Introduction to Data Structures and Algorithms

What is a Data Structure?

A data structure is a particular way of organizing real-life data in a computer so that it can be used effectively. It's an **intentional arrangement** of a collection of data. The **intentional arrangement** means the arrangement done on purpose to impose, some kind of systematic organization on the data.

To understand data structures, we must first understand data. In every programming language, we have some data types. In some programming languages, we have to explicitly express the data type by using keywords like **`int`, `float`, `bool`, `char`** etc. We might not see those in languages like JavaScript but we still have those. Just that they are hidden from the programmer's eyes. **The interpreter handles them for us, and this behavior is counted in feature of the language.**

Why do we study Data Structure?

Because it's useful, it makes our lives easier, and it's easy to manage when you keep related information together.

We must all be familiar with arrays, if not then no problem for now just understand that An array is a special variable, which can hold more than one value.
Arrays are great for some problems, but for many complex problems, they are simply not sophisticated enough. Most experienced programmers will admit that for many programming problems, once they come up with the proper data structure, the algorithms needed to solve the problem are easier to design and implement.

An example of a data structure that leads to efficient algorithms is the binary search tree (BST). A binary search tree is designed so that it is easy to find the minimum and maximum values of a set of data, yielding an algorithm that is more efficient than the best search algorithms available.

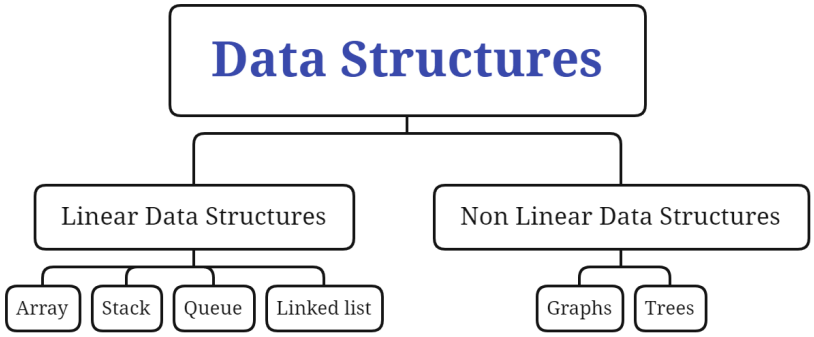
Programmers unfamiliar with BSTs will instead probably use a simpler data structure that ends up being less efficient.

In every data structure there are some fundamental behaviors such as access, insert, delete, find, & sort. These are the operations that you will most likely be going to perform in your programs. **However Not all of Data Structures** have the five fundamental behaviors.

For example, many data structures don't support any kind of searching behavior. It's just a big collection, a big container of stuff, and If you need to find something, you just go through all of it yourself. And many don't provide any kind of sorting behavior. Others are naturally sorted.

Data structures are divided into following categories:-

1. **Linear Data Structures**
    a. Array
    b. Stack
    c. Queue
    d. Linked List
2. **Non Linear Data Structures**
    a. Graphs
    b. Trees

