CPE 112 Programming with Data Structures

A Digital Flashcard Learning

Created By

Nattakritta	Phetpranee	66070503418
Chutipon	Homsuwan	67070503463
Wai	Phyo	67070503487

May 5, 2025

King Mongkut's University of Technology Thonburi

CONTENTS

1. Team Responsibilities

1.1 Member Role and Contributions

- Member 1 [Nattakritta Phetpranee 66070503418]
  - Created the user login and signup system using a CSV file.
  - Implement load/save flashcards in a CSV file for each user.
- Member 2 [Chutipon Homsuwan 67070503463]
  - Developed flashcard management features (add, view, edit, delete).
  - Implemented category and subcategory classification for flashcards.
- Member 3 [Wai Phyo 67070503487]
  - Designed and implemented the practice mode with score sorting and feedback.
  - Created the score summary and flashcard scoring system.

2. Introduction and Problem Statement

2.1 Project Background and Problem Statement

Many students struggle during exam time because of limited preparation time, heavy workload, and lack of effective study methods. Standard learning methods, such as taking notes on paper, can be time-consuming, unorganized, and lack feedback. These methods may not provide clear feedback or suit every student's learning style, which can lead to poor performance or difficulty focusing. To solve this problem, we aim to create a digital flashcard learning system that allows students to design their study paths. The system aims to make learning more efficient through interactive feedback, organize content by category, and help students focus on the topics they find most difficult. With features like repetition, helpful feedback, and personalized learning paths, this system makes studying more effective, helps students learn faster, and remember better.

2.2 Project Objective

- To help users organize flashcards by category and assign a score to each card, representing how many points it's worth during practice.
- Allow users to create, view, edit, and delete flashcards easily.
- To support a practice mode with feedback that helps users evaluate their knowledge.
- Allow users to save and load their flashcards from external files (CSV), ensuring their data is safely stored and always available.
- Provide a user login and registration system to ensure that each user.

2.3 Project Scope

- Flashcard management (add, view, edit, delete).
- Practice mode with option sorting by score.
- Feedback for each question attempted.
- Category and subcategory classification.
- Score tracking and progress summary.
- Saving and loading flashcards using CSV files.
- Use of basic data structures such as linked lists, hash tables and array.
- User login and registration system to manage individual user access and store personal flashcards.

3. Conceptual Understanding

3.1 Data Structures Used and Why

1. Linked List

The primary storage for flashcards is a singly linked list. Each flashcard is represented by a struct card, and the list is connected via the next pointer.

- Easy to insert new Flashcards at the end.
- Linear traversal for viewing, editing, and deleting.
- Flexibility in dynamic memory allocation.

2. Hash Table

A hash table is used to store flashcards by their ID for fast lookup. It maps each flashcard's ID to its memory location using a simple hash function.

- Quick access for editing or deleing flashcards.
- Avoids linear traversal of the entire list.
- Improves performance as the number of flashcards increases.

3. Array

An array of pointers is used during practice mode to store flashcards that match the selected category and subcategory

- Easy sorting based on score using bubble sort.
- Fast access to flashcards by index during practice.
- Temporary storage without modifying the main linked list.
- Memory efficiency by storing only one pointer.

3.2 Comparison to Alternatives

1. Linked List vs Array

- Array requires resizing and shifting elements when inserting or deleting.
- Linked lists allow easier dynamic memory management and faster insert or delete operation.

2. Hash Table vs Binary Search Tree

- BTS provides O(log n) search time if balanced, but they are more complex to implement.
- Hash tables provide O(1) average-case lookup and are easier to use in this context.

3. Array of Pointers vs Sorting a Linked List

- Sorting a linked list involves complex pointer rearrangements.
- Arrays allow fast, easy sorting using simple algorithms and index-based access.

3.3 Pros and Cons of the Approach

Pros

- Simple and easy to implement using basic data structures.
- Fast flashcard access with a hash table.
- Flexible flashcard storage with a linked list.
- Organized by category and subcategory.

