



COMSATS UNIVERSITY ISLAMABAD, ATTOCK CAMPUS

DEPARTMENT OF COMPUTER SCIENCE

Program: BS-SE

Course: Data Structure and Algorithm

Name: Hana Siddiquie

Registration Number: SP23-BSE-016

Date: 24th September, 2024

Assignment: 01

Submitted to: Sir Kamran

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;
    } while (choice != 5);
}

```

```
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 objective of this assignment is to create a task management system using a singly linked list, where each task is a node containing a unique task ID, description, and priority. The system allows users to add tasks in descending order of priority, ensuring that higher priority tasks are placed first in the list. Additionally, the program enables removing the highest priority task (from the head of the list), removing specific tasks by their ID, and viewing all tasks. This project highlights how linked lists efficiently manage tasks and their priorities dynamically.

Code Explanation:

1. createTask Function:

This function creates a new task node by dynamically allocating memory for it. It sets the “taskID”, “description”, and “priority” of the task and initializes the “next” pointer to “nullptr”. This function returns a pointer to the newly created task, which will later be inserted into the linked list.

2. addTask Function:

This function inserts a new task into the linked list in the correct position based on its priority. If the list is empty or the new task has a higher priority than the head of the list, it becomes the new head. Otherwise, the function traverses the list to find the appropriate position for the task, ensuring that the list remains sorted in descending order of priority. The new task is then inserted in the correct position.

3. removeHighestPriorityTask Function:

This function removes the task at the head of the list, which is always the task with the highest priority. It checks if the list is empty and if not, it deletes the head node and updates the head pointer to point to the next task in the list. The task with the highest priority is then removed, and memory is freed.

4. removeTaskByID Function:

This function removes a task from the list by its unique “taskID”. If the task to be removed is at the head of the list, the head is updated to the next task. If the task is located further down the list, the function traverses the list, finds the task by its ID, and updates the previous node's `next` pointer to skip the node being removed. The memory for the removed task is freed.

5. displayTasks Function:

This function traverses the entire linked list and prints out the details of each task (ID, description, and priority). If the list is empty, it informs the user that no tasks are available. Otherwise, it iterates through each node and displays the task information.

6. Destructor (“TaskManager”):

The destructor is responsible for cleaning up the dynamically allocated memory when the program ends. It traverses the linked list and deletes each node (task) to ensure there are no memory leaks after the program terminates.

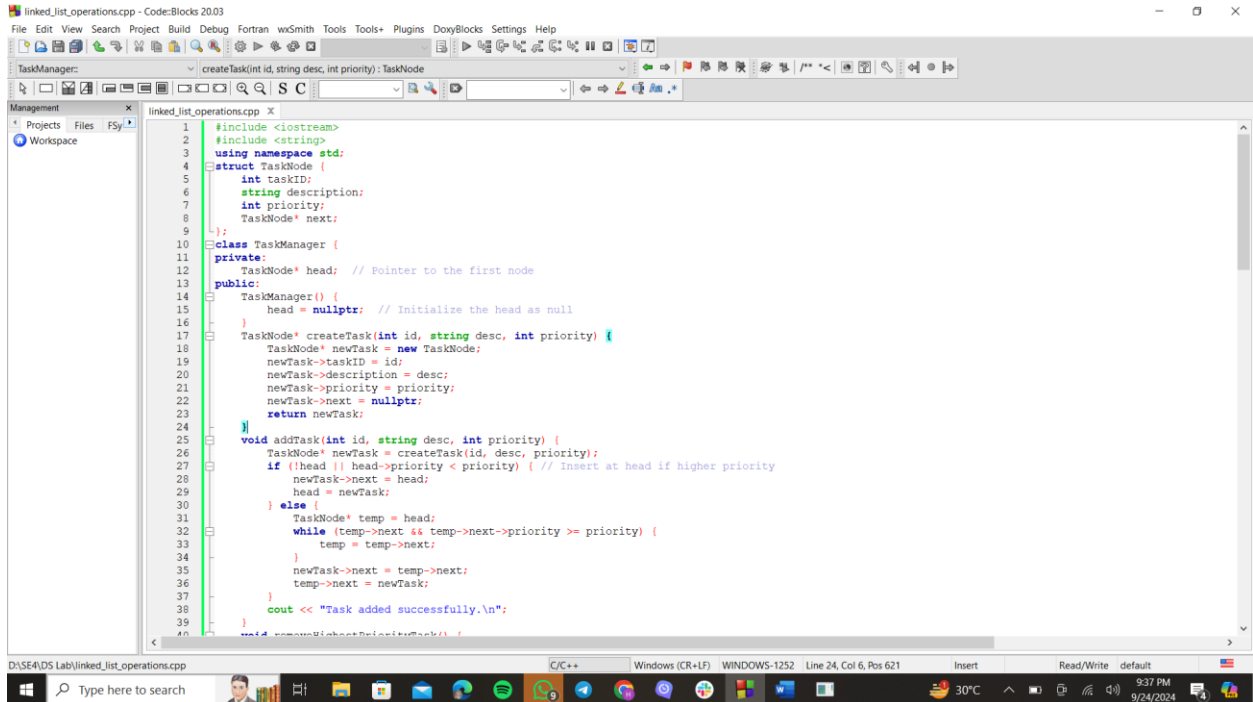
7. Main Function:

The “main()” function presents a console-based menu that allows users to interact with the task management system. Users can add new tasks, view the current tasks, remove the highest priority task, or remove a task by its ID. The menu loops until the user chooses to exit the program, ensuring continuous interaction with the task list.

