The Airline Reservation System is a C program designed to manage flight and passenger information. It allows for adding, editing, and deleting flight records, as well as enabling passengers to reserve seats and view flight schedules. The program utilizes file handling to store and retrieve data, ensuring persistence across sessions.

**Program Features:**

The program offers the following functionalities:

* **Reserve Seat**: Allows users to reserve seats on available flights.
* **User Ticket**: (Planned feature, not fully implemented in provided code)
* **Flights Schedule**: Displays details of a specific flight based on its flight number.
* **Display Passenger**: (Planned feature, not fully implemented in provided code)
* **Flight Details**: Provides authorized personnel access to manage flight records.
* **Add Flight**: Adds new flight information to the system.
* **Edit Flight**: Modifies existing flight records.
* **Delete Flight**: Removes flight records from the system.
* **Flight Leave and Arrive**: Allows for updating departure and arrival times of flights.
* **Back To Menu**: Navigates back to the main or previous menu.

**Program Structure and Pipelines:**

The program is structured using several C functions and two primary structures: Flights and Passenger.

**1. Data Structures:**

* **struct Flights**:
  + FlightNo (char array): Unique identifier for the flight.
  + FlightName (char array): Name of the airline or flight.
  + From (char array): Departure location.
  + Destination (char array): Arrival destination.
  + Departure (char array): Departure time.
  + Arrival (char array): Arrival time.
  + Price (char array/int): Seat price. (Note: tempProject.c uses int Price, while ARS.c and ARSYSTEM.c use char Price[10]).
  + *Future Enhancement*: A Total Seats value will be added to this struct to manage seat availability during reservations.
* **struct Passenger**:
  + FlightNo (char array): (Only in tempProject.c) The flight number for the reserved seat.
  + ID (int): (Only in tempProject.c) Passenger's ID.
  + PassengerID (int/char array): (Different types in different files) Unique identifier for the passenger.
  + Name (char array): Passenger's name.
  + Seats (int): Number of seats reserved by the passenger.

**2. File Handling:**

* **Flights.txt**: This file is used to store flight data in a comma-separated format.
* **filereadFlights()**: This function reads flight data from Flights.txt into the F array (array of Flights structs). It tokenizes each line based on commas to populate the Flights struct members.
* **filewriteFlights()**: This function writes the current flight data from the F array back into Flights.txt.

**3. Core Functions and Their Interactions:**

* **main()**:
  + This is the entry point of the program.
  + It presents the main menu to the user with options like "Reserve Seat," "Flights Schedule," "Passenger Records," and "Flight Details".
  + A

switch statement is used to call the appropriate function based on user input.

* + system("cls") (or clrscr()) is used to clear the screen for a cleaner user interface.
* **FlightDetails()**:
  + This function serves as an administrative section, accessible only after entering a correct PIN code (e.g.,

9078).

* + It displays a sub-menu with options for managing flight records: "Add New Flight," "Edit Flight Record," "Delete Flight," "Flight Arrival and Departure," and "Back to Main Menu".
  + A switch statement handles navigation to these specific flight management functions.
* **AddFlight()**:
  + Prompts the user to enter details for a new flight (Flight No, Flight Name, Departure Location, Destination, Departure Time, Arrival Time, Seat Price).
  + Includes input validation loops (goto statements) to ensure valid character inputs for string fields in tempProject.c.
  + After input, the new flight data is added to the F array.
  + The filewriteFlights() function is called to save the updated flight data to Flights.txt.
  + Asks the user if they want to add another flight or return to the main menu.
* **FlightsTable()**: (Only in tempProject.c)
  + Prints a formatted table of all currently loaded flights, displaying Flight No, Flight Name, Departure Location, Destination, Departure Time, Arrival Time, and Seat Price.
* **FlightNoSearch(char num[10])**:
  + Takes a flight number as input.
  + Searches the F array for a matching flight number.
  + Returns the index of the found flight or -1 if not found.
  + filereadFlights() is called within this function in ARS.c to ensure the flight data is up-to-date before searching.
* **EditFlight()**:
  + Asks the user for the flight number they wish to edit.
  + Uses FlightNoSearch() to find the flight.
  + If found, it displays the current details of the flight.
  + Prompts the user to enter new details for each field.
  + Updates the F array with the modified data.
  + Calls filewriteFlights() to save changes to the file.
  + Asks the user if they want to edit another flight or return to the main menu.
* **ArrivalDeparture()**:
  + Allows authorized personnel to update the departure and arrival times of a flight.
  + Asks for the flight number, searches for it, and then prompts for the new departure and arrival times.
  + Calls filewriteFlights() to persist the changes (implied, but not explicitly called in the provided ARSYSTEM.c and ARS.c versions).
* **DeleteFlight()**: (Partially implemented/commented out in ARSYSTEM.c and ARS.c, fully implemented in tempProject.c)
  + Takes a flight number as input.
  + Finds the flight using FlightNoSearch().
  + Shifts subsequent flight records in the F array to fill the gap created by the deleted flight.
  + The intention is to call

