ISWZ1102 - Programación I

Proyecto Final RC2

Jose Sanchez - Alessandro Paredes - Alejandro Vasquez

Facultad de Ingeniería y Ciencias Aplicadas

06 JULIO 2025

1. Introducción

- En la actualidad, el correcto seguimiento de tratamientos médicos representa un desafío para muchas personas, especialmente para adultos mayores.

  La falta de organización o el olvido en la toma de medicamentos puede generar complicaciones en la salud, disminuyendo la eficacia del tratamiento y afectando el bienestar del paciente. Frente a esta problemática, se presenta el desarrollo de un sistema de software que permite registrar, visualizar y actualizar la información relacionada con medicamentos de uso personal.

  Este sistema, construido en lenguaje de programación C, tiene como finalidad facilitar la gestión de medicamentos, permitiendo al usuario consultar las dosis, horarios y frecuencias de administración, así como marcar aquellos que ya han sido tomados.

  Este proyecto no solo fortalece las habilidades de programación estructurada mediante el uso de estructuras, archivos y modularidad, sino que también responde a una necesidad social real, integrando consideraciones técnicas con un enfoque práctico y humanitario.

  El presente informe expone el proceso completo de análisis, diseño, desarrollo y validación de la solución implementada.
-

2. Objetivos

- Diseñar una solución de software utilizando técnicas y métodos óptimos considerando las restricciones y limitaciones del problema dentro de los ámbitos sociales, ambientales y económicos.

- Realizar un análisis integral del problema, identificando sus principales variables y restricciones; para plantear varias alternativas de solución y evaluarlas para seleccionar aquella que resuelva de manera integral el problema identificado.

- Diseñar una solución de software utilizando técnicas y métodos óptimos considerando las restricciones y limitaciones del problema dentro de los ámbitos sociales, ambientales y económicos.

3. Tabla con la formulación del problema e identificación de variables

Nombre del sistema:	**MedControl**
Usuarios:	• Pacientes adultos que necesitan llevar un control organizado de sus medicamentos. • Personas mayores que requieren apoyo para recordar horarios y dosis. • Cuidadores o familiares que gestionan el tratamiento de un ser querido. • Estudiantes de programación que usan el sistema como práctica educativa. • Personal de salud en entornos pequeños o domiciliarios que desea registrar medicamentos sin necesidad de sistemas complejos.
Objetivo del sistema:	Facilitar el registro, consulta y control de medicamentos personales, permitiendo a los usuarios gestionar de forma sencilla la información sobre dosis, horarios y frecuencia de toma, y mantener actualizado su tratamiento mediante un sistema interactivo, accesible y de fácil uso.
Variables de entrada:	• Nombre del medicamento • Dosis • Frecuencia en horas • Hora de la próxima toma • Minutos de la próxima toma • Formato horario AM/PM • Opción del menú seleccionada por el usuario
Variables de salida:	• Lista de medicamentos registrados, incluyendo: Nombre del medicamento, dosis, frecuencia de administración, próxima hora de toma. • Mensajes de confirmación. • Mensajes de error o advertencia. • Menú principal del sistema.
Procesos asociados:	• Carga de medicamentos. • Visualización de medicamentos registrados en pantalla con sus datos completos. • Ingreso de nuevos medicamentos, con

	captura y validación de datos ingresados por el usuario. • Cálculo de próxima toma al marcar un medicamento como tomado, sumando la frecuencia horaria. • Actualización y almacenamiento de los datos en el archivo para mantener persistencia. • Navegación por menú interactivo mediante selección numérica.

Restricciones	• El nombre del medicamento debe ingresarse con exactitud para que pueda ser reconocido al marcarlo como tomado.
	• El formato horario debe respetar los valores válidos, es decir:
	  ○ Horas entre 1 y 12
	  ○ Minutos entre 0 y 59
	  ○ Indicador de AM/PM correcto
	• El usuario debe recordar qué medicamento tomó y escribirlo manualmente para actualizar la hora de la próxima toma.

Limitaciones	• Lenguaje de programación fijo: C, sin acceso a librerías gráficas ni bases de datos avanzadas. • Almacenamiento en archivo plano .txt, lo que limita la seguridad, escalabilidad y estructura de los datos. • Capacidad limitada a 100 medicamentos, debido al tamaño fijo del arreglo Medicamento medicamentos[100]. • Sin manejo de errores sofisticado, como validación profunda de entrada o recuperación ante fallos del archivo. • No hay interfaz gráfica, el sistema es completamente por consola, lo que puede ser menos intuitivo para algunos usuarios. • Cálculo de próxima toma básico, no considera fechas, solo suma horas en formato de 24 horas. • Los cambios se guardan solo tras ciertas acciones como al agregar un

	medicamento, no hay auto-guardado continuo.