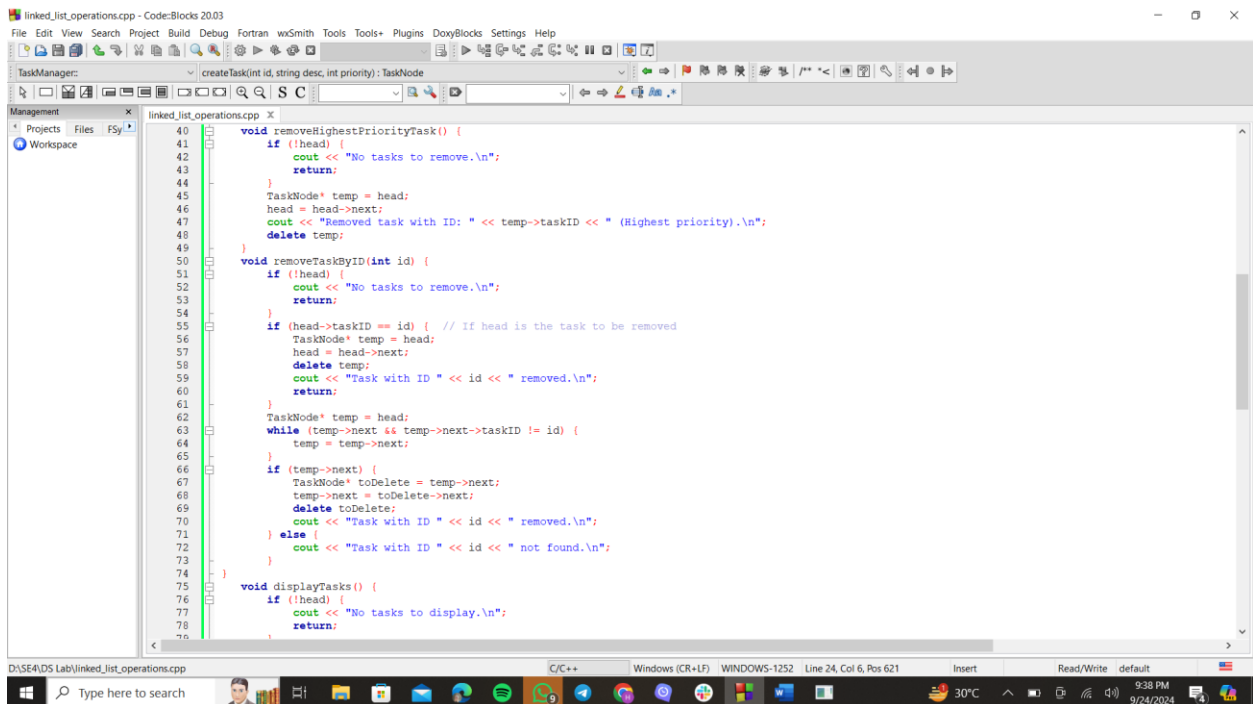
Conclusion:

Through this assignment, I gained a deeper understanding of how singly linked lists can be used to dynamically manage data in real-time applications, such as a task management system. I learned how to implement core operations like adding, removing, and traversing nodes in a linked list while maintaining the correct order based on task priority. The main challenge I faced was ensuring that tasks were inserted at the appropriate position in the list to preserve the priority order. Additionally, managing the removal of specific nodes, especially at the head of the list, required careful pointer manipulation to avoid memory leaks. Overall, this assignment enhanced my problem-solving skills in managing dynamic data structures and improved my ability to write clean and efficient code in C++.

