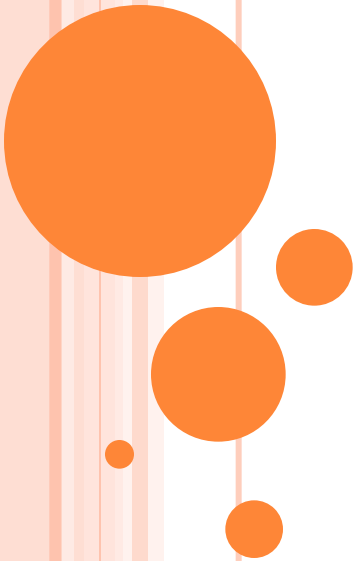
DATA STRUCTURES AND ALGORITHMS



DEFINITION

- Data structure is representation of the logical relationship existing between individual elements of data.
- In other words, a data structure is a way of organizing all data items that considers not only the elements stored but also their relationship to each other.

INTRODUCTION

- Data structure affects the design of both structural & functional aspects of a program.

  Program=algorithm + Data Structure

- You know that a algorithm is a step by step procedure to solve a particular function.

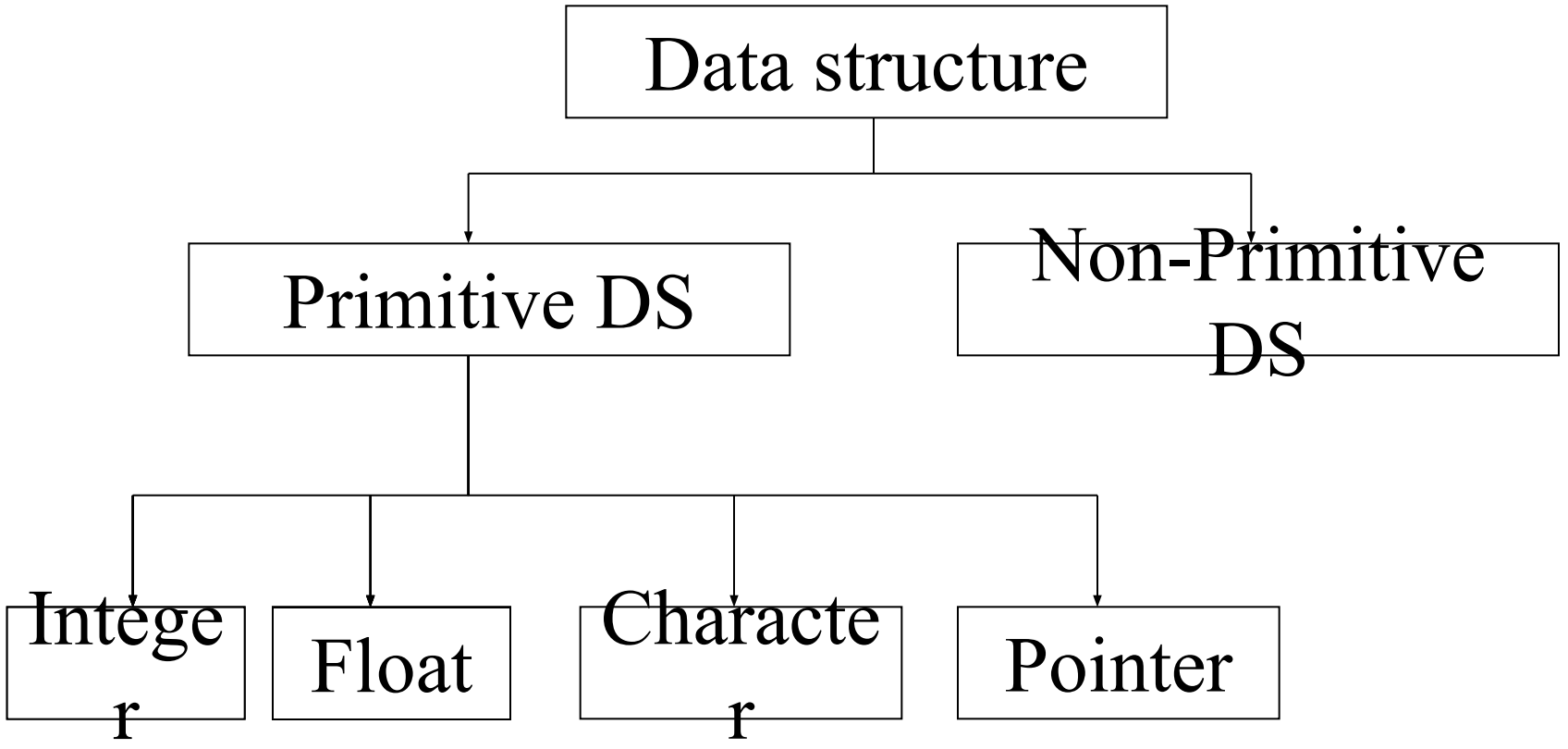
INTRODUCTION

- That means, algorithm is a set of instruction written to carry out certain tasks & the data structure is the way of organizing the data with their logical relationship retained.
- To develop a program of an algorithm, we should select an appropriate data structure for that algorithm.
- Therefore algorithm and its associated data structures from a program.

