# Subject Notes
# CS-302 – Data Structure
# Unit -1

## REVIEW OF C PROGRAMMING LANGUAGE:

C is a general-purpose, imperative computer programming language, supporting structured programming, lexical variable scope and recursion, while a static type system prevents many unintended operations. By design, C provides constructs that map efficiently to typical machine instructions, and therefore it has found lasting use in applications that had formerly been coded in assembly language, including operating systems, as well as various application software for computers ranging from supercomputers to embedded systems.
C was originally developed by Dennis Ritchie between 1969 and 1973 at Bell Labs.C is an imperative procedural language. It was designed to be compiled using a relatively straightforward compiler, to provide low-level access to memory, to provide language constructs that map efficiently to machine instructions, and to require minimal run-time support.

## INTRODUCTION:

Computer Science is the study of data, its representation and transformation by Computer. For every data object, we consider the class of operations to be performed and then the way to represent the object so that these operations may be efficiently carried out. We require two techniques for this:

- Devise alternative forms of data representation
- Analyse the algorithm which operates on the structure.

These are several terms involved above which we need to know carefully before we proceed. These include data structure, data type and data representation.

A data type is a term which refers to the kinds of data that variables may hold. With every programming language there is a set of built-in data types. This means that the language allows variables to name data of that type and provides a set of operations which meaningfully manipulates these variables. Some data types are easy to provide because they are built-in into the computer's machine language instruction set, such as integer, character etc. Other data types require considerably more efficient to implement. In some languages, these are features which allow one to construct combinations of the built-in types( like structures in 'C'). However, it is necessary to have such mechanism to create the new complex data types which are not provided by the programming language. The new type also must be meaningful for manipulations. Such meaningful data types are referred as abstract data type.

## DATA STRUCTURE:

Data Structures are the programmatic way of storing data so that data can be used efficiently. Almost every enterprise application uses various types of data structures in one or the other way.
Thus, a data structure is the portion of memory allotted for a model, in which the required data can be arranged in a proper fashion.:Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage. For example, we have data player's name "Virat" and age 26. Here "Virat" is of **String** data type and 26 is of **integer** data type.
**Programming:** Programming is the process of taking an algorithm and encoding it into a notation, a programming language, so that it can be executed by a computer. Although many programming languages and many different types of computers exist, the important first step is the need to have the solution. Without an

algorithm there can be no program.Computer science is not the study of programming. Programming, however, is an important part of what a computer scientist does. Programming is often the way that we create a representation for our solutions. Therefore, this language representation and the process of creating it becomes a fundamental part of the discipline.

## CONCEPTS OF DATA AND INFORMATION:

**Data:** Data are simply values or set of values. A data item refers to a single unit of item.
**Information:** Information is any entity or form that provides the answer to a question of some kind or resolves uncertainty. It is thus related to data and knowledge, as data represents values attributed to parameters, and knowledge signifies understanding of real things or abstract concepts.

