INTRODUCTION TO DATA STRUCTURES

INTRODUCTION AND DEFINITION

- A data structure is a specialized format for organizing and storing data.
- It enables efficient access and modification.
- Examples: arrays, linked lists, stacks, queues, trees, and graphs.

CLASSIFICATION OF DATA

- In **data structures**, *classification of data* refers to how data is organized, stored, and accessed. Here's a clear and structured classification:

1. Primitive Data Types

- These are the basic building blocks provided by programming languages.
- **Integer** (int)
- **Float** (float/double)
- **Character** (char)
- **Boolean** (true/false)

2. Non-Primitive Data Types

These are derived from primitive types and can store multiple values.

a) Linear data structures

Data elements are arranged sequentially.

- **Array** – fixed-size collection of elements of the same type.
- **Linked list** – nodes with data and a reference to the next node.
- **Stack** – LIFO (last in first out) structure.
- **Queue** – FIFO (first in first out) structure.

B) Non-linear data structures

Data elements are not arranged sequentially.

- **Tree** – hierarchical structure (eg., Binary tree, BST).
- **Graph** – set of nodes (vertices) connected by edges.

3. Abstract Data Types (ADTs)

Abstract Data Types (ADTs) are **theoretical concepts** in computer science that define **how data is organized and what operations are allowed**, without specifying how they are implemented.

ADTs are used

- To **separate logic from implementation**
- To build **modular and reusable code**
- To allow **flexibility** in changing the implementation without affecting the user
- Example: Stack ADT

Operations:

- push(x) – Add element x to top
- pop() – Remove and return top element
- peek() – Return top element without removing

4. Static vs. Dynamic Data Structures

Static data structures

Definition:

Static data structures have a fixed size. The size is determined at the time of declaration and cannot be changed at runtime.

Characteristics:

- Memory is allocated at compile time
- Faster access (since memory is contiguous)
- Less flexible (size cannot grow/shrink)

Examples: array

- Static stack (implemented using array)
- Static queue

Example in : int arr[10]; // Fixed-size array with 10 elements

Dynamic Data Structures

- **Definition:**
  dynamic data structures can **grow or shrink during program execution**, depending on the need.

Characteristics:

- Memory is allocated **at runtime**
- **More flexible** in size
- May be **slightly slower** due to extra memory handling (pointers, heap)

Examples:

- **Linked list**
- **Dynamic stack** (using linked list or dynamic array)
- **Dynamic queue**
- **Tree, graph**

5. Homogeneous vs. Heterogeneous Data

- **Homogeneous** – all elements are of the same type (eg: Array of integers).
- **Heterogeneous** – elements can be of different types (eg: Structures in C).

DATA STRUCTURES……

- The organized collection of data is called Data Structure.

An example for a data structure is an 'Array'

Array is a linear structure in with each elements are arranged in a sequence.

10	20	30	40	50

DATA STRUCTURE

- Given a set of values 10,5,20 50 etc…
- You can organize values in the form of sorted array

DATA STRUCTURE

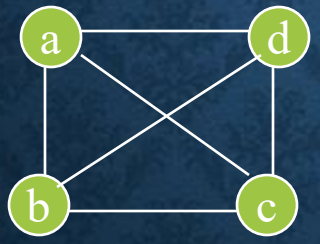
- You can organize the data in the form of a linked list

- **You can organize the data in the form of a tree or graph**

DATA STRUCTURE

Data Structure=Organised Data+Allowed Operations

Another example for data structures are



Graph

Trees

Record

E001	Abhi	Clerk

CLASSIFICATION OF DATA STRUCTURE

- Linear Data structure
  - In linear data structure the data items are arranged in a linear sequence like in an array.

Classification of Data Structure

- Non-Linear Data structure
  - In non linear data structure the data items are not in a sequence.
  - Eg: a Tree

CLASSIFICATION OF DATA STRUCTURE

- Homogenous Data structure
  - If all the elements are of the same type it is called homogenous data structure

  Eg: Array

Classification of Data Structure

- Non-Homogenous Data structure

  In non-homogenous data structure the elements may or may not be of the same type.

  Eg: Record

E001	Anu	Manager	10000

Classification of Data Structure

- Another way of classifying data structure is static data structure and dynamic data structure.
- Static structures are ones whose sizes are fixed at compile time
- Eg: Array
- Dynamic data structures ones which expand or shrink as required during the program execution.
- Eg: Linked List

OPERATIONS ON DATA STRUCTURES

- **The operations performed on any data structure are:-**
- **Traversal – processing each element**
- **Search – Finding the location of the element with a given key value or the record with a given key**
- **Insertion Adding a new element**
- **Deletion Removing an element**
- **Sorting Arranging the element in some type of order**
- **Merging**

THE NEED FOR DATA STRUCTURES

- **Data structure organize data. As a result, you can create more efficient programs that runs on less space and less time.**
- **With data structure you can create applications that operate on more complex data.**
- **With data structure you can perform complex calculations.**
- **The choice of data structure and algorithm can make the difference between a program running in a few seconds or many hours.**

SELECTING A DATA STRUCTURE

Select a data structure as follows:

1. Analyze the problem to determine the resource constraints a solution must meet.
2. Determine the basic operations that must be supported.
3. Select the data structure that best meets these requirements.

STATIC AND DYNAMIC MEMORY ALLOCATION IN C

There are two types of memory allocations:

- Compile-time or Static Memory Allocation
- Run-time or Dynamic Memory Allocation

Static Memory Allocation	Dynamic Memory Allocation
In the static memory allocation, **variable**s get allocated permanently, till the program executes or function call finishes.	In the Dynamic memory allocation, the memory is controlled by the programmer. It gets allocated whenever a malloc() is executed gets deallocated wherever the **free()** is executed.
Static Memory Allocation is done before program execution.	Dynamic Memory Allocation is done during program execution.
It uses stack for managing the static allocation of memory	It uses heap (not heap data structure) of memory for managing the dynamic allocation of memory
It is less efficient	It is more efficient
In Static Memory Allocation, there is no memory re-usability	In Dynamic Memory Allocation, there is memory re-usability and memory can be freed when not required
In static memory allocation, once the memory is allocated, the memory size can not change.	In dynamic memory allocation, when memory is allocated the memory size can be changed.
In this memory allocation scheme, we cannot reuse the unused memory.	This allows reusing the memory. The user can allocate more memory when required. Also, the user can release the memory when the user needs it.
In this memory allocation scheme, execution is faster than dynamic memory allocation.	In this memory allocation scheme, execution is slower than static memory allocation.
In this memory is allocated at compile time.	In this memory is allocated at run time.
In this allocated memory remains from start to end of the program.	In this allocated memory can be released at any time during the program.