* Data Structure is a **way of organizing data**. So that it can be **used effectively.**
* Data Structure is the **representation** of the **logical relationship exists between the individual element of data**.
* Data Structure is **logical or mathematical model of a particular organization**.
* Data Structure is the way of organizing data items that consider not only the element store but also their relationship between each other.

***Which Data Structure to use*?**

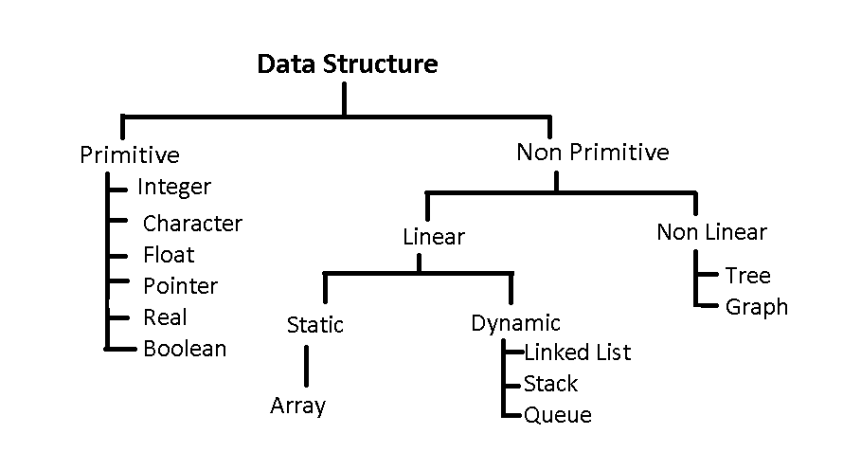
It depends on the solution approach to solve the problem, the problem and the nature of the data being processed.

There is only one data type which is integer. Data type depends on the line of abstraction that distinguished between datatypes.

As Floats are collection of integers.

5 4 .6 9 here 5, 4, 6, 9 belong to class of integer.

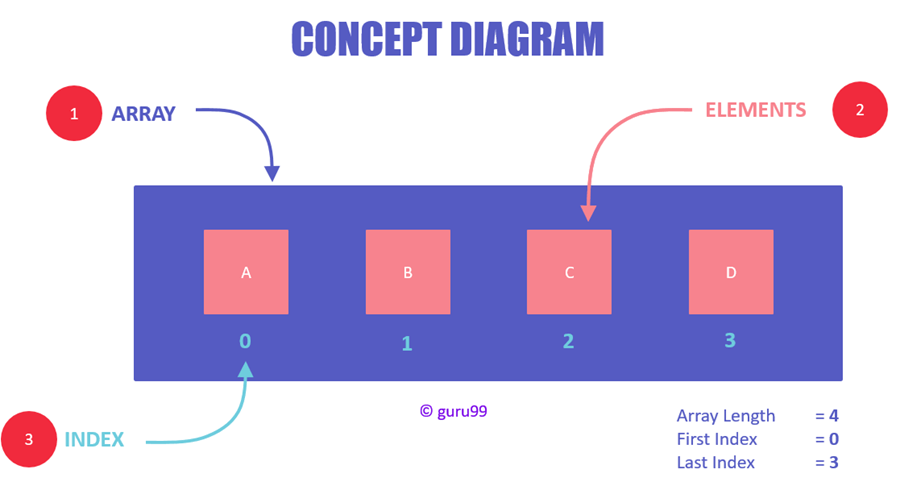
# Classification



|  |  |  |
| --- | --- | --- |
| Data Structure | Linear | Non-Linear |
| Operations | Sequential | Random |

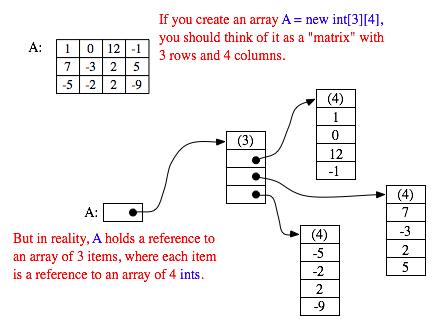
# Array

* Linear data structure
* Collection of similar data types
* Elements are stored in contiguous memory locations.
* Address of first element is known as base address and other elements are stored with a displacement of size of data type stored.
* Type – 1D, 2D, MD



1D array

Note- Array doesn’t store the element value but it stores the address of element.



