SUBJECT :- Data Structure. CLASS :- 4th Sem.

Introduction To Data Structures

A data structure is a special way of organizing and storing data in a computer so that it can be used efficiently.

Types of Data Structures

There are two types of Data Structures:

1. Linear Data Structure 2. Non-Linear Data Structure

1. Linear Data structures :→ Elements of linear data structure are accessed in a sequential manner, however the elements can be stored in these data structures in any order.
e.g: Linked list, stack, Queue and Array.

2. Non-Linear data structures :— Elements of non-linear data structures are stored and accessed in non-linear order.

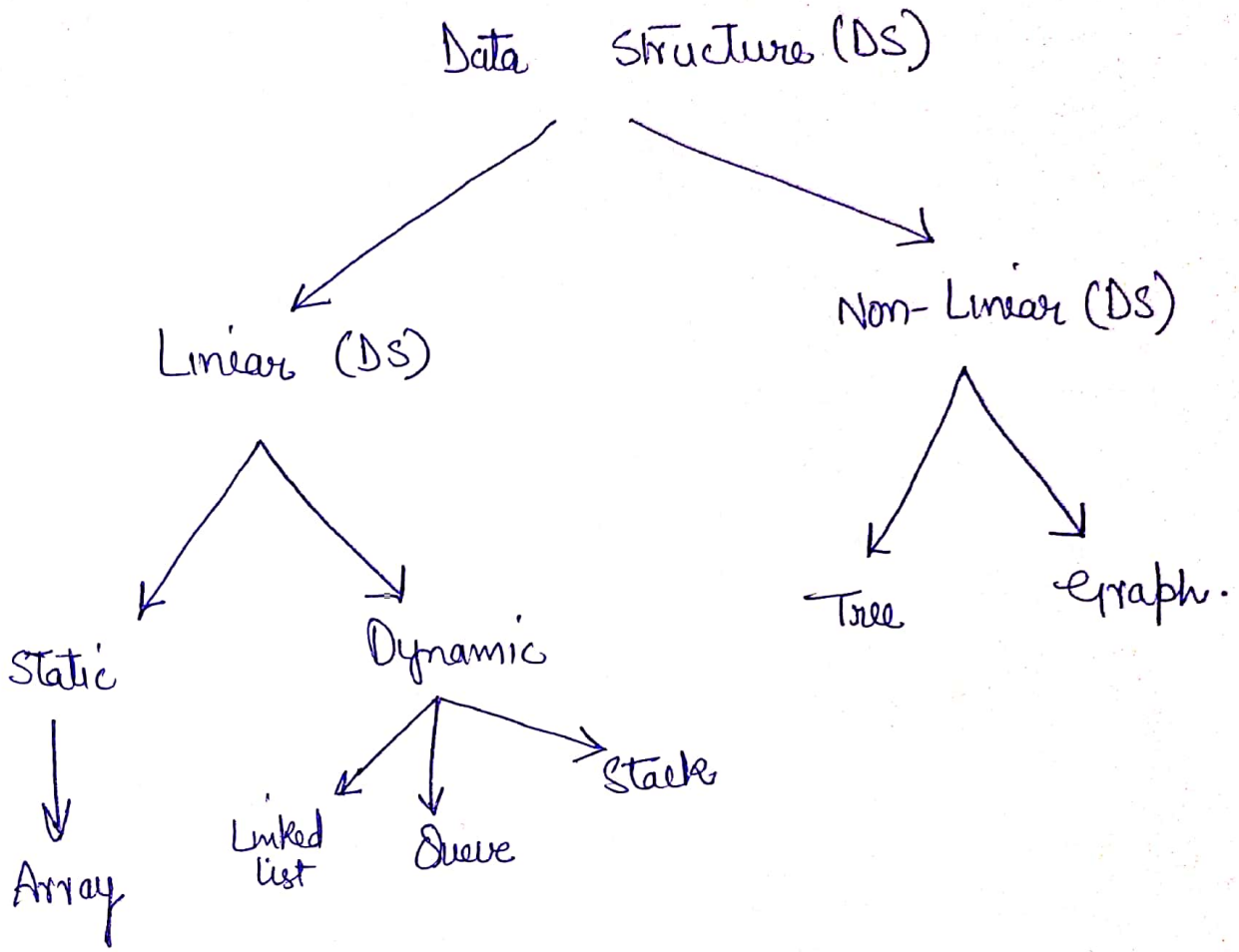
e.g: Tree & Graph.

