# INTRODUCTION TO DATA STRUCTURE

## 1.1 What is Data Structure?

- **Data Structure:** Data structure is a way to storing and organizing data in a computer so that it can be used efficiently.
- Data structure is used in almost all program and software system.
- Data structure usually consist of two things:

  **1.** The type of structure that we are going to use stores the data.

  **2.** How efficiently we perform operations onto that data stored in structure so we can reduce execution time and amount of memory space.

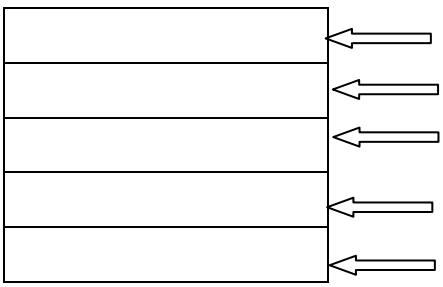
## 1.2 Data and Information.

- **Data:** - Data is a collection of raw facts(or information) and it may or may not be meaningful.
- **Information: -** meaningful data is known as Information.
- **Example:**
  - Program- set of instruction.
  - 67.8- Weight of person.
  - 13/3/1999- Date of birth of a person.

## 1.3 Define following terms:

**(a) Cell:**

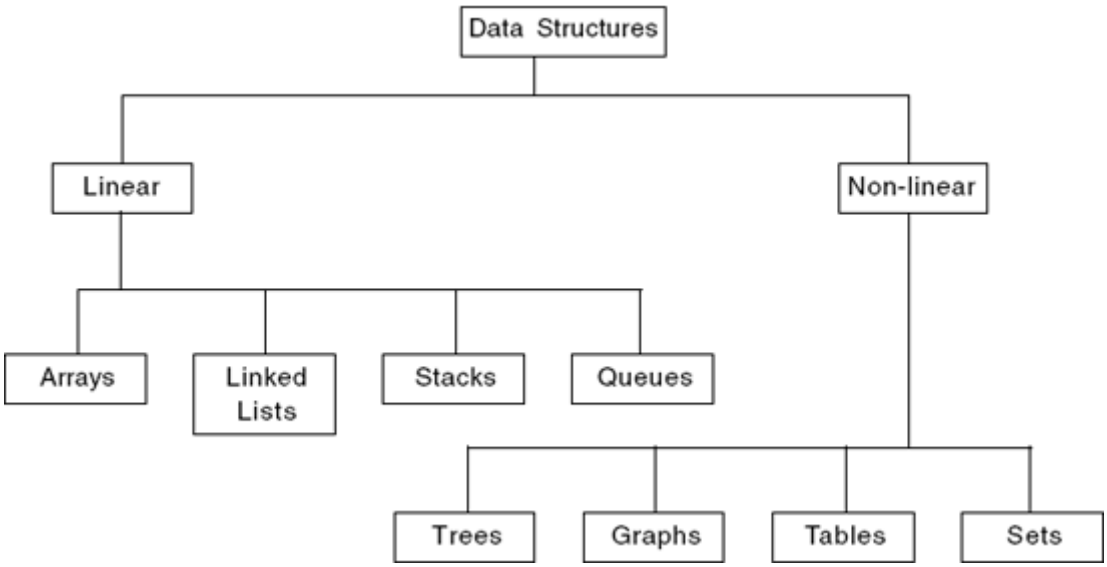
- The smallest fundamental structural unit which represents a data entity.
- Cell is a memory location which stores elements of data items.



