COMSATS University Islamabad

Attock Campus

Department of Computer Science

Program: BS-SE

Submitted by: HARIS KHAN

Registration no: SP23-BSE-006

Assignment no: 01

Submitted to: Sir Kamran Ghulam

Course: DS

Date: 24 - 09 - 2024

Introduction:

The objective of this assignment is to create a simple task management system using a singly linked list in C++. The system stores and manages tasks, where each task is represented as a node in the list. The tasks are ordered based on their priority (with higher priority tasks appearing earlier in the list). The system allows users to add new tasks, remove tasks, and view tasks through a console-based menu. Operations such as inserting tasks based on priority, removing the highest priority task, and removing tasks by their unique ID are implemented.

Code Explanation:

1. struct TaskNode:

- The **TaskNode** structure represents an individual task.

- Each task has three fields: **taskID** (an integer), **description** (a string), and **priority** (an integer). The **next** pointer is used to link one node to the next task in the list.

2. TaskManagement Class:

- TaskManagement() Constructor: Initializes the **head** of the singly linked list to **nullptr**, indicating that the list is empty at the beginning.

- addTask(int id, string desc, int priority):

- This function adds a new task to the list in the correct position based on its priority.

- If the list is empty or if the new task has a higher priority than the head, the new task is inserted at the beginning of the list.

- Otherwise, the function traverses the list to find the appropriate position and inserts the new task based on its priority, ensuring the list remains sorted.

- removeHighestPriorityTask():

- This function removes the task at the head of the list (i.e., the highest priority task).

- If the list is empty, a message is printed to indicate there are no tasks to remove.

- Otherwise, the head is updated to the next node, and the memory for the old head is deallocated.

- removeTaskByID(int id):

- This function removes a task with a specified task ID.

- If the task to be removed is the head, the head is updated to the next node.

- If the task is elsewhere in the list, the function traverses the list to find the task, removes it, and updates the links.

- If the task is not found, it prints an appropriate message.

- displayTasks():

- This function displays all tasks in the list.

- It traverses the linked list from the head to the end, printing the details (task ID, description, priority) of each task.

- If the list is empty, it prints that there are no tasks to display.

3. displayMenu():

- This function presents the user with the menu options: adding a task, viewing tasks, removing the highest priority task, removing a task by ID, and exiting.

- The menu is displayed in a loop, allowing the user to continuously interact with the system until they choose to exit.

4. main():

- The **main()** function runs the program, allowing the user to input their choice from the menu and execute the respective operations.

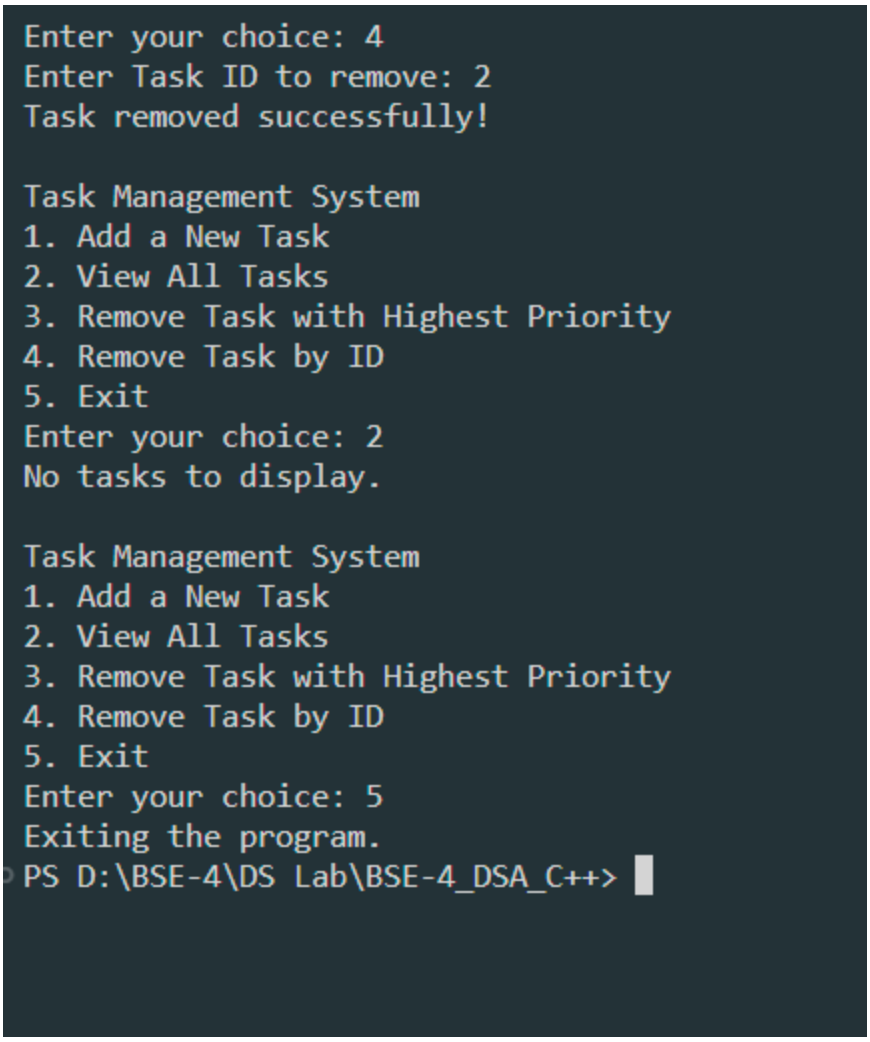
- Based on user input, tasks are added, viewed, or removed.

- It handles inputs and interacts with the **TaskManagement** class to perform operations.

OUTPUT:

```
Task Management System
1. Add a New Task
2. View All Tasks
3. Remove Task with Highest Priority
4. Remove Task by ID
5. Exit
Enter your choice: 1
Enter Task ID: 6
Enter Task Description: Most important task
Enter Task Priority (higher number means higher priority): 10
Task added successfully!

Task Management System
1. Add a New Task
2. View All Tasks
3. Remove Task with Highest Priority
4. Remove Task by ID
5. Exit
Enter your choice: 1
Enter Task ID: 2
Enter Task Description: very important task
Enter Task Priority (higher number means higher priority): 5
Task added successfully!

Task Management System
1. Add a New Task
2. View All Tasks
3. Remove Task with Highest Priority
4. Remove Task by ID
5. Exit
Enter your choice: 2

Current Tasks:
```

```
Enter your choice: 2

Current Tasks:
Task ID: 6 | Description: Most important task | Priority: 10
Task ID: 2 | Description: very important task | Priority: 5

Task Management System
1. Add a New Task
2. View All Tasks
3. Remove Task with Highest Priority
4. Remove Task by ID
5. Exit
Enter your choice: 3
Removing Task ID: 6 with Priority: 10

Task Management System
1. Add a New Task
2. View All Tasks
3. Remove Task with Highest Priority
4. Remove Task by ID
5. Exit
Enter your choice: 2

Current Tasks:
Task ID: 2 | Description: very important task | Priority: 5

Task Management System
1. Add a New Task
2. View All Tasks
3. Remove Task with Highest Priority
4. Remove Task by ID
5. Exit
Enter your choice: 4
```



Conclusion:

Through this assignment, I learned how to implement a task management system using a singly linked list in C++. The linked list structure allows for dynamic memory allocation and efficient insertion and deletion operations. I also practiced managing tasks based on priority and using pointers to link nodes in the list.