Abstract data types & Arrays

ADTs are the way of classifying data structures by providing a minimal expected interface and list of methods

ADT → Minimal required functionality
ADT → Operations

ARRAY - ADT

An array ADT holds the collection of given elements accessible by an index

elements → (can be int, float, custom)

Minimal functionality → get(i) → get element i
Set(i, num) → Set element i to num.

representation

Operations:- Max()
Min()
Search(num)
Insert(i, num)
Append(x)

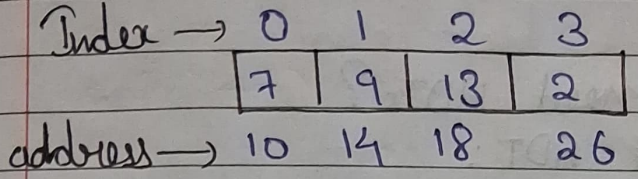
Static and Dynamic arrays

Static arrays → Size cannot be changed

Dynamic arrays → Size can be changed

RANKA
DATE / /
PAGE

Memory representation of Arrays



⇒ Array of Size 4

Elements in an array are stored in contigous memory locations

Elements in an array can be accessed using the base address in constant time → O(1)

(*) what is an Abstract Data type (ADT)?

Defination (very important)

An Abstract Data type (ADT) is a logical description of a data structure that defines:

- what data is stored
- what Operations can be performed
- what each Operation does

without Specifying:

- How the data is stored in memory
- How the operations are implemented

in short

ADT = what + why (not How)

Simple analogy (lock & key)

- you use a lock
- you know:
  - insert key
  - turn key
  - door Opens

- you don't care how bits move inside
  → that lock is an ADT

(*) why do we even need ADTs?
problem without ADT:
- Code tightly coupled to implementation
- changing logic breaks entire codebase
- No flexibility
- Hard to scale

ADT solves this by:
- Separating interface from implementation
- Allowing multiple implementations
- Making code reusable
- Making algorithms cleaner

DSA is not about data structures. It is about how you model data

(*) ADT vs Data Structure

ADT	Data structure
Conceptual	Concrete
Logical view	physical implementation
Defines Operations	Defines memory layout
language independent	language dependent

Example:
Stack ADT:
- push()
- pop()
- peek()
- isEmpty()

RANKA
DATE / /
PAGE

Stack implementation:
- Using array
- Using linked list

(*) Key Concepts or Components of an ADT
Every ADT has three parts:

(*) Data
what type of values are stored?
Example
- Stack → elements
- Queue → elements
- Set → unique elements

(*) Operations
what actions are allowed?
Example:
- insert
- delete
- search
- traverse

(*) Rule (Behavior)
what does each operation do?
Example:
- Stack follows LIFO
- Queue follows FIFO

} we will study latter

Types of Abstract Data types
(1) primitive ADTs
Build-in, basic data types.

Examples:
- int
- float
- char
- boolean

} Already abstracted by language

(2) Non-primitive ADTs

User-defined logical structures

{ Such user-defined logical structures we are going to watch in detail }