COMSATS UNIVERSITY ISLAMABAD ATTOCK CAMPUS

PROGRAM BSSE

NAME: **VAREESHA NISA**

REG NO: **SP23-BSE-021**

DATE: **24 SEP 2024**

SUBMITTED TO: **SIR KAMRAN**

ASSIGNMENT NO: **01(Theory)**

Defining Structure for Task Node:

linked_list_operations.cpp

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

using namespace std;

// Define the structure for a task node
struct NODE {
    int taskID;
    string description;
    int priority;
    NODE* next;
};
```

Function To Create A New Task Node:

```cpp
// Function to create a new task node
NODE* createTaskNode(int id, string desc, int priority) {
    NODE* newTaskNode = new NODE();
    newTaskNode->taskID = id;
    newTaskNode->description = desc;
    newTaskNode->priority = priority;
    newTaskNode->next = NULL;
    return newTaskNode;
}
```

Function To Add A Task To The List Based On Priority:

```
// Function to add a task to the list based on priority
void addTaskNode(NODE*& head, int id, string desc, int priority) {
    NODE* newTaskNode = createTaskNode(id, desc, priority);
    if (!head || head->priority < priority) {
        newTaskNode->next = head;
        head = newTaskNode;
    } else {
        NODE* current = head;
        while (current->next && current->next->priority >= priority) {
            current = current->next;
        }
        newTaskNode->next = current->next;
        current->next = newTaskNode;
    }
}
```

Function To Remove The Task With The Highest Priority:

```
// Function to remove the task with the highest priority
void removeHighestPriorityTaskNode(NODE*& head) {
    if (head) {
        NODE* temp = head;
        head = head->next;
        delete temp;
        cout << "Highest priority task removed." << endl;
    } else {
        cout << "No tasks to remove." << endl;
    }}
```

Function To Remove A Specific Task By ID:

```
// Function to remove a specific task by ID
void removeTaskNodeByID(NODE*& head, int id) {
    if (!head) {
        cout << "No tasks to remove." << endl;
        return;}
    if (head->taskID == id) {
        NODE* temp = head;
        head = head->next;
        delete temp;
        return; }
    NODE* current = head;
    while (current->next && current->next->taskID != id) {
        current = current->next;}
    if (current->next) {
        NODE* temp = current->next;
        current->next = current->next->next;
        delete temp;
    } else {
        cout << "Task ID not found." << endl;}}
```

Function To View All Tasks:

```
// Function to view all tasks
void viewTaskNodes(NODE* head) {
    if (!head) {
        cout << "No tasks available." << endl;
        return;
    }
    NODE* current = head;
    while (current) {
        cout << "Task ID: " << current->taskID << ", Description: " << current->description << "
        , Priority: " << current->priority << endl;
        current = current->next;
    }
}
```

Function To Display The Menu:

```
// Function to display the menu
void displayMenu() {
    cout << "1. Add a new task" << endl;
    cout << "2. View all tasks" << endl;
    cout << "3. Remove the highest priority task" << endl;
    cout << "4. Remove a task by ID" << endl;
    cout << "5. Exit" << endl;
}
```

Main :

```
int main() {
    NODE* head = NULL;
    int choice, id, priority;
    string description;
 while (true) {
        displayMenu();
        cout << "Enter your choice: ";
        cin >> choice;
    switch (choice) {
            case 1:
                cout << "Enter task ID: ";
                cin >> id;
                cin.ignore(); // Ignore newline character
                cout << "Enter task description: ";
                getline(cin, description);
                cout << "Enter task priority: ";
                cin >> priority;
                addTaskNode(head, id, description, priority);
                break;
            case 2:
                viewTaskNodes(head);
                break;
            case 3:
                removeHighestPriorityTaskNode(head);
                break;
            case 4:
                cout << "Enter task ID to remove: ";
                cin >> id;
                removeTaskNodeByID(head, id);
                break;
            case 5:
                cout << "Exiting..." << endl;
                return 0;
            default:
                cout << "Invalid choice. Please try again." << endl;
        } }
    return 0;}
```

Output

```
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: 12
Enter task description: task management system
Enter task priority: 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: 1
Enter task ID: 13
Enter task description: assignment
Enter task priority: 4
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: 13
Enter task description: sdsf
Enter task priority: 5
1. Add a new task
2. View all tasks
3. Remove the highest priority task
```

```
Enter your choice: 1
Enter task ID: 13
Enter task description: sdsf
Enter task priority: 5
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: 15
Enter task description: yjy
Enter task priority: 7
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: 15, Description: yjy, Priority: 7
Task ID: 13, Description: sdsf, Priority: 5
Task ID: 13, Description: assignment, Priority: 4
Task ID: 12, Description: task management system, Priority: 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
Highest priority task removed.
```

```
4. Remove a task by ID
5. Exit
Enter your choice: 3
Highest priority task removed.
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: 13
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: 13, Description: assignment, Priority: 4
Task ID: 12, Description: task management system, Priority: 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: 5
Exiting...

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Introduction

The objective of this assignment is to implement a task management system using a singly linked list in C++. This system allows the user to perform several operations on tasks, including adding tasks with a priority, viewing all tasks, removing the highest-priority task, and deleting a specific task by its ID. Each task is represented as a node in the linked list, with fields for task ID, description, and priority. The system organizes the tasks based on their priority, ensuring that higher-priority tasks appear earlier in the list.

Code Explanation

NODE Structure

The NODE structure is used to represent each task. It contains the following fields:

taskID: An integer representing the unique identifier of the task.

description: A string describing the task.

priority: An integer indicating the priority of the task.

next: A pointer to the next task in the linked list.

Create task node Function.

This function creates a new task node. It takes the task ID, description, and priority as parameters and returns a pointer to the newly created node. The node's next pointer is initialized to NULL because it is not connected to any other node upon creation.

Add task node Function.

This function adds a new task node to the list, keeping the tasks sorted based on priority. The logic checks if the head is empty or if the priority of the new node is higher than the current head. If so, the new node becomes the new head. Otherwise, the function iterates through the list to find the correct position where the new node's priority fits. The list is sorted in descending order of priority, so tasks with higher priorities appear first.

Remove the highest priority task node Function.

This function removes the task with the highest priority, which is always the first node in the list. If the list is empty, it prints a message stating that there are no tasks to remove. Otherwise, it removes the head node and sets the next node as the new head.

Remove task node by id Function.

This function allows the user to remove a specific task by providing its task ID. It first checks if the head matches the ID. If not, it traverses the list to find the node with the given ID. If the node is found, it is deleted, and the list is updated. If the task ID is not found, a message is displayed.

View task nodes Function

This function displays all the tasks in the list. It iterates through the linked list, printing each task's ID, description, and priority. If the list is empty, it outputs a message indicating that there are no tasks available.

Display menu Function

This function presents a menu with five options:

- Add a new task.
- View all tasks.
- Remove the highest priority task.
- Remove a task by its ID.
- Exit the program.

Main Function

The main function controls the flow of the program. It displays the menu and processes the user's choice. Based on the user's input, it calls the appropriate function (e.g., adding a new task, viewing tasks, removing the highest-priority task, or removing a task by ID). The program continues running until the user chooses to exit.

Conclusion

This assignment taught me how to implement a single linked list and manage dynamic memory in C++. I gained experience designing an application that involves inserting, deleting, and displaying nodes, all while maintaining the order of elements based on priority. The main challenge was ensuring that the list remained sorted by priority after each insertion and deletion.