## What is Data Structure?

Data structure is a particular way of storing and organizing data in a computer so that it can be retrieved and used most productively. There are different kinds of data structures for different kinds of applications, and some are highly specialized to specific tasks.

**Data structure = Data+ Algorithm**

Criteria to choose correct Data Structure:

a) The size of data will be used.

b) The way that data will be used.

c) How often the data will be changed.

d) The time required to access any information in the data structure.

e) Required capacity.

f) The programming language/ method will be used.

## Primitive and non-primitive data Types

Data type specifies the type of data stored in a variable. The data type can be classified into two types, primitive data type and non-primitive data type.

**Primitive Data types**: are the basic data types that are available in most of the programming languages. The primitive data types are used to represent single values like:

**Integer**: This is used to represent a number without decimal point. Example: 12, 90

**Float and Double**: This is used to represent a number with decimal point. Example: 45.1, 67.3

**Character**: This is used to represent single character Example: ‘C’, ‘a’

**String**: This is used to represent group of characters. Example: "CIHAN University"

**Boolean**: This is used represent logical values either true or false.

**Non-Primitive Data types**: are derived from primary data types and used to store group of values like:

• Arrays

• Structure

• linked list

• Stacks

• Queue

**Operations that can be performed on data structures:**

1. Traversing: It is used to access each data item exactly once so that it can be processed.

2. Searching: It is used to find out the location of the data item if it exists in the given collection of data items.

3. Inserting: It is used to add a new data item in the given collection of data items.

4. Deleting: It is used to delete an existing data item from the given collection of data items.

5. Sorting: It is used to arrange the data items in some order i.e. in ascending or descending order in case of numerical data and in dictionary order in case of alphanumeric data.

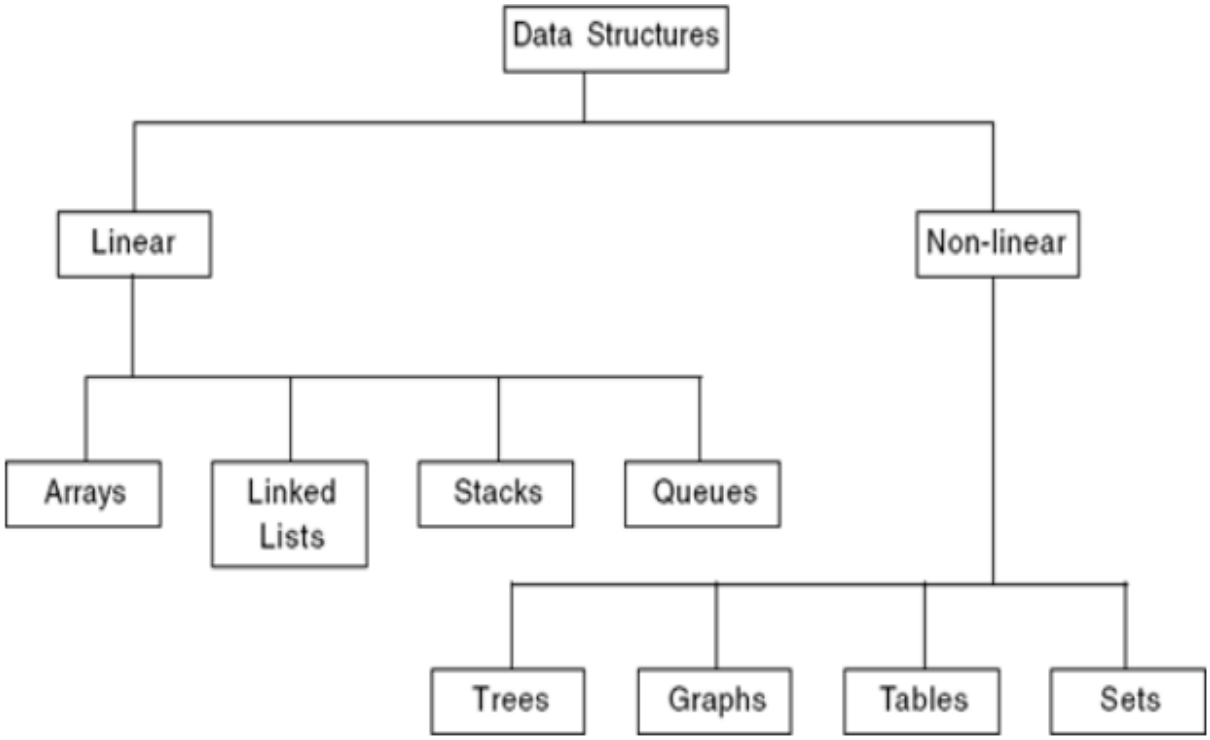
6. Merging: It is used to combine the data items of two sorted files into single file in the sorted form.

**Types of data structures**

There are two types of data structures, these are:

**Linear Data Structure**: Is the data structure that every element is linked with the next one sequentially. Since the data items are arranged in sequence. Samples of a linear data structure are the stack and the queue.

