UTM

UNIVERSITI TEKNOLOGI MALAYSIA

UNIVERSITI TEKNOLOGI MALAYSIA (UTM), CAMPUS SKUDAI

SECJ2013

DATA STRUCTURE AND ALGORITHM

ASSIGNMENT TITLE:

ASSIGNMENT 2

PREPARED BY (PAS):

NOOR ALIYAH BINTI AHMAD SAUFI (A22EC0234)

SERI NUR AYUNI BINTI SALIMI (A22EC0268)

PUTERI NURIN I'RDINA BINTI FADZIL (A22EC0261)

SECTION:

SECJ2013-04

LECTURER'S NAME:

DR LIZAWATI BINTI MI YUSUF

DATE OF SUBMISSION:

31st DECEMBER 2023

SOURCE CODE SUBMISSION LINK:

https://github.com/jjn7702/SECJ2013-DSA/tree/main/Submission/sec04/PAS/Assignment2

TABLE OF CONTENTS

Page

OBJECTIVES

We have outlined a few objectives for this task, which are crucial in order to provide a guideline that facilitates an efficient execution of the tasks. These said objectives are are as follows:

- To develop a system that is user-friendly and produce an output display that is neat and comprehensive.
- To develop the system using C++ programming language.
- To ensure that the developed system can run efficiently and effectively.
- To allow insertion and deletion of book information to/from the library book list using linked list.
- To develop a library management system that can conduct processes based on certain criteria.

SYNOPSIS

The Library Management System is designed to manage and view the library's books that have attributes of book identification, title, author's name, genre and year of publication. In Assignment 2, we implemented the linked list concept in our system to let users add new book information or delete an existing book. For both addition and deletion of books, users can either do at the beginning, middle or the end of the list. We also provide a function where users can find a book in the linked list based on their entered search key.

DESIGN

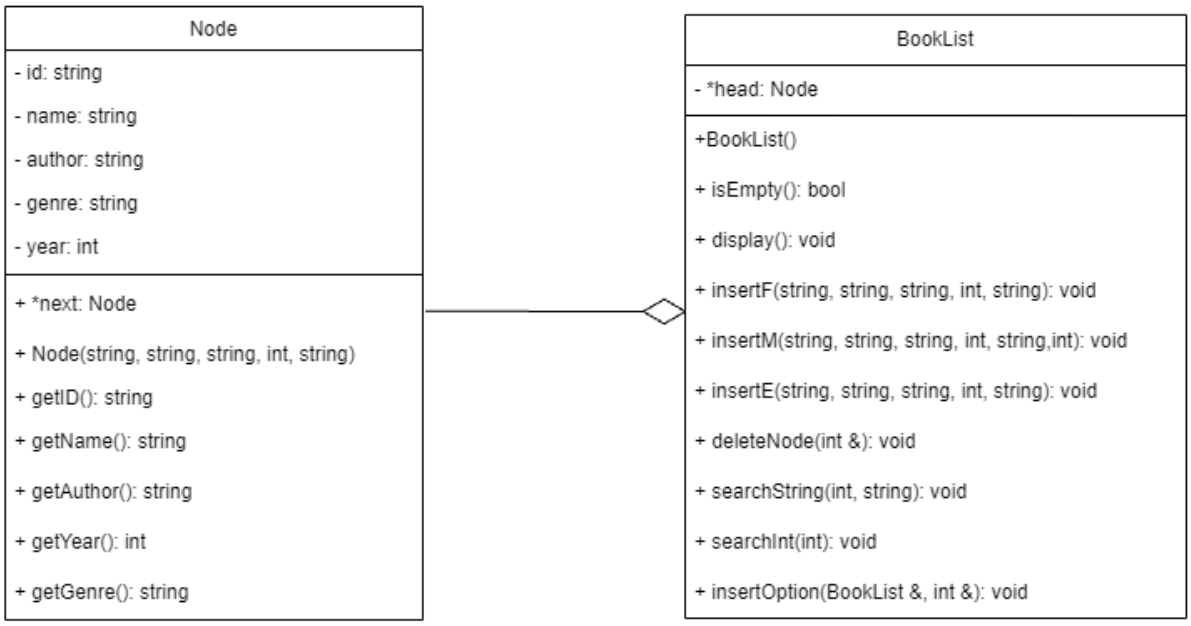

Figure 1: Overall Class Diagram of the Library Management System Program

The diagram above shows the overall class diagram of the library management system program. In the diagram, we implement two different classes which are Node and BookList. These two classes are joined with each other through an aggregation relationship due to the declaration of a Node pointer inside the BookList class.

