Why learn data structures:

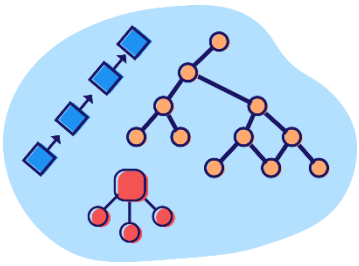
- Understanding data structures is essential for computer science and programming, as they provide the foundation for building efficient algorithms and software solutions.

What is Data Structure:

- Data Structure is a way to store and organize data so that it can be used efficiently.
- Data structures provide a way to manage large amounts of data, enabling effective data manipulation and retrieval.
    - There are many ways of organizing the data in the memory some of those are:
        - Array
        - Stack

Ways to organize data in memory:

- Linear (Array, Linked List, Stack, Queue) - The arrangement of data in a sequential manner is known as a linear data structure.
- Non-Linear (Tree, Graph, Hash Table) - When one element is connected to the 'n' number of elements known as a non-linear data structure.



- **Linear** data structures are not very **memory** friendly and are not utilizing **memory** efficiently. **Non-linear** data structures use **memory** very efficiently

Static vs Dynamic Memory:

- there are two primary ways to allocate memory: **static allocation** and **dynamic allocation**.
- Each method has its own characteristics, use cases, advantages, and disadvantages.

1. Static Memory Allocation:

- Static memory allocation is the **process of allocating memory at compile time**, before the program is executed. **The size of the memory required is determined during compilation**.

Characteristics of static memory allocation:

- **Fixed Size:** The size of the allocated memory must be known at compile time and cannot change at runtime.
- **Automatic Deallocation**: Memory is automatically deallocated when the variable goes out of scope (for local variables) or when the program terminates (for global variables).

- **Faster Access:** Since the memory is allocated in a fixed location, access to static memory is generally faster than dynamic memory.

2. Dynamic Memory Allocation

- **Dynamic memory allocation allows programs to request memory at runtime**, enabling more flexible memory management**. Memory is allocated on the heap,** and **the size can be determined during execution.**

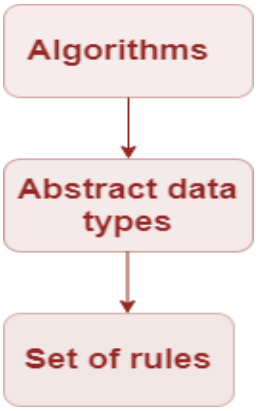
Characteristics of Dynamic Memory:

- **Variable Size**: The size of the allocated memory can be determined at runtime, allowing for more flexible data structures.
- **Manual Deallocation**: The programmer must manually release the memory when it is no longer needed to prevent memory leaks.
- **Overhead**: There is a slight overhead for managing dynamic memory compared to static memory, as the memory allocator must find and manage free blocks.

Difference between linear and non-linear memory organization:

Sr. No.	Key	Linear Data Structures	Non-linear Data Structures
1	Data Element Arrangement	In linear data structure, data elements are sequentially connected and each element is traversable through a single run.	In non-linear data structure, data elements are hierarchically connected and are present at various levels.
2	Levels	In linear data structure, all data elements are present at a single level.	In non-linear data structure, data elements are present at multiple levels.
3	Implementation complexity	Linear data structures are easier to implement.	Non-linear data structures are difficult to understand and implement as compared to linear data structures.
Examples		Array, List, Queue, Stack.	Graph, Map, Tree.

DS and ADT:



- The data structure is not any programming language like C, C++, java, etc. It is a set of algorithms that we can use in any programming language to structure the data in the memory.
- **To structure the data in memory, 'n' number of algorithms were proposed, and all these algorithms are known as Abstract data types**. These abstract data types are the set of rules.

Types of Data Structures:

There are two types of data structures:

- Primitive data structure
- Non-primitive data structure

1. Primitive Data Structures

- Primitive data structures are the basic building blocks for data manipulation in programming. They are predefined by programming languages and typically correspond directly to the underlying hardware.