Cons

- No hash tale collision handling.
- Viewing still uses slow list traversal.
- Bubble sort is not ideal for large data.

4. System Overview and Functionalities

4.1 System Overview

A digital flashcard learning system is a console-based application that allows users to manage, review, and practice flashcards. The user can add new flashcards with category, subcategory, and score, then can view or modify them at any time. During practice mode, the system helps users focus on specific topics and gives feedback after each answer. All flashcards are saved and loaded from a CSV file to ensure data is not lost between sessions.

4.2 Key Functionalities

1. User Login and Registration

   The user can create new flashcards by entering a question, answer, score, category, and subcategory.

2. Add Flashcards

   The user can create new flashcards by entering a question, answer, score, category, and subcategory.

3. View Flashcards

   Displays all flashcards currently stored, showing ID, question, answer, score, category, and subcategory.

4. Edit Flashcard

   Allow users to update the question, answer, or score of a selected flashcard by ID

5. Delete Flashcards

   Removes a flashcard from the system after confirmation.

6. Practice Flashcard

   Users can choose a category and a subcategory to practice. This system sorts flashcards by score and gives feedback on each response.

7. Score Summary

   After practice, users receive a performance summary, including total score and number of correct answers.

8. Category Management

   Flashcards are organized by category and subcategory to help user focus their study.

9. Data Saving & Loading (CSV)

   Flashcards are automatically saved to a CSV file and loaded at the start of the program, to make sure user data is saved between sessions.

5. Data Structures and Implementation

5.1 Data Structures Used

- **Singly Linked List**
  - Used to store all flashcards in memory dynamically.
  - Allows efficient traversal, insertion, and deletion without size limits.

- **Temporary Dynamic Array**
  - Built from the linked list during sorting (e.g., practice mode by score).
  - Allows efficient sorting using standard algorithms like bubble sort.

- **Hash Table (Optional/Advanced Feature)**
  - Used to categorize flashcards or optimize lookups.
  - Each bucket points to a list of flashcards that share the same category.
  - Simple hash function based on the first character or a sum of ASCII values of the category name.
  - Helps in quick access to flashcards for a specific category.

5.2 Code Walkthrough

- **Authentication**
  - `SignupUser()` appends user credentials to `users.csv`.
  - `loginUser()` scans credentials from the file and validates input.
  - On success, the username is stored in `currentuser.username`.
- **Flashcard File I/O**
  - On login, flashcards are loaded from `<username>_flashcards.csv` using `loadFlashcardsFromCSV()`.
  - Each flashcard is parsed and inserted into the linked list and hash table (if used).
- **Main Application Loop**
  - After login, the user accesses the main menu via `showMenu()`.
  - Options include:
    - Add Flashcard
    - View Flashcards
    - Edit Flashcard
    - Delete Flashcard
    - Practice Flashcards
- **Practice Mode**
  - User selects a category to practice.
  - Flashcards from that category (filtered via hash table) are sorted by score.
  - Score is updated depending on user's answer accuracy.
- **Flashcard CRUD Operations**
  - `addFlashcard()` – Adds a new flashcard to both the list and hash table.
  - `editFlashcard()` – Edits an existing flashcard by ID.
  - `deleteFlashcard()` – Removes a flashcard from the list and hash table.
  - `viewFlashcards()` – Displays all flashcards.

5.3 Tools Used and Constraints

Tools Used:

- **Language:** C
- **Libraries:** Standard C libraries (`stdio.h`, `stdlib.h`, `string.h`)
- **Editor:** Any (VS Code, Code::Blocks, etc.)
- **File Format:** CSV for both users and flashcards

Constraints:

- No external libraries (e.g., no OpenSSL for password hashing).
- User credentials are stored as plain text (security risk for production).
- Flashcards are stored in memory; large datasets may slow performance.
- Hash table uses simple hashing, which may lead to collisions.

