**LAR Liyanage**

**27336**

**Tutorial 01**

1. **Write down the difference between an array and the structure.**

* Array :- An array is a data structure that stores a fixed-size sequence of elements of the same type. It allows for random access to its elements using an index. The elements in an array are typically stored in contiguous memory locations, which enables efficient element access. Arrays are suitable for situations where a collection of similar elements needs to be stored and accessed efficiently. They are often used for storing and manipulating large sets of data, like storing a list of numbers or characters.
* Structure :- A structure, also known as a record, is a data structure that allows for storing related data of different types together. It is defined as a user-defined data type that contains one or more variables, each with its own data type. Structures provide a way to group related data into a single unit, making it easier to organize and manipulate data. Unlike arrays, structures can have elements of different types and sizes. They are often used to represent entities with multiple attributes, such as a person with a name, age, and address.

1. **Where should you use data structures?**

* Database management systems :- Data structures like indexes, hash tables, and trees are used to efficiently store and retrieve data in databases.
* Operating systems :- Data structures like queues, stacks, and linked lists are used in the implementation of various data management algorithms and data manipulation operations.
* Network programming :- Data structures like graphs and trees are used to represent network topologies and optimize network routing algorithms.
* Compiler design :- Data structures like symbol tables, parse trees, and abstract syntax trees are used in the compilation process to store and manipulate program information.
* Artificial intelligence :- Data structures like graphs and trees are used to represent complex relationships and hierarchies in AI algorithms like decision trees and neural networks.

1. **What are the types of data structures?**

* Arrays :- Arrays are a collection of elements of the same type that are stored in contiguous memory locations. They provide efficient access to elements using indices but have a fixed size.
* Linked Lists :- Linked lists are made up of nodes that contain both data and a reference to the next node. They allow for efficient insertion and deletion but require sequential traversal for access.
* Stacks :- Stacks are a Last-In-First-Out (LIFO) data structure. Elements are added and removed from only one end called the top of the stack.
* Queues :- Queues are a First-In-First-Out (FIFO) data structure. Elements are added at one end called the rear and removed from the other end called the front.
* Trees :- Trees are hierarchical data structures with nodes connected by edges. They have a root node, and each node can have child nodes. Examples include binary trees, AVL trees, and B-trees.
* Graphs :- Graphs are a collection of nodes, called vertices, and edges that connect these nodes. They represent complex relationships and can be directed or undirected.
* Hash Tables :- Hash tables use a hash function to map keys to an index in an array. They provide efficient key-value pair lookups but can have collisions if multiple keys map to the same index.

1. **What is a linked list data structure?**

* A linked list is a linear data structure in which elements , called nodes, are connected together using pointers or references. Each node contains both data and a reference to the next node in the sequence. Unlike arrays, linked lists do not require contiguous memory allocation, which allows for efficient insertion and deletion of elements at any position. However, accessing an element in a linked list requires traversing the list from the beginning, which can be slower compared to arrays. Linked lists are commonly used when dynamic memory allocation is required or when the number of elements is unknown or frequently changing.

1. **What is a recursive data structure?**

* A recursive data structure is a type of data structure that is defined in terms of itself. In other words, it is a data structure that includes references or pointers to instances of the same data structure. This self-referential definition allows for the creation of complex, hierarchical structures by reusing a smaller version of the structure within itself. Recursive data structures are often used to represent hierarchical relationships or nested structures. Examples of recursive data structures include linked lists, trees, and graphs.

1. **Compare and contrast linear data structures Vs Non-linear data structures.**

* Linear data structures :- Linear data structures have elements arranged in a sequential manner. Each element has a direct predecessor and successor, except for the first and last elements. Examples of linear data structures include arrays, linked lists, stacks, and queues.
* Elements are accessed and processed one by one in a linear order.
* Efficient for searching, accessing, and manipulating data in a sequential manner.
* Insertion and deletion operations are relatively simple.
* Examples: arrays, linked lists, stacks, queues.
* Non-linear data structures :- Non-linear data structures do not have a sequential arrangement of elements. Elements may have multiple predecessors and successors, forming complex relationships. Examples of non-linear data structures include trees, graphs, and hash tables.
* Elements are accessed and processed in a non-sequential manner, following various relationships.
* Efficient for representing complex relationships or hierarchies.
* Insertion and deletion operations may involve traversing multiple elements.
* Examples: trees, graphs, hash tables.