Chapter 1 Introduction to Data Structures

1.1 Definition

1.1.1 Data and Information

The terms data and information are often used interchangeably. But they are ifferent and used at different levels in information processing life cycle.

Data is raw facts and information is processed data. Let's understand with an example.Data is raw facts. Eg. if we write a name "Arun" and a number 24. These do not have a meaning on its own. Information is processed data. Eg. If we are told that person named "Arun" has age 24. Then both the data are meaningful. When multiple data fields are put together and connections are established between them, then it is information.

Another example can be: say in a company A, B and C employees work in different departments. If we just mentioned the names: A, B and C, it is still data but it has no meaning. When It is told that A, B and C work in same company, it becomes meaningful. When more data is attached that they work in different departments and the formula to calculate their salaries and salary slabs are given then the collective set of data from which conclusions can be drawn is information.

Data is pieces of unconnected facts and information connects these pieces through processing.

This can be taken one step ahead. Suppose in a school, the students name, parents name, house address and marks in five subjects are data about the student. The percentage of student and grade in class is derived from this basic data. The result of the student is information and the collective structure that defines the student is information.

1.1.2 Data Type

Data Types specify the nature of data whether it is integer, character, float and how it is stored in memory. It deals with the amount of space used in memory. The integers, char and float are treated differently. The floats are stored using biased exponent method in 4 bytes. Integers are stored in

big endian or small endian format in 2 bytes or 4 bytes. Char is stored in big endian or small endian format in 1 byte if ASCII or 16 bytes if Unicode.

Data type is different from data structure. Data type is programming language dependent but data structure is independent of any programming language.

1.1.3 Data Structure

Data Structure is Organization of Data in memory; how it is stored, processed and manipulated. It provides the way of storing as well as the operations that can be performed on the data structure.

Lets consider an example

We want to represent data of 20 students in a program and calculate the result of all 20 students. There are four ways:

- One: We introduce a for loop that reads data of one student at a time and calculates percentage. But we cannot find rank in this case and data is lost. The corresponding program is given below:

```
Void main()
{
        int stuname[15];
        int  m1, m2, m3, total, i=0;
        float percentage;
        while ( I < 20)
        {
                scanf("%s,%d,%d,%d", stuname, &m1, &m2, &m3);
                total = (m1 +m2 + m3);
                percentage = (total * 100)/300;
                  printf(" The percentage of %s is %f", stuname, percentage);
                i++;
        }
}
```

Here we read the student data one by one and calculate the percentage. Then forget this student and read data of another student.

- Two: We introduce 20 variables for 20 students. This makes program very lengthy and tedious.

```
void main()
{
        int stuname[15]; stuname2[15],……..stuname20[15];
```

```
        int  m11, m21, m31, m12, m22, m32, ……..m120, m220, m320;
total, i=0;
        float percentage;
        // write following lines for each student.
                scanf("%s,%d,%d,%d", stuname, &m1, &m2, &m3);
                total = (m1 +m2 + m3);
                percentage = (total * 100)/300;
                  printf(" The percentage of %s is %f", stuname, percentage);
}
```

- Three: We make an array of students names, class and marks. This makes rank calculation also easy.

```
void main()
{
        int stuname[20][15];
        int  m1[20], m2[20], m3[20], total[20], i=0;
        float percentage[20];
        float max=0.0f; int indexofMax;
        while ( i < 20)
        {
                scanf("%s,%d,%d,%d", stuname[i], &m1[i], &m2[i], &m3[i]);
                total = (m1[i] +m2[i] + m3[i]);
                percentage[i] = (total[i] * 100)/300;
                  printf(" The percentage of %s is %f", stuname[i], percentage[i]);
                  if (percentage[i] > max)  { max= stuname[i], percentage[i]; indexofMax = i;}
                i++;
        }
        printf("highest percentage is of %s with %f makrs",stuname[indexofMax],
        percentage[indexofMax]);
}
```

- Four: We make a structure of student and declare an array of this structure.

```
struct studentStr
{
          char stuname[15];
        int  m1, m2, m3, total;
        float percentage;
};
void main()
{
        struct studentStr student[20];
        int  i=0; int indexOfMax; float max;
```

```
while ( i < 20)
{
scanf("%s,%d,%d,%d", student[i].stuname, &student[i].m1, &student[i].m2, &student[i].m3);
        student[i].total = (student[i].m1 + student[i].m2 + student[i].m3);
        student[i].percentage = (student[i].total * 100)/300;
          printf(" The percentage of %s is %f", student[i].stuname, student[i].percentage);
           if (student[i].percentage > max)  { max= student[i].percentage; indexofMax = i;}
        i++;
}
printf("highest percentage is of %s with %f makrs",student[indexofMax].stuname, student[indexofMax].percentage);
}
```

- Five: We make a hash table that stores the student record by name as key and makes data access really fast.