filewriteFlights() after deletion to update the file.

* **ReserveSeats()**: (Only in tempProject.c, conceptual in others)
  + Prompts for departure and destination locations.
  + Calls SearchFlight() to display available flights matching the criteria.
  + Asks the user to choose a flight and specify the number of seats.
  + *Future Enhancement*: Display ticket information after reservation.
  + *Future Enhancement*: Implement a check against Total Seats to prevent over-reservation.
* **SearchFlight(char dep[20], char des[20])**: (Only in tempProject.c)
  + Searches for flights based on given departure and destination.
  + Prints the details of matching flights.
* **FlightSchedule()**:
  + Asks the user for a flight number.
  + Uses

FlightNoSearch() to retrieve and display the details of that specific flight (Flight No, Name, Boarding Location, Destination, Departure, Arrival, Price).

* + Asks the user if they want to check another flight schedule.
* **DisplayPassengers()/PassengerRecords()**: (Planned feature, largely commented out or not fully implemented in provided code)
  + Expected to take passenger name and ID as input.
  + Intended to display all current, future, and previous flight details and reserved seats for that passenger.
  + Will likely involve reading from a separate "Passengers.txt" file (as hinted by filereadPassengers() in ARSYSTEM.c).

**4. Key Design Principles:**

* **Modularity**: The program is broken down into smaller, manageable functions, each responsible for a specific task.
* **Data Structures**: Use of struct for organizing related data (Flights and Passengers).
* **File I/O**: Persistent storage of data using text files.
* **User Interface**: Basic command-line interface with clear menus and prompts.
* **Security (Partial)**: A PIN system is included for accessing sensitive "Flight Details" (administrative) functions.
* **Realism**: Acknowledges real-world scenarios like flight delays (hence the "Flight Arrival and Departure" function).

**5. Pipeline Summary:**

1. **Initialization**: main() starts, potentially loading initial flight data using filereadFlights().
2. **Main Menu Display**: The user is presented with the main options.
3. **User Choice**:
   * **Reserve Seat**: User provides departure/destination, sees available flights, selects a flight and seats.
   * **Flights Schedule**: User provides flight number, sees flight details.
   * **Flight Details (Admin)**: User enters PIN. If authorized, they access the flight management sub-menu.
     + **Add Flight**: User inputs new flight data, which is stored and written to Flights.txt.
     + **Edit Flight**: User selects a flight, modifies its details, and changes are written to Flights.txt.
     + **Delete Flight**: User selects a flight, it's removed, and changes are written to Flights.txt.
     + **Arrival/Departure**: User updates times for a flight.
4. **Data Persistence**: All modifications to flight data are intended to be saved to Flights.txt via filewriteFlights().
5. **Exit**: The program can be quit from the main menu.

**Future Considerations/Incomplete Features (based on provided code and documentation):**

* **User Ticket and Display Passenger**: These features are mentioned in the AR System.docx but are largely absent or commented out in the C code. They would require more robust passenger data handling and potentially a separate file for passenger records.
* **Total Seats/Availability**: The concept of "total seats on the flight" and checking against it during reservation is highlighted but not fully implemented in the provided code.
* **Input Validation (Robustness)**: While tempProject.c includes some goto-based input validation, a more robust and user-friendly validation system would improve the program.
* **DeleteFlight() Implementation**: This function is incomplete or commented out in some versions, requiring full implementation with proper file writing.
* **ReserveSeat() Passenger Details**: The ReserveSeats() function in tempProject.c only takes FlightNo and ID, without passenger name, which is present in the Passenger struct.
* **Nested Structures**: The idea of "nested structure technique to connect flight details to passenger details" is mentioned but not explicitly demonstrated in the provided code. This could imply a more complex relationship where passenger records directly link to specific flight instances.

Sources