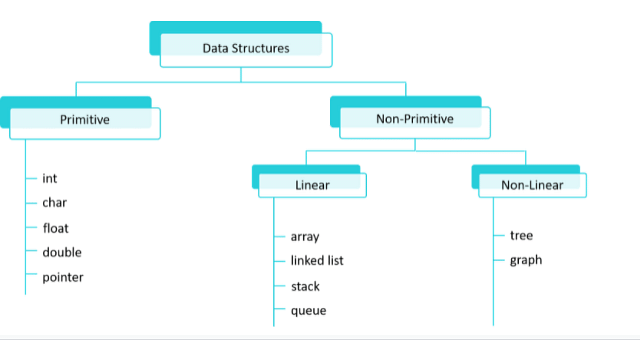
What is data structure?

Data structure is a specialized format for organizing, processing, retrieving, updating, and storing data.

Types of data structures:



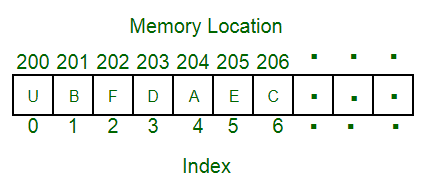
Primitive data structures:

* Character (char)
* - This data type can store a single character. It is typically 1 byte in size.
* Integer (int)
* - This data type can store whole numbers. It is typically 4 bytes in size.
* Floating-point (float)
* - This data type can store real numbers
* with decimal points. It is typically 4 bytes in size.
* Double (double)
* - This data type can store real numbers with decimal points, but it is more precise than float. It is typically 8 bytes in size

Linear Dara structures: Linear data structures are commonly used for organising and manipulating data in a sequential fashion. Some of the most common linear data structures include:

1. **Arrays:** A collection of elements stored in contiguous memory locations.
2. **Linked Lists:** A collection of nodes, each containing an element and a reference to the next node.
3. **Stacks:** A collection of elements with **Last-In-First-Out (LIFO)** order.
4. **Queues:** A collection of elements with **First-In-First-Out (FIFO)** order.

Array:An array is a collection of items of same data type stored at contiguous memory locations.



### **Types of Array operations:**

* **Accessing Elements:** Accessing a specific element in an array by its index is a constant-time operation. It has a time complexity of O(1).
* **Insertion:** Appending an element to the end of an array is usually a constant-time operation, O(1) but insertion at the beginning or any specific index takes O(n) time because it requires shifting all of the elements.
* **Deletion:** Same as insertion, deleting the last element is a constant-time operation, O(1) but deletion of element at the beginning or any specific index takes O(n) time because it requires shifting all of the elements.
* **Searching:** Linear Search takes O(n) time which is useful for unsorted data and Binary Search takes O(logn) time which is useful for sorted data.

Linkedlist: A Linked List is a linear data structure which looks like a chain of nodes, where each node contains a data field and a reference(link) to the next node in the list. Unlike Arrays, Linked List elements are not stored at a contiguous location.

**Common Features of Linked List:**

* **Node:** Each element in a linked list is represented by a node, which contains two components:
  + **Data:** The actual data or value associated with the element.
  + **Next Pointer(or Link):** A reference or pointer to the next node in the linked list.
  + **Head:** The first node in a linked list is called the “head.” It serves as the starting point for traversing the list.
  + **Tail:** The last node in a linked list is called the “tail.”

Types of Linked List operations:

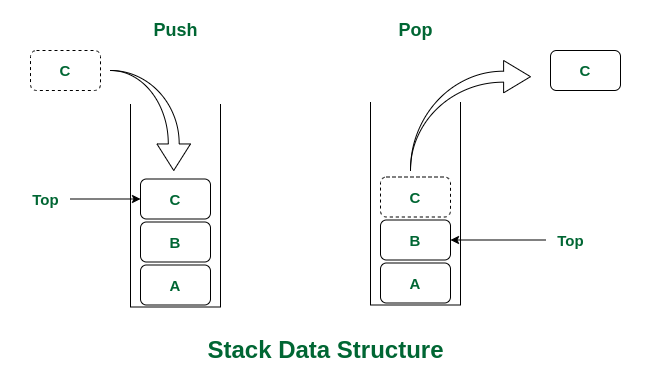
Accessing Elements: Accessing a specific element in a linked list takes O(n) time since nodes are stored in non conitgous locations so random access if not possible.

Searching: Searching of a node in linked list takes O(n) time as whole list needs to travesed in worst case.

