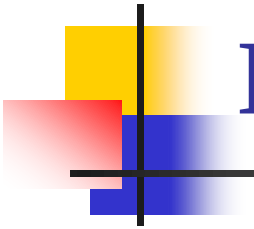
Introduction to Data Structure

israa Mahmoud hayder

Course Objectives
Objectives:

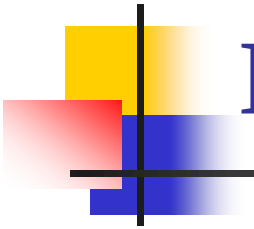
- 1. Encourages students to explore data structures by implementing them, a
- process through which students discover how data structures work and
- how they can be applied.
- 2. Students will be expected to write C++ language programs, ranging from
- very short programs to more elaborate systems.
- **Pre-requisite:**

• A good knowledge of C++ language, use of Function and structures.



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.

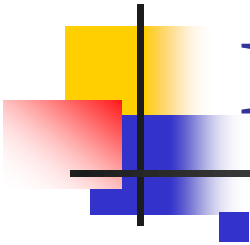


Introduction

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

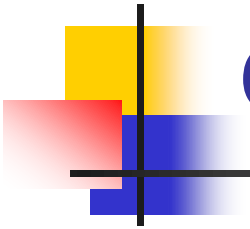
  Program=algorithm + Data Structure

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



Introduction

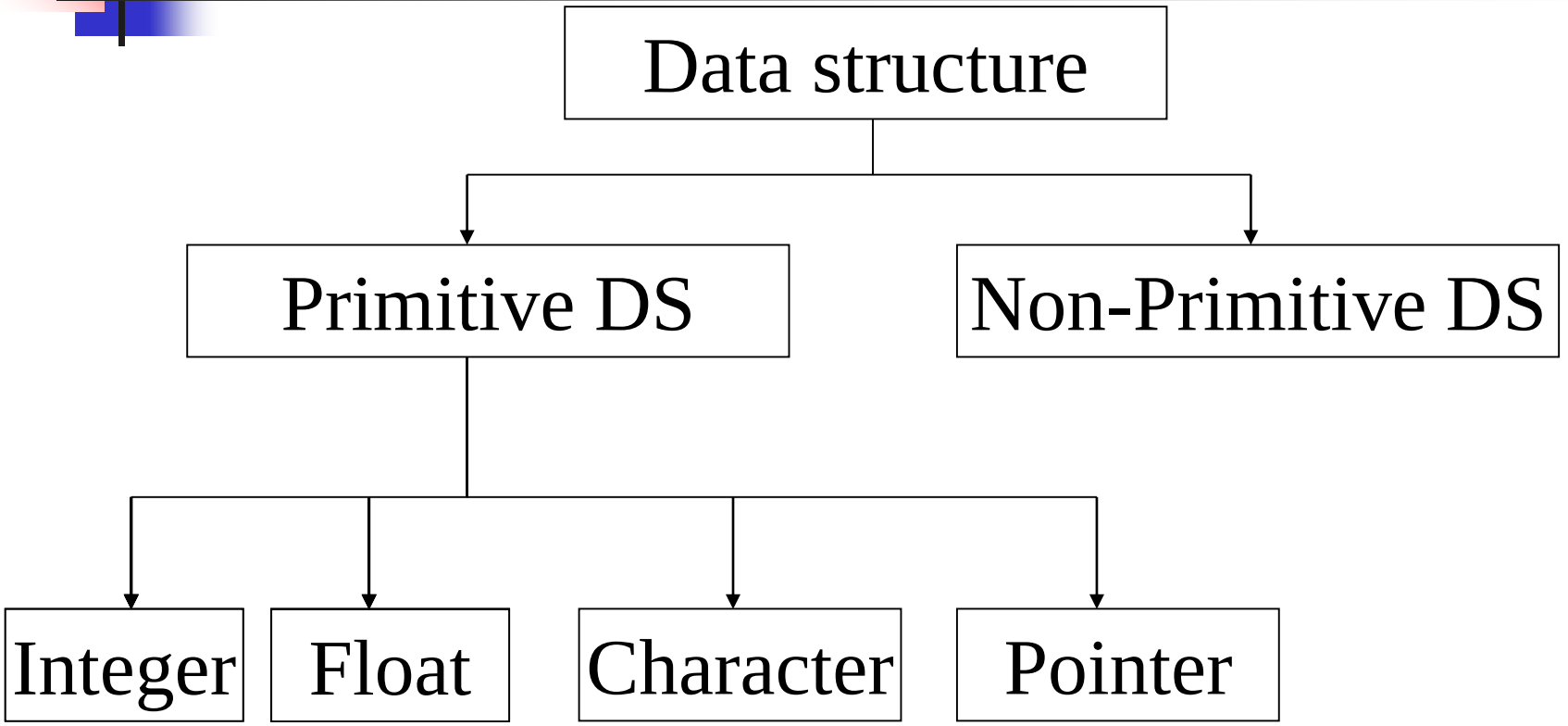
- That means, algorithm is a set of instruction written to carry out certain tasks & the data structure is the way of organizing the data with their logical relationship retained.
- To develop a program of an algorithm, we should select an appropriate data structure for that algorithm.
- Therefore algorithm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



