Data Structures

Unit:1

1. Introduction to Data Structures

2. Array

Chapter1: Introduction to Data Structures

- **Data Structures**
- **Classification of Data Structure-**
  - Primitive Data types
  - Abstract Data types
  - Data type v/s File organisation
- **Operation on Data Structure**
- **Importance of Algorithm analysis**
- **Complexity of an Algorithm-**
  - Asymptotic analysis and Notations
  - Big O Notation
  - Big Omega Notation
  - Big Theta Notation
  - Rate of Growth and Big O Notation

INTRODUCTION TO DATA STRUCTURE

Data Structures

- Data structure is a data organisation, management and storage format that enables efficient access and modification.
- It is a way in which data is stored on a computer.
- Data structure is the most efficient way of searching, storing and organising data in a computer so that it can be retrieved and used most productively.
- Each data structure allows data to be stored in a specific manner.
- Specific data structures are decided to work for specific problems.

Algorithm

- It outlines the essential of a computational procedure, step by step instructions.

Program

- Implementation of algorithms in some programming languages.

Data Structure

- Organisation of data needed to solve the program.

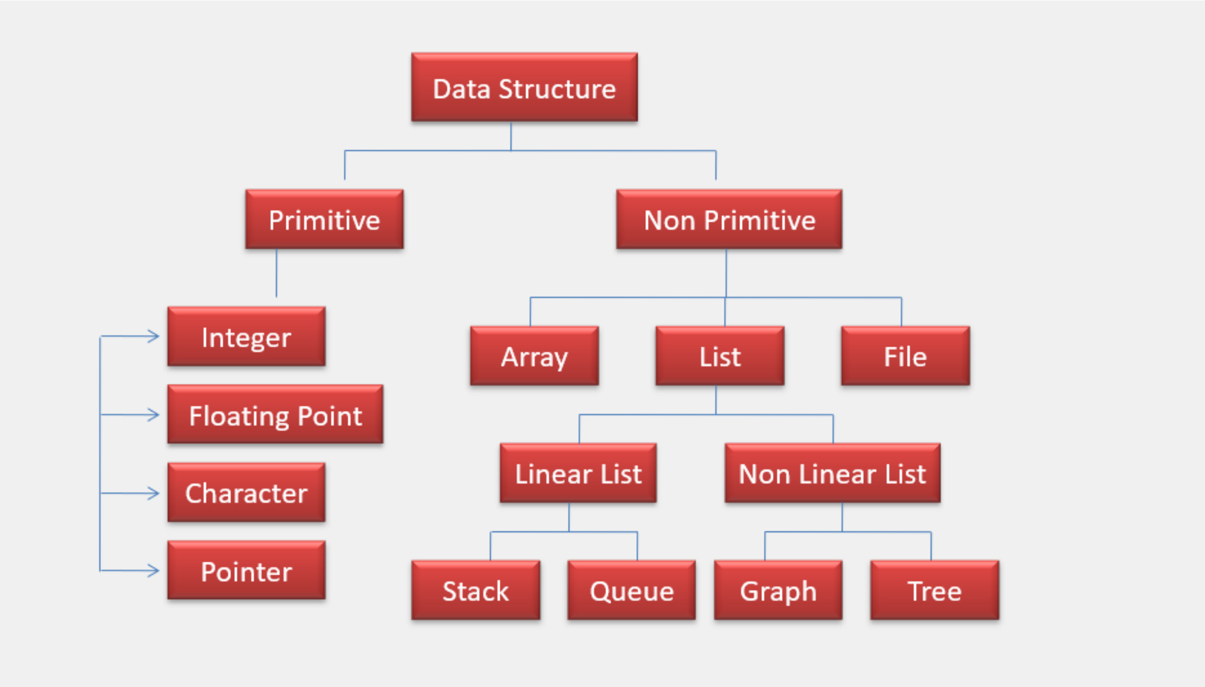
Algorithm	Pseudocode
● Logical approach, step by step procedure for computer program solving. ● Expressed using natural language, flowcharts etc.	● Methods for representing an algorithm. ● Includes control structures like loops, conditionals in human readable form.

Data structure mainly specifies the following things:

- Organization of Data.
- Accessing methods and Manipulating data methods.
- Representation of data in memory.

- Operations performed on that data.

Classification of Data Structure



Primitive data structure

- Primitive data structures are basic structures and are directly operated upon by machine instructions.

