Tutorial 01

1.

Aspect	Array	Structure
Definition	Homogeneous collection of elements of the same data type.	Composite data type that groups variables of different data types.
Data Type	All elements must have the same data type.	Fields (variables) can have different data types within the same structure.
Memory Allocation	Contiguous memory allocation for the entire array.	Independent memory allocation for each field within the structure.
Size	Fixed size, determined at the time of declaration.	Size is the sum of the sizes of its individual fields, can be dynamic.
Accessing Elements	Accessed using an index (position in the array).	Accessed using member names (field names) followed by a dot (.).
Usage	Suitable for lists, vectors, matrices, and collections of similar data.	Suitable for representing records, objects, or complex data entities.

2.

- Storing Collections of Data
- Searching and Retrieving Data
- Sorting Data
- Graph Representation
- Managing Memory
- String Processing
- Caching and Memorization
- Handling Large Datasets
- Simulation and Modeling
- Optimizing Algorithm Performance

3.

- Linear Data Structures

- Non-Linear Data Structures

4. A linked list is a linear data structure where elements (nodes) are connected using pointers. Each node contains data and a reference to the next node. Linked lists allow dynamic memory allocation and efficient insertion/deletion. They are commonly used for dynamic collections and serve as the foundation for other data structures like stacks and queues.

5. A recursive data structure is a type of data structure that defines itself in terms of instances of the same structure. It allows for nested representations, such as trees and linked lists. Recursive data structures are useful for organizing hierarchical data but require careful handling to avoid infinite loops in recursive algorithms.

6.

Aspect	Linear Data Structures	Non-linear Data Structures
Definition	Elements are arranged in a linear or sequential order.	Elements have hierarchical or non-sequential relationships.
Representation	Typically represented as a sequence of nodes connected one after the other.	Represented as branching or interconnected structures.
Traversal	Linear traversal from the first element to the last.	Non-linear traversal, with multiple paths and directions.
Access	Random access to elements by index is efficient (e.g., arrays).	Random access can be inefficient, as it may involve traversal (e.g., trees).
Examples	Arrays, Linked Lists, Stacks, Queues	Trees (Binary Trees, AVL Trees, etc.), Graphs
Memory Allocation	Contiguous memory allocation in arrays.	Dynamic memory allocation using pointers or references.
Insertion/Deletion	Insertion/deletion can be efficient depending on the position (e.g., stacks, queues).	Insertion/deletion can be more complex, especially in trees and graphs.