Data Structures are a specialized means of organizing and storing data in computers in such a way that we can perform operations on the stored data more efficiently. Data structures have a wide and diverse scope of usage across the fields of Computer Science and Software Engineering.

Data structures are being used in almost every program or software system that has been developed. Moreover, data structures come under the fundamentals of Computer Science and Software Engineering. It is a key topic when it comes to Software Engineering interview questions. Hence as developers, we must have good knowledge about data structures.

In this article, I will be briefly explaining 8 commonly used data structures every programmer must know.

1. Arrays

An **array** is a structure of fixed-size, which can hold items of the same data type. It can be an array of integers, an array of floating-point numbers, an array of strings or even an array of arrays (such as *2-dimensional arrays*). Arrays are indexed, meaning that random access is possible.

Array operations

- **Traverse**: Go through the elements and print them.
- **Search**: Search for an element in the array. You can search the element by its value or its index
- **Update**: Update the value of an existing element at a given index

Inserting elements to an array and **deleting** elements from an array cannot be done straight away as arrays are fixed in size. If you want to insert an element to an array, first you will have to

create a new array with increased size (current size + 1), copy the existing elements and add the new element. The same goes for the deletion with a new array of reduced size.

Applications of arrays

- Used as the building blocks to build other data structures such as array lists, heaps, hash tables, vectors and matrices.
- Used for different sorting algorithms such as insertion sort, quick sort, bubble sort and merge sort.

2. Linked Lists

A **linked list** is a sequential structure that consists of a sequence of items in linear order which are linked to each other. Hence, you have to access data sequentially and random access is not

possible. Linked lists provide a simple and flexible representation of dynamic sets.

Let's consider the following terms regarding linked lists. You can get a clear idea by referring to Figure 2.

- Elements in a linked list are known as **nodes**.
- Each node contains a **key** and a pointer to its successor node, known as **next**.
- The attribute named **head** points to the first element of the linked list.
- The last element of the linked list is known as the **tail**.