Characteristics of Primitive Data Structures

- Simplicity: They represent single values and are straightforward to use.
- Direct Mapping: They have a direct representation in the machine's memory.
- Built-in Support: Most programming languages provide built-in support for these data types.
- Common primitive data structure: Int, char, float, double,

2. Non-Primitive Data Structures

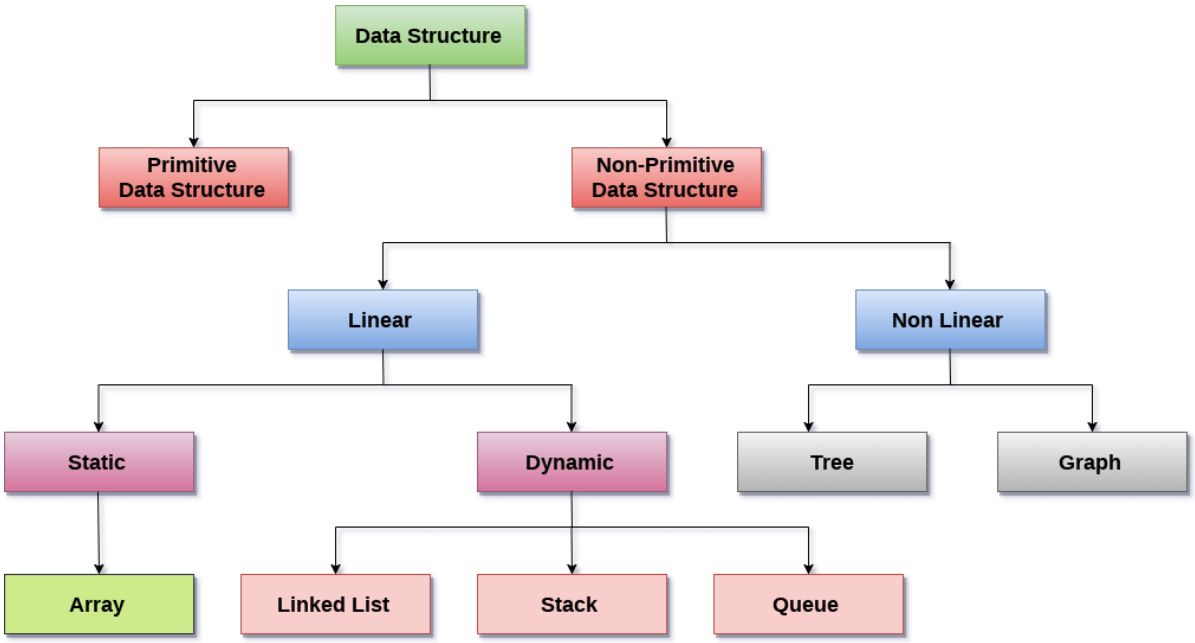
- Non-primitive data structures are more complex than primitive data structures. They can be constructed using primitive data types and can store multiple values or a collection of values.

Characteristics of Non-Primitive Data Structures

- Complexity: They can store multiple values, often of different types.
- User-Defined: They can be defined by the user, providing flexibility and the ability to model complex relationships.
- More Efficient: They allow for more efficient data organization and management compared to primitive types.
- Common non-primitive data structures: Array, Stack & Queue, Linked Lists, Structure & Union, Tree, Graph

Data structures can also be classified as:

- **Static data structure:** It is a type of data structure where the size is allocated at the compile time. Therefore, the maximum size is fixed.
- **Dynamic data structure:** It is a type of data structure where the size is allocated at the run time. Therefore, the maximum size is flexible.



Major Operations:

The major or the common operations that can be performed on the data structures are:

- **Searching:** We can search for any element in a data structure.
- **Sorting:** We can sort the elements of a data structure either in an ascending or descending order.
- **Insertion:** We can also insert the new element in a data structure.
- **Updation:** We can also update the element, i.e., we can replace the element with another element.
- **Deletion:** We can also perform the delete operation to remove the element from the data structure.

Which Data Structure?

- data structure is a way of organizing the data so that it can be used **efficiently**. Here, we have used the word **efficiently**, which **in terms of both the space and time**.
- **different data structures can be implemented in a particular ADT**, but the different **implementations are compared for time and space.**
- For example**, the Stack ADT can be implemented by both Arrays and linked list.** Suppose the array is providing **time efficiency** while the linked list is providing **space efficiency**, so the one which is the **best suited for the current user's requirements will be selected.**

Advantages of Data structures

The following are the advantages of a data structure:

- **Efficiency:** If the choice of a data structure for implementing a particular ADT is proper, it makes the program very efficient in terms of time and space.
- **Reusability:** The data structure provides reusability means that multiple client programs can use the data structure.
- **Abstraction:** The data structure specified by an ADT also provides the level of abstraction. The client cannot see the internal working of the data structure, so it does not have to worry about the implementation part. The client can only see the interface.

s