COMSATS UNIVERSITY ISLAMABAD (ATTOCK CAMPUSS)

PROGRAM BS-SE

NAME

Malaika Wasiq

REG NO

SP23-BSE-041

DATE

September 24, 2024

ASSIGNMENT

 # 01

COURSE

DS (THEORY)

SUBMITTED TO

SIR MUHAMMAD KAMRAAN

CODE

```
#include <iostream>
#include <string>
using namespace std;
struct TaskNode {
   int taskID;
   string description;
   int priority;
   TaskNode* next;
};
class TaskManager {
private:
   TaskNode* head;  // Pointer to the first node
public:
   TaskManager() {
      head = nullptr;  // Initialize the head as null
   }
   TaskNode* createTask(int id, string desc, int priority) {
      TaskNode* newTask = new TaskNode;
      newTask->taskID = id;
      newTask->description = desc;
      newTask->priority = priority;
      newTask->next = nullptr;
      return newTask;
   }
   void addTask(int id, string desc, int priority) {
      TaskNode* newTask = createTask(id, desc, priority);
      if (!head || head->priority < priority) { // Insert at head if higher priority
         newTask->next = head;
```

```
        head = newTask;
    } else {
        TaskNode* temp = head;
        while (temp->next && temp->next->priority >= priority) {
            temp = temp->next;
        }
        newTask->next = temp->next;
        temp->next = newTask;
    }
    cout << "Task added successfully.\n";
}
void removeHighestPriorityTask() {
    if (!head) {
        cout << "No tasks to remove.\n";
        return;
    }
    TaskNode* temp = head;
    head = head->next;
    cout << "Removed task with ID: " << temp->taskID << " (Highest priority).\n";
    delete temp;
}
void removeTaskByID(int id) {
    if (!head) {
        cout << "No tasks to remove.\n";
        return;
    }
    if (head->taskID == id) {  // If head is the task to be removed
        TaskNode* temp = head;
        head = head->next;
```

```
        delete temp;
        cout << "Task with ID " << id << " removed.\n";
        return;
    }
    TaskNode* temp = head;
    while (temp->next && temp->next->taskID != id) {
        temp = temp->next;
    }
    if (temp->next) {
        TaskNode* toDelete = temp->next;
        temp->next = toDelete->next;
        delete toDelete;
        cout << "Task with ID " << id << " removed.\n";
    } else {
        cout << "Task with ID " << id << " not found.\n";
    }
}
  void displayTasks() {
    if (!head) {
        cout << "No tasks to display.\n";
        return;
    }
    TaskNode* temp = head;
    while (temp) {
        cout << "Task ID: " << temp->taskID << ", Description: " << temp->description
             << ", Priority: " << temp->priority << "\n";
        temp = temp->next;
    }
  }
```

```
    ~TaskManager() {
        while (head) {
            TaskNode* temp = head;
            head = head->next;
            delete temp;
        }
    }
};
int main() {
    TaskManager manager;
    int choice, id, priority;
    string description;
    do {
        cout << "\nTask Manager Menu:\n";
        cout << "1. Add a new task\n";
        cout << "2. View all tasks\n";
        cout << "3. Remove the highest priority task\n";
        cout << "4. Remove a task by ID\n";
        cout << "5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                cout << "Enter task ID: ";
                cin >> id;
                cout << "Enter task description: ";
                cin.ignore();  // To ignore the newline character left by previous input
                getline(cin, description);
                cout << "Enter task priority: ";
```

```
            cin >> priority;
            manager.addTask(id, description, priority);
            break;
        case 2:
            cout << "All Tasks:\n";
            manager.displayTasks();
            break;
        case 3:
            manager.removeHighestPriorityTask();
            break;
        case 4:
            cout << "Enter task ID to remove: ";
            cin >> id;
            manager.removeTaskByID(id);
            break;
        case 5:
            cout << "Exiting Task Manager.\n";
            break;
        default:
            cout << "Invalid choice. Try again.\n";
      }
   } while (choice != 5);
   return 0;
}
```

Introduction

The provided C++ code implements a simple Task Manager application that allows users to manage tasks based on their priority. Using a linked list data structure, the program enables users to add new tasks, view all existing tasks, remove the highest priority task, and delete specific tasks by their unique ID. The Task Manager is designed to be user-friendly, providing a

console-based menu interface for easy interaction. This report outlines the functionality of the Task Manager and highlights its key features.

CODE EXPLANATION

The provided C++ code implements a task management system using a singly linked list. This system allows users to manage tasks by adding, viewing, and removing them based on their priority. Below is a detailed explanation of the code, including its structure, functionality, and expected output.

Structure

1. TaskNode Struct:

This struct represents a single task in the linked list.

It contains:

`taskID`: An integer representing the unique identifier for the task.

`description`: A string that describes the task.

`priority`: An integer indicating the priority level of the task (higher values indicate higher priority).

`next`: A pointer to the next `TaskNode` in the linked list.

2. TaskManager Class:

This class manages the linked list of tasks and provides methods to manipulate it.

Private Member:

`head`: A pointer to the first node (task) in the linked list.

