# MsDS - Data Structures and Algorithms

Module No: 2

## Revision from Last Lecture

1. Abstraction is a strategic approach for handling complexities in software design, promoting cleanliness and organisation.
2. Data abstraction and encapsulation obscure internal data mechanisms, enhancing data integrity and protecting against interruptions.
3. Interface vs. implementation differentiation promotes flexibility and modularity in software design, allowing for updates without disrupting existing interfaces.
4. Abstract Data Types (ADTs) provide conceptual frameworks for data structures, allowing developers to focus on higher-level logic and optimise implementations.

## Expected Learning Outcomes

1. Understand the essential distinctions between static and dynamic memory allocation, comparing arrays and linked lists in particular.
2. Understand the idea of access-restricted lists and how they may be used in different computing environments.
3. Understand the fundamentals, functionality, and applications of stacks in data structure management.
4. Examine the features, traits, and real-world applications of queues and DEQueues in computing activities.

## Static Allocation vs Dynamic Allocation

* Memory allocation is a crucial aspect of software development, with static and dynamic allocation being the two main approaches.
* Static allocation assigns a predetermined amount of memory at compilation, while dynamic allocation allows for memory to be requested during runtime.
* Static allocation provides predictability but lacks flexibility, while dynamic allocation offers flexibility but requires manual memory management and can lead to memory leaks and fragmentation.

## Access Restricted Lists

* Access restricted lists impose limitations on how data can be accessed, added, or withdrawn, guaranteeing data integrity and optimising performance.
* Stacks follow the Last In, First Out (LIFO) concept, while queues adhere to the First In, First Out (FIFO) concept.
* Double-Ended Queues (DEQueues) combine the principles of stacks and queues, allowing for flexibility in various scenarios.

## Stacks

* Stacks are a data structure that follows the Last In, First Out (LIFO) principle.
* They allow efficient insertion and deletion of components from one end.
* Stacks are used in various computing contexts, such as expression evaluation and recursive function calls.

## Queues

* Queues are a linear data structure that follows the First In, First Out (FIFO) principle.
* They are used in various applications such as task scheduling in operating systems and managing data packets in computer networks.
* Queues offer enqueue and dequeue operations, and sometimes peek operations, making them essential in data structures and algorithms.

## DEQueues

* Double-Ended Queues (DEQueues) allow for adding and removing components from both the front and the back, providing more flexibility than traditional queues.
* DEQueues can operate as either a stack or a queue, or a combination of both, making them versatile for various applications.
* DEQueues are useful in scenarios such as the sliding window algorithm for processing arrays and strings, and in cache management techniques like the 'Least Recently Used' (LRU) method.

## Important Terminologies

**Static Allocation:** The memory allocation approach that involves a predefined size, often connected with arrays, is known as static memory allocation.

**Dynamic Allocation:** Dynamic memory allocation is a versatile technique that allows for the adjustment of memory size as required, often seen in the context of linked lists.

**Access Restricted Lists:** Data structures that impose restrictions on data access or modification based on specified criteria or locations.

**Stacks:** A data structure that follows the Last-In-First-Out (LIFO) principle, wherein the most recently inserted element is the first to be deleted.

**Queues:** The structure follows a First-In-First-Out (FIFO) approach, where the first element inserted is the initial element to be withdrawn.

**DEQueues:** Double-ended queues, also known as deques, are data structures that enable the insertion and removal of elements from both ends. They combine the characteristics of both stacks and queues.

## Summary

* Module 2 compares static allocation (arrays) with dynamic allocation (linked lists) in terms of memory allocation and access efficiency.
* Access-restricted lists are explored, emphasising the importance of restricting data access for operational efficiency and optimisation.
* Stacks and queues are fundamental data structures that follow LIFO and FIFO principles, respectively, with specific use cases in function calls, expression evaluation, work scheduling, and breadth-first search algorithms.