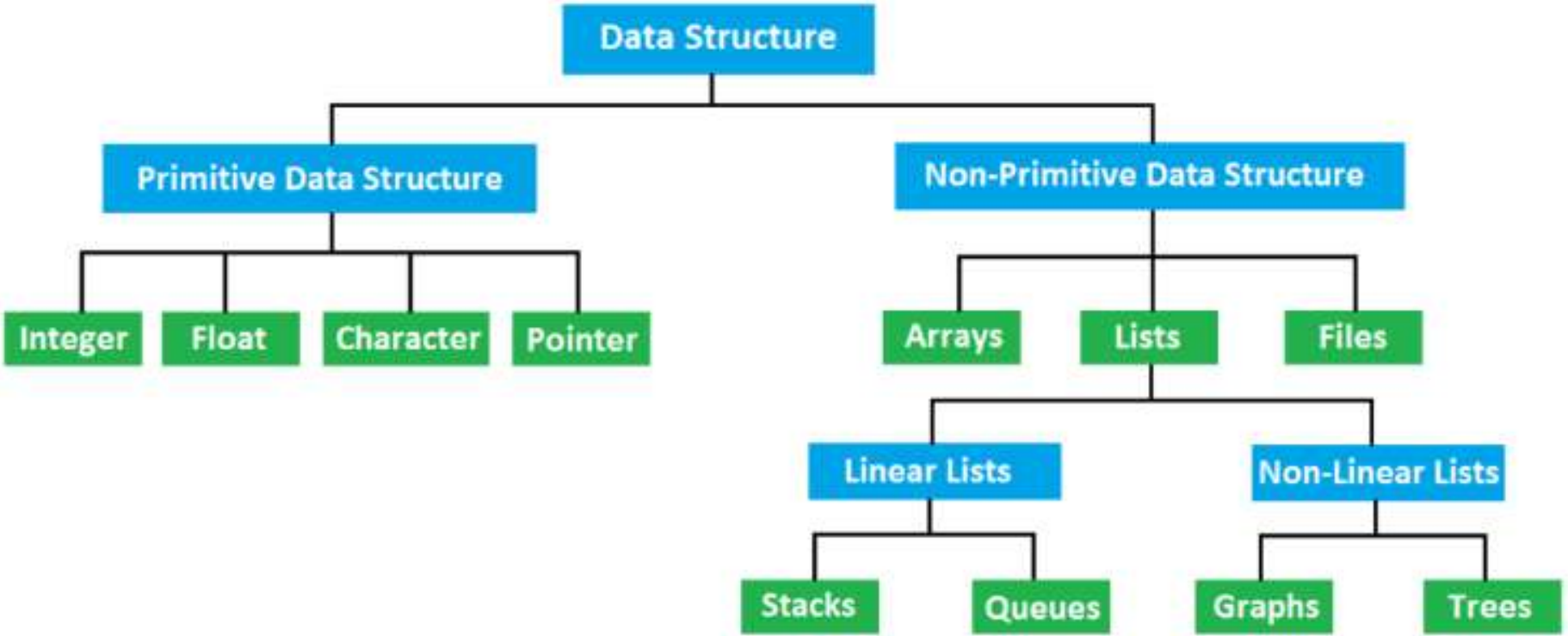
Need for Data Structures

- Data Structures organize data = efficient programs.
- The choice of data structure??
- A data structure requires certain amount of:

  "Time , Space, Programming efforts."

- Data Structures deals with the study of how data is organized in the memory, how efficiently the data can be retrieved and manipulated, and possibly the different data items are logically related.

Classification of Data Structures



Classification of Data Structures

- Dynamic Data Structures: grow and shrink during execution
- Linked Lists: insertions and removal made anywhere
- Stacks: insertions and removal made only at top of the stack
- Queues: insertion made at the back and removal made from the front
- Binary Trees: high speed searching and sorting of data
- Graphs: non-linear data structure consisting nodes and edges

TCS Exam Example

Problem Statement: Checking if a given year is leap year or not

Explanation:

To check whether a year is leap or not

Step 1:

- We first divide the year by 4.
- If it is not divisible by 4 then it is not a leap
- If it is divisible by 4 leaving remainder 0

Step 2:

- We divide the year by 100
- **If it is not divisible by 100 then it is a leap year.**
- If it is divisible by 100 leaving remainder 0

Step 3:

- We divide the year by 400
- If it is not divisible by 400 then it is a leap year.
- If it is divisible by 400 leaving remainder 0

Then it is a leap year

leap year

a year, occurring once every four years, which has 366 days including 29 February as an intercalary day.