Which organization is better? The first three representations require a lot of programming effort and maintenance of data variables. The fourth implementation can make the data variable maintenance simple. Last method makes access faster. There are more representations using linked list and trees.

1.2 Operations on Data Structures

The manipulation of data is very important in data structures. A data structure that we cannot modify is of no use. Various operations that must be provided for are:

1) Creation
2) Insertion
3) Deletion
4) Traversal
5) Searching
6) Updation
7) Sorting
8) Merging
9) Reversal
10) And many more…

1.2.1 Creation:

It is concerned with creating the data structure and allocating space and initializing support variables.

Eg in array defining size and type.

In linked list defining the node structure, start pointer.

In stack defining the array or linked list and maintaining the top variable.

In Queue maintaining the front, rear and array or linked list.

In tree defining the tree structure and root.

In graph defining the nodes and interconnections using matrix.

1.2.2 Insertion

It is adding a new value in the already existing data structure created in creation phase. Value may be added in :

- Beginning of the structure
- End of the structure
- At a given location
- Inserting after a specific element by searching it.
- In sorted order

Eg. If in a class, we have a list of students at the beginning of the session:

Arjun

Birender

Raghav

If a new student with name Aakash joins the insertion is done at top, if with name Ziaur joins insertion is done at end and if with name Nilesh joins then after second element.

1.2.3 Deletion

Removing an element from the data structure. It can be done at:

- Beginning of the structure
- End of the structure
- At a given location
- Deleting a particular element by searching it.

- Deleting in a sorted order structure.

1.2.4 Traversal

Visiting each element of the structure for processing or printing it.

Eg. In the program below, all the elements of an array are visited and printed. Accessing an element is traversal.

```
Void main()
{
        int arr[5], I;
        // scan the elements
        for ( i=0; i<5; i++) scanf("%d", &arr[i] );
    //Traverse the elements
        for ( i=0; i<5; i++) printf("%d", arr[i] );
}
```

Eg. If we have 100 students enrolled in a course in a University. The question paper contains a question that is out of syllabus. Then university decides to give grace marks to all the students. Each student record is visited and processed by adding grace marks to the respective subject marks.

1.2.5 Searching

To find an element in a data structure.

Eg. Searching for an element in an array can be performed using Linear search or Binary search.

1.2.6 Updation

To update or change the data contained in a record is updation.

1.2.7 Sorting

The sorting means arranging the data in ascending or descending order. There are several methods of sorting. The algorithms discussed in this book are:

1) Bubble Sort
2) Selection Sort
3) Insertion Sort
4) Merge Sort
5) Quick Sort

1.2.8 Merging

Merging means to combine two data structures together into one.. Generally the data structures to combine are of same type. Example merging two arrays in sorted order, merging two linked lists, attaching a tree to another as a subtree.

1.2.9 Reversal

Reversing the order of elements. Example if we have a list of names:

Harish

Suresh

Somnath

Tripurari

Then reversal will make the list as :