6. Time and Space Complexity

6.1 Time Complexity

Operation	Best / Average Case	Worst Case	Notes
Login/Signup	O(n)	O(n)	Scans up to `n` lines in `users.csv`.
Add Flashcard	O(1)	O(1)	Insert at end of linked list or hash table bucket.
View Flashcards	O(n)	O(n)	Traverse the entire linked list.
Edit/Delete Flashcard	O(n)	O(n)	Linear search by ID through list.
Practice Flashcards	O(k log k)	$O(k^2)$ (bubble sort)	Sorting flashcards in a category (k = number in category).
Hash Table Insert/Find	O(1) (amortized)	O(n) (in case of collision)	If chaining is used, worst case is linear in a bucket.
Load from CSV	O(n)	O(n)	One read and parse per line.
Save to CSV	O(n)	O(n)	One write per flashcard.

n = number of flashcards currently in memory.

6.2 Space Complexity

Component	Space Usage	Notes
Linked List	O(n)	One `struct card` per flashcard.
Hash Table (optional)	O(n + m)	`n` flashcards + `m` buckets; typically `m << n`.
Temporary Array (Sort)	O(k)	Only during practice mode (copy of a category's flashcards).
User Info	O(1)	Only stores one active user in memory.
File Buffers	O(1)	Temporary buffers for reading lines from file.

- **Time Complexity:** Most user operations are linear due to file I/O and linked list traversal. Practice mode sorting could be improved by using a faster sort.
- **Space Complexity:** Mainly linear in the number of flashcards. Optional hash table adds minor overhead but improves lookup time.

7. Testing

7.1 Test Plan and Test Cases

Test Case ID	Description	Input	Expected Output	Actual Output	Status
TC01	Sign up with valid credentials	Username: `user1`, Password: `pass123`	Registration successful message	As expected	Pass
TC02	Sign up with same username	Username: `user1`, Password: `pass456`	Duplicate user check (if implemented) or allow	Allowed	Pass
TC03	Login with correct credentials	Username: `user1`, Password: `pass123`	Login successful message	As expected	Pass
TC04	Login with wrong password	Username: `user1`, Password: `wrongpass`	Incorrect password message	As expected	Pass
TC05	Login with non-existing user	Username: `userX`, Password: `pass`	Username not found message	As expected	Pass
TC06	Add a flashcard	Term: `CPU`, Definition: `Central Unit`	Flashcard added to file	As expected	Pass
TC07	View flashcards	-	Flashcard list displayed	As expected	Pass
TC08	Edit existing flashcard	Old term: `CPU`, New: `CPU`, New def: `Central Processing Unit`	Updated correctly	As expected	Pass
TC09	Delete a flashcard	Term: `CPU`	Flashcard deleted from file	As expected	Pass
TC10	Practice flashcards (quiz)	-	Prompts questions with answer check	As expected	Pass
TC11	Exit program	Choice: 6	Program terminates	As expected	Pass

7.2 Testing Results

- The application passed all 11 test cases.
- Core functionalities including user registration, login, flashcard CRUD operations, and quiz mode worked reliably.
- No crashes or unexpected behaviors were found during basic functional tests.
- Limitations such as no duplicate username check or input validation are noted as potential future improvement

8. Results and Discussion

8.1 System Output

```
--- Welcome to Flashcard Learning System ---
1. Sign Up
2. Login
3. Exit
Enter your choice: 1
Enter new username: admin
Enter new password: 123
Registration successful! You can log in to your account.

--- Welcome to Flashcard Learning System ---
1. Sign Up
2. Login
3. Exit
Enter your choice: 2
Enter username: admin
Enter password: 123
Login Successful! Welcome <admin>.
No existing data found. Creating new data now.

1. Add Flashcard
2. View Flashcards
3. Edit Flashcard
4. Delete Flashcard
5. Practice Flashcards
6. Logout
Enter your choice (1-6): _
```

