Daily Tracking System in C

Instructor: Mohsin F. Dar
University of Petroleum and Energy Studies

1. Title Page

- **Title:** Daily Tracking System in C
- **Student Name:** Ashish Kumar Singh
- **SAP ID :** 590024214
- **prof:** Mohsin F. Dar
- **Date:** 3rd December 2025

2. Abstract

This project implements a **Daily Tracking System** in C language. It allows users to create a list of tasks, assign time limits, monitor progress, and receive halfway reminders. The program demonstrates key C programming concepts including **structs, pointers, preprocessor macros, functions, loops, and conditional statements**. The system is modular, extensible, and aligned with the official repository and documentation guidelines.

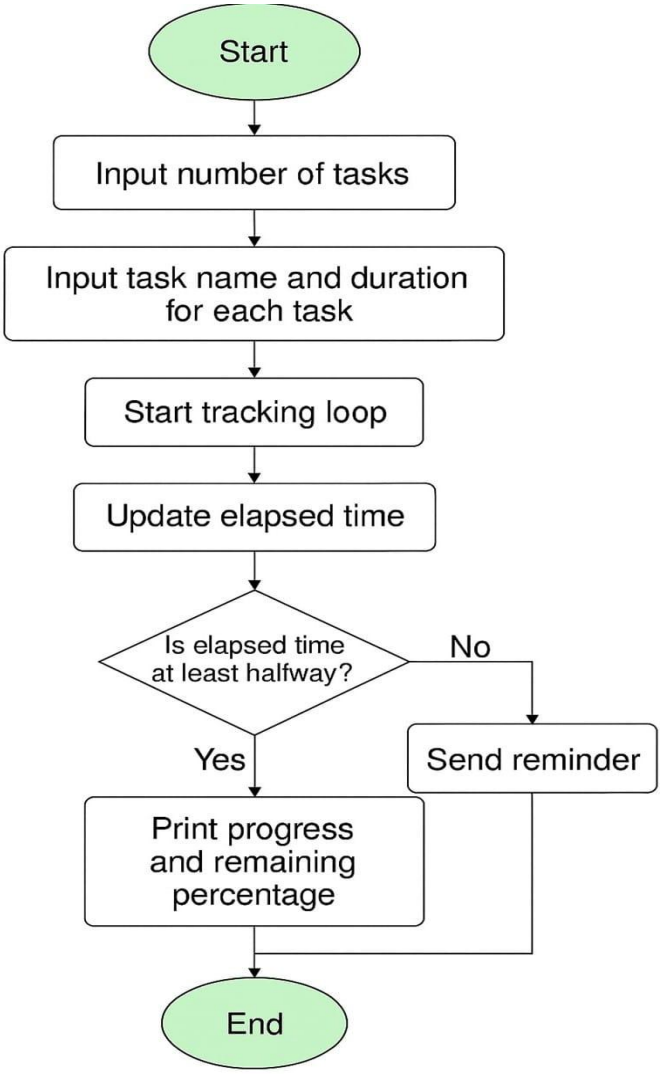
3. Problem Definition

Managing daily tasks efficiently is a common challenge. Traditional to-do lists lack real-time tracking and reminders. The problem addressed here is to design a **console-based system** that:

- Records tasks with time limits.
- Tracks progress dynamically.
- Provides reminders at halfway points.
- Reports completion status with percentages.

4. System Design

- 4.1 Flowchart: -



Steps:

1. Input number of tasks.
2. For each task: enter name and duration.
3. Start tracking loop.
4. Update elapsed time.
5. If elapsed ≥ half duration → send reminder.
6. Display progress and remaining percentage.
7. End when all tasks complete.

4.2 Algorithm: -

1. Initialize task list using struct Task.
2. For each task, store name, duration, start time.
3. Use loop to increment elapsed time.
4. Use if-else to check halfway condition.
5. Print progress using functions.
6. Repeat until all tasks complete.

5. Implementation Details

- **Structs:** Task struct stores name, duration, elapsed time, reminder flag.
- **Pointers:** Functions receive Task * for updates.
- **Macros:** LOG and ERR macros standardize output.
- **Functions:** init_task (), update_task (), maybe_send_half_reminder (), print_task_status ().
- **Loops:** while loop tracks progress until completion.
- **Conditionals:** if-else ensures reminders and validation.
- **Files:** Modular separation into src/ and include/.