**(b) Field:** Field is used to store particular kind of data.

**(c) Record:** a record is a collection of related data items. For example: Employee details

## 1.4 Types of Data Structure




**Primitive Data Structure:**

- The Data structures that are directly processed by machine using its instructions are known as primitive data structure.
- Following are the primitive data structure:

  **Integer:**

  - Integer represents numerical values which are whole quantities.
  - The number of objects are countable can be represented by an integer.
  - The set of integer is:{.....-(n+1), -n , ..... , -2, -1,0,1,2,…….n,n+1}
  - The sign and magnitude method is used to represent integer numbers. In this method place a sign symbol in front of the number.
  - Ex: 1000

  **Real:**

  - The number having fractional part i.e decimal point is called real number.
  - Common way of representing real number is normalized floating point representation.
  - In this method the real number is expressed as a combination of mantissa and exponent.
  - Ex: 123.45

**Character:**

- Character data structure is used to store nonnumeric information.
- It can be letters [A-Z], [a-z], operators and special symbols.
- A character is represented in memory as a sequence bits.
- Two most commonly known character set supported by computer are ASCII and EBCDIC.

**Logical:**

- A logical data item is a primitive data structure that can assume the values of either "true" or "false".
- Most commonly used logical operators Are AND, OR and NOT.

**Pointer:**

- Pointer is a variable which points to the memory address. This memory address is the location of other variable in memory.
- It provides homogeneous method of referencing any data structure, regardless of the structure's type or complexity.
- Another characteristic is that it provides faster insertion and deletion of elements.

**Non Primitive Data Structure:**

- The data structures that are not directly processed by machine using its instructions are known as non primitive data structure.
- Following are the non primitive data structure:

**Array:**

- Array is an ordered set which consist of a fixed number of object.
- Insertion and deletion operation can be performed on array.
- We can only change the value of the element in array.

**List:**

- List is an ordered set which consist of variable number of elements or object.
- Insertion and deletion can be performed on list.

**File:**

- A file is a large list that is stored in the external memory of computer.
- A file may be used as a repository for list items commonly called records.

- **Non Primitive Data Structure is classified into two categories:**
  1) Linear Data structure
  2) Non linear Data structure

## 1.5 Linear and Non Linear data structure.

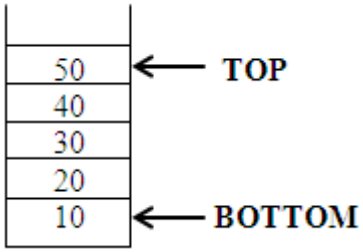
### Linear Data Structure:

- In the linear data structure processing of data items is possible in linear fashion.
- Data are processed one by one sequentially.
- Examples of the linear data structure are:

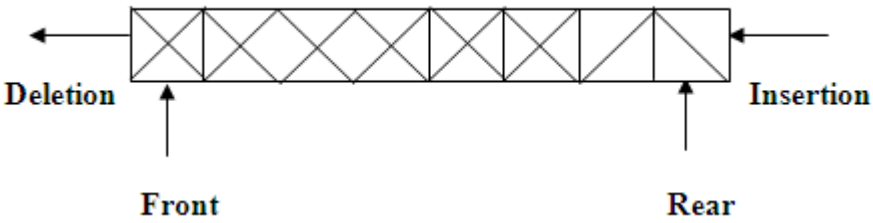
  (A) Array (B) Stack (C) Queue (D) Linked List

**(A) Array:** Array is an ordered set which consist of a fixed number of object.

**(B) Stack:** A stack is a linier list in which insertion and deletion operations are performed at only one end of the list.




**(C) Queue:** A queue is a linier list in which insertion is performed at one end called rear and deletion is performed at another end of the list called front.

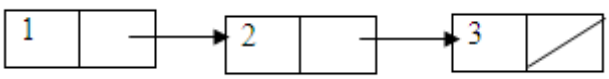



**(D) Linked list:** A linked list is a collection of nodes.

Each node has two fields:

**(i) Information**

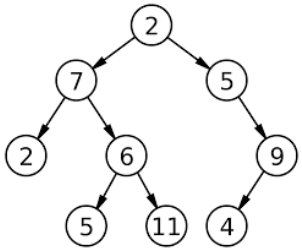
**(ii) Address, which contains the address of the next node.**



