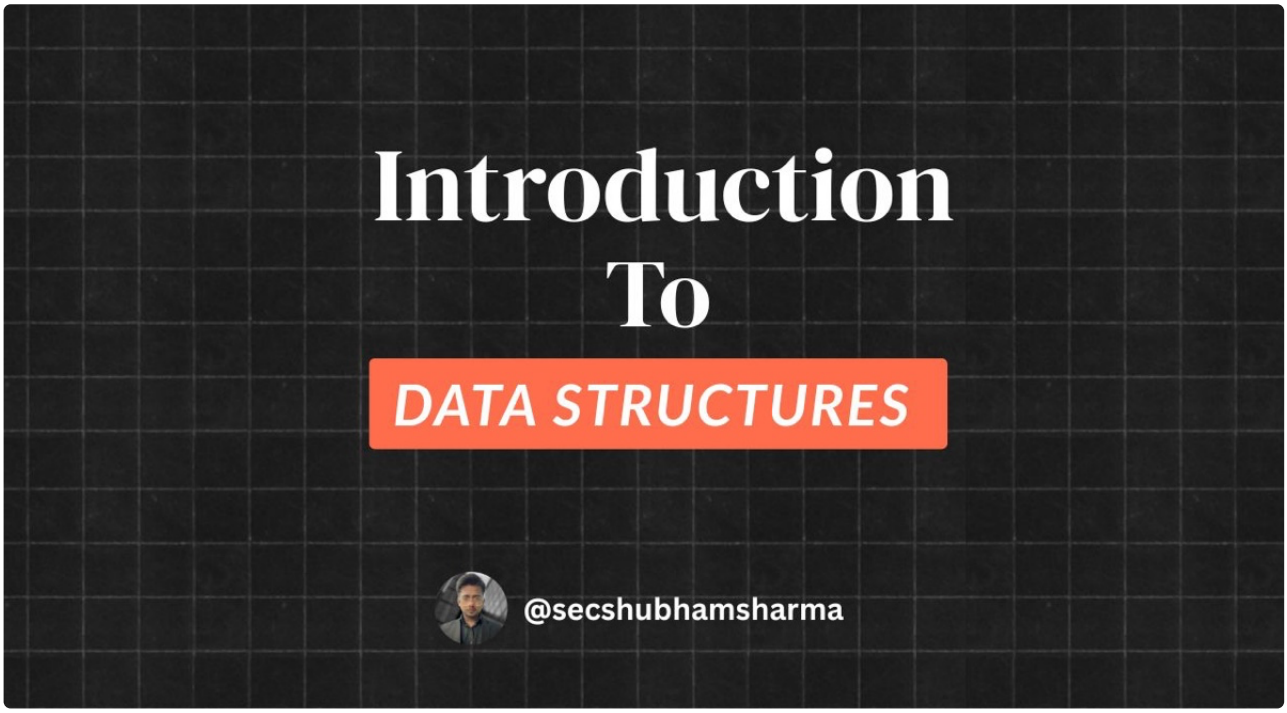
Introduction To Data Structures



A **data structure** is a format to store, organize, and manage data efficiently. There are two major categories:

Types of Data Structures

Primitive Data Structures

Primitive Data Structures are the **most basic types** of data available in a programming language. These are **directly supported by the compiler** and provide the foundation for building more complex structures.

Key Features:

- Store single values.
- Fixed memory size (in most cases).
- Direct operations allowed (add, compare, etc).
- Fast to use but limited in structure.

Types of Primitive Data Structures:

Type	Description	Example in C/C++/Java
Integer	Stores whole numbers	`int x = 5;`
Float	Stores real numbers (decimals)	`float pi = 3.14;`
Character	Stores a single character	`char c = 'A';`
Boolean	Stores `true` or `false`	`bool isActive = true;`
Pointers (C/C++)	Stores memory addresses	`int *ptr = &x;`

Non-Primitive Data Structures

Non-Primitive Data Structures are **derived from primitive types** and allow us to **store multiple values**and perform complex operations.

Key Features:

- Organize large data efficiently.
- Allow operations like insertion, deletion, traversal, searching.
- Can be **linear** or **non-linear**.
- Help optimize time and space.

there are three types of Non-Primitive Data Strcutures

A. **Linear Data Structures**

Elements are arranged **sequentially** (one after another).

Structure	Description	Examples/Use Cases
Array	Fixed-size, same-type elements in contiguous memory	Store numbers, names, etc.
Linked List	Dynamic nodes where each node points to the next	Dynamic memory usage
Stack	LIFO (Last In, First Out)	Undo operation, Expression evaluation
Queue	FIFO (First In, First Out)	Task scheduling
Deque	Insert/delete from both ends	Sliding window problems

Linear DS Operations:

- Traversal
- Insertion/Deletion
- Search

B. **Non-Linear Data Structures**

Elements are arranged **hierarchically** or with complex relationships (not one-after-another).

Structure	Description	Examples/Use Cases
Tree	Hierarchical structure with parent-child nodes	File system, DOM
Binary Tree	Max two children per node	Expression trees
Binary Search Tree (BST)	Sorted binary tree	Fast search & insert
Heap	Complete binary tree (min/max heap)	Priority queue
Trie	Tree for storing strings with prefix sharing	Auto-complete
Graph	Nodes connected with edges	Social networks, GPS maps

Non-linear DS Operations:

- Traversal (DFS, BFS)
- Search
- Shortest path
- Topological sorting

C. **Hash-Based Data Structures**

Use **hash functions** to map keys to values for fast lookup.

Structure	Description	Examples/Use Cases
Hash Table / Hash Map	Store key-value pairs	Dictionary, database indexing
Set	Store unique elements	Remove duplicates, fast lookups