D1166570 Levi

Programming Assignment 4 Report:

Implementing a Queue with a Doubly-Linked List

I embarked on a programming journey to construct a queue utilizing a doubly-linked list in C++. This assignment was not merely a task but an exploration into the depths of data structures and their practical applications. Here, I outline the steps I undertook, the challenges I faced, and the insights I gained throughout this engaging process.

Dynamic Memory Management:

The heart of this assignment lay in managing memory efficiently. I honed my skills in dynamic memory allocation through the creation and management of nodes in a doubly-linked list. Each node, dynamically allocated, held data and pointers to its neighboring nodes. Learning to precisely control memory allocation and deallocation, I ensured robust handling of the queue's lifecycle without memory leaks.

Constructing the Queue Operations:

My primary task was to implement the fundamental operations of a queue: enqueue and dequeue. The enqueue operation involved appending a new node to the end of the list, while dequeue removed a node from the front, ensuring FIFO (First In, First Out) behavior. These operations required meticulous attention to pointer manipulation, particularly when adding or removing nodes from the edges of the list.

Dealing with Edge Cases:

A significant challenge was managing edge cases, such as dequeuing from an empty list or handling multiple consecutive operations efficiently. I implemented checks to prevent erroneous operations, which reinforced the robustness of the queue implementation.

Testing the Implementation:

To validate the functionality of my queue, I conducted randomized trials as specified in the assignment brief. Each trial consisted of a series of enqueue and dequeue operations, the randomness of which tested the queue's integrity under varying conditions. This not only demonstrated the queue's operational correctness but also its stability and performance under stress.

Organizing the Code:

I adopted a modular approach in organizing my code. This strategy involved separating the functionalities into different files (Node.h/cpp for node operations and IQueue.h/cpp for queue functionalities), enhancing maintainability and readability. Such organization allowed for easier debugging and future enhancements.

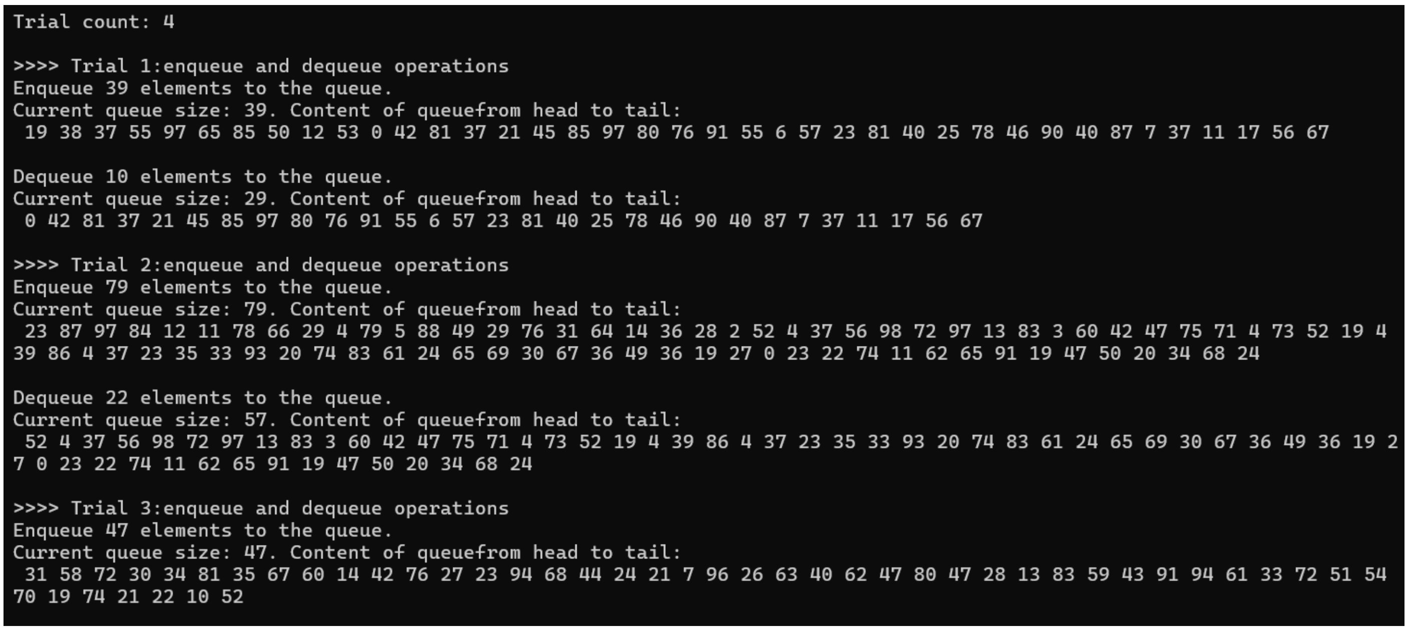
Reflections and Conclusions:

This project was a profound exercise in understanding and implementing complex data structures. It was a testament to the power of C++ in facilitating low-level memory management and object-oriented programming. Each line of code refined my programming skills and deepened my appreciation for meticulously crafted software solutions.

D1166570 Levi

In conclusion, this project was more than just an academic requirement; it was a narrative of personal growth and a stepping stone to more advanced programming endeavors. As I move forward, I carry with me the lessons learned and a renewed curiosity to explore further into the realms of computer science.

The following includes the result of my assignment.



```
Trial count: 4

>>>> Trial 1:enqueue and dequeue operations
Enqueue 39 elements to the queue.
Current queue size: 39. Content of queuefrom head to tail:
 19 38 37 55 97 65 85 50 12 53 0 42 81 37 21 45 85 97 80 76 91 55 6 57 23 81 40 25 78 46 90 40 87 7 37 11 17 56 67

Dequeue 10 elements to the queue.
Current queue size: 29. Content of queuefrom head to tail:
 0 42 81 37 21 45 85 97 80 76 91 55 6 57 23 81 40 25 78 46 90 40 87 7 37 11 17 56 67

>>>> Trial 2:enqueue and dequeue operations
Enqueue 79 elements to the queue.
Current queue size: 79. Content of queuefrom head to tail:
 23 87 97 84 12 11 78 66 29 4 79 5 88 49 29 76 31 64 14 36 28 2 52 4 37 56 98 72 97 13 83 3 60 42 47 75 71 4 73 52 19 4
39 86 4 37 23 35 33 93 20 74 83 61 24 65 69 30 67 36 49 36 19 27 0 23 22 74 11 62 65 91 19 47 50 20 34 68 24

Dequeue 22 elements to the queue.
Current queue size: 57. Content of queuefrom head to tail:
 52 4 37 56 98 72 97 13 83 3 60 42 47 75 71 4 73 52 19 4 39 86 4 37 23 35 33 93 20 74 83 61 24 65 69 30 67 36 49 36 19 2
7 0 23 22 74 11 62 65 91 19 47 50 20 34 68 24

>>>> Trial 3:enqueue and dequeue operations
Enqueue 47 elements to the queue.
Current queue size: 47. Content of queuefrom head to tail:
 31 58 72 30 34 81 35 67 60 14 42 76 27 23 94 68 44 24 21 7 96 26 63 40 62 47 80 47 28 13 83 59 43 91 94 61 33 72 51 54
70 19 74 21 22 10 52
```

2024/5/5

D1166570 Levi

2024/5/5