Introduction to Data Structures

Design good program

A good program is defined as a program that

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

- A program should undoubtedly give correct results, but along with that it should also run efficiently.
- A program is said to be efficient when it executes in **minimum time** and with **minimum memory space**.
- In order to write efficient programs we need to apply certain data management concepts.
- The concept of data management is a complex task that includes activities like data collection, organization of data into appropriate structures, and developing and maintaining routines for quality assurance.

Data structure

- A **data structure** is basically a group of data elements that are put together under one name, and which defines a particular way of storing and organizing data in a computer so that it can be used efficiently.
- Data structures are used in almost every program or software system. Some common examples of data structures are arrays, linked lists, queues, stacks, binary trees, and hash tables.
- Data structures are widely applied in the following areas:
  - Compiler design
  - Operating system
  - DBMS
  - Artificial intelligence etc.

Criteria for selection

- 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.

Basic Terminology

- The term **data** means a value or set of values. It specifies either the value of a variable or a constant.

  e.g., marks of students, name of an employee, address of a customer, value of pi, etc.

- While a data item that does not have subordinate data items is categorized as an elementary item, the one that is composed of one or more subordinate data items is called a **group item**.

  e.g., a student's name may be divided into three sub-items—first name, middle name, and last name—but his roll number would normally be treated as a single item.

- A **record** is a collection of data items.

  e.g., the name, address, course, and marks obtained are individual data items. But all these data items can be grouped together to form a record.

- A **file** is a collection of related records.

  e.g., if there are 60 students in a class, then there are 60 records of the students. All these related records are stored in a file.

Classification of Data Structures: **Primitive Non-primitive**

- 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. The terms 'data type', 'basic data type', and 'primitive data type' are often used interchangeably.
- **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 data structure

- 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 structure

- 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.

Types of Data Structures

- Arrays
- Stack
- Queue
- Linked List
- Trees
- Graphs