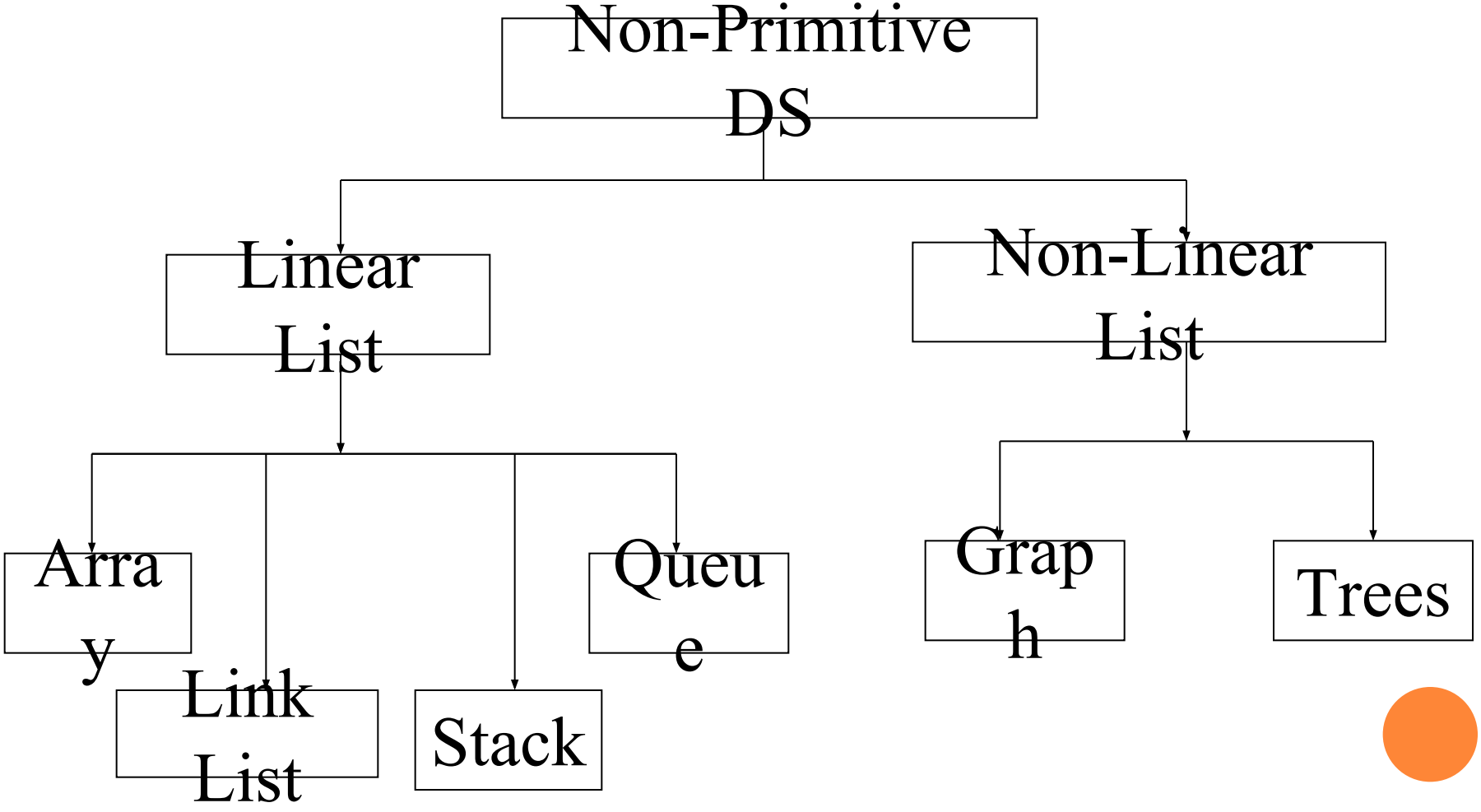
CLASSIFICATION OF DATA STRUCTURE

- Data structure are normally divided into two broad categories:
  - Primitive Data Structure
  - Non-Primitive Data Structure

CLASSIFICATION OF DATA STRUCTURE



CLASSIFICATION OF DATA STRUCTURE



PRIMITIVE DATA STRUCTURE

- There are basic structures and directly operated upon by the machine instructions.
- In general, there are different representation on different computers.
- Integer, Floating-point number, Character constants, string constants, pointers etc, fall in this category.

NON-PRIMITIVE DATA STRUCTURE

- There are more sophisticated data structures.
- These are derived from the primitive data structures.
- The non-primitive data structures emphasize on structuring of a group of homogeneous (same type) or heterogeneous (different type) data items.

NON-PRIMITIVE DATA STRUCTURE

- Lists, Stack, Queue, Tree, Graph are example of non-primitive data structures.
- The design of an efficient data structure must take operations to be performed on the data structure.

NON-PRIMITIVE DATA STRUCTURE

- The most commonly used operation on data structure are broadly categorized into following types:
  - Create
  - Selection
  - Updating
  - Searching
  - Sorting
  - Merging
  - Destroy or Delete

DIFFERENT BETWEEN THEM

- A primitive data structure is generally a basic structure that is usually built into the language, such as an integer, a float.
- A non-primitive data structure is built out of primitive data structures linked together in meaningful ways, such as a or a linked-list, binary search tree, AVL Tree, graph etc.

DESCRIPTION OF VARIOUS DATA STRUCTURES : ARRAYS

- An array is defined as a set of finite number of homogeneous elements or same data items.
- It means an array can contain one type of data only, either all integer, all float-point number or all character.

ARRAYS

- Simply, declaration of array is as follows:

  int arr[10]

- Where int specifies the data type or type of elements arrays stores.
- “arr” is the name of array & the number specified inside the square brackets is the number of elements an array can store, this is also called sized or length of array.

ARRAYS

- Following are some of the concepts to be remembered about arrays:
  - The individual element of an array can be accessed by specifying name of the array, following by index or subscript inside square brackets.
  - The first element of the array has index zero[0]. It means the first element and last element will be specified as:arr[0] & arr[9]

    Respectively.

ARRAYS

- The elements of array will always be stored in the consecutive (continues) memory location.
- The number of elements that can be stored in an array, that is the size of array or its length is given by the following equation:

  (Upperbound-lowerbound)+1

ARRAYS

- For the above array it would be

  (9-0)+1=10,where 0 is the lower bound of array and 9 is the upper bound of array.
- Array can always be read or written through loop. If we read a one-dimensional array it require one loop for reading and other for writing the array.

