Qassim University

Daily Tasks Management System

Report

Course Code: CS 214
Instructor: Dr. Arwa Abdulaziz Al-Marzouqi

Prepared by:

Lama Alghofaily (Team Leader)

Member 2

Member 3

Introduction:

This project, titled **"Daily Tasks Management System"**, is designed to provide an effective solution for individuals who face difficulties managing their daily tasks. With the growing number of responsibilities in both personal and professional life, many people find it challenging to keep track of their tasks, meet deadlines, and prioritize their work. The system developed in this project aims to simplify the process by offering a text-based interface that allows users to create, organize, and categorize their daily tasks. The system provides features such as task status updates (e.g., New, In Progress, Completed), the ability to delete tasks, and the option to view tasks for today or all tasks. Through this system, users will be able to optimize their daily routines, enhance productivity, and reduce stress by staying organized and focused.

Project Idea (Problem):

In the modern world, time management has become one of the greatest challenges faced by individuals. The increasing number of daily tasks and obligations often results in missed deadlines, forgotten activities, and overall disorganization. Many people struggle with balancing multiple responsibilities, which leads to stress, inefficiency, and procrastination. The core problem that this project addresses is the lack of an efficient, user-friendly tool for managing daily tasks, setting priorities, and staying on top of deadlines. The main idea of this project is to create a digital solution that allows users to easily plan, organize, and track their tasks. By offering a clear structure and a user-friendly text-based interface, this system seeks to eliminate the chaos that comes with task overload and help individuals achieve better time management and productivity.

Objectives:

• **Organize daily tasks** by providing a structured platform for task creation, categorization, and prioritization.

• **Increase productivity** by helping users stay organized and on top of their tasks.

• **Enhance time management** by allowing users to easily prioritize tasks and track their progress.

• **Reduce stress and procrastination** by offering a clear overview of tasks and a manageable daily plan.

Explanation of the Code:

In this project, we have used a **linked list** as the data structure to manage the tasks. The reason for using a linked list is its efficiency in handling dynamic data. A linked list allows us to easily add, delete, or modify tasks without the need for reallocating or shifting large chunks of data, as would be required in an array. Each task is represented by a `Task` structure, which contains information about the task's name, date, status, and a pointer to the next task in the list. This structure allows for efficient management of tasks, where new tasks can be added at the beginning of the list, and tasks can be deleted from any point in the list by simply updating the appropriate pointers. Here are the main functions in the program:

1. AddTask()

This function is used to add a new task to the beginning of the linked list. It takes the task's name and date as input, creates a new task, and sets its status to "New" by default. The new task points to the current head of the list, and then the head is updated to point to the new task.

2. displayTasks()

This function allows the user to display tasks in two ways: either only today's tasks or all tasks. The function prompts the user to choose one of these options. It then traverses the linked list and displays the tasks based on the selected choice. If the user chooses to see today's tasks, it compares the date of each task to the current system date.

3. changeTaskStatus()

This function allows the user to change the status of a task. The status can be changed from "New" to "In Progress," or from "In Progress" to "Completed." The function searches through the linked list to find the task by its name and then updates its status if valid. If the task is not found or the status change is not allowed, the function provides an appropriate message.

4. deleteTask()

This function is used to delete a task from the list. It searches for the task by its name, and if found, it removes it from the list by updating the pointers. If the task is the head of the list, it updates the head pointer. If the task is found in the middle or end of the list, the previous task's pointer is updated to skip over the deleted task.

5. getCurrentDate()

This function retrieves the current date from the system and formats it in the "YYYY-MM-DD" format. It uses the C++ time functions to get the current time and then formats it accordingly to display as the current date.

Explanation of the main Function:

The main function is the entry point of the program. It runs an infinite loop that continuously presents the user with a menu of options to interact with the to-do list. Here is a breakdown of its structure:

1. **Menu Display:** The main function first displays a menu with 5 options for the user:

   1- Add a new task. 2-Display tasks. 3-Change the status of a task.

   4-Delete a task. 5-Exit the program.

2. **Input Handling:** After displaying the menu, the program waits for the user to enter a choice (an integer between 1 and 5). The program reads the user's choice and processes it. if the user enters an invalid option (a number outside of the range 1-5), the program uses “cin.clear()” to clear the error state and “cin.clear()” to remove any leftover input, then prompts the user to try again.

3. **Switch Statement:** The program uses a switch statement to determine which action to take based on the user's choice:

   - **Option 1 (Add New Task):**
     The user is prompted to enter the task name and the date for the task. The addTask() function is then called to add the new task to the list.

   - **Option 2 (Display Tasks):**
     The displayTasks() function is called, which displays either today's tasks or all tasks based on the user's input.

