

# MANUAL TÉCNICO

- Estos son los mínimos requisitos del sistema que utiliza Dev-C++:
  - Microsoft Windows 95, 98, NT 4, 2000, XP
  - 8 MB de RAM con un archivo de intercambio grande
  - Procesador compatible Intel a 100 Mhz
  - 30 MB de espacio libre en el disco duro
- Estos son los requisitos recomendados de Dev-C++:
  - Microsoft Windows 2000, XP
  - 32 MB de RAM
  - Procesador compatible Intel a 400
  - 200 MB de espacio libre en el disco duro
- Estructura del proyecto:

```
~~~
202100081_EDD_PRACTICA
|— Airplane
|— Aviones(1).json
|— aviones_disponibles.dot
|— aviones_disponibles.png
|— aviones_mantenimiento.dot
|— aviones_mantenimiento.png
|— aviones.json
|— CircularDoubleList.h
|— CircularList.h
|— DoubleLinkedList.h
|— json.hpp
|— main.cpp
|— main.exe
|— movimientos.txt
|— Node.h
|— Pasajeros(1).json
|— pasajeros_cola.dot
|— pasajeros_cola.png
|— pasajeros_lista.dot
|— pasajeros_lista.png
|— pasajeros_pila.dot
|— pasajeros_pila.png
|— Passenger.h
|— Queue.h
|— README.md
|— Stack
~~~
```

## Dependencias Utilizadas

1. `#include <iostream>`: Proporciona facilidades de entrada y salida estándar en C++, como `std::cout` para salida en la consola y `std::cin` para entrada desde la consola.
2. `#include <fstream>`: Ofrece clases y funciones para la manipulación de archivos, incluyendo `std::ifstream` para leer archivos y `std::ofstream` para escribir archivos.
3. `#include <string>`: Define la clase `std::string` y funciones asociadas para manejar cadenas de caracteres de manera segura y eficiente.
4. `#include <limits>`: Proporciona características que definen los límites de los tipos de datos primitivos, como el valor máximo y mínimo que pueden almacenar.
5. `#include "json.hpp"`: Incluye la biblioteca JSON para C++ (generalmente `nlohmann::json`), que facilita la manipulación y el análisis de datos en formato JSON.

`json.hpp` es una biblioteca para C++ proporciona una manera fácil y eficiente de manipular datos en formato JSON. Esta biblioteca permite:

- Crear y modificar objetos JSON: Puedes construir objetos JSON, añadir y eliminar elementos de forma sencilla.
- Analizar y serializar: Facilita la conversión de cadenas JSON en estructuras de datos C++ y viceversa.
- Compatibilidad: Funciona bien con las STL (Standard Template Library) y soporta una amplia variedad de tipos de datos de C++.
- Simplicidad y eficiencia: Diseñada para ser fácil de usar y con un buen rendimiento.

Para usar y instalar la biblioteca podemos dirigirnos a [json](#) y buscar el archivo: `json.hpp`, únicamente podemos descargarlo y copiarlo/pegarlo a la carpeta donde queremos ejecutar.

6. `#include <filesystem>`: Introduce funciones y clases para la manipulación de sistemas de archivos, permitiendo operaciones como la creación, eliminación y consulta de archivos y directorios.

7. `#include <vector>`: Proporciona la plantilla de clase `std::vector`, una estructura de datos dinámica que permite almacenar y gestionar una colección de elementos de manera eficiente.

8. Graphviz: es una herramienta de software de código abierto para la visualización de grafos, muy utilizada para generar gráficos a partir de descripciones de grafos en un lenguaje de script simple.

Graphviz debe ser instalada al igual que `json.hpp` solo que en este caso podemos seguir los pasos de la siguiente página [graphviz](#)

