COMSATS University Islamabad, Attock campus

Program: BSE

Name:	Maria Noor
Registration #:	SP23-BSE-009
Course:	DS
Date:	23-09-2024
Assignment #:	01(Theory)
Submitted To:	Sir Muhammad Kamran

Introduction:

The objective of my assignment is to create a simple **Task Manager** using a **single linked list** to store tasks. Each task has a unique ID, description, and priority in it. The tasks are dynamically managed in the list based on their priority. The operations implemented include:

1. Adding a task at the correct position in the list based on priority.
2. Viewing all tasks in the list.
3. Removing the highest priority task.
4. Removing a task by its ID.
5. Main Method.

Code Explanation:

1. **Structure:** The structure defines the task id, description, priority, and a pointer to the next task in the linked list.

2. **Create Task function:** This function creates a new task and assigns the task ID, description, and priority, also initializes the next pointer to NULL.

3. **Adding a new Task function:** This function adds a new task based on its priority. If the head is null, the new task becomes the head. If the new task's priority is higher than the current head, it becomes the new head. Otherwise, the list is traversed, and the task is inserted where its priority fits.

4. **View all Tasks function:** Displays all tasks in the list. If the list is empty, it tells the user. Otherwise, it iterates through the list and prints the details of each task.

5. **Remove the Highest Priority Task function:** Removes the task with the highest priority. If the list is empty, it tells the user. Otherwise, it deletes the task at the head and moves to the head pointer to the next task.

6. **Remove the Task by Id function:** Removes a task by its unique ID. It searches through the list to find the task with the given ID, adjusts pointers, and deletes the task.

7. **Main Function:** It presents a task menu with options for adding, viewing, and removing tasks. Based on user input, it calls the function.

Screenshots:

TDM-GCC 4.9.2 64-bit Release

(globals)

[*] linked_list_operations.cpp

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

using namespace std;

struct Task {
    // unique task id
    int id;
    // task description
    string description;
    // task priority
    int priority;
    Task* next;       // Pointer to the next task in the linked list
};

// Function to create a new task
Task* createTask(int id, string description, int priority) {
    //  memory allocated(dynamic) for a new task
    Task* newTask = new Task;
    // assign a task ID
    newTask->id = id;
    // assign the task description
    newTask->description = description;
    // assign the task priority
    newTask->priority = priority;
    //  next pointer to null (i.e.,end of the list for now)
    newTask->next = NULL;
    // return the pointer to the newly task that is created
    return newTask;
}
```

Compiler | Resources | Compile Log | Debug | Find Results

C:\Users\ATECH\Desktop\A.Document\linked_list_operations.cpp - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project [*] linked_list_operations.cpp

```cpp

// function to add a new task
void addTask(Task*& head, int id, string description, int priority) {
    Task* newTask = createTask(id, description, priority);
    // check if the list is empty or if the new task has higher priority than the current head
    if (head == NULL || head->priority < priority) {

     // insert the new task before the current head
        newTask->next = head;
        head = newTask;        // update head to point to the new task
    } else {
        // start from the head to find the correct insertion point
        Task* current = head;
        // traverse the list until you find the correct position for the new task based on priority
        while (current->next != NULL && current->next->priority >= priority) {
            current = current->next;
        }
        // insert the new task in the correct position
        newTask->next = current->next;
        current->next = newTask;
    }
    cout << "Task added successfully.\n";
}

// Function to remove the task with the highest priority
void removeHighestPriorityTask(Task*& head) {
    // if the list is empty
    if (head == NULL) {
        cout << "No tasks available to remove.\n";
        return;
```

Compiler Resources Compile Log Debug Find Results

C:\Users\ATECH\Desktop\A.Document\linked_list_operations.cpp - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project [*] linked_list_operations.cpp

```cpp
    if (head == NULL) {
        cout << "No tasks available to remove.\n";
        return;
    }
    // remove the head (highest priority task) and update the head to the next task
    Task* temp = head;
    head = head->next;
    cout << "Removed task with ID: " << temp->id << "\n";
    // free the memory allocated for the removed task
    delete temp;
}

// Function to remove a specific task by its ID
void removeTaskById(Task*& head, int id) {
    // if the list is empty
    if (head == NULL) {
        cout << "No tasks available.\n";
        return;
    }
    // if the task to be removed is the head, update the head
    if (head->id == id) {
        Task* temp = head;
        head = head->next;
        cout << "Removed task with ID: " << temp->id << "\n";
        delete temp;
        return;
    }
    // Traverse the list to find the task to be removed
    Task* current = head;
    while (current->next != NULL && current->next->id != id) {
```

Compiler Resources Compile Log Debug Find Results

Line: 55 Col: 57 Sel: 0 Lines: 180 Length: 5831 Insert Done parsing in 0.078 seconds

C:\Users\ATECH\Desktop\A.Document\linked_list_operations.cpp - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project | [*] linked_list_operations.cpp