4. Planteamiento de Alternativas de Solución

- Alternativa **1 – Arreglos de estructuras + Archivos**

  - Utiliza un arreglo de estructuras Medicamento para almacenar los datos.
  - Lee y guarda los datos en un archivo .txt.
  - Se emplea programación modular con funciones como cargarMedicamentos, guardarMedicamentos, etc.
  - Utiliza funciones como stricmp para comparación de nombres sin importar mayúsculas/minúsculas.

- **Alternativa 2 – Punteros dinámicos y estructuras enlazadas**
  - Implementar listas enlazadas para almacenar los medicamentos de forma dinámica.
  - Almacenar los datos en archivos binarios para mayor eficiencia.
  - Utilización de punteros para recorrer la lista y asignar memoria en tiempo de ejecución.

5. Análisis de Alternativas

Criterio	**Alternativa 1**	**Alternativa 2**
Facilidad de implementación	Alta, uso de estructuras básicas	Media-baja, requiere punteros y manejo dinámico
Claridad del código	Clara, modular y lineal	Menos clara, mayor complejidad
Uso de memoria	Fijo (100 elementos)	Dinámico (asigna según uso)
Velocidad de ejecución	Rápida para pocos medicamentos	Similar, pero mejor escalabilidad
Dificultad para estudiantes	Baja (ideal para nivel principiante)	Alta (requiere dominio de memoria dinámica)

a. ¿Cuál es la más fácil de implementar y por qué?

La Alternativa 1 es la más fácil de implementar, ya que se basa en arreglos estáticos y estructuras simples, permitiendo una mejor comprensión del flujo del programa para estudiantes de nivel básico.

b. ¿Qué aspectos son importantes al elegir una solución eficiente?

Se deben considerar la claridad del código, la eficiencia de uso de memoria, la facilidad de mantenimiento, y que el programa sea escalable sin complicar la estructura innecesariamente.

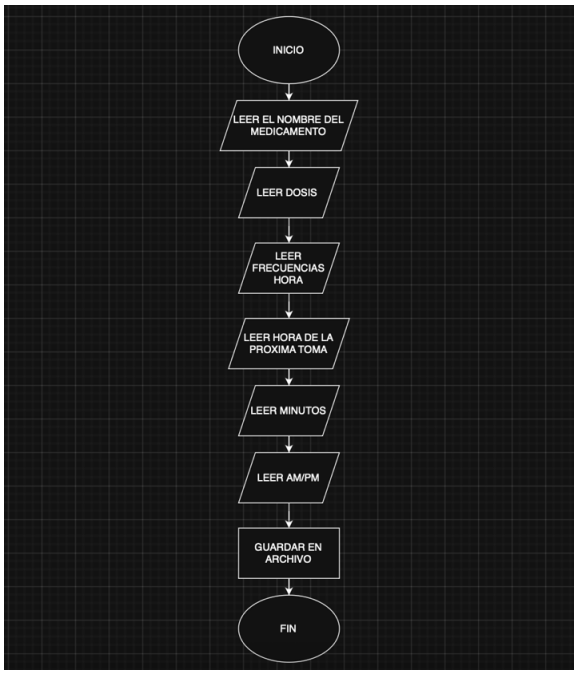
6. Selección de la mejor alternativa de solución, con su respectiva fundamentación técnica y teórica

La Alternativa 1 es la alternativa que seleccionamos ya que es la más adecuada para el nivel de conocimientos requerido, tiene el menor riesgo de errores con punteros, y cumple con todos los requerimientos del proyecto utilizando estructuras, arreglos, archivos y funciones.

