DATA STRUCTURES AND ITS APPLICATIONS

UE22CS252A

3rd Semester, Academic Year 2023

MINI PROJECT

LIBRARY MANAGEMENT SYSTEM

Team Members :

Aarav Raj - PES2UG22CS007

Aathil Nishad - PES2UG22CS011

Abhishek R Rathod - PES2UG22CS020

Akash G Gaonkar - PES2UG22CS043

ADT definition and Data Structures used :-

Data Structures used for implementation – Singly Linked List

1. book *createBook()

    · This function creates a node using the user defined data type, struct books.

    · Nodes are allotted memory dynamically. Each node consists of 6 fields, those are – id (int), title (char array), author (char array), available (char), readerid (int), *next (pointer of struct books type).

    · It takes input from the user for all these data members.

- It’s implemented using Singly Linked List data structure.

2. reader *createReader()

   - This function creates a node using the user defined data type, struct reader.

   - Nodes are allotted memory dynamically. Each node consists of 5 fields, those are – id (int), name (char array), due (char type), duebookid (int array), *next (pointer of struct reader type).

   - It takes input from the user for all these data members.

   - It’s implemented using Singly Linked List data structure.

3. void insertBook(book **head)

   - This is an insert function, it’s used to when a new book node is created, this is called upon. i.e. when a new book needs to be added.

   - It has one argument, a double pointer head of struct reader type.

   - Inside this function, an if-else is used to check for the first book entry, or adding to an already existing list.

   - It’s implemented using Singly Linked List data structure.

4. void searchBook(book *head, int searchId)

· This function is used to search a particular book in the library.

· It has 2 arguments, a pointer head of book type and another normal variable searchId of int type.

· It starts with the first node and compares it with searchId until a match is found. To continuously do this, a while loop is used.

· Data Structures :-

- *int searchId* : A variable which stores the Id of the book.

5. void insertReader(reader **head)

· This is an insert function, it's used to when a new reader node is created, this is called upon. i.e. when a new reader needs to be added.

· It has one argument, a double pointer head of struct reader type.

· Inside this function, an if-else is used to check for the first reader entry, or adding to an already existing list.

· It's implemented using Singly Linked List data structure.

6. void searchReader(reader *head, int searchId)

· This function searches for a reader with the specified Id (searchId) in the list of readers. If found, it prints the reader's information (Id, name, due status, and due book IDs). If not found, it displays a message indicating that the reader was not found.

· It has 2 arguments, head pointer of reader type and a searchId of int type.

· Data Structures :-

- *reader *head*: A pointer to the head of a linked list containing the readers.
- *int searchId* : A variable which stores the Id of the reader.

7. void issueBook(book *bookList, reader *readerList, int bookId, int readerId)

· This function handles the process of issuing a book to a reader in a library system. It involves checking the availability of the book, the due status of the reader, and assigning the book to the reader if all conditions are met.

· It has 4 arguments, a pointer of bookList and readerList of book and reader data type respectively, bookId and readerId of int type.

· Data Structures :-

- *book *bookList*: A pointer to the head of a linked list containing book information.
- reader *readerList: A pointer to the head of a linked list containing reader information.

- *bookId, readerId :* Variable which stores the Id of books and readers.

8. void submitBook(book *booklist, reader *readerList, int bookId)

    · This function manages the process of submitting a book to the library. It verifies the book's issued status, the reader who has the book, and updates the book's availability, reader information, and due book Ids accordingly.

    · It has 3 arguments, , a pointer of bookList and readerList of book and reader data type respectively and a variable bookId of int type.

    · Data Structures :-

    - *book *bookList*: A pointer to the head of a linked list containing book information.
    - *reader *readerList*: A pointer to the head of a linked list containing reader information.

9. void deleteBook(book **head, int bookId)

    · This function manages the removal of a book from the library list. It searches and removes the book with a specific ID from the book list. It updates the linked list by adjusting pointers and deallocates memory for the deleted book.

    · It has 2 arguments, a pointer to a pointer and a variable bookId of int type.

    · Data Structures :-

- *book **head :* A pointer to a pointer, allowing for the modification of the head of the linked list containing book information.

10. void deleteReader(book **head, int readerId)

    - This function manages the removal of a reader from the reader's list. It searches and removes the reader with a specific ID from the reader's list. It updates the linked list by adjusting pointers and deallocates memory for the deleted reader.

    - It has 2 arguments, a pointer to a pointer and a variable readerId of int type.

    - Data Structures :-

        - *reader **head* : A pointer to a pointer, allowing for the modification of the head of the linked list containing reader information.

11. void listBooks(book *head)

    - This function lists and displays the information of all books in the library system. It prints the details of each book in format, including the book's ID, title, author, availability, and the ID of the reader if the book is currently checked out.

    - It has an argument, which is a pointer of book type.

12. void listReaders(reader *head)

- This function lists and displays the information of all readers in the library system. It prints the details of each reader in format, including the reader's ID, name, Due status,, and the ID's of the books that are currently due.
- It has an argument, which is a pointer of reader type.
- It's implemented using Singly Linked List data structure.