DS ASSIGNMENT 1

SP23-BSE-066

SEPTEMBER 24, 2024

HUZAIFA RIZWAN

QUESTION :

Implement the code for the assignment in a single C++ file (linked_list_operations.cpp). Ensure the code is well-documented with comments explaining the logic and functionality of each function.

SOLUTION :

Task Management System using Linked List:

Overview:

This system manages tasks using a singly linked list. Each task has an ID, description, and priority. Tasks are inserted in order of priority, with higher priority tasks at the beginning.

Key Operations:

1.**Create Task:** Creates a new task node with the given ID, description, and priority.
2.**Insert Task:** Inserts the new task into the list based on its priority. Higher priority tasks go first.

3.**Remove Highest Priority Task:** Removes the task at the beginning of the list (highest priority).

4.**Remove Task by ID:** Finds and removes a specific task based on its ID.

5.**View Tasks:** Displays all tasks in the list.

Code Structure:

• **Task Structure:** Defines the properties of a task (ID, description, priority, and a pointer to the next task).

• **Functions:**

   o createTask: Creates a new task node. o insertTask: Inserts a task into the list based on priority.

   o removeHighestPriorityTask: Removes the first node (highest priority). o   removeTaskById: Searches for and removes a task by ID.

   o viewTasks: Displays all tasks in the list.

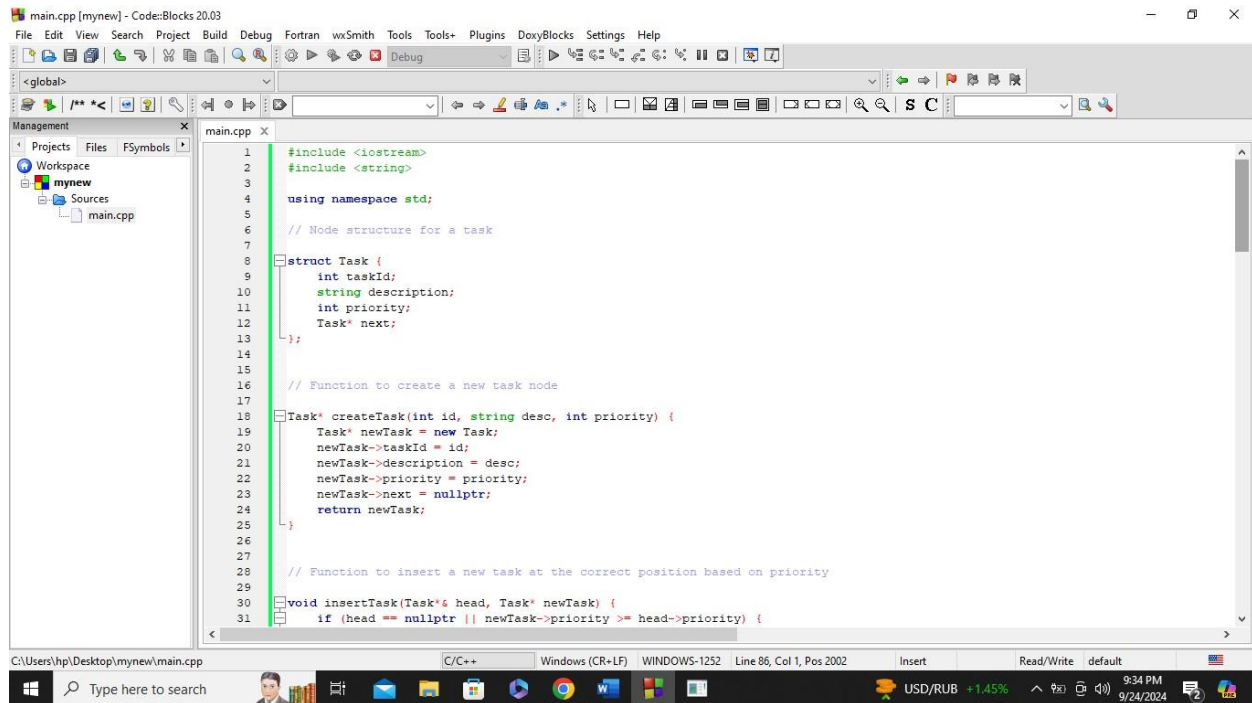• **Main Function:** Handles user input and calls the appropriate functions.

<u>Example Usage:</u>

1. User adds a task with a high priority.

2. User adds another task with a lower priority.

3. System displays all tasks, showing the higher priority task first.

4. User removes the highest priority task.

5. System displays the remaining task.

<u>Benefits of Linked Lists:</u>

- Efficient insertion and deletion at any position.

- Dynamic size.

- No need to know the size in advance.