3. Public Methods:

Constructor: Initializes `head` to `nullptr`, indicating that there are no tasks initially.

createTask(int id, string desc, int priority): Creates a new task node and returns a pointer to it.

addTask(int id, string desc, int priority): Adds a new task to the list based on its priority. If the new task has a higher priority than the current head, it becomes the new head; otherwise, it finds the correct position in the list.

removeHighestPriorityTask(): Removes and deletes the highest priority task (the head of the list).

removeTaskByID(int id): Searches for a task by its ID and removes it from the list if found.

displayTasks(): Displays all tasks currently in the list, including their ID, description, and priority.

Destructor: Cleans up all tasks when an instance of `TaskManager` is destroyed to prevent memory leaks.

Main Function

The `main()` function provides a menu-driven interface for user interaction:

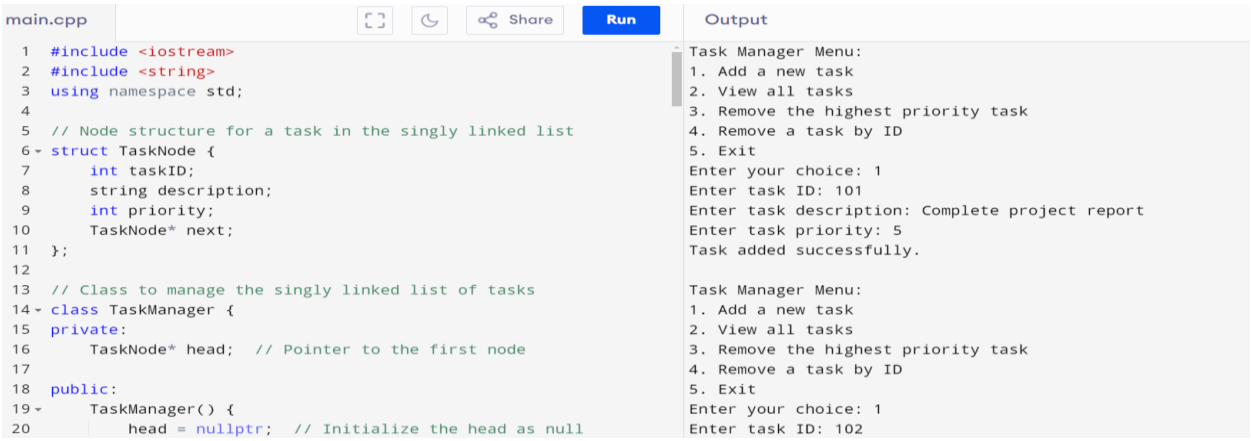
It displays options for adding tasks, viewing all tasks, removing the highest priority task, removing a specific task by ID, or exiting the program.

It uses a loop to continuously prompt the user until they choose to exit.

Conclusion

In conclusion, the Task Manager application effectively demonstrates the use of linked lists for dynamic data management in C++. By allowing users to prioritize tasks and perform various operations such as adding, viewing, and removing tasks, it serves as a practical tool for task management. The program's structure is modular and easy to understand, making it an excellent starting point for further enhancements, such as adding features like task editing or saving/loading tasks from a file. Overall, this Task Manager provides a solid foundation for managing tasks efficiently and can be expanded upon to meet more complex requirements in future iterations.

OUTPUT



main.cpp

Share | Run

```
    }

    // Function to create a new task node
    TaskNode* createTask(int id, string desc, int priority) {
        TaskNode* newTask = new TaskNode;
        newTask->taskID = id;
        newTask->description = desc;
        newTask->priority = priority;
        newTask->next = nullptr;
        return newTask;
    }

    // Function to add a new task in the list based on
        priority (higher priority first)
    void addTask(int id, string desc, int priority) {
        TaskNode* newTask = createTask(id, desc, priority);
        if (!head || head->priority < priority) { // Insert at
            head if higher priority
            newTask->next = head;
            head = newTask;
```

Output | Cle

```
Enter task description: Prepare presentation slides
Enter task priority: 3
Task added successfully.

Task Manager Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 2
All Tasks:
Task ID: 101, Description: Complete project report, Priority: 5
Task ID: 102, Description: Prepare presentation slides, Priority:

Task Manager Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
```

main.cpp

Share | Run

```
    } else {
        TaskNode* temp = head;
        while (temp->next && temp->next->priority >=
            priority) {
            temp = temp->next;
        }
        newTask->next = temp->next;
        temp->next = newTask;
    }
    cout << "Task added successfully.\n";
}

// Function to remove the highest priority task (from the
    start)
void removeHighestPriorityTask() {
    if (!head) {
        cout << "No tasks to remove.\n";
        return;
    }
    TaskNode* temp = head;
```

Output | Clear

```
5. Exit
Enter your choice: 3
Removed task with ID: 101 (Highest priority).

Task Manager Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 2
All Tasks:
Task ID: 102, Description: Prepare presentation slides, Priority: 3

Task Manager Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
```

main.cpp

Share Run

```
    } else {
        TaskNode* temp = head;
        while (temp->next && temp->next->priority >=
            priority) {
            temp = temp->next;
        }
        newTask->next = temp->next;
        temp->next = newTask;
    }
    cout << "Task added successfully.\n";
}

// Function to remove the highest priority task (from the
    start)
void removeHighestPriorityTask() {
    if (!head) {
        cout << "No tasks to remove.\n";
        return;
    }
    TaskNode* temp = head;
```

Output Clear

```
5. Exit
Enter your choice: 4
Enter task ID to remove: 102
Task with ID 102 removed.

Task Manager Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 2
No tasks to display.

Task Manager Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
```