

DA-IICT Club Manager

CAPSTONE PROJECT REPORT – IT205 | APRIL 07, 2024

STUDENT INFORMATION :

1. Ayush Mittal – 202301145
2. Sri Sadana Dharavath - 202301121
3. Zeel Parmar - 202301171
4. Sanya Vaishnavi – 202301117

Problem Statement – P6 :

You need to build a manager for all the DA-IICT clubs. The manager ensures that a club member can be looked up in minimum time. A member can either be a faculty or a student. One should be able to search by name, ID, specific club name, or club category (i.e., arts, science & technology, sports, culture). Note that the user of this manager may not be a DA-IICT-ian and, therefore, may not know the clubs' names.

Algorithm :

1. Define Club and Member Structures :

- Create club structures with details about the club's name, category, and members (students and faculty members).
- Create structures for members containing information such as name, ID, and club memberships.

2. Initialize Club Database :

- Create a database containing details on every DA-IICT club, such as names, groups, and membership.

3. Search Member Function (*searchMember*) :

- Searches for a member by name or ID using *membersByName* and *membersById*.
- Utilizes lowercase conversion for case-insensitive search.
- Time Complexity: $O(1)$ on average for ID search, $O(N)$ for name search where N is the number of members.
- Space Complexity: $O(1)$
- Display all clubs in that category along with their details.

4. Handling Non-DA-IICT Users :

- If the user is not familiar with DA-IICT clubs' names.
- Provide a list of club categories as options for the user to choose from.
- Once a category is selected, display all clubs in that category for further selection.
- Finally, display the details of the selected club or its members.

5. Optimize Search Time :

- Implement data structures like hash tables or binary search trees to optimize search time for faster lookup of members or clubs.
- Time Complexity: $O(N*M)$, where N is the number of members and M is the average number of clubs per member.

- Space Complexity: $O(N+M)$, where N is the number of members and M is the number of unique clubs.

6. Error Handling :

- Implement error handling to handle cases such as invalid input or non-existent clubs/members.

- Provide appropriate error messages to guide the user.

7. Main Function :

- Handles user interaction and input processing.

- Executes search functions based on user choices.

- Time Complexity: Depends on the chosen operation.

- Space Complexity: Depends on the data retrieved from the search functions.

8. Time complexity part by part

-Reading the CSV file and parsing each line: $O(n \cdot m)$

-Searching for a member (*searchMember* function): $O(1)$.

-Searching for a club by name (*searchClub* function): $O(K)$

- Searching for clubs by category (*searchClubByCategory* function): $O(1)$

- loading data: $O(n \cdot m)$
- search member (by name): $O(n)$
- search club (by name): $O(k)$
- search club by category: $O(1)(\text{category lookup}) + O(m)$ per club

Data Structures Used :

1) Unordered map :

Use: For storing relationships between values (Member or Club objects) and keys (such as member names, IDs, and club names).

Why Used: With an average time complexity of $O(1)$ for insertion, deletion, and search operations, unordered maps offer quick lookup speeds.

Space Complexity: An unordered map has a space complexity of $O(n)$, where n is the total number of elements that the map stores. This is due to the fact that the map's space requirements increase linearly with the quantity of key-value pairs it stores.

Justification: Because unordered maps use constant-time lookup operations, they are appropriate for efficient searching because they enable speedy retrieval of members and clubs by their names and IDs.

2) Vectors :

Use: Lists of clubs within each category are stored for the purpose of enabling category-specific searches.

Why Used: Vectors are useful in situations where sequential access to items is necessary because they provide constant-time access to elements by index.

Space Complexity: A vector has a space complexity of $O(n)$, where n is the number of components it contains. This is a result of vectors' dynamic memory allocation and resizing.

Justification: When looking for clubs inside particular categories, efficient iteration and access are made possible by storing clubs in vectors. Because there can be variations in the number of clubs in each category, vectors' dynamic scaling guarantees the best possible memory utilization.

3) Enums :

Useful for defining separate club types to improve code readability and organization.

Why Used: By mandating a limited number of values, enums improve code readability by offering a means to describe fixed sets of named constants.

Space Complexity: Enums usually don't significantly increase the program's space complexity. Typically, numerical values are used to represent them.

Justification: The code is less prone to errors and more self-explanatory when handling club categories thanks to the use of enums. Enumerations lessen the possibility of using wrong category names and aid in ensuring type safety.

4) Struct:

Use: To create unique data structures for clubs and members that combine related data into a single entity.

Why Used: Structures link relevant attributes together to make managing and manipulating member and club information easier.

Space Complexity: A structure's space complexity is influenced by the magnitude of each of its constituent variables. In this case, the size of the strings (name) and integer variables (ID) inside the structures dictates the space complexity.

Justification: The code becomes more modular and organized by establishing structures for clubs and members. It also improves maintainability and readability of the code by logically grouping attributes related to clubs and members.

Code Test Runs :

CASE 1 :

```
-----> Welcome to DA-IICT Club Community <-----
```

```
SEARCH OPTIONS :
```

1. Search by name
2. Search by ID
3. Search by club name
4. Search by club category
5. Exit

```
ENTER YOUR CHOICE : 1
```

```
Enter the Member Name : aRJUn
```

```
Member Name : Arjun
```

```
ID : 101
```

```
Clubs : Muse, PMMC, Dance
```

```
SEARCH OPTIONS :
```

1. Search by name
2. Search by ID
3. Search by club name
4. Search by club category
5. Exit

```
ENTER YOUR CHOICE : █
```

CASE 2 :

```
ENTER YOUR CHOICE : 2

Enter the ID : 169

Member name : Shaan
ID : 169
Clubs : PMMC

SEARCH OPTIONS :
1. Search by name
2. Search by ID
3. Search by club name
4. Search by club category
5. Exit

ENTER YOUR CHOICE : █
```

CASE 3 :

```
ENTER YOUR CHOICE : 3

Enter the Club Name : Cubing

Members of the Cubing club :
1) Rajesh      ID : 142
2) Veer        ID : 161
3) Pratham     ID : 181
4) Raghav      ID : 189
5) Anand       ID : 144
6) Vivan       ID : 199

SEARCH OPTIONS :
1. Search by name
2. Search by ID
3. Search by club name
4. Search by club category
5. Exit

ENTER YOUR CHOICE : █
```

CASE 4 :

ENTER YOUR CHOICE : 4

Enter the Category (Arts, Science, Sports, Culture) : scieNce

Clubs in the Category scieNce :

1] Programming :

- 1) Kashvi ID : 200
- 2) Sneha ID : 137
- 3) Jaya ID : 131
- 4) Aditya ID : 111
- 5) Meenakshi ID : 192
- 6) Armaan ID : 151
- 7) Aman ID : 165
- 8) Rahul ID : 103

2] Business :

- 1) Vikram ID : 107
- 2) Suresh ID : 150
- 3) Armaan ID : 151
- 4) Arnav ID : 179
- 5) Ashwin ID : 159
- 6) Pratham ID : 181
- 7) Megha ID : 162
- 8) Saanvi ID : 168
- 9) Ishita ID : 170
- 10) Isha ID : 198
- 11) Sanjay ID : 140

3] MSTC :

- 1) Pranav ID : 123
- 2) Lavanya ID : 135
- 3) Amit ID : 146
- 4) Nisha ID : 116
- 5) Sameer ID : 155
- 6) Vinayak ID : 171

4] Research :

- 1) Ishika ID : 176
- 2) Meera ID : 154
- 3) Devika ID : 184
- 4) Nikhil ID : 136
- 5) Manan ID : 197
- 6) Kabir ID : 185
- 7) Raghav ID : 189

5] Press :

- 1) Aryan ID : 187
- 2) Vaishali ID : 152
- 3) Arnav ID : 179
- 4) Rhea ID : 182

CASE 5:

```
ENTER YOUR CHOICE : 5
```

```
THANK YOU!
```

```
Exiting program...
```