## CLASSIFICATION OF DATA STRUCTURES: -

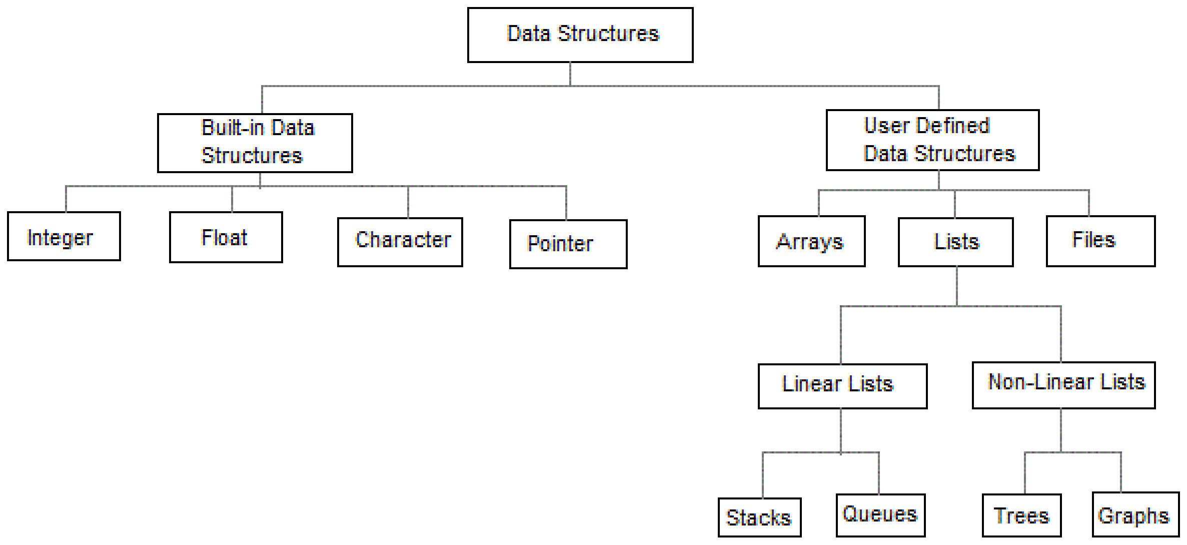

**Figure 1.1: Classification of Data Structure**

A data structure can be broadly classified into
(i) Built-in/Primitive data structure
(ii) User Defined/Non-primitive data structure

**(i)Primitive data structure**

The data structures, typically those data structure that are directly operated upon by machine level instructions i.e. the fundamental data types such as int, float, double incase of 'c' are known as primitive data structures.

**(ii) Non-primitive data structure**

The data structures, which are not primitive, are called non-primitive data structures.
There are two types of non-primitive data structures.

**Linear Data Structures:-**

A list, which shows the relationship of adjacency between elements, is said to be linear data structure. The most, simplest linear data structure is a 1-D array, but because of its deficiency, list is frequently used for different kinds of data.

**Non-linear data structure:-**
A list, which doesn't show the relationship of adjacency between elements, is said to be non-linear data structure.

**Linear Data Structure:**
A list is an ordered list, which consists of different data items connected by means of a link or pointer. This type of list is also called a linked list. A linked list may be a single list or double linked list.

- **Single linked list: -** A single linked list is used to traverse among the nodes in one direction.
- **Double linked list: -** A double linked list is used to traverse among the nodes in both the directions.

A linked list is normally used to represent any data used in word-processing applications, also applied in different DBMS packages.
A list has two subsets. They are: -

- **Stack: -** It is also called as last-in-first-out (LIFO) system. It is a linear list in which insertion and deletion take place only at one end. It is used to evaluate different expressions.
- **Queue:** - It is also called as first-in-first-out (FIFO) system. It is a linear list in which insertion takes place at once end and deletion takes place at other end. It is generally used to schedule a job in operating systems and networks.

**Non-linear data structure:-**

The frequently used non-linear data structures are

- **Trees: -** It maintains hierarchical relationship between various elements
- **Graphs:** - It maintains random relationship or point-to-point relationship between various elements.

The data structures can also be classified on the basis of the following characteristics:

**ABSTRACT DATA TYPES:**
Abstract Data type (ADT) is a type or class for objects whose behavior is defined by a set of value and a set of operations.ADT only mentions what operations are to be performed but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations. It is called "abstract" because it gives an implementation independent view. The process of providing only the essentials and hiding the details is known as abstraction.
Example: Stack, Queue etc.

**IMPLEMENTATION ASPECTS:**

**MEMORY REPRESENTATION:**

There are 2 types of memory allocations possible in C:
Compile time or static allocation
Runtime or Dynamic allocation

1. **Compile time or static allocation:** Compile time memory allocation means reserving memory for variables, constants during the compilation process. So you must exactly know how many bytes you require? This type of allocation is done with the help of Array.The biggest disadvantage of compile time memory allocation, we do not have control on allocated memory. You cannot increase, decrease or free memory, the compiler takes care of memory management. We can also refer compile time memory allocation as static or stack memory allocation.

