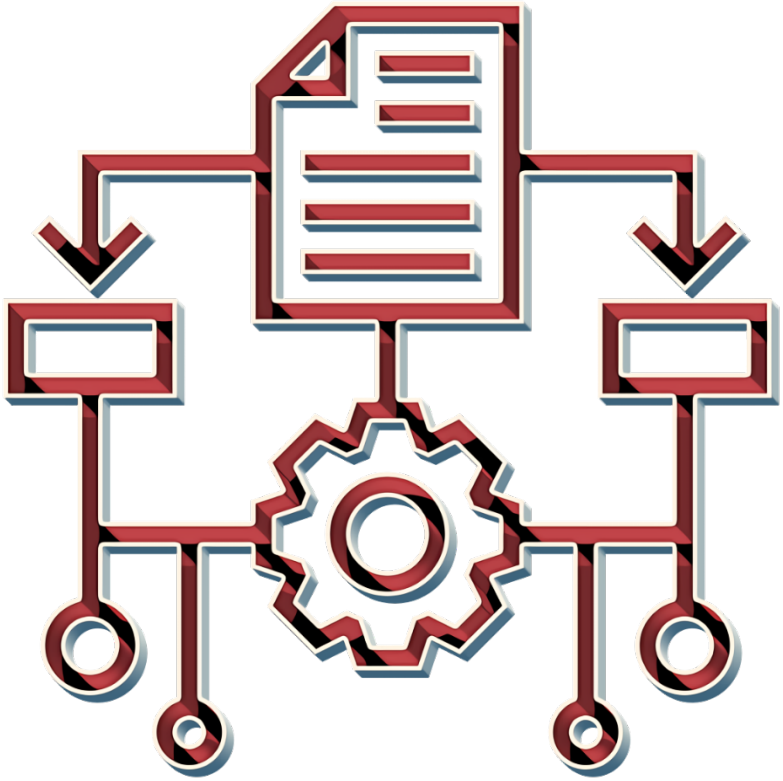
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Laboratorio Di Algoritmi e Strutture Dati

Progetto 1 a.a. 2019-2020

Programma per la gestione di una biblioteca per studenti



Di Giovanni Aldo *Matricola N86002319*

Approccio alla soluzione

Il problema è stato approcciato in maniera alquanto pratica, cercando la soluzione ottimale per tempistiche personali e complessità del problema stesso.

Si richiedeva un programma in grado di gestire le funzionalità basilari di una biblioteca di ateneo, ovvero il prestito dei volumi agli studenti e il recupero dei libri prestati agli stessi.

Ho pensato a molte possibilità, fra cui un "simulato" approccio a oggetti, con due strutture che contenessero rispettivamente libri e studenti e un'altra che contenesse invece le relazioni fra le precedenti due.

Tuttavia, valutando anche il poco tempo a mia disposizione a causa di impegni universitari precedenti, ho scelto una direzione più pratica ed immediata anche se tuttavia legata a quella accennata sopra.

Nella soluzione finale, le strutture sono rimaste due ma sensibilmente diverse.

La prima, chiamata "libro" contiene in realtà le informazioni che riguardano il prestito, cioè le relazioni fra le ipotetiche strutture di cui prima, rappresentazione dell'interazione fra i due attori, ovvero l'insieme degli studenti che ha accesso alla biblioteca e la biblioteca stessa. Sono quindi state agglomerate le informazioni delle due strutture dell'idea originaria, andando a convogliare nella suddetta struttura libro che mantiene il nome del libro (che è qui l'unione di Autore/i e Titolo del libro) e la matricola dello studente al quale il preciso volume sia stato eventualmente prestato oppure una matricola di default, indicante la disponibilità del volume. Il "preciso volume" indica l'effettiva e reale entità che stiamo andando a considerare, individuata univocamente, nel database, dalla coppia (Nome del Libro, Numero della Copia).

L'altra struttura è quella invece chiamata Prestito. Essa mantiene le informazioni riguardanti le richieste sospese, ovvero quelle richieste che non possono venire esaudite nel momento stesso in cui vengono presentate a causa dell'indisponibilità del volume richiesto da uno studente.

Sono quindi conservate, in ogni “istanza”, le informazioni riguardanti il libro richiesto e la matricola dell’utente che abbia fatto la richiesta.

Qualora due studenti abbiano fatto richiesta per lo stesso libro, l’ordine in cui le richieste siano state fatte sarà rispettato grazie ad un’opportuna progettazione del codice, in modo che il primo studente che abbia fatto richiesta sia il primo a vedersela esaudita, qualora questo sia possibile.

Il database accennato prima, in questa sede, si è deciso essere basato sull’uso di semplici file di testo. Nell’implementazione pratica, i file di testo sono due, uno per ognuna delle due strutture sopra presentate, in cui, ogni riga, contiene le informazioni riguardanti le rispettive strutture.

