COMSATS University Islamabad, Attock Campus

Department of Computer Science

Program: BS-SE

Title: Task Management System

Course: Data Structure

Name: Mehwish Tariq

Reg No: SP23-BSE-018

Assignment No: 01

Date: 24 Sep, 2024

Submitted to: Sir M. Kamran

Introduction:

The objective of this assignment is to create a simple console-based Task Manager using C++. This program allows users to manage tasks by adding them to a list, viewing all tasks, and removing them either by their priority or by a specific task ID. The tasks are stored in a linked list, and their order is based on their priority (higher priority tasks appear earlier in the list).

The key operations implemented are:

1. Adding a new task with an ID, description, and priority.
2. Viewing all the tasks.
3. Removing the task with the highest priority.
4. Removing a task by its ID.

Code Explanation:

1. addTask ():

Purpose: Adds a new task, ensuring it is placed based on priority (higher priority tasks come first).

Logic: If the list is empty or the task has the highest priority, it goes to the front. Otherwise, it finds the right position to insert it based on its priority.

2. removeHighestPriorityTask ()

Purpose: Removes the task with the highest priority (first task in the list).

Logic: If the list is empty, it shows a message. If not, it removes the first task and moves the second task to the front.

3. removeTaskById ()

Purpose: Removes a task based on its unique ID.

Logic: If the task is at the front, it's removed. If it's further in the list, the function searches for it and removes it without breaking the list structure.

4. viewTasks ()

Purpose: Displays all tasks in the list.

Logic: If the list is empty, it shows a message. If there are tasks, it goes through each one, printing the ID, description, and priority.

5. Main function:

Purpose: Provides a menu for the user to add, view, or remove tasks, or exit the program.

Logic: Runs in a loop, showing menu options and calling the right function based on the user's choice (e.g., adding a task or removing one).

Code:

[*] linked_list_operations.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

// Structure to represent each task in the linked list
struct Task {
    int id;                 // Unique ID for each task
    string description;      // Description of the task
    int priority;            // Priority level of the task (higher value means higher priority)
    Task* next;              // Pointer to the next task in the list
};
// Pointer to the head (start) of the task list
Task* head = NULL;  // NULL is used to indicate that the list is initially empty

// Function to add a new task in the list based on priority (higher priority comes first)
void addTask(int id, string description, int priority) {
    // Create a new task
    Task* newTask = new Task();     // Allocate memory for a new task node
    newTask->id = id;               // Set the task ID
    newTask->description = description; // Set the task description
    newTask->priority = priority;       // Set the task priority
    newTask->next = NULL;           // Initially, the new task doesn't point to any other task

    // If the list is empty, or the new task has the highest priority, insert it at the start
    if (head == NULL || head->priority < priority) {
        newTask->next = head;       // Point the new task to the current head
        head = newTask;             // Make the new task the head of the list
    } else {
        // Traverse the list to find the correct position for the new task
        Task* current = head;
        while (current->next != NULL && current->next->priority >= priority) {
            current = current->next;  // Move to the next task in the list
        }
        // Insert the new task at the correct position
        newTask->next = current->next;
        current->next = newTask;
    }

    cout << "Task added successfully!\n";
}
// Function to remove the task with the highest priority (first task in the list)
```

[*] linked_list_operations.cpp

```cpp
// Function to remove the task with the highest priority (first task in the list
void removeHighestPriorityTask() {
    if (head == NULL) {
        // If the list is empty, display a message
        cout << "Task list is empty.\n";
        return;
    }

    // Remove the head (highest priority task) and update the head pointer
    Task* temp = head;
    head = head->next;   // Move the head to the next task
    cout << "Removed task with ID: " << temp->id << endl;
    delete temp;         // Free the memory of the removed task
}

// Function to remove a task by its ID
void removeTaskById(int id) {
    if (head == NULL) {
        // If the list is empty, display a message
        cout << "Task list is empty.\n";
        return;
    }

    // If the task to remove is the first task (head)
    if (head->id == id) {
        Task* temp = head;
        head = head->next;  // Move the head to the next task
        cout << "Removed task with ID: " << id << endl;
        delete temp;        // Free the memory of the removed task
        return;
    }

    // Traverse the list to find the task with the given ID
    Task* current = head;
    while (current->next != NULL && current->next->id != id) {
        current = current->next;  // Move to the next task
    }