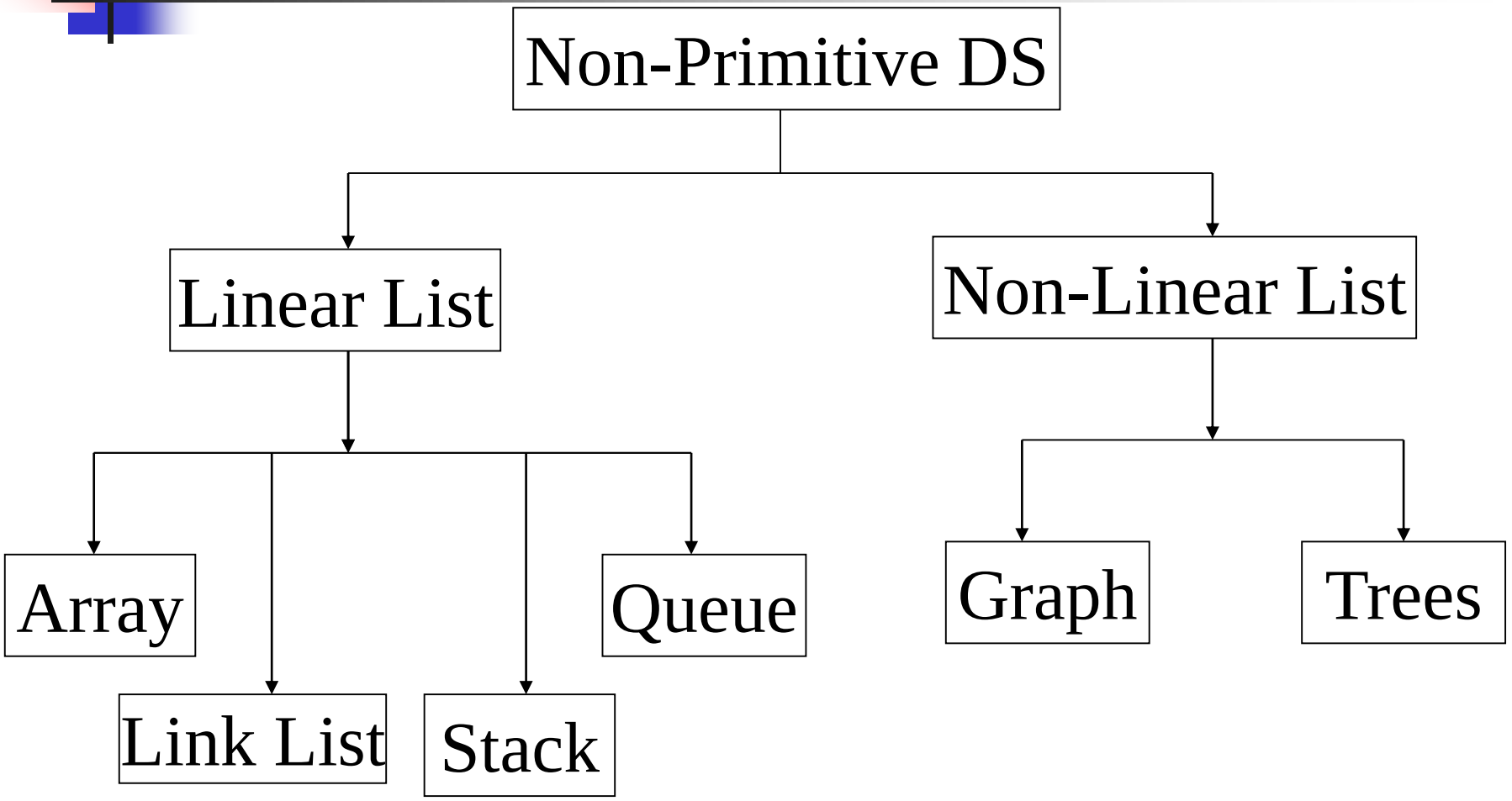
Classification of Data Structure

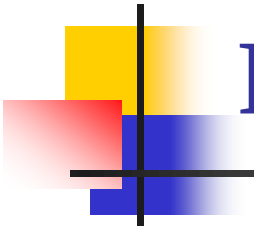
- Data structure are normally divided into two broad categories:
  - Primitive Data Structure
  - Non-Primitive Data Structure

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

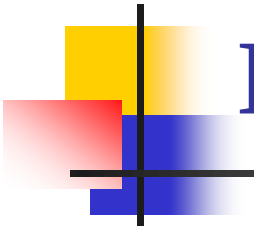
- There are more sophisticated data structures.
- These are derived from the primitive data structures.
- The non-primitive data structures emphasize on structuring of a group of homogeneous (same type) or heterogeneous (different type) data items.

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.

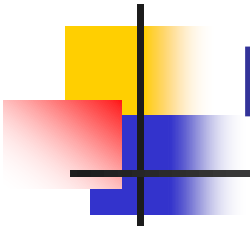
Non-Primitive Data Structure

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



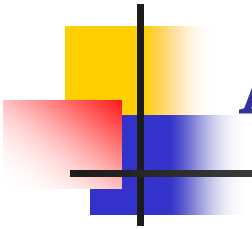
Different between them

- 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.



Linear Data Structure:: 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.

