# International Islamic University Chittagong

Department of Computer Science & Engineering

Autumn - 2022

## Course Code: CSE-2321

## Course Title: Data Structures

**Mohammed Shamsul Alam**

Professor, Dept. of CSE, IIUC

# Lecture – 1

## Introduction & Overview on Data Structures

# Data

- **Data** means value or set of values. In both the singular and plural form, this term is data. Following are some examples of data:

  34

  09/06/1974

  ISBN 81-203-0000-0

  10, 20, 30, 40, 50

- A ***data item*** refers to a single unit of values. Data items that are divided into sub items are called ***group items***; those that are not are called ***elementary items***. For example, a student’s name may be divided into three sub items – first name, middle name and last name – but the roll number of a student normally be treated as a single item.

# Data

- Collections of data are frequently organized into a hierarchy of **fields (attributes), records (entity) and files (entity set)**.

- An ***entity*** is something that has certain ***attributes*** or properties which may be assigned values. For example, an employee in an organization is an entity. The possible attributes and the corresponding values for an entity is given bellow:

| | | | | |
|---|---|---|---|---|
| Entity | : EMPLOYEE | | | |
| Attributes | : NAME | AGE | SEX | DESIGNATION |
| Values | : Mr. X | 30 | M | Director |

Entities with similar attributes (e.g. all the employees in an organization) constitute an ***entity set***.

# Data

- Each attribute of an entity set has a range of values, and is called the ***domain of attribute***. Domain is the set of all possible values that could be assigned to the particular attribute. For example, in the EMPLOYEE entity, the attribute SEX has domain as {M, F}.

- Each record (entity) in a file (entity set) may contain many field (attribute) items. But the value in a certain field may uniquely determine the record in the file. Such a field K is called a ***primary key***, and the values $k_1$, $k_2$, … in such a field are called ***keys* or *key values*.**

- Example 1.1: Page-1.2 [Data Structures – Seymour Lipschutz]

# Information

- The term **information** is used for data with its attribute(s). In other words, information can be defined as ***meaningful data*** or ***processed data***. For example, 34 is a data but when we say that 34 is the age of a person then it is information.

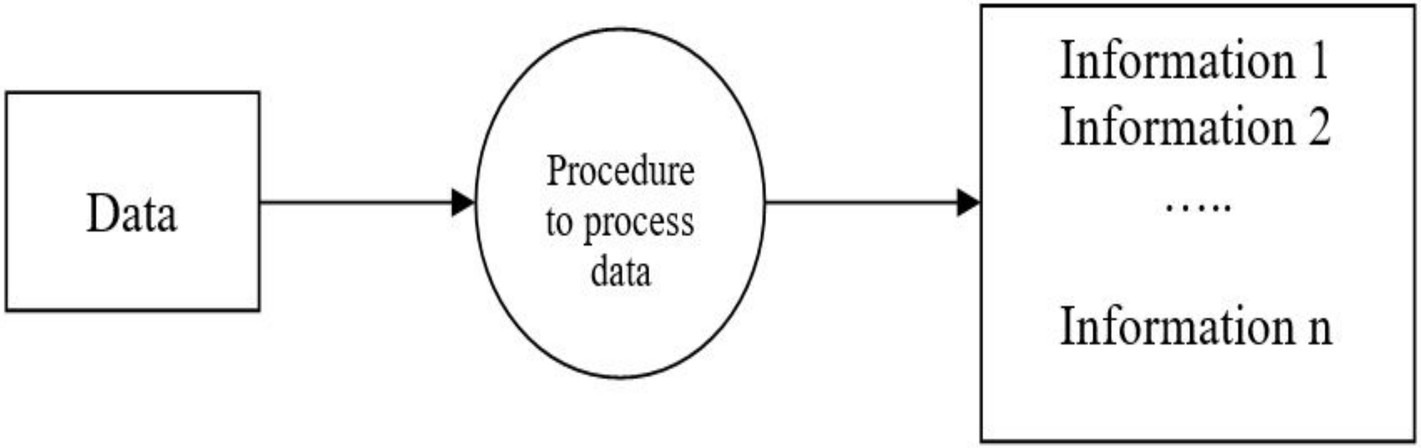


Fig: Relation between data and information

# Data type

A data type is a term which refers to the kind of data that may appear in computation.