Affinché l’applicativo possa funzionare correttamente, la struttura dei due file dev’essere soggetta ad alcune regole o vincoli:

1) La matricola di uno studente non può essere uguale a quella di default, che è “xxxxxxxxx”, virgolette escluse.
2) Il nome di un libro (che come abbiamo visto è in realtà l’unione di Autore e Titolo) non può iniziare con “0”. Nell’improbabile caso che il nome di un autore inizi con 0 si dovrà agire opportunamente dal lato database, durante l’inserimento di un nuovo libro.
3) Ogni volume è indicato univocamente dalla coppia (Nome del Libro, Numero della Copia).
4) Volumi uguali sono memorizzati in posizioni contigue per facilitare il lavoro all’applicativo nell’ipotesi di un database grande quanto possa esserlo uno reale. Nel codice il controllo parte infatti dal nodo successivo a quello considerato ma, qualora questa regola non voglia essere rispettata, basterebbe farlo partire dalla testa della lista.
5) Devono essere inserite tutte le colonne di un’ipotetica visione tabellare del file (cioè devono esserci tutti i campi titolo, matricola etc), sebbene l’applicativo non vada comunque in crash in caso contrario.
6) I titoli e le matricole non possono contenere il carattere “|”.

All’apertura dell’applicativo saranno caricate in memoria, per velocità e semplicità di manipolazione dei dati, entrambe le strutture, in modo che le richieste sospese create durante le precedenti sessioni d’uso del programma non vengano perse, così da soddisfare correttamente gli studenti.

Il database dei libri sarà aggiornato alla fine di ogni sessione d’uso per avere un modello quanto più simile ad un applicativo reale.

La privacy è garantita per ogni utente, difatti la matricola associata a un libro non disponibile non è mai visibile.

Nel codice è tuttavia presente una funzione di stampa capace di fornire tutte le informazioni, lasciata per eventuali test ma comunque mai chiamata.

Avevo pensato di rendere l’inserimento del titolo case insensitive (magari usando semplicemente delle opportune funzioni strcasestr) ma ho poi deciso di evitare per irrobustire la “sicurezza” e non lasciare così tanta libertà all’utente.

Per lo stesso motivo la lunghezza della matricola è automaticamente troncata una volta raggiunti i 9 caratteri fissati.

Scelta delle strutture dati

Non ho individuato particolari motivi per usare strutture dati complesse quali alberi o grafi oppure semplici quali pile o code che avrebbero richiesto codice in più per operazioni essenziali ma immediatamente realizzabili sfruttando altre strutture. Avendo escluso strutture statiche quali gli array per permettere l’eventuale modifica al database iniziale, ho optato per una lista concatenata e una lista doppiamente concatenata.

La prima è stata scelta per la struttura Libro, dato che, considerando la natura piuttosto stabile del database, non ho osservato particolari motivi

per dover usare strutture più complicate e conservare altri dati in memoria (specialmente andando a considerare uno scenario reale con decine di migliaia di volumi).

La seconda invece è l'implementazione adottata per la struttura Presito. A differenza della precedente, qui le informazioni da conservare sono molto più "volatili" e ho scelto questa struttura per una maggiore velocità, semplicità e pulizia di codice, nel particolare caso della rimozione di un nodo contenente una richiesta esaudita. Il controllo di eventuali richieste sospese viene infatti eseguito ogniqualvolta un utente ceda il posto a un altro (e diviene quindi plausibile che, durante la sua sessione d'uso, egli abbia restituito un libro richiesto da un altro studente che fosse, nel frattempo, in coda).

Le liste saranno poi efficacemente deallocate all'uscita del programma e i campi destinati all'input dell'utente sono opportunamente controllati affinché non ci siano eventuali problemi in memoria di dati sovrascritti "per errore".

Esempio d'esecuzione

I titoli dei libri, nel database d'esempio, sono stati scelti per rendere l'esperienza di test più verosimile possibile. Risultano quindi sì scomodi da scrivere ma mi permetto di suggerire la copia con le combinazioni da tastiera del titolo dalla schermata d'esecuzione stessa, quando il programma ci chiede se l'utente desideri visualizzare l'elenco dei libri oppure no.

Elenco di input dell'utente separati da spazi, i titoli dei libri sono abbreviati per praticità e dopo il trattino troviamo il resoconto di quanto avvenuto.

Si avvii il programma:

uno 2 1 Marcellini - l'utente prova a restituire un libro che non ha

1 0 Marcellini - prende il libro, forse precedentemente si era sbagliato

2 0 Marcellini - ci ha ripensato e lo restituisce

1 1 Sipser - ne prende un altro, il Sipser

1 1 Sipser 0 - prova a prenderlo di nuovo e poi esce

N86002319 1 0 Sipser 0 - il nuovo utente prova a prendere un libro non disponibile e poi esce; la sua richiesta viene salvata

uno 1 1 Cormen – uno fa richiesta per il Cormen

2 0 Sipser – uno restituisce il Sipser

0 0 “premo invio” – uno esce e poi termina anche il programma

Adesso N86002319 ha ottenuto il Sipser e uno è in attesa del Cormen, com’è possibile osservare aprendo i rispettivi “database”.

Faccio ripartire il programma di nuovo:

tre 1 1 Cormen – l’utente tre fa richiesta per il Cormen

1 1 Mazzoldi 0 – prende il Mazzoldi e poi esce

N86002319 2 0 Cormen – N86002319 restituisce il Cormen

2 0 0 - l’utente fa per restituire un libro ma si accorge di aver sbagliato, ne voleva invece chiedere in prestito uno

1 Carlomagno 0 0 - fa quindi richiesta per il Carlomagno, esce e termina il programma.

Ora ho in attesa tre per il Cormen e N86002319 per il Carlomagno.

Tre ha adesso il Mazzoldi e uno invece il Cormen.