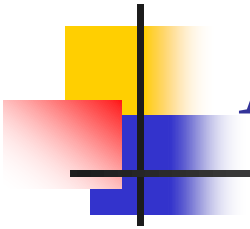


Arrays

- Simply, declaration of array is as follows:

  int arr[10]

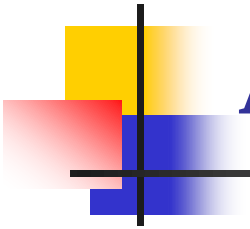
- Where int specifies the data type or type of elements arrays stores.
- “arr” is the name of array & the number specified inside the square brackets is the number of elements an array can store, this is also called sized or length of array.



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]

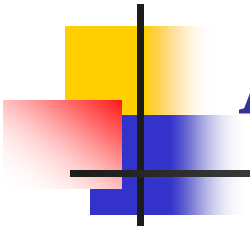
    Respectively.



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

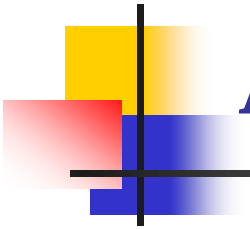


Arrays

- 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.



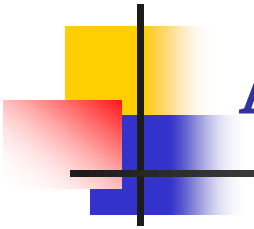
Arrays

- For example: Reading an array

```
For(i=0;i<=9;i++)
  cout << arr[i];
```

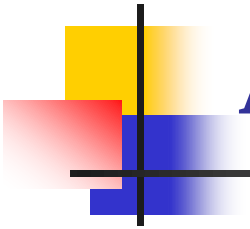
- For example: Writing an array

```
For(i=0;i<=9;i++)
   cin>> 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