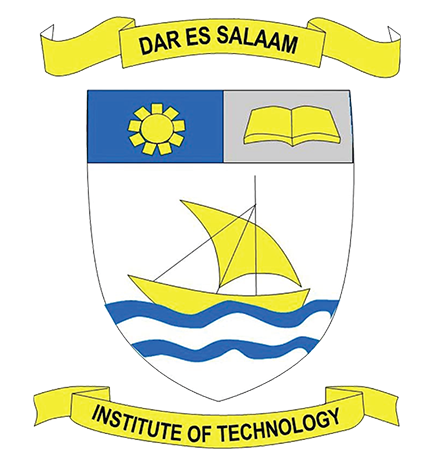
**DAR ES SALAAM INSTITUTE OF TECHNOLOGY (DIT)**

****

**DEPARTMENT OF COMPUTER STUDIES**

**ORDINARY DIPLOMA IN COMPUTER ENGINEERING**

**DATA STRUCTURE FOR TECHNICIANS**

**GROUP ASSIGNMENT**

|  |  |  |
| --- | --- | --- |
| **S/N** | **GROUP MEMBERS** | **REGISTRATION NO** |
| **1** | **FEISAL KHERI AHAMAD** | **220222498749** |
| **2** | **MISSIKE MLOLI KAMONJA** | **220222398442** |

**What is Queue Data Structure?**

The queue is a linear data structure that follows the principle of First-In-First-Out (FIFO) which means the element which is added first is the first one to be removed. It is like a real-world queue where the first person is the first to be served. In a queue data structure, the elements are added at one end, known as the rear, and removed from the other end, known as the front. The rear end is also known as the tail, and the front end is known as the head. Queues are commonly used in computer algorithms, such as breadth-first search, and scheduling.

**What is Enqueue Operation in Queue?**

Enqueue is the basic operation performed in a queue data structure to insert an element at the rear end of the queue. This operation is also referred to as "push" or "insert". When a new element is added to the queue using the enqueue operation, it is added after the last element of the queue. The rear pointer is then moved one position forward to point to the new element. The enqueue operation follows the principle of first-in-first-out (FIFO), which means that the element added first will be the first one to be removed using the dequeue operation.

**Time complexity of enqueue:** O(1), since enqueue only involves the addition of a new element at the end and the updation of the rear pointer.

**What is Dequeue Operation in Queue?**

Dequeue is an operation in a queue data structure that removes an element from the front end of the queue. It is also known as “delete” or “pop”.When the dequeue operation is performed on a queue, the first element of the queue is removed, and the front pointer is incremented by one position to point to the next element in the queue. The removed element is returned as the output of the dequeue operation. If the queue is empty, the dequeue operation cannot be performed, and it is called an underflow condition.

**Time complexity of the dequeue:** O(1), since it only involves removing the first element and moving the front pointer to the next element.

**Example to Understand Enqueue and Dequeue in Data Structure**

Let’s understand the enqueue and dequeue in data structure with the help of the following example.

Let’s take an empty queue and we want to enqueue and dequeue the following elements in the queue: 1, 2, 3, 4, and 5

The queue after each operation is given below.

* enqueue(1): 1
* enqueue(2): 1 2
* enqueue(3): 1 2 3
* enqueue(4): 1 2 3 4
* enqueue(5): 1 2 3 4 5
* dequeue(): 2 3 4 5
* dequeue(): 3 4 5
* dequeue(): 4 5
* dequeue(): 5

We can clearly see the queue follows the FIFO principle.

**Enqueue and Dequeue Algorithm**

The enqueue and dequeue algorithm is given below in a stepwise manner.

**Enqueue Algorithm:**

* **Step 1:** Check if the queue is full or not by comparing the number of elements in the queue with the maximum size of the queue.
* **Step 2:** If the queue is full, then display an overflow message and return.
* **Step 3:** If the queue is not full, then increment the rear pointer and insert the new element at the position pointed by the rear pointer.

**Dequeue Algorithm:**

* **Step 1:** Check if the queue is empty or not by comparing the number of elements in the queue with 0.
* **Step 2:** If the queue is empty, then display an underflow message and end the program.
* **Step 3:** If the queue is not empty, then remove the element at the front of the queue and increment the front pointer.

**Code for Enqueue and Dequeue in Python**

****

**Conclusion**  
In conclusion, enqueue and dequeue in data structure are fundamental operations that allow us to add and remove elements in a specific order. They are essential in many algorithms and applications where data needs to be processed in a sequential manner. Proper implementation of these operations ensures efficient use of memory and optimal performance. In this article, we discussed what enqueue and dequeue operations are in queue data structure and the implementation of enqueue and dequeue in C++, C, and Java languages.