**Corso di Laboratorio di Algoritmi e Strutture Dati**

**REPORT PROGETTO II: COMPAGNIA AEREA**

# Anselmi Gianmarco – N86001394

# Nikolov Aleks Nikolaev – N86003002

**Lama Lorenzo – N86002845**

**Mappa grafo**

Immagine che contiene testo, mappa

Descrizione generata automaticamente

# Implementazione

# La traccia prevede la realizzazione di un programma per la gestione delle prenotazioni di voli.

Prima di essere eseguito, il programma popola l’aeroporto con 20 destinazioni e tratte arbitrarie, riempendo il grafo *voli*. Viene inoltre popolato l’abr contenente gli utenti con 3 admin e 3 clienti e a quest’ultimi vengono inizializzate alcune prenotazioni di voli.

**Per motivi di semplicità, si assume che la disponibilità dei voli non sia vincolata a date nè al numero dei posti disponibili sul volo stesso.**

Successivamente, un ciclo do-while stampa un menu finchè non si inserisce un input d’uscita (4) con successiva richiesta di conferma di chiusura del programma. Il menu è gestito da uno switch case.

Il programma offre le seguenti funzionalità:

* **Effettua nuova registrazione ,** permette la registrazione di un nuovo cliente nel sistema. Si inserisce un nome utente e una password rispettando il fatto che queste ultime non possono contenere spazi e la loro lunghezza non deve superare i 20 caratteri. **Gli input sono case-sensitive**. E’ possibile registrarsi come clienti ma non come admin (si veda scelta 3 per la visualizzazione delle credenziali per un admin)
* **Termina programma,** permette di terminare il programma
* **Visualizza utenti,** permette di visualizzare gli utenti registrati
* **Esegui accesso**, implementata dalla funzione ***eseguiAccesso*** e ***gestisciUtente***.

***eseguiAccesso*** restituisce 0 se il login fallisce, 1 se le credenziali appartengono a un semplice cliente, 2 se invece appartengono a un admin

***gestisciUtente*** mostrerà la schermata menù del cliente o dell’admin in base all’esisto della funzione eseguiAccesso

Da qui distingueremo i 2 casi d’accesso differentemente:

1. ***Menù cliente:*** da qui un ciclo do while stampa un menu finchè non si inserisce un input di uscita(4). Il menù è gestito da uno switch case e offre queste funzionalità:

* **Effettua prenotazione** dalla quale è possibile :
* **Effettuare una prenotazione inserendo partenza e destinazione**

Si inserisce un aeroporto di partenza e uno di destinazione, dopo di che si effettua una breadth first search per controllare che l'aeroporto di destinazione sia raggiungibile da quello di partenza. Se la tratta esiste allora si chiederà di scegliere tra una tratta più economica e una più breve, che verrà trovata con l’applicazione dell’algoritmo di Dijkstra; verrà stampata la tratta desiderata e si chiederà se si vuole applicare uno sconto per l’acquisto. Dopo avere o non avere applicato uno sconto, si richiederà la conferma di prenotazione al cliente;In seguito alla conferma, la prenotazione verrà inserita nella lista delle prenotazioni all’interno del campo t\_utente del nodo utenteCorrente, le cui informazioni andranno inserite nei vari campi della struttura t\_prenotazione, prezzo, tempo e scali

* **Effettuare una prenotazione inserendo solo partenza**

Si inserisce un aeroporto di partenza, dopo di che si ha la possibilità di scegliere la destinazione più economica e destinazione più gettonata (tramite la visita nell’albero delle precedenti prenotazioni degli altri clienti). Per entrambe le opzioni si ha la possibilità di applicare uno sconto al prezzo del biglietto e successivamente di confermare l’acquisto

* **Mostrare i voli**

Mostra tutti i voli, stampati come liste di adiacenza

* **Tornare indietro al menù utente**

Torna indietro alla schermata del menù utente

* **Stampa prenotazioni attive**

Stampa tutte le prenotazioni effettuate dall’utente che ha effettuato l’accesso.

* **Gestisci punti e tickets** dalla quale è possibile:
* **Vedere i punti disponibili**

Viene stampato il numero dei punti disponibili per l’utente

* **Vedere tickets disponibili**

Da qui è possibile visionare i tickets disponibili (punti convertiti in buoni sconto)

* **Convertire punti in tickets**

Da qui è possibile convertire i punti posseduti dal cliente in tickets sconto per i voli. Le regole di conversione sono le seguenti:

1 – E’ possibile convertire punti in tickets a patto che il numero di punti sia compreso tra 50 e 250

2 – I tickets di sconto ottenibili vanno dal 5%, 10% fino al 25%

3 – 50 punti corrispondono a uno sconto del 5%, 100 punti del 10%.. fino a 250 che corrispondo a un ticket del 25%

4 – Inserire numeri diversi da 50, 100, 150, 200, 250 è inutile (con 80 punti non otterrete uno sconto dell’8% ma del 5% e vi saranno sottratti solo 50 punti) ma non errato

* **Torna indietro al menù cliente**

Torna indietro alla schermata del menù cliente

* **Eseguire il logout**

Esegue il logout dell’utente corrente e fa tornare al menù iniziale.