2D array

# Stack

* Linear data structure
* One open end and another closed end
* All operation is performed from opened end
* Uses **LIFO** (last in first out)
* The topmost element is pointed with a pointer called **TOP**
* Insertion operation is known as **push** and deletion as **pop**

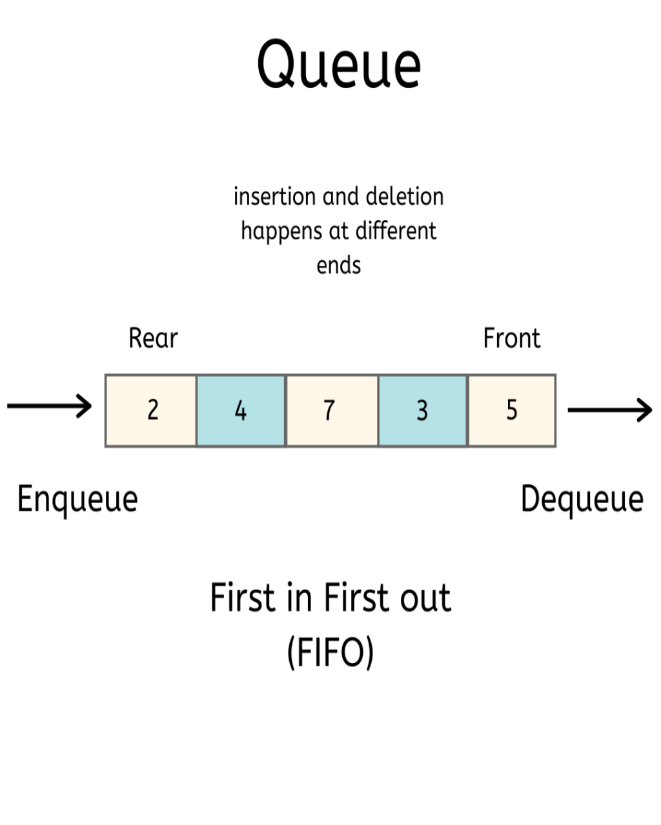


In fact, Stack & Queue is an abstract data type, which doesn’t define the underlying structure itself.

Stack only defines a set of supported operations that we can use as it is implement by different concrete data structures (such as arrays or linked lists).

# Queue

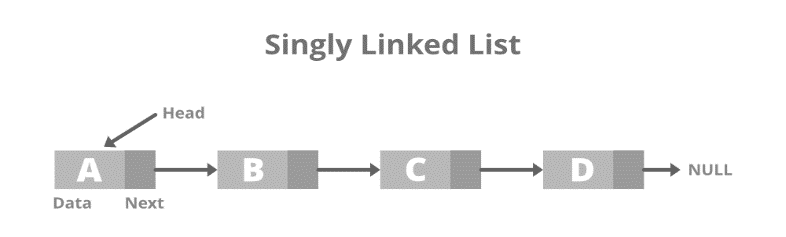
* Linear data structure
* Both ends are open
* Uses **FIFO** (first in first out)
* It used two pointer **FRONT** and **REAR**
* **Insertion *(enqueue operation)*** are performed from rear and **deletion *(dequeue operation)*** are perform by front end