    // If the task with the given ID is found
    if (current->next != NULL) {
        Task* temp = current->next;
        current->next = current->next->next;  // Bypass the task to remove it
        cout << "Removed task with ID: " << id << endl;
        delete temp;       // Free the memory of the removed task
```

[*] linked_list_operations.cpp

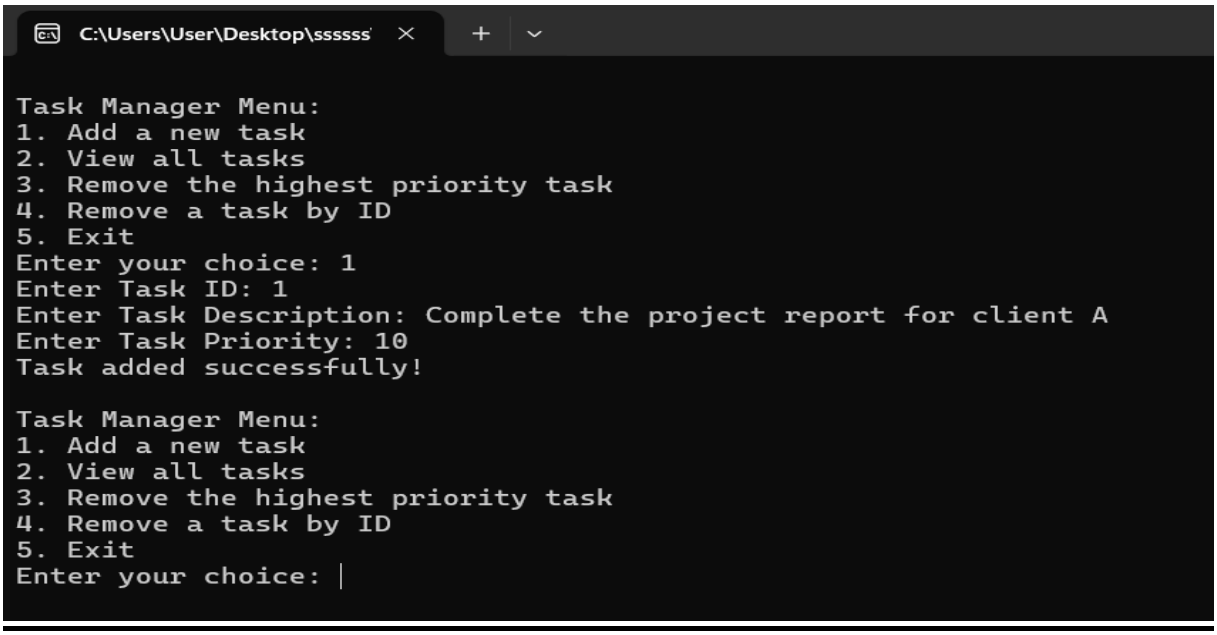
```
85      } else {
86          // If the task with the given ID was not found
87          cout << "Task with ID " << id << " not found.\n";
88      }
89  }
90
91  // Function to display all the tasks in the list
92  void viewTasks() {
93      if (head == NULL) {
94          // If the list is empty, display a message
95          cout << "Task list is empty.\n";
96          return;
97      }
98
99      // Traverse and display each task
100     Task* current = head;
101     cout << "Task List:\n";
102     while (current != NULL) {
103         // Print task details
104         cout << "ID: " << current->id << ", Description: " << current->description << ", Priority: " << current->priority << endl;
105         current = current->next;  // Move to the next task
106     }
107 }
108
109 // Main function: Console-based menu for user interaction
110 int main() {