Tripurari

Somnath

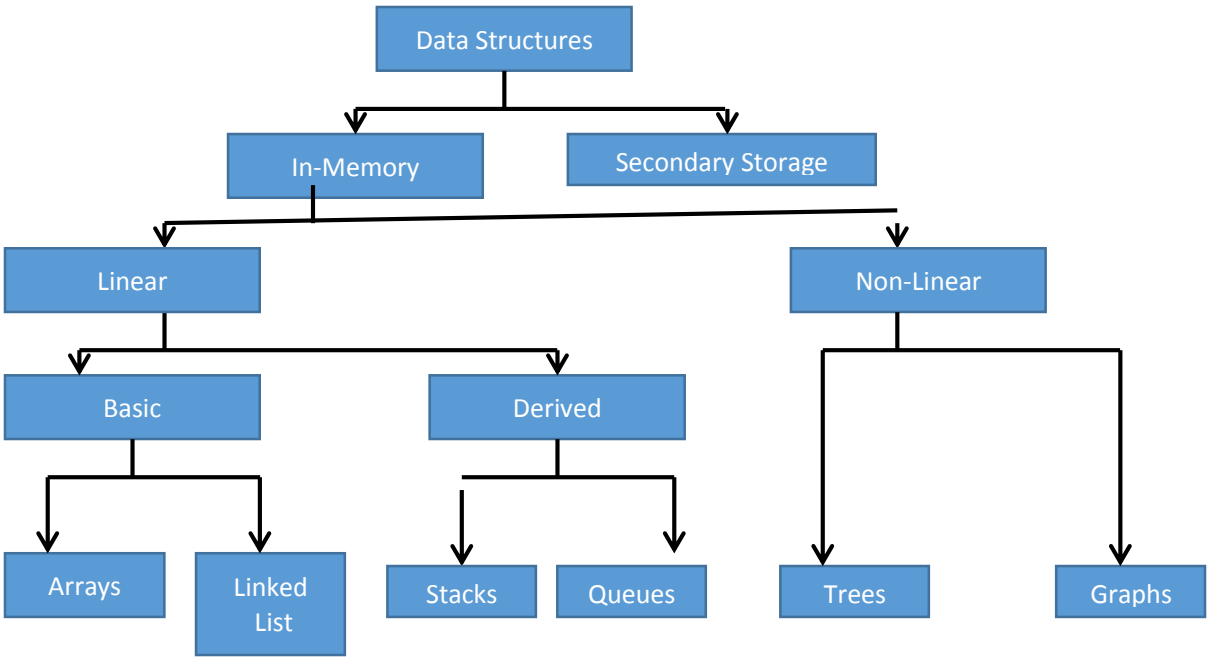
Suresh

Harish

1.3 Types of Data Structures

Data structures can be Memory Data Structures and File Data Structures. Memory Data Structures are used for storage of data in main memory while the program is still running. For example, how to represent the data in the program using variables, arrays, linked lists, heaps, stacks, queues, tree, graph and so on. File Data Structures are used for storage of data on the secondary storage. The representation will vary according to the media on which the storage will be done. Example, in oracle the tables are logically represented as table structure or relations but internally the tables are stored using pointers and complex oracle customized data structures. The indexed files, FAT file system, NTFS file system are all examples of disk data structures.

Linear data Structures are those which have a sequential order among its elements. We can very clearly determine which element is first, then second, and so on. Eg. In an array of 10 elements it is clear The first element is at index 0, second at index 1 and so on. These data structures are flat and sequential.



Non-Linear Data Structure are those which have a non-sequential order and which can be arranged in a number of ways. Eg. Tree is hierarchical in nature. Graphs are peer to peer network. We can move through or traverse these data structures in more than one way.

1.3.1 Array

An array is a collection of homogeneous or similar elements. All the elements have same name. To refer to each individual element, index number is used.

For declaring an array, we specify its size and data type

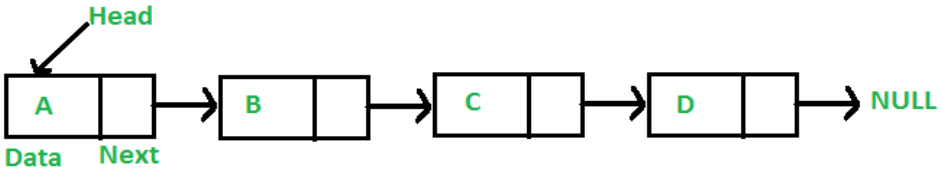
Eg. int arr[5] = {25,35,45,55,65};

Array index	**Arr[0]**	**Arr[1]**	**Arr[2]**	**Arr[3]**	**Arr[4]**
Value	25	35	45	55	65
Memory address	240	242	244	246	248

1.3.2 Linked list

In an array, bulk of memory is allocated at the time of declaration and the size of array cannot be modified. Even for dynamic array the amount of effort for resizing and copying an array is huge. In case the random access to individual elements is not a requirement of the application, then the linked list can be used.

Linked list is created by creating links between scattered memory allocations creating a logical list that is physically scattered in different parts of memory. This allows for linked list to allocate just as many locations as are the elements. As new element keep adding the size of linked list can increase.



There is a data or info part containing A, B, C and D. And next pointer of type struct node that contains address of next node.

Head or start is a pointer of type of struct node that points to first node.

There is no way of reaching D directly. One has to traverse through the links created in above diagram between A, B, C and D to access D.

Take a example of a elements 1,2 ,3 4,6

1	->	**2**	->	**3**	->	**4**	->	**6**	->	**Null**	

The information 1,2,3, etc are placed inside a node structure. This node structure has two parts- info for information and next for pointer to next list item.

