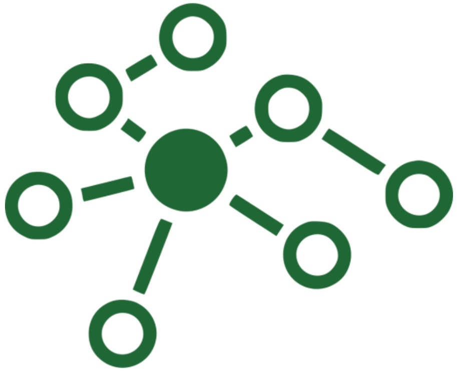
WHAT IS DATA?

- **Data** is the basic fact or entity that is utilized in calculation or manipulation.
- There are two different **types of data** **Numeric** data and **Alphanumeric** data.
- When a programmer collects such type of data for **processing**, he would require **to store them in computer’s main memory**.
- The process of storing data items in computer’s main memory is called ***representation.***
- **Data** to be processed must be **organized in a particular fashion**, these organization leads to structuring of data, and hence the mission to study the Data Structures.

WHAT IS DATA STRUCTURE?

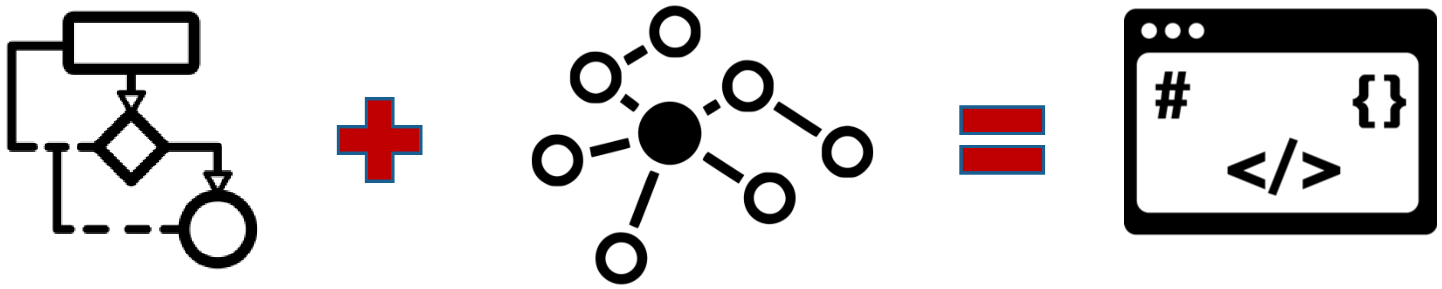
- ***Data Structure*** is a representation of the logical relationship existing between individual elements of data.
- In other words, a data structure is a **way of organizing all data items** that **considers** not only the **elements stored** but also their **relationship to each other.**
- We can also define data structure as a **mathematical or logical model** of a particular **organization** of **data items.**
- Data Structure mainly specifies the following four things
  - Organization of Data
  - Accessing Methods
  - Degree of Associativity
  - Processing alternatives for information



WHAT IS DATA STRUCTURE? CONT..

