Universidad Mariano Gálvez de Guatemala

Centro Universitario Antigua Guatemala

Facultad de Ingeniería

Algoritmos

Manual Técnico del Sistema de Control de Pacientes

Antigua Guatemala

Grupo 2

Introducción

Este documento describe el funcionamiento de un sistema de control de pacientes. El programa permite gestionar datos de pacientes, incluyendo ingreso, modificación, eliminación, reporte y búsqueda de pacientes.

Estructura del Código

El código está escrito en C++ y utiliza bibliotecas estándar como <iostream>, <vector>, <string>, y <fstream>. Las funciones principales se detallan a continuación.

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.

Funciones Principales

1. ingresarPaciente

- **Propósito**: Agregar un nuevo paciente al sistema.
- **Parámetros**: Vectores de ids, nombres, edades, géneros, direcciones, teléfonos, fechasIngreso, diagnosticos.
- **Proceso**:

- Solicita al usuario los datos del paciente.
- Valida que el ID sea único.
- Valida que la información obligatoria no esté vacía.
- Agrega el paciente a los vectores.

2. modificarPaciente

- **Propósito**: Modificar la información de un paciente existente.
- **Parámetros**: Mismos vectores que ingresarPaciente.
- **Proceso**:
  - Solicita el ID del paciente a modificar.
  - Si el paciente existe, permite modificar sus datos.

3. eliminarPaciente

- **Propósito**: Eliminar un paciente del sistema.
- **Parámetros**: Mismos vectores que ingresarPaciente.
- **Proceso**:
  - Solicita el ID del paciente a eliminar.
  - Elimina al paciente si existe.

4. reporteGeneral

- **Propósito**: Mostrar un reporte de todos los pacientes.
- **Parámetros**: Mismos vectores que ingresarPaciente.
- **Proceso**:
  - Recorre los vectores y muestra los datos de cada paciente.

5. generarArchivo

- **Propósito**: Generar un archivo de texto con el reporte de pacientes.
- **Parámetros**: Mismos vectores que ingresarPaciente.
- **Proceso**:
  - Crea un archivo reporte_pacientes.txt.
  - Escribe los datos de cada paciente en el archivo.

6. buscarPacientePorID

- **Propósito**: Buscar y mostrar un paciente por su ID.
- **Parámetros**: Mismos vectores que ingresarPaciente.
- **Proceso**:
    - Solicita el ID del paciente.
    - Si existe, muestra su información.

7. mostrarMenu

- **Propósito**: Mostrar el menú principal del sistema.
- **Proceso**:
    - Presenta las opciones disponibles al usuario.

Función Principal: main

- **Lógica del Ciclo**:
    - Muestra el menú.
    - Solicita una opción al usuario.
    - Ejecuta la función correspondiente a la opción seleccionada.
    - Repite hasta seleccionar la opción de salida.

Uso del Sistema

1. **Ingreso de Paciente**: Permite agregar un nuevo paciente al sistema.
2. **Modificación de Datos**: Modifica los datos de un paciente existente.
3. **Eliminar Paciente**: Elimina un paciente del sistema.
4. **Reporte General**: Muestra todos los pacientes registrados.
5. **Generar Archivo**: Crea un archivo con el reporte de pacientes.
6. **Buscar por ID**: Busca un paciente utilizando su ID.
7. **Salir**: Termina la ejecución del programa.

Consideraciones Finales

- **Validaciones**: Se realizan validaciones para asegurar que los datos ingresados sean correctos y completos.
- **Archivos**: Los reportes se generan en un archivo de texto para facilitar su revisión.

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.

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.