N.B.K.R. INSTITUTE OF SCIENCE AND TECHNOLOGY

(Vidyanagar)

DIGITAL DAIRY

Course: Data Structures

Branch: CSE

Section: D

Year: I

Semester: II

Submitted To: Ashok Selva Kumar E

Submitted By:

P. Deekshitha (24KB1A0DG)

K. Priyanka (24KB1A05V1)

0. Gnana Prasunamba (24KB1A05CG)

K. Likitha(24KB1A05T4)

Acknowledgment:

I would like to express my sincere gratitude to all those who have inspired and supported me throughout the process of maintaining this digital diary. This platform has allowed me to reflect, grow, and preserve valuable memories. Special thanks to my inner circle and well-wishers for their constant encouragement and understanding.

Abstract:

This digital diary project, developed using the C programming language, is designed to help users store, manage, and retrieve personal entries efficiently. By integrating fundamental concepts of Data Structures and Algorithms (DSA), such as linked lists, file handling, and search/sort operations, the system ensures organized data storage and quick access to diary entries. The program provides features like adding, viewing, editing, deleting, and searching diary records, all through a user-friendly console interface. This project not only demonstrates the practical application of C and DSA but also enhances understanding of memory management and structured programming.

Introduction:

In the digital age, maintaining personal records has evolved from traditional handwritten journals to computer-based diary systems. This project, a Digital Diary Program developed in the C programming language, provides a secure and structured platform for users to store their daily thoughts, events, and important notes. The application utilizes core programming concepts and data structures to offer features such as adding, editing, viewing, deleting, and searching diary entries. Designed with a user-friendly interface and efficient file handling, this program aims to simplify personal record-keeping while enhancing the understanding of programming logic and data organization in c.

Objective:

The primary objective of the Digital Diary Program is to develop a console-based application using the C programming language that allows users to securely create, store, manage, and retrieve personal diary entries. By implementing key concepts from Data Structures and Algorithms (DSA), the program aims to offer efficient data organization, easy navigation, and reliable file handling. This project also seeks to enhance practical knowledge of structured programming and logical problem-solving in real-world application.

Methodology:

The development of the Digital Diary Program follows a systematic approach divided into the following phases:

1. Requirement Analysis

Identify the core functionalities needed, such as creating, reading, updating, deleting, and searching diary entries. Determine the target user and environment for execution (console-based interface).

2. Design

Create a modular design using functions for each major feature. Choose appropriate data structures (e.g., structures, arrays, linked lists) for storing and managing diary entries.

3. Implementation

Develop the program using the C language. Implement file handling to enable data persistence, and apply DSA concepts (like linked lists or search algorithms) for efficient data operations.

4. Testing and Debugging

Test each function individually and as part of the whole system. Perform debugging to handle errors, ensure data integrity, and improve usability.

5. Evaluation

Verify that the program meets the defined objectives, such as ease of use, data security (like password protection), and reliability in storing diary entries.

6. Documentation

Document the source code and provide user guidance for installation, usage, and future enhancements.

Algorithm for Digital Diary Program:

1. Start

2. Display Main Menu

Options:

1. Add Entry

2. View Entry

3. Edit Entry

4. Delete Entry

5. Search Entry

6. Exit

3. Accept User Choice

4. Perform Action Based on Choice:

Add Entry

a. Prompt user for date and entry content

b. Open file in append mode

c. Save entry to file

d. Close file

View Entry

a. Prompt user for date or list all entries

b. Open file in read mode

c. Read and display matching entry/entries

d. Close file

Edit Entry

a. Prompt user for date of entry to edit

b. Load entries into temporary memory (e.g., array or linked list)

c. Modify the selected entry

d. Overwrite original file with updated entries

Delete Entry

a. Prompt user for date of entry to delete

b. Load entries into temporary memory

c. Remove the selected entry

d. Overwrite original file with remaining entries

Search Entry

a. Prompt for keyword or date

b. Open file and search through entries

c. Display matching result(s)