111     int choice, id, priority;
112     string description;
113
114     // Infinite loop for menu-based interaction
115     while (true) {
116         // Display the menu options
117
118         cout << "\nTask Manager Menu:\n";
119         cout << "1. Add a new task\n";
120         cout << "2. View all tasks\n";
121         cout << "3. Remove the highest priority task\n";
122         cout << "4. Remove a task by ID\n";
123         cout << "5. Exit\n";
124
125         cout << "Enter your choice: ";
126         cin >> choice;
127
128         // Perform actions based on user choice
```

[*] linked_list_operations.cpp

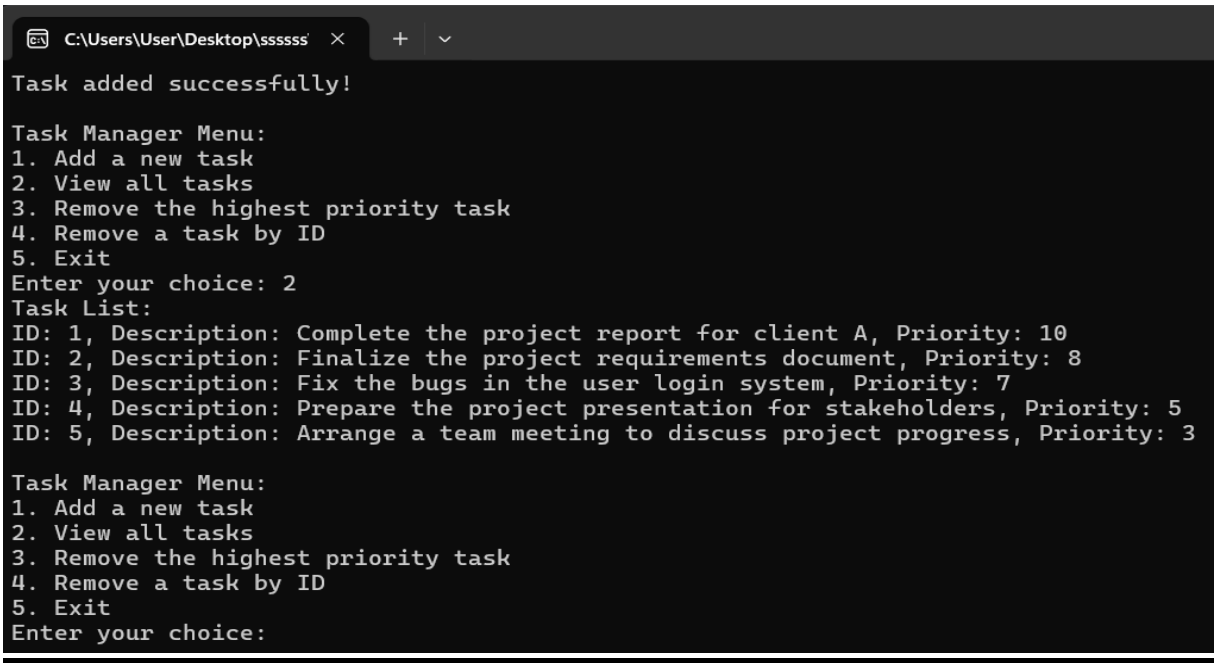
```cpp
        // Perform actions based on user choice
        switch (choice) {
            case 1:
                // Add a new task
                cout << "Enter Task ID: ";
                cin >> id;
                cin.ignore();  // Ignore the newline character after the previous input
                cout << "Enter Task Description: ";
                getline(cin, description);  // Read the full line as task description
                cout << "Enter Task Priority: ";
                cin >> priority;
                addTask(id, description, priority);  // Add the task
                break;