Asymptotic Notations

- The efficiency of an algorithm is dependent on amount of time, storage and other resources. The efficiency is measured with the help of asymptotic notations.
- Asymptotic notations are mathematical notations used to describe the running time of an algorithm.
- Example: Bubble Sort:

1. If input array is already sorted: best case
2. If array is in reverse condition: worst case
3. Array is neither sorted nor in reverse condition: average case

Asymptotic Notations

There are mainly three asymptotic notations:

- **Big-O Notation (O-notation)** ----------→ represents upper bound of running time of algo. (worst case complexity)
- **Omega Notation (Ω-notation)**----------→ represents lower bound of the running time of algo. (best case complexity)
- **Theta Notation (Θ-notation)**----------→represents upper & lower bound of the running time of algo. (average case

Divide and Conquer Algo.

A **divide and conquer algorithm** is a strategy of solving a large problem by

1. breaking the problem into smaller sub-problems
2. solving the sub-problems, and
3. combining them to get the desired output.

How Divide and Conquer Algorithms Work?

Here are the steps involved:

1.Divide: Divide the given problem into sub-problems using recursion.

2.Conquer: Solve the smaller sub-problems recursively. If the subproblem is small enough, then solve it directly.

3.Combine: Combine the solutions of the sub-problems that are part of the recursive process to solve the actual problem.

Divide and Conquer Algo.

Advantages:

- The complexity for the multiplication of two matrices using the naive method is $O(n^3)$, whereas using the divide and conquer approach (i.e. Strassen's matrix multiplication) is $O(n^{2.8074})$. This approach also simplifies other problems, such as the Tower of Hanoi.
- This approach is suitable for multiprocessing systems.
- It makes efficient use of memory caches.

Divide and Conquer Algo.

Applications:

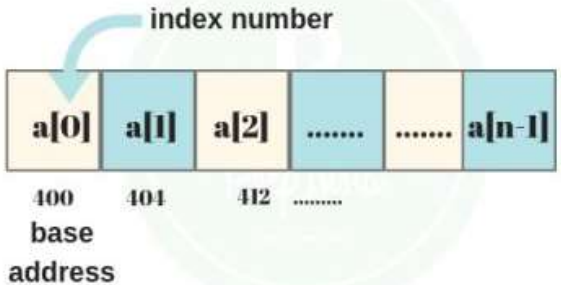
- Binary Search
- Merge Sort
- Quick Sort
- Strassen's Matrix Multiplications
- Karatsuba Algorithm

Array

Instead of creating separate variables to store data, it is an efficient organization to store data in a single variable called array

Definition: An array is defined as a collection of items stored at contiguous memory locations under the same name

Ex: int (a,b,c,d,e)can be grouped in a single variable as int a[5]: now five co location are assigned with the same n



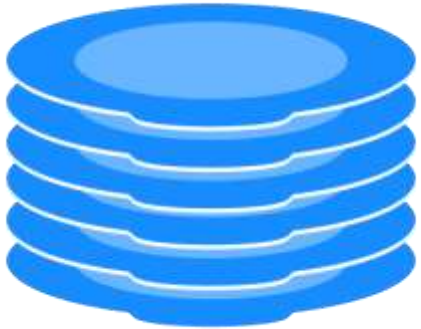
Array

```
#include<iostraem>
using namespace std;
int main()
{
int a[5]={6,9,1,5,3};                    //direct declartion and initialisation
cout<<"elements are:\n";
for(int i=0;i<5;i++)                     // iterate all the elements
cout<<"a["<<i<<"]="<<a[i]<<"\n";          //access though index
return 0;
}
```

Array: Imp Points

1. An array index cannot be negative or zero.
2. Unused space in the array is always filled with zeros.
3. An array elements are Garbage values before initialization of values.
4. Declaring an array without size is invalid.

STACK Data Structure



- A stack is a useful data structure in programming.
- It is just like a pile of plates kept on top of each other.

Think about the things you can do with such a pile of plates

•Put a new plate on top
•Remove the top plate

If you want the plate at the bottom, you must first remove all the plates on top. Such an arrangement is called **Last In First Out** - the last item that is the first item to go out.

LIFO Principle of Stack

In programming terms, putting an item on top of the stack is called **push** and removing an item is called **pop**.