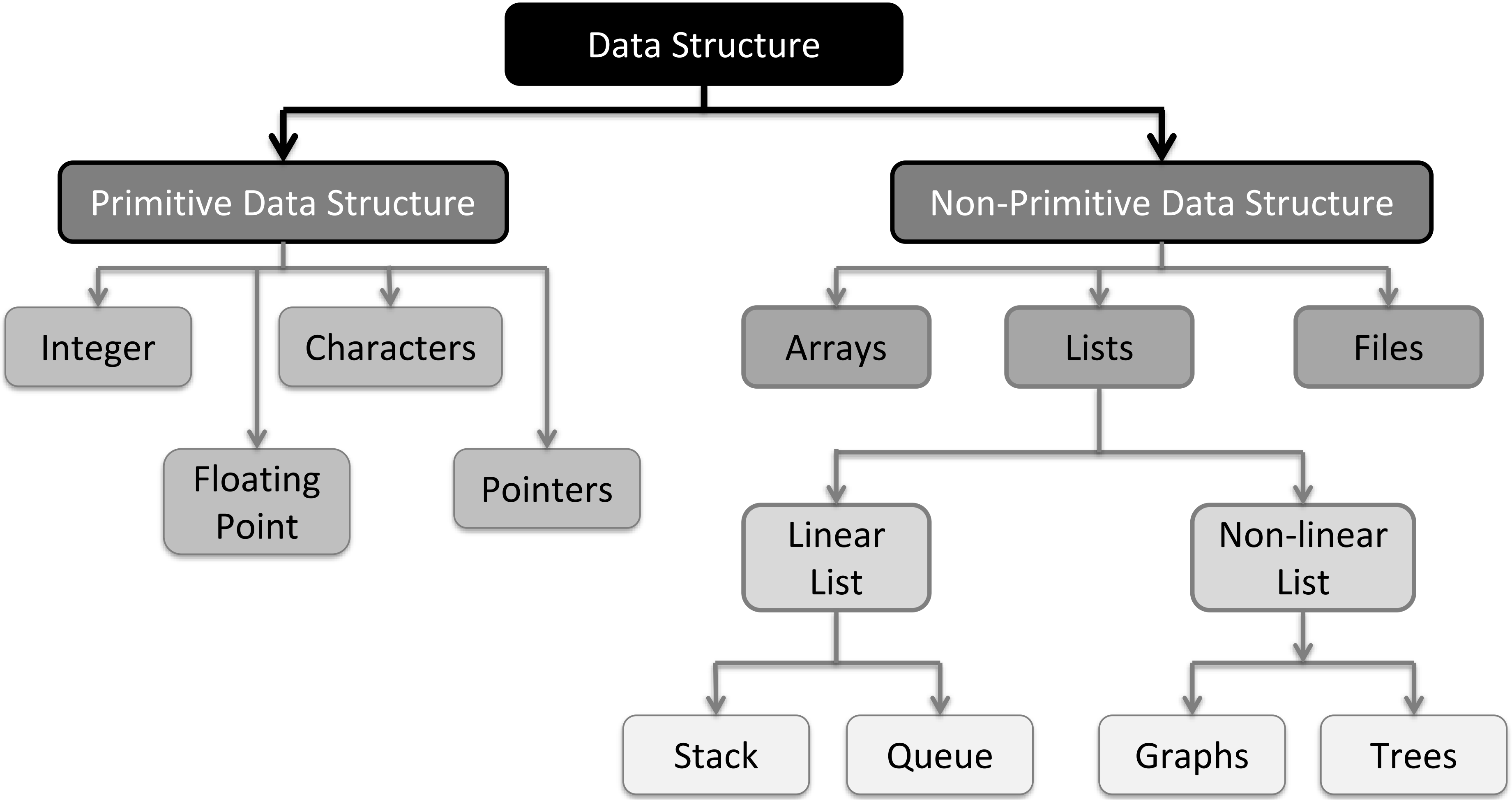
- The **representation** of a particular data **structure in the memory** of a computer is called ***Storage Structure.***

- The storage structure **representation in auxiliary memory** is called as ***File Structure.***



Algorithm **Data Structure** **Program**

CLASSIFICATION OF DATA STRUCTURE

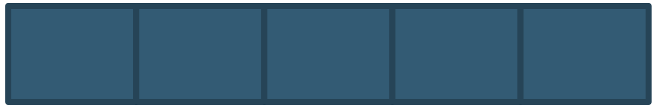


PRIMITIVE / NON-PRIMITIVE DATA STRUCTURES

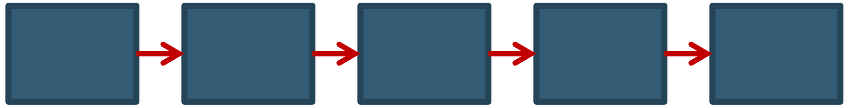
- **Primitive data structures**
  - Primitive data structures are basic structures and are directly operated upon by machine instructions.
  - ***Integers***, ***floats***, ***character*** and ***pointers*** are examples of primitive data structures.
- **Non primitive data structure**
  - These are derived from primitive data structures.
  - The non-primitive data structures emphasize on structuring of a group of homogeneous or heterogeneous data items.
  - Examples of Non-primitive data type are ***Array***, ***List***, and ***File***.

NON PRIMITIVE DATA STRUCTURE

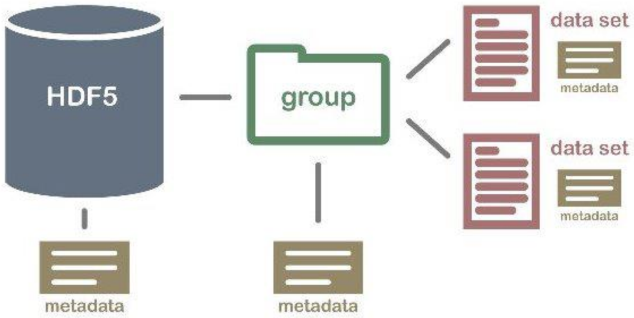
- **Array:** An array is a fixed-size sequenced collection of elements of the same data type.
- **List:** An ordered set containing variable number of elements is called as Lists.
- **File:** A file is a collection of logically related information. It can be viewed as a large list of records consisting of various fields.



