Introduction to Data Structures and Algorithms

Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage. For example, we have some data which has, player's **name** "Virat" and **age** 26. Here "Virat" is of **String** data type and 26 is of **integer** data type.

We can organize this data as a record like **Player** record, which will have both player's name and age in it. Now we can collect and store player's records in a file or database as a data structure. **For example**: "Dhoni" 30, "Gambhir" 31, "Sehwag" 33

If you are aware of Object Oriented programming concepts, then a `class` also does the same thing, it collects different type of data under one single entity. The only difference being, data structures provides for techniques to access and manipulate data efficiently.

In simple language, Data Structures are structures programmed to store ordered data, so that various operations can be performed on it easily. It represents the knowledge of data to be organized in memory. It should be designed and implemented in such a way that it reduces the complexity and increases the efficiency.


Basic types of Data Structures

As we have discussed above, anything that can store data can be called as a data structure, hence Integer, Float, Boolean, Char etc, all are data structures. They are known as **Primitive Data Structures**.

Then we also have some complex Data Structures, which are used to store large and connected data. Some example of **Abstract Data Structure** are :

- Linked List
- Tree
- Graph
- Stack, Queue etc.

All these data structures allow us to perform different operations on data. We select these data structures based on which type of operation is required. We will look into these data structures in more details in our later lessons.
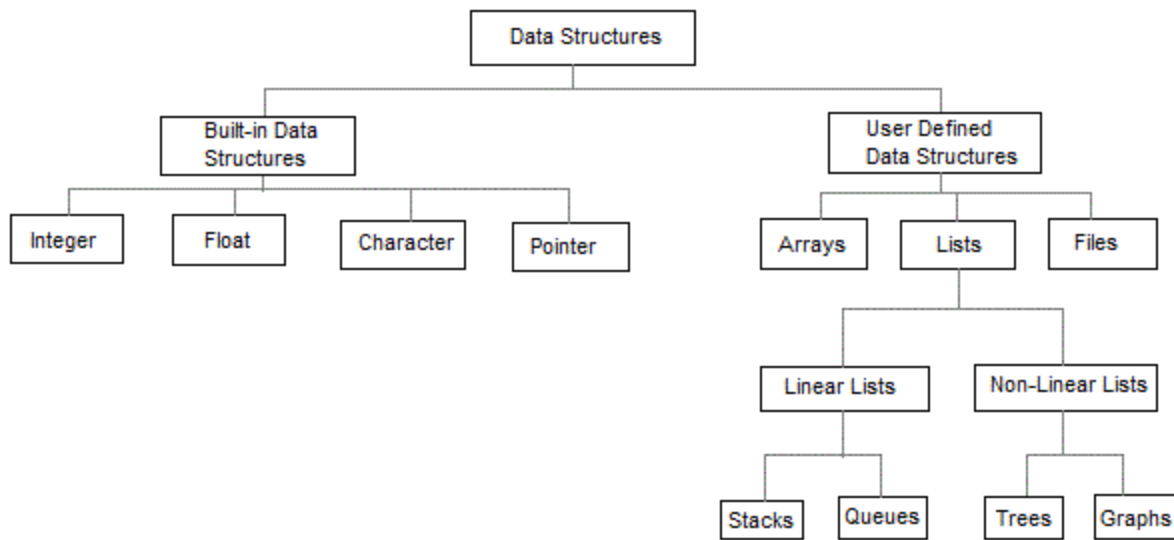
INTRODUCTION TO DATA STRUCTURES

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

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

What is an Algorithm ?

An algorithm is a finite set of instructions or logic, written in order, to accomplish a certain predefined task. Algorithm is not the complete code or program, it is just the core logic(solution) of a problem, which can be expressed either as an informal high level description as **pseudocode** or using a **flowchart**.

Every Algorithm must satisfy the following properties:

1. **Input**- There should be 0 or more inputs supplied externally to the algorithm.
2. **Output**- There should be atleast 1 output obtained.
3. **Definiteness**- Every step of the algorithm should be clear and well defined.
4. **Finiteness**- The algorithm should have finite number of steps.
5. **Correctness**- Every step of the algorithm must generate a correct output.

An algorithm is said to be efficient and fast, if it takes less time to execute and consumes less memory space. The performance of an algorithm is measured on the basis of following properties :

1. Time Complexity
2. Space Complexity

Space Complexity

Its the amount of memory space required by the algorithm, during the course of its execution. Space complexity must be taken seriously for multi-user systems and in situations where limited memory is available.

An algorithm generally requires space for following components :

- **Instruction Space:** Its the space required to store the executable version of the program. This space is fixed, but varies depending upon the number of lines of code in the program.
- **Data Space:** Its the space required to store all the constants and variables(including temporary variables) value.
- **Environment Space:** Its the space required to store the environment information needed to resume the suspended function.

To learn about Space Complexity in detail, jump to the Space Complexity (space-complexity-of-algorithms) tutorial.

Time Complexity

Time Complexity is a way to represent the amount of time required by the program to run till its completion. It's generally a good practice to try to keep the time required minimum, so that our algorithm completes it's execution in the minimum time possible. We will study about Time Complexity (time-complexity-of-algorithms) in details in later sections.

NOTE: Before going deep into data structure, you should have a good knowledge of programming either in C or in C++ or Java or Python etc.