Advantages of Data Structures :-

1. Data Organization :- We need a proper way of organizing the data so that it can be accessed efficiently when we need that particular data. Data structure provides different ways of data organization so we have options to store the data in different data structures based on the requirement.

2. Efficiency :- The main reason we organize the data is to improve the efficiency. We can store the data in arrays then why do we need linked lists and other data structures ...? because when we need to perform several operations such as add, delete update and search on arrays, it takes more time in arrays than some of the other data structures. So, the fact why we are interested in other data structures is because of the efficiency.

Q:- CLASSIFICATION OF DATA STRUCTURES



Array :- (Introduction)

An Array is a collection of homogenous (same type) data items stored in contiguous memory locations. For example; if an array is of type "int", it can only store integer elements and cannot allow the elements of other types such as; double, float, char etc.

Array is particularly useful when we are dealing with lot of variables of same type. e.g; lets says I need to store the marks in a particular subject of 100 students. To solve this particular problem, either I have to create the 100 variables of int type or create an array of int type with the size 100.

Obviously, the second option is best, because keeping track of the 100 different variables is a tedious task. On the other hand, dealing with array is simple and easy, all 100 values can be stored in the same array at different indexes (0-99).

Advantages and Disadvantages of Arrays

Advantages :-

1. Reading an array element is simple and efficient. This is because any element can be instantly read using indexes without traversing the whole array.
2. Array is a foundation of other data structures. e.g: other data structures such as, Linked list, stack, Queue etc. are implemented using arrays.
3. All the elements of an array can be accessed using a single name (array name) along with the index, which is readable, user-friendly and efficient rather than storing those elements in different-2 variables.

Disadvantages :→

(1) while using array, we must need to make the decision of the size of the array in the beginning, so if we are not aware how many elements we are going to store in array, it would make the task difficult (2) The size of the array is fixed, so if at later point, we need to store more elements in it, then it can't be done. (5)

Array in data Structure :-

An array data structure or simply an array, is a data structure consisting of a collection of elements, each identified by at least one array index or key. An array is stored such that the position of each element can be computed from its index by a mathematical formula. The simplest type of data structure is a linear array, also called one-dimensional array.

The memory address of the first element of an array is called first-address, foundation address, or base address. Arrays are among the oldest and most important data structures, and are used by almost every program. They are also used to implement many other data structures, such as lists and strings. They effectively exploit the addressing logic of computers. In most modern computers, the memory is a one-dimensional array of words, whose indices are their addresses.

One-dimensional or Single dimension array :-

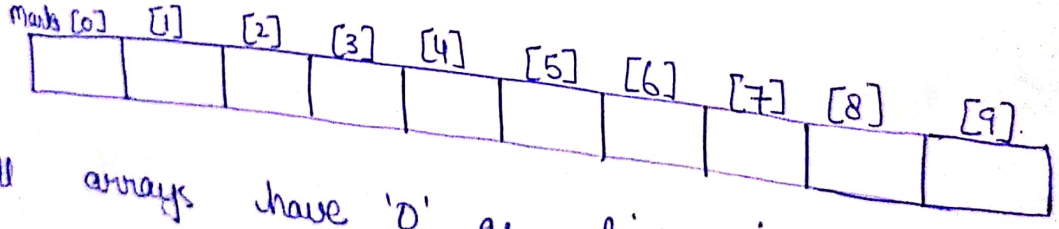
One-dimensional array is a type of linear array. Accessing its elements involves a single subscript which can either represent a row or column index.