Questo per quanto riguarda il menù cliente; Vediamo adesso:

1. ***Menù admin:*** da qui un ciclo do while stampa un menu finchè non si inserisce un input di uscita (6). Il menù è gestito da uno switch case.

* ***Aggiungi una nuova destinazione:***

Si chiede di inserire il nome del nuovo aeroporto, che viene inserito nel grafo dei voli

* ***Aggiungi un nuovo volo:***

Permette di aggiungere una nuova tratta.

Le richieste da soddisfare sono: partenza, destinazione, prezzo e durata

* ***Rimuovi una destinazione:***

Rimuove una destinazione da un aeroporto

* ***Rimuovi un volo:***

Rimuove un volo dal grafo dei voli

* ***Stampa tutti i voli:***

Stampa il grafo dei voli come una lista di adiacenze

* ***Logout***

Torna indietro al menù utente

**Scelta delle strutture dati**

La struttura dati utilizzata per la registrazione dei voli è un grafo orientato, composto da una struct *t\_grf* in concomitanza con una struct *t\_arc* per gli adiacenti. Un vertice di tale grafo è dato da una struct contenente il *nome* dell’aeroporto, una variabile *popolarità* usata nella ricerca della destinazione più gettonata, un puntatore a *t\_grf* per raggiungere gli altri vertici e un puntatore a *t\_arc* per raggiungere gli adiacenti di un vertice; infine una variabile intera *camminoMinimo* e un puntatore a t\_grf chiamato *prev,* usati nell’algoritmo di Dijkstra per tenere traccia del percorso minimo fino a un certo vertice. Inoltre nella struct *t\_arc* sono presenti i pesi *prezzo* e *durata* utili per la memorizzazione di informazione per i voli.

Il grafo è stato implementato come una lista di liste semplicemente concatenate, per permettere la facile e dinamica aggiunta/rimozione di aeroporti, e la comoda applicazione di funzioni ricorsive per le varie operazioni.

La struttura dati utilizzata per la registrazione degli utenti è un albero di ricerca binario, dove ogni nodo presenta un campo utente di tipo t\_utente, contenente a sua volta i campi nomeUtente, password, l’intero admin che se a 1 rappresenta che l’utente è un admin, se a 0 cliente, i campi punti e tickets(vettore di interi), che rappresentano rispettivamente i punti e i tickets di sconto a disposizione dell’utente, e una lista di tipo t\_lista\_P di prenotazioni contenente il campo prenotazione di tipo t\_prenotazione. Quest’ultimo presenta i campi prezzo e tempo, rispettivamente prezzo totale del viaggio e tempo totale di volo, scali, una lista di tipo t\_lista\_S di stringhe contenente i vari scali del viaggio, e i due campi stringhe partenza e destinazione. Per maggiori informazioni si rimanda ai file abr.h e liste.h.

La scelta dell’utilizzo di un abr come struttura contenitore degli utenti è dovuta alla disponibilità di una ricerca logaritmica su di essa, fondamentale ipotizzando che una compagnia abbia un discreto numero di utenti registrati.

La scelta dell’utilizzo di liste, sia per contenere le varie prenotazioni di ciascun utente sia per contenere i vari scali di ciascuna prenotazione, è ovviamente dovuta in primis alla necessità di una struttura dinamica per entrambe.

Inoltre, il numero di prenotazioni(di un utente) o scali(di una prenotazione) non può essere così elevato da mettere in imbarazzo una ricerca lineare, più che sufficiente con strutture con caradinalità bassa. Apprezziamo quindi il risparmio di memoria dovuto al singolo puntatore next rispetto ad altre strutture come abr o altre implementazioni di lista, specialmente considerando che quest’ultimo avviene per ciascuna prenotazione di ciascun singolo utente nel nostro sistema.

**Suddivisione del lavoro tra i componenti del gruppo**

Lama Lorenzo: strutture e funzioni per la gestione dell’abr degli utenti e delle liste di prenotazioni e scali, sistema di gestione e conversione di punti/tickets, funzioni per la gestione del main e gran parte delle funzioni per la gestione del cliente.

Nikolov Aleks Nikolaev: strutture e funzioni per la gestione del grafo dei voli, funzioni per le ricerche sui voli, prenotazione di volo da sola partenza

Anselmi Gianmarco: abbellimento interfaccia, testing, report, mappa grafo e menù admin

**Esempi di esecuzione**

**Registrazione ed accesso come cliente**

[*https://s7.gifyu.com/images/accesso6968ec02d94f0770.gif*](https://s7.gifyu.com/images/accesso6968ec02d94f0770.gif)

**Prenotazione volo con destinazione**

[*https://s7.gifyu.com/images/prenotazione.gif*](https://s7.gifyu.com/images/prenotazione.gif)

**Gestione dei ticket**

[*https://s7.gifyu.com/images/punti.gif*](https://s7.gifyu.com/images/punti.gif)

**Operazioni admin**

[*https://s7.gifyu.com/images/admin.gif*](https://s7.gifyu.com/images/admin.gif)