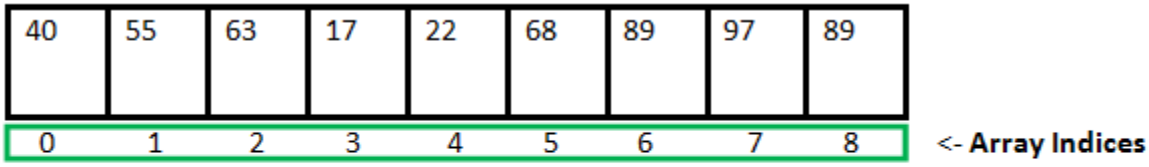
Array: Arrays store elements in contiguous memory locations, resulting in easily calculable addresses for the elements stored and this allows faster access to an element at a specific index.



Array Length = 9
First Index = 0
Last Index = 8

```
//Javascript  program to illustrate arrays
//Initializing an array of size 4
const arr = [1,3,5,3];
console.log(arr[3]); //gives the element at index 3
console.log(arr[2]); //gives the element at index 2
```

Advantages of array data structure:

- **Efficient access to elements**: Arrays provide direct and efficient access to any element in the collection. Accessing an element in an array is an O(1) operation, meaning that the time required to access an element is constant and does not depend on the size of the array.
- **Fast data retrieval**: Arrays allow for fast data retrieval because the data is stored in contiguous memory locations. This means that the data can be accessed quickly and efficiently without the need for complex data structures or algorithms.
- **Memory efficiency:** Arrays are a memory-efficient way of storing data. Because the elements of an array are stored in contiguous memory locations, the size of the array is known at compile time. This means

that memory can be allocated for the entire array in one block, reducing memory fragmentation.

- **Easy to implement:** Arrays are easy to implement and understand, making them an ideal choice for beginners learning computer programming.

Disadvantages of array data structure:

- **Fixed size:** Arrays have a fixed size that is determined at the time of creation. This means that if the size of the array needs to be increased, a new array must be created and the data must be copied from the old array to the new array, which can be time-consuming and memory-intensive.
- **Insertion and deletion issues**: Inserting or deleting an element from an array can be inefficient and time-consuming because all the elements after the insertion or deletion point must be shifted to accommodate the change.
- **Wasted space:** If an array is not fully populated, there can be wasted space in the memory allocated for the array. This can be a concern if memory is limited.

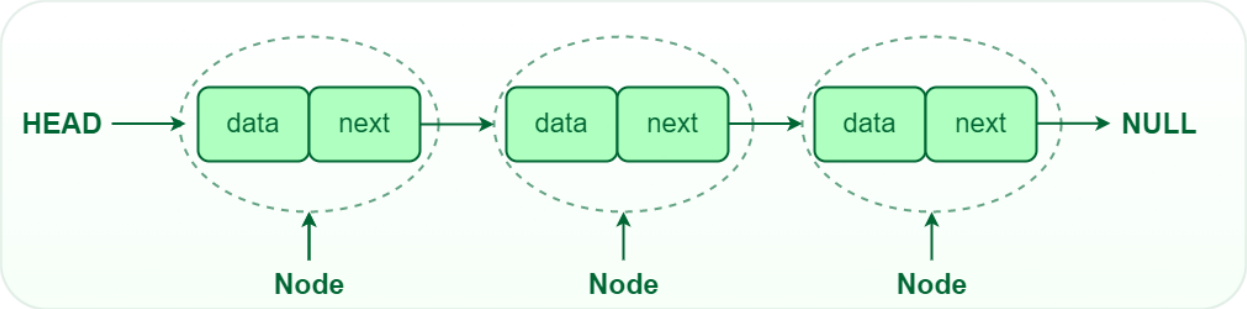
ARRAY	LINKED LISTS
1. Arrays are stored in contiguous location.	1. Linked lists are not stored in contiguous location.
2. Fixed in size.	2. Dynamic in size.
3. Memory is allocated at compile time.	3. Memory is allocated at run time.
4. Uses less memory than linked lists.	4. Uses more memory because it stores both data and the address of next node.
5. Elements can be accessed easily.	5. Element accessing requires the traversal of whole linked list.
6. Insertion and deletion operation takes time.	6. Insertion and deletion operation is faster.