2. **Runtime or Dynamic allocation:** Memory allocated at runtime either through malloc(),calloc() or realloc().You can also refer runtime memory allocation as dynamic or heap memory allocation. Professional

programmers prefer dynamic memory allocation more over static memory allocation. Since, we have full control over the allocated memory. Which means we can allocate, de-allocate and can also reallocate memory when needed.

a. **malloc():**allocates requested size of bytes and return a void pointer pointing to the first byte the allocated space.
   Syntax: malloc (no. of elements* size of each element);
   For example:
   int *ptr;
   ptr=(int*)malloc(10*sizeof(int));
b. **calloc():** allocate space for an array of element and initialize them to zero and then return a void pointer to memory.
   Syntax: malloc (no. of elements, size of data type);
   For example:
   int *ptr;
   ptr=(int*)calloc(10,2);
c. **realloc():** modify the size of previously allocated space.
   Syntax: ptr=realloc(ptr,newsize);
d. **free():** releases previously allocated memory.

**Memory Allocation Process:**

- Global variable, static variable and program instruction get their memory in permanent storage and where local variable are stored in memory area called stack.
- The memory space between 2 regions is known as Heap area. This region is used for dynamic memory allocation during execution of program. The sizes of heap keep changing.

## OPERATION ON DATA STRUCTURESAND ITS COST ESTIMATION:

The four major operations performed on data structures are:

- **Insertion:** - Insertion means adding new details or new node into the data structure.
- **Deletion:** - Deletion means removing a node from the data structure.
- **Traversal: -** Traversing means accessing each node exactly once so that the nodes of a data structure can be processed. Traversing is also called as visiting.
- **Searching: -** Searching means finding the location of node for a given key value.

Apart from the four operations mentioned above, there are two more operations occasionally performed on data structures. They are:

- **Sorting: -**Sorting means arranging the data in a particular order.
- **Merging: -**Merging means joining two lists.

## INTRODUCTION TO LINEAR DATA STRUCTURES:

### ARRAYS:

Array is set of homogenous data items represented in contiguous memory locations using a common name and sequence of indices starting from 0. Array is a simplest data structure that makes use of computed address to locate its elements. An array size is fixed and therefore requires a fixed number of memory locations.

Suppose A is an array of n elements and the starting address is given then the location and element I will be

$$\mathbf{LOC(A_i) = Base\ address\ of\ A + (i - 1) * W}$$

Where W is the width of each element.

**1 Dimensional Arrays:**A one-dimensional array (or single dimension array) is a type of linear array. Accessing its elements involves a single subscript which can either represent a row or column index.
As an example consider the C declaration intanArrayName[10];
Syntax :datatypeArrayname[sizeofArray];

**2 Dimensional Arrays:**A multidimensional array can be represented by an equivalent one-dimensional array. A two dimensional array consisting of number of rows and columns is a combination of more than 1 one-dimensional array. A 2 dimensional array is referred in two different ways. Considering row as major order or column as major order any array may be used to refer the elements.
**Declaration:** inta[MAX_ROWS][MAX_COLS];
We can initialize all elements of an array to 0 like:
for(i = 0; i < MAX_ROWS; i++)
for(j = 0; j < MAX_COLS; j++)
a[i][j] = 0;

**LINKED LIST:**

A linked list is a linked representation of the ordered list. It is a linear collection of data elements termed as nodes whose linear order is given by means of link or pointer. Every node consists of two parts. The first part is called INFO, contains information of the data and second part is called LINK, contains the address of the next node in the list. A variable called START, always points to the first node of the list and the link part of the last node always contains null value. A null value in the START variable denotes that the list is empty.