<u>SCREENSHOTS:</u>

<u>CODE;</u>

main.cpp

```cpp
#include <iostream>
#include <string>

using namespace std;

// Node structure for a task

struct Task {
    int taskId;
    string description;
    int priority;
    Task* next;
};


// Function to create a new task node

Task* createTask(int id, string desc, int priority) {
    Task* newTask = new Task;
    newTask->taskId = id;
    newTask->description = desc;
    newTask->priority = priority;
    newTask->next = nullptr;
    return newTask;
}


// Function to insert a new task at the correct position based on priority

void insertTask(Task*& head, Task* newTask) {
    if (head == nullptr || newTask->priority >= head->priority) {
```

```cpp
23          newTask->next = nullptr;
24          return newTask;
25    }
26
27
28      // Function to insert a new task at the correct position based on priority
29
30      void insertTask(Task*& head, Task* newTask) {
31          if (head == nullptr || newTask->priority >= head->priority) {
32              newTask->next = head;
33              head = newTask;
34          } else {
35              Task* current = head;
36              while (current->next != nullptr && current->next->priority > newTask->priority) {
37                  current = current->next;
38              }
39              newTask->next = current->next;
40              current->next = newTask;
41          }
42    }
43
44
45      // Function to remove the task with the highest priority
46
47      void removeHighestPriorityTask(Task*& head) {
48          if (head == nullptr) {
49              cout << "List is empty." << endl;
50              return;
51          }
52          Task* temp = head;
53          head = head->next;
```

```cpp
41          }
42    }
43
44
45      // Function to remove the task with the highest priority
46
47      void removeHighestPriorityTask(Task*& head) {
48          if (head == nullptr) {
49              cout << "List is empty." << endl;
50              return;
51          }
52          Task* temp = head;
53          head = head->next;
54          delete temp;
55    }
56
57
58      // Function to remove a task by its ID
59
60      void removeTaskById(Task*& head, int taskId) {
61          if (head == nullptr) {
62              cout << "List is empty." << endl;
63              return;
64          }
65          if (head->taskId == taskId) {
66              Task* temp = head;
67              head = head->next;
68              delete temp;
69              return;
70          }
71          Task* current = head;
```

```cpp
63              return;
64          }
65          if (head->taskId == taskId) {
66              Task* temp = head;
67              head = head->next;
68              delete temp;
69              return;
70          }
71          Task* current = head;
72          while (current->next != nullptr && current->next->taskId != taskId) {
73              current = current->next;
74          }
75          if (current->next != nullptr) {
76              Task* temp = current->next;
77              current->next = temp->next;
78              delete temp;
79          } else {
80              cout << "Task with ID " << taskId << " not found." << endl;
81          }
82      }
83
84
85      // Function to view all tasks
86
87      void viewTasks(Task* head) {
88          if (head == nullptr) {
89              cout << "List is empty." << endl;
90              return;
91          }
92          Task* current = head;
93          while (current != nullptr) {
```

```cpp
81          }
82      }
83
84
85      // Function to view all tasks
86
87      void viewTasks(Task* head) {
88          if (head == nullptr) {
89              cout << "List is empty." << endl;
90              return;
91          }
92          Task* current = head;
93          while (current != nullptr) {
94              cout << "Task ID: " << current->taskId << endl;
95              cout << "Description: " << current->description << endl;
96              cout << "Priority: " << current->priority << endl;
97              cout << endl;
98              current = current->next;
99          }
100     }
101
102     int main() {
103         Task* head = nullptr;
104         int choice;
105
106         while (true) {
107             cout << "1. Add a new task" << endl;
108             cout << "2. View all tasks" << endl;
109             cout << "3. Remove the highest priority task" << endl;
110             cout << "4. Remove a task by ID" << endl;
111             cout << "5. Exit" << endl;
```

```cpp
101
102  int main() {
103      Task* head = nullptr;
104      int choice;
105
106      while (true) {
107          cout << "1. Add a new task" << endl;
108          cout << "2. View all tasks" << endl;
109          cout << "3. Remove the highest priority task" << endl;
110          cout << "4. Remove a task by ID" << endl;
111          cout << "5. Exit" << endl;
112          cout << "Enter your choice: ";
113          cin >> choice;
114
115          switch (choice) {
116              case 1: {
117                  int id, priority;
118                  string description;
119                  cout << "Enter task ID: ";
120                  cin >> id;
121                  cout << "Enter description: ";
122                  cin.ignore();
123                  getline(cin, description);
124                  cout << "Enter priority: ";
125                  cin >> priority;
126                  Task* newTask = createTask(id, description, priority);
127                  insertTask(head, newTask);
128                  break;
129              }
130              case 2:
131                  viewTasks(head);
```

C:\Users\hp\Desktop\mynew\main.cpp   C/C++   Windows (CR+LF)   WINDOWS-1252   Line 86, Col 1, Pos 2002   Insert   Read/Write   default