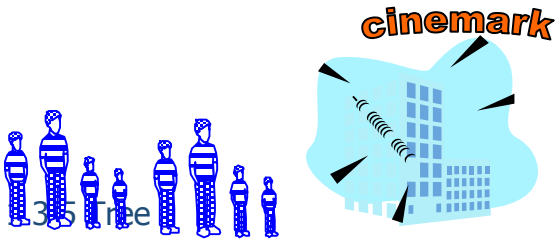
```
struct node
{               int info;
        struct node * next;
};
```

1.3.3 Stack

Stores a set of elements in a particular order. Stack principle: LAST IN FIRST OUT = LIFO.It means: the last element inserted is the first one to be removed

1.3.4 Queue

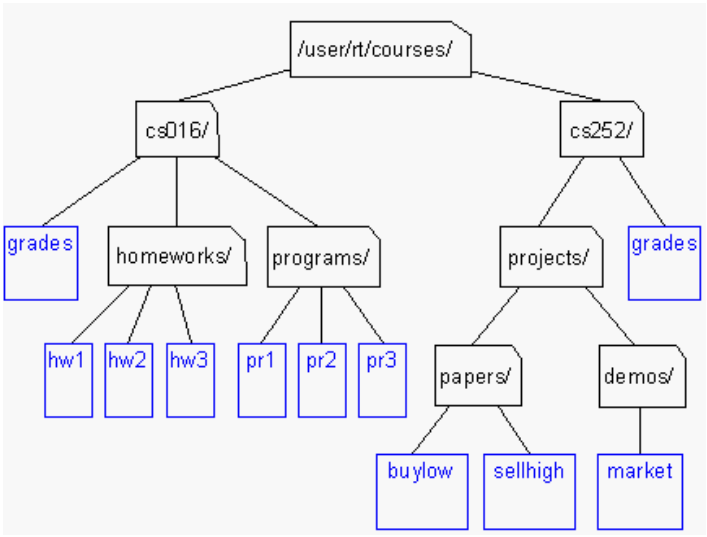
Stores a set of elements in a particular order. Stack principle: FIRST IN FIRST OUT = FIFO. It means: the first element inserted is the first one to be removed.



1.3.5 Tree

It is a non-linear data structure or Hierarchical data structure. By hierarchial we mean, there is a parent child relation among the nodes.

Example: Unix / Windows file structure. The root is the main drive c: or \ is at the top. Then come all the folders which further have subdirectories. To visit a internal folder we have to go through the root to that folder path.



1.3.6 Graph

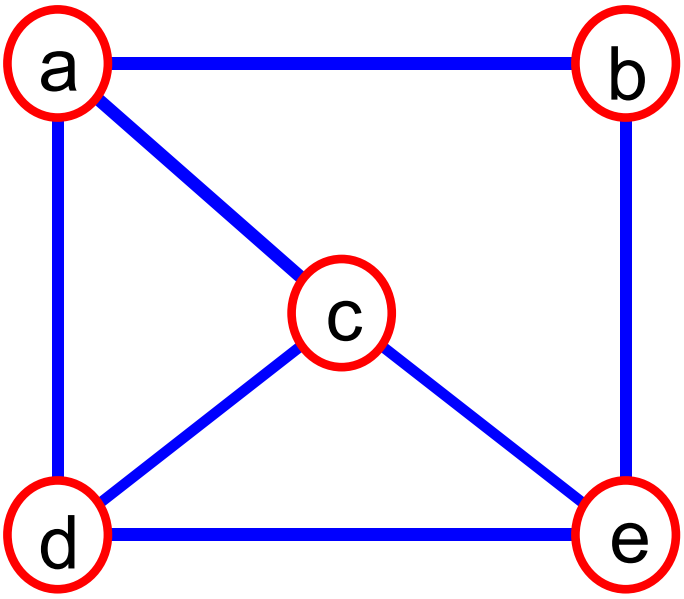
A graph G = (V,E) is composed of:

V: set of vertices

E: set of edges connecting the vertices in V

An edge e = (u,v) is a pair of vertices

Example:



V= {a,b,c,d,e}

E= {(a,b),(a,c),(a,d),
(b,e),(c,d),(c,e),(d,e)}

1.4 Conclusion

From all the above discussion it follows that-

> *"Data Structure is organization of data and operations on the data. The organization means whether the data structure is linear or nonlinear. The operations means how the data is traversed, added, deleted and searched for."*