❑***Built in data type***: With every programming language, there is a set of data types called built-in data types. For example, in C the built in data types are ***int, float, char, double*** etc.

❑***Abstract data type***: When an application requires a special kind of data which is not available as built-in data type then it is the programmer’s burden to implement his own kind of data. This programmers’ own data type is termed as **abstract data type**. It is also known as ***user defined data type***.

For example, we want to process dates of the form dd/mm/yy. For this, no built in data type is known in C. If a programmer wants to process dates, then an abstract data type, say ***Date***, can be implemented.

❑An abstract data type can be built with the help of built-in data types and other abstract data type(s) already built by the programmer. In C/C++ programmers can define their own data types by using **structure/class**.

# Abstract Data Type (ADT)

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.

- The definition of 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.

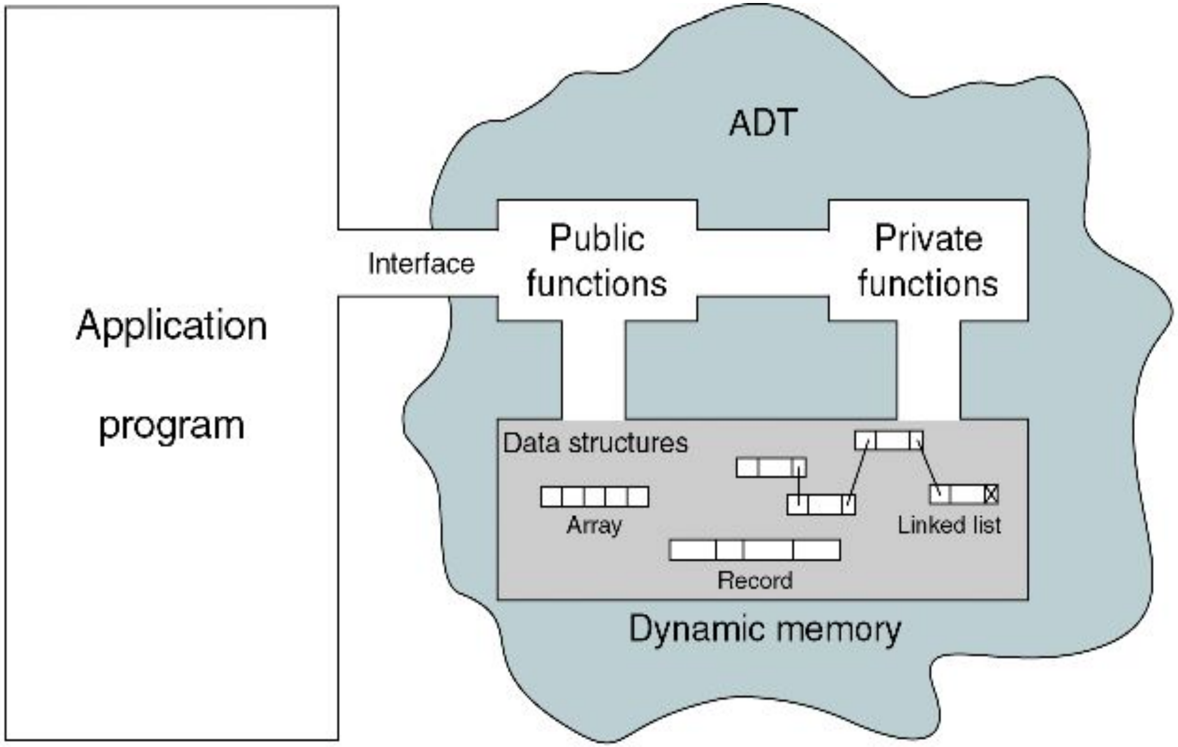


FIGURE 1-2 Abstract Data Type Model

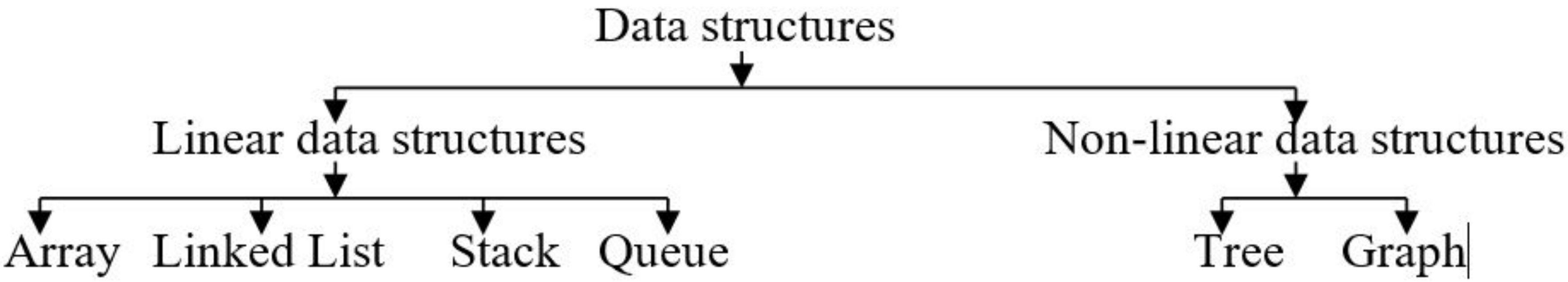
# Data structure

- Data may be organized in many different ways. The logical or mathematical model of a particular organization of data is called a ***data structure***.

- The choice of a particular data model depends on two considerations.
  - It must be rich enough in structure to mirror the actual relationships of the data in the real world.
  - The structure should be simple enough that one can effectively process the data when necessary.

# Classification of Data Structures

- All the data structured can be classified into two main classes: ***linear data structures*** and ***non-linear data structures***.
