PCET's

Pimpri Chinchwad College of Engineering

Department of Computer Engineering

B. Tech. (Computer Engineering)

Course: Data Structures and Algorithms

(BCE3401)

Prepared By: Prof. Meghana P. Lokhande

Faculty Name: Prof. Meghana P. Lokhande

Qualification: BE, MTech, PhD Pursuing

Location: 209FC

Contact No: 9823334381

Mail id: meghana.lokhande@pccoepune.org

Role: Course Teacher, Class Teacher

Syllabus

- **Unit I: Introduction to Data structures, Hashing:**

Types of Data Structure - Linear & Nonlinear, Static & Dynamic, Characteristics of algorithms, Analysis of algorithms – Frequency Count, Time & Space complexity Hashing: Concepts - Hash table, issues in hashing, hash functions- properties of good hash function, division, multiplication, extraction, mid-square, folding and universal, Collision resolution strategies- open addressing and chaining.

- **Unit II: Linked List:**

Dynamic Memory Management, Basics of Linked List, Comparison of sequential and linked organizations, Types of linked list, Singly linked list, Doubly linked list, Circular linked list. Applications: Polynomial Operations.

- **Unit III: Stack and Queue:**

Fundamentals of stack, Stack representation using array and linked List, Operations on stack. Applications: Recursion, Validity of parentheses, Expression Conversion. Fundamentals of queue, Queue representation using array and Linked List, Types of queue – Linear Queue, Circular Queue, Double Ended Queue, Priority Queue. Applications: Job Scheduling, Josephus problem.

- **Unit IV: Tree**

Basic terminology, representation using array and linked list, Recursive and Non recursive Tree Traversals, Operations on binary tree: Finding Height, Leaf nodes, counting no of Nodes, Construction of binary tree from traversals, Binary Search tree (BST): Insertion, deletion of a node from BST. Threaded Binary tree (TBT): Creation and traversals on TBT. Height Balanced Tree- AVL tree.

- **Unit V: Graph**

Basic Concepts, Storage representation, Adjacency matrix, adjacency list, adjacency multi list, inverse adjacency list. Traversals-depth first and breadth first search,

- **Unit VI: Sorting Techniques & Multi way Trees:**

Sorting methods- Quick sort and Merge Sort, Radix Sort, Heap sort, Shell sort. Multi way Trees: B tree, B+ tree.

Books:

1. Ellis Horowitz, Sartaj Sahni, Dinesh Mehta, “Fundamentals of Data Structures in C++”, University Press(India) Pvt. Ltd., 2nd Edition, 2008, ISBN-10: 8173716064/ ISBN-13:978-8173716065.
2. Varsha H. Patil, "Data Structures using C++", Oxford University Press, 1st Edition, 2012,ISBN-10: 0-19-806623-6/ ISBN-13: 978-0-19-806623-1.

Teaching Scheme and Exam

- Teaching: 4 Hours/week
- Credits to earn: 4 Credits
- Lab: 2 turns/week
- Exam:

IE	MTE	ETE	Total
20	30	50	100

Course Outcomes

At the end of this course, you will be able to

- Develop logic building skills to solve real life problems using various data structures and algorithms
- Apply linear data structures to solve various computing problems.
- Apply nonlinear data structures such as trees and graphs to solve various computing problems.
- Analyze and apply various sorting and hashing techniques to solve computing problems.
- Evaluate algorithms and data structures in terms of time and memory complexity of basic operations.
- Select appropriate data structure and demonstrate a working solution for a given problem.

.

Introduction to Data Structure

- Computer is an electronic machine which is used for data processing and manipulation.
- When programmer collects such type of data for processing, he would require to store all of them in computers main memory.
- In order to make how computer work we need to know

  Representation of data in computer.

  Accessing of data.

  How to solve problem step by step.

- For doing all of this task we used Data Structure

Data Structure

Introduction to Data Structures

- A data structure is a way of storing data in a computer so that it can be used efficiently and it will allow the most efficient algorithm to be used.
- A data structure should be seen as a logical concept that must address two fundamental concerns.