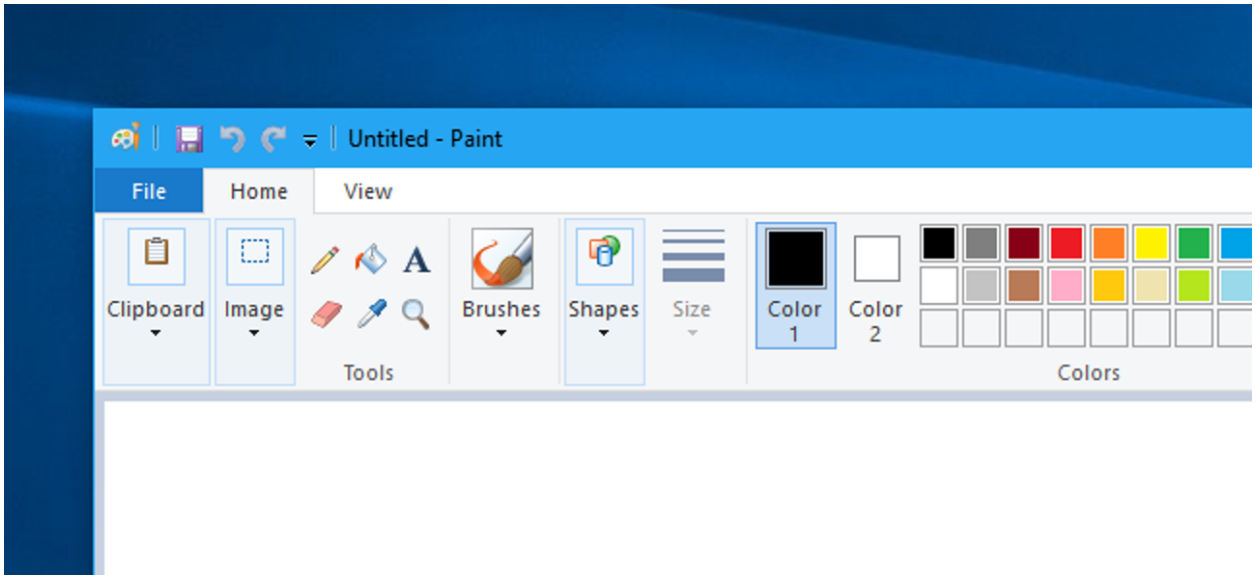


Data Structures In our Life

We use data structures in our daily life activities like while going for a shopping we will use an **array to store the items that we need to buy** or **while listening to music like for example**

spotify which uses array. Similarly a recipe, a telephone directory, a dictionary all have a structure and have a specific format.

MS-Paint drawings and shapes are connected via a linked list on canvas.



LinkedIn, Facebook, Instagram, and all social media networking sites in this every user is Node and we use graph data structure.

LinkedIn shows the number of followers and connections 1st, 2nd and 3rd using the data structure Graphs.

What is an Algorithm?

In computer science, a set of steps that need to be followed to solve a particular problem. These steps are collectively known as an algorithm.

Algorithms are nothing but steps that you will follow to find the answer of the solution. Consider an algorithm for making coffee !

1. Put the coffee powder in a cup.
2. Fill the kettle with water.
3. Boil the water in the kettle.
4. Pour some of the boiled water into the cup.
5. Add milk to the cup.
6. Add sugar according to the person's taste, to the cup.
7. Stir the coffee.
8. Drink the coffee.

Now, since you got what algorithm is,

Try solving this problem where you have two integers **"num1"** and **"num2"** and you want to find the sum of those two numbers.
How will you approach this problem, write your algorithm?

Solution:-

1. Input two integers as “**num1”** and **“num2”**
2. Create a variable "**total_sum**" to store the sum of two integers num1 and num2
3. Store the sum of those two variables in the "**sum**" variable
4. Output the **"sum"** variable

Code:-

```
var num1 = 5;
var num2 = 45;
var sum = num1 + num2; // to store the sum of two numbers
```

```
console.log("Sum:",sum);
```

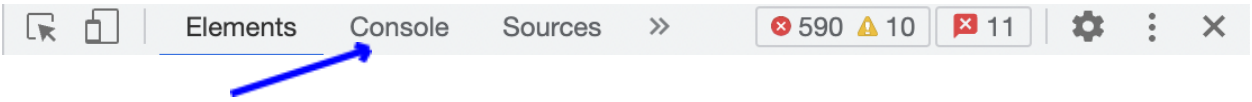
Output:-

```
> var num1 = 5;
  var num2 = 45;
  var sum = num1 + num2; // to store the sum of two numbers
  console.log("Sum:",sum);
  Sum: 50                                                    VM1356:4
```

To run the program you can use your chrome, any browser console window:-

To open your console window, follow these steps :-

Right click anywhere on any webpage -> Select Inspect -> choose Console -> Write your program and then press enter.



Why do we study Algorithms?

Studying algorithms is important because there is always more than one algorithm that can be used to solve a problem, and knowing which ones are the most efficient is important for the productive programmer. For example, there are at least six or seven ways to sort a list of data, but knowing that the Quicksort algorithm is more efficient than the selection sort algorithm will lead to a much more efficient sorting process. Or that it's fairly easy to implement a sequential or linear search algorithm for a list of data, but knowing that the binary sort algorithm can sometimes be twice as efficient as the sequential search will lead to a better program.

The more efficient & suitable the algorithm, the more you will have an optimized data structure. These algorithms might be built-in or implemented by developers to manage and run these data structures.

The comprehensive study of data structures and algorithms teaches you not only which data structures and which algorithms are the most efficient, but you also learn how to decide which data structures and which algorithms are the most appropriate for the problem at hand.

Examples of types of Algorithms:-

1. Simple recursive Algorithms
2. Backtracking Algorithms
3. Divide and conquer Algorithms
4. Dynamic programming Algorithms
5. Greedy Algorithms
6. Branch and bound Algorithms
7. Brute force Algorithms
8. Randomized Algorithms