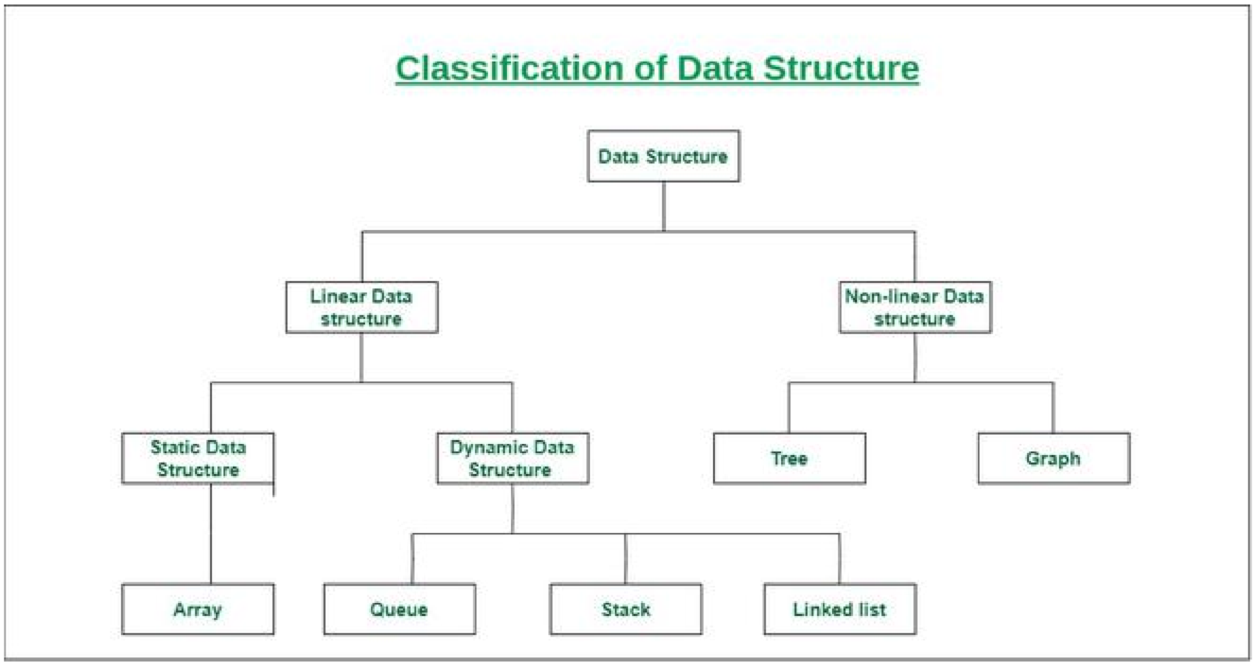
What is Data Structure?

A **data structure** is a storage that is used to store and organize data. It is a way of arranging data on a computer so that it can be accessed and updated efficiently.

A data structure is not only used for organizing the data. It is also used for processing, retrieving, and storing data. There are different basic and advanced types of data structures that are used in almost every program or software system that has been developed. So we must have good knowledge about data structures.



Classification of Data Structure:

1. **Linear Data Structure:** Data structure in which data elements are arranged sequentially or linearly, where each element is attached to its previous and next adjacent elements, is called a linear data structure.
   Example: Array, Stack, Queue, Linked List, etc.
2. **Static Data Structure:** Static data structure has a fixed memory size. It is easier to access the elements in a static data structure.
   Example: array.
3. **Dynamic Data Structure:** In dynamic data structure, the size is not fixed. It can be randomly updated during the runtime which may be considered efficient concerning the memory (space) complexity of the code.
   Example: Queue, Stack.

4. **Non-Linear Data Structure:** Data structures where data elements are not placed sequentially or linearly are called non-linear data structures. In a non-linear data structure, we can’t traverse all the elements in a single run only.
   Examples: Trees and Graphs.

What is a Linked List?

A linked list is a sequence of nodes in which each node is made up of two parts:

- **Data**: The value stored in the node.
- **Pointer**: A reference to the next node in the sequence.

Types of Linked List in C:

Linked list can be classified on the basis of the type of structure they form as a whole and the direction of access. Based on this classification, there are five types of linked lists:

1. Singly Linked List
2. Doubly Linked List
3. Circular Linked List

Stacks:

A stack is a linear data structure that follows the Last In, First Out (LIFO) principle. This means that the last element added to the stack is the first one to be removed. It can be visualized as a pile of plates where you can only add or remove the top plate.

Operations on Stack:

The primary operations associated with a stack are:

- **Push**: Adds an element to the top of the stack.
- **Pop**: Removes and returns the top element of the stack.
- **Peek (or Top)**: Returns the top element of the stack without removing it.
- **IsEmpty**: Checks if the stack is empty.
- **Size**: Returns the number of elements in the stack.