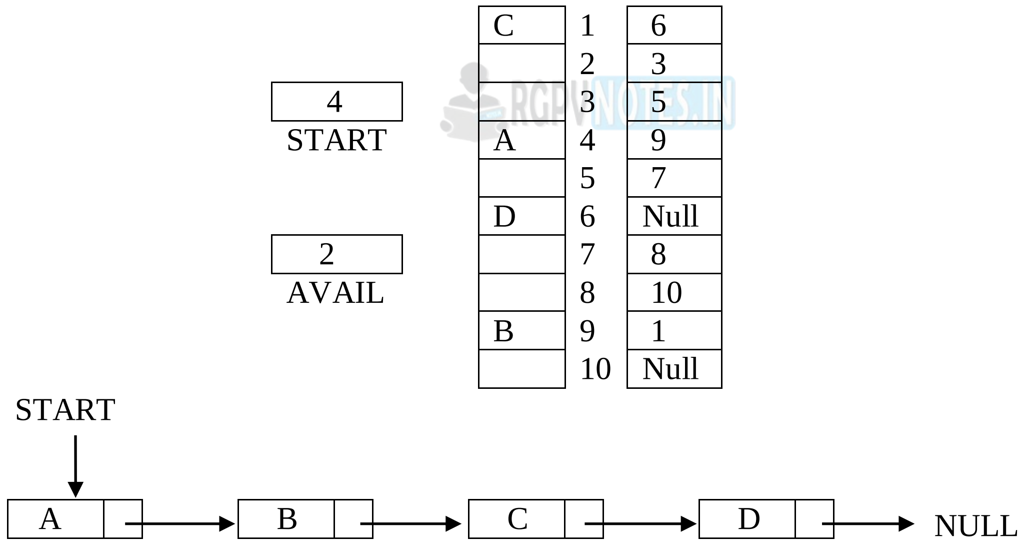

**Figure 1.2: Linked List Representation**

A **linked list** is a way to store a collection of elements. Like an array these can be character or integers. Each element in a linked list is stored in the form of a **node**.

**REPRESENTATION OF LINKED LIST IN MEMORY:**

A linked list can be implemented using structure and pointers.

```
struct LinkedList{
int data;
struct LinkedList *next;
 };
```

malloc() is used to dynamically allocate a single block of memory in C, it is available in the header file stdlib.h.
sizeof() is used to determine size in bytes of an element in C. Here it is used to determine size of each node and sent as a parameter to malloc.

## DIFFERENT IMPLEMENTATION OF LINKED LIST:

We can implement linked list in four ways:

1. Singly Linked List
2. Doubly Linked List
3. Circular Linked List
4. Circular doubly Linked List

**Singly Linked List:**
Singly linked lists contain nodes which have a data part as well as an address part i.e. next, which points to the next node in the sequence of nodes.
The operations we can perform on singly linked lists are insertion, deletion and traversal.

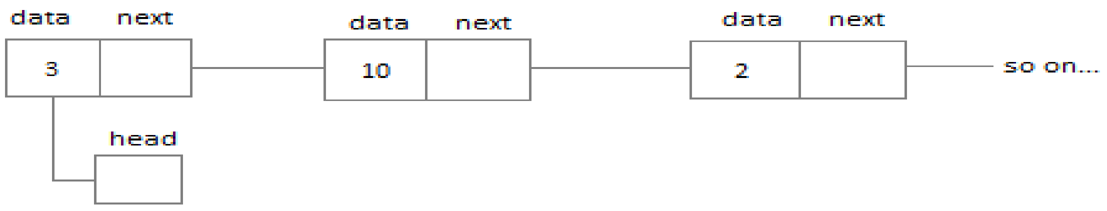

**Figure 1.3: Singly Linked List Representation**

**Doubly Linked List:**
In a doubly linked list, each node contains a **data** part and two addresses, one for the **previous** node and one for the **next** node.