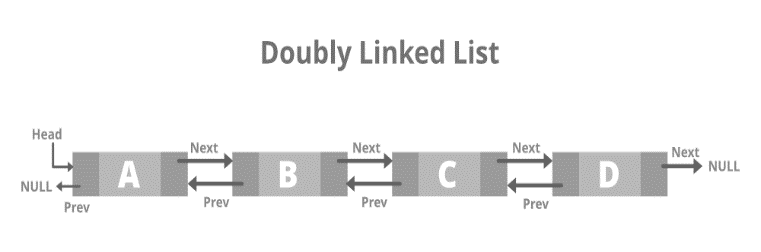


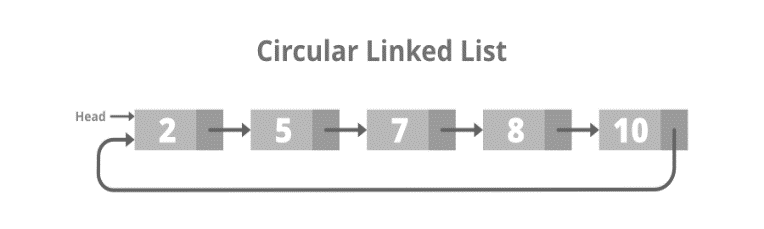
# Linked List

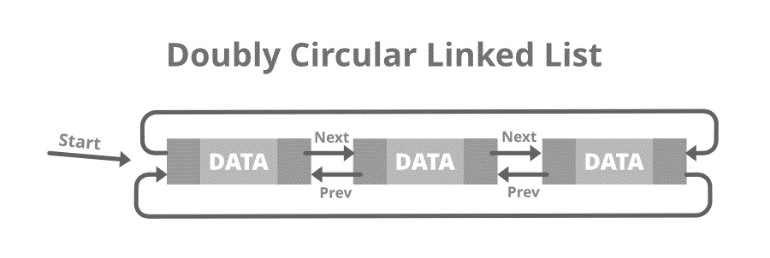
* Linear data structure.
* Element are stored inside nodes.
* Node contains at least two fields (1) Information/ Data (2) Address of next element.
* LinkedList is collection of nodes.

***Types***



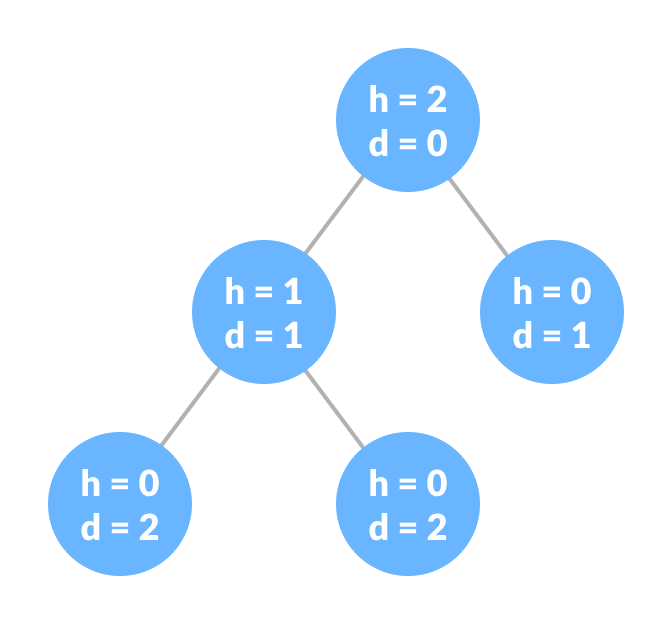






# Tree

root

  
  
  
**Node-** Individual data.  
  
**Root**- It is the **topmost node** of a tree.  
  
**Height of a Node**- The height of a node is the number of edges from the node to the deepest leaf (ie. the longest path from the node to a leaf node).  
  
**Depth of a Node**- The depth of a node is the number of edges from the root to the node.  
  
**Height of a Tree**- The height of a Tree is the height of the root node or the depth of the deepest node.  
  
**Degree of a Node**- The degree of a node is the total number of branches of that node.  
  
**Forest**- A collection of disjoint trees is called a forest.

Child

parent

Siblingss

Leaf- Nodes without children’s

# Binary Tree

Is a tree in which each node only have either 0, 1 and 2 children’s (at most 2 nodes).

# Generic Tree

Is a tree in which there can be n number of node.

Binary Search Tree  
It is a node-based binary tree data structure which has the following properties:

The left subtree of a node contains only nodes with keys lesser than the node’s key.

The right subtree of a node contains only nodes with keys greater than the node’s key.

The left and right subtree each must also be a binary search tree.

No duplicate nodes.

