# Unit 1: Introduction to Data Structure & Algorithm

*Presented By*
***Jendi Bade Shrestha,***
***ME Computer***
***jendibade@gmail.com***
***9851125364***

# Data Structure

- Data Structure is a way of collecting and organizing data in such a way that we can perform operations on these data in an effective way.
- Data structure affects the design of both the structural and functional aspects of a program
- Data structure = Organized data + Operations

# Data Structure

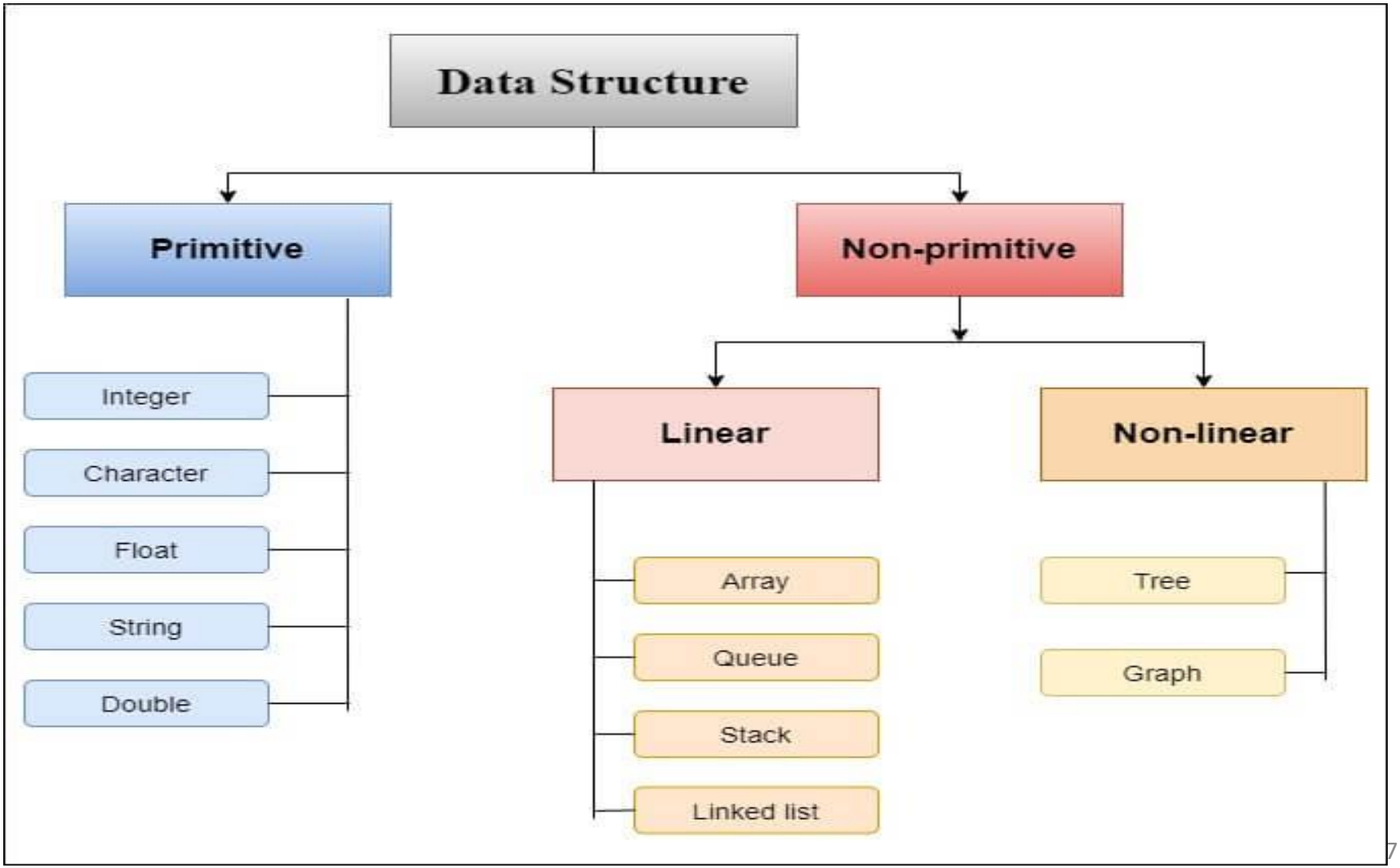
- In other words, data structure is a data organization, management, and storage format that enables efficient access and modification
- More precisely, a data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data

- Algorithm + Data Structure = Program
- Data structures are the building blocks of a program.
- Hence proper selection of data structure increases the productivity of programmers due to the proper designing and the use of efficient algorithms.

- 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.

# Types of Data Structures



# Primitive data structures

- These are the basic data structures and are directly operated upon by the machine instructions, which is in a primitive level.
- They are integers, floating point numbers, characters, string constants, pointers etc.

# Non-primitive data structures

- It is a more sophisticated data structure emphasizing on structuring of a group of homogeneous (same type) or heterogeneous (different type) data items.
- Array, list, files, linked list, trees and graphs fall in this category.

# Abstract data type

- Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of value and a set of operations.
- The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented.
- It is called “abstract” because it gives an implementation-independent view.
- The process of providing only the essentials and hiding the details is known as abstraction.

- Formally, an abstract data type is a data declaration packaged together with the operations that are meaningful on the data type.
- In other words, we can encapsulate the data and the operation on data and we hide them from user.

