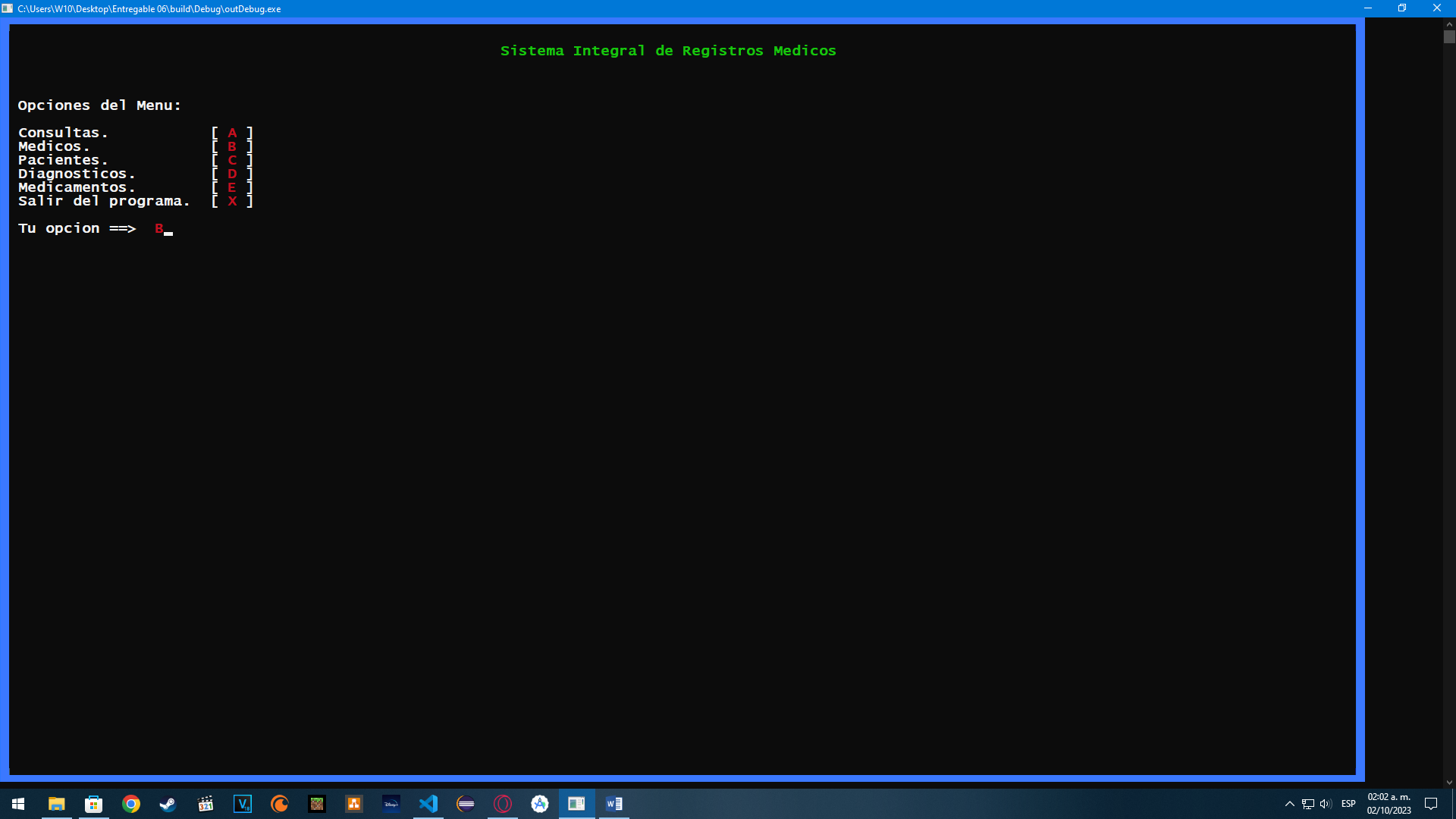
}

void menuMedicos::importarArchivo() {

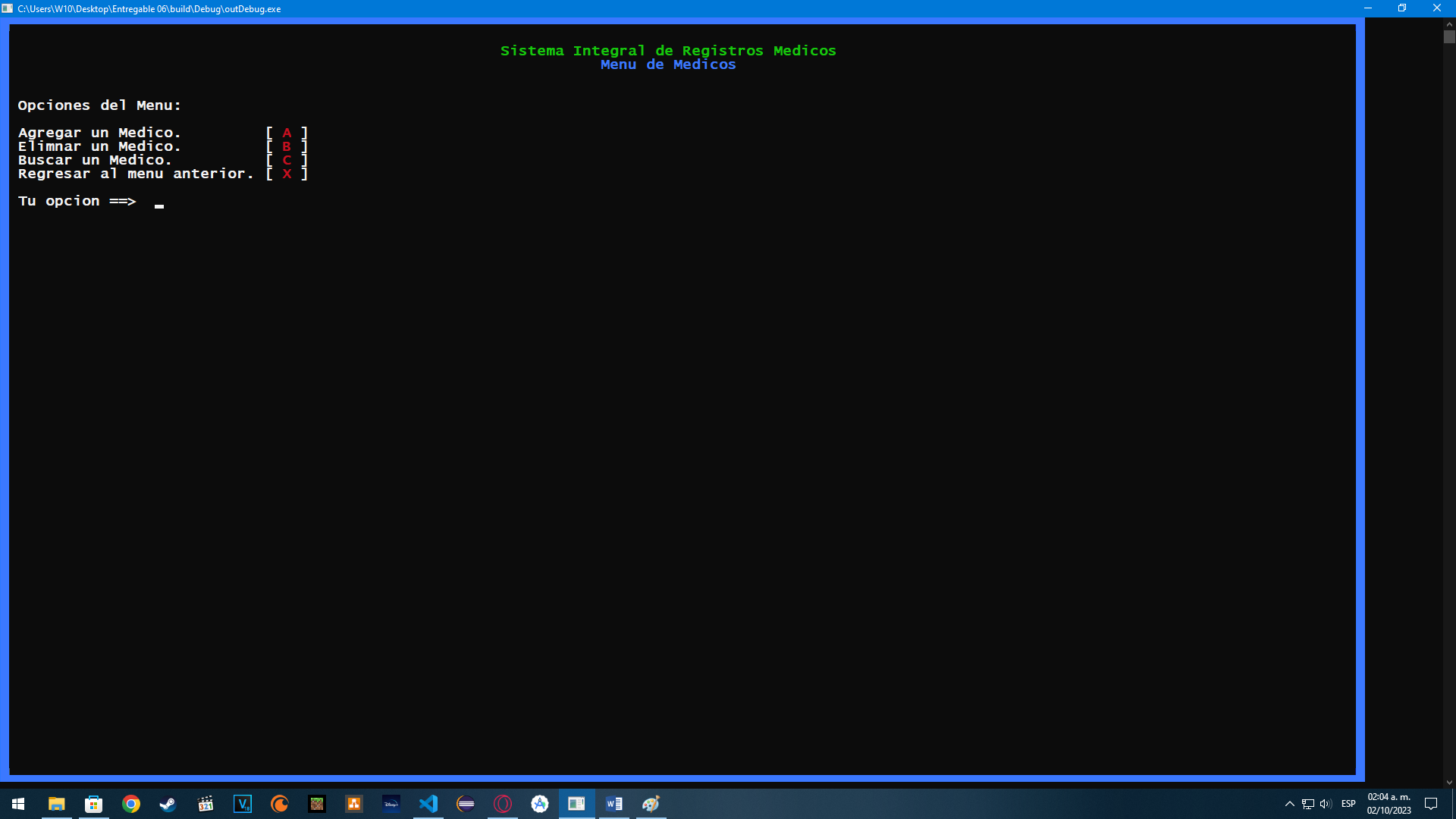
    miArchivoMedico.importBackup();

}

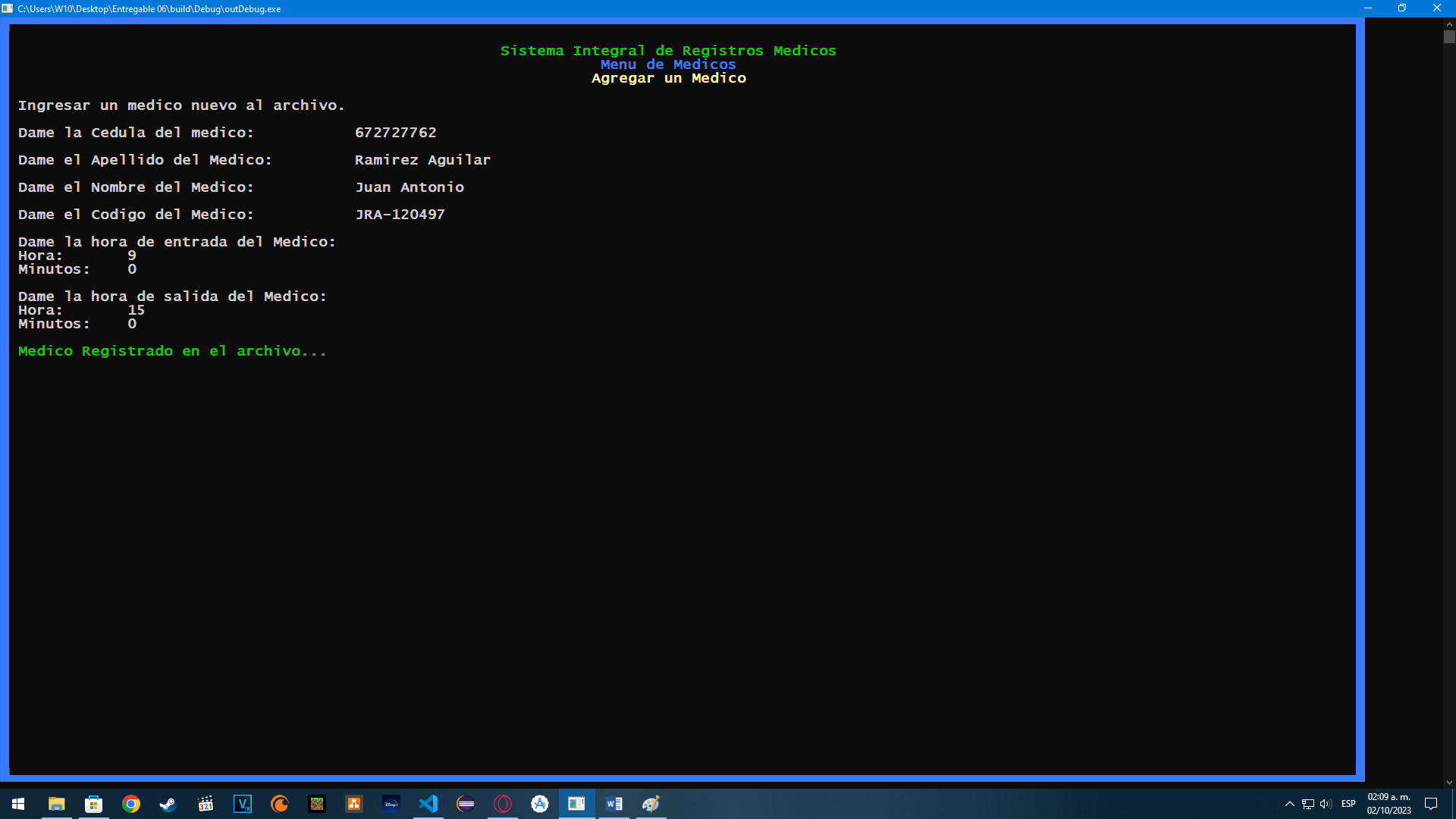
Cuando se abre el menú, debería verse así con todas las opciones:



