**PROGETTAZIONE ADT hash.c/.h**

* Size (intero)
* Table (puntatore ad un array di puntatori a iscritto)

**PROGETTAZIONE FUNZIONI**

newHashtable:

1. Alloco lo spazio per una variabili di tipo hashtable.
2. Controllo che la memoria sia stata allocata correttamente.
3. Assegno alla size della tabella creata, la size passata tra i parametri.
4. Inizializzo dinamicamente tutti gli elementi della tabella come puntatori a iscritto.
5. Ritorno la tabella hash.

insertHash:

1. Dichiaro una variabile intera idx, che mi rappresenta l’indice di posizione in cui andro ad inserire il mio iscritto.
2. Assegno a idx il valore di ritorno generato dalla funzione di hashing, hashfun, passandogli tra i parametri l’ID dell’iscritto e la size della tabella.
3. Imposta 'curr' e 'head' (puntatori a iscritto) al primo elemento della lista in questo indice.
4. Apro un ciclo per scorrere la lista, per capire se è gia presente un iscritto con lo stesso ID, se presente ritorna 0 e il duplicato viene scartato.
5. Se l’elemento non è presente, viene inserito in cima alla lista e ritorna 1.

hashDelete:

1. Dichiaro una variabile intera idx e ne calcolo il valore tramite la funzione hashfun, che prende come stringa l’IDCliente.
2. Imposto i tre puntatori al primo elemento della lista in questo indice.
3. Scorro la lista con un while.
4. Se si trova il nodo con IDCliente uguale a quello passato come parametro, viene rimosso, se in testa viene aggiornato il primo elemento della lista, se nella lista viene fatto scalare.
5. Se trovato il cliente viene restituito, toccherà a noi richiamare la funzione eliminaCliente per deallocare lo spazio occupato.
6. Se non viene trovato nulla viene restituito NULL.

destroyHastable:

1. Viene chiamato un for che intera da 0 alla dimensione della tabella.
2. Nel corpo del for viene richiamata la funzione deleteList, che prende la lista di iscritti della posizione i della tabella.
3. Termina il for.
4. Viene deallocata la tabella hash.

deleteList:

1. Controlla che l’iscritto passato come parametro non sia NULL (caso di terminazione)
2. Richiama se stessa, ma come parametro l’elemento successivo della lista.
3. Dealloca l’iscritto.

hashSearch:

1. Controllo che la tabella e l’ID sia validi (caso negativo ritorno NULL).
2. Calcolo la posizione dall’ID tramite l’hashFun.
3. Scorro la lista affinché non trovi un iscritto con lo stesso ID.
4. Quando la trovo ritorno l’iscritto stesso, altrimenti restituisco NULL.

ricercaGenerica:

1. Inizializzo una variabile intera trovato a 0.
2. Scorro tutta la tabella (non utilizzo hashfun in quanto non sto ricercando per chiave primaria).
3. Tramite un selettore, comando lo switch in base a che tipo di ricerca devo svolgere:

* Ricerca per nome 🡪 0
* Ricerca per cognome 🡪 1
* Ricerca per durata Abbonamento 🡪 2

1. Quando trovo l’elemento, stampo i dettagli del cliente tramite la funzione stampaCliente e imposto trovato a 1.
2. Ritorno trovato, per capire se l’elemento fosse presente o meno nella tabella.

stampaHash:

1. Controllo che la tabella non sia vuota.
2. Scorro l’intera tabella e stampo, quando li trova, stampa i dettagli del cliente tramite la funzione stampaCliente.

stampaMinimaHash:

1. Controllo che la tabella non sia vuota.
2. Scorro l’intera tabella e stampo, quando li trova, stampa i dettagli del cliente tramite la funzione stampaMinimaCliente.

scriviFileClienti:

1. Apro il file iscritti.txt in modalità scrittura, e controllo la corretta apertura del file.
2. Scorro la tabella tramite un for.
3. Tramite la funzione scriviCliente, scrivo su iscritti.txt i campi di iscritto.
4. Chiudo il file iscritto.txt.

hashFun:

1. Algoritmo di Hashing FNV-1a