- In case of linear data structures, all the elements form a sequence or maintain a linear ordering. On the other hand, no such sequence in elements, rather all the elements are distributed over a plane in case of non-linear data structures.

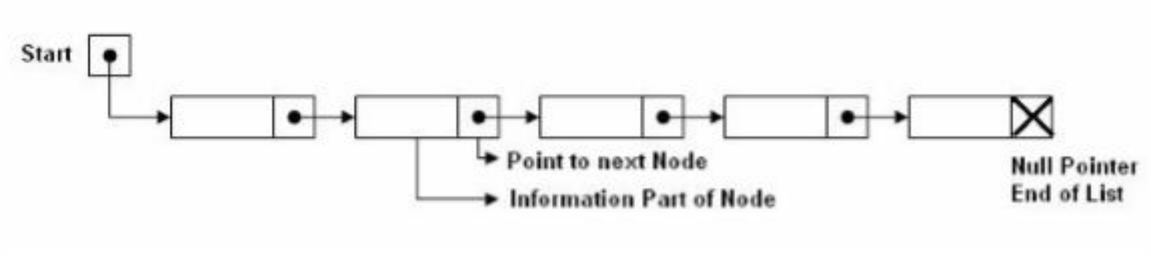


# Array

- An array is a ***finite, ordered*** and collection of ***homogeneous*** data elements. Following are some examples:
  - An array of integers to store the age of all students in a class.
  - An array of strings to store the name of all students in a class.
- There are two types of array: ***one-dimensional*** **array** and ***multi-dimensional*** **array**.
- By an one-dimensional (or linear) array, we mean a list of a finite number n of similar data elements referenced respectively by a set of n consecutive numbers, usually 1, 2, 3, …, n.
- If **A** is the name of an 1D array, then the elements of A are denoted by A[1], A[2], …, A[N].
- The number **K** in A[K] is called a *subscript* and **A[K]** is called a **subscripted variable**.
- We will discuss about multidimensional arrays later on.

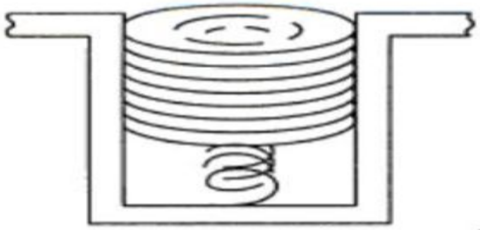
# Linked list

- A ***linked list*** is an ordered collection of finite, homogeneous data elements, called ***nodes***, where the linear order is maintained by means of ***links*** or ***pointers***.
- In linked list, each node is divided into two parts: the first part contains the information of the element, and the second part, called the *link field* or next *pointer field*, contains the address of the next node in the list.
- There is a special pointer ***START*** contains the address of first node in the list. The pointer of the last node contains a special value, called ***null*** pointer.
- The following Fig shows a schematic diagram of a linked list with five nodes.



