Universidad Mariano Gálvez de Guatemala

Centro Universitario Antigua Guatemala

Facultad de Ingeniería

Algoritmos

Manual de Técnico del Sistema de Control de Pacientes

Antigua Guatemala

Grupo 2

Manual Técnico: Sistema de Control de Pacientes

Nombre del Proyecto: Sistema de Control de Pacientes

Lenguaje de Programación: C++

Archivos involucrados:

- base_datos.txt: Archivo de almacenamiento persistente de los datos de los pacientes.
- reporte_pacientes.txt: Archivo generado para reportar los pacientes registrados.

Descripción del Sistema

Este sistema permite gestionar los datos de pacientes en un hospital o clínica. El sistema está diseñado para almacenar, modificar, buscar, eliminar y generar reportes de pacientes de manera eficiente, además de mantener los datos en un archivo para su persistencia. La estructura del sistema es modular y cada función tiene un propósito específico.

Bibliotecas Utilizadas

1. <iostream>

Propósito: Manejar la entrada y salida estándar.
Funciones Principales: cout para salida en consola y cin para entrada desde teclado.

2. <vector>

Propósito: Proveer la estructura de datos de vectores dinámicos.
Uso: Almacenar listas de datos como IDs, nombres, edades, etc., permitiendo redimensionamiento automático.

3. <string>

Propósito: Manipular cadenas de caracteres.
Uso: Gestionar datos como nombres, direcciones, diagnósticos.

4. <fstream>

Propósito: Manejar operaciones de archivo.
Funciones Principales: fstream para escribir en archivos, permitiendo generar reportes en formato de texto.

5. <ctime>

Propósito: Proporcionar utilidades para trabajar con fechas y horas, tanto para obtener el tiempo actual como para manipularlo y formatearlo.

Funciones Principales: Formatear una fecha/hora en una cadena según un formato especificado.

Funciones Principales

1. **obtenerFechaActual()**:
    - **Descripción**: Obtiene la fecha actual del sistema en formato dd-mm-aaaa para usar en registros.
    - **Parámetros**: Ninguno.
    - **Valor de retorno**: string con la fecha actual.
2. **cargarDatos()**:
    - **Descripción**: Carga los datos de los pacientes desde el archivo base_datos.txt.
    - **Parámetros**:
        - vector<string>& ids: Referencia al vector que almacena los IDs de los pacientes.
        - vector<string>& nombres: Referencia al vector de nombres de pacientes.
        - vector<int>& edades: Referencia al vector de edades.
        - vector<string>& generos: Referencia al vector de géneros.
        - vector<string>& direcciones: Referencia al vector de direcciones.
        - vector<string>& telefonos: Referencia al vector de teléfonos.
        - vector<string>& fechasIngreso: Referencia al vector de fechas de ingreso.
        - vector<string>& diagnosticos: Referencia al vector de diagnósticos.
    - **Valor de retorno**: Ninguno.
3. **guardarDatos()**:
    - **Descripción**: Guarda los datos de los pacientes en el archivo base_datos.txt.
    - **Parámetros**: Igual que en cargarDatos().

- **Valor de retorno**: Ninguno.

4. **ingresarPaciente()**:
    - **Descripción**: Permite ingresar los datos de un nuevo paciente y agregarlos a las estructuras en memoria.
    - **Validaciones**: Asegura que el ID del paciente sea único, que todos los campos sean válidos, y que los datos no estén vacíos.
    - **Parámetros**: Igual que en cargarDatos().
    - **Valor de retorno**: Ninguno.
5. **modificarPaciente()**:
    - **Descripción**: Modifica los datos de un paciente existente, identificado por su ID.
    - **Parámetros**: Igual que en cargarDatos().
    - **Valor de retorno**: Ninguno.
6. **eliminarPaciente()**:
    - **Descripción**: Elimina los datos de un paciente, identificado por su ID.
    - **Parámetros**: Igual que en cargarDatos().
    - **Valor de retorno**: Ninguno.
7. **reporteGeneral()**:
    - **Descripción**: Muestra en pantalla el reporte de todos los pacientes con todos sus detalles.
    - **Parámetros**: Igual que en cargarDatos().
    - **Valor de retorno**: Ninguno.
8. **generarArchivo()**:
    - **Descripción**: Genera un archivo reporte_pacientes.txt con el reporte detallado de los pacientes.
    - **Parámetros**: Igual que en cargarDatos().
    - **Valor de retorno**: Ninguno.
9. **buscarPacientePorID()**:
    - **Descripción**: Busca un paciente en los datos almacenados utilizando su ID.

