**Data Structures**

**Data Structure**

Data structure is a way of storing and organizing data in computer so that it can be used efficiently.

Different kinds of data structure are suited to different kind of applications and some are highly specialized to specific tasks.

There are two types of data structure.

1. Linear data structure
2. Non-Linear data structure

**Linear Data structure**

A data structure is said to be linear if its elements form a sequence or linear list.

Examples: Stack, Queue, Array, and Linked List

**Non-Linear Data structure**

A data structure is said to be non-linear if traversal of nodes is nonlinear in nature.

Examples: Graph and Tree

**What are the various operations that can be performed on different Data Structures?**

**Insertion** Add a new data item in the given collection of data items.

**Deletion**Delete an existing data item from the given collection of data items.

**Traversal** Access each data item exactly once so that it can be processed.

**Searching** Find out the location of the data item if it exists in the given collection of data items.

**Sorting** Arranging the data items in some order i.e. in ascending or descending order in case of numerical data and in dictionary order in case of alphanumeric data.

**Stack**

Stack is a linear data structure which demonstrates the Last In First Out (LIFO) or First In Last Out (FILO)

**Push** inserts the elements at the top.

**Pop** Return the top element after removing from the stack.

**Top** Return the top element without removing from the stack.

Examples:

Sort values in a stack.

Check balanced parentheses in an expression

Reverse a string

**Queue**

Queue is another linear data structure that stores the element in a sequential manner.

The significant difference between Stack and Queue is that instead of using LIFO method, Queue implements First In First Out (FIFO)

A queue is an ordered list in which insertions are done at one end (rear) and deletions are done at another end (front).

When an element is inserted in queue, the concept is called Enqueue.

When an element is removed from the queue, the concept is called Dequeue.

**Enqueue** () – Inserts the elements to the end of the queue

**Dequeue** () – Removes an element from the start of the queue

**Top** () – Returns the first element of the queue

## **Arrays**

int[] array = {1, 3, 5, 2, 6, 9};

/\*

\* Index: 0 1 2 3 4 5

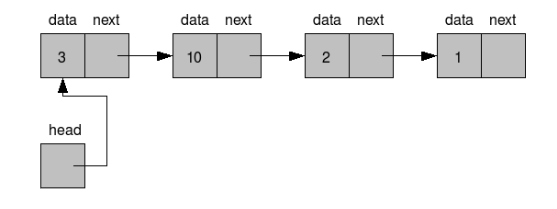
\* Value: 1 3 5 2 6 9

\*/

**linked list**

A linked list is a linear data structure (like arrays)

Linked List is the collection of randomly stored data objects called nodes. In Linked List, each node is linked to its adjacent node through a pointer. A node contains two fields, i.e. Data Field and Link Field.



A linked list is considered both linear and non-linear data structure depending upon the situation.

* On the basis of data storage, it is considered as a non-linear data structure.
* On the basis of the access strategy, it is considered as a linear data-structure.

# List Performance

|  |  |  |
| --- | --- | --- |
|  | Array | List |
| Access | O(1) | O(n) |
| Update | O(1) | O(n) |
| Append | O(1)/O(n) | O(1) |
| Traversal | O(n) | O(n) |

**advantages of Linked List over an array?**

* The size of a linked list can be incremented at runtime which is impossible in the case of the array.
* The List is not required to be contiguously present in the main memory, if the contiguous space is not available, the nodes can be stored anywhere in the memory connected through the links.
* The List is dynamically stored in the main memory and grows as per the program demand while the array is statically stored in the main memory, size of which must be declared at compile time.
* The number of elements in the linked list are limited to the available memory space while the number of elements in the array is limited to the size of an array.

**Types of Linked List :**

1. **Singly Linked List :**In this type of linked list, every node stores address or reference of next node in list and the last node has next address or reference as NULL. For example 1->2->3->4->NULL
2. **Doubly Linked List :**Here,here are two references associated with each node, One of the reference points to the next node and one to the previous node. Eg. NULL<-1<->2<->3->NULL
3. **Circular Linked List :**Circular linked list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list. Eg. 1->2->3->1 [The next pointer of last node is pointing to the first]