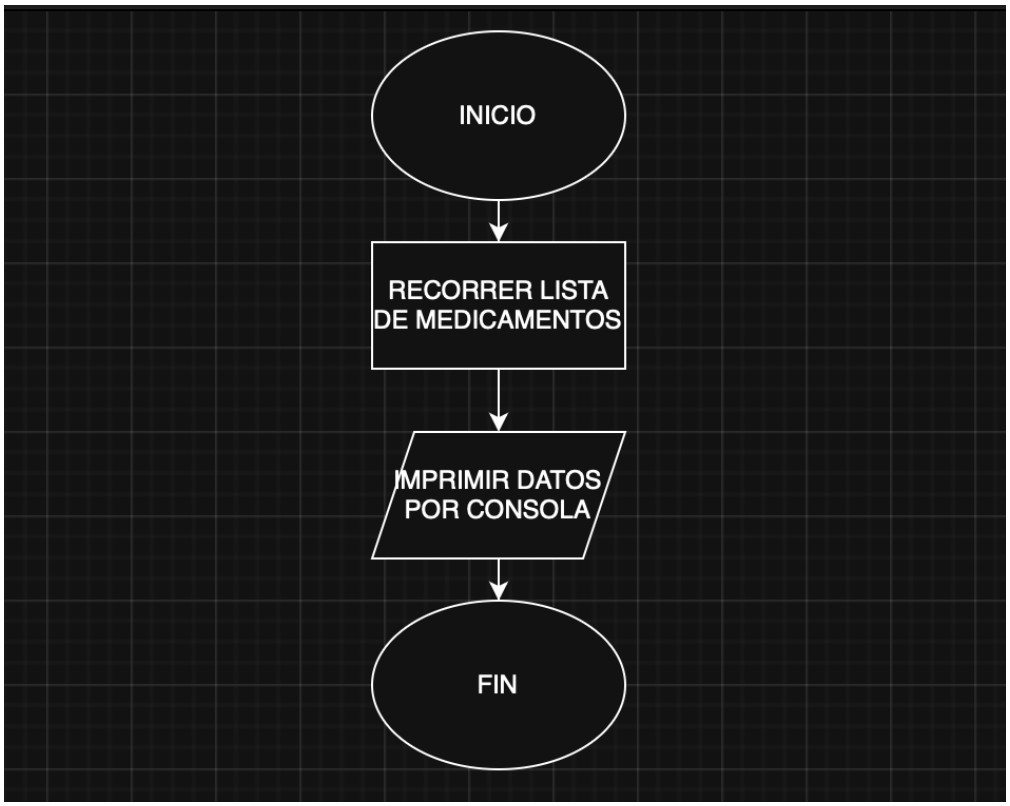
7. Desarrollo de la alternativa seleccionada

7.1 Diagrama de flujo del programa

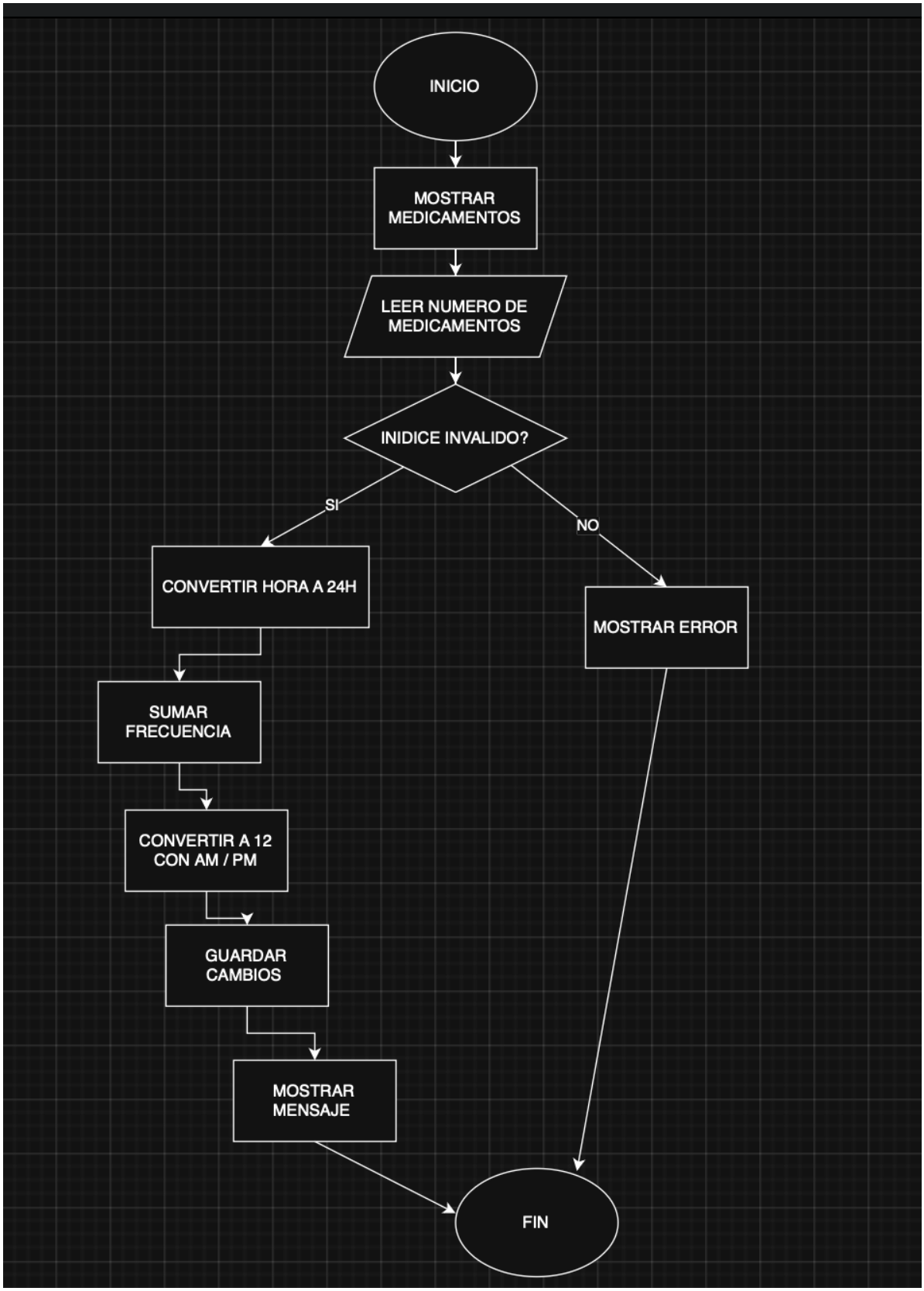
MAIN.c



MEDICAMENTOS.h



MEDICAMENTOS.c



7.2 Enlace de Github

https://github.com/JOSE-2144/PROYECTO_FINAL_PROGRAMACION.git

Insertar gcc main.c medicamento.c -o medicamento.exe

.\medicamento.exe

Para correr el programa en la terminal

7.3 Imágenes de las partes más importantes del código explicadas

1.

```c
    do {
        printf("\n--- SISTEMA DE MEDICAMENTOS ---\n");
        printf("1. Ver medicamentos\n");
        printf("2. Registrar nuevo medicamento\n");
        printf("3. Marcar medicamento como tomado\n");
        printf("4. Salir\n");
        printf("Seleccione una opcion: ");
        scanf("%d", &opcion);

        switch (opcion) {
            case 1:
                mostrarMedicamentos(medicamentos, cantidad);
                break;
            case 2:
                agregarMedicamento(medicamentos, &cantidad);
                break;
            case 3:
                marcarComoTomado(medicamentos, cantidad);
                break;
            case 4:
                printf("Hasta pronto.\n");
                break;
            default:
                printf("Opcion invalida.\n");
        }
    } while (opcion != 4);
```

Esta sección controla el flujo del programa y permite al usuario interactuar con las principales funciones: ver, agregar o marcar medicamentos.

2.

```c
typedef struct {
    char nombre[50];
    char dosis[30];
    int frecuencia_horas;
    int hora;
    int minutos;
    char am_pm[3];
} Medicamento;
```

Define la estructura Medicamento, que contiene los campos necesarios para almacenar toda la información de cada medicina.

3.

```c
void agregarMedicamento(Medicamento meds[], int *n) {
    printf("Nombre del medicamento: ");
    getchar();
    fgets(meds[*n].nombre, 50, stdin);
    meds[*n].nombre[strcspn(meds[*n].nombre, "\n")] = 0;

    printf("Dosis (ej: 1 tableta): ");
    fgets(meds[*n].dosis, 30, stdin);
    meds[*n].dosis[strcspn(meds[*n].dosis, "\n")] = 0;

    printf("Frecuencia (en horas): ");
    scanf("%d", &meds[*n].frecuencia_horas);

    printf("Hora de la proxima toma (ej: 8): ");
    scanf("%d", &meds[*n].hora);

    printf("Minutos (ej: 30): ");
    scanf("%d", &meds[*n].minutos);

    printf("AM o PM: ");
    scanf("%s", meds[*n].am_pm);

    (*n)++;
    guardarMedicamentos(meds, *n);
    printf("Medicamento registrado correctamente.\n");
}
```

Solicita al usuario los datos de un nuevo medicamento, los almacena en el arreglo y los guarda en el archivo.