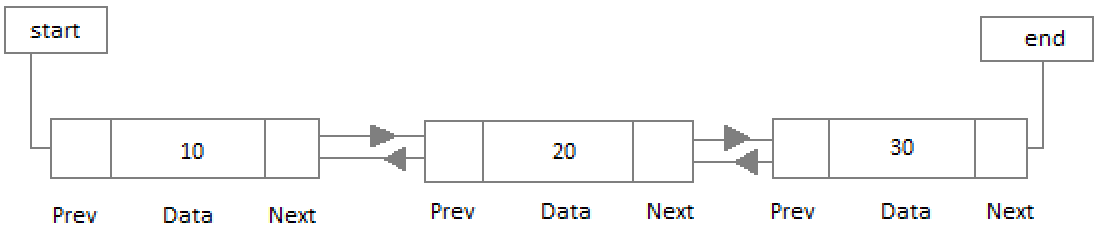

**Figure 1.4: Doubly Linked List Representation**

## CIRCULAR LINKED LIST:

In Circular linked list the last node of the list holds the address of the first node hence forming a circular chain.
Advantages of Circular Linked Lists:

1) Any node can be a starting point. We can traverse the whole list by starting from any point. We just need to stop when the first visited node is visited again.
2) Useful for implementation of queue. Unlike this implementation, we don't need to maintain two pointers for front and rear if we use circular linked list.

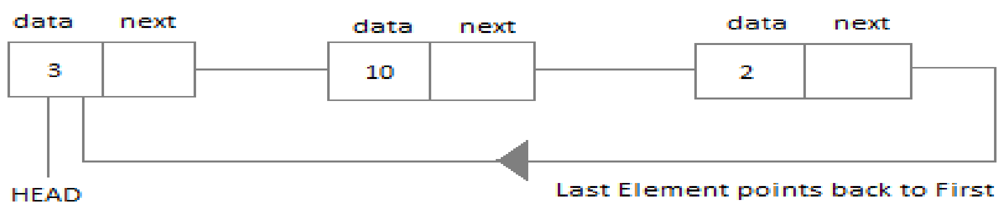

**Figure 1.5: Circular Linked List Representation**

**Circular doubly Linked List:**
Circular Doubly Linked List has properties of both doubly linked list and circular linked list in which two consecutive elements are linked or connected by previous and next pointer and the last node points to first node by next pointer and also the first node points to last node by previous pointer.

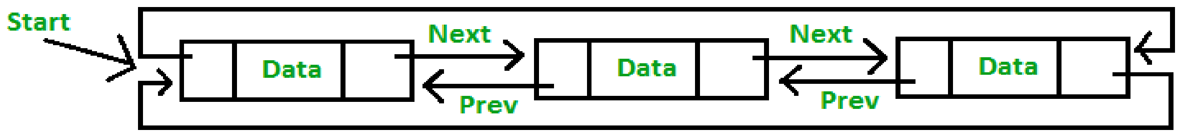

**Figure 1.6: Circular Doubly Linked List Representation**

**<u>APPLICATION OF LINKED LIST:</u>**
**Polynomial Manipulation:**
A polynomial has multiple terms with same information such as coefficient and powers. Each term of a polynomial is treated as a node of a list and normally a linked list used to represent a polynomial. The implementation of polynomial addition is the only operation that is discussed many place. Multiplication of polynomials can be obtained by performing repeated additions.
Each polynomial is stored in decreasing order of by term according to the criteria of that polynomial. i.e. The term whose powers are more are stored at first node and the least power term is stored at last. This ordering of polynomials makes the addition of polynomials easy. In fact two polynomials can be added or multiplied by scanning each of their terms only once.

**Linked Dictionary:**
An important part of any compiler is the construction and maintenance of a dictionary containing names and their associated values. Such dictionary is also called Symbol Table. There may be several symbols corresponding to variable names, labels, literals, etc.
The constraints, which must be considered in the design of the symbol tables, are processing time and memory space. There are many phases associated with the construction of symbol tables. The main phases are building and referencing.
It is very easy to construct a very fast symbol table system, provided that a large section of memory is available. In such case a unique address is assigned to each name. The most straightforward method of accessing a symbol table is linear search technique. This method involves arranging the symbols sequentially in memory via an array or by using a simple linked list. An insertion can be easily handled by adding new element to the end of the list. When it is desired to access a particular symbol, the table is searched sequentially from its beginning until it is found. It will take n/2 comparisons to find a particular symbol.