Sign up and Login system

```
1. Add Flashcard
2. View Flashcards
3. Edit Flashcard
4. Delete Flashcard
5. Practice Flashcards
6. Logout
Enter your choice (1-6): 1
Enter question: 2+2=?
Enter answer: 4
Enter score (1-10): 5
Enter category: math
Enter subcategory: algebra
Flashcard added successfully with ID 1.

1. Add Flashcard
2. View Flashcards
3. Edit Flashcard
4. Delete Flashcard
5. Practice Flashcards
6. Logout
Enter your choice (1-6): 1
Enter question: 3+4=?
Enter answer: 7
Enter score (1-10): 5
Enter category: math
Enter subcategory: algebra
Flashcard added successfully with ID 2.
```

Add Flashcard

```
1. Add Flashcard
2. View Flashcards
3. Edit Flashcard
4. Delete Flashcard
5. Practice Flashcards
6. Logout
Enter your choice (1-6): 2

--- Flashcards ---
ID: 1
Category: math
Subcategory: algebra
Question: 2+2=?
Answer: 4
Score: 5

ID: 2
Category: math
Subcategory: algebra
Question: 3+4=?
Answer: 7
Score: 5
```

View Flashcard

```
1. Add Flashcard
2. View Flashcards
3. Edit Flashcard
4. Delete Flashcard
5. Practice Flashcards
6. Logout
Enter your choice (1-6): 5

Available Categories and Subcategories:
- math / algebra
Enter category: math
Enter subcategory: algebra

Sort by score?
1. Lowest to Highest
2. Highest to Lowest
3. None
Enter your choice (1-3): 3

Q: 5+9=?
Your answer: 14
Correct!
Do you want to continue? (y/n): y

Q: 3+4=?
Your answer: 7
Correct!
Do you want to continue? (y/n): y

Practice Summary: 2 correct out of 2 questions.
Total Score: 10
```

Practice Flashcard

```
1. Add Flashcard
2. View Flashcards
3. Edit Flashcard
4. Delete Flashcard
5. Practice Flashcards
6. Logout
Enter your choice (1-6): 4

--- Available Flashcards ---
ID: 1 | Category: math | Subcategory: algebra | Question: 5+9=?
ID: 4 | Category: math | Subcategory: algebra | Question: 3+4=?
ID: 5 | Category: math | Subcategory: algebra | Question: 231+512=?

Enter the ID of the flashcard to delete: 5

Flashcard found:
ID: 5
Category: math
Subcategory: algebra
Question: 231+512=?
Answer: 743
Score: 7
Are you sure you want to delete this flashcard? (y/n): y
Flashcard deleted successfully.
```

Delete Flashcard

```
1. Add Flashcard
2. View Flashcards
3. Edit Flashcard
4. Delete Flashcard
5. Practice Flashcards
6. Logout
Enter your choice (1-6): 3

--- Available Flashcards ---
ID: 1 | Category: math | Subcategory: algebra | Question: 2+2=?
ID: 2 | Category: math | Subcategory: algebra | Question: 3+4=?

Enter the ID of the flashcard to edit: 1

Flashcard found:
ID: 1
Category: math
Subcategory: algebra
Question: 2+2=?
Answer: 4
Score: 5

You can edit any field, or just press Enter to keep it the same.
New Question (or press Enter to keep): 5+9=?
New Answer (or press Enter to keep): 14
New score (or press Enter to keep):
Flashcard updated successfully.
```

Edit Flashcard

8.2 Discussion

Goals Met:

The system successfully achieved the goal of creating a digital flashcard learning platform. It allows users to add, view, edit, and delete flashcards, as well as practice them with feedback. The integration of categories and subcategories helps users organize flashcards efficiently. The practice mode, where flashcards are sorted by score and feedback is provided, makes the learning process interactive and engaging. The ability to save and load flashcards using CSV files ensures data persistence between sessions. Overall, the project meets its main objectives and provides a functional and user-friendly solution.

