Introduction to Data Structure

Data Structure is a way of collecting and organizing data in a computer in such a way that we can perform operations on these data in an effective way. Almost every application uses various types of data structures in one or the other way. For example, we can store a list of items in the array data structure. It is designed and implemented in such a way that it reduces complexity and increases efficiency.

## Data Definition

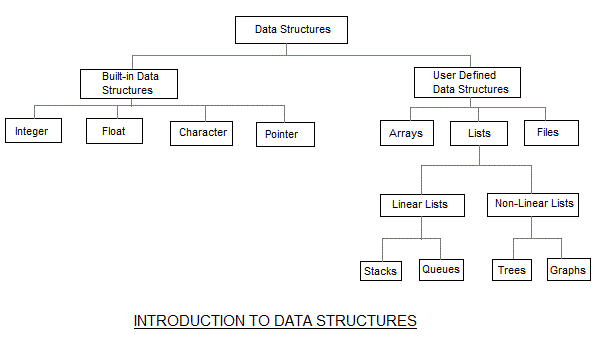
Data Definition defines a particular data with the following characteristics.

* Atomic − Definition should define a single concept.
* Traceable − Definition should be able to be mapped to some data element.
* Accurate − Definition should be unambiguous.
* Clear and Concise − Definition should be understandable.

## Data Type

The data type is a way to classify various types of data such as integer, string, etc. which determines the values that can be used with the corresponding type of data, the type of operations that can be performed on the corresponding type of data. There are two data types −

* Built-in Data Type
* Derived Data Type



### Built-in Data Type

Those data types for which a language has built-in support are known as Built-in Datatypes. For example, most of the languages provide the following built-in data types.

Integers

* Boolean (true, false)
* Floating (Decimal numbers)
* Character and Strings

### Derived Data Type

Those data types which are implementation-independent as they can be implemented in one or the other way are known as derived data types. These data types are normally built by the combination of primary or built-in data types and associated operations on them. For example −

* List
* Array
* Stack
* Queue

The data structures can also be classified on the basis of the following characteristics:

1. Linear: In Linear data structures, the data items are arranged in a linear sequence. Example: Array
2. Non-Linear: In Non-Linear data structures, the data items are not in sequence. Example: Tree, Graph
3. Homogeneous: Inhomogeneous data structures, all the elements are of the same type. Example: Array
4. Non-Homogeneous: In a Non-Homogeneous data structure, the elements may or may not be of the same type. Example: Structures
5. Static: Static data structures are those whose sizes and structures associated memory locations are fixed, at compile time. Example: Array
6. Dynamic: Dynamic structures are those which expand or shrinks depending upon the program’s need and its execution. Also, their associated memory locations changes. Example: Linked List created using pointers

# Why Learn Data Structure and Algorithms?

As applications are getting complex and data-rich, there are three common problems that applications face nowadays.

* **Data Search:** Consider an inventory of 1 million(106) items of a store. If the application is to search an item, it has to search an item in 1 million(106) items every time slowing down the search. As data grows, the search will become slower.
* **Processor speed:** Processor speed although being very high, falls limited if the data grows to billion records.
* **Multiple requests:** As thousands of users can search data simultaneously on a web server, even the fast server fails while searching the data.

To solve the above-mentioned problems, data structures come to the rescue. Data can be organized in a data structure in such a way that all items may not be required to be searched, and the required data can be searched almost instantly.

The following computer problems can be solved using Data Structures:

* Fibonacci number series
* Knapsack problem
* Tower of Hanoi
* All pair shortest path by Floyd-Warshall
* Shortest path by Dijkstra
* Project scheduling

## Major Operations on data structure

The data in the data structures are processed by certain operations. The particular data structure chosen largely depends on the frequency of the operation that needs to be performed on the data structure.

* Traversing: Traversing the data structure means visiting each element of the data structure in order to perform some specific operation like searching or sorting.
* Insertion: Insertion can be defined as the process of adding the elements to the data structure at any location.
* Deletion: The process of removing an element from the data structure is called Deletion. We can delete an element from the data structure at any random location.
* Searching: The process of finding the location of an element within the data structure is called Searching.
* Sorting: The process of arranging the data structure in a specific order is known as Sorting.
* Merging: When two lists List A and List B of size M and N respectively, of similar type of elements, joined to produce the third list, List C of size (M+N), then this process is called merging.

## Characteristics of a Data Structure

* Correctness: Data structure implementation should implement its interface correctly.
* Time Complexity: Running time or the execution time of operations of the data structure must be as small as possible.
* Space Complexity: Memory usage of a data structure operation should be as little as possible.

## Execution Time Cases

There are three cases that are usually used to compare various data structure's execution times in a relative manner.

* Worst Case − This is the scenario where a particular data structure operation takes maximum time it can take. If an operation's worst-case time is ƒ(n) then this operation will not take more than ƒ(n) time where ƒ(n) represents the function of n.
* Average Case − This is the scene depicting the average execution time of an operation of a data structure. If an operation takes ƒ(n) time in execution, then m operations will take mƒ(n) time.
* Best Case − This is the scene depicting the least possible execution time of an operation of a data structure. If an operation takes ƒ(n) time in execution, then the actual operation may take time as the random number which would be maximum as ƒ(n).

## Basic Terminology

* Data − Data are values or set of values.
* Data Item − Data item refers to a single unit of values.
* Data Object represents an object having data.
* Group Items − Data items that are divided into sub-items are called Group Items.
* Elementary Items − Data items that cannot be divided are called Elementary Items.
* Attribute and Entity − An entity is that which contains certain attributes or properties, which may be assigned values.
* Entity Set − Entities of similar attributes form an entity set.
* Field − Field is a single elementary unit of information representing an attribute of an entity.
* Record − Record is a collection of field values of a given entity.
* File − File is a collection of records of the entities in a given entity set.
* Pointer is used to points the address of the value stored anywhere in the computer memory. To obtain the value stored at the location is known as dereferencing the pointer. Pointer improves the performance for repetitive processes such as: Traversing String, Lookup Tables, Control Tables, Tree Structures.
* A structure is a composite data type that defines a grouped list of variables that are to be placed under one name in a block of memory. It allows different variables to be accessed by using a single pointer to the structure. It can hold variables of different data types. We can create objects containing different types of attributes. It is used to implement other data structures like linked lists, stacks, queues, trees, graphs etc.

The following terms are the foundation terms of a data structure.

* Interface: Each data structure has an interface. The interface represents the set of operations that a data structure supports. An interface only provides the list of supported operations, the type of parameters they can accept, and the return type of these operations.
* Implementation: Implementation provides the internal representation of a data structure. The implementation also provides the definition of the algorithms used in the operations of the data structure.

# ==============================================