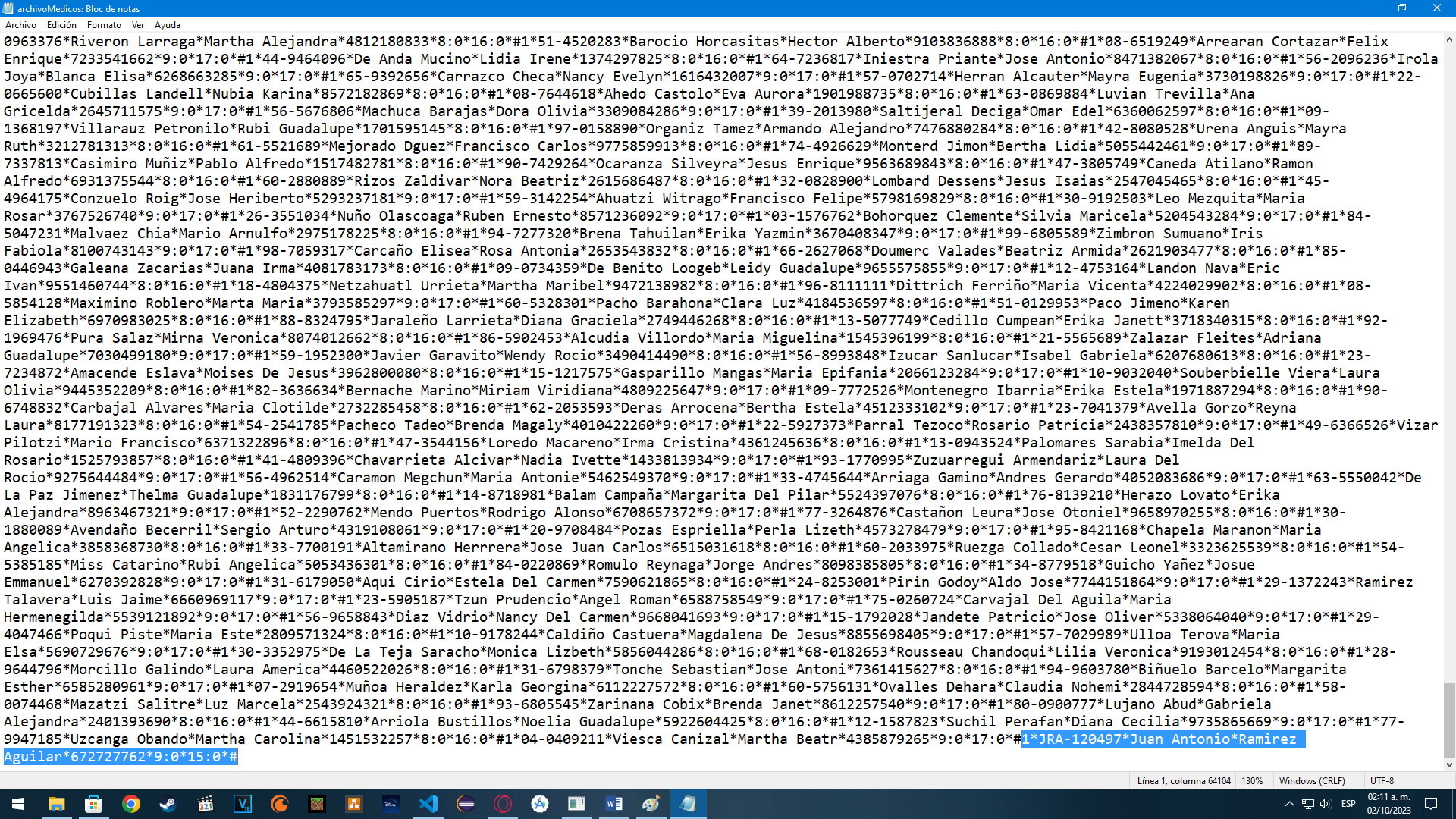
Al ingresar la opción B, entraremos a el menú de Médicos, nótese como aparece un subtítulo junto al título principal del programa.



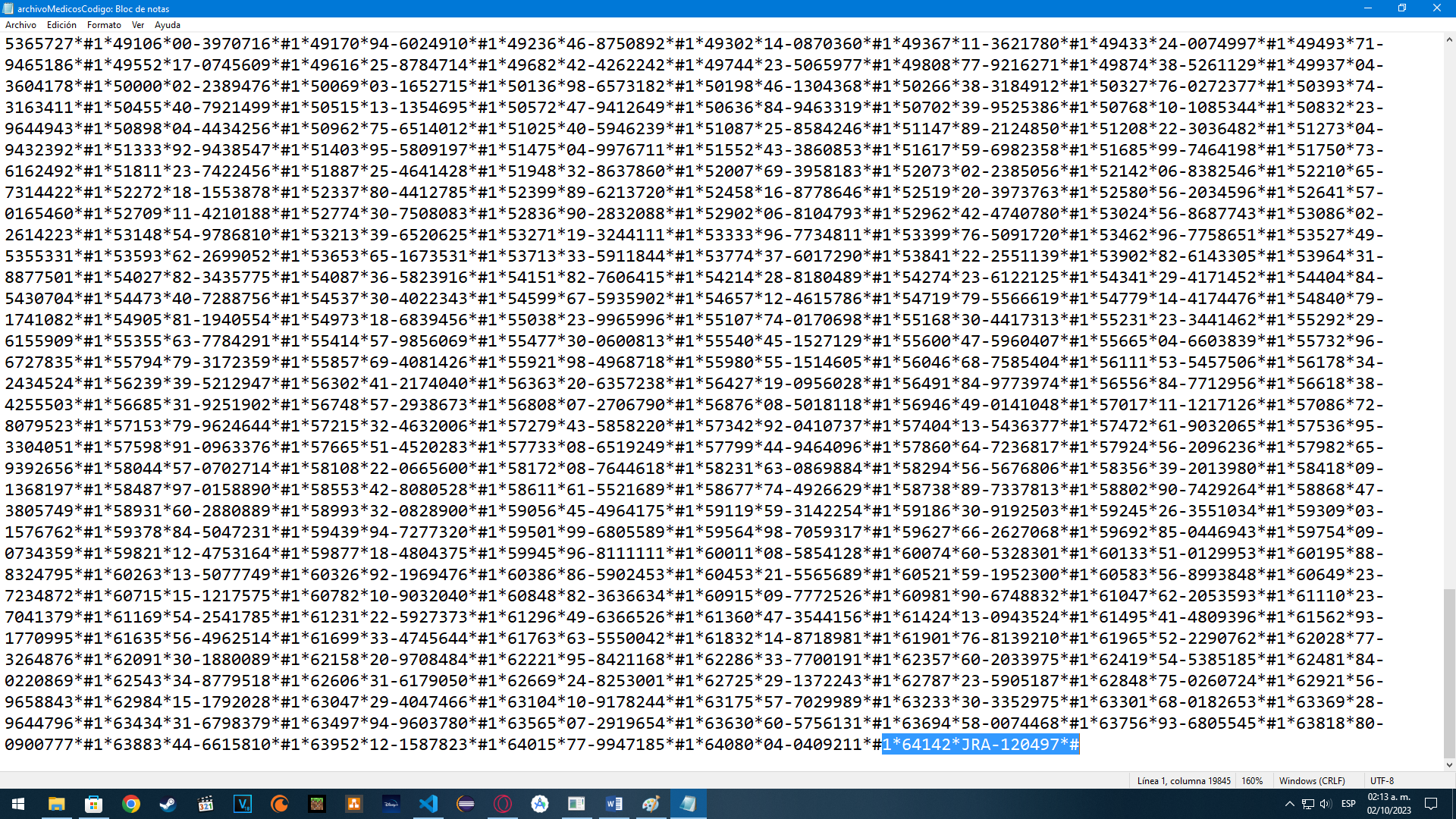
Al ingresar A en el menú, entraremos a la operación de agregación de n medico al archivo, se nos pedirá los datos del médico a registrar para inmediatamente después ser ingresado al archivo.



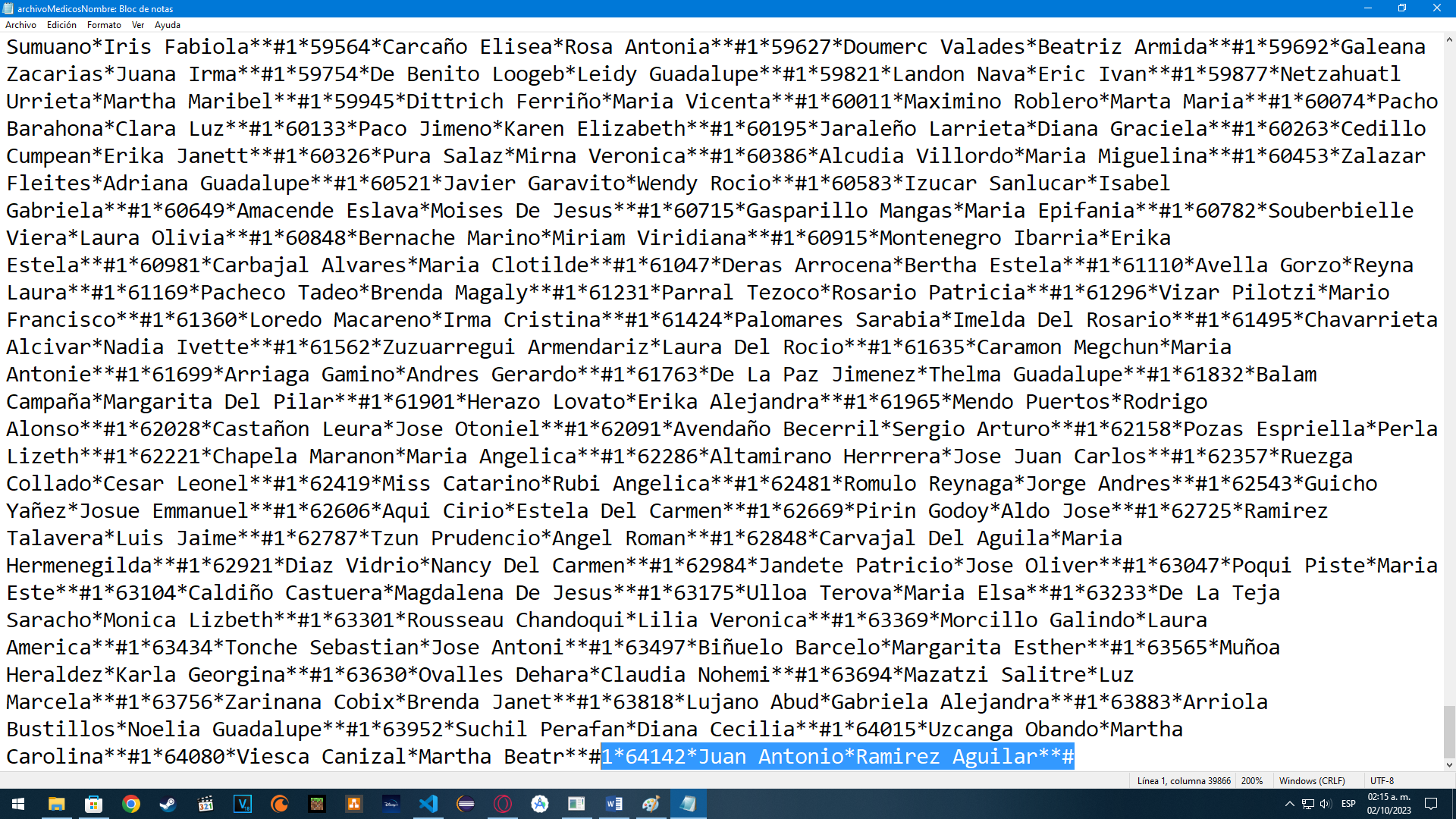
El registro del nuevo medico debería de aparecer ya en el archivo.



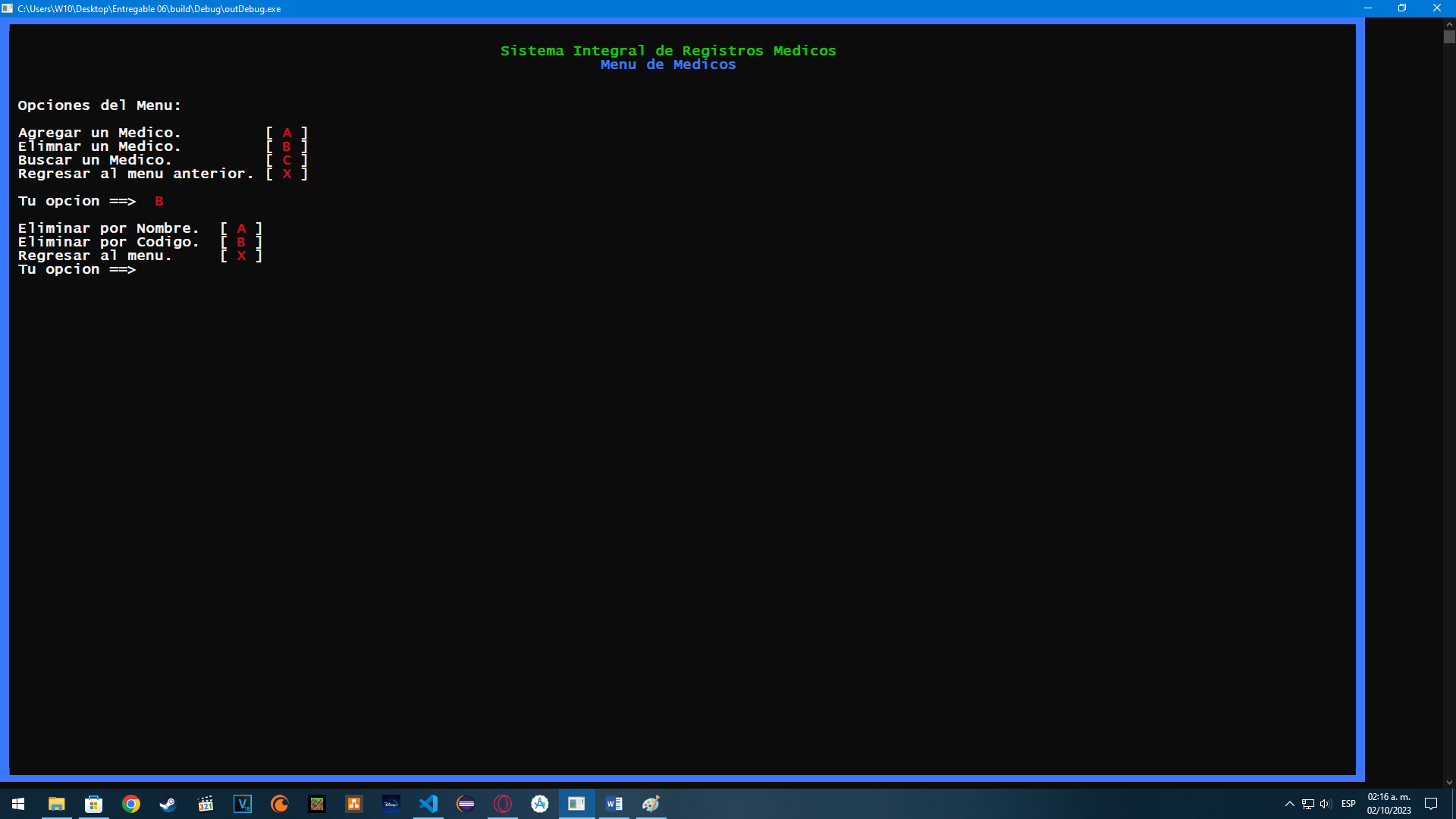
También debería aparecer en los archivos para los índices al re indexarse cuando agregué el registro.

Archivo medico por codigo:

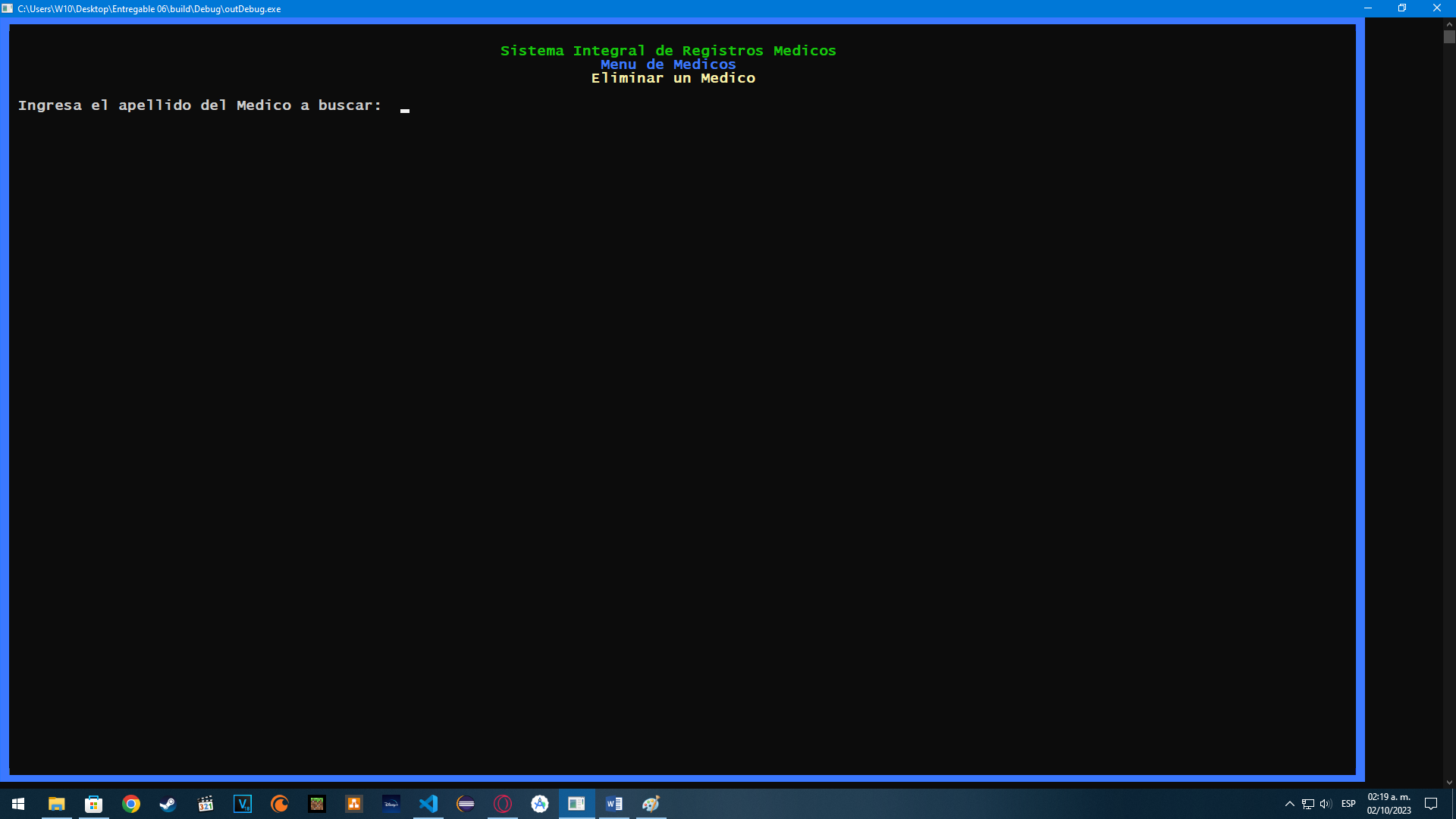
Archivo medico por nombre:



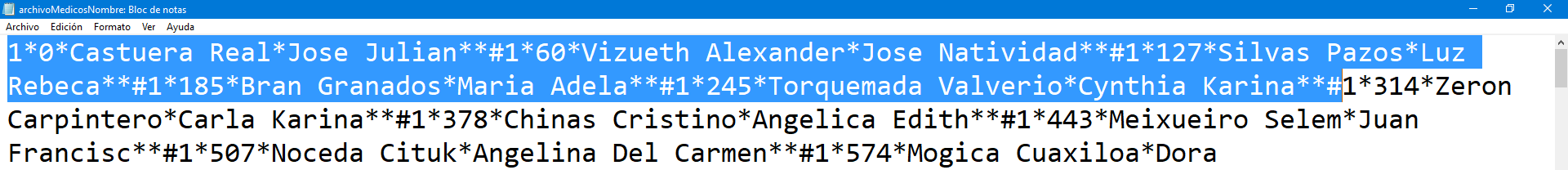
Cuando se elige eliminar, se despliega un submenú que te da dos opciones para eliminar, por nombre o por código:

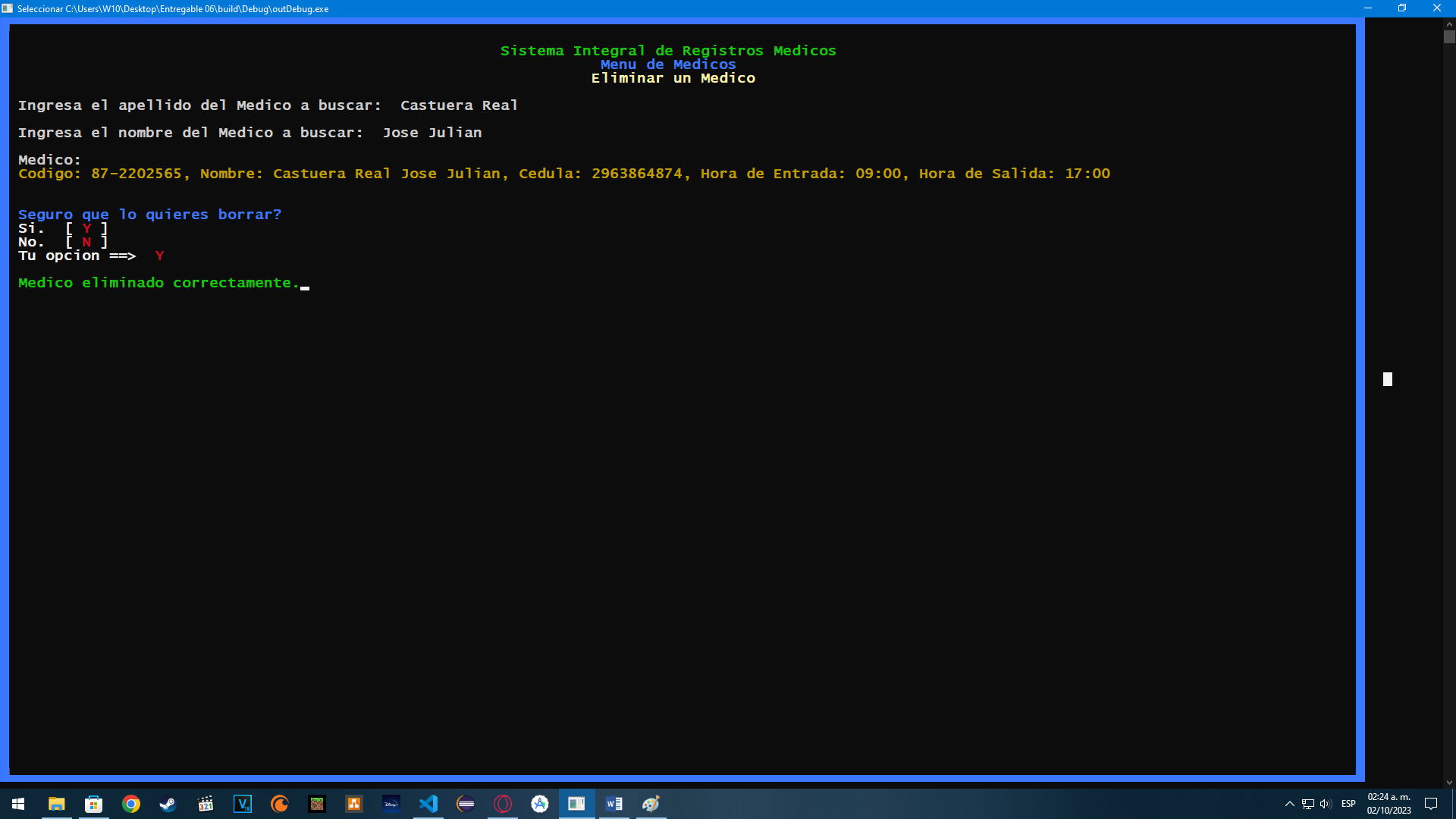


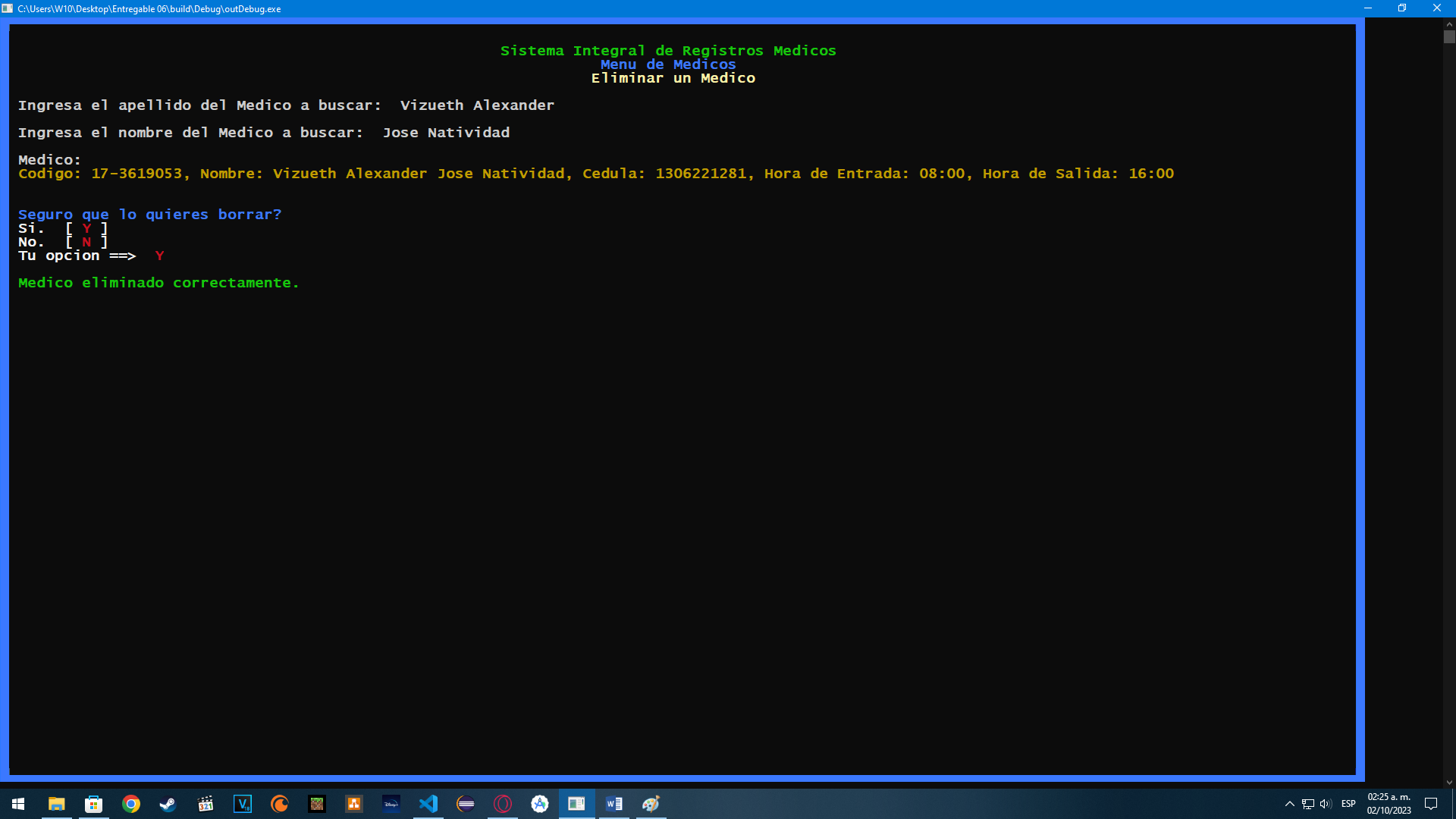
Cuando se escoge por nombre, se te pide que ingreses el nombre y el apellido del medico al que deseas eliminar.

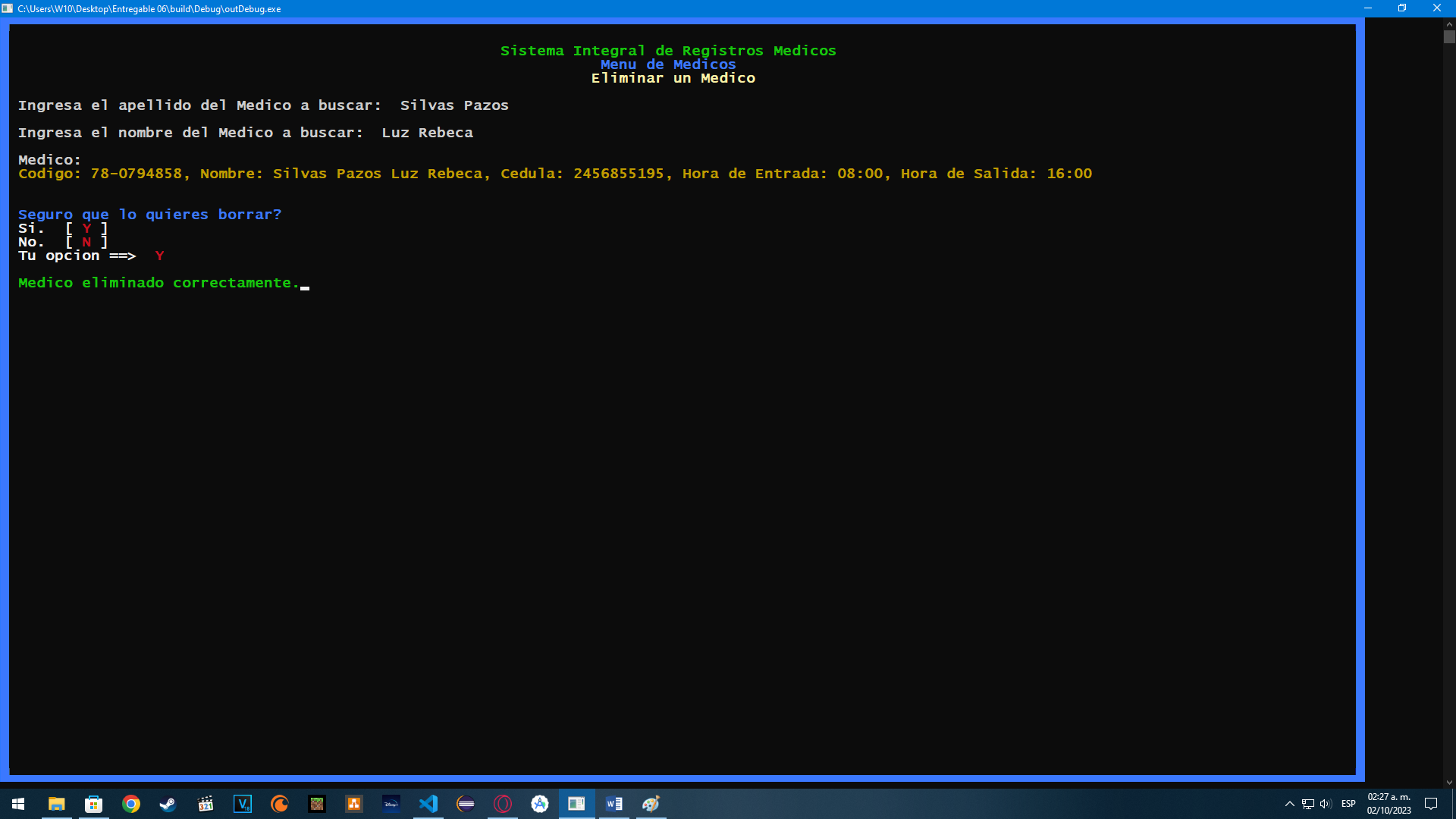


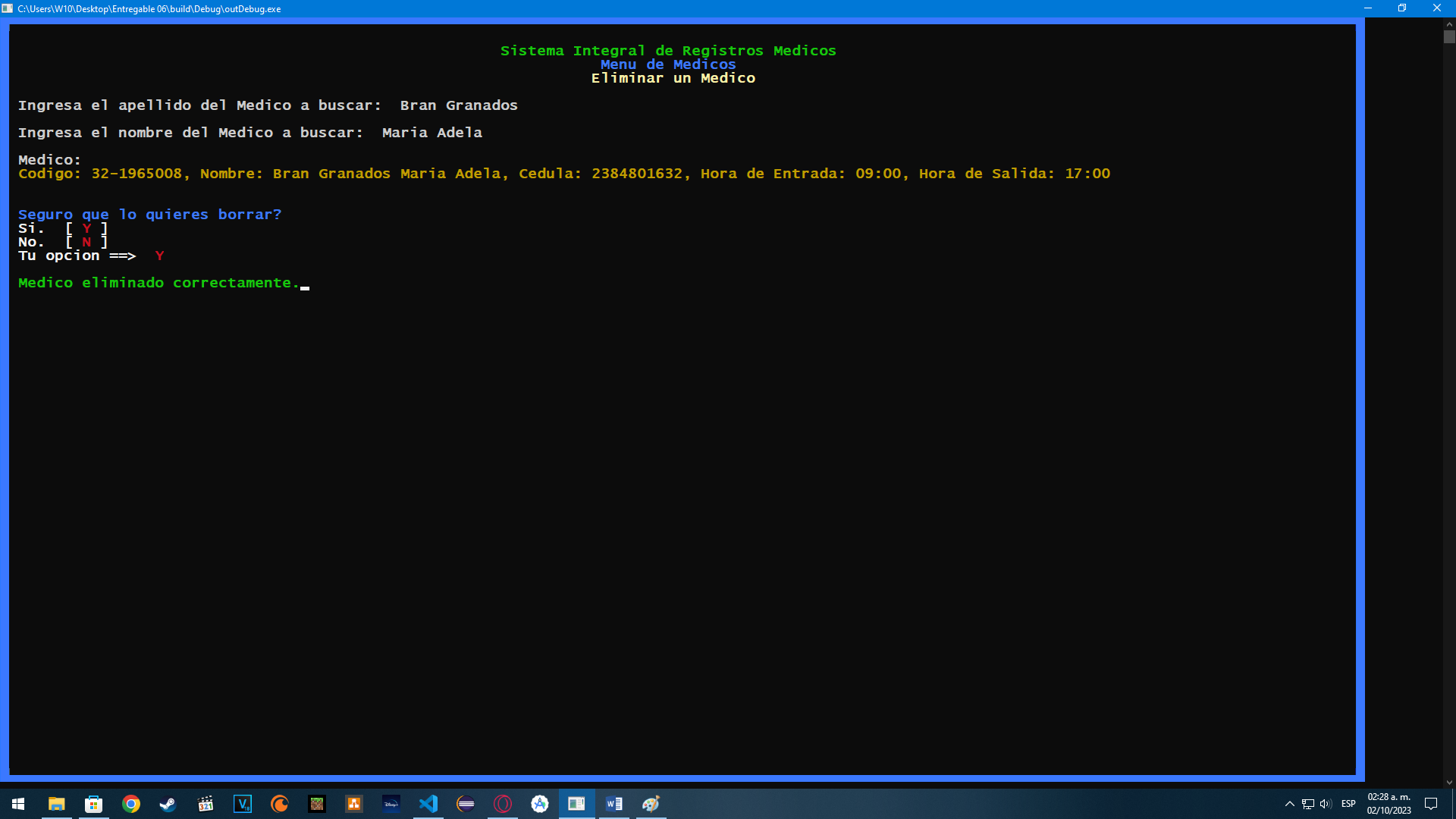
Para el ejemplo de eliminación elegiré los primeros 5 registros que hay en el archivo.

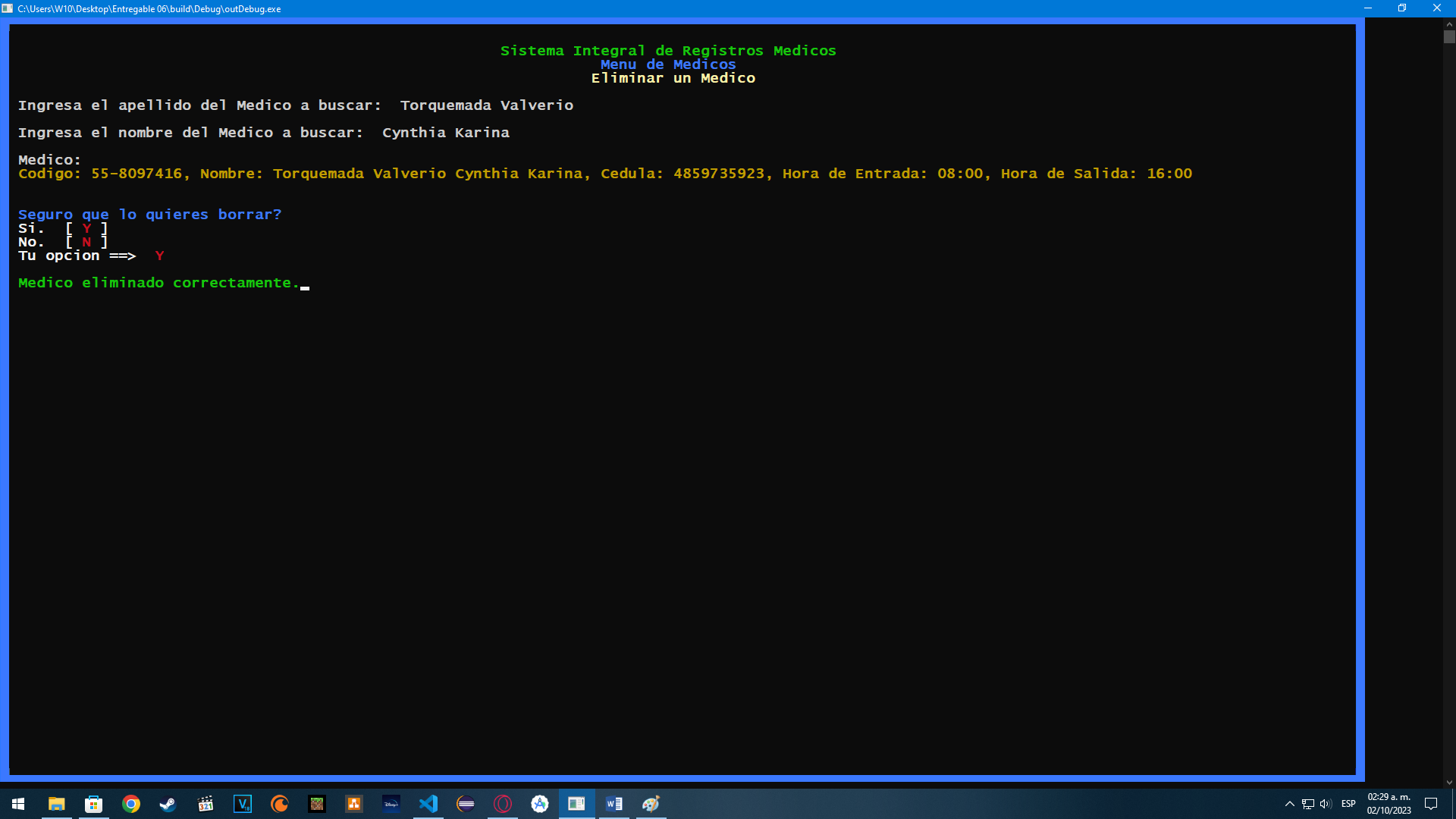




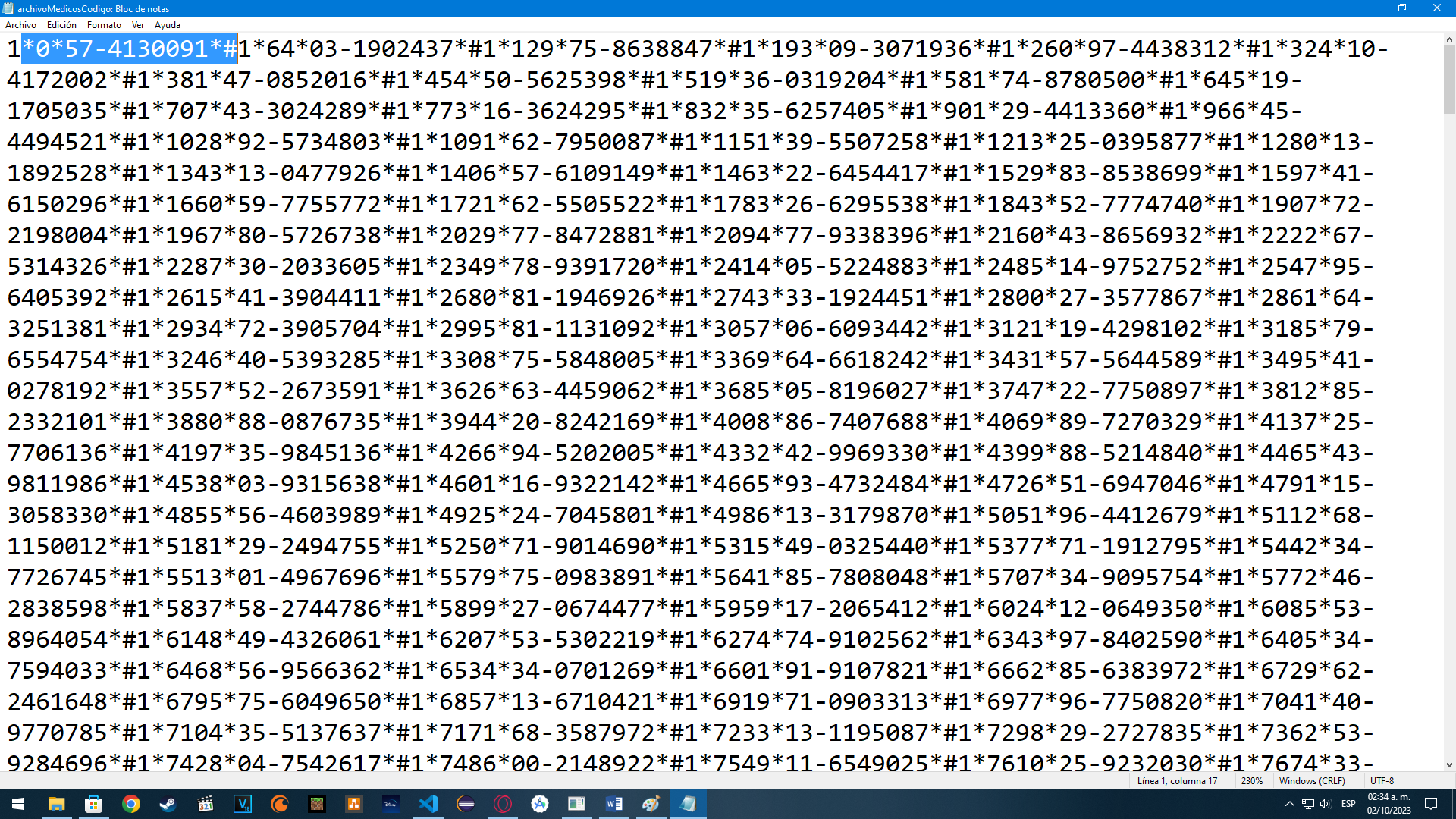




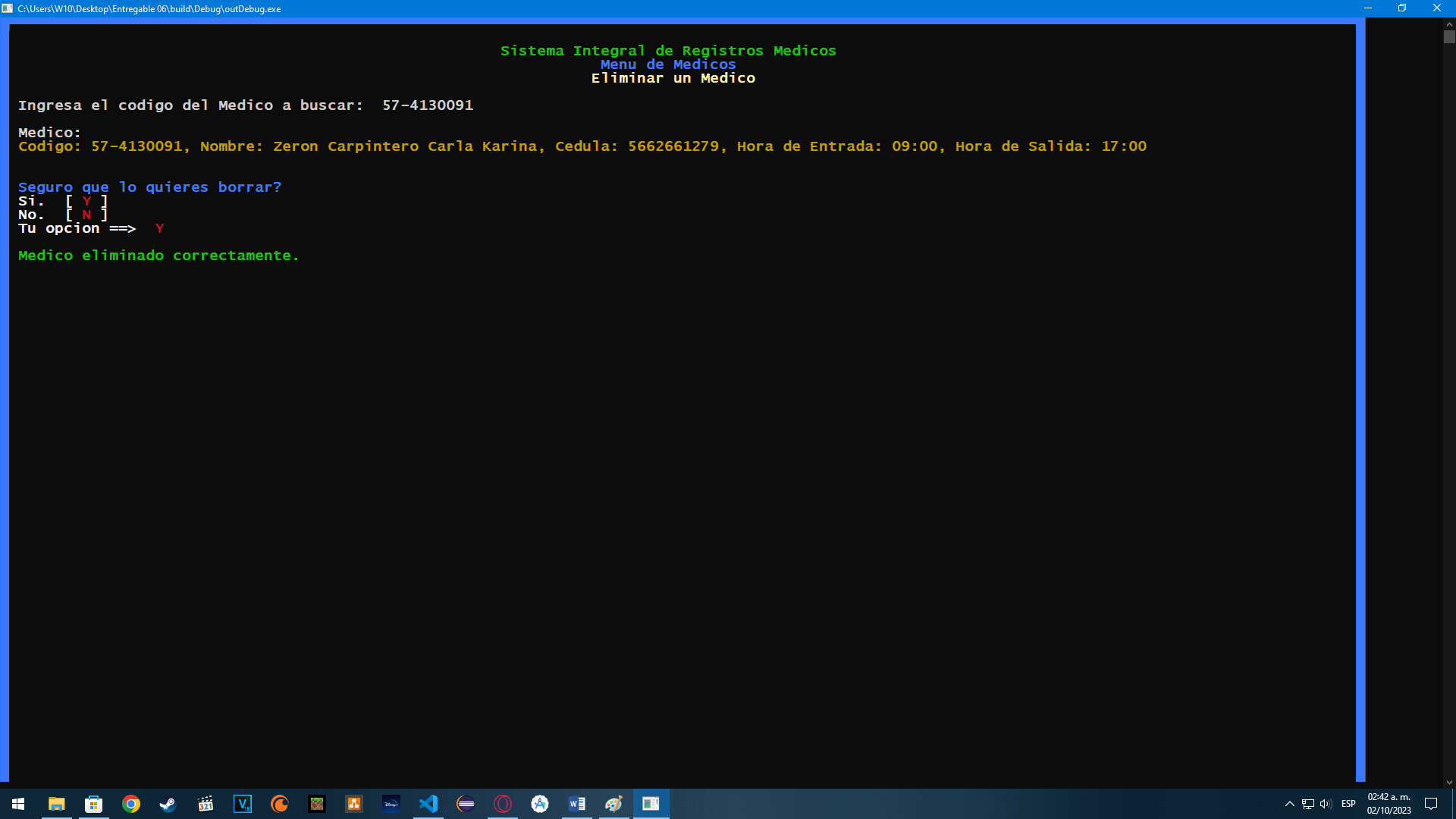




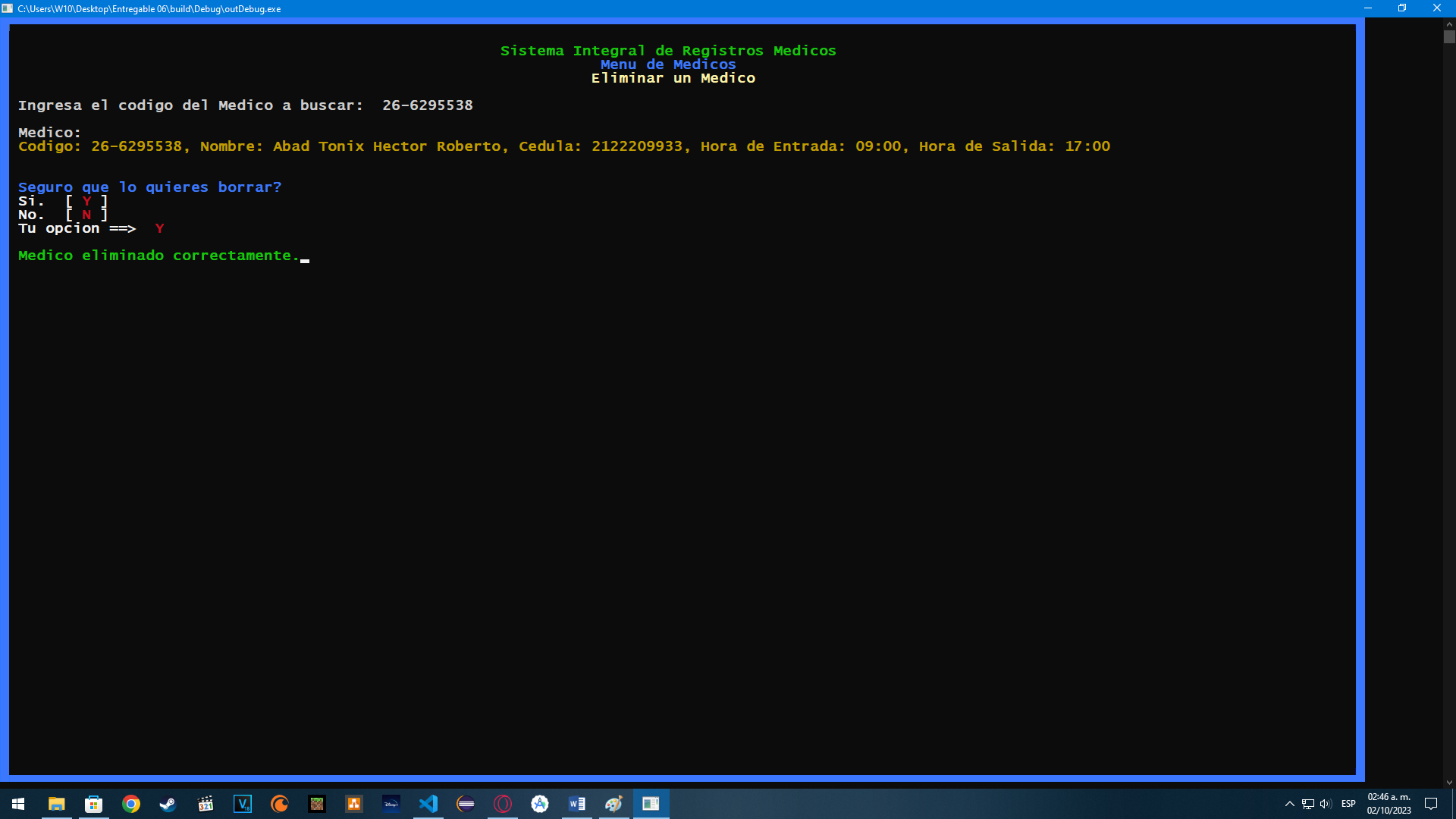
Como se pude ver en el archivo ya no aparecen los datos de los medicos que eliminamos anteriormente, ahora el primer registro que nos aparece es Zeron Carpintero y es que era el que seguia despues de los registros que borramos, ahora vermenos como lo hace por codigo, vamos a borrar a Zeron.

Este deberia ser su codigo: 

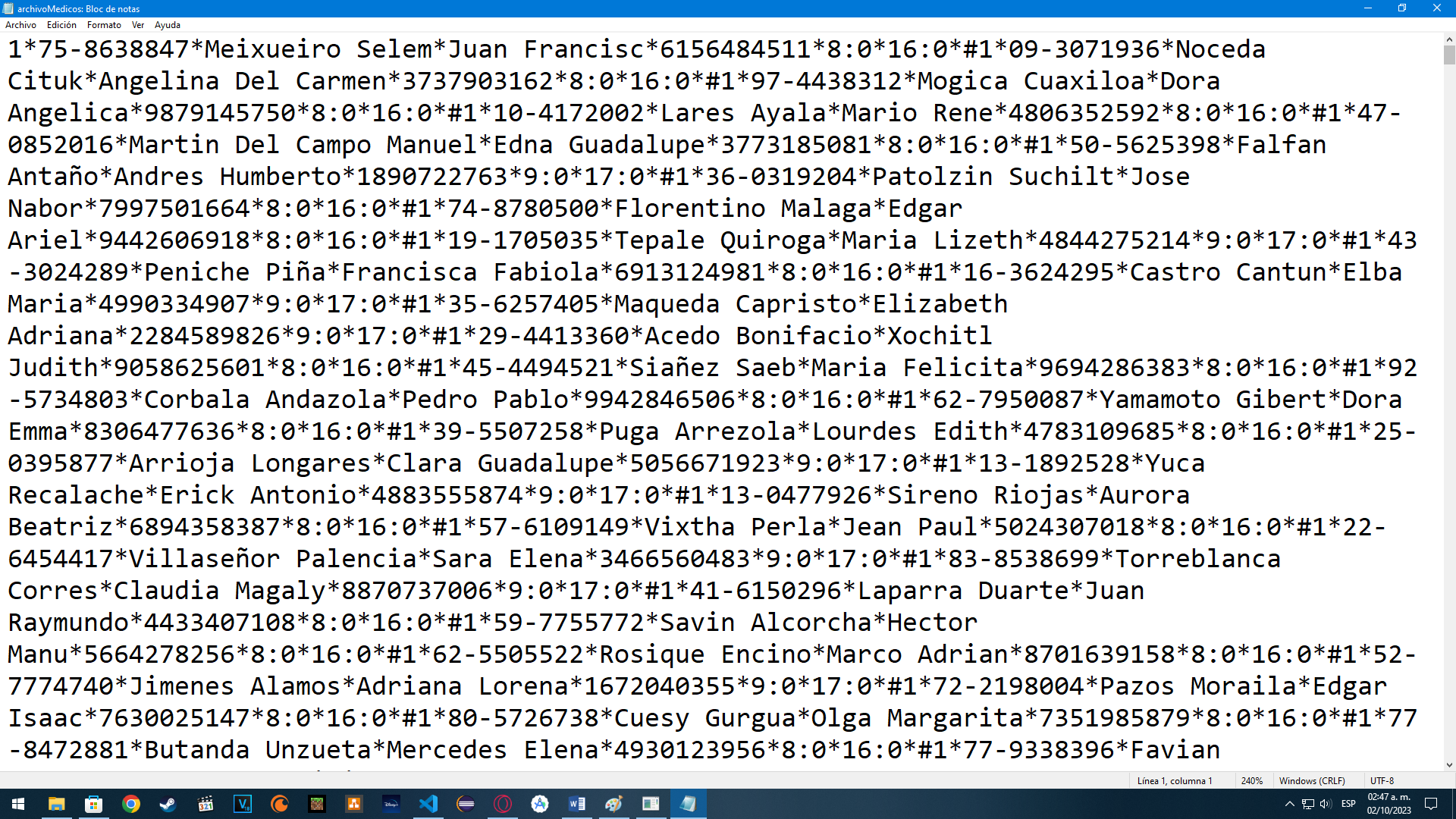
Y ahí esta el registro de Zeron, se va a borrar.



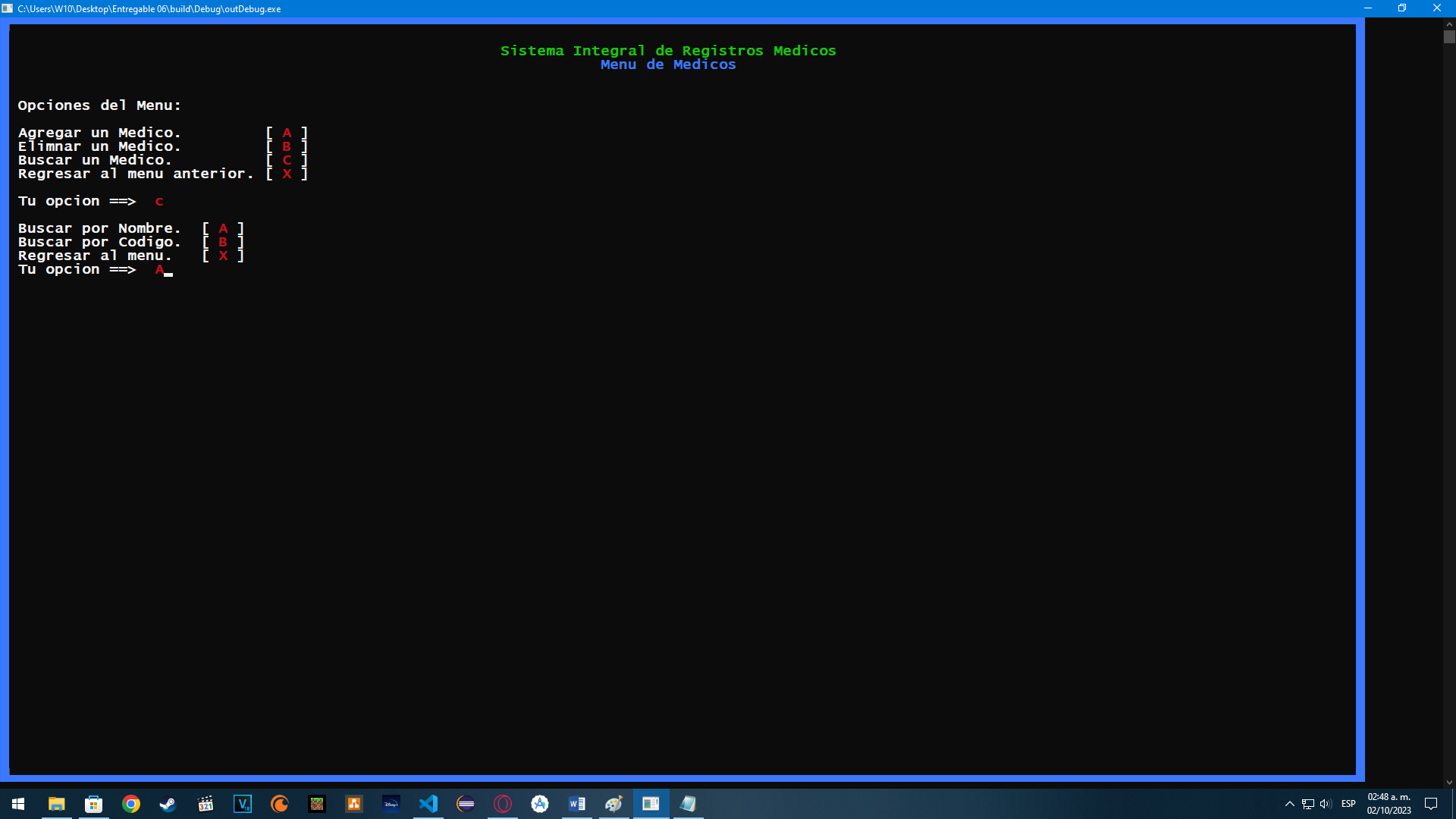
Ahora tambien borraremos otro registro:



Podemos apreciar que en el archivo de medicos ya no aparece Zeron.

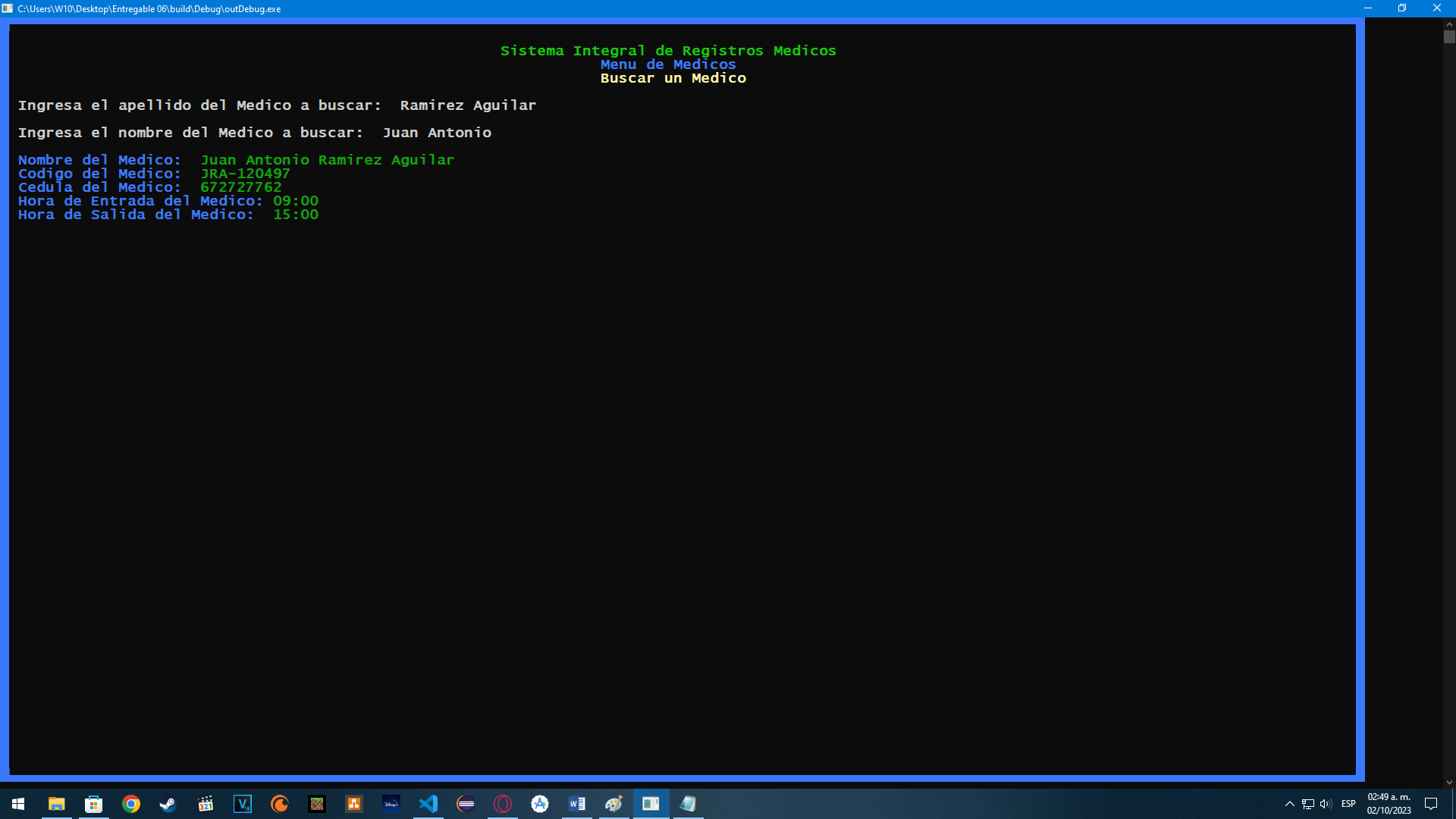


Para buscar registros tomaremos 4 ejemplos: 2 por nombre y dos por codigo. La operación buscar tambien despliega un submenu que te deja elegir con que termino vas a buscar.

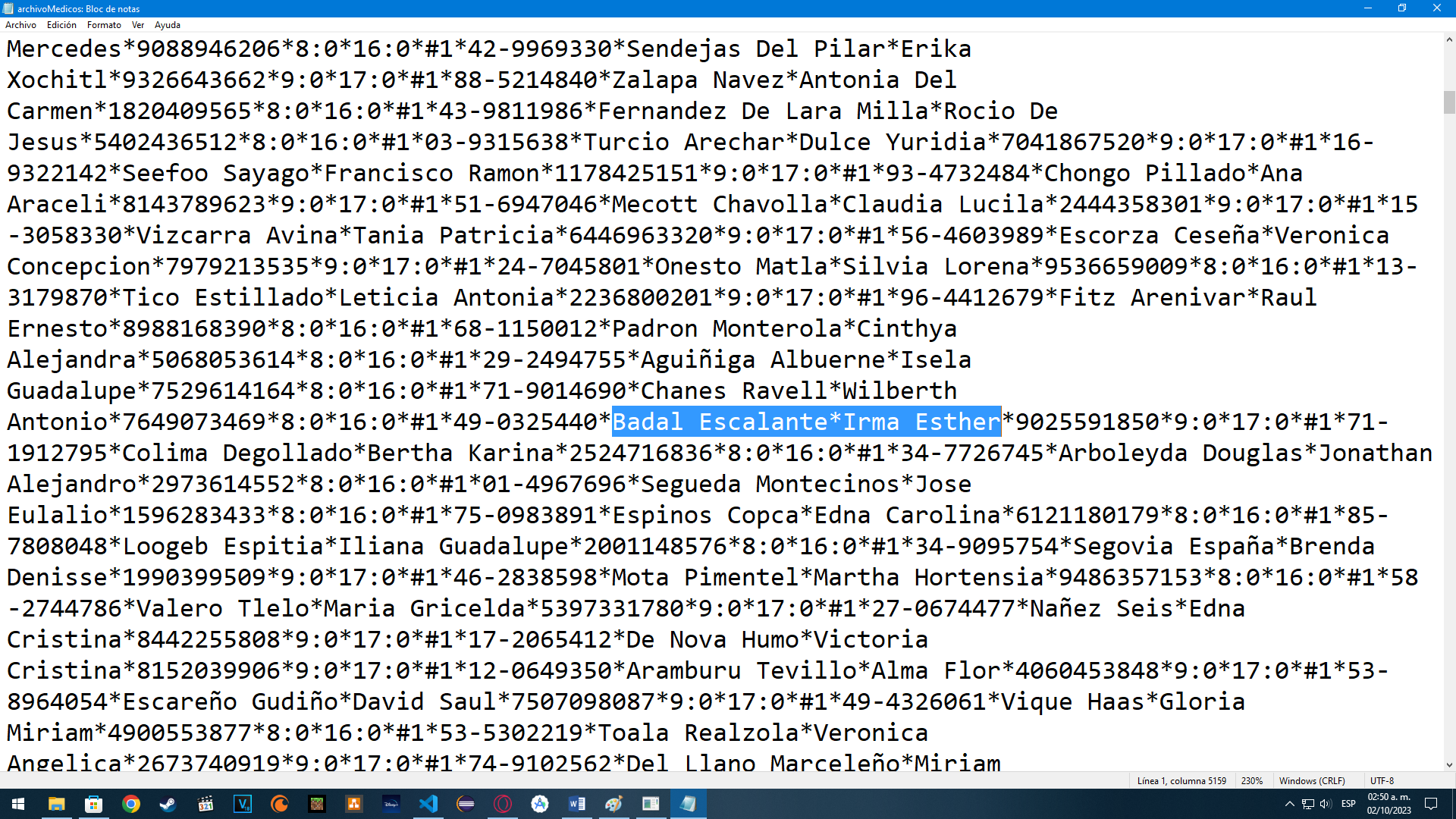


Para el primer registro a buscar, lo haremos con el registro que registramos anteriormente.

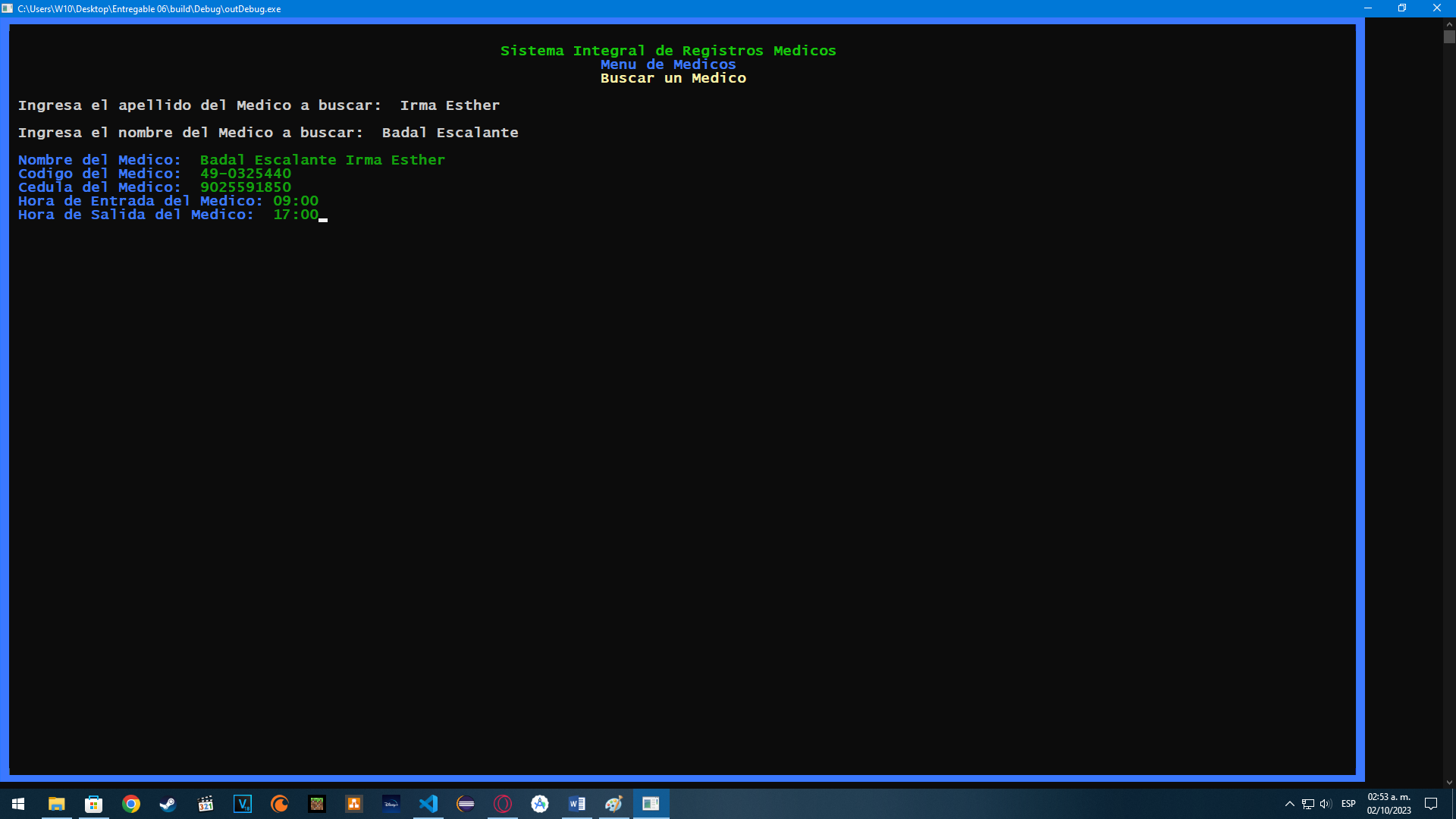
Como vemos aparece los datos del medico que buscamos.



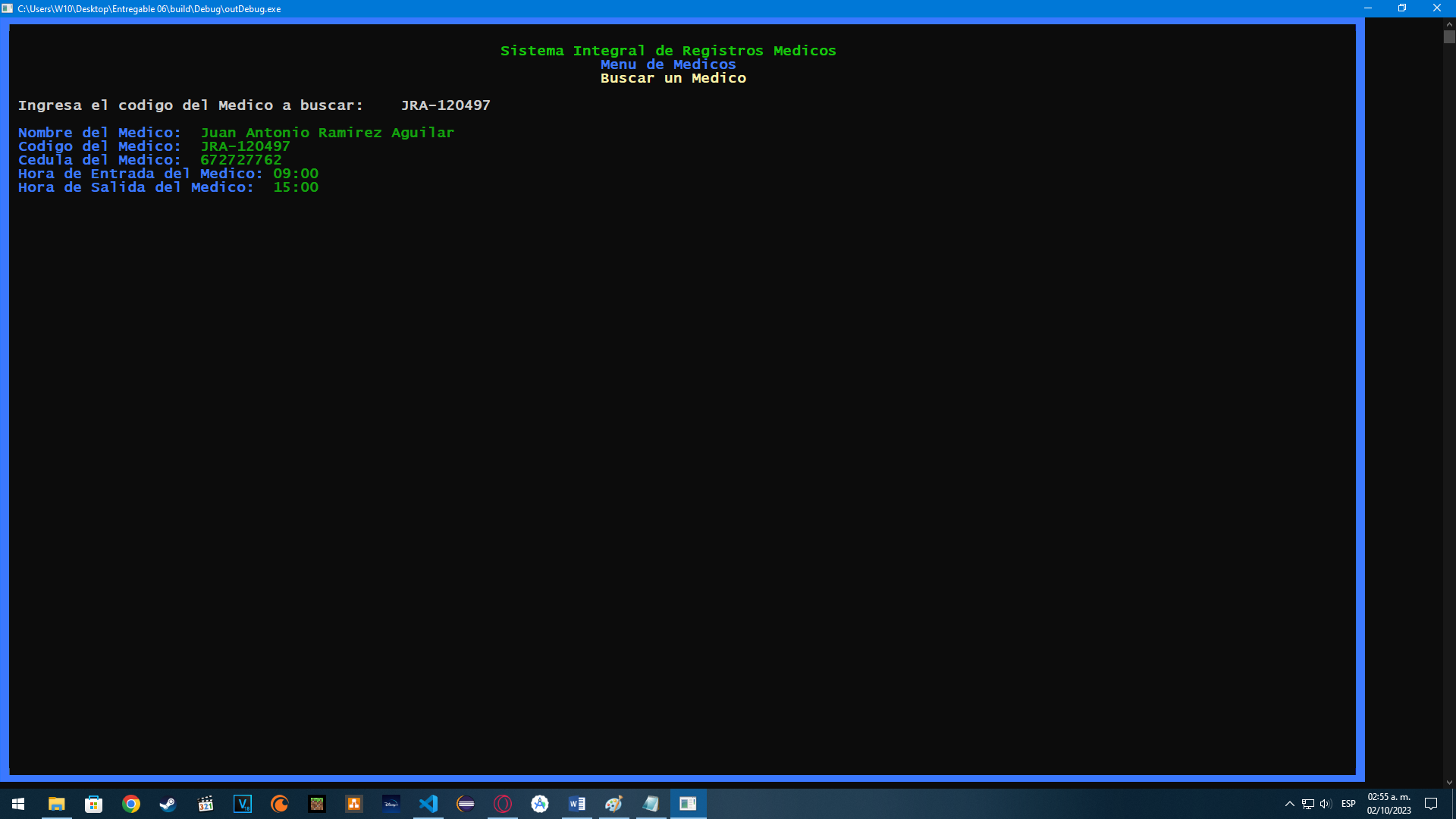
Buscaremos este registro:



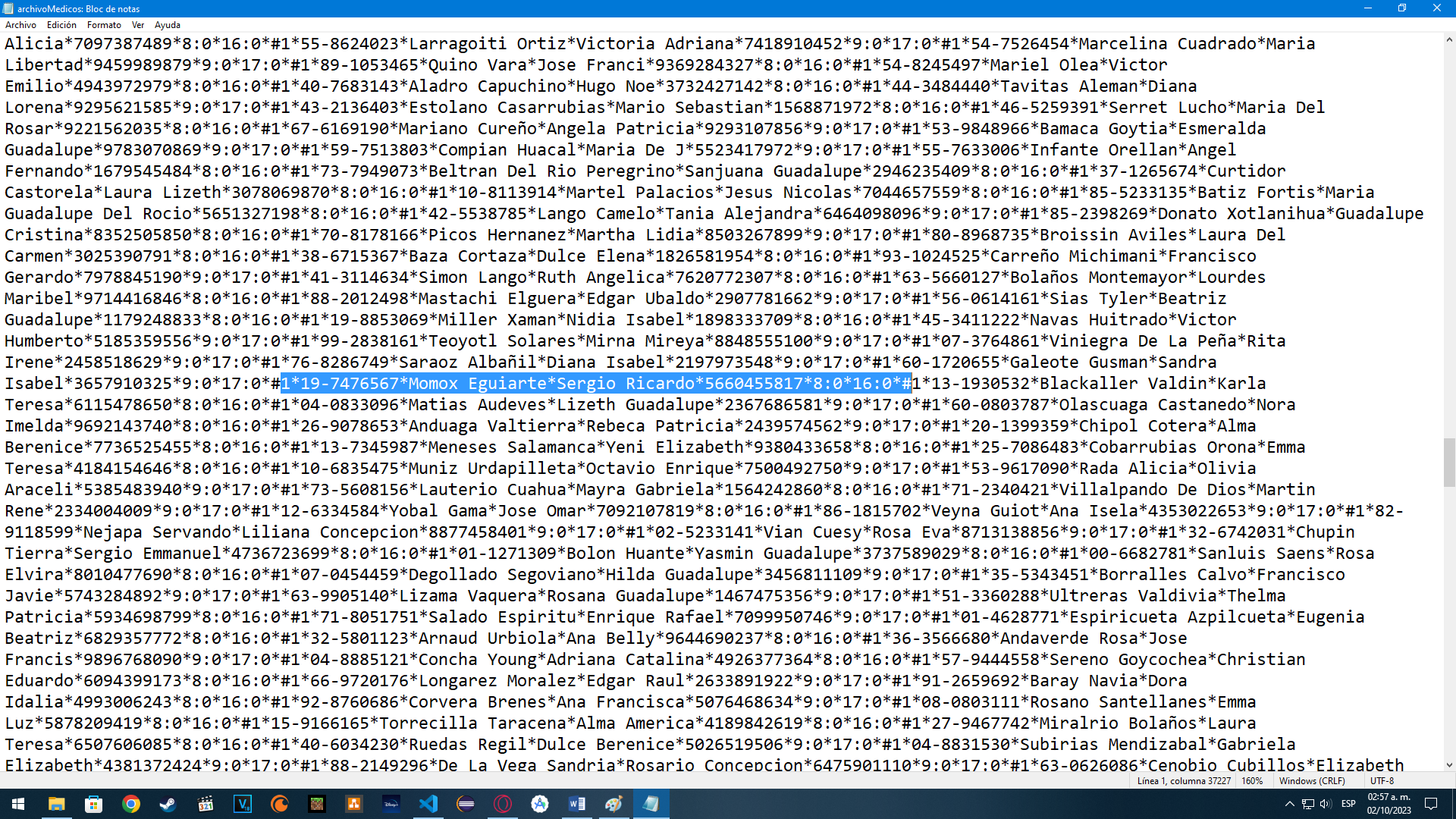
Aquí esta:

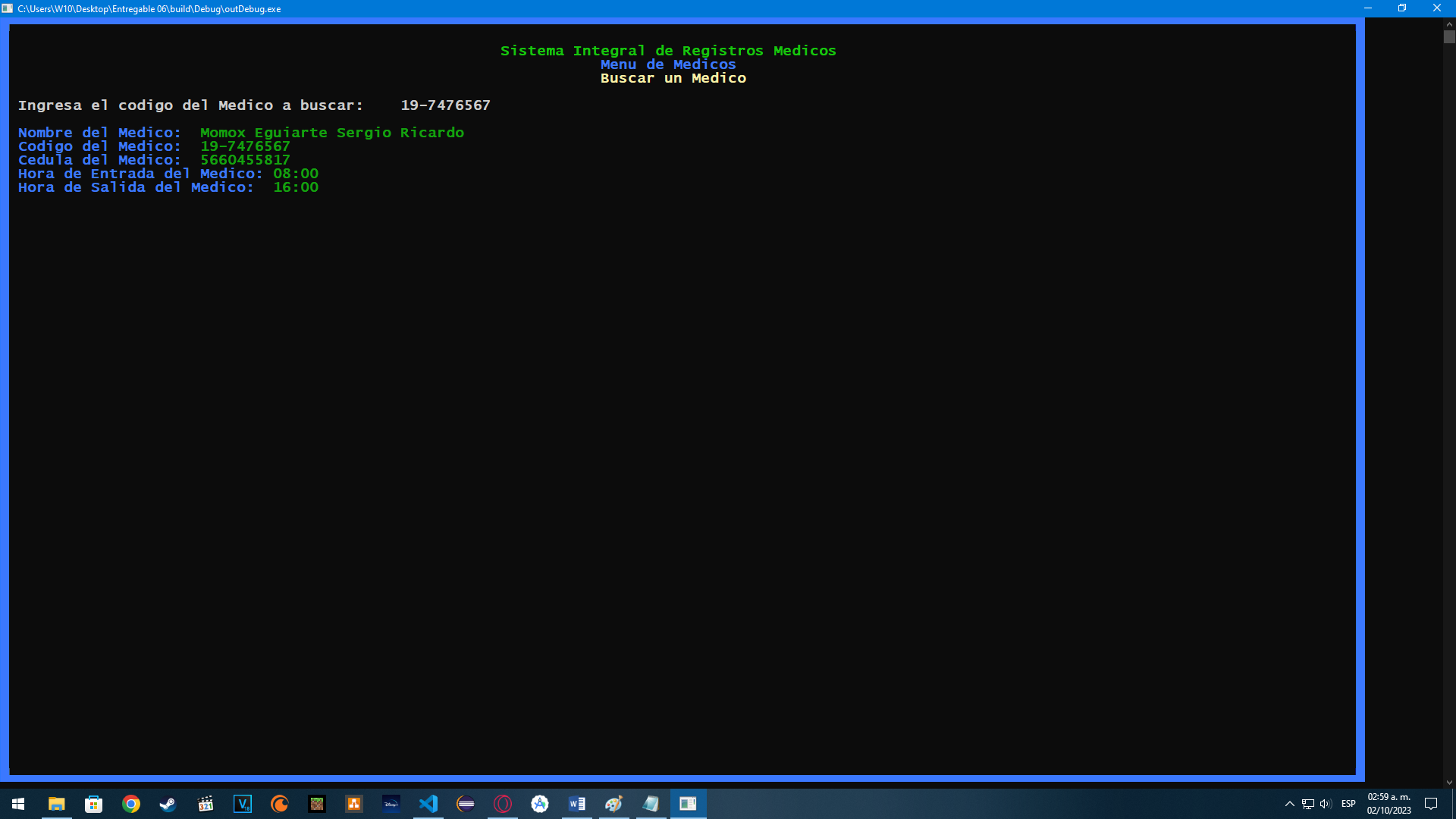


Para la primera búsqueda por código, volveremos a utilizar el registro que previamente había ingresado.



Y el ultimo código a buscar.





**Conclusión**

Este programa me trajo varios problemas, el tener todo listo y funcionando dependiendo de más de 15 archivos no es una tarea fácil, tienes que checar hasta el más mínimo error en caso de que te hayas equivocado. Me tomo más de lo esperado, pero por fin funciono y me alegro bastante.

Eh aprendido mucho más en esta clase que en catedra donde vamos más lento, ha sido difícil, pero; no es nada que no pueda hacer con mis habilidades actuales.