
AROR UNIVERSITY
OF ART, ARCHITECTURE,
DESIGN & HERITAGE,
SUKKUR, SINDH

# CSC-221

## Data Structures

## Fall- 2025

**Lecture - 01**

# Lecture Outline

- Data structures
- Importance and types.
- Abstract Data Types.
- Time and Space Complexity (Big O)

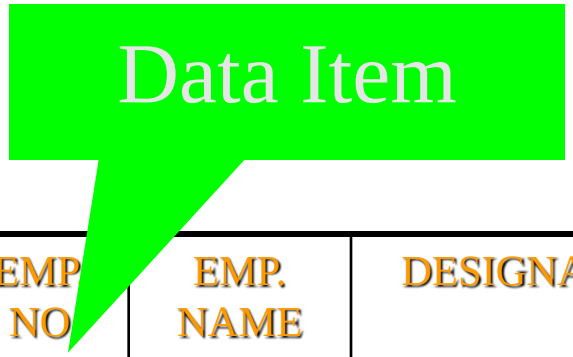
# What is Data?

Data consists of a series of facts or statements that may have been collected, stored and manipulated but have not been organized or placed into context.

When data is organized, it becomes information. In other words, meaningful or processed data.

# What is Data Item?

**A data item refers to a single unit of values, also called a field.**




| EMP NO | EMP. NAME | DESIGNATION | DEPARTMENT | DATE OF JOINING | SALARY |
|---|---|---|---|---|---|
| 101 | AHMED | SALESMAN | MARKETING | 1ST JAN, 05 | 10000 |
| 102 | IMRAN | ANALYST | IT | 5TH SEP, 04 | 25000 |

# What are Group Items?

**Data Items that are divided into sub-items are called group items**

For example, an employee name may be divided into three sub-items

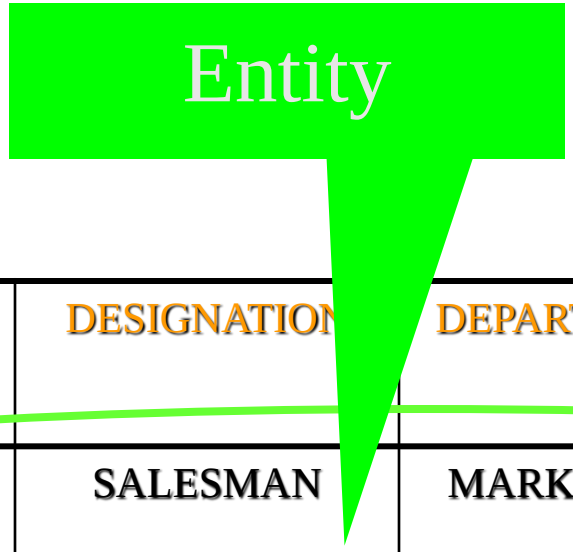
1. First Name
2. Middle Initial
3. Last Name

**Collection of data are frequently organized into a hierarchy of fields, records and files.**

**To make them more precise, we introduce some additional terminology.**

# What is an Entity?

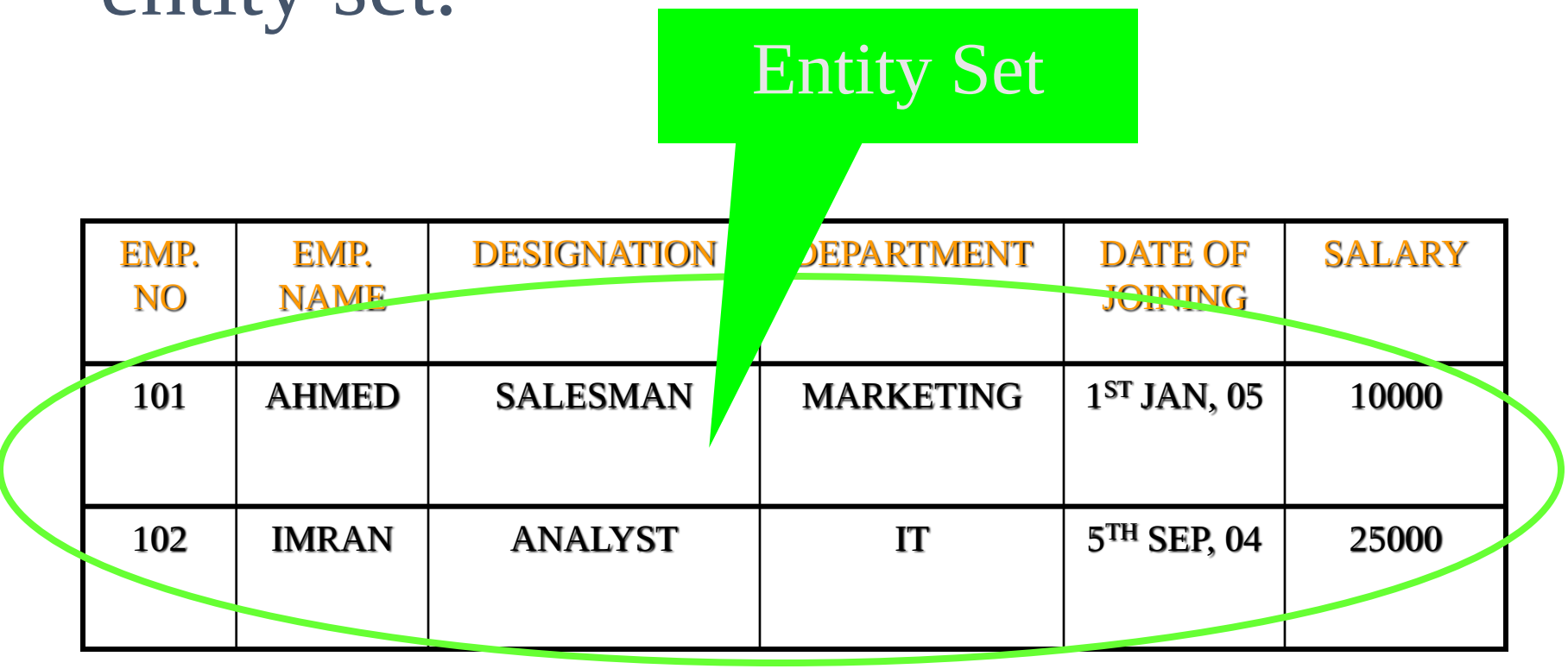
An entity is something that has certain attributes or properties which may be assigned values.




| EMP. NO | EMP. NAME | DESIGNATION | DEPARTMENT | DATE OF JOINING | SALARY |
|---|---|---|---|---|---|
| 101 | AHMED | SALESMAN | MARKETING | 1ST JAN, 05 | 10000 |
| 102 | IMRAN | ANALYST | IT | 5TH SEP, 04 | 25000 |

# What is an Entity Set?

Entities with similar attributes form an entity set.




| EMP. NO | EMP. NAME | DESIGNATION | DEPARTMENT | DATE OF JOINING | SALARY |
|---|---|---|---|---|---|
| 101 | AHMED | SALESMAN | MARKETING | 1ST JAN, 05 | 10000 |
| 102 | IMRAN | ANALYST | IT | 5TH SEP, 04 | 25000 |

# Problem Solving

1. To design an algorithm that is easy to understand, code, and debug.
2. To design an algorithm that makes efficient use of the computer's resources.

# The Need for Data Structures

You might think that with ever more powerful computers, program efficiency is becoming less important. After all, processor speed and memory size still seem to double every couple of years. Won't any efficiency problem we might have today be solved by tomorrow's hardware?

# The Need for Data Structures

- we develop more powerful computers.
- computing power to tackle more complex problems,
- bigger problem sizes, or new problems previously deemed computationally infeasible.
- More complex problems demand more computation, making the need for efficient programs even greater.
- Worse yet, as tasks become more complex, they become less like our everyday experience. Today's computer scientists must be trained to have a thorough

# The Need for Data Structures

understanding of the principles behind efficient program design, because their ordinary life experiences often do not apply when designing computer programs

# The Need for Data Structures

a data structure is any data representation and its associated operations. Even an integer or floating point number stored on the computer can be viewed as a simple data structure.

# The Need for Data Structures

- A solution is said to be efficient if it solves the problem within the required resource constraints.

# The Need for Data Structures

When selecting a data structure to solve a problem, you should follow these steps.

1. Analyze your problem to determine the basic operations that must be supported. Examples of basic operations include inserting a data item into the data structure, deleting a data item from the data structure, and finