Taller Colaborativo

SEMANA 11

Nombre de Integrantes:
Andrés Mullo, Cristian Tutin

Introducción

La Fundación Salud para Todos, es una organización no gubernamental sin fines de lucro, la cual, a través de donaciones de empresas privadas y población civil, financia hasta un 90% de los tratamientos de pacientes con enfermedades raras en el Ecuador considerando su criticidad y situación socio económica, sin embargo, debido a la pandemia producida por el COVID 19, los ingresos de la fundación se han visto reducidos en gran medida, teniendo que dejar de financiar parcialmente o en su totalidad varios de los tratamientos que hasta antes de la pandemia se encontraban cubiertos.

En tal motivo y con el fin de volver a brindar los tratamientos completos a los pacientes, a la vez de optimizar el recurso económico, la Fundación con ayuda de entes internacionales implementó una unidad médica gestionada y administrada por la misma Fundación.

Sin embargo, una vez puesta en marcha dicha unidad médica, la Fundación se encontró con el siguiente problema:
El agendamiento de citas el cual es realizado manualmente por el personal de la fundación es deficiente, produciéndose una perdida de información y confusión en las citas agendadas.

Por lo tanto, se busca crear un sistema de agendamiento de citas que evite la pérdida de esta información tan importante. Mediante la programación se puede conseguir una manera más eficiente de almacenar las citas para poder ayudar al personal y evitar errores graves.

Justificación del trabajo

Mediante el uso de archivos, funciones, estructuras de datos en el lenguaje C se puede solucionar este problema al crear un programa que permita hacer más fácil el control de esta información que la fundación necesita para ejercer correctamente su trabajo.

Que una persona haga el agendamiento de citas manualmente puede ser demasiado contraproducente si esta persona pierde la información. Mientras que un sistema que está programado tiene mejor capacidad de almacenamiento.

Objetivos del trabajo

- Crear un sistema que permita almacenar el agendamiento de citas para solucionar el problema de la pérdida de la información.
- Optimizar el código para un correcto uso y asegurarse que su uso sea fácil de manejar para el personal de la fundación.
- Analizar los métodos necesarios para lograr crear un sistema eficiente y comprensible.

Análisis del problema elegido

Dada la importancia crucial de una gestión efectiva de datos para el agendamiento de citas, resulta fundamental que la fundación sea competente en la administración de esta información para garantizar el correcto funcionamiento de su unidad médica y ofrecer un servicio de calidad a los pacientes. Sin embargo, el sistema actual de agendamiento manual presenta una serie de inconvenientes, tales como la fatiga experimentada por los empleados durante las jornadas laborales, el riesgo de pérdida de información debido a descuidos del personal, la ambigüedad de ciertos datos debido a la caligrafía, y la limitación de espacio físico para almacenar las citas programadas. En este sentido, la implementación de un sistema de software dedicado a la gestión de estos datos y la realización de respaldos periódicos podrían ser soluciones efectivas para superar estos desafíos.

Establecimiento de propuestas de solución

Propuesta 1

Se implementará un sistema de agendamiento de citas, haciendo uso de los conocimientos en manejo de archivos planos, específicamente usando un archivo de texto en el cual se almacenará la información vital del paciente como: cédula, nombre, apellido, edad, altura, peso, fecha y hora para la cita médica y su respectivo doctor que le atenderá.

Se utilizará una estructura de datos para poder pedir y recolectar los datos antes mencionados en un archivo llamado “Agenda.txt” Además se creará un header en el cual estén todas las funciones necesarias para poder agregar, editar, eliminar y visualizar las citas, claro también estarán las variables globales.

Propuesta 2

En esta propuesta, se plantea un sistema de agendamiento de citas con un enfoque más simplista y menos eficiente en términos de organización y manejo de datos. Se prescindirá del uso de estructuras para representar la información de las citas, y en su lugar, se utilizarán variables globales para almacenar los datos directamente en el archivo "Agenda.txt". Además, todas las funciones necesarias para gestionar las citas estarán dentro del archivo principal, sin la creación de un header para modularizar el código, pues estamos buscando hacer un código “simple”.

Evaluación de propuestas de solución

Evaluación de la Propuesta 1:

La Propuesta 1 demuestra un enfoque más estructurado y eficiente en comparación con la Propuesta 2. Al implementar un sistema de agendamiento de citas basado en el uso de archivos de texto plano y

estructuras de datos, la Propuesta 1 destaca por los siguientes aspectos:

Organización y Modularidad: La utilización de un archivo header y la implementación de funciones específicas para agregar, editar, eliminar y visualizar citas proporcionan una organización clara y modularidad al código. Esto facilita el mantenimiento, la expansión y la corrección de errores.

Uso de Estructuras: La Propuesta 1 utiliza estructuras de datos para representar la información de las citas, lo cual mejora la legibilidad y facilita la manipulación de datos relacionados. Esta elección estructurada permite un código más claro y mantenible.

Manejo de Archivos: Al manejar los archivos de manera centralizada y a través de funciones dedicadas, se evitan posibles errores relacionados con la manipulación incorrecta de archivos. La propuesta también considera el límite de citas para evitar problemas de desbordamiento de memoria.

Evaluación de la Propuesta 2:

La Propuesta 2 presenta un enfoque menos óptimo en comparación con la Propuesta 1. Al utilizar un enfoque más simplista y menos estructurado, esta propuesta presenta los siguientes puntos:

Falta de Modularidad: La ausencia de un archivo header y la colocación de todas las funciones directamente en el archivo principal resultan en un código menos modular. Esto puede hacer que el código sea más difícil de entender y mantener a medida que crece en complejidad.

Uso de Variables Globales: El uso de variables globales para almacenar los datos de las citas directamente en el archivo puede conducir a problemas de mantenimiento y organización. Las variables globales pueden ser propensas a conflictos y dificultan la comprensión de la relación entre los datos.

Complejidad del Código Principal: Al tener todas las funciones y variables en un solo bloque de código, el archivo principal puede volverse más difícil de gestionar y entender, especialmente a medida que se agregan más funcionalidades.

Conclusión sobre las propuestas:

La Propuesta 1 es mejor evaluada en términos de organización, modularidad y manejo de datos, ya que utiliza estructuras y funciones modulares para lograr un código más claro y mantenible. La Propuesta 2, al carecer de estas características, se considera menos óptima en términos de eficiencia y estructura.

GitHub

https://github.com/AndresMV10/Agenda-de-citas-medicas

Explicación del algoritmo implementado en C

Agenda_citas.h

1. Este parte del código incluye las librerías necesarias para el desarrollo de las funciones, además declaramos una estructura de datos global, y un puntero al archivo.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

FILE *fd;

struct Cita {
    char cedula[15];
    char nombre[20];
    char apellido[20];
    char email[50];
    int altura;
    int peso;
    int edad;
    char fecha[11];  // Aumenté un espacio para incluir el carácter nulo '\0'
    char hora[6];    // Aumenté un espacio para incluir el carácter nulo '\0'
    char doctor[30];
};
```

2. La función agendarCita se encarga de agregar nuevas citas médicas al archivo "Agenda.txt". Primero abre el archivo en modo de escritura al final, después verificamos si la apertura fue exitosa o no, consecuentemente pedimos los datos necesarios para la cita, además utilizamos el do-while para el múltiple agendamiento de citas, finalmente escribimos los datos obtenidos en el archivo de texto, con un formato ya predefinido.

```c
void agendarCita() {
    char rpt;
    fd = fopen("Agenda.txt", "at");

    if (fd == NULL) {
        printf("Error al tratar de abrir el archivo");
        return;
    }

    printf("\n\t.:Agendando cita médica:.\n");

    do {
        struct Cita nuevaCita;

        fflush(stdin);
        printf("\nCédula: ");
        scanf("%s", nuevaCita.cedula);

        printf("Nombre: ");
        scanf("%s", nuevaCita.nombre);

        printf("Apellido: ");
        scanf("%s", nuevaCita.apellido);

        printf("Email: ");
        scanf("%s", nuevaCita.email);

        printf("Altura (cm): ");
        scanf("%d", &nuevaCita.altura);

        printf("Peso (kg): ");
        scanf("%d", &nuevaCita.peso);
```

```
        printf("Edad: ");
        scanf("%d", &nuevaCita.edad);

        printf("Fecha de la cita (DD/MM/AAAA): ");
        scanf("%s", nuevaCita.fecha);

        printf("Hora de la cita (HH:mm): ");
        scanf("%s", nuevaCita.hora);

        printf("Doctor para la cita: ");
        scanf("%s", nuevaCita.doctor);

        fprintf(fd, "Cédula: %s\nNombre: %s\nApellido: %s\nEmail: %s\nAltura: %d\nPeso: %d\nE
                nuevaCita.cedula, nuevaCita.nombre, nuevaCita.apellido, nuevaCita.email,
                nuevaCita.altura, nuevaCita.peso, nuevaCita.edad, nuevaCita.fecha,
                nuevaCita.hora, nuevaCita.doctor);
        printf("\nDesea agendar más citas (s): ");
        scanf(" %c", &rpt);
    } while (rpt == 's');
    fclose(fd);
}
```

3. La función visualizarCitas tiene como objetivo mostrar en la consola el contenido del archivo "Agenda.txt", que almacena la información de las citas médicas. Primero, se intenta abrir el archivo en modo de lectura ("r"), y se verifica si la operación de apertura fue exitosa. En caso de error, se muestra un mensaje y se abandona la función. Luego, se utiliza un bucle while para leer cada carácter del archivo hasta alcanzar el final del mismo (EOF). Durante este proceso, cada carácter se imprime en la consola utilizando la función putchar. Finalmente, se cierra el archivo para liberar recursos. En resumen, la función proporciona una visualización simple del contenido del archivo de citas médicas en la consola.

```c
void visualizarCitas() {
    int c;
    fd = fopen("Agenda.txt", "r");

    if (fd == NULL) {
        printf("Error al tratar de abrir el archivo");
        return;
    }

    while ((c = fgetc(fd)) != EOF) {
        putchar(c);
    }

    fclose(fd);
}
```

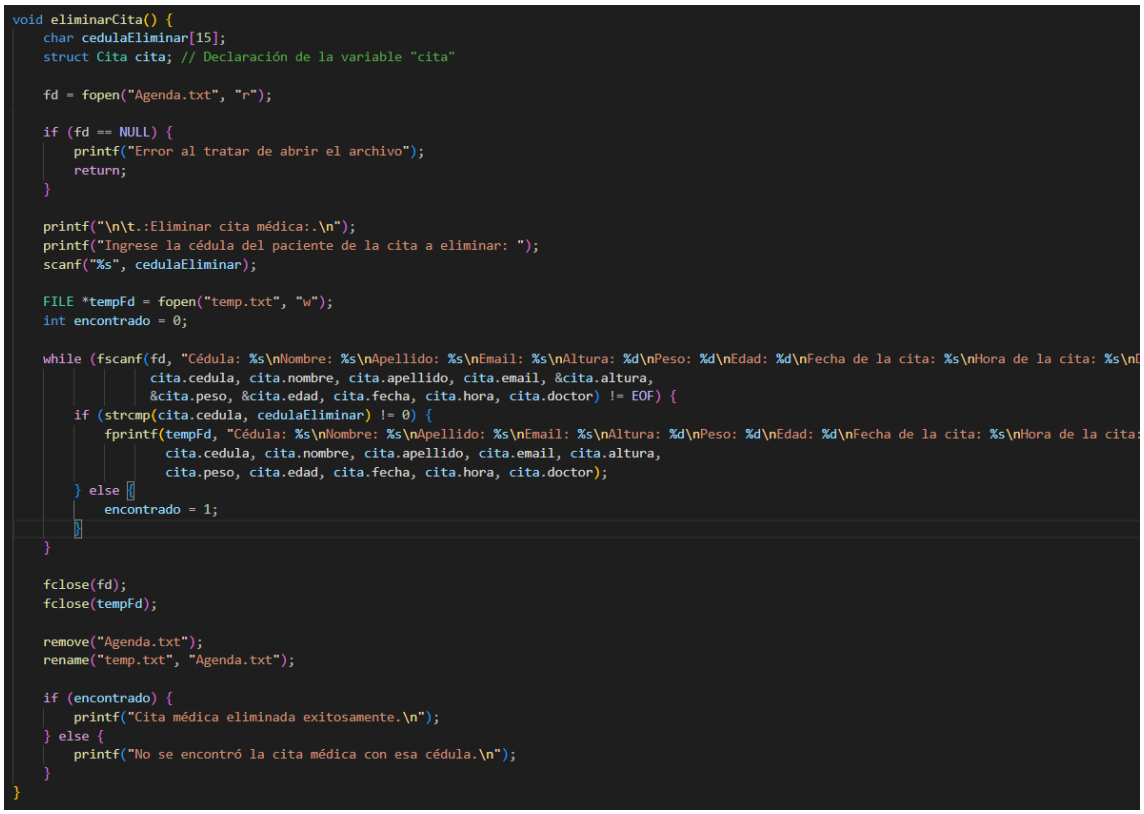
4. La función editarCita permite al usuario modificar los detalles de una cita médica existente. Primero, se solicita la cédula del paciente cuya cita se desea editar. Luego, se lee el contenido del archivo "Agenda.txt" y se compara la cédula proporcionada con la cédula de cada cita. Si se encuentra una coincidencia, se permiten las modificaciones en altura, peso, edad, fecha, hora y doctor para esa cita específica. Los cambios se escriben en un archivo temporal ("temp.txt"). Al finalizar, se reemplaza el archivo original con el archivo temporal y se informa al usuario sobre el éxito o la falta de coincidencia con la cédula buscada.

```c
void editarCita() {
    char cedulaEditar[15];
    struct Cita cita; // Declaración de la variable "cita"

    fd = fopen("Agenda.txt", "r");

    if (fd == NULL) {
        printf("Error al tratar de abrir el archivo");
        return;
    }

    printf("\n\t.:Editar cita médica:.\n");
    printf("Ingrese la cédula del paciente de la cita a editar: ");
    scanf("%s", cedulaEditar);

    FILE *tempFd = fopen("temp.txt", "w");
    int encontrado = 0;

    while (fscanf(fd, "Cédula: %s\nNombre: %s\nApellido: %s\nEmail: %s\nAltura: %d\nPeso: %d\nEdad: %d\nFecha de la cita: %s\nHora de la ci
        cita.cedula, cita.nombre, cita.apellido, cita.email, &cita.altura,
        &cita.peso, &cita.edad, cita.fecha, cita.hora, cita.doctor) != EOF) {
        if (strcmp(cita.cedula, cedulaEditar) == 0) {
            printf("Nueva altura para %s: ", cita.nombre);
            scanf("%d", &cita.altura);

            printf("Nuevo peso para %s: ", cita.nombre);
            scanf("%d", &cita.peso);

            printf("Nueva edad para %s: ", cita.nombre);
            scanf("%d", &cita.edad);

            printf("Nueva fecha de la cita para %s (DD/MM/AAAA): ", cita.nombre);
            scanf("%s", cita.fecha);

            printf("Nueva hora de la cita para %s (HH:mm): ", cita.nombre);
            scanf("%s", cita.hora);

            printf("Nuevo doctor para la cita para %s: ", cita.nombre);
            scanf("%s", cita.doctor);

            encontrado = 1;
        }
```

```c
            fprintf(tempFd, "Cédula: %s\nNombre: %s\nApellido: %s\nEmail: %s\nAltura: %d\nPeso: %d\nEdad: %d\nFecha de la cita:
                    cita.cedula, cita.nombre, cita.apellido, cita.email, cita.altura,
                    cita.peso, cita.edad, cita.fecha, cita.hora, cita.doctor);
        }

        fclose(fd);
        fclose(tempFd);

        remove("Agenda.txt");
        rename("temp.txt", "Agenda.txt");

        if (encontrado) {
            printf("Cita médica editada exitosamente.\n");
        } else {
            printf("No se encontró la cita médica con esa cédula.\n");
        }
    }
```

5. La función eliminarCita permite al usuario eliminar una cita médica proporcionando la cédula del paciente asociado. Se abre el archivo "Agenda.txt" para lectura y se solicita la cédula a eliminar. Se crea un archivo temporal ("temp.txt") donde se copian todas las citas excepto aquella con la cédula a eliminar. Luego, se reemplaza el archivo original con el archivo temporal, indicando al usuario si la

cita fue eliminada con éxito o si no se encontró una cita con la cédula proporcionada.



```c
void eliminarCita() {
    char cedulaEliminar[15];
    struct Cita cita; // Declaración de la variable "cita"

    fd = fopen("Agenda.txt", "r");

    if (fd == NULL) {
        printf("Error al tratar de abrir el archivo");
        return;
    }

    printf("\n\t.:Eliminar cita médica:.\n");
    printf("Ingrese la cédula del paciente de la cita a eliminar: ");
    scanf("%s", cedulaEliminar);

    FILE *tempFd = fopen("temp.txt", "w");
    int encontrado = 0;

    while (fscanf(fd, "Cédula: %s\nNombre: %s\nApellido: %s\nEmail: %s\nAltura: %d\nPeso: %d\nEdad: %d\nFecha de la cita: %s\nHora de la cita: %s\nD
                  cita.cedula, cita.nombre, cita.apellido, cita.email, &cita.altura,
                  &cita.peso, &cita.edad, cita.fecha, cita.hora, cita.doctor) != EOF) {
        if (strcmp(cita.cedula, cedulaEliminar) != 0) {
            fprintf(tempFd, "Cédula: %s\nNombre: %s\nApellido: %s\nEmail: %s\nAltura: %d\nPeso: %d\nEdad: %d\nFecha de la cita: %s\nHora de la cita:
                    cita.cedula, cita.nombre, cita.apellido, cita.email, cita.altura,
                    cita.peso, cita.edad, cita.fecha, cita.hora, cita.doctor);
        } else {
            encontrado = 1;
        }
    }

    fclose(fd);
    fclose(tempFd);

    remove("Agenda.txt");
    rename("temp.txt", "Agenda.txt");

    if (encontrado) {
        printf("Cita médica eliminada exitosamente.\n");
    } else {
        printf("No se encontró la cita médica con esa cédula.\n");
    }
}
```

Proyecto_1.c

1. Agregamos el archivo “Agenda_citas.h”, la cual es una librería que contiene las funciones antes descritas.

2. Creamos un menú simple en el cual el usuario puede eligir entre las cinco opciones existentes mediante el uso de los números en el teclado, además controlamos que no se ingrese caracteres, puesto que este generaría un bucle infinito.

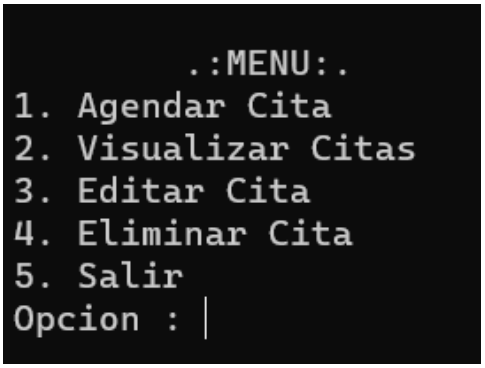
```c
int main() {
    int opc;

    do {
        printf("\n\t.:MENU:.\n");
        printf("1. Agendar Cita\n");
        printf("2. Visualizar Citas\n");
        printf("3. Editar Cita\n");
        printf("4. Eliminar Cita\n");
        printf("5. Salir\n");

        printf("Opcion : ");
        if (scanf("%i", &opc) != 1) {
            // Limpiar el búfer de entrada en caso de entrada no válida
            scanf("%*s");
            printf("Por favor, ingrese un número válido.\n");
            continue;  // Vuelve al principio del bucle
        }
```

3. Este bloque de código implementa un switch para ejecutar acciones específicas según la opción seleccionada por el usuario en el menú anterior. Dependiendo del valor de la variable opc, que representa la opción elegida, se ejecuta una de las siguientes funciones: agendarCita (1), visualizarCitas (2), editarCita (3), eliminarCita (4). Si el usuario elige salir (opción 5), se muestra un mensaje de salida y el bucle del menú se termina, finalizando así la ejecución del programa. En caso de que la opción ingresada no coincida con ninguna de las anteriores, se informa al usuario que la opción no es válida y se le solicita que elija una opción del 1 al 5. El bucle continuará hasta que el usuario elija salir (opción 5).

```c
        switch (opc) {
            case 1:
                agendarCita();
                break;
            case 2:
                visualizarCitas();
                break;
            case 3:
                editarCita();
                break;
            case 4:
                eliminarCita();
                break;
            case 5:
                printf("Saliendo...\n");
                break;
            default:
                printf("Opción no válida. Por favor, elija una opción del 1 al 5.\n");
        }

    } while (opc != 5);
    return 0;
}
```

Explicación de la ejecución de cada sección del programa

1. Al ejecutar el programa, nos aparece esta pantalla con un menú



2. Al elegir la primera opción podemos ver que funciona bien, pues nos pide los datos y los almacena en el archivo de texto.

```
Cédula: 0705536696
Nombre: Josue
Apellido: Mullo
Email: josuemullo19@gmail.com
Altura: 170
Peso: 75
Edad: 20
Fecha de la cita: 24/01/2024
Hora de la cita: 16:45
Doctor para la cita: Fernando

Cédula: 0705536687
Nombre: Alex
Apellido: Vega
Email: alexajmw@hotmail.com
Altura: 168
Peso: 80
Edad: 19
Fecha de la cita: 25/01/2024
Hora de la cita: 12:00
Doctor para la cita: Mario
```

```
        .:MENU:.
1. Agendar Cita
2. Visualizar Citas
3. Editar Cita
4. Eliminar Cita
5. Salir
Opcion : 1

        .:Agendando cita m├®dica:.

C├®dula: 0705536696
Nombre: Josue
Apellido: Mullo
Email: josuemullo19@gmail.com
Altura (cm): 170
Peso (kg): 75
Edad: 20
Fecha de la cita (DD/MM/AAAA): 24/01/2024
Hora de la cita (HH:mm): 16:45
Doctor para la cita: Fernando
```

3. Al probar la función de visualizar cita, se puede comprobar que también funciona, pues muestra en la consola las citas agendadas.

```
        .:MENU:.
1. Agendar Cita
2. Visualizar Citas
3. Editar Cita
4. Eliminar Cita
5. Salir
Opcion : 2
C├®dula: 0705536696
Nombre: Josue
Apellido: Mullo
Email: josuemullo19@gmail.com
Altura: 170
Peso: 75
Edad: 20
Fecha de la cita: 24/01/2024
Hora de la cita: 16:45
Doctor para la cita: Fernando

C├®dula: 0705536687
Nombre: Alex
Apellido: Vega
Email: alexajmw@hotmail.com
Altura: 168
Peso: 80
Edad: 19
```

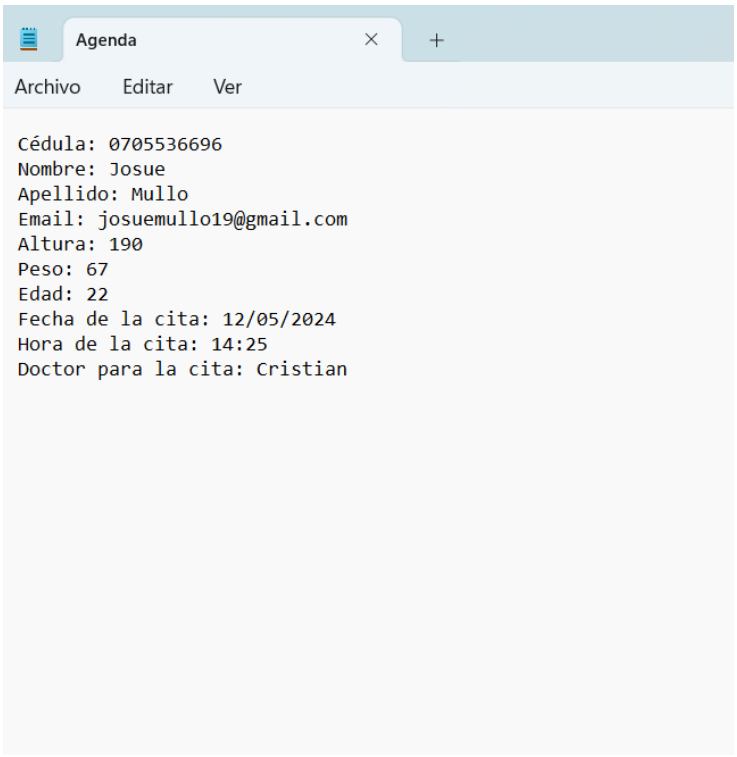
4. Al elegir la función de editar cita, podemos ver que el programa nos pide la cedula como identificador de la cita, y procede a pedir nuevos datos para el paciente. Podemos comprobar que funciona bien, al comparar los datos actuales del paciente Josue, con los datos anteriores, los cuales están en capturas anteriores.

```
        .:MENU:.
1. Agendar Cita
2. Visualizar Citas
3. Editar Cita
4. Eliminar Cita
5. Salir
Opcion : 3

        .:Editar cita m├®dica:.
Ingrese la c├®dula del paciente de la cita a editar: 0705536696
Nueva altura para Josue: 190
Nuevo peso para Josue: 67
Nueva edad para Josue: 22
Nueva fecha de la cita para Josue (DD/MM/AAAA): 12/05/2024
Nueva hora de la cita para Josue (HH:mm): 14:25
Nuevo doctor para la cita para Josue: Cristian
Cita m├®dica editada exitosamente.
```

```
Cédula: 0705536696
Nombre: Josue
Apellido: Mullo
Email: josuemullo19@gmail.com
Altura: 190
Peso: 67
Edad: 22
Fecha de la cita: 12/05/2024
Hora de la cita: 14:25
Doctor para la cita: Cristian

Cédula: 0705536687
Nombre: Alex
Apellido: Vega
Email: alexajmw@hotmail.com
Altura: 168
Peso: 80
Edad: 19
Fecha de la cita: 25/01/2024
Hora de la cita: 12:00
Doctor para la cita: Mario
```

5. **Por ultimo probamos la función de eliminar cita, el programa nos pide la cedula del paciente que tiene cita para poder eliminarla de la agenda.**

```
        .:MENU:.
1. Agendar Cita
2. Visualizar Citas
3. Editar Cita
4. Eliminar Cita
5. Salir
Opcion : 4

        .:Eliminar cita m├®dica:.
Ingrese la c├®dula del paciente de la cita a eliminar: 0705536687
Cita m├®dica eliminada exitosamente.
```



Recomendaciones

- Colocar las librerías principales, datos de salida y parte del proceso en un header ya que de esa manera se ahorrará espacio

en las líneas de código del programa en C, para poder evitar errores y tener una mejor organización.

- Usar una sentencia switch para poder crear el menú, de las sentencias de repetición es el que tiene una estructura que puede facilitar el proceso.
- Las estructuras de repetición if son muy útiles para evitar que se ingresen datos erróneos al momento de pedir información del producto. Por ejemplo, que se ingresen números cuando se pide el nombre.

Conclusiones

En conclusión, el sistema de inventario es necesario para facilitar el almacenamiento de los productos de una tienda, para realizar correctamente este sistema se debe tener en cuenta el uso correcto de las sentencias de repetición, de condición y las funciones ya que de esta forma el código puede resultar más funcional.