Manual Técnico

Sistema de Gestión de Aeropuerto
Práctica – Estructuras de Datos
Universidad de San Carlos de Guatemala
Escuela de Ciencias y Sistemas
Jonathan Gabriel López Reyes
Carné. 202404730

2. Requisitos del Sistema

2.1. Sistema Operativo

- Windows 10 o superior (recomendado)
- Compatible con cualquier distribución de Windows que soporte MSYS2

2.2. Herramientas y Dependencias

- **Compilador**: GCC (incluido en MSYS2 MinGW-w64)
- **Entorno de desarrollo**: MSYS2 (distribución MinGW-w64)
- **Biblioteca de JSON**: nlohmann/json (solo cabecera, archivo json.hpp)
- **Herramienta de visualización**: Graphviz (versión mínima 2.40+)
- **Editor de código (opcional)**: Visual Studio Code, Code::Blocks, etc.

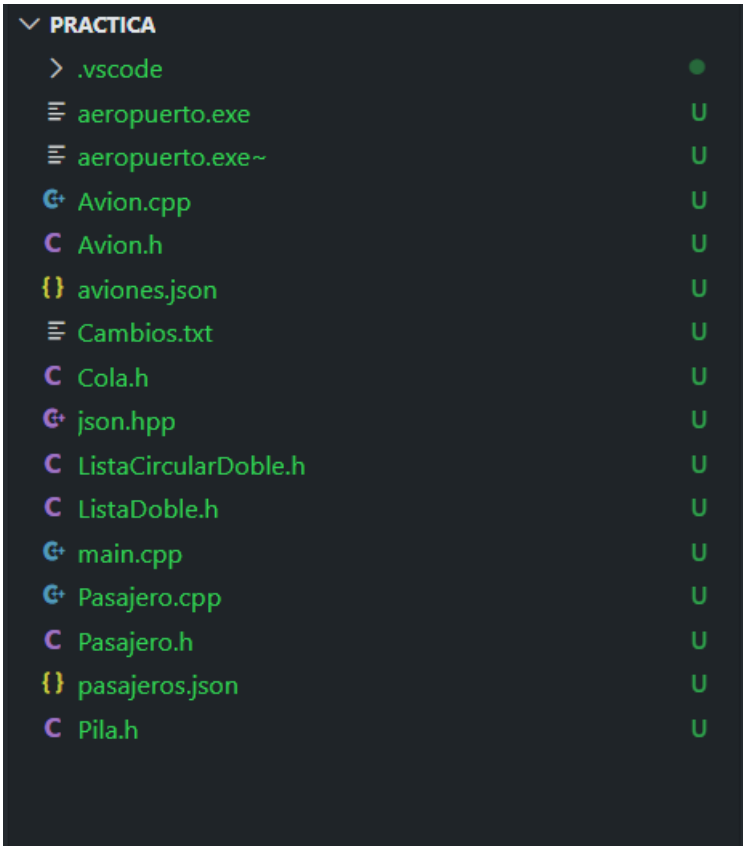
3. Instalación y Configuración

3.1. Instalación de MSYS2

3.2. Instalación de Graphviz

3.3. Estructura del Proyecto

El proyecto debe organizarse de la siguiente manera:



4. Compilación y Ejecución

4.1. Compilación

Abrir la terminal **MSYS2 MinGW-w64**, navegar a la carpeta del proyecto y ejecutar:

g++ -std=c++17 -I. main.cpp Avion.cpp Pasajero.cpp -o aeropuerto.exe

4.2. Ejecución

Ejecutar el programa con:

./aeropuerto.exe

4.3. Uso del Menú

El programa presenta un menú con las siguientes opciones:

1. **Carga de aviones**: Lee aviones.json y los distribuye en dos listas circulares dobles.
2. **Carga de pasajeros**: Lee pasajeros.json y los encola.
3. **Carga de movimientos**: Procesa Cambios.txt (registro de pasajeros y mantenimiento de aviones).
4. **Consultar pasajero**: Busca un pasajero por número de pasaporte en la lista doble.
5. **Visualizar reportes**: Genera archivos .dot y .png en la carpeta reportes/ usando Graphviz.
6. **Salir**: Finaliza la aplicación.

5. Estructuras de Datos Implementadas

Todas las estructuras fueron implementadas manualmente con **memoria dinámica y punteros**:

Estructura	Uso en el sistema
Lista circular doble	Almacena aviones en estado "Disponible" y "Mantenimiento"
Cola	Simula la ventanilla de registro de pasajeros
Pila	Almacena la cantidad de equipaje facturado (LIFO)
Lista enlazada doble	Guarda pasajeros registrados, ordenados por vuelo y asiento

6. Archivos de Entrada

- **aviones.json**: Contiene la información de los aviones (registro, modelo, estado, etc.).

- **pasajeros.json**: Contiene la información de los pasajeros (nombre, vuelo, equipaje, etc.).
- **Cambios.txt**: Contiene comandos para:
    - IngresoEquipajes;: Procesa un pasajero desde la cola.
    - MantenimientoAviones,Ingreso/Slida,<registro>;: Mueve aviones entre listas.

7. Reportes Generados

Al seleccionar la opción **5**, el sistema genera los siguientes reportes en la carpeta reportes/:

1. aviones_disponibles.png
2. aviones_mantenimiento.png
3. cola_registro.png
4. pila_equipaje.png
5. pasajeros_registrados.png

Cada reporte refleja el estado actual de la estructura de datos correspondiente.

8. Consideraciones Finales

- El proyecto **no utiliza librerías externas** para estructuras de datos (solo nlohmann/json para lectura de archivos).
- Todo el manejo de memoria es **responsabilidad del programador** (sin fugas ni accesos inválidos).
- El sistema está diseñado para ser **eficiente, seguro y modular**.