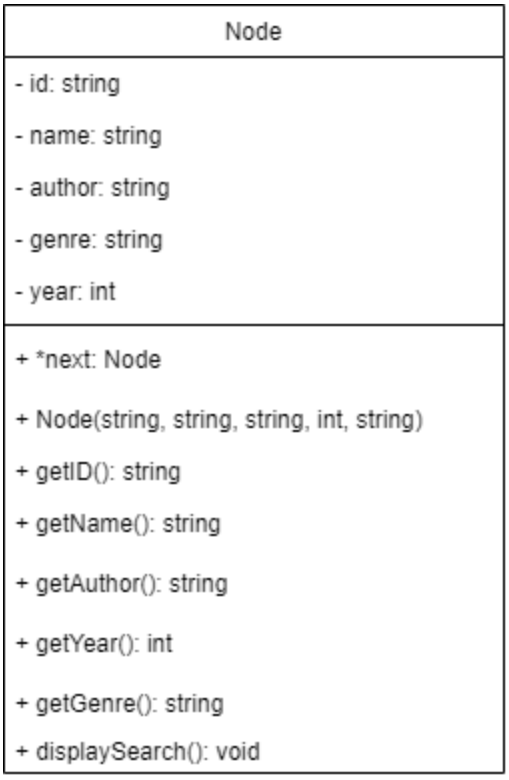

Figure 2: Class Design of Class Node

The diagram above illustrates the class design for class Node. This class stores private attributes which are id, name, author, genre, and year. This class also contains a Node object named next which is a pointer, it is used to point to the next address of the data in the list. In this class, we also implemented member functions such as default argument and the accessors of the private attributes.

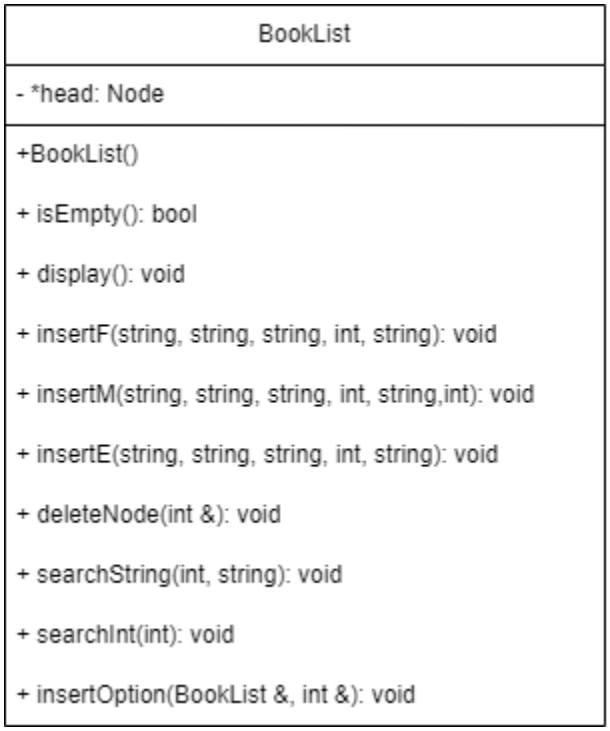

Figure 3: Class Design of Class BookList

The diagram above illustrates the class design for class BookList. This class stores a private Node object which is a pointer named head. In this class, we implemented various member functions inside this class. These said functions are such as isEmpty(), display(), insertF(), insertM(), insertE(), deleteNode(), searchString(), searchInt(), and insertOption().