Linked List:

In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

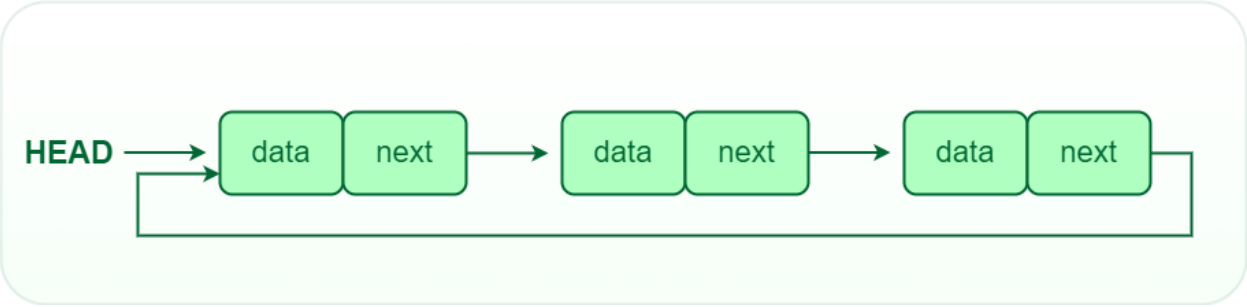
Types of Linked List:

1> Singly Linked List:



2> Doubly Linked List

3> Circular Linked List



Advantages of Linked Lists:

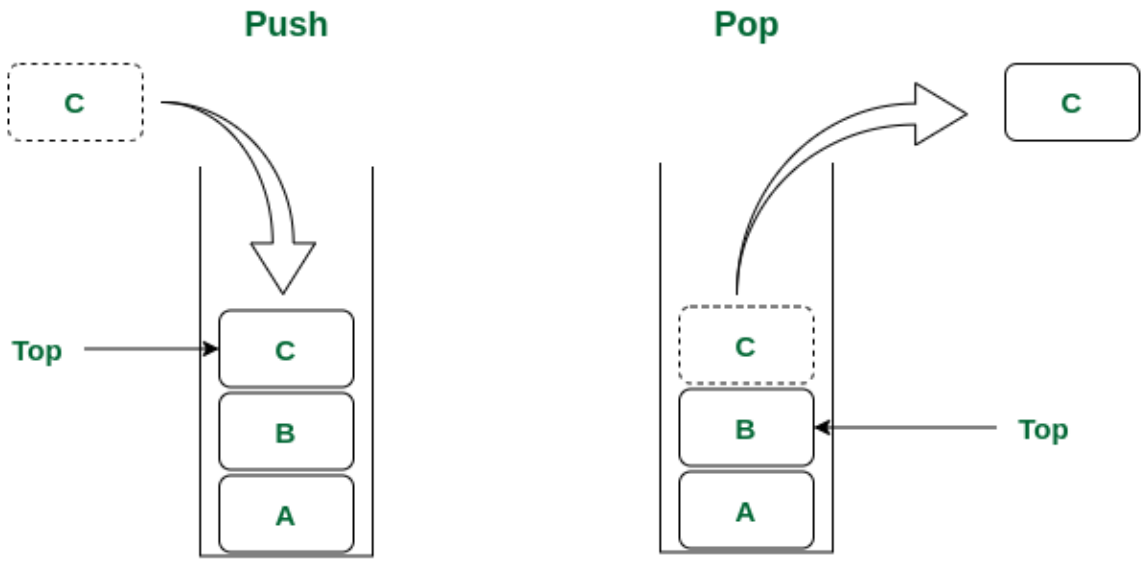
- The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of usage, and in practical uses, the upper limit is rarely reached.
- Inserting a new element in an array of elements is expensive because a room has to be created for the new elements and to create a room existing elements have to be shifted.

Disadvantages of Linked Lists:

- Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do a binary search with linked lists.
- Extra memory space for a pointer is required for each element of the list.
- Arrays have a better cache locality that can make a pretty big difference in performance.

Stack:

To implement the stack, it is required to maintain the pointer to the top of the stack, which is the last element to be inserted because we can access the elements only on the top of the stack.



Stack Data Structure