```cpp
    // Traverse the list to find the task to be removed
    Task* current = head;
    while (current->next != NULL && current->next->id != id) {
        current = current->next;
    }
    // if the task with the specific ID was not found
    if (current->next == NULL) {
        cout << "Task with ID " << id << " not found.\n";
    } else {
        // Remove the task by adjusting pointers and freeing memory
        Task* temp = current->next;
        current->next = temp->next;
        cout << "Removed task with ID: " << temp->id << "\n";
        delete temp;
    }
}

// Function to view all tasks
void viewTasks(Task* head) {
    // if the list is empty
    if (head == NULL) {
        cout << "No tasks available.\n";
        return;
    }
    // Traverse the list and display each task information
    Task* current = head;
    while (current != NULL) {
        cout << "ID: " << current->id << ", Description: " << current->description << ", Priority: " << current->priority << "\n";
            //go to the next task
        current = current->next;
```

Compiler | Resources | Compile Log | Debug | Find Results

Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

ect | linked_list_operations.cpp

```cpp
            //go to the next task
        current = current->next;
    }
}
// Main function
int main() {
    // initialize the head of the task list to NULL
    Task* head = NULL;

    // variables for user input
    int choice, id, priority;
    string description;
    //while loop until the user exit
    while (true) {

        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;

        switch (choice) {
            case 1:
                // add a new task
                cout << "Enter task ID: ";
                cin >> id;
```

Compiler | Resources | Compile Log | Debug | Find Results

167 Col: 55 Sel: 0 Lines: 181 Length: 5862 Insert Done parsing in 0.047 seconds

C:\Users\ATECH\Desktop\A.Document\linked_list_operations.cpp - Dev-C++ 5.11

ile Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

roject C

linked_list_operations.cpp

```cpp

        switch (choice) {
            case 1:
                // add a new task
                cout << "Enter task ID: ";
                cin >> id;
                cin.ignore();

                cout << "Enter task description: ";
               // Get the task description
                getline(cin, description);

                cout << "Enter task priority: ";
                cin >> priority;

                // Call the function to add the task
                addTask(head, id, description, priority);
                break;
            case 2:
                viewTasks(head);
                break;
            case 3:
                removeHighestPriorityTask(head);
                break;
            case 4:
                cout << "Enter task ID to remove: ";
                cin >> id;
                removeTaskById(head, id);
                break;
            case 5:
                // free memory before exit the program
               cin.ignore() ;
                cout << "Exiting...\n";
                return 0;
            default:
                cout << "Invalid choice. Try again.\n";
        }
    }

```

OUTPUT:

```
C:\Users\ATECH\Desktop\A.Document\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: write a report
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: 1
Enter task ID: 102
Enter task description: read a book
Enter task priority: 3
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: 1
Enter task ID: 103
Enter task description: fix the error
Enter task priority: 10
Task added successfully.
```

C:\Users\ATECH\Desktop\A.Document\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: 2
ID: 103, Description: fix the error , Priority: 10
ID: 101, Description: write a report, Priority: 4
ID: 102, Description: read a book, Priority: 3

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: 103

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: 102
Removed task with ID: 102

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
ID: 101, Description: write a report, Priority: 4

Task Manager Menu:
```

C:\Users\ATECH\Desktop\A.Document\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: 4
Enter task ID to remove: 101
Removed task with ID: 101

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: 111
No tasks available.

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...

--------------------------------
Process exited after 161.6 seconds with return value 0
Press any key to continue . . .
```

Conclusion:

From this assignment, I learn and understand the following concepts:

- What is the linked list and how to work with linked lists.
- I learn how to add, remove, and traverse nodes in a linked list.
- Memory is allocated dynamically (using new) when we create a new task, and the memory is freed (using delete) when we remove a task. This helps optimize memory usage in my code.
- I learn that how to work with different types of input using cin, and how to handle input related issues like in the code I use cin.ignore() to resolve the issue of unwanted newline characters left in the input buffer after using function cin.

- **Problems/Challenges:** When I add a new task, the system asks me for the task ID and priority but skips the description due to a memory management issue with handling strings. The problem may be happening because the string is not allocated memory properly. To handle this problem I used the cin.ignore().