6. Testing & Results

Test Case 1: Valid Input

Input: 3 study, time duration – 2min

Output:

Valid input execution for task 'study'

```
ashis@Ashish_1603 MINGW64 ~/OneDrive/Desktop/project
$ ./tracker
[LOG] Daily Tracking - Create your task list
Enter number of tasks: 1
Task 1 name: study
Time limit (minutes) for 'study': 2
[LOG] Started task 'study' for 2 minutes.
Task: study | Progress: 0.0% | Left: 100.0% | Remaining: 120 sec
Task: study | Progress: 0.8% | Left: 99.2% | Remaining: 119 sec
Task: study | Progress: 1.7% | Left: 98.3% | Remaining: 118 sec
Task: study | Progress: 2.5% | Left: 97.5% | Remaining: 117 sec
Task: study | Progress: 3.3% | Left: 96.7% | Remaining: 116 sec
```

Test Case 2: Invalid Input

Input: 0

Output:

Invalid input handling when task count is zero

```
ashis@Ashish_1603 MINGW64 ~/OneDrive/Desktop/project
$ ./tracker
[LOG] Daily Tracking - Create your task list
Enter number of tasks: 0
[ERR] Invalid number of tasks. Must be between 1 and 100.
```

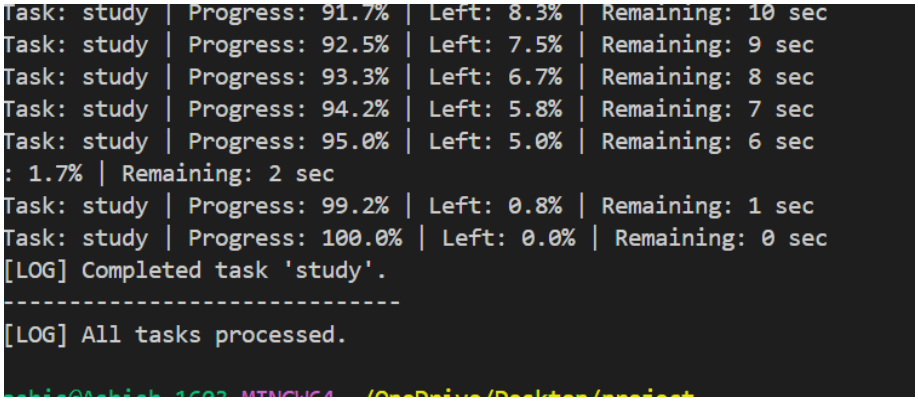
Output:

Halfway reminder triggered at 50% progress

```
Task: study | Progress: 45.0% | Left: 55.0% | Remaining: 66 sec
Task: study | Progress: 45.8% | Left: 54.2% | Remaining: 65 sec
Task: study | Progress: 46.7% | Left: 53.3% | Remaining: 64 sec
Task: study | Progress: 47.5% | Left: 52.5% | Remaining: 63 sec
Task: study | Progress: 48.3% | Left: 51.7% | Remaining: 62 sec
Task: study | Progress: 49.2% | Left: 50.8% | Remaining: 61 sec
Task: study | Progress: 50.0% | Left: 50.0% | Remaining: 60 sec
[LOG] Reminder: You're halfway through 'study'. Keep going!
Task: study | Progress: 50.8% | Left: 49.2% | Remaining: 59 sec
Task: study | Progress: 51.7% | Left: 48.3% | Remaining: 58 sec
Task: study | Progress: 52.5% | Left: 47.5% | Remaining: 57 sec
```

Output:

Final progress reaching 100% completion



7. Conclusion & Future Work

The Daily Tracking System successfully demonstrates modular C programming with real-time task tracking. It meets the rubric requirements for **problem definition, implementation, documentation, GitHub usage, originality, and execution validity**.

Future Enhancements:

- Persistent storage of tasks in files.
- Parallel tracking of multiple tasks.
- Pause/resume functionality.
- Configurable tick intervals for faster demos.

8. References

- Balagurusamy, E. – Programming in ANSI C, Tata McGrawHill.
- Online Tutorials – GeeksforGeeks (C Programming Basics and File Handling).
- Lecture notes provided by the faculty for C programming basics.
- Sample reference codes from GitHub (Open-source C projects for learning only).

THANKYOU