            case 2:
                // View all tasks in the list
                viewTasks();
                break;

            case 3:
                // Remove the task with the highest priority
                removeHighestPriorityTask();
                break;

            case 4:
                // Remove a task by its ID
                cout << "Enter Task ID to remove: ";
                cin >> id;
                removeTaskById(id);
                break;

            case 5:
                // Exit the program
                cout << "Exiting Task Manager.\n";
                return 0;

            default:
                // Handle invalid menu options
                cout << "Invalid choice, please try again.\n";
                break;
        }
    }
}
```

OUTPUT:

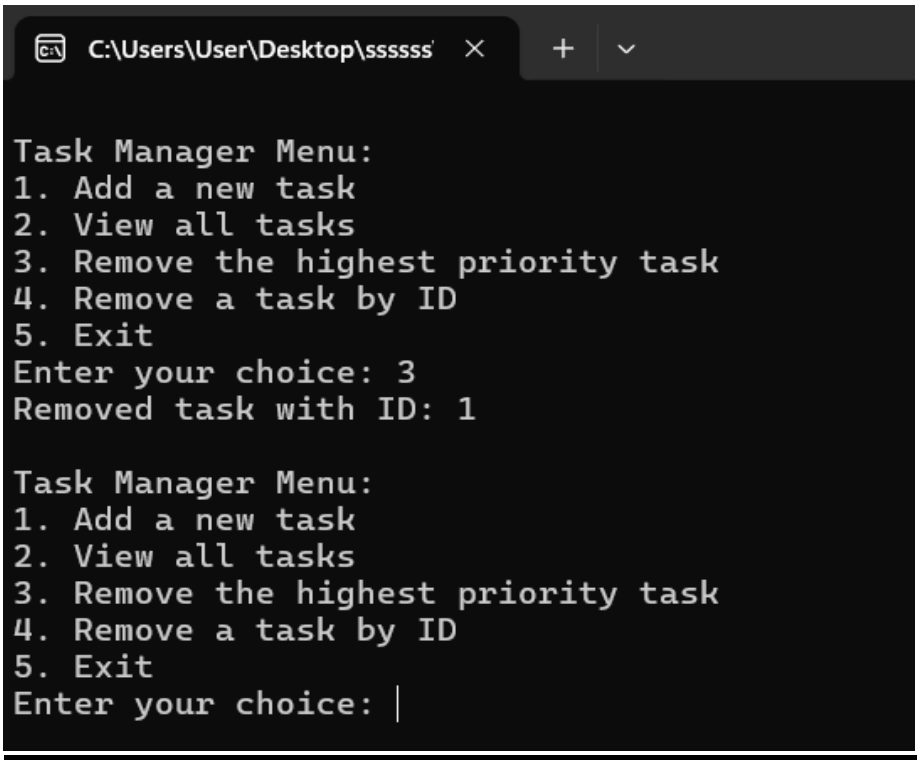
1. Adding a new task:



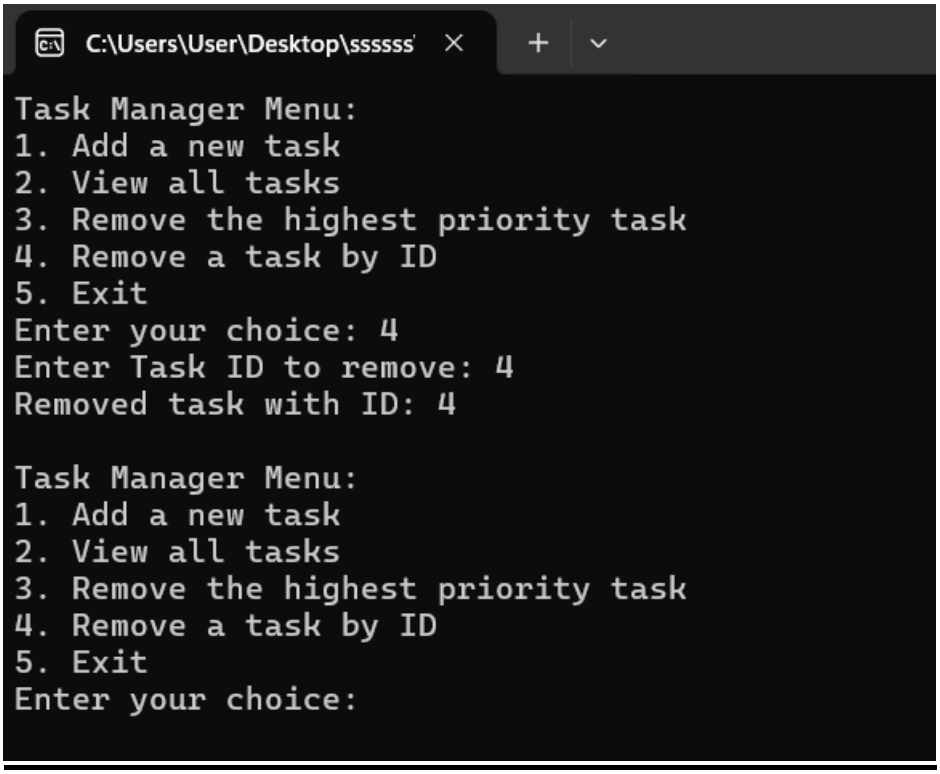
