

COMSATS UNIVERSITY ISLAMABAD

ATTOCK CAMPUS

NAME: Hamza Ahmad

REG. NO SP23-BSE-024

SUBJECT DS

ASSIGNMENT 1

Date 24th September, 2024

SUBMITTED TO Mr. Muhammad Kamran

Introduction:

The objective of this assignment is to implement a simple task management system using a singly linked list, where each task is represented as a node. The system allows adding tasks based on priority, removing the highest priority task, and removing specific tasks by their ID.

Code Explanation:

TaskNode Struct

- Represents a task in the linked list.
- Contains a unique task ID, a description of the task, a priority level, and a pointer to the next task.

TaskManager Class

- Manages the linked list of tasks.
- Contains a pointer to the head of the list, initialized to `NULL`.

Add Task Function

- Adds a new task based on its priority.
- If the list is empty or the new task has higher priority than the current head, it becomes the new head.
- Otherwise, it finds the correct position to insert the new task, maintaining priority order.

Remove Highest Priority Task

- Removes the task at the head of the list, which has the highest priority.
- If the list is empty, it informs the user.
- Updates the head to the next task and deletes the previous head.

Remove Task by ID

- Removes a specific task using its unique ID.
- If the list is empty, it notifies the user.
- If the task is the head, it updates the head pointer; otherwise, it searches for the task in the list and removes it.

Display Tasks

- Displays all tasks in the list.
- If the list is empty, it informs the user.
- Otherwise, it prints details of each task.

Destructor

- Cleans up memory when a `TaskManager` object is destroyed.
- Continuously removes tasks until the list is empty to prevent memory leaks.

Menu Function

- Provides a console-based interface for user interaction.
- Displays options and calls appropriate functions based on user input.

Main Function

- Entry point of the program.
- Calls the menu function to start the task management system.

CODE:

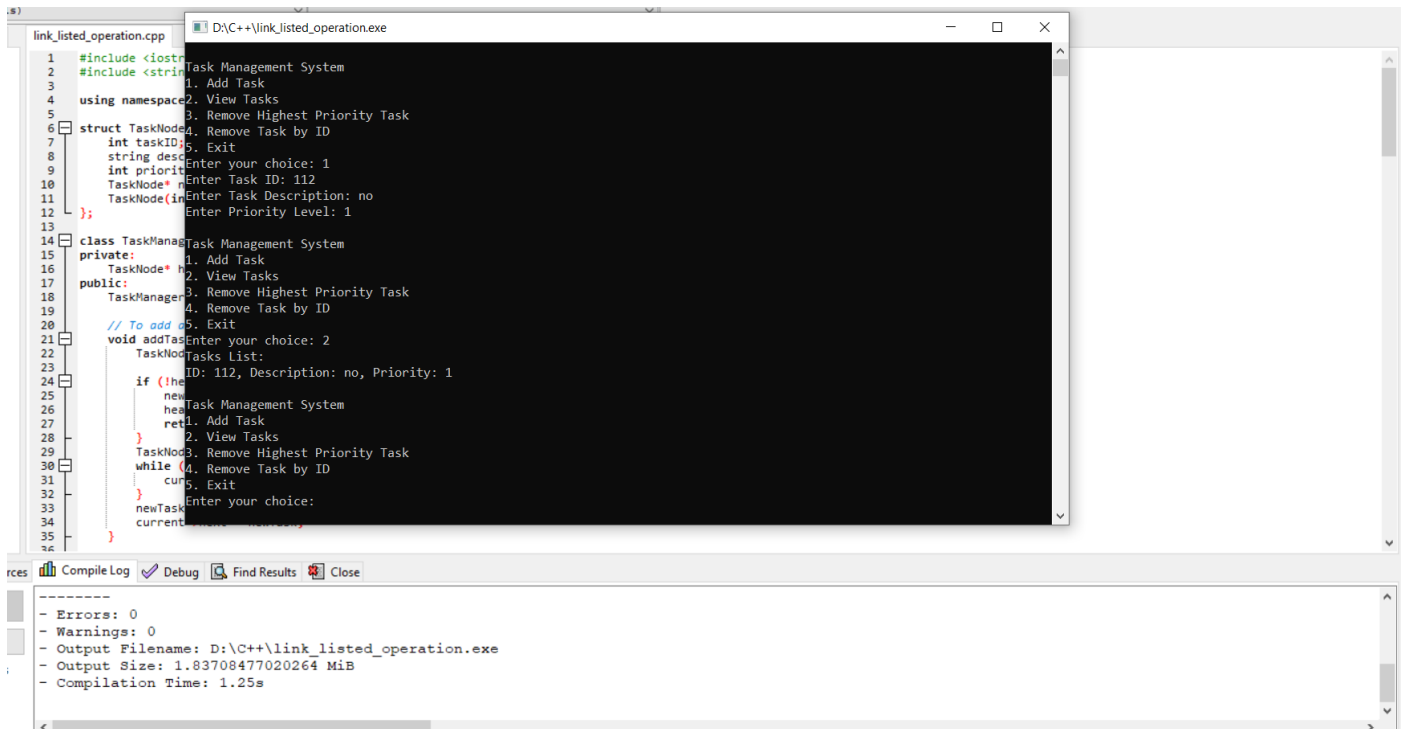
```
[*] link_listed_operation.cpp
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  struct TaskNode {
7      int taskID;
8      string description;
9      int priority;
10     TaskNode* next;
11     TaskNode(int id, string desc, int prio) : taskID(id), description(desc), priority(prio), next(NULL) {}
12 };
13
14 class TaskManager {
15 private:
16     TaskNode* head;
17 public:
18     TaskManager() : head(NULL) {}
19
20     // To add a new task
21     void addTask(int id, string desc, int prio) {
22         TaskNode* newTask = new TaskNode(id, desc, prio);
23
24         if (!head || head->priority < prio) {
25             newTask->next = head;
26             head = newTask;
27             return;
28         }
29         TaskNode* current = head;
30         while (current->next && current->next->priority >= prio) {
31             current = current->next;
32         }
33         newTask->next = current->next;
34         current->next = newTask;
35     }
36
37     // to display all tasks
38     void displayTasks() {
39         if (!head) {
40             cout << "No tasks available." << endl;
41             return;
42         }
43         TaskNode* current = head;
44         cout << "Tasks list:" << endl;
45         while (current) {
46             cout << "ID: " << current->taskID << ", Description: " << current->description
47             | << ", Priority: " << current->priority << endl;
48             current = current->next;
49         }
50     }
51
52     // To remove highest priority task
53     void removeHighestPriorityTask() {
54         if (!head) {
55             cout << "No tasks to remove." << endl;
56             return;
57         }
58         TaskNode* temp = head;
59         head = head->next;
60         delete temp;
61         cout << "Removed highest priority task." << endl;
62     }
63
64     // To remove a task by ID
65     void removeTaskByID(int id) {
66         if (!head) {
67             cout << "No tasks to remove." << endl;
68             return;
69         }
70         // task to remove head
71         if (head->taskID == id) {
72             TaskNode* temp = head;
73             head = head->next;
74             delete temp;
75             cout << "Removed task with ID: " << id << endl;
76             return;
77         }
78         // to find the task with the specified ID
79         TaskNode* current = head;
80         while (current->next && current->next->taskID != id) {
81             current = current->next;
82         }
83         // If the task was found
84         if (current->next) {
85             TaskNode* temp = current->next;
86             current->next = current->next->next;
87             delete temp;
88             cout << "Removed task with ID: " << id << endl;
89         } else {
90             cout << "Task with ID: " << id << " not found." << endl;
91         }
92     }
93 }
```

```

93     }
94 }
95
96 // free memory allocated for the tasks
97 ~TaskManager() {
98     while (head) {
99         removeHighestPriorityTask();
100     }
101 }
102 };
103
104 void displayMenu() {
105     TaskManager taskManager;
106     int choice, id, priority;
107     string description;
108
109     do {
110         cout << "\nTask Management System" << endl;
111         cout << "1. Add Task" << endl;
112         cout << "2. View Tasks" << endl;
113         cout << "3. Remove Highest Priority Task" << endl;
114         cout << "4. Remove Task by ID" << endl;
115         cout << "5. Exit" << endl;
116         cout << "Enter your choice: ";
117         cin >> choice;
118
119         switch (choice) {
120             case 1:
121                 cout << "Enter Task ID: ";
122                 cin >> id;
123                 cout << "Enter Task Description: ";
124                 cin.ignore();
125                 getline(cin, description);
126                 cout << "Enter Priority Level: ";
127                 cin >> priority;
128                 taskManager.addTask(id, description, priority);
129                 break;
130             case 2:
131                 taskManager.displayTasks();
132                 break;
133             case 3:
134                 taskManager.removeHighestPriorityTask();
135                 break;
136             case 4:
137                 cout << "Enter Task ID to remove: ";
138                 cin >> id;
139                 taskManager.removeTaskByID(id);
140                 break;
141             case 5:
142                 cout << "Exiting..." << endl;
143                 break;
144             default:
145                 cout << "Invalid choice, please try again." << endl;
146         }
147     } while (choice != 5);
148 }
149
150 int main() {
151     displayMenu();
152     return 0;
153 }
154
155
156
157
158