Insertion: Insertion takes O(1) time if we are at the position where we have to insert an element.

Deletion: Deletion takes O(1) time if we know the position of the element to be deleted.

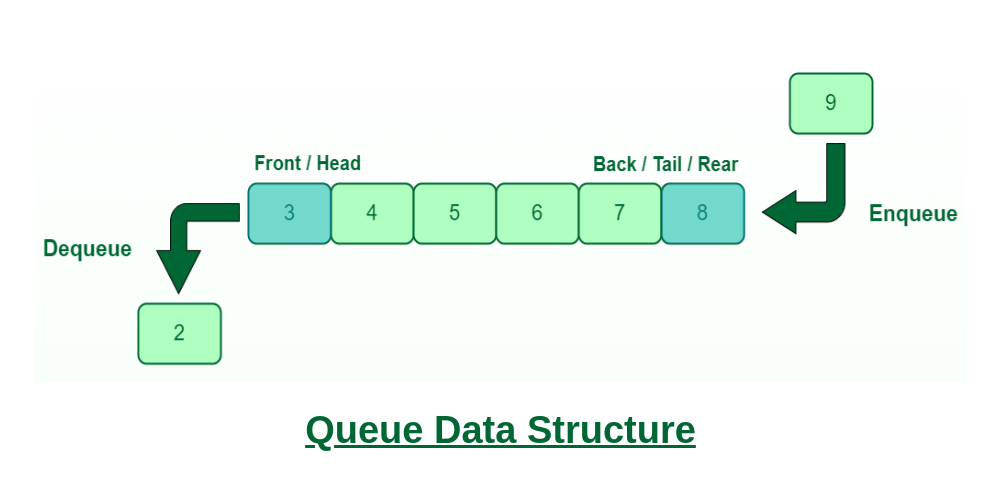
Stack:A stack is a linear data structure that follows the Last-In-First-Out (LIFO) principle, meaning that the last element added to the stack is the first one to be removed.



### **Stack Operations:**

* **push():** When this operation is performed, an element is inserted into the stack.
* **pop():** When this operation is performed, an element is removed from the top of the stack and is returned.
* **top():** This operation will return the last inserted element that is at the top without removing it.
* **size():** This operation will return the size of the stack i.e. the total number of elements present in the stack.
* **isEmpty():** This operation indicates whether the stack isempty or not.

Queue: A **queue** is a linear data structure that follows the **First-In-First-Out (FIFO)** principle. In a queue, the first element added is the first one to be removed.

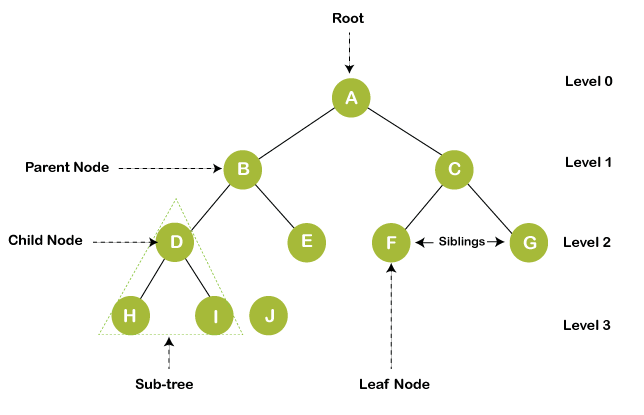


### **Queue Operations:**

* **Enqueue():** Adds (or stores) an element to the end of the queue..
* **Dequeue():** Removal of elements from the queue.
* **Peek() or front():** Acquires the data element available at the front node of the queue without deleting it.
* **rear():** This operation returns the element at the rear end without removing it.
* **isFull():** Validates if the queue is full.
* **isNull():** Checks if the queue is empty.

Non linear data structures:Data structures where data elements are not arranged sequentially or linearly are called **non-linear data structures**.

Trees: A tree data structure consists of various nodes linked together. The structure of a tree is hierarchical that forms a relationship like that of the parent and a child. The structure of the tree is formed in a way that there is one connection for every parent-child node relationship. Only one path should exist between the root to a node in the tree. Various types of trees are present based on their structures like AVL tree, binary tree, binary search tree, etc.



Graph:Graphs are those types of non-linear data structures which consist of a definite quantity of vertices and edges. The vertices or the nodes are involved in storing data and the edges show the vertices relationship. The difference between a graph to a tree is that in a graph there are no specific rules for the connection of nodes. Real-life problems like social networks, telephone networks, etc. can be represented through the graphs.

