**Arrays**

Ram (Random Access Memory) is measured in bytes.

1 GB = 109

1 byte = 8 bits

RAM consists of two things, a value and an address. A value will be stored in a particular address in the memory.

If you want to read a value in the array, it has O(1) big 0(1). It can read the value instantly.

If you want to insert a value in the array, one after the other, at the end of the array, it is O(1).

If you want to insert a value in a particular position, and shift all values, it becomes 0(n). That is worst case scenario. N is the number of elements.

**Dynamic Arrays**

It is a static array which can be resized. The basic functionality would be to check the length of the array with the capacity, and if they are equal a resized array which is doubled in size is returned.

**Stack vs List**

A stack is a collection that follows LIFO (Last In First Out) approach. It means that the last element of the list inputted is the first one to be removed. Push(), Pop(), Peak().

On the contrary, a list keeps the order of the items inputted. Add(), Remove(), RemoveAt().

**Linked List**

<https://www.c-sharpcorner.com/article/linked-list-implementation-in-c-sharp/>

A linked list is made up of list nodes. Each List node is made up of A value and Address.

Multiple List nodes make a Linked List.

A linked list is a data structure made up of a Value and Address

**Types of linked list:**

Singly Linked List: Points only to the next node and the last point will point to null.

Functionalities of a singly linked list:

1. It has two ListNodes which are head and tail
2. Insert Front
3. Insert End
4. Reverse Order

Doubly linked list: Each node contains two links, previous and next node. The first node and the last node, prev and next will point to null.

**Singly Linked List**

public class ListNode

{

Int value;

ListNode next;

public ListNode(int value)

{

this.value = value;

this.next = null;

}

}

**Doubly Linked List**

Public class DoubleListNode

{

Int value;

DoubleListNode prev;

DoubleListNode next;

Public DoubleListNode(int value)

{

This.value = value;

This.prev = null;

This.next = null;

}

}