Introduction to Data Structures in C

Simple Explanation with Real-Life Examples

What is a Data Structure?

- A Data Structure is a way to organize and store data in a computer so that it can be accessed and modified efficiently.
- - Helps perform operations like searching, sorting, insertion, deletion.
- - Makes programming easier and more powerful.

Common Data Structures in C

- 1. Array - stores elements in fixed size.
- 2. Structure - groups different data types.
- 3. Linked List - dynamic list using pointers.
- 4. Stack - LIFO structure.
- 5. Queue - FIFO structure.
- 6. Tree - hierarchical data structure.
- 7. Graph - nodes connected by edges.

Real-Life Examples

-  Bookshelf - Array
-  Train Coaches - Linked List
-  Browser History - Stack
-  Ticket Line - Queue
-  Folder in Folder - Tree
-  Cities & Roads - Graph

Why Do We Need Data Structures?

- 📦 Efficient storage of large data
- ⚡ Faster access & searching
- 🧠 Logical data grouping
- 🔄 Dynamic memory usage
- 🔁 Reusable in real-world apps

Self-Referential Structure

- A self-referential structure is a struct in C that contains a pointer to itself—meaning it can refer to another variable of the same structure type
- 🧠 **Simple Explanation:**
- "It’s like a person holding the hand of another person of the same type — forming a chain."

C Code Example

- struct Node {
- int data;
- struct Node* next;
- };

- struct Node n1, n2;
- n1.data = 10; n2.data = 20;
- n1.next = &n2; n2.next = NULL;

Memory Diagram

- +--------+ +--------+
- | data=10| ---> | data=20|
- | next | | next=NULL |
- +--------+ +--------+
- n1 n2

What is a Data Type?

- 🔹 A data type defines what kind of data a variable can hold.
- 🔹 It helps the compiler allocate memory and perform operations.
- 🔸 Two categories:
- ✅ Primitive Data Types
- 📦 Non-Primitive Data Types

✅ Primitive Data Types

- ✔️ Built-in types that store a single value.
- Examples:
- 🔹 int – Whole numbers
- 🔹 float – Decimal numbers
- 🔹 char – Single character
- 🔹 double – Precise decimal numbers

📦 Non-Primitive Data Types

- ✔️ Derived or user-defined types.
- Used to store multiple values or complex data.
- **Examples**:
- 🔹 Array – List of same type elements
- 🔹 Structure – Group of mixed types
- 🔹 Union – Shared memory group
- 🔹 Pointer – Stores memory address
- 💡 Analogy: Box or folder (can hold many items)

What is a Linear Data Structure?

- In Linear Data Structures, the data elements are arranged sequentially or in a linear order, one after the other.
- **Examples of Linear Data Structures:**
- Array
- Linked List
- Stack
- Queue

- What is a Non-Linear Data Structure?

- In Non-Linear Data Structures, the data elements are not arranged sequentially.
- Instead, they are connected hierarchically or in a graph-like structure.
- **Examples of Non-Linear Data Structures:**
- Tree
- Graph
- Heap

- **Algorithm, Analysis of Algorithm, and Big O Notation**

What is an Algorithm?

- - A step-by-step procedure to solve a problem.

- - Real-Life Example: Making Tea
- 1. Boil water.
- 2. Add tea leaves.
- 3. Pour into cup.
- 4. Add sugar.
- - Analogy: Like a recipe.
- Like a recipe or instruction manual.

Why Do We Need Algorithms?

- **How to say it:**
- **“Algorithms help us:**
- Solve problems step by step.
- Save time.
- Reduce mistakes.
- Handle lots of data.”
- ✅ **Analogy:**
- “Just like you follow steps to assemble furniture, computers follow steps to process data.”

🌿 3. What is Analysis of Algorithm?

- **How to say it:**
- “Once we have an algorithm, we need to ask:
- Is it fast?
- Does it use too much memory?
- This is called analysis.”
- ✅ **Example to use:**
- “Think of a traffic light. If the timer is fixed, it may cause jams at busy times. But a smart sensor that adjusts the light duration is more efficient.”

🌿 5. What is Big O Notation?

- **How to say it:**
- “Big O notation shows how the algorithm’s time or memory grows as the input gets bigger.”
- ✅ **Example**:
- “If you look for a word in a dictionary:
- Page by page: linear time (O(n)).
- Open in the middle and narrow down: binary search (O(log n)).”
- 🎯 **Tip**: Use hand gestures—show “big” with arms when explaining scaling.

Big O Chart

- **How to say it:**
- “Different Big O notations mean different speeds:
- O(1): Fastest, doesn’t change.
- O(log n): Very fast.
- O(n): Slower.
- O(n^2): Much slower.
- O(2^n): Extremely slow for big inputs.”