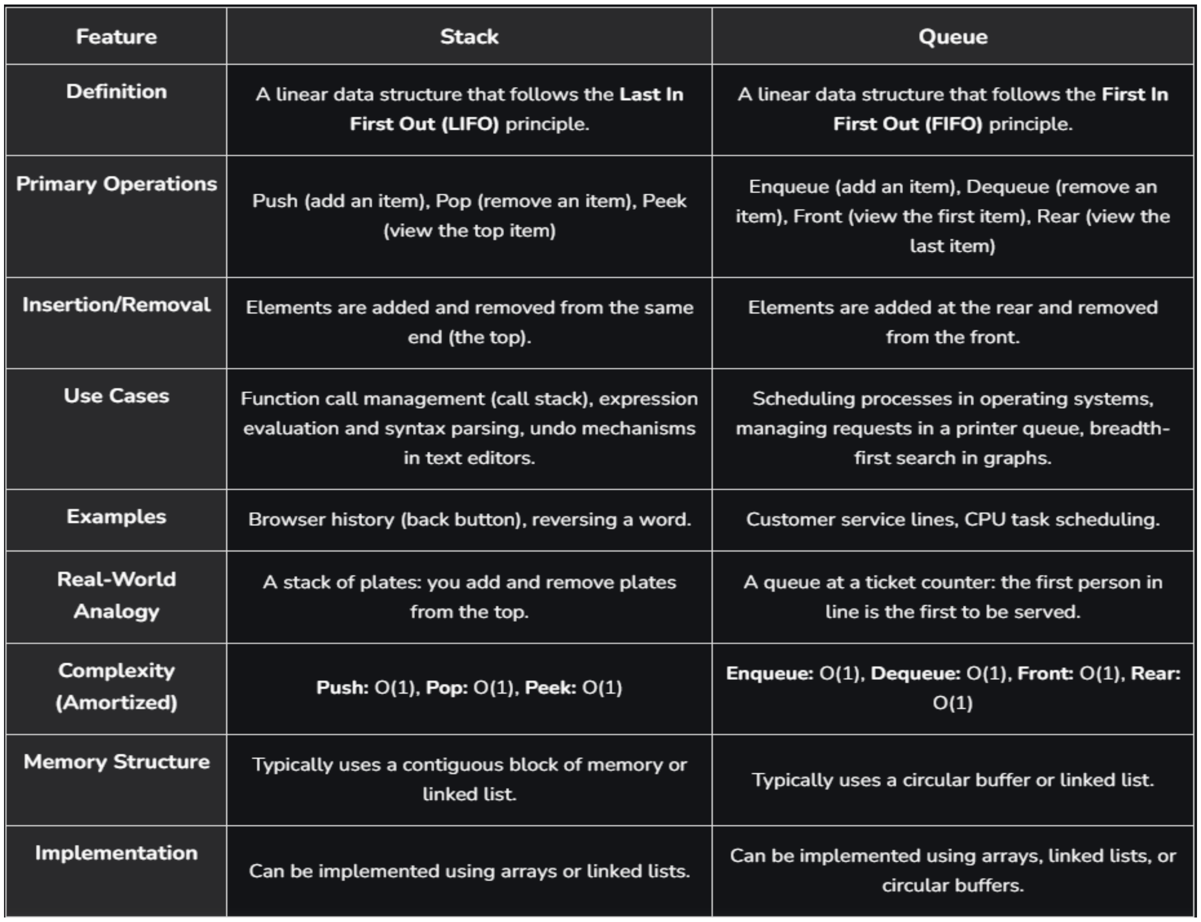
Queues:

A **queue** is a linear data structure that follows the First In, First Out (FIFO) principle. This means that the first element added to the queue is the first one to be removed. It can be visualized as a line of people waiting for a service, where the first person in line is the first to be served.

Operations on Queue:

The primary operations associated with a queue are:

- **Enqueue**: Adds an element to the end (rear) of the queue.
- **Dequeue**: Removes and returns the front element of the queue.
- **Front (or Peek)**: Returns the front element of the queue without removing it.
- **IsEmpty**: Checks if the queue is empty.
- **Size**: Returns the number of elements in the queue.



Feature	Stack	Queue
Definition	A linear data structure that follows the **Last In First Out (LIFO)** principle.	A linear data structure that follows the **First In First Out (FIFO)** principle.
Primary Operations	Push (add an item), Pop (remove an item), Peek (view the top item)	Enqueue (add an item), Dequeue (remove an item), Front (view the first item), Rear (view the last item)
Insertion/Removal	Elements are added and removed from the same end (the top).	Elements are added at the rear and removed from the front.
Use Cases	Function call management (call stack), expression evaluation and syntax parsing, undo mechanisms in text editors.	Scheduling processes in operating systems, managing requests in a printer queue, breadth-first search in graphs.
Examples	Browser history (back button), reversing a word.	Customer service lines, CPU task scheduling.
Real-World Analogy	A stack of plates: you add and remove plates from the top.	A queue at a ticket counter: the first person in line is the first to be served.
Complexity (Amortized)	**Push:** O(1), **Pop:** O(1), **Peek:** O(1)	**Enqueue:** O(1), **Dequeue:** O(1), **Front:** O(1), **Rear:** O(1)
Memory Structure	Typically uses a contiguous block of memory or linked list.	Typically uses a circular buffer or linked list.
Implementation	Can be implemented using arrays or linked lists.	Can be implemented using arrays, linked lists, or circular buffers.