Array



List



File

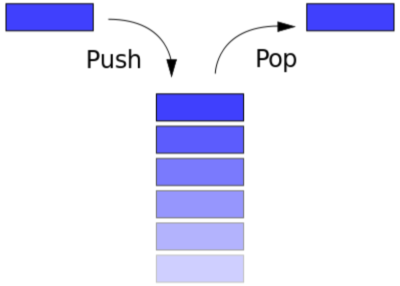
LINEAR / NON-LINEAR DATA STRUCTURE

Linear data structures

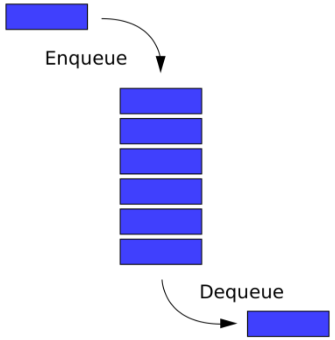
- A data structure is said to be Linear, if its elements are connected in linear fashion by means of logically or in sequence memory locations.
- Examples of Linear Data Structure are ***Stack*** and **Queue**.

Nonlinear data structures

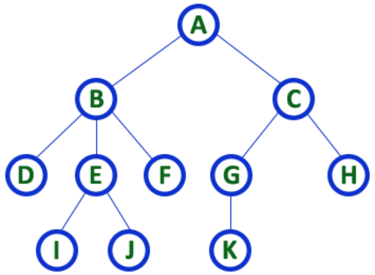
- Nonlinear data structures are those data structure in which data items are not arranged in a sequence.
- Examples of Non-linear Data Structure are ***Tree*** and ***Graph***.



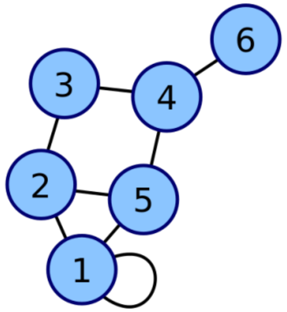
Stack



Queue



Tree



Graph

OPERATIONS OF DATA STRUCTURE

- **Create:** It results in reserving memory for program elements.
- **Destroy:** It destroys memory space allocated for specified data structure.
- **Selection:** It deals with accessing a particular data within a data structure.
- **Updation:** It updates or modifies the data in the data structure.
- **Searching:** It finds the presence of desired data item in the list of data items.
- **Sorting:** It is a process of arranging all data items in a data structure in a particular order.
- **Merging:** It is a process of combining the data items of two different sorted list into a single sorted list.
- **Splitting:** It is a process of partitioning single list to multiple list.
- **Traversal:** It is a process of visiting each and every node of a list in systematic manner.

TIME AND SPACE ANALYSIS OF ALGORITHMS

- Sometimes, there are more than one way to solve a problem.
- We need to learn how to compare the performance of different algorithms and choose the best one to solve a particular problem.
- While analyzing an algorithm, we mostly consider time complexity and space complexity.
- ***Time complexity*** of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the input.
- ***Space complexity*** of an algorithm quantifies the amount of space or memory taken by an algorithm to run as a function of the length of the input.
- Time & space complexity depends on lots of things like hardware, operating system, processors, etc.
- However, we don't consider any of these factors while analyzing the algorithm. We will only consider the execution time of an algorithm.

CALCULATE TIME COMPLEXITY OF ALGORITHM

- **Time Complexity** is most commonly **estimated** by **counting** the **number of elementary functions performed** by the algorithm.
- Since the algorithm's performance may vary with different types of input data,
  - hence for an algorithm we usually use the **worst-case Time complexity** of an algorithm because that is the maximum time taken for any input size.

CALCULATING TIME COMPLEXITY

- Calculate Time Complexity of Sum of elements of List (One dimensional Array)

```
SumOfList(A,n)
{
Line 1   total = 0
Line 2   for i = 0 to n-1
Line 3     total = total + A[i]
Line 4   return total
}
```

A is array, n is no of elements in array

Line	Cost	No of Times
1	1	1
2	2	n + 1
3	2	n
4	1	1

TSumOfList $= 1 + 2\,(n+1) + 2n + 1$

$= 4n + 4$ ← We can neglate constant 4

$= n$

Time complexity of given algorithm is ***n*** unit time