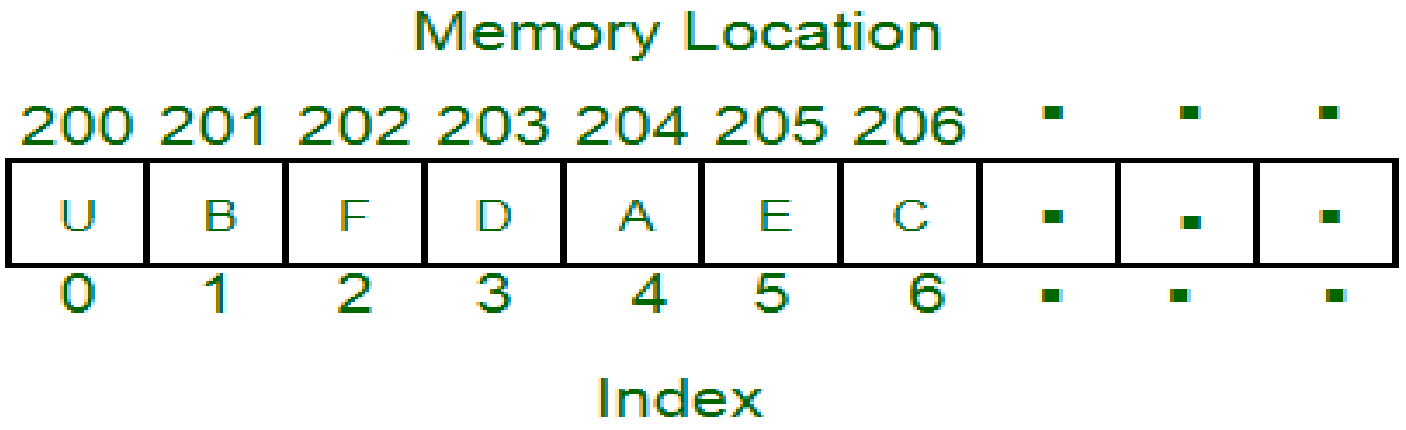
I. First, how the data will be stored, and

II. Second, what operations will be performed on it.

Data Structure

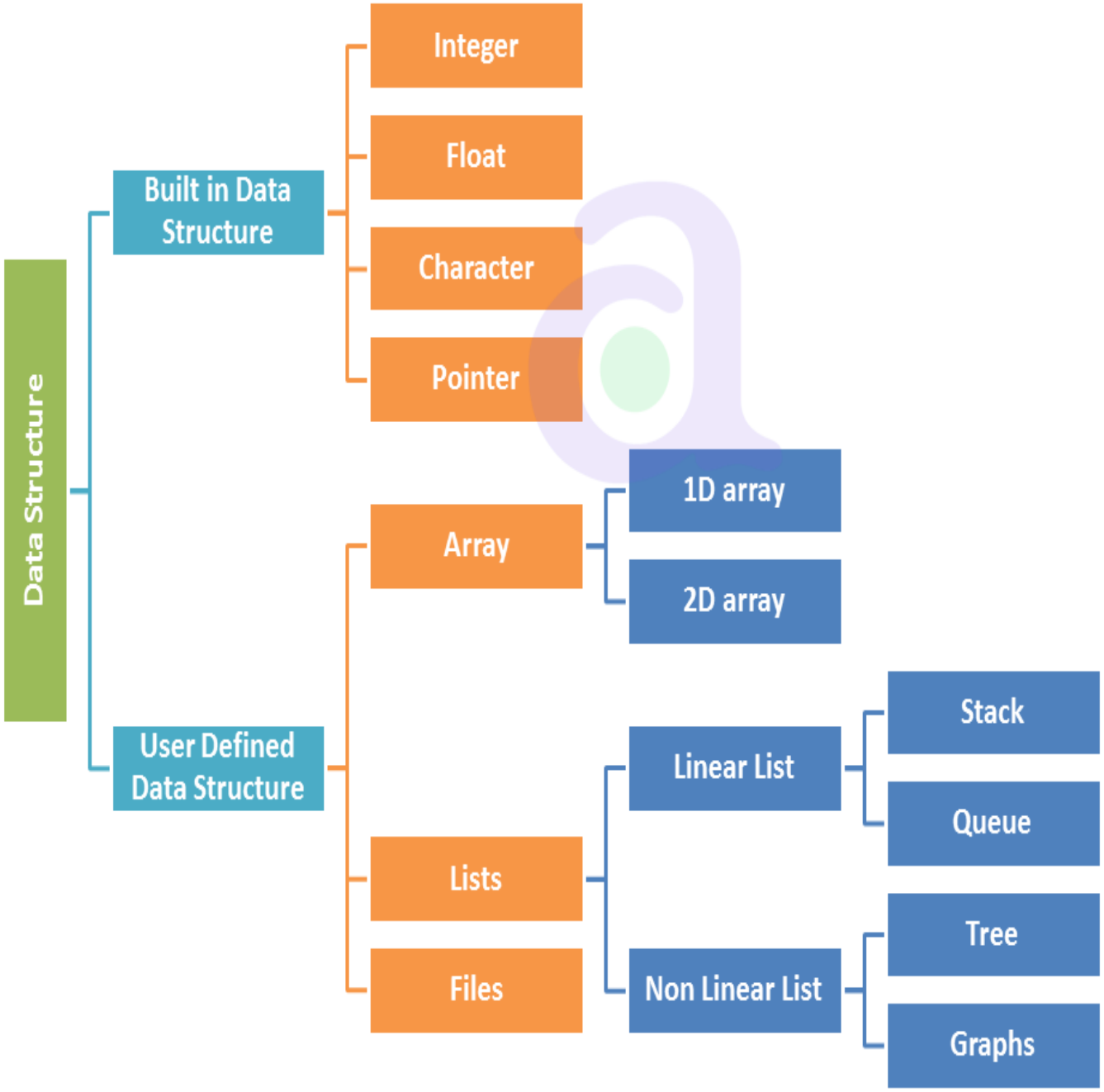
- A **data structure** is a particular way of organizing data in a computer so that it can be used effectively.

