ABSTRACT and CONCRETE DATA STRUCTURES

ABSTRTCT DATA TYPE

An abstract data type (ADT) is a way of describing the behavior and properties of a data type, without specifying how it is implemented.

ABSTRTCT DATA STRUCTURE

- Defined by its behavior from users perspective
  - Data description and
  - Operations to manipulate it
- Describes only what needs to be done
- Does not describe how it is done

CONCRETE DATA STRUCTURE

- Describes
  - Data value
  - Relations among the data
  - Operations applied to the data
- Describes exactly how the data are organized
- Describes how tasks are performed

E.g. ABSTRTCT and CONCRETE DATA STRUCTURE

- E.g. A STACK is an ADS that defines data and its organization that supports adding and removing elements in a last-in, first-out (LIFO) order.
- However, an ADS does not tell how to store or access the elements of a stack, or what kind of data it can hold. That is left to the Concrete Data Structure.

ABSTRTCT vs. CONCRETE DATA TYPE/STRUCTURE

Abstract Data Type/Structure	Concrete Data Type/Structure
Describes behavior and properties of a data type, without specifying how it is implemented.	Describes data, operations and its implementation. A CDT can also have additional features or constraints that are not part of the ADT.
It is usable beyond its original use.	It is rarely reusable beyond its original use
It hides the internal details	It doesn’t hide anything.
Allow programmers to work with data structures based on their functionality rather than the specific implementation.	CDTs are specific implementations of ADTs that define the internal representation, organization, and algorithms used to realize the operations of the ADT
E.g.- lists, sets, stacks.	E.g.- Arrays, linked lists, trees, graphs.

ABSTRTCT and CONCRETE DATA STRUCTURE Analogy

- **Playlist** analogy, ADTs resemble the concept of a playlist, allowing actions like "add song" without detailing the method, offering managerial flexibility.
- Conversely, CDTs compare to choosing between **Spotify** or **CDs** for your playlist.
- Spotify, like an 'ArrayList', permits easy changes and broad selection, while CDs, similar to a 'LinkedList', provide a sequential experience.
- Starting with CDs and transitioning to Spotify demonstrates the blending of ADTs and CDTs.
- The 'Playlist' ADT remains consistent, but its implementation (CDT) adapts, emphasizing ADTs for defining functions and CDTs for specific realizations, crucial in adaptable, efficient system design.