**PROGETTAZIONE ADT test.c/.h**

**PROGETTAZIONE FUNZIONI**

mainTest:

1. Apro in lettura testSuite.txt e in scrittura testResult.txt
2. Controllo che l’apertura dei file sia avvenuta correttamente
3. Dichiaro una stringa che mi conterrà il nome del test che andrò ad effettuare e ne controllo l’allocazione.
4. Avvio un ciclo che termina quando il file viene letto completamente.
5. Ad ogni lettura assegno ad una variabile intera result il ritorno di runTest, che prende: il nome del test, il selettore, che mi indica quale test sto facendo, le tre strutture dati che contengono gli iscritti, i corsi e le prenotazioni.
6. Se result è uguale ad 1, scrivo nel file testResult il nome del test e Pass (TC1\_Prenotazioni: Pass), che me ne indica il successo; altrimenti il nome del file e Fail (TC1\_Prenotazioni: Fail), che ne rappresenta il fallimento.
7. Chiudo il file aperti all’inizio e libero la memoria del nome del test.

runTest:

1. Costruisco le tre stringhe che mi rappresentano il nome del file di input, di output e l’oracolo del test che eseguiremo:

* TC1\_Prenotazione\_Input.txt
* TC1\_Prenotazione\_Output.txt
* TC1\_Prenotazione\_Oracle.txt

1. Tramite il valore del selettore passato tra i parametri, uno switch sceglie che tipo di test devo eseguire:
   1. Verifica della corretta registrazione delle prenotazioni e dell'aggiornamento delle disponibilità.
   2. Test della gestione degli abbonamenti e della verifica della validità.
   3. Verifica che il report generato contenga informazioni corrette e complete sulle prenotazioni.
2. Ognuno di questi casi ritorna il valore della funzione chiamata, le funzioni sono:
   1. testCorrettaRegistrazionePrenotazione
   2. testValiditàAbbonamento
   3. testReport

testCorrettaRegistrazionePrenotazione:

1. Apro il file fileInput in lettura, e fileOutput in scrittura.
2. Controllo che vengano aperti correttamente.
3. Alloco spazio per due stringhe, che conterranno l’IDCorso e l’IDCliente, e controllo che l’allocazione sia andata a buon fine.
4. Creo una nuova lista di corsi, che mi servirà per controllare che il corso esista, successivamente mi servirà anche per incrementare e decrementare il numero di partecipanti al corso.
5. Avvio un ciclo che termina quando viene letto l’intero fileInput.
6. Ad ogni iterazione controllo che gli ID letti da file esistano:
   1. Caso Negativo: scrivo sul fileOutput “Cliente o Corso Inesistente”
   2. Caso Positivo:
7. Creo la nuova prenotazione con i dati letti dal file.
8. Controllo la corretta creazione della prenotazione.
9. Incremento il numero di partecipanti del corso a cui fa riferimento la prenotazione.
10. Scrivo sul fileOutput i due ID e il numero di partecipanti al corso.
11. Decremento il numero di partecipanti.
12. Chiudo i due file e dealloco lo spazio dedicato ai due ID.
13. Richiamo la funzione confrontaFile, per confrontare l’oracolo e l’output, se uguali ritorno 1, altrimenti 0.

testValiditàAbbonamento:

1. Apro il file fileInput in lettura, e fileOutput in scrittura.
2. Controllo che vengano aperti correttamente.
3. Alloco spazio per una stringa, che conterranno l’IDCliente, e controllo che l’allocazione sia andata a buon fine.
4. Creo una variabile iscritto i, che mi conterrà l’iscritto dell’ID letto da file.
5. Avvio un ciclo che termina quando viene letto l’intero fileInput.
6. Ad ogni iterazione assegno ad i, l’iscritto trovato dalla funzione hashSearch, che data la tabella hash e l’ID dell’iscritto, restituisce un puntatore ad esso.
7. Se i è uguale a NULL, scrivo sul fileOutput “Cliente inesistente”.
8. Se i è diverso da NULL, controllo che l’abbonamento sia valido, tramite la funzione controlloAbbonamento, se il ritorno della funzione è 0, scrivo “Abbonamento valido”, altrimenti “Abbonamento Scaduto”.
9. Chiudo i due file e dealloco lo spazio dedicato all’ID.
10. Richiamo la funzione confrontaFile, per confrontare l’oracolo e l’output, se uguali ritorno 1, altrimenti 0.

testReport:

1. Apro il file fileInput in lettura, e fileOutput in scrittura.
2. Controllo che vengano aperti correttamente.
3. Creo un array di stringhe contenenti tutti i mesi, mi servirà poi per determinare l’output del programma.
4. Assegno ad una variabile intera meseR, il valore del mese corrente.
5. Creo due liste, una per Corsi e una per Prenotazioni, e le carico tramite le funzioni caricaFileCorso e caricaFilePrenotazioni. (non uso le strutture già riempite dal programma, in quanto le modifiche che devo apportare per effettuare il test dovrammo poi essere cancellate, in questo modo non dovrò riaggiornare le strutture che vengono riempite all’inizio del main).
6. Alloco spazio per due stringhe, che conterranno l’IDCorso e l’IDCliente, e controllo che l’allocazione sia andata a buon fine.
7. Creo tre liste di Corsi:

* resultCorso: per controllare che il corso esista.
* corsiDelMese: lista contenente solo i corsi del mese corrente.
* Classifica: lista dei tre corsi più richiesti del mese.

1. Ad ogni iterazione controllo che gli ID letti da file esistano:
2. Caso Negativo: scrivo sul fileOutput “Cliente o Corso Inesistente”
3. Caso Positivo:
4. Creo la nuova prenotazione con i dati letti dal file.
5. Controllo la corretta creazione della prenotazione.
6. Aggiungo la prenotazione alla lista Prenotazioni già precedentemente riempita.
7. Incremento il numero di partecipanti del corso a cui fa riferimento la prenotazione.
8. Assegno a corsi del mese, il ritorno della ricercaMese.
9. Assegno a classifica, la lista proveniente dalla funzione lezioniInEvidenza, che restituisce una lista con i tre corsi più richiesti del mese corrente.
10. Stampo il mese corrente, il numero di prenotazioni del mese, ottenibile grazie alla size della lista prenotati, e i primi tre corsi del mese, tramite la funzione scriviLezioniInEvidenza, che prende la lista classifica e il puntatore a file del file sul cui scrivere.
11. Chiudo i file precedentemente aperti e dealloco le variabili allocate dinamicamente
12. Richiamo la funzione confrontaFile, per confrontare l’oracolo e l’output, se uguali ritorno 1, altrimenti 0.

confrontaFile:

1. Apro due file in lettura, con i nomi dei file presi dai parametri.
2. Controllo la corretta apertura dei file.
3. Alloco lo spazio per due stringhe, la prima stringa leggerà dal primo file, la seconda stringa dal secondo file. Ne controllo l’allocazione.
4. Pongo di base che i due file siano uguali, ponendo una variabile intera uguali=1.
5. Avvio un ciclo che termina quando entrambi i file sono terminati.
6. Ad ogni iterazione confronto le due stringhe tramite strcmp, se sono diversi cambio il valore di uguali, ponendolo a 0 ed esco dal ciclo.
7. Libero lo spazio occupato dalle due stringhe e chiudo i due file.
8. Ritorno il valore di uguali.