# Stack & Queue

- A **stack**, also called **a *last-in first-out* (LIFO)** system, is an ordered collection of homogeneous data elements where insertion and deletion operations take place only at one end, called the ***top***. This structure is similar in its operation to a ***stack of dishes*** on a spring system, as shown in Fig (a).
- A **queue**, also called a ***first-in first-out* (FIFO)** system, is an ordered collection of homogeneous data elements where deletion operations can take place only at one end of the list, the "***front***" of the list, and insertion operations can take place only at the other end of the list, the "***rear***" of the list. This structure operates in much the same way as a line of *people* ***waiting at a bus stop***, as shown in Fig. (b).

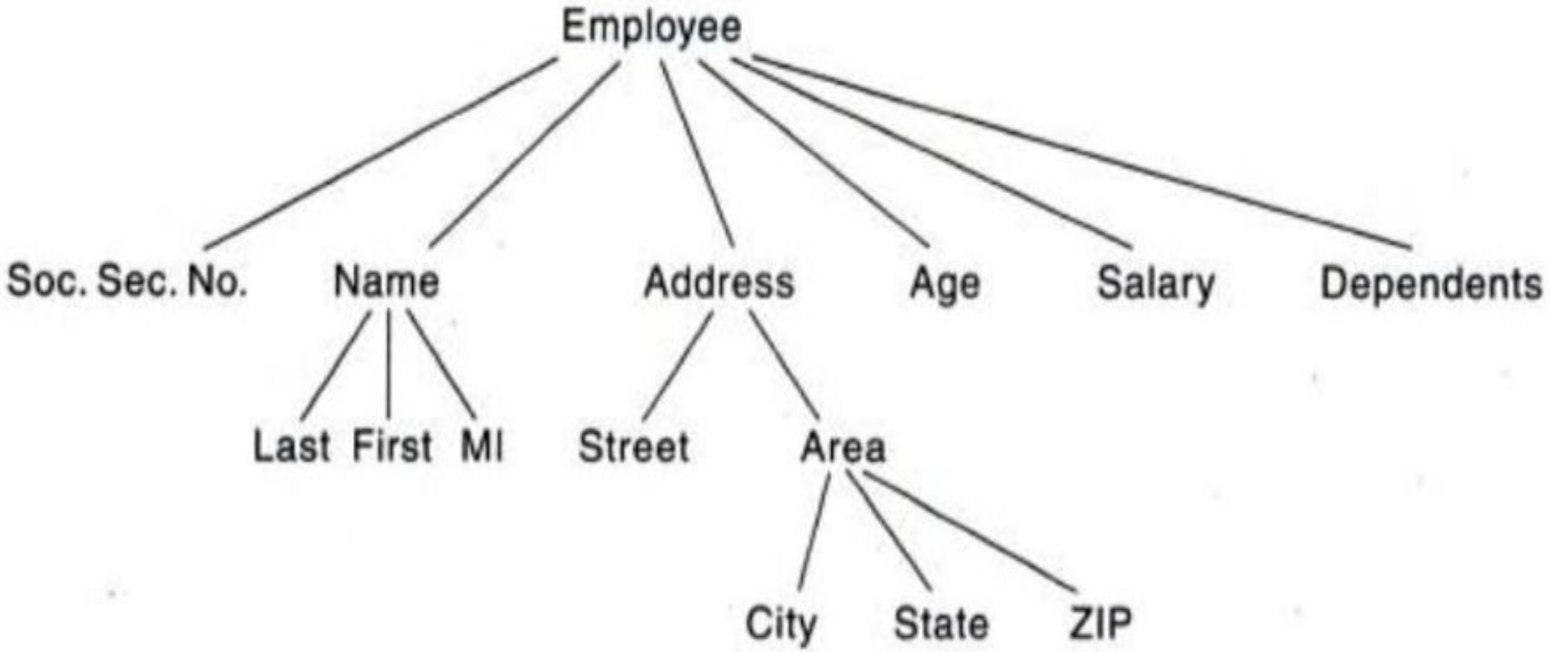


(a) Stack of dishes

(b) Queue waiting for a bus

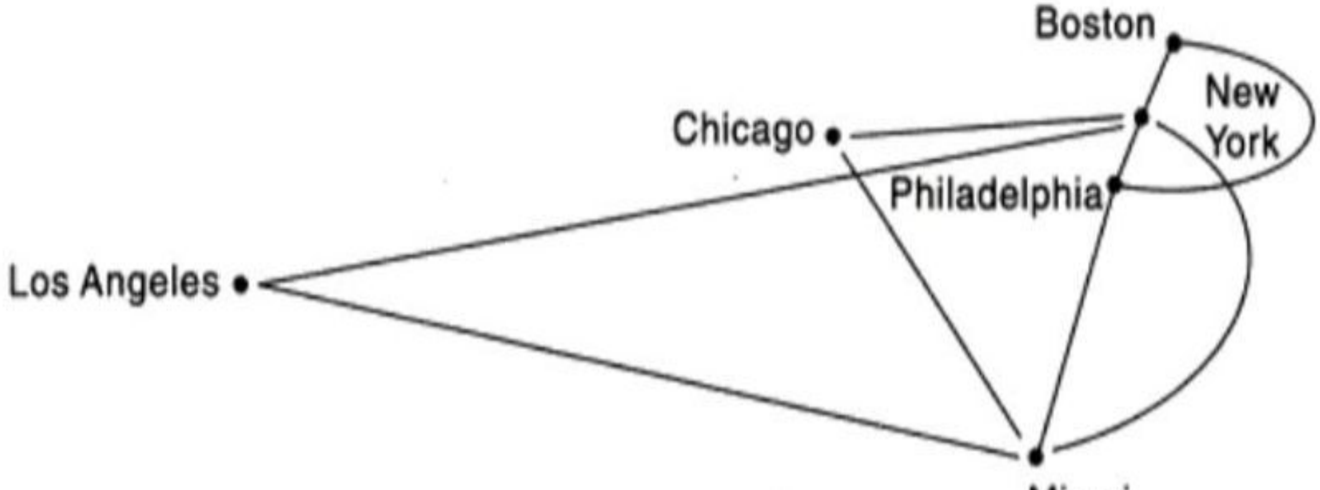
# Tree

- Data frequently contain a hierarchical relationship between various elements. The data structure, which reflects this relationship, is called a **tree**. Fig. 1-7 shows an example of tree.



# Graph

- Data sometimes contain a relationship between pairs of elements, which is not necessarily hierarchical in nature. For example, suppose, an airline flies only between the cities connected by the lines in Fig. The data structure, which reflects this type of relationship, is called a **graph**.



(c) Airline flights

# Data Structure Operations

- **Traversing**: Processing each element in the list
- **Search**: Finding the location of the element with a given value or the record with a given key.
- **Insertion**: Adding a new element to the list.
- **Deletion**: Removing an element from the list.
- **Sorting**: Arranging the elements in some type of order.
- **Merging**: Combining two sorted lists into a single list.

# Complexity of an Algorithm

- An **algorithm** is a finite step-by-step list of well-defined instructions for solving a particular problem. The time and space it uses are two major measurers of the efficiency of an algorithm.
- The **complexity of an algorithm** M is a function f(n) which gives the running time and/or storage space requirement of the algorithm in terms of the size n of the input data.
- Frequently, the storage space required by an algorithm is a simply a multiple of the data size n. Accordingly, the term “complexity“ generally refers to the running time of the algorithm.

- **Searching Algorithms**
  - **Linear Search**:
  - **Binary Search**:

# Time-Space Tradeoff of Algorithms

- The **time-space tradeoff** refers to a choice between algorithmic solutions of a data processing problem that allows one to decrease the running time of an algorithmic solution by increasing the space to store the data and vice-versa.

# Home Task

- **Problem**: Write a C/C++ program to create an array of n elements and then display all the elements of the list.