Declaration

int ArrayName [10];

(One-dimensional array of Ten integers)

e.g: int marks [10]



① All arrays have '0' as first index.

② If the size of an array is n, to access the last element, (n-1) index is used.

③ It is possible to initialize an array during declaration.

C Program to find the average of n (n<10) numbers using arrays :-

```
#include <stdio.h>
int main()
{
int marks[10], i, n, sum=0, average;
Printf ("Enter n:");
Scanf ("%d", &n);
for (i=0 ; i<n ; i++)
{
Printf ("Enter number %d :", i+1);
Scanf ("%d", & marks[i]);
Sum += marks[i];
}
average = sum/n;
Printf ("Average = %d", average);
return 0;
}
```

Two-dimensional array :- (2D array)

A two dimensional array is an array of arrays. 2-dimensional arrays are the most commonly used. They are used to store data in a tabular manner.

To declare a 2D array, we must specify the following :

Row-size :- Defines the number of rows.

Column-size :- Defines the number of columns.

Type of array :- Defines the type of elements to be stored in the array. i.e, either a number, character or other such datatypes.

e.g; type arr [row_size] [column_size]

or. int arr [3] [5];

or, int arr [3] [5] = {{5, 12, 17, 9, 3}, {13, 4, 8, 14, 1}, {9, 6, 3, 7, 21}};

Columns.

Rows					
	5	12	17	9	3
	13	4	8	14	1
	9	6	3	7	21

(2D array of size 3 x 5)

Three-dimensional Array :- (3D Array)

A three-dimensional (3D) array is an array of arrays of arrays. An array can have two, three, or even ten or more dimensions.

A Three-dimensional array is declared using the following syntax :

type array_name [d1][d2][d3]

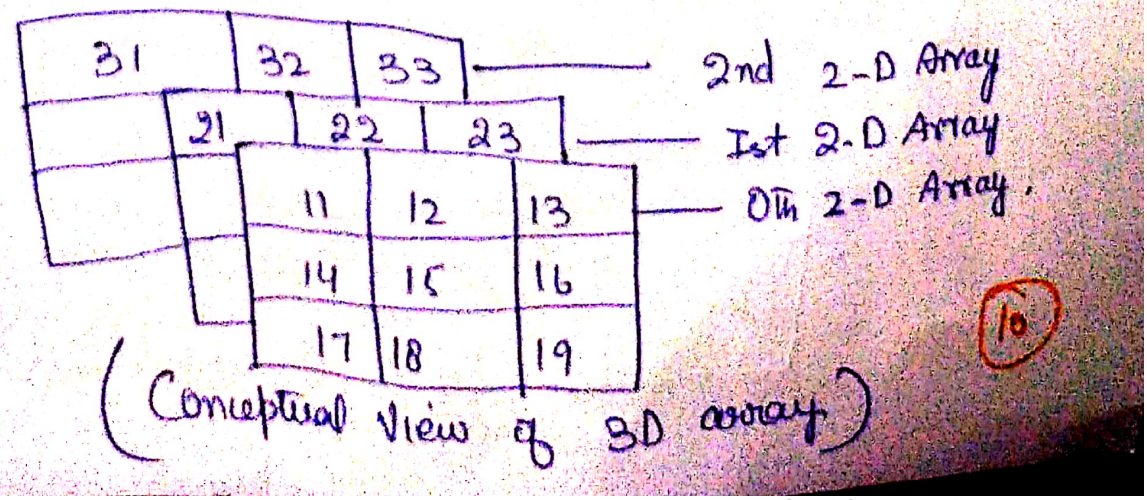
where d is a dimension.

eg; int table [5] [5] [20];

In this example;

- int designates the array type integer.
- table is the name of our 3D array.
- Our array can hold 500 inter-type elements. This number is reached by multiplying the value of each dimension as;

  5 x 5 x 20 = 500.



(Conceptual View of 3D array)