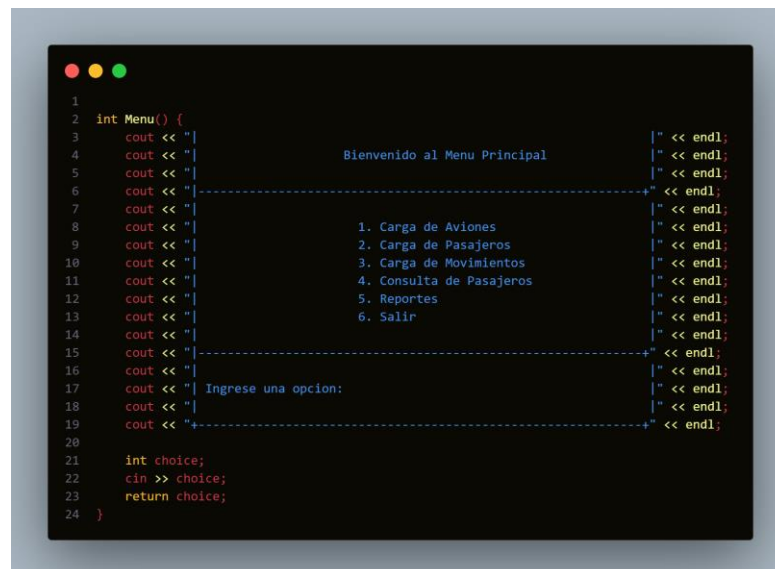
### Compilación y ejecución del código

Para compilar el programa, se puede usar `g++` o cualquier otro compilador de C++ que se tenga. El siguiente comando compila el programa utilizando `g++`:

```
cls && c++ .\main.cpp -o .\main && .\main.exe /ó/ g++ main.cpp -o main && main
```

Ejemplo del Código:

### Menú

A screenshot of a C++ code editor with a dark background and syntax highlighting. The code defines a function named 'Menu()' which displays a main menu. The menu has a title 'Bienvenido al Menu Principal' followed by a list of six options: '1. Carga de Aviones', '2. Carga de Pasajeros', '3. Carga de Movimientos', '4. Consulta de Pasajeros', '5. Reportes', and '6. Salir'. The options are separated by dashed lines. After the menu, the user is prompted to 'Ingrese una opción:'. The function then reads the user's input into a variable 'choice' and returns it. The code is as follows:

```
1
2 int Menu() {
3     cout << "
4     cout << "
5     cout << "
6     cout << "
7     cout << "
8     cout << "
9     cout << "
10    cout << "
11    cout << "
12    cout << "
13    cout << "
14    cout << "
15    cout << "
16    cout << "
17    cout << " Ingrese una opción:
18    cout << "
19    cout << "
20
21    int choice;
22    cin >> choice;
23    return choice;
24 }
```

## Función:

```
1
2 void readJsonAvion(const string& filepath) {
3     // Open the JSON file
4     ifstream file(filepath);
5     if (!file.is_open()) {
6         cout << "Could not open the file." << endl;
7         return;
8     }
9
10    try {
11        // Parseo del archivo JSON
12        json jsonData;
13        file >> jsonData;
14
15        // Acceso a los datos
16        for (const auto& item : jsonData) {
17            cout << "Vuelo: " << item.value("vuelo", "N/A") << endl;
18            cout << "Numero de Registro: " << item.value("numero_de_registro", "N/A") << endl;
19            cout << "Modelo: " << item.value("modelo", "N/A") << endl;
20            cout << "Fabricante: " << item.value("fabricante", "N/A") << endl;
21            cout << "Año de Fabricacion: " << item.value("ano_fabricacion", 0) << endl;
22            cout << "Capacidad: " << item.value("capacidad", 0) << endl;
23            cout << "Peso Max Despegue: " << item.value("peso_max_despegue", 0) << endl;
24            cout << "Aerolina: " << item.value("aerolinea", "N/A") << endl;
25            cout << "Estado: " << item.value("estado", "N/A") << endl;
26            cout << "-----" << endl;
27            Airplane *avion = new Airplane(item.value("vuelo", "N/A"),
28                                           item.value("numero_de_registro", "N/A"),
29                                           item.value("modelo", "N/A"),
30                                           item.value("fabricante", "N/A"),
31                                           item.value("ano_fabricacion", 0),
32                                           item.value("capacidad", 0),
33                                           item.value("peso_max_despegue", 0),
34                                           item.value("aerolinea", "N/A"),
35                                           item.value("estado", "N/A"));
36            ListaAvionesDisponibles.insert(*avion);
37            ListaAvionesMantenimiento.insert(*avion);
38        }
39    } catch (const json::exception& e) {
40        cout << "Error parsing the JSON file: " << e.what() << endl;
41    }
42
43    file.close(); // Se cierra el archivo después de leerlo
44 }
45 }
```