```

OUTPUT:



```

link_listed_operation.cpp
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  struct TaskNode {
7      int taskID;
8      string description;
9      int priority;
10     TaskNode* next;
11 };
12
13
14 class TaskManager {
15 private:
16     TaskNode* head;
17 public:
18     TaskManager() { head = nullptr; }
19
20     // To add a task
21     void addTask(int id, string description, int priority) {
22         TaskNode* newNode = new TaskNode(id, description, priority);
23         if (head == nullptr) {
24             head = newNode;
25         } else {
26             TaskNode* temp = head;
27             while (temp->next != nullptr) {
28                 temp = temp->next;
29             }
30             temp->next = newNode;
31         }
32     }
33
34     // To display tasks
35     void displayTasks() {
36         TaskNode* temp = head;
37         while (temp != nullptr) {
38             cout << "Task ID: " << temp->taskID << ", Description: " << temp->description << ", Priority: " << temp->priority << endl;
39             temp = temp->next;
40         }
41     }
42
43     // To remove highest priority task
44     void removeHighestPriorityTask() {
45         TaskNode* temp = head;
46         TaskNode* prev = nullptr;
47         while (temp != nullptr) {
48             if (temp->priority < prev->priority) {
49                 prev = temp;
50             }
51             temp = temp->next;
52         }
53         if (prev != nullptr) {
54             prev->next = temp->next;
55             delete temp;
56         }
57     }
58
59     // To remove task by ID
60     void removeTaskByID(int id) {
61         TaskNode* temp = head;
62         TaskNode* prev = nullptr;
63         while (temp != nullptr) {
64             if (temp->taskID == id) {
65                 if (prev == nullptr) {
66                     head = temp->next;
67                 } else {
68                     prev->next = temp->next;
69                 }
70                 delete temp;
71             }
72             prev = temp;
73             temp = temp->next;
74         }
75     }
76
77     // To exit
78     void exit() {
79         cout << "Exiting..." << endl;
80     }
81 };
82
83 int main() {
84     TaskManager taskManager;
85     int choice, id, priority;
86     string description;
87
88     do {
89         cout << "\nTask Management System" << endl;
90         cout << "1. Add Task" << endl;
91         cout << "2. View Tasks" << endl;
92         cout << "3. Remove Highest Priority Task" << endl;
93         cout << "4. Remove Task by ID" << endl;
94         cout << "5. Exit" << endl;
95         cout << "Enter your choice: ";
96         cin >> choice;
97
98         switch (choice) {
99             case 1:
100                 cout << "Enter Task ID: ";
101                 cin >> id;
102                 cout << "Enter Task Description: ";
103                 cin.ignore();
104                 getline(cin, description);
105                 cout << "Enter Priority Level: ";
106                 cin >> priority;
107                 taskManager.addTask(id, description, priority);
108                 break;
109             case 2:
110                 taskManager.displayTasks();
111                 break;
112             case 3:
113                 taskManager.removeHighestPriorityTask();
114                 break;
115             case 4:
116                 cout << "Enter Task ID to remove: ";
117                 cin >> id;
118                 taskManager.removeTaskByID(id);
119                 break;
120             case 5:
121                 cout << "Exiting..." << endl;
122                 break;
123             default:
124                 cout << "Invalid choice, please try again." << endl;
125         }
126     } while (choice != 5);
127 }

```

```

-----
- Errors: 0
- Warnings: 0
- Output Filename: D:\C++\link_listed_operation.exe
- Output Size: 1.83708477020264 MiB
- Compilation Time: 1.25s

```