



## Resumen General del Programa

Este programa es un **buscador de aviones**. Carga una base de datos desde un archivo `aviones.txt`, permite al usuario seleccionar **criterios de búsqueda** (tipo, motor, ala, capacidad mínima, alcance mínimo, etc.), y luego **filtra e imprime** los aviones que cumplan con todos esos criterios.

---



## Estructura de Datos: Avion

```
typedef struct {
    char nombre[100];
    int capacidad;
    int alcance;
    float precio;
    char tipo[20];
    char motor[20];
    float precioHora;
    char ala[20];
} Avion;
```

Cada avión contiene:

- **nombre:** modelo del avión.
  - **capacidad:** número de pasajeros.
  - **alcance:** cuántos km puede volar.
  - **precio:** costo del avión en millones de USD.
  - **tipo:** Comercial, Privado, Militar.
  - **motor:** Turbofan, Hélice, Reactivo.
  - **precioHora:** cuánto cuesta volarlo por hora.
  - **ala:** tipo de ala (Fija, Delta, etc.).
- 



```
int cargarAviones(...)
```

Lee los datos de aviones desde el archivo de texto `aviones.txt`.


## Detalles:

- Abre el archivo en modo lectura.
- Usa `fscanf` para leer líneas con formato delimitado por comas.
- Guarda los datos en el arreglo `aviones[]` hasta alcanzar el máximo permitido (`MAX_AVIONES = 150`).

```
fscanf(archivo, " %99[^,],%i,%i,%f,%19[^,],%19[^,],%f,%19[^\n]\n", ...);
```

Esta línea **extrae campos** separados por comas:

- `%[^,]` lee hasta la siguiente coma (ej. una cadena).
  - `%i` para enteros.
  - `%f` para flotantes.
  - El prefijo numérico (como `99`) evita desbordamientos de búfer.
- 

 `void seleccionarOpcion(...)`

Muestra un menú con varias opciones (`tipos, motores, etc.`) y **pide al usuario que seleccione una**.

Internamente:

- Valida que el número ingresado esté dentro del rango.
  - Devuelve el índice seleccionado (desde 0) mediante un puntero.
- 

 `void pedirParametrosBusqueda(...)`

Pide al usuario que ingrese los filtros numéricos:

- Capacidad mínima.
- Alcance mínimo.
- Precio máximo.
- Precio por hora máximo.

Estos datos se usarán para filtrar los aviones en la búsqueda.

---

 `int cumpleCriterios(...)`

Función **clave**: verifica si un avión cumple con todos los filtros ingresados.


Evalúa:

- Si el tipo, motor o ala coinciden (o se ignoran si son `NULL`).
- Si la capacidad, alcance, precio y precio por hora están dentro de los límites permitidos.

```
return
    (tipo == NULL || strcmp(a.tipo, tipo) == 0) &&
```


...

---

 `void imprimirAvion(Avion a)`

Muestra los datos de un avión en una línea, con formato legible.

---

 `int SeleccionInvalida(...)`

Verifica combinaciones **inválidas** entre tipo de motor y tipo de ala.

Ejemplos de reglas:

- Motor Hélice no puede tener ala Delta ni Variable.
- Motor Reactivo no puede tener ala Tandem.

Esto fuerza combinaciones realistas en la aeronáutica.

---

 `main()`

## 1. Carga los datos

```
int total = cargarAviones(aviones, "aviones.txt");
```

## 2. Muestra mensaje de inicio

```
printf("=== Buscador de Aviones ===\n");
```

## 3. Pide selección de filtros de texto

Usa `seleccionarOpcion(...)` para elegir tipo, motor, y ala.

```
do {  
    ...  
} while (SeleccionInvalida(...));
```

→ Repite si la combinación no es válida.

## 4. Pide filtros numéricos

```
pedirParametrosBusqueda(&capacidadMin, &alcanceMin, &precioMax, &precioHoraMax);
```

## 5. Determina qué filtros están activos

Convierte índices seleccionados a NULL o a la cadena correspondiente:

```
const char *tipoFiltro = (tipo_index == 0) ? NULL : tipos[tipo_index];
```

## 6. Busca e imprime resultados

```
for (int i = 0; i < total; i++) {  
    if (cumpleCriterios(...)) {  
        imprimirAvion(aviones[i]);  
        encontrados++;  
    }  
}
```

Si encontrados == 0, imprime mensaje de no coincidencias.

---



## Formato esperado del archivo `aviones.txt`

Cada línea debe tener este formato:

Nombre, Capacidad, Alcance, Precio, Tipo, Motor, PrecioHora, Ala

Ejemplo:

Boeing 747, 416, 13845, 240.0, Comercial, Turbofan, 26000.0, Fija  
F-16 Falcon, 1, 4220, 18.8, Militar, Reactivo, 14000.0, Delta

---



## Resumen Final

Este programa:

- Carga una base de datos de aviones.
- Permite al usuario buscar según criterios técnicos y económicos.
- Valida combinaciones lógicas (motor + ala).
- Imprime solo los aviones que coincidan.