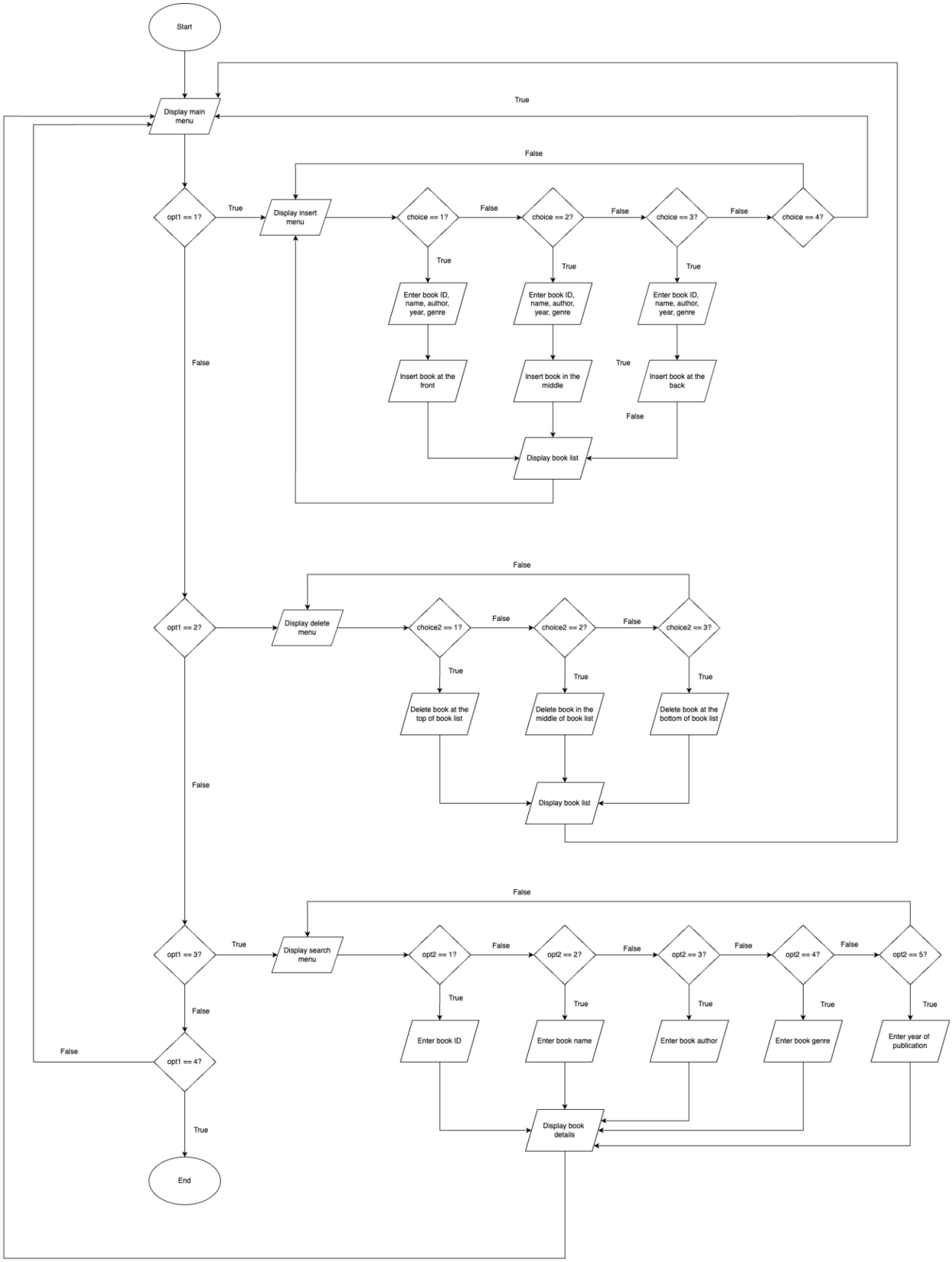

Figure 4: Flowchart of the Library Management System's Operation

The diagram above visualizes how the library management system operates from start to finish. The system goes through multiple layers of processes, such as inserting, deleting and finding a book information. As illustrated above, we also implemented multiple conditions in order to achieve its objectives and desired output. It is crucial for this system to carry out each of the operations and conditions in order to not only produce output correctly but also efficiently .

DESCRIPTION

1. **ADDING A NEW NODE**

For insertion of a new node in the library list, we define three different functions, insertF to insert at the beginning, insertM to insert in the middle and insertE to insert at the end. For all functions, we declare a new node using class Node and we check if the list is empty. If yes, the new node will be assigned as the head node.

In the insertF function, if the list is not empty, the new node next points to the head node and new node assigned as the head node.

Next, in the insertM, we also declare variable x representing the index and variable count to count the node. If the value of the temp next node points to non null and the count value is less than x minus 1, the temp next node is assigned as the temp node. The process continued until we found the index we wanted. The temp next will be the new node next and new node is assigned to temp next

Last, in the insertE function, the new node will be added and linked at the end of the list if the temp next is equal to null value.

2. **DELETE A NODE**

We declare one function in the libraryList to do the deletion of a node named deleteNode. This function takes one argument, int n, representing the number of nodes in the list. In this function, we have two objects of class Node named temp and delnode. Head is assigned to the temp node. We used a switch case to facilitate the process of deleting.

Case 1 is to delete at the beginning of the list or the book at the top of the displayed book list. If the list contains only one node, then the temp node is deleted and the null value is assigned to the head node while if the list contains many nodes, deletion will occur after we have assigned the next node as the new head node.

Case 2 is to delete at the end of the list and we define temp as head. If the available node

is only one, the temp node is deleted and null value is assigned to the head node; else if the list has many nodes, the temp node will go through nodes in the list until it finds a null value and the temp node is deleted.

Case 3 is to delete a node in the middle of the list based on the position entered by the user. We declare a variable named count and index to help us find the position of the node. We use the while loop with a condition where the temp next node cannot be a null value and the count must be less than index minus 1. Then the temp node is assigned to delnode and the temp next node is assigned to the temp. When the loop ends, the temp next node is assigned to delnode and delnode next is assigned to temp next. Lastly, the delnode will be deleted.

3. FIND A NODE

In this system, we implemented a searching feature to find information related to the book that matches with the key data entered by the user. We implemented 2 member functions in the class BookList for this feature which are searchString() that does the finding operation for string data type and searchInt() that does the finding operation for int data type.

In searchInt(), we use a while loop to compare the published year stored in the system with the key year entered by the user. When the years are matched, the function will call the displaySearch() function from class Node.

In searchString(), we used the same mechanism as in searchInt() but we included switch case to facilitate the process of finding data for string data type. Case 1 will operate when user input option 1 in the search option menu where the system will search for a book id that matches with the key id entered by the user. Secondly, Case 2 will operate when user input option 2 in the search option menu where the system will search for a book title that matches with the key title entered by the user. In the third case, the system will search for a book author that matches with the key author entered by the user and lastly, the system will search for a book genre that matches with the key genre entered by the user when user chooses option 4 in the search option menu.