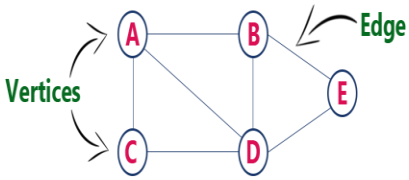

### Non Linear data structure:

- In the Non linier data structure processing of data items is not possible in linier fashion.
- Examples of the non linier data structure are:

**(A) Tree:** In tree data contains hierarchical relationship.




**(B) Graph:** this data structure contains relationship between pairs of elements.




## 1.6 Operations on Data Structure:

1. **Traversing:** Access each element of the data structure and doing some processing over the data.
2. **Inserting:** Insertion of new elements into the data structure.
3. **Deleting:** Deletion of specific elements.
4. **Searching:** Searching for a specific element.
5. **Sorting:** Sorting the data in ascending or descending ordered.
6. **Merging:** Combination of two data structure.

## 1.7 Algorithms:

- Algorithm is a stepwise solution to a problem.
- **Algorithm** is a finite set of instructions that is followed to accomplish a particular task.
- In mathematics and computing, an algorithm is a procedure for accomplishing some task which will terminate in a defined end-state.
- The computational complexity and efficient implementation of the algorithm are important in computing, and this depends on suitable data structure.

- **<u>Example:</u>**

  **Write Algorithm to compute the average of n elements.**

  **Steps:**

  **ALGORITHM: FIND AVERAGE OF N NUMBER**

  Assume that array contains **n** integer values.

  1. [INITIALIZE] **sum ←0.**
  2. [PERFORM SUM OPERATION]

     **Repeat steps from i=0 to n-1**

     **sum←sum+a[i]**
  3. [CALCULATE AVERAGE]

     **avg←sum/n**

4. Write **avg**
5. [FINISHED]

**Exit**

- **<u>Properties required for algorithm.</u>**
  1. **Finiteness:** "An algorithm must always terminate after a finite number of steps".
  2. **Definiteness:** "Each steps of algorithm must be precisely defined".
  3. **Input:** "quantities which are given to it initially before the algorithm begins".
  4. **Output:** "quantities which have a specified relation to the input".
  5. **Effectiveness:** "all the operations to be performed in the algorithm must be sufficient".
- **<u>Complexity of Algorithms</u>**
  - **Time complexity**: Retuning time of the program.
  - **Space complexity**: Amount of computer memory required during the program execution.

### 1.8 Difference between List and Array

| List | Array |
|---|---|
| (1) No. of elements in a list are not fixed (variable).(Dynamic Data structure) | **(1)** No. of elements in an array is fixed. (Static Data structure) |
| **(2)**It uses pointer variable which occupies an extra memory space. | **(2)** It does not use pointer variable so it does not occupies extra memory space. |
| **(3)** Insertion and deletion operation are very easy to perform. | **(3)** Insertion and deletion operation are very difficult to perform. |
| **(4)** There are two types of List:<br>1. Linear List<br>2. Non Linear List | **(4)** There are two types of array:<br>1. One dimensional array<br>2. Two dimensional array |
| **(5)** Searching is slower in case of List. | **(5)** Searching is faster in case of array. |

**1.9 Difference between Static Memory allocation and Dynamic Memory allocation.**

| Static Memory Allocation | Dynamic Memory Allocation |
|---|---|
| **(1)** Static Memory Allocation is Performed at Compile Time. | **(1)** Dynamic Memory Allocation is Performed at Runtime. |
| **(2)** In Static Memory Allocation Memory may be wasted. | **(2)** In Dynamic Memory Allocation memory is allocated when it is needed. So there is no wastage of memory. |
| **(3)** Array is an example of static memory allocation. | **(3)** Linked List is an example of dynamic memory allocation. We can use malloc () function to allocate memory at runtime |
| **(4)** In static memory allocation memory is allocated sequentially | **(4)** In dynamic memory allocation memory is not allocated sequentially. |