INTRODUCTION TO DATA STRUCTURE

A program is said to be efficient when it executes in minimum time and with minimum memory space.

Good program is a program that

- It runs correctly
- It is easy to read and understand
- It is easy to debug and
- It is easy to modify.

DATA STRUCTURE

Definition

A data structure is a particular way of storing and **organizing data either in computer's memory or on the disk storage so that it can be used efficiently**. Data structures are used in almost every program or software system. Common examples of data structures are arrays, linked lists, queues, stacks, binary trees, and hash tables etc.

Applications of Data Structures

- Compiler design
- Operating system
- Statistical analysis package
- DBMS
- Numerical analysis
- Simulation
- Artificial Intelligence

The major data structures used in the Network data model is graphs, Hierarchical data model is trees, and RDBMS is arrays. Specific data structures are essential ingredients of many efficient algorithms as they enable the programmers to manage huge amounts of data easily and efficiently.

When selecting a data structure to solve a problem, the following steps must be performed.

1.Analysis of the problem to determine the basic operations that must be supported. For example, basic operation may include inserting/deleting/searching a data item from the data structure.

2. Quantify the resource constraints for each operation.

3. Select the data structure that best meets these requirements.

In this approach, there are three concern

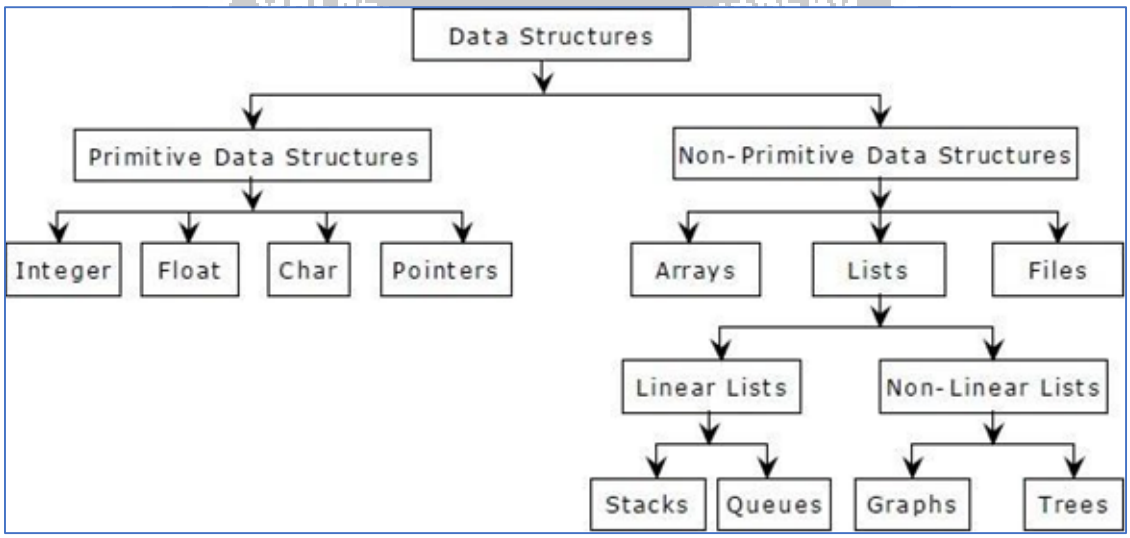
- The first concern is data and the operations that are to be performed on them.
- The second concern about representation of the data.
- The third concern about the implementation of that representation.

ELEMENTARY DATA STRUCTURE ORGANIZATION

Data structures are building blocks of a program. The term data means a value or set of values. It specifies either the value of a variable or a constant. A record is a collection of data items. A file is a collection of related records. Each record in a file may consist of multiple data items but the value of a certain data item uniquely identifies the record in the file. Such a data item K is called a primary key, and the values K1, K2 ... in such field are called keys or key values.

CLASSIFICATION OF DATA STRUCTURES

Data structures are generally categorized into two classes: primitive and non-primitive data structures.



- Primitive data structures are the fundamental data types which are supported by a programming language. Some basic data types are integer, real, character, and boolean.
- Non-primitive data structures are those data structures which are created using primitive data structures. Examples of such data structures include linked lists, stacks, trees, and graphs. Non-primitive data structures can further be classified into two categories: linear and non- linear data structures.

LINEAR AND NON-LINEAR STRUCTURES

LINEAR DATA STRUCTURES

If the elements of a data structure are stored in a linear or sequential order, then it is a linear data structure. Examples include arrays, linked lists, stacks, and queues. Linear data structures can be represented in memory in two different ways. One way is to have to a linear relationship between elements by means of sequential memory locations. The other way is to have a linear relationship between elements by means of links.

NON-LINEAR DATA STRUCTURES

If the elements of a data structure are not stored in a sequential order, then it is a non-linear data structure. The relationship of adjacency is not maintained between elements of a non-linear data structure. Examples include trees and graphs.

OPERATIONS ON DATA STRUCTURES

Traversal: Visit every part of the data structure.

Search: Traversal through the data structure for a given element.

Insertion: Adding new elements to the data structure.

Deletion: Removing an element from the data structure.

Sorting: Rearranging the elements in some type of order (e.g Increasing or Decreasing).

Merging: Combining two similar data structures into one.

ABSTRACT DATA TYPE

An abstract data type (ADT) is the way we look at a data structure, **focusing on what it does and ignoring how it does its job.** The abstract data type is a triple of D-set of Domains, F-Set of functions, A-Axioms in which only what is to be done is mentioned but how is to be done is not mentioned.

ADT = Type + Function Names + Behavior of each function

Examples:

- Stacks
- Queues
- Linked List

ADT Operations

- While modeling the problems the necessary details are separated out from the unnecessary details. This process of modeling the problem is called abstraction.
- The model defines an abstract view to the problem. It focuses only on problem related stuff and that you try to define properties of the problem.

These properties include

- The data which are affected
- The operations which are identified.

Abstract data type operations are

Create: create the database.

Display: displaying all the elements of the data structure.

Insertion: elements can be inserted at any desired position.

Deletion: desired element can be deleted from the data structure.

Modification: any desired element can be deleted from the data structure.

Advantage of using ADTs

- It is reusable, robust
- It can be re-used at several places and it reduces coding efforts
- Encapsulation ensures that data cannot be corrupted
- The Working of various integrated operation cannot be tampered with by the application program
- ADT ensures a robust data structure