---

```cpp
115          switch (choice) {
116              case 1: {
117                  int id, priority;
118                  string description;
119                  cout << "Enter task ID: ";
120                  cin >> id;
121                  cout << "Enter description: ";
122                  cin.ignore();
123                  getline(cin, description);
124                  cout << "Enter priority: ";
125                  cin >> priority;
126                  Task* newTask = createTask(id, description, priority);
127                  insertTask(head, newTask);
128                  break;
129              }
130              case 2:
131                  viewTasks(head);
132                  break;
133              case 3:
134                  removeHighestPriorityTask(head);
135                  break;
136              case 4: {
137                  int taskId;
138                  cout << "Enter task ID to remove: ";
139                  cin >> taskId;
140                  removeTaskById(head, taskId);
141                  break;
142              }
143              case 5:
144                  cout << "Exiting..." << endl;
145                  return 0;
```
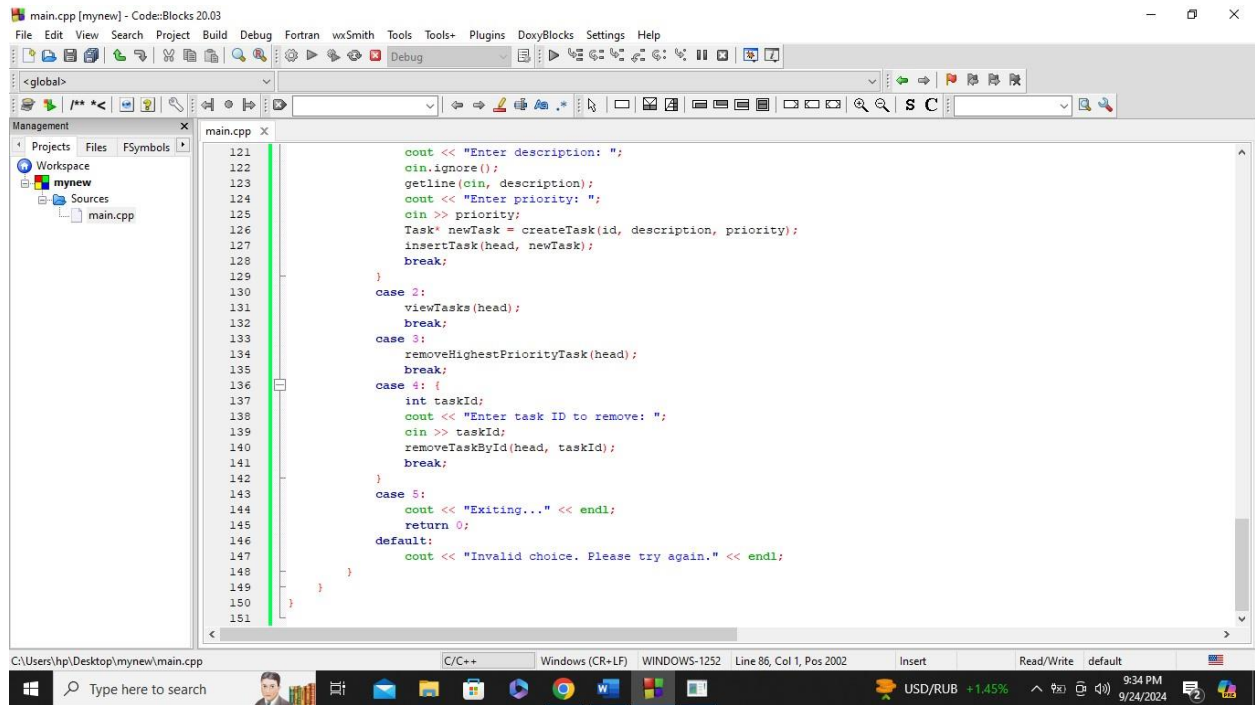
C:\Users\hp\Desktop\mynew\main.cpp   C/C++   Windows (CR+LF)   WINDOWS-1252   Line 86, Col 1, Pos 2002   Insert   Read/Write   default

```cpp
                cout << "Enter description: ";
                cin.ignore();
                getline(cin, description);
                cout << "Enter priority: ";
                cin >> priority;
                Task* newTask = createTask(id, description, priority);
                insertTask(head, newTask);
                break;
            }
            case 2:
                viewTasks(head);
                break;
            case 3:
                removeHighestPriorityTask(head);
                break;
            case 4: {
                int taskId;
                cout << "Enter task ID to remove: ";
                cin >> taskId;
                removeTaskById(head, taskId);
                break;
            }
            case 5:
                cout << "Exiting..." << endl;
                return 0;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    }
}
```

OUTPUT:


```
C:\Users\hp\Desktop\mynew\bin\Debug\mynew.exe                                    —  □  ✕
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: 111
Enter description: 2222
Enter priority: 1
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
Task ID: 111
Description: 2222
Priority: 1

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
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
List is empty.
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: 3
List is empty.
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
```



```
C:\Users\hp\Desktop\mynew\bin\Debug\mynew.exe                                    —  □  ✕
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 3
List is empty.
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: 3
List is empty.
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...

Process returned 0 (0x0)   execution time : 58.259 s
Press any key to continue.
```

Conclusion:

This task management system effectively utilizes a singly linked list to store and manipulate tasks. The system allows users to add, view, and remove tasks based on their priority, providing a flexible and efficient solution for organizing and managing tasks.