Screenshots:



```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4 struct TaskNode {
5     int taskID;
6     string description;
7     int priority;
8     TaskNode* next;
9 };
10 class TaskManager {
11 private:
12     TaskNode* head; // Pointer to the first node
13 public:
14     TaskManager() {
15         head = nullptr; // Initialize the head as null
16     }
17     TaskNode* createTask(int id, string desc, int priority) {
18         TaskNode* newTask = new TaskNode;
19         newTask->taskID = id;
20         newTask->description = desc;
21         newTask->priority = priority;
22         newTask->next = nullptr;
23         return newTask;
24     }
25     void addTask(int id, string desc, int priority) {
26         TaskNode* newTask = createTask(id, desc, priority);
27         if (!head || head->priority < priority) { // Insert at head if higher priority
28             newTask->next = head;
29             head = newTask;
30         } else {
31             TaskNode* temp = head;
32             while (temp->next && temp->next->priority >= priority) {
33                 temp = temp->next;
34             }
35             newTask->next = temp->next;
36             temp->next = newTask;
37         }
38         cout << "Task added successfully.\n";
39     }
40     void removeHighestPriorityTask() {
```



```
40     void removeHighestPriorityTask() {
41         if (!head) {
42             cout << "No tasks to remove.\n";
43             return;
44         }
45         TaskNode* temp = head;
46         head = head->next;
47         cout << "Removed task with ID: " << temp->taskID << " (Highest priority).\n";
48         delete temp;
49     }
50     void removeTaskByID(int id) {
51         if (!head) {
52             cout << "No tasks to remove.\n";
53             return;
54         }
55         if (head->taskID == id) { // If head is the task to be removed
56             TaskNode* temp = head;
57             head = head->next;
58             delete temp;
59             cout << "Task with ID " << id << " removed.\n";
60             return;
61         }
62         TaskNode* temp = head;
63         while (temp->next && temp->next->taskID != id) {
64             temp = temp->next;
65         }
66         if (temp->next) {
67             TaskNode* toDelete = temp->next;
68             temp->next = toDelete->next;
69             delete toDelete;
70             cout << "Task with ID " << id << " removed.\n";
71         } else {
72             cout << "Task with ID " << id << " not found.\n";
73         }
74     }
75     void displayTasks() {
76         if (!head) {
77             cout << "No tasks to display.\n";
78             return;
79         }
80     }
```

linked_list_operations.cpp - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

TaskManager: createTask(int id, string desc, int priority): TaskNode

Management

Projects Files FSy Workspace

```
76 if (!head) {
77     cout << "No tasks to display.\n";
78     return;
79 }
80 TaskNode* temp = head;
81 while (temp) {
82     cout << "Task ID: " << temp->taskID << ", Description: " << temp->description
83         << ", Priority: " << temp->priority << "\n";
84     temp = temp->next;
85 }
86
87 ~TaskManager() {
88     while (head) {
89         TaskNode* temp = head;
90         head = head->next;
91         delete temp;
92     }
93 }
94
95 int main() {
96     TaskManager manager;
97     int choice, id, priority;
98     string description;
99     do {
100         cout << "\nTask Manager Menu:\n";
101         cout << "1. Add a new task\n";
102         cout << "2. View all tasks\n";
103         cout << "3. Remove the highest priority task\n";
104         cout << "4. Remove a task by ID\n";
105         cout << "5. Exit\n";
106         cout << "Enter your choice: ";
107         cin >> choice;
108         switch (choice) {
109             case 1:
110                 cout << "Enter task ID: ";
111                 cin >> id;
112                 cout << "Enter task description: ";
113                 cin.ignore(); // To ignore the newline character left by previous input
114                 getline(cin, description);
115                 cout << "Enter task priority: ";
```

D:\SE4\DS Lab\linked_list_operations.cpp C/C++ Windows (CR+LF) WINDOWS-1252 Line 24, Col 6, Pos 621 Insert Read/Write default 9:39 PM 9/24/2024

linked_list_operations.cpp - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

TaskManager: createTask(int id, string desc, int priority): TaskNode

Management

Projects Files FSy Workspace

```
102 cout << "2. View all tasks\n";
103 cout << "3. Remove the highest priority task\n";
104 cout << "4. Remove a task by ID\n";
105 cout << "5. Exit\n";
106 cout << "Enter your choice: ";
107 cin >> choice;
108 switch (choice) {
109     case 1:
110         cout << "Enter task ID: ";
111         cin >> id;
112         cout << "Enter task description: ";
113         cin.ignore(); // To ignore the newline character left by previous input
114         getline(cin, description);
115         cout << "Enter task priority: ";
116         cin >> priority;
117         manager.addTask(id, description, priority);
118         break;
119     case 2:
120         cout << "All Tasks:\n";
121         manager.displayTasks();
122         break;
123     case 3:
124         manager.removeHighestPriorityTask();
125         break;
126     case 4:
127         cout << "Enter task ID to remove: ";
128         cin >> id;
129         manager.removeTaskByID(id);
130         break;
131     case 5:
132         cout << "Exiting Task Manager.\n";
133         break;
134     default:
135         cout << "Invalid choice. Try again.\n";
136 }
137 } while (choice != 5);
138 return 0;
139 }
140 }
```

D:\SE4\DS Lab\linked_list_operations.cpp C/C++ Windows (CR+LF) WINDOWS-1252 Line 24, Col 6, Pos 621 Insert Read/Write default 9:40 PM 9/24/2024

Output:

```
"D:\SEA\DS Lab\linked_list_operations.exe"
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: 1
Enter task ID: 101
Enter task description: Complete Assignment
Enter task priority: 4
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 Assignment, Priority: 4

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: 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: 4
Enter task ID to remove: 101
No tasks to remove.

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: 5
Exiting Task Manager.
```