Introduction to Data Structure

Data Types:

In programming, a data type specifies the type of data a variable can hold. It defines the values a variable can store, as well as the operations that can be performed on that data. Data types are typically divided into two categories: primitive (system-defined) and non-primitive (user-defined).

1. Primitive (System-Defined) Data Types: These are the basic data types provided by the programming language. They are predefined, meaning the programmer does not need to define them. For example: Integer, Floating Point, Character, String.

2. Non-Primitive (User-Defined) Data Types: These are more complex data types created by the programmer. They may consist of multiple primitive data types grouped together. For example: Array, Linked List.

Abstract Data Types:

An Abstract Data Type (ADT) is a theoretical model that defines the behavior of a data structure purely by the operations it can perform, without specifying how these operations will be implemented. ADTs focus on what the data structure should do, not how it does it. For example, a stack is defined by push, pop, and peek operations.

Data Structures:

A Data Structure is the actual implementation of an ADT. It specifies the way data is organized and stored in memory to support the operations defined by the ADT. Data structures focus on the how, meaning how data is stored, accessed, and manipulated to perform the desired operations efficiently. For example, a stack can be implemented using an array or linked list which then makes a stack a data structure. Data structures take into account factors like:

Time complexity: How fast operations like insertion, deletion, and searching can be performed.
Space complexity: How much memory the structure uses.

Depending on the organization of the elements, data structures are classified into two types:

- Linear data structures: Elements are accessed in a sequential order but it is not compulsory to store all elements sequentially. For example: Linked Lists, Stacks, and Queues.
- Non–linear data structures: Elements of this data structure are stored/accessed in a non-linear order. For example: Trees and Graphs.

Data Structures will have 3 core operations:

- a way to add things
- a way to remove things
- a way to access things