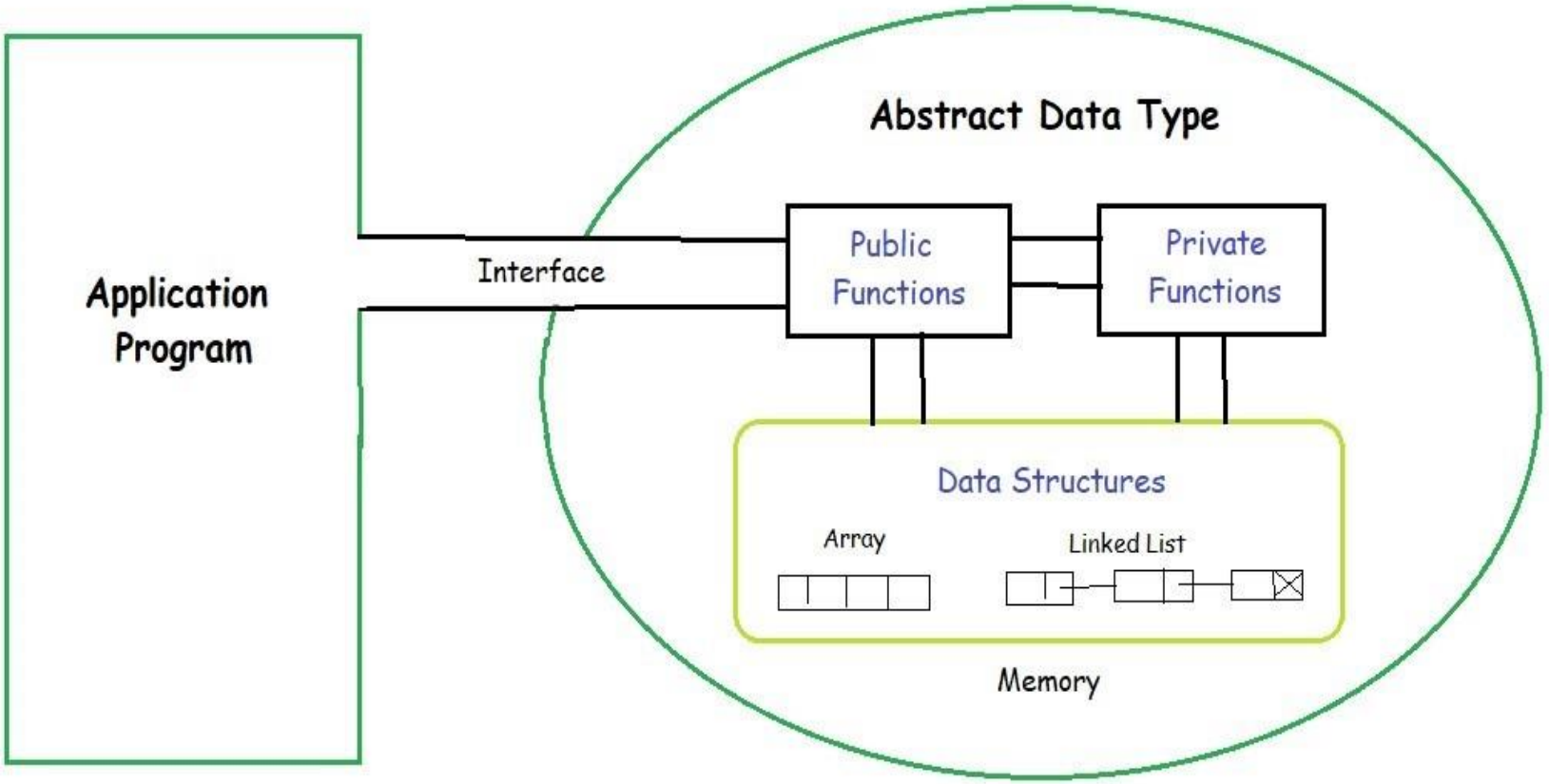



Fig: Abstract data type

# Cont…

- The user of data type does not need to know how that data type is implemented,
- for example, we have been using Primitive values like int, float, char data types only with the knowledge that these data type can operate and be performed
- on without any idea of how they are implemented.
- So a user only needs to know what a data type can do, but not how it will be implemented.

# Cont…

- Think of ADT as a black box which hides the inner structure and design of the data type.
- Now we'll define three ADTs namely List ADT, Stack ADT, Queue ADT.

# Data Structure Vs ADT

- Space and time complexity are not concerned matter in ADT but are concerned in DS.
- DS is implementation dependent whereas ADT is implementation independent.
- ADT is logical view of data but DS is physical quantity which can be broken down in to smaller units.

# Asymptotic Notations

- Asymptotic notations are used to represent the complexities( time, space and programming requirements) of algorithms. These notations are mathematical tools to represent the complexities. There are three notations that are commonly used.

**Big Oh Notation**

- Big-Oh (O) notation gives an upper bound for a function f(n) to within a constant factor.

**Little o Notations**

- There are some other notations present except the Big-Oh, Big-Omega and Big-Theta notations. The little o notation is one of them.
- Little o notation is used to describe an upper bound that cannot be tight. In other words, loose upper bound of f(n).

**Big Omega Notation**

- Big-Omega (Ω) notation gives a lower bound for a function f(n) to within a constant factor.

**Little ω Notations**

- Another asymptotic notation is little omega notation. it is denoted by (ω).
- Little omega (ω) notation is used to describe a loose lower bound of f(n).

**Big Theta Notation**

- Big-Theta(Θ) notation gives bound for a function f(n) to within a constant factor.

# Big ‘O’ notation (O)

- It allows to measure the properties of algorithm such as performance and/or memory requirement.
- It can be determined by ignoring the implementation dependent factors, by eliminating constant factors in the analysis of algorithm.

| Complexity | Performance |
|---|---|
| O(1) | Excellent |
| O(log(n)) | Good |
| O(n) | Fair |
| O(n log(n)) | Bad |
| O(n^2) | Horrible |
| O(2^n) | Horrible |
| O(n!) | Horrible |


Thank
You