ARRAYS

- For example: Reading an array

```
For(i=0;i<=9;i++)
    scanf("%d",&arr[i]);
```

- For example: Writing an array

```
For(i=0;i<=9;i++)
    printf("%d",arr[i]);
```

ARRAYS

- If we are reading or writing two-dimensional array it would require two loops. And similarly the array of a N dimension would required N loops.
- Some common operation performed on array are:
  - Creation of an array
  - Traversing an array

ARRAYS

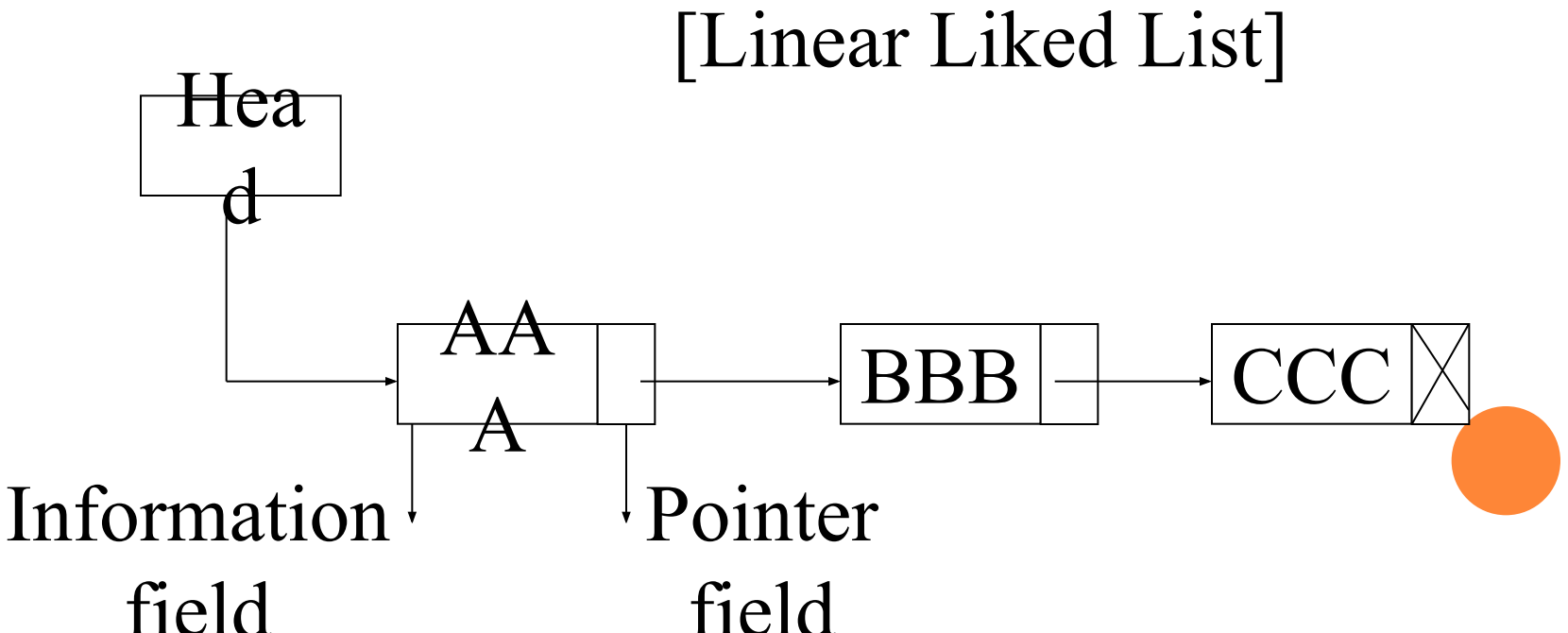
- Insertion of new element
- Deletion of required element
- Modification of an element
- Merging of arrays

LISTS

- A lists (Linear linked list) can be defined as a collection of variable number of data items.
- Lists are the most commonly used non-primitive data structures.
- An element of list must contain at least two fields, one for storing data or information and other for storing address of next element.
- As you know for storing address we have a special data structure of list the address must be pointer type.

