Data Structures usng C and C++

Section 1: Before we Start

Introduction

Data structures are defined as the way data is organized in main memory so that a program can use it efficiently during execution.

Programs consist of **code and data**; data structures focus on how data is arranged for efficient operations.

Types of Data Structures

- **Physical data structures**: Arrays, Matrices, Linked Lists (define memory arrangement).
- **Logical data structures**: Stacks, Queues, Trees, Graphs, etc. (define how data is used).
- Arrays and matrices are built into most languages; linked lists must be implemented manually in C.

Why Study Data Structures

- They are a **core subject** in computer science academics.
- Essential for **software development**; applications cannot be built without them.

Levels of Learning Data Structures

1. Basic understanding of what they are and where to use them.
2. In-depth knowledge of operations and **time/space complexity analysis**.
3. Ability to **implement data structures from scratch**.

- The course targets **Level 3**, including implementation and analysis.

Programming Languages

- Any language can be used conceptually.
- Modern languages provide built-in data structures (STL in C++, Collections in Java/C#, containers in Python, etc.).
- Understanding usage requires only basic knowledge, but implementation requires deeper study.

Why C Language Is Used

- C has **no built-in data structures**, making it ideal for learning implementations from scratch.
- Helps clearly understand internal operations.
- Concepts can be easily transferred to **C++**, and adapted to Java or C#.

Course Organization

- Starts with a **brush-up of essential C and C++ concepts** (functions, structures, classes, templates, parameter passing).
- Covers **sorting techniques** (bubble, selection, insertion, etc.) with implementation and analysis.
- Begins with a detailed section on **recursion**, explaining its importance in problem-solving despite efficiency concerns.

Recursion and Problem Solving

- Recursion is fundamental to **mathematical problem-solving**.
- Many problems are first solved recursively, then converted to loops for efficiency.
- Understanding recursion is crucial for strong problem-solving skills.

Algorithms Clarification

- The course focuses on **data structures and algorithms applied to them**, not large-scale industry algorithms (e.g., Google or Facebook algorithms).
- General algorithms are treated as a **separate subject** and covered elsewhere.

Section 2: Essential C and C++ Concepts

Arrays Basics

.