Exit

a. Terminate the program

5. Repeat from Step 2 Until User Chooses to Exit

6. End

Program Code:

```
include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Structure to represent a Diary Entry
typedef struct DiaryEntry {
    char date[15];    // Date in format YYYY-MM-DD
    char time[10];    // Time in format HH:MM
    char note[500];   // The content of the diary entry
    struct DiaryEntry *next;  // Pointer to the next entry
in the list
} DiaryEntry;

// Function to add a new entry to the diary
void addEntry(DiaryEntry **head) {
    DiaryEntry *newEntry = (DiaryEntry
*)malloc(sizeof(DiaryEntry));
```

```
    printf("Enter Date (YYYY-MM-DD): ");
    scanf("%s", newEntry->date);
    printf("Enter Time (HH:MM): ");
    scanf("%s", newEntry->time);
    getchar();  // To consume the newline character left by scanf
    printf("Enter your diary note: ");
    fgets(newEntry->note, sizeof(newEntry->note), stdin);

    newEntry->next = NULL;

    // Insert the new entry at the end of the list
    if (*head == NULL) {
        *head = newEntry;
    } else {
        DiaryEntry *temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newEntry;
```

```
    }
    printf("Diary entry added successfully!\n");
}

// Function to display all entries in the diary
void viewEntries(DiaryEntry *head) {
    if (head == NULL) {
        printf("No diary entries found!\n");
        return;
    }

    DiaryEntry *temp = head;
    while (temp != NULL) {
        printf("\nDate: %s\n", temp->date);
        printf("Time: %s\n", temp->time);
        printf("Note: %s\n", temp->note);
        temp = temp->next;
    }
}

// Function to edit an entry based on date
void editEntry(DiaryEntry *head, char *date) {
```

```
    DiaryEntry *temp = head;
    int found = 0;

    while (temp != NULL) {
        if (strcmp(temp->date, date) == 0) {
            found = 1;
            printf("Editing entry for %s:\n", date);
            printf("Enter new Time (HH:MM): ");
            scanf("%s", temp->time);
            getchar();  // To consume newline
            printf("Enter new diary note: ");
            fgets(temp->note, sizeof(temp->note), stdin);
            printf("Diary entry updated successfully!\n");
            break;
        }
        temp = temp->next;
    }

    if (!found) {
        printf("No entry found for the given date.\n");
    }
}
```

```
// Function to delete an entry based on date
void deleteEntry(DiaryEntry **head, char *date) {
    if (*head == NULL) {
        printf("Diary is empty.\n");
        return;
    }

    DiaryEntry *temp = *head, *prev = NULL;
    if (strcmp(temp->date, date) == 0) {
        *head = temp->next;  // Move head to the next
entry
        free(temp);  // Free the memory of the deleted
entry
        printf("Diary entry for %s deleted successfully!\n",
date);
        return;
    }

    while (temp != NULL && strcmp(temp->date, date)
!= 0) {
        prev = temp;
        temp = temp->next;
```

```
    }

    if (temp == NULL) {
        printf("No entry found for the given date.\n");
        return;
    }

    prev->next = temp->next;  // Unlink the node from the list
    free(temp);  // Free the memory of the deleted entry
    printf("Diary entry for %s deleted successfully!\n", date);
}

// Main function to interact with the user
int main() {
    DiaryEntry *head = NULL;
    int choice;
    char date[15];

    while (1) {
        printf("\nDigital Diary Menu:\n");
```

```
        printf("1. Add Entry\n");
        printf("2. View Entries\n");
        printf("3. Edit Entry\n");
        printf("4. Delete Entry\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();  // To consume newline after scanf

        switch (choice) {
            case 1:
                addEntry(&head);
                break;
            case 2:
                viewEntries(head);
                break;
            case 3:
                printf("Enter the date of the entry to edit
(YYYY-MM-DD): ");
                scanf("%s", date);
                editEntry(head, date);
                break;
```

```
            case 4:
                printf("Enter the date of the entry to delete
(YYYY-MM-DD): ");
                scanf("%s", date);
                deleteEntry(&head, date);
                break;
            case 5:
                printf("Exiting Digital Diary...\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```

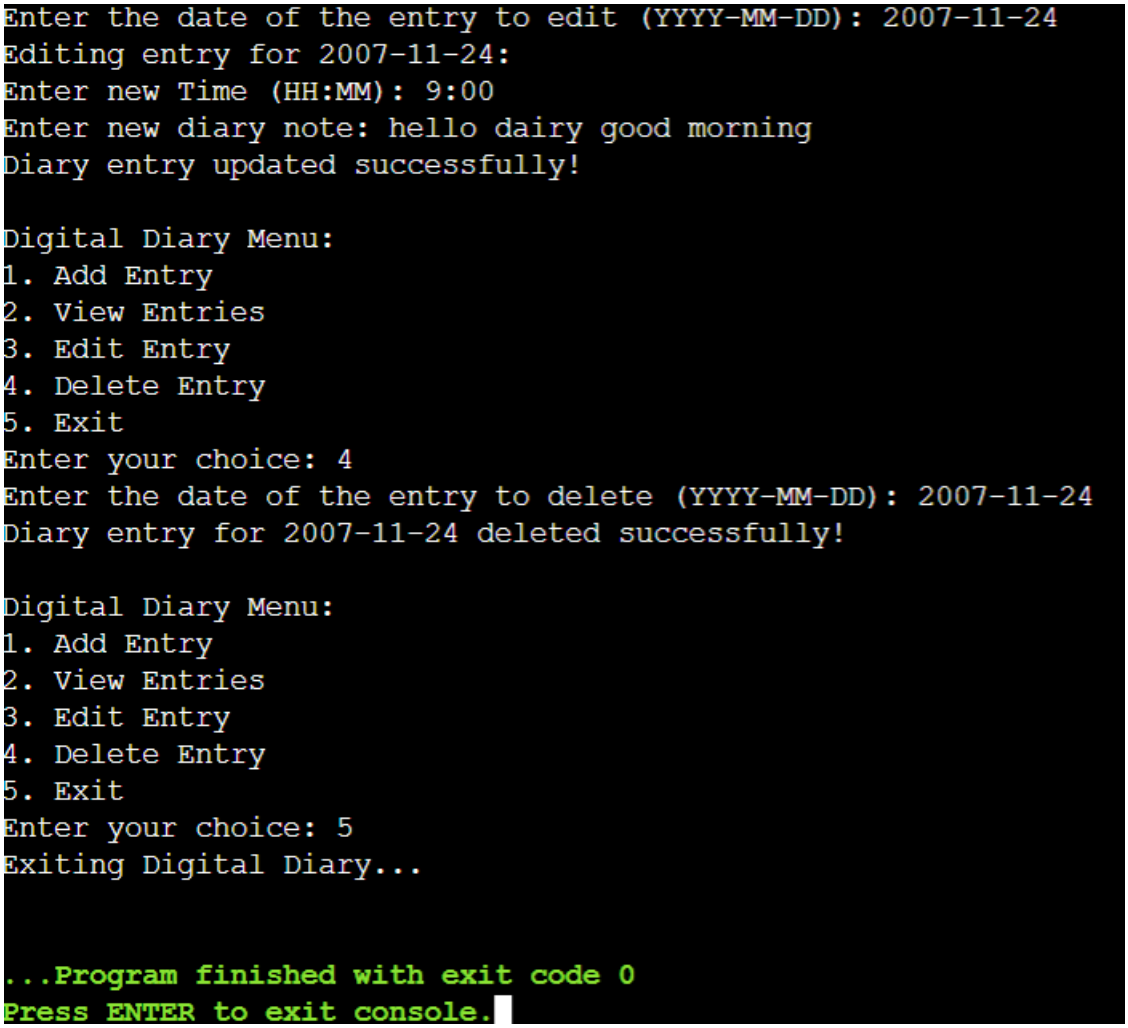
Output:

