NED UNIVERSITY OF ENGINEERING AND TECHNOLOGY

COMPUTER ENGINEERING WORKSHOP
COMPLEX ENGINEERING PROJECT:

Task Tracker Utility Program in C

Course Code: CS-218

Submitted to:

Ms. Hina Danish

Submitted by:

Usman Rasheed Siddiqui (CS-24038)

Huzaifa Hanif (CS-24039)

Muhammad Ahmed Qazi (CS-24045)

Mujtaba Jawaid Rao (CS-24047)

Table of Contents

1 Project Overview

The Task Tracker Utility Program is a simple command-line tool developed in C that allows users to manage tasks efficiently. Users can **add, view, modify, and delete tasks**, with data persisted in a text file. This utility demonstrates the use of **file handling, dynamic memory allocation, and structures in C**.

2 Objective

1. To develop a task management system in C.
2. To implement **dynamic memory allocation** for handling variable numbers of tasks.
3. To learn **file handling** techniques for saving and retrieving task data.
4. To practice **structured programming** using functions and structures.
5. To allow basic CRUD (Create, Read, Update, Delete) operations for tasks.

3 Technologies Utilized

1. **Programming Language:** C
2. **Compiler:** GCC
3. **IDE/Text Editor:** Visual Studio Code
4. **Data Storage:** Text file (`Task_Database.txt`)

3.1 Program Design

3.2 Data Structure

```
typedef struct {
    int id;
    char title[100];
} Task;
```

- **id:** Unique Identity to each task
- **title:** Title/Description of each task

3.3 Functions Used

3.3.1 Saving And Loading

3.3.1.1 Loading Tasks

```
Task *loadTask(int *count)
```

- Reads the tasks from the text file (`Task_Database.txt`)
- Dynamically allocates memory to each task using `realloc`
- Returns pointer to the array of tasks and updates the task count

3.3.1.2 Saving Tasks

```
void saveTasks(Task *tasks, int count)
```

- Writes all the tasks back to the text file (`Task_Database.txt`)
- Updates the file whenever Add, Modify or Delete is called

3.3.2 Main Functions

3.3.2.1 Adding a Task

```
void Add()
```

- Prompts the user to enter the title of the task
- Automatically assigns an ID to the task
- Adds the task to memory and saves it to the file

3.3.2.2 Displaying a Task

```
void Display()
```

- Loads the tasks from the text file (`Task_Database.txt`)
- Displays all the tasks in tabular form showing IDs and Titles

3.3.2.3 Modifying a Task

```
void Modify()
```

- Prompts the user to enter the ID of the task (Can be viewed in Display)
- Allows updating the selected task's title
- Saves changes back to the file

3.3.2.4 Deleting a Task

```
void Delete()
```

- Prompts the user to enter the respective ID of the task to be deleted
- Removes the task from the memory and updates the file
- Reallocates IDs sequentially after deletion to maintain consistency

4 How it Works

4.1 Menu

When the Program Starts, the user sees a menu showing:

1. Add Task
2. Display Tasks
3. Delete Tasks
4. Modify Task
5. Exit

4.2 Adding a Task

- Program reads existing tasks using `loadTask`.
- New task ID is automatically generated.
- Task is added to memory and saved to file.

4.3 Displaying Tasks

- Loads all tasks from the file.
- Prints tasks in a table format.

4.4 Deleting a Task

- User inputs ID of the task to be deleted.
- Task array is updated; IDs are re-sequenced.
- Updated list is saved to file.

4.5 Modifying a Task

- User inputs ID to modify.
- Task title is updated and saved to the file.

4.6 Exit

- Program exits safely and frees all allocated memory.

5 Program Features

5.1 Dynamic Memory Allocation

Memory for a task is allocated using realloc:

```
tasks = realloc(tasks, sizeof(Task)*(count + 1));
```

- This allows the program to handle any number of tasks without a fixed array size.
- After the given operations, `free(tasks)` is used to release memory.

5.2 File Handling

Tasks are stored in `Task_Database.txt`. Format is as follows:

```
1 | Task 1 title

2 | Task 2 title
```

- File is opened in read mode for loading tasks.
- File is opened in write mode when saving tasks to overwrite old data.

6 Sample Output

```
---Task Tracker----
1. Add Task
2. Display Tasks
3. Delete Tasks
4. Modify Task
5. Exit
Enter your choice: 1
Enter Task title: Finish Assignment
Task Added Successfully ✅

Enter your choice: 2
ID    | TITLE
1     | Finish Assignment

Enter your choice: 1
Enter Task title: Buy groceries
Task Added Successfully ✅

Enter your choice: 1
Enter Task title: Complete projects
Task Added Successfully ✅

Enter your choice: 2
ID    | TITLE
1     | Finish Assignment
2     | Buy groceries
3     | Complete projects

Enter your choice: 3
Enter Task ID to delete: 2
Task Deleted Successfully ✅

Enter your choice: 2
ID    | TITLE
1     | Finish Assignment
2     | Complete projects

Enter your choice: 4
Enter Task ID to modify: 2
Enter new Title: Complete CEW project
Tasks Modified And Saved Successfully ✅

Enter your choice: 2
ID    | TITLE
1     | Finish Assignment
2     | Complete CEW project

Enter your choice: 5
Exiting...
```

7 Conclusion

The Task Tracker Utility program is a lightweight application that demonstrates dynamic memory allocation, file handling, and basic data structures in C. It allows users to efficiently manage tasks and can be expanded in the future to include features like deadlines, priorities, and task categories.

8 Future Improvements

While the current Task Tracker utility is functional, there are several enhancements that can make it more user-friendly and feature-rich:

- **Search Functionality:** Implement search functionality to quickly find tasks by ID, title, or keywords.
- **Deadlines:** Add due dates for tasks.
- **Task Sorting:** Implement sorting tasks by ID or name.