   - **Option 3 (Change Task Status):**
     The user is asked for the task name and the new status (either "In Progress", or "Completed"). The changeTaskStatus() function is called to update the status of the task.

- **Option 4 (Delete Task):**
  The user is asked for the name of the task they want to delete. The deleteTask() function is then called to remove the task from the list.

- **Option 5 (Exit):**
  If the user selects this option, the program prints a message indicating the exit and returns 0, which terminates the program.

4. **Error Handling:** If the user enters an invalid choice (anything other than 1, 2, 3, 4, or 5), the program will notify the user and ask them to try again by continuing the loop. The program ensures that the user can only proceed with valid choices.

Output:

ADD TASK

```
Welcome to To-Do List!
1. Add New Task
2. Display Tasks
3. Change Task Status
4. Delete a Task
5. Exit
Enter your choice: 1

Enter task name: Finish homework
Enter task date (YYYY-MM-DD): 2025-4-28
Task added: Finish homework for 2025-4-28

Welcome to To-Do List!
1. Add New Task
2. Display Tasks
3. Change Task Status
4. Delete a Task
5. Exit
Enter your choice: 1

Enter task name: Buy groceries
Enter task date (YYYY-MM-DD): 2025-4-28
Task added: Buy groceries for 2025-4-28

Welcome to To-Do List!
1. Add New Task
2. Display Tasks
3. Change Task Status
4. Delete a Task
5. Exit
Enter your choice: 1

Enter task name: Read a new book
Enter task date (YYYY-MM-DD): 2025-4-29
Task added: Read a new book for 2025-4-29
```

DISPLAY TASK

```
Welcome to To-Do List!
1. Add New Task
2. Display Tasks
3. Change Task Status
4. Delete a Task
5. Exit
Enter your choice: 2

Choose an option:
1. Display tasks for today
2. Display all tasks
1
Tasks for today:
1. Buy groceries - 2025-4-28 - Status: New
2. Finish homework - 2025-4-28 - Status: New

Welcome to To-Do List!
1. Add New Task
2. Display Tasks
3. Change Task Status
4. Delete a Task
5. Exit
Enter your choice: 2

Choose an option:
1. Display tasks for today
2. Display all tasks
2
All tasks:
1. Read a new book - 2025-4-29 - Status: New
2. Buy groceries - 2025-4-28 - Status: New
3. Finish homework - 2025-4-28 - Status: New
```

```
Welcome to To-Do List!
1. Add New Task
2. Display Tasks
3. Change Task Status
4. Delete a Task
5. Exit
Enter your choice: 3

Enter task name to change status: Finish homework
Enter new status (New, In Progress, Completed): Completed
Task "Finish homework" status changed to Completed

Welcome to To-Do List!
1. Add New Task
2. Display Tasks
3. Change Task Status
4. Delete a Task
5. Exit
Enter your choice: 2

Choose an option:
1. Display tasks for today
2. Display all tasks
2
All tasks:
1. Read a new book - 2025-4-29 - Status: New
2. Buy groceries - 2025-4-28 - Status: New
3. Finish homework - 2025-4-28 - Status: Completed
```

CHANGE STATUS TASK

```
Welcome to To-Do List!
1. Add New Task
2. Display Tasks
3. Change Task Status
4. Delete a Task
5. Exit
Enter your choice: 4

Enter task name to delete: Buy groceries
Task "Buy groceries" deleted!

Welcome to To-Do List!
1. Add New Task
2. Display Tasks
3. Change Task Status
4. Delete a Task
5. Exit
Enter your choice: 2

Choose an option:
1. Display tasks for today
2. Display all tasks
2
All tasks:
1. Read a new book - 2025-4-29 - Status: New
2. Finish homework - 2025-4-28 - Status: Completed

Welcome to To-Do List!
1. Add New Task
2. Display Tasks
3. Change Task Status
4. Delete a Task
5. Exit
Enter your choice: 5

Exiting the program.
PS C:\Users\Lsgh5>
```

DELETE TASK

Conclusion:

In conclusion, our **"Daily Tasks Management System"** project effectively demonstrates the use of singly linked lists to manage dynamic data efficiently. We implemented core functionalities such as adding new tasks, displaying tasks (for today or all), changing the status of tasks, and deleting tasks. By relying on linked lists, we ensured that the system could handle a changing number of tasks without worrying about memory allocation issues. Throughout this project, we deepened our understanding of dynamic memory management, pointers, and data organization in C++. In addition to strengthening our technical programming skills, this project also enhanced our abilities in system design, logical thinking, and teamwork. Overall, we successfully built a flexible and user-friendly application that can assist users in organizing their daily tasks efficiently. Future improvements could include integrating a graphical user interface (GUI) and additional features like task prioritization and reminders.

Thank you.