Non Primitive data structure

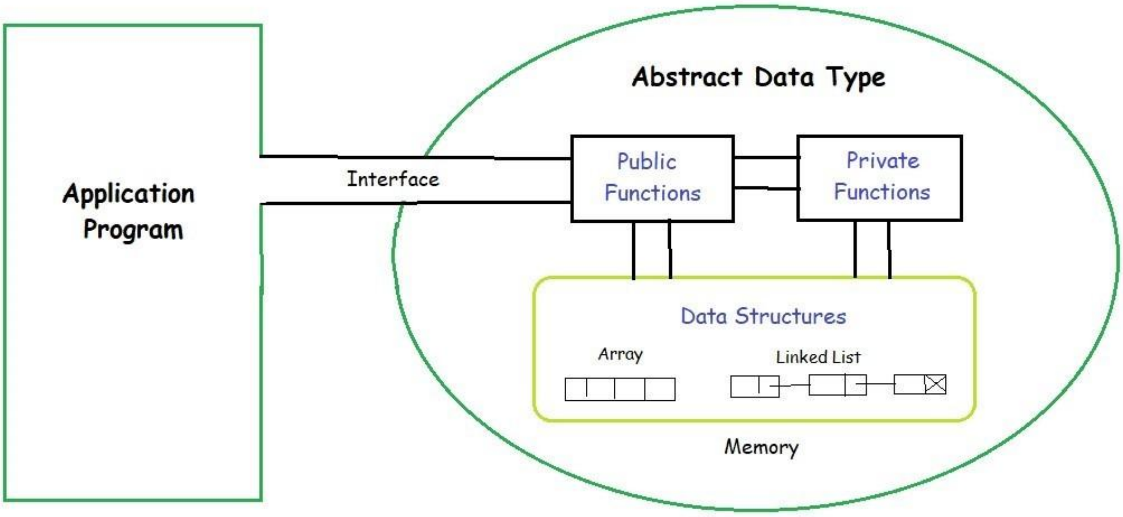
- Are advanced data types that store and manage data efficiently.

Linear Data Structure	Non Linear Data Structure
• Every item is related to its previous and next time. • Data is arranged in linear sequence. • Data items can be traversed in a single run.	• Every item is attached with many other items. • Data is not arranged in sequence. • Data can’t be traversed in a single run.

• Implementation is easy. • **Ex-** Array,Stacks, Queue and linked list.	• Implementation is difficult. • **Ex-** Tree,Graph.

Abstract Data Type

- Abstract Data Types are like user defined data types which define operations on values using functions without specifying what is there inside the function and how the operations are performed.
- **Ex-** Stack ADT.



Data Type	File Organization
• A data type is a classification that specifies which type of value a variable can hold, what operations can be performed on those values, and how they are stored in memory. • Data types help in organising and interpreting data, ensuring proper usage and manipulation of information within a program. • **Ex-** integers, floating-point numbers, characters, and user-defined types like structures and classes.	• File organisation refers to the way data is arranged and stored in a file. It defines how records are stored and accessed within a file. • File organisation is crucial for efficient data retrieval, insertion, and deletion operations. It influences how quickly and easily data can be accessed from a file. • **Ex-** sequential, indexed, and hashed. Each has its advantages and trade-offs in terms of speed, space efficiency, and ease of maintenance.

Operation on Data Structure

- **Create**
  - Allocate memory for data structure either during compile or run-time ensuring sufficient storage space.
- **Destroy**
  - Release allocated memory, preventing memory leaks and efficiently managing system resources.
- **Selection**
  - Access specific data elements based on conditions to retrieve or process relevant information.
- **Updation**
  - Modify or update data within a structure ensuring data remains accurate and relevant.
- **Searching**
  - Find data items or their locations based on specific criteria or search conditions.
- **Sorting**
  - Reorganise data items into a particular order for efficient retrieval.
- **Merging**
  - Combine data from two sorted lists into a single.
- **Splitting**
  - Divide a list into multiple lists or partitions.
- **Traversal**
  - Systematically visit each element within a structure, typically used for examining or processing data.

Importance of Algorithm Analysis

- **Efficiency enhancement**
    - Algorithm analysis helps select and design efficient algorithms, improving design efficient algorithms, improving software performance and responsiveness.
- **Resource optimization**
    - Efficient algorithms reduce CPU, memory and storage usage, benefiting resource constrained systems and cost savings.
- **Scalability**
    - Algorithms that scale well handle larger workloads, making them essential for big data and growing applications.
- **Predictable performance**
    - Understanding algorithm efficiency ensures predictable system behavior, crucial for quality or service and user experience.
- **Competitive advantage**
    - Superior algorithms offer a competitive edge, especially in industries where speed and cost effectiveness are critical.