LISTS

- Technically each such element is referred to as a node, therefore a list can be defined as a collection of nodes as show bellow:



LISTS

- Types of linked lists:
  - Single linked list
  - Doubly linked list
  - Single circular linked list
  - Doubly circular linked list

STACK

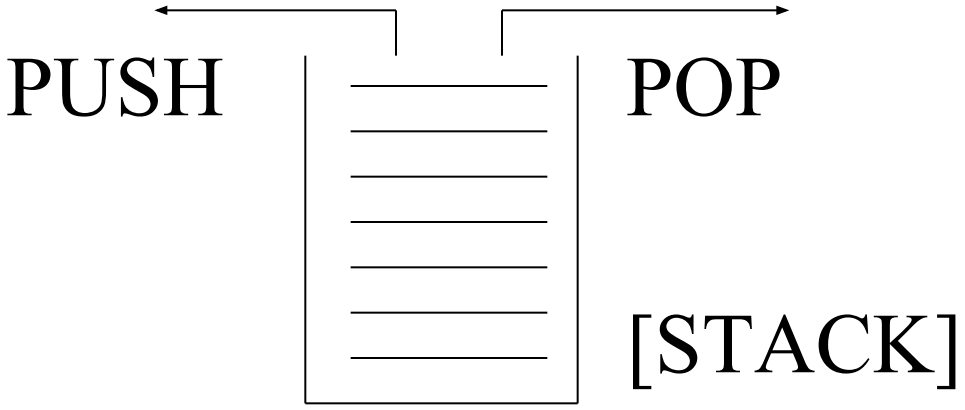
- A stack is also an ordered collection of elements like arrays, but it has a special feature that deletion and insertion of elements can be done only from one end called the top of the stack (TOP)
- Due to this property it is also called as last in first out type of data structure (LIFO).

STACK

- It could be through of just like a stack of plates placed on table in a party, a guest always takes off a fresh plate from the top and the new plates are placed on to the stack at the top.
- It is a non-primitive data structure.
- When an element is inserted into a stack or removed from the stack, its base remains fixed where the top of stack changes.

STACK

- Insertion of element into stack is called PUSH and deletion of element from stack is called POP.
- The bellow show figure how the operations take place on a stack:



STACK

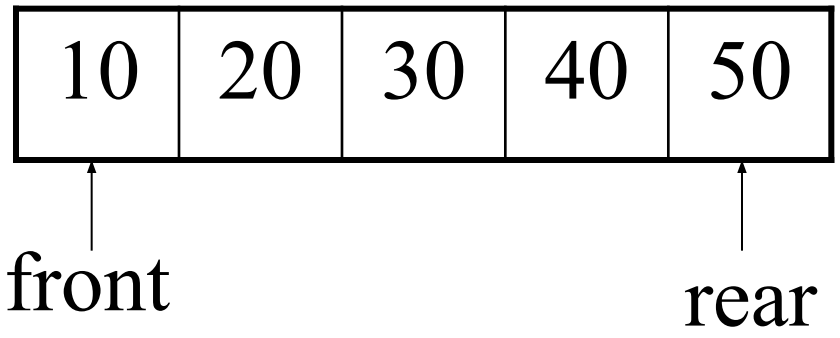
- The stack can be implemented into two ways:
  - Using arrays (Static implementation)
  - Using pointer (Dynamic implementation)

QUEUE

- Queue are first in first out type of data structure (i.e. FIFO)
- In a queue new elements are added to the queue from one end called REAR end and the element are always removed from other end called the FRONT end.
- The people standing in a railway reservation row are an example of queue.

QUEUE

- Each new person comes and stands at the end of the row and person getting their reservation confirmed get out of the row from the front end.
- The bellow show figure how the operations take place on a stack:



QUEUE

- The queue can be implemented into two ways:
  - Using arrays (Static implementation)
  - Using pointer (Dynamic implementation)