Performance Issues or Trade-offs:

A few trade-offs were encountered during the project:

- **Hash Table Collision Handling**: The hash table implementation does not handle collisions, which could become an issue if the system scales up or if multiple flashcards end up with the same ID hash.

- **Slow List Traversal for Viewing**: The program's reliance on linked list traversal for viewing and editing flashcards can be inefficient, especially as the number of flashcards increases. Although a hash table is used for fast lookups, certain operations still involve linear traversal through the list.

- **Bubble Sort Efficiency**: Bubble sort was used for sorting flashcards by score during practice mode. While simple to implement, bubble sort is inefficient for larger data sets (O(n^2)), which could hinder performance as the flashcard collection grows.

- **Fixed Hash Table Size**: The hash table size is fixed at 100, which could limit scalability. If the number of flashcards exceeds this limit, performance and data integrity could be compromised.

9. Challenges and Lessons Learned

Major Coding/Debugging Challenges:

- **Hash Table Implementation**: The primary challenge involved implementing the hash table, particularly with the decision not to handle collisions. This limited the scalability and could cause problems as more flashcards were added. Debugging the hash table and ensuring IDs were being mapped correctly took considerable effort.
- **Memory Management in Linked List**: Ensuring proper dynamic memory allocation and deallocation in the linked list was another challenge. Specifically, when removing flashcards from the linked list, memory management had to be handled carefully to avoid memory leaks or invalid memory access.
- **Sorting Flashcards**: Implementing the sorting mechanism using bubble sort was straightforward, but it led to inefficiencies as the number of flashcards grew. While it worked well for small datasets, optimizing the sorting algorithm for larger datasets was a challenge that could be addressed in future versions.
- **CSV File Handling**: Ensuring data consistency between in-memory structures and the CSV file posed its own set of challenges. The system needed to properly save and load data without causing duplication or loss of flashcards.

New Concepts or Skills Gained:

- **Data Structure Integration**: Working with linked lists, hash tables, and arrays in combination provided a deeper understanding of how to manage different data structures within a single project.
- **File I/O and Persistence**: The project emphasized the importance of data persistence, as it required loading and saving flashcards from CSV files. This taught the significance of file handling in software development.
- **Sorting Algorithms**: Though bubble sort was used, the project introduced the concept of sorting data and analyzing its performance. Optimizing sorting algorithms would be a key takeaway for future projects.
- **Efficient Lookup with Hashing**: The use of hash tables for efficient lookups taught the importance of optimizing search operations, particularly in cases where quick access to data is critical.

10. Real-World Reflections

Educational Application:

The digital flashcard learning system is a solid foundation for an educational tool. In its current form, it allows students to organize, review, and practice flashcards effectively. To make it more robust for real-world use, additional features like spaced repetition, progress tracking, and customizable practice modes could be incorporated. The use of categories and subcategories also provides a way for students to focus on the most challenging subjects. With more interactive and personalized features, this system could help students prepare for exams more efficiently.

Relation to Class Concepts:

- **Data Structures**: The project is deeply connected to class concepts, particularly with the use of linked lists, hash tables, and arrays. Understanding how to manage dynamic memory, optimize lookups, and efficiently store and retrieve data was central to the project.

- **Algorithm Optimization**: The experience with bubble sort and recognizing its inefficiency in larger datasets highlighted the importance of selecting the right algorithm for performance. This aligns with class teachings on algorithm complexity and optimization.

- **Memory Management**: The use of dynamic memory allocation with malloc() and freeing memory using free() in the linked list underscored key class lessons on memory management and preventing memory leaks.

- **File Handling**: The necessity of saving and loading data from a CSV file connected the project to concepts of data persistence and file input/output, which are important in real-world applications where data needs to be retained between sessions.

These reflections provide insights into how the project relates to key concepts learned in class and how the system could be expanded for real-world applications. It emphasizes the importance of efficient data handling, memory management, and algorithm optimization in building scalable and effective systems.