- For example, we can store a list of items having the same data-type using the *array* data structure.



Data Structure

- examples of Data Structures are arrays, Linked List, Stack, Queue, etc.
- Data Structures are widely used in almost every aspect of Computer Science i.e. Operating System, Compiler Design, Artificial intelligence, Graphics and many more.
- It plays a vital role in enhancing the performance of a software or a program as the main function of the software is to store and retrieve the user's data as fast as possible



Types Of DS

The DS are divided into two types:

1) Primitive
2) Non primitive

Non primitive divided into two type

1) Linear DS
2) Non linear DS

Primitive Data Structure

- Primitive Data Structure are basic structure and directly operated upon by machine instructions.
- Primitive data structures have different representations on different computers.
- Integers, floats, character and pointers are example of primitive data structures.
- These data types are available in most programming languages as built in type.

Integer: It is a data type which allows all values without fraction part. We can used it for whole numbers.

Float: It is a data type which is use for storing fraction numbers.

Character: It is a data type which is used for character values.

Pointer: A variable that hold memory address of another variable are called pointer.

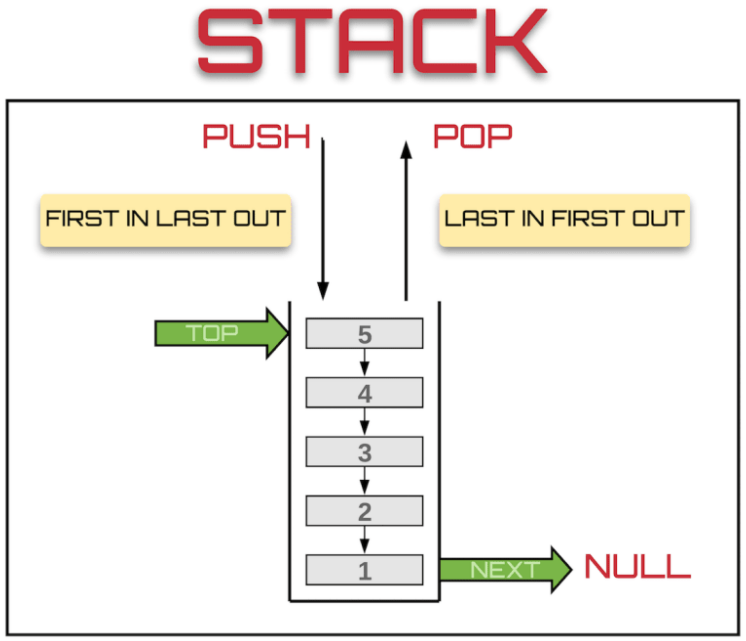
Non Primitive Data Type

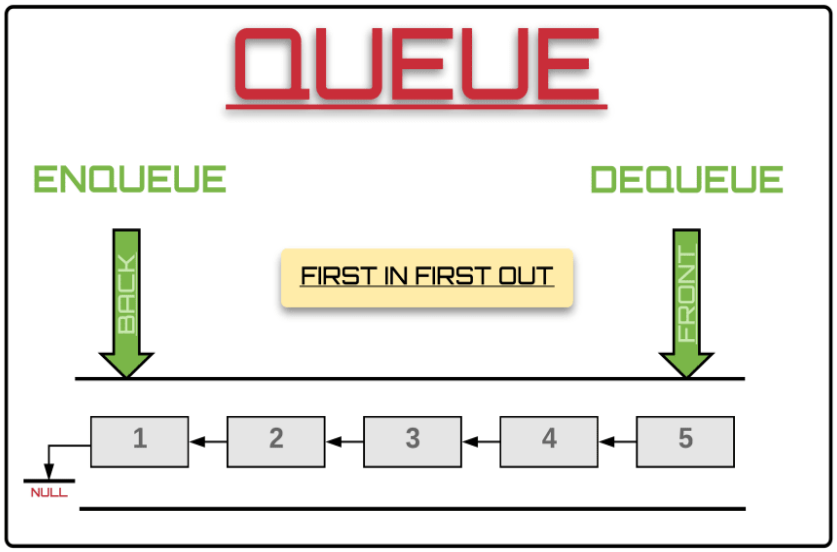
- These are more sophisticated data structures.
- These are derived from primitive data structure.
- The non – primitive data structures emphasize structuring of a group of homogeneous or heterogeneous data items.
- Example of non – primitive data types are Array, List, and File etc.
- A non – primitive data type is further divided into Linear and non – Linear data structure.

Array: An array is a fixed size sequenced collection of elements of the same data type.

List: An ordered set containing variable number of elements is called as List.

File: A file is a collection of logically related information. It can be viewed as a large list of records consisting of various fields.





- Example of Linear data structure are Stack and Queue

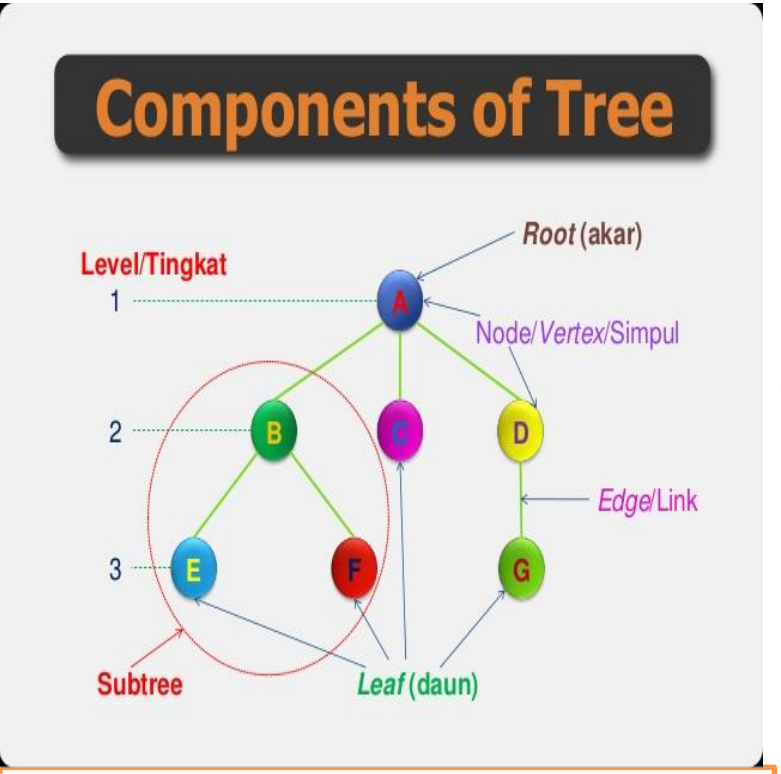
Stack

- Stack is a data structure in which insertion and deletion operations are performed at one end only.
- The insertion operation is referred to as 'PUSH' and deletion is referred as 'POP' operation
- Stack is also called as Last In First Out (LIFO) data structure.