2. Viewing all tasks:



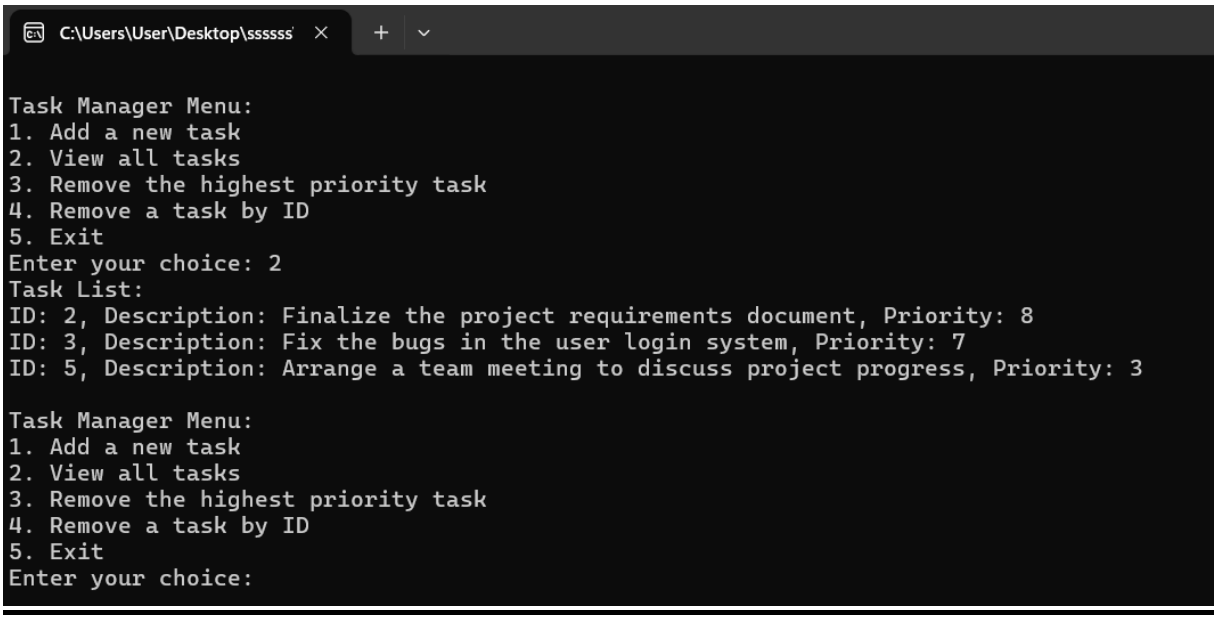
3. Removing the highest priority task:



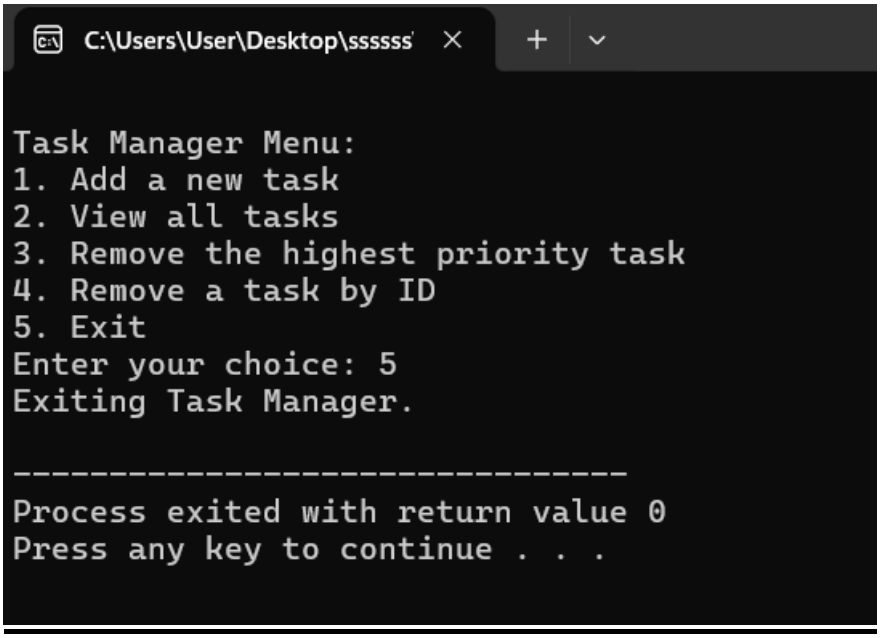
4. Removing task by ID:



5. Viewing all Tasks again after removing:



6. Exiting Task Management System:



Conclusion:

In this project, I learned how to manage tasks using a linked list. The program allows tasks to be added with different priorities, and it organizes them so that higher priority tasks are handled first. I also learned how to remove tasks either by their priority or by their unique ID. This assignment helped me understand the importance of organizing data efficiently and how to work with dynamic memory using pointers. One of the key challenges was making sure the list worked properly when tasks were added or removed, but it helped improve my understanding of linked lists and task management.