```
Digital Diary Menu:
1. Add Entry
2. View Entries
3. Edit Entry
4. Delete Entry
5. Exit
Enter your choice: 1
Enter Date (YYYY-MM-DD): 2007-11-24
Enter Time (HH:MM): 8:00
Enter your diary note: hello dairy
Diary entry added successfully!

Digital Diary Menu:
1. Add Entry
2. View Entries
3. Edit Entry
4. Delete Entry
5. Exit
Enter your choice: 2

Date: 2007-11-24
Time: 8:00
Note: hello dairy


Digital Diary Menu:
1. Add Entry
2. View Entries
3. Edit Entry
4. Delete Entry
5. Exit
Enter your choice: 3
```



Limitations:

1. Console-Based Interface

The program operates through a text-based interface, which may not be as user-friendly or visually appealing as graphical user interfaces (GUIs).

2. No Data Encryption

ntries are usually stored in plain text files without encryption, making sensitive data vulnerable to unauthorized access.

3. Limited File Handling Features

The program may not handle very large files efficiently, and lacks advanced features like file compression, backup, or cloud sync.

4. Basic Security

If implemented, password protection is typically minimal and not encrypted, which may be easy to bypass.

5. No Multi-User Support

The system is designed for a single user and does not support multiple profiles or access levels.

6. Limited Error Handling

User input validation and exception handling may be basic, which could lead to unexpected program behavior or crashes.

7. Scalability Issues

As the number of entries grows, performance may degrade due to linear search methods and lack of optimized data structures.

8. Platform Dependency

Depending on how file paths and system calls are written, the program may not run consistently across different operating systems.

Conclusion:

The Digital Diary Program successfully demonstrates how core concepts of C programming and data structures can be applied to create a practical and user-friendly application. It allows users to maintain personal records by adding, viewing, editing, deleting, and searching diary entries through a simple console interface. The project highlights the use of file handling for data storage and the importance of modular design and structured logic. While it has certain limitations, such as basic security and a lack of graphical interface, it lays a strong foundation for further enhancement and real-world application. This project has not only improved programming skills but also provided valuable experience in designing and implementing a complete software solution.

References:

1. E. Balagurusamy, Programming in ANSI C, McGraw-Hill Education – For understanding C programming basics and file handling.

2. Yashavant Kanetkar, Let Us C, BPB Publications – For concepts on pointers, structures, and modular programming.

3. Mark Allen Weiss, Data Structures and Algorithm Analysis in C – For learning efficient data organization techniques.

4. Online C programming tutorials – www.geeksforgeeks.org, www.programiz.com, www.tutorialspoint.com – For implementation help and code examples.

5. Official GCC documentation – https://gcc.gnu.org/ – For compiler-specific information and troubleshooting.