**Chapter One**

**Introduction to Data Structure**

* 1. **Introduction**

A program is written in order to solve a problem. A solution to a problem actually consists of two things:

* A way to organize the data---------------------------------------------Data Structure
* Sequence of steps to solve the problem ------------------------------Algorithm

The way data are organized in a computer’s memory is said to be Data Structure and the sequence of computational steps to solve a problem is said to be an algorithm. Therefore, a program is nothing but data structures plus algorithms.

Data structure is the structural representation of logical relationships between elements of data. In other words a data structure is a way of organizing data items by considering its relationship to each other.

Data structure mainly specifies the structured organization of data, by providing accessing methods with correct degree of associability. Data structure affects the design of both the structural and functional aspects of a program.

Algorithm + Data Structure = Program

Data structures are the building blocks of a program; here the selection of a particular data structure will help the programmer to design more efficient programs as the complexity and volume of the problems solved by the computer is steadily increasing day by day. The programmers have to strive hard to solve these problems. If the problem is analyzed and divided into sub problems, the task will be much easier.

A complex problem usually cannot be divided and programmed by set of modules unless its solution is structured or organized. This is because when we divide the big problems into sub problems, these sub problems will be programmed by different programmers or group of programmers. But all the programmers should follow a standard structural method so as to make easy and efficient integration of these modules. Such type of hierarchical structuring of program modules and sub modules should not only reduce the complexity and control the flow of program statements but also promote the proper structuring of information. By choosing a particular structure (or data structure) for the data items, certain data items become friends while others loses its relations.

The representation of a particular data structure in the memory of a computer is called a storage structure. That is, data structure should be represented in such a way that it utilizes maximum efficiency. The data structure can be represented in the both main memory and auxiliary memory of the computer. A storage structure representation in auxiliary memory is often called a file structure.

It is clear from the above discussion that the data structure and the operations on organized data items can integrally solve the problem using a computer

Data structure = organized data +Operations

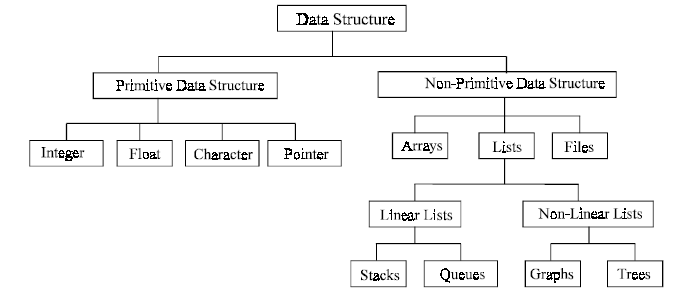
**Classification of Data Structure**

Data structures are broadly divided into two:

1. *Primitive data structures:* These are the basic data structures and are directly operated upon by the machine instructions, which is in a primitive level. They are integers, floating point numbers, characters, string constants, pointers etc.

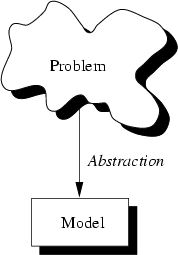
These primitive data structures are the basis for the discussion of more sophisticated (non-primitive) data structures.

1. *Non-primitive data structures:* It is a more sophisticated data structure emphasizing on structuring of a group of homogeneous (same type) or heterogeneous (different type) data items. Array, list, files, linked list, trees and graphs fall in this category.



* 1. **Abstract Data Type and Abstraction**

Given a problem, the first step to solve the problem is obtaining one’s own abstract view, or *model*, of the problem.This process of modeling is called *abstraction*



The model defines an abstract view to the problem. This implies that the model focuses only on problem related stuff and that a programmer tries to define the *properties* of the problem.

These properties include

* The *data* which are affected and
* The *operations* that are involved in the problem.

With abstraction you create a well-defined entity that can be properly handled. These entities define the *data structure* of the program.

An entity with the properties just described is called an *abstract data type* (ADT).

An ADT consists of an abstract data structure and operations. Put in other terms, an ADT is an abstraction of a data structure.

The ADT specifies:

1. What can be stored in the Abstract Data Type?

2. What operations can be done on/by the Abstract Data Type?

For example, if we are going to model employees of an organization:

* This ADT stores employees with their relevant attributes and discarding irrelevant attributes.
* This ADT supports hiring, firing, retiring … operations.

ADT: is a set of objects together with a set of operations; they are mathematical abstractions, objects such as lists and graphs along with operations can be abstract data types.(Booleans, integers along with operations also an abstract data type)

A data structure is a language construct that the programmer has defined in order to implement an abstract data type.

There are lots of formalized and standard Abstract data types such as Stacks, Queues, Trees, etc.

Do all characteristics need to be modeled?

Not at all

* It depends on the scope of the model
* It depends on the reason for developing the model

Abstraction is a process of classifying characteristics as relevant and irrelevant for the particular purpose at hand and ignoring the irrelevant ones.

Applying abstraction correctly is the essence of successful programming

How do data structures model the world or some part of the world?

* The value held by a data structure represents some specific characteristic of the world
* The characteristic being modeled restricts the possible values held by a data structure
* The characteristic being modeled restricts the possible operations to be performed on the data structure.

**Note**: Notice the relation between characteristic, value, and data structures