- **Parámetros**: Igual que en cargarDatos().
- **Valor de retorno**: Ninguno.

10. **mostrarMenu()**:
    - **Descripción**: Muestra las opciones del menú principal.
    - **Parámetros**: Ninguno.
    - **Valor de retorno**: Ninguno.

Flujo del Programa

1. **Inicio del Programa**:
    - El programa carga los datos almacenados en el archivo base_datos.txt en los vectores en memoria.
    - El menú principal es mostrado, esperando que el usuario seleccione una opción.
2. **Opciones del Menú**:
    - **Ingreso de Paciente**: Se ingresa un nuevo paciente con validaciones sobre los campos.
    - **Modificación de Datos del Paciente**: Permite modificar la información de un paciente existente.
    - **Reporte General**: Muestra los datos de todos los pacientes registrados.
    - **Buscar Paciente por ID**: Permite buscar un paciente por su ID.
    - **Generar Archivo**: Crea un archivo con el reporte de todos los pacientes.
    - **Eliminar Paciente**: Elimina los datos de un paciente seleccionado por ID.
    - **Salir**: Guarda los datos en el archivo base_datos.txt y finaliza el programa.

Recomendaciones

- **Respaldo del archivo**: Se recomienda hacer un respaldo del archivo base_datos.txt para evitar la pérdida de datos en caso de corrupción del archivo.
- **Validación de entradas**: Mantener las validaciones activas para evitar errores de entrada que puedan dañar la integridad de los datos.

- **Ampliación del sistema**: Este sistema puede ser fácilmente ampliado añadiendo nuevas funcionalidades como búsqueda por otros criterios (nombre, fecha de ingreso, etc.).

Posibles Mejoras Futuras

- **Persistencia de Datos**: Implementar la lectura de archivos para cargar datos existentes al iniciar el programa, y no solo la escritura.
- **Mejorar Validaciones**: Agregar validaciones más robustas para campos como fechas y números de teléfono, asegurando formatos correctos.
- **Interfaz Gráfica de Usuario (GUI**): Desarrollar una interfaz gráfica para mejorar la experiencia del usuario, haciéndola más intuitiva.
- **Uso de Clases y Objetos:** Refactorizar el código para usar programación orientada a objetos, creando una clase Paciente que encapsule los datos y métodos relacionados.
- **Base de Datos**: Integrar una base de datos para manejar grandes volúmenes de datos de pacientes y realizar búsquedas más eficientes.
- **Manejo de Errores:** Implementar manejo de excepciones para gestionar errores inesperados de manera más robusta.

Decisiones de Diseño

- **Uso de Vectores:** Se eligieron vectores dinámicos para almacenar datos de pacientes, permitiendo un fácil manejo y crecimiento de las listas.
- **Separación de Funciones:** Cada operación (ingresar, modificar, eliminar, etc.) se encapsula en una función específica, mejorando la claridad y mantenibilidad del código.
- **Validación de Datos:** Se implementaron validaciones básicas para asegurar que los datos ingresados sean correctos y completos, evitando errores comunes.
- **Interfaz de Texto Sencilla:** El menú y las interacciones se diseñaron para ser fáciles de entender y usar, sin necesidad de una interfaz gráfica compleja.