4.

```
93
94      for (int i = 0; i < n; i++) {
95          if (stricmp(nombre, meds[i].nombre)) {
96              int totalHoras = meds[i].hora;
97              if ((strcmp(meds[i].am_pm, "PM") == 0 || strcmp(meds[i].am_pm, "pm") == 0) && totalHoras != 12)
98                  totalHoras += 12;
99              if ((strcmp(meds[i].am_pm, "AM") == 0 || strcmp(meds[i].am_pm, "am") == 0) && totalHoras == 12)
100                 totalHoras = 0;
101
102             totalHoras += meds[i].frecuencia_horas;
103             totalHoras %= 24;
104
```

Permite actualizar la próxima hora de toma al sumar la frecuencia en horas. Convierte a formato 24 horas para hacer el cálculo.

5.

```
31  void guardarMedicamentos(Medicamento meds[], int n) {
32      FILE *f = fopen("medicamentos.txt", "w");
33      for (int i = 0; i < n; i++) {
34          fprintf(f, "%s;%s;%d;%d;%d;%s\n",
35                  meds[i].nombre,
36                  meds[i].dosis,
37                  meds[i].frecuencia_horas,
38                  meds[i].hora,
39                  meds[i].minutos,
40                  meds[i].am_pm);
41      }
42      fclose(f);
```

Escribe todos los medicamentos en el archivo medicamentos.txt para que estén disponibles al reiniciar el programa.

7.4 Imágenes de la ejecución de cada sección del programa explicadas

- Carga y escritura de medicamentos

```c
void cargarMedicamentos(Medicamento meds[], int *n) {
    FILE *f = fopen("medicamentos.txt", "r");
    if (f == NULL) return;

    while (fscanf(f, "%49[^;];%29[^;];%d;%d;%d;%2s\n",
                  meds[*n].nombre,
                  meds[*n].dosis,
                  &meds[*n].frecuencia_horas,
                  &meds[*n].hora,
                  &meds[*n].minutos,
                  meds[*n].am_pm) == 6) {
        (*n)++;
    }
    fclose(f);
}
```

Se leen/escriben medicamentos desde un archivo medicamentos.txt usando fscanf y fprintf.

- Actualización de próxima toma

```c
            totalHoras += meds[i].frecuencia_horas;
            totalHoras %= 24;

            if (totalHoras >= 12) strcpy(meds[i].am_pm, "PM");
            else strcpy(meds[i].am_pm, "AM");

            if (totalHoras == 0) meds[i].hora = 12;
            else if (totalHoras > 12) meds[i].hora = totalHoras - 12;
            else meds[i].hora = totalHoras;

            guardarMedicamentos(meds, n);
            printf("Proxima toma actualizada correctamente.\n");
            return;
        }
    }
```

Se convierte la hora al formato 24h para sumar correctamente la frecuencia, y luego se vuelve al formato 12h con AM/PM.

8. Ejecución del programa

Caso	Entrada	Salida esperada	Resultado
1	Paracetamol, 6h, 8:00 AM	Próxima toma 2:00 PM	Correcto
2	Ibuprofeno, 8h, 9:30 PM	Próxima toma 5:30 AM	Correcto
3	Nombre incorrecto al marcar como tomado	"Medicamento no encontrado"	Correcto
4	Visualización con varios medicamentos	Todos listados con datos correctos	Correcto

9. Conclusiones y Recomendaciones

- Conclusiones

  El desarrollo de este sistema de gestión de medicamentos no solo representa una práctica valiosa en el ámbito de la programación, sino que también responde a una necesidad social concreta, que es apoyar a personas que requieren un control constante sobre su medicación.

  Se logró desarrollar un sistema funcional para gestionar medicamentos con estructuras y archivos en C, además la modularidad del código facilita su lectura y mantenimiento, lo cual permite al sistema registrar medicamentos, mostrarlos y actualizar su próxima toma correctamente.

  Este proyecto final representó una experiencia enriquecedora y gratificante para nuestro equipo, ya que nos permitió aplicar de manera práctica todos los conocimientos adquiridos a lo largo del semestre en la materia de Programación I, desde la planificación, el diseño de estructuras, la manipulación de archivos, hasta la modularización del código, cada componente del sistema fue una oportunidad para reforzar nuestras habilidades y comprobar cuánto hemos avanzado.

- **Recomendaciones**
  - Validar mejor la entrada de AM/PM para prevenir errores como am o a.m.
  - Añadir opción para eliminar medicamentos.
  - Crear una versión con interfaz gráfica o conectividad móvil para usuarios reales.