**Data Structures in C**

**Data Structures** in **C** are used to store data in an organised and efficient manner. The C Programming language has many data structures like an *array, stack, queue, linked list, tree, etc.* A programmer selects an appropriate data structure and uses it according to their convenience.

Let us look into some of these data structures:



**ARRAY**

An Array is a sequential collection of elements, of the same **data type.** They are stored sequentially in memory. An Array is a data structure that holds a similar type of elements. The array elements are not treated as objects in c like they are in java.

There are two types of arrays:

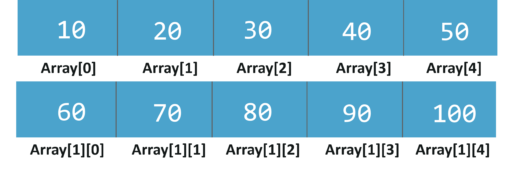
* Single dimensional array



The syntax for one-dimensional Arrays:

**data\_type array\_name[array\_size];**

* Multidimensional array

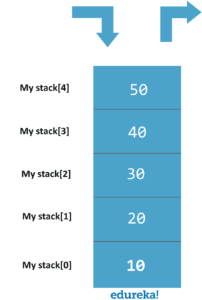


The syntax for two-dimensional Arrays:

**data\_type array\_name[rows][columns];**

**STACK**

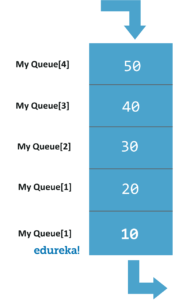
A stack is a linear data structure. It follows the**last in first out** approach. A new item is added at the top of a stack. Both insert and deletion operation is performed from one end of the stack.



There are two functions associated with stacks. Push function to add elements to the stack and pop function to remove elements from the stack.

**QUEUE**

A Queue is a linear data structure that stores a collection of elements. The queue operates on**first in first out** (FIFO) algorithm.



There are 2 pointers, the front is at the front of the queue and rear is at the back of the queue. We add elements from the back of the queue and remove them from the front of the queue.

**LINKEDLIST**A Linked List is a data structure. It is linear. The Linked List is like an array but, the Linked List is not stored sequentially in the memory. Every linked list has 2 parts, the**data section** and the **address section** that holds the address of the next element in the list, which is called a node.

The size of the linked list is not fixed, and data items can be added at any locations in the list. The disadvantage is that to get to a node, we must traverse to from the first node to the node that we require.

There are three types of link lists:

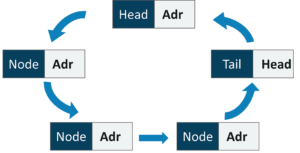
* **Singly link list**

Data-Structures-in-c-singly-linked-list-edureka

* **Doubly Link List**

Data-Structures-in-c-doubly-linked-list-edureka

* **Circular link list**

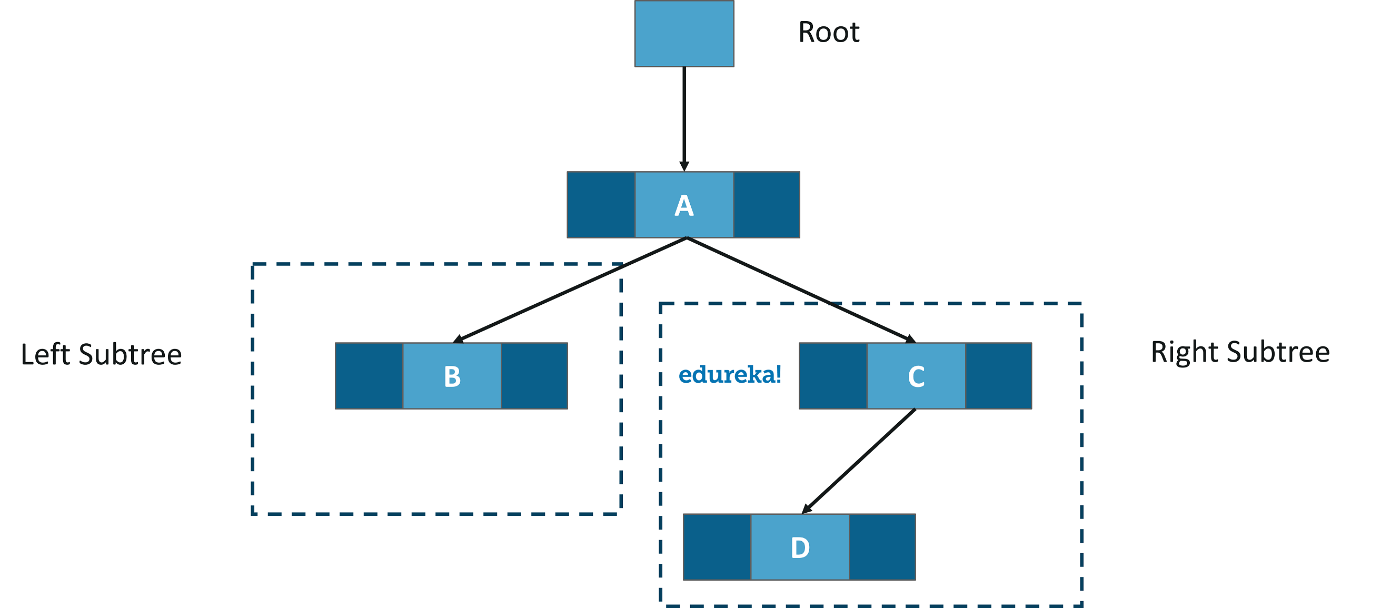


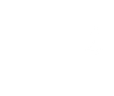
Format:[data ,address]

**TREES**



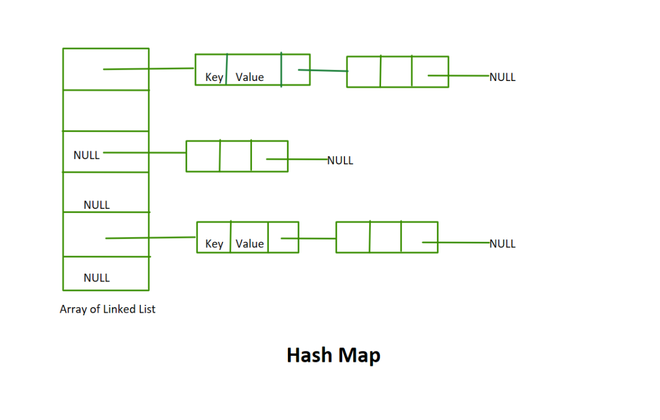
A tree is a data structure that has one **root node** and many **sub-nodes.** It is another one of the data structures which are designed on top of a linked list.





**HASHING**

Hash table is another data structure. It is used to implement an associative array, a structure that can map **keys** to **values.** Hash table uses a hash function to compute an index into an array of buckets. Hash tables are very useful data structures.





## **Graph Data Structure**

A graph is a collection of nodes (vertices) interconnected by edges. This abstraction allows us to represent various relationships between objects or entities. Formally, a graph G is defined as a pair V,E), where V represents the set of vertices or nodes, and E represents the set of edges connecting these nodes.  
In computer science and mathematics, the graph data structure stands as a fundamental concept with far-reaching applications. From social networks to transportation systems, algorithms leveraging graphs power a wide range of modern technologies.