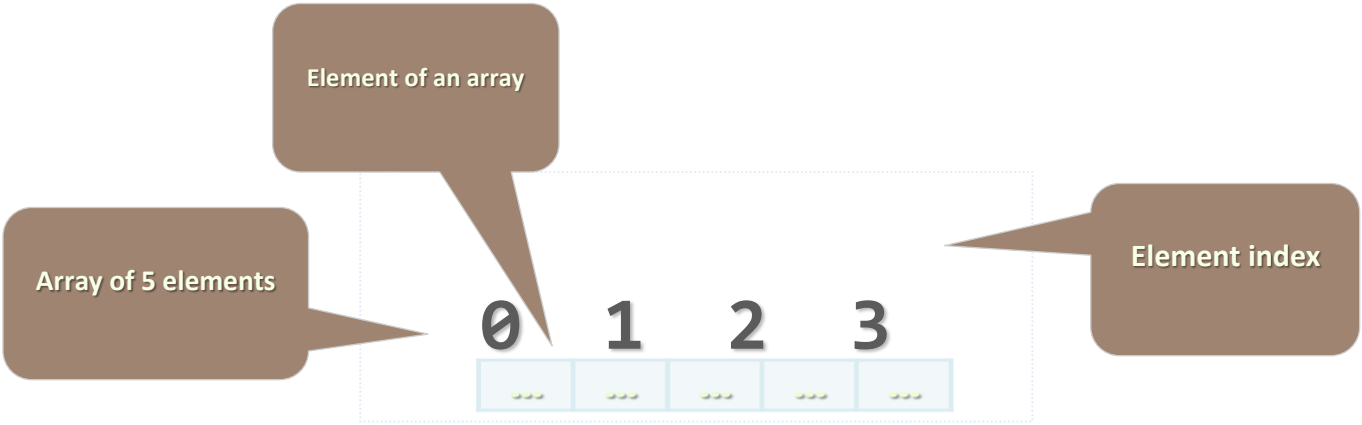
**Nonlinear data structure**: is the data structure where the element may attach to more than one element and data items are not in sequence. A sample of nonlinear data structure is a tree.




## Array

An array is a sequence of elements

- All elements are of the same type
- The order of the elements is fixed
- Has fixed size (Array.Length)



- Array elements are accessed using the square brackets operator [] (indexer)
- Array indexer takes element's index as parameter
- The first element has index 0
- The last element has index Length-1
- Array elements can be retrieved and changed by the [] operator

**Example:**

```
int[] array = new int[] {1, 2, 3, 4, 5};  or

int[] array = new int[5] ;
```

**Reading Arrays From the Console:**

```
int[] arr = new int[n];

for (int i=0; i<n; i++)

{

  arr[i] = int.Parse(Console.ReadLine());

}
```

**Printing Arrays on the Console**

```
for (int index = 0; index < array.Length; index++)

{

   // Print each element on a separate line

   Console.WriteLine("element[{0}] = {1}", index, array[index]);
```

**Operations on an Array:**

**1) Insert Element in an Array**

1. Initialize an array arr[] of size 6.
2. Define the item to insert (item = 100 for example) and the position (k = 3 for example).

3. Loop from the last index of the array to the first:
    - Shift each element one position to **the right**.
4. Insert the new item at index k.
5. Print the elements of the updated array.

**2) Search Element in an Array**

1) initialize the array:

- Declare an array array[] containing a set of elements.
- Define the item that you want to search for in the array.
- Set a boolean variable found = false to track if the item is found.

2) Print all elements of the array:

- Loop through the array from index i = 0 to i = array.Length - 1 and print each element with its index.

**3) Search for the item:**

- Loop through the array from index i = 0 to i = array.Length - 1:
    - Compare array[i] with item.
    - If array[i] == item:
        - Set found = true.
        - Print that the element has been found at position i (index).
        - Break the loop (exit the loop since the item is found).

4) Check if the item was not found:

- After the loop, if found is still false, print a message indicating that the element was not found in the array.

**4) Delete Element from an Array**

1) **Initialize the Array:**
    a. Define the array la[] with initial values.
    b. Set k to the (**position**) of the item you want to delete.

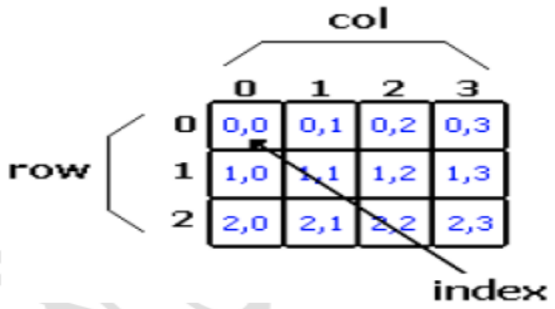
c. Set n to the total number of elements in the array.

2) **Print the Original Array:**
   a. Loop through the array and print each element before deletion.
3) **Shift Elements to the Left:**
   a. Set j = k (**the position of the item to delete**).
   b. Start a loop that runs from j to n-1:
      i. Assign la[j - 1] = la[j] to shift elements to the left, overwriting the item at position k.
   c. Decrease the size n of the array by 1 (since the array now has one fewer element).
4) **Print the Array After Deletion:**
   a. Loop through the updated array and print each element after the deletion.

## Array of Two Dimensions

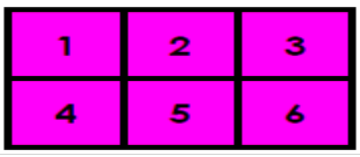
Arrays can have higher dimension. There can be arrays of two dimension which is array of arrays. It is accessed with two indexes. Also, there can be arrays of dimension higher than two. Examples:

int [,] num =new int [3,4];




**Initializing 2D-Array Elements**

int [,] a=new int [2,3] = { {1, 2, 3} , {4, 5, 6} };




Example:

Write C# program, to read 4*4 2D-array, then find the summation of the array elements. namespace ConsoleApplication3

```
{ class Program
{ static void Main(string[] args)
 {int [,] a=new int[4,4];
int i , j, sum = 0;
for ( i = 0 ; i < 4; i++ )
for ( j = 0 ; j < 4; j++ )
a[i,j]=int.Parse(Console.ReadLine());
 for ( i = 0 ; i < 4; i++ )
 for ( j = 0 ; j < 4; j++ )
sum += a [i,j];
Console.WriteLine("summation is: ",sum );
for ( i = 0 ; i < 4; i++ )
{ for ( j = 0 ; j < 4; j++ )
Console.WriteLine( a[i,j]); } } }
```