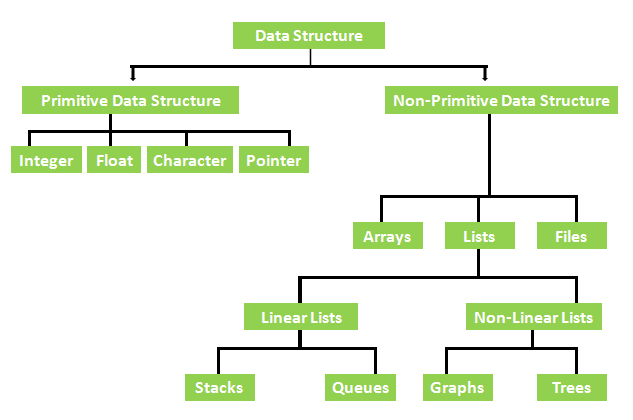
=: Introduction :=

**Data Structure:** A data structure is a way of storing data in a computer so that it can be used efficiently and it will allow the most efficient algorithm to be used. Data structure introduction refers to an arrangement of data in computer's memory in such a way that it could make the data quickly available to the processor for required calculations

The choice of the data structure begins from the choice of an abstract data type (ADT). ADT (Abstract Data Type) which is independent of implementation. The way in which the data is organized affects the performance of a program for different tasks. Computer programmers decide which data structures to use based on the nature of the data and the processes that need to be performed on that data. Some of the more commonly used data structures include lists, arrays, stacks, queues, heaps, trees, and graphs.

**Classification of Data Structure:**



**Linear Data Structure:** Linear data structures can be constructed as a continuous arrangement of data elements in the memory. It can be constructed by using array data type.

Operations applied on linear data structure: The following list of operations applied on linear data structures

1. Add an element

2. Delete an element

3. Traverse

4. Sort the list of elements

5. Search for a data element

For example Stack, Queue, Tables, List, and Linked Lists.

**Non-linear Data Structure**: Non-linear data structure can be constructed as a collection of randomly distributed set of data item joined together by using a special pointer (tag). In non-linear Data structure the relationship of adjacency is not maintained between the data items.

Operations applied on non-linear data structures: The following list of operations applied on non-linear data structures. 1. Add elements 2. Delete elements 3. Display the elements 4. Sort the list of elements 5. Search for a data element

For example Tree, Decision tree, Graph and Forest.

**Time Complexity:** The time complexity of an algorithm or a program is a function of the running time of the algorithm or a program. In other words, it is the amount of computer time it needs to run to completion.

**Space Complexity**: The space complexity of an algorithm or program is a function of the space needed by the algorithm or program to run to completion.

**Space complexity**

The Space complexity of a program is defined as the amount of memory it needs to run to completion.

As said above the space complexity is one of the factor which account s for the performance of the program. The space complexity can be measured using experimental method, which is done by running the program and then measuring the actual space occupied by the program during execution. But this is done very rarely. We estimate the space complexity of the program before running the program.

Space complexity is the sum of the following components:

(i) Instruction space : The program which is written by the user is the source program. When this program is compiled, a compiled version of the program is generated.

For executing the program an executable version of the program is generated. The space occupied by these three when the program is under execution, will account for the instruction space.

(ii) Data space : The space needed by the constants , simple variables , arrays , structures and other data structures will account for the data space.

The Data space depends on the following factors:  Structure size – It is the sum of the size of component variables of the structure.  Array size – Total size of the array is the product of the size of the data type and the number of array locations.

(iii ) Environment stack space : The Environment stack space is used for saving information needed to resume execution of partially completed functions. That is whenever the control of the program is transferred from one function to another during a function call, then the values of the local variable of that function and return address are stored in the environment stack. This information is retrieved when the control comes back to the same function.

The environment stack space depends on the following factors:  Return address

 Values of all local variables and formal parameters.

The Total space occupied by the program during the execution of the program is the sum of the fixed space and the variable space.

(i) Fixed space - The space occupied by the instruction space, simple variable s and constants. (ii) Variable space – The dynamically allocated space to the various data structures and the environment stack space varies according to the input from the user.

Space complexity S(P) = c + S p, Where c fixed space or constant space and Sp is variable space.

**TIME COMPLEXITY**

Time complexity: Time complexity of the program is defined as the amount of computer time it needs to run to completion.

The time complexity can be measured, by measuring the time taken by the program when it is executed. This is an experimental method. But this is done very rarely. We always try to estimate the time consumed by the program even before it is run for the first time.

The time complexity of the program depends on the following factors:  C o m pile r u s e d – some compilers produce optimized code which consume s less time to get executed.  Compiler options – The optimization options can be set in the options of the compiler .

 T a r g e t c o m p u t e r – The speed of the computer or the number of instructions executed per second differs from one computer to another.

The total time taken for the execution of the program is the sum of the compilation time and the execution time. (i) Compile time – The time taken for the compilation of the program to produce the intermediate object code or the compiler version of the program. The compilation time is taken only once as it is enough if the program is compiled once. If optimized code is to be generated, then the compilation time will be higher . (ii) Run time or Execution time - The time taken for the execution of the program. The optimized code will take less time to get executed. Time complexity T(P) = c + T p , where c is compile time and Tp is Run time or execution time.