

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | Airport Management System | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Hamza Asif F2022376135M. Usman Shahid F2022376068Jasim Abrar F2022376138 |  | | | |
|  | | | | 15-01-2024—Data Structures and Algorithms—Muhammad Amjad Khan |  | | | |
|  | | |  | | |  | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  |  | | |  |
|  | Overview | | | | | | |  |
|  |  | | |  |  | | |  |
|  |  | |  | | |  | |  |
|  |  |  | The Airline Reservation System is a simple C++ program that simulates basic functionalities of an airline reservation system. It includes classes to represent airports, flights, passengers, baggage, and provides a command-line interface for users to interact with the system. The program allows users to add airports, book tickets, display passengers, check reservations, and more | | |  |  |  |
|  | | | | |
|  |  |  |  |
|  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | Components | | |  |
|  |  |  |  |  |

## Classes

### **1. Airport**

* Represents an airport with a name and airport code.
* Derived classes: InternationalAirport and DomesticAirport.
* Additional attributes for international airports: country.
* Additional attributes for domestic airports: region.

### **2. Flight**

* Represents a flight with a flight number and destination.

### **3. Passenger**

* Represents a passenger with a name, passport number, and an optional special service.
* Special services can be added to passengers.

### **4**. **Baggage**

* Represents baggage with weight and description.

### **5. BoardingQueue**

* Simulates a boarding queue for airports.
* Allows enqueueing and dequeuing airports.
* Keeps track of airports in a queue.

### **6. LinkedList**

* Represents a linked list to manage passengers Data booking tickets online managing the online system data of the user.
* Supports basic operations like insertion, deletion, and display.

## Usage

The program provides a command-line interface with the following options:

1. **Add Airport**
   * Allows users to add international or domestic airports to the boarding queue.
2. **Remove Airport**
   * Removes an airport from the boarding queue.



1. **Book Ticket**
   * Books a ticket for a passenger, allowing the user to input passenger details and optional special services.

****

1. **Display Passengers**
   * Displays the list of passengers.
2. **Check Reservations**
   * Checks and displays existing reservations.
3. **Cancel Reservation**
   * Cancels a reservation by specifying the passenger's passport number.



1. **Book Ticket Online**
   * Books a ticket for a passenger using an online method, similar to the manual booking option.



**8.Check Airport List**

* + Checks and displays the list of airports in the boarding queue.

**9.Exit**

* + Exits the program.

## DSA Concepts

1. **Array Operations:**
   * The program utilizes various array operations such as copying strings using strcpy, comparing strings using strcmp, and obtaining the length of a string using strlen.
2. **Stack:**
   * The BoardingQueue class uses a stack-based approach to manage airports. The stack array is used to store instances of the Airport class, and operations like enqueue and dequeue simulate stack operations.
3. **Linked List:**
   * The LinkedList class is implemented as a linked list to manage passengers. The Node structure is used to create nodes containing passenger data, and operations like insertion, deletion, and display are performed on this linked list.

### **DSA Concepts Applied:**

1. **Array-Based Data Structures:**
   * Arrays are used to implement data structures like stacks (stack array) and queues (queue array).
   * Arrays are used to store and manipulate strings, representing various attributes of objects in the program.
2. **String Operations:**
   * String manipulation using array-based operations like copying and comparing strings.
3. **Dynamic Array Usage:**
   * The stack array in the BoardingQueue class is dynamically managed to simulate a stack. The array expands as airports are enqueued.
4. **Array Traversal:**
   * Traversing arrays to display information and perform operations, e.g., displaying the list of passengers stored in the linked list.

### **Algorithms:**

1. **Enqueue and Dequeue Operations:**
   * The BoardingQueue class utilizes enqueue and dequeue operations to simulate adding and removing airports from the boarding queue. These operations are fundamental to various data structures, and in this case, they are implemented using a stack.
2. **Linear Search in Linked List:**
   * The dele\_end function in the LinkedList class performs a linear search to find the last node in the linked list. This is done to efficiently delete the last node when canceling a reservation.
3. **Passenger Search and Deletion:**
   * When canceling a reservation, the program performs a linear search in the linked list of passengers based on the passport number. This involves traversing the linked list to find the node to be deleted.

### DSA Concepts Applied:

1. **Time Complexity Analysis:**
   * The time complexity of various operations (enqueue, dequeue, insertion, deletion, search) can be analyzed. For example, linear search in the linked list has a time complexity of O(n), where n is the number of nodes.
2. **Stack Operations:**
   * Understanding and implementing stack operations for managing airports in the boarding queue.
3. **Linked List Operations:**
   * Implementing basic linked list operations, such as insertion, deletion, and traversal, to manage the list of passengers efficiently.
4. **Search Algorithms:**
   * Utilizing linear search algorithms for finding specific elements in the linked list (e.g., searching for a passenger based on passport number).
5. **Memory Management:**
   * Dynamic memory management concepts are applied in the new and delete operations when creating and deleting nodes in the linked list.

By incorporating these DSA concepts, the program efficiently manages the data related to airports, flights, passengers, and baggage. Understanding the underlying data structures and algorithms helps in optimizing the code and making it more scalable and maintainable.

## Conclusion

The Airline Reservation System provides a basic framework for simulating airline reservation functionalities.