Queue

- The data structure which permits the insertion at one and deletion at another end, known as Queue.
- End at which deletion occurs is known as FRONT end and another end at which insertion occurs is known as REAR end.
- Queue is also called as First In First Out (FIFO)

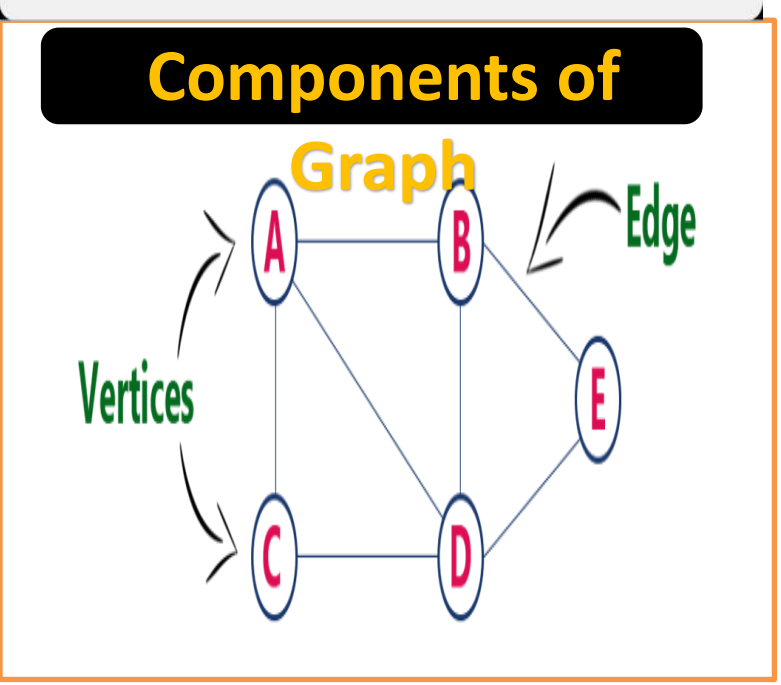
Non-Linear Data Structure



- Non linear DS are those data structure in which data items are not arranged in a sequence.
- Example on Non Linear DS are Tree and Graph.

TREE

- A Tree can be define as finite data items (nodes) in which data items are arranged in branches and sub branches
- Tree represent the hierarchical relationship between various elements
- Tree consist of nodes connected by edge, the represented by circle and edge lives connecting to circle.



Graph

- Graph is collection of nodes (information) and connecting edges (Logical relation) between nodes.
- A tree can be viewed as restricted graph
- Graph have many types:

1) Simple graph 2) Mixed graph 3) Multi graph 4) Directed graph 5) Un-directed graph

Difference Between Linear and Non Linear Data Structure

Linear Data Structure

- Every item is related to its previous and next item.
- Data is arranged in linear sequence.
- Data items can be traversed in a single run
- E.g. Array, Stacks, Linked list, Queue
- Implementation is easy.

Non – Linear Data Structure

- Every item is attached with many other items.
- Data is not arranged in sequence.
- Data cannot be traversed in a single run.
- E.g. Tree, Graph
- Implementation is difficult.

Classification of Data Structures

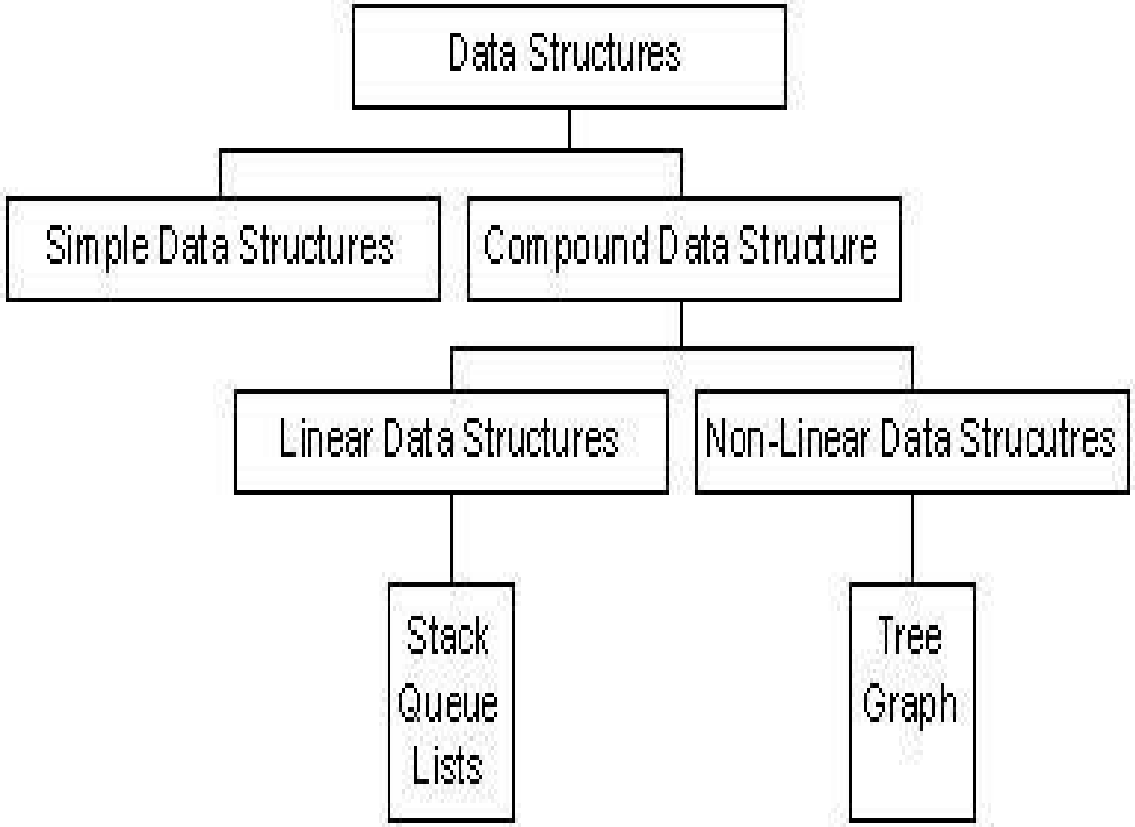
Data structures can be classified as

i. Simple data structure

ii. Compound data structure

iii. Linear data structure

iv. Non linear data structure



Simple and Compound Data Structures

Simple Data Structure:

- Simple data structure can be constructed with the help of primitive data structure.
- A primitive data structure used to represent the standard data types of any one of the computer languages.
- Variables, arrays, pointers, structures, unions, etc. are examples of primitive data structures.

Compound Data structure:

- Compound data structure can be constructed with the help of any one of the primitive data structure and it is having a specific functionality.
- It can be designed by user. It can be classified as

    i. Linear data structure

    ii. Non-linear data structure

Linear and Non-linear Data Structures

Linear Data Structure:

- Linear data structures can be constructed as a continuous arrangement of data elements in the memory.
- It can be constructed by using array data type.
- In the linear Data Structures the relationship of adjacency is maintained between the data elements.

Non-linear Data Structure:

- Non-linear data be constructed as a collection of randomly distributed set of data item joined together by using a special pointer (tag).
- In non-linear Data structure the relationship of adjacency is not maintained between the data items.

Operations on Data Structures

i. Add an element

ii. Delete an element

iii. Traverse

iv. Sort the list of elements

v. Search for a data element

Algorithm

- An **Algorithm** may be defined as a finite sequence of instructions each of which has a clear meaning and can be performed with a finite amount of effort in a finite length of time.
- The word algorithm originates from the Arabic word **Algorism** which is linked to the name of the Arabic Mathematician **AI Khwarizmi**.
- **AI Khwarizmi** is considered to be the first algorithm designer for adding numbers.

Structure of an Algorithm

An algorithm has the following structure:

–Input Step

–Assignment Step

–Decision Step

–Repetitive Step

–Output Step

Properties of an Algorithm

- **Finiteness:-** An algorithm must terminate after finite number of steps.
- **Definiteness:-**The steps of the algorithm must be precisely defined.
- **Generality:-** An algorithm must be generic enough to solve all problems of a particular class.
- **Effectiveness:-** The operations of the algorithm must be basic enough to be put down on pencil and paper.
- **Input-Output:-** The algorithm must have certain initial and precise inputs, and outputs that may be generated both at its intermediate and final steps

Algorithm Analysis and Complexity

- The performances of algorithms can be measured on the scales of **Time and Space**.
- The **Time Complexity** of an algorithm or a program is a function of the running time of the algorithm or a program.
- The **Space Complexity** of an algorithm or a program is a function of the space needed by the algorithm or program to run to completion.