# Introduction to Queue Data Structure

A queue is a linear data structure that follows the First-In-First-Out (FIFO) principle. It's a fundamental concept in computer science with various applications.

# Definition of Queue

A queue is an ordered collection of elements where the addition of new elements happens at the rear, and the removal of existing elements occurs at the front.

**Operations in Queue**

* Enqueue: Add an element to the rear of the queue.
* Dequeue: Remove and return the element from the front of the queue.
* Peek: Retrieve the element at the front of the queue without removing it.
* IsEmpty: Check if the queue is empty.

# Enqueue and Dequeue Processes

#### Enqueue

Add an element to the rear of the queue.

#### Dequeue

Remove and return the element from the front of the queue.

#### Peek

Retrieve the element at the front without removing it.

**Time Complexity of Queue Operations**

|  |  |
| --- | --- |
| Operation | Time Complexity |
| Enqueue | O(1) |
| Dequeue | O(1) |
| Peek | O(1) |

The constant time complexity makes queues efficient for many applications.

# Applications of Queue

#### Task Scheduling

Queues are used to manage the order of tasks, such as in operating systems, printer spoolers, and job scheduling.

#### Breadth-First Search

Queues are essential for implementing the Breadth-First Search (BFS) algorithm in graph theory.

#### Event Handling

Queues are used to manage the order of events, like in user interface event handling.

# Implementing Queue using Arrays

Queues can be implemented using a fixed-size array with two pointers: front and rear. Enqueue adds to the rear, and dequeue removes from the front.

# Implementing Queue using Linked Lists

Queues can also be implemented using a singly linked list. The front pointer tracks the first element, and the rear pointer tracks the last element.

Sample Code:

#define MAX 100

int queue[MAX], front = -1, rear = -1;

void enqueue(int x) { queue[++rear] = x; if (front == -1) front = 0; }

int dequeue() { int x = queue[front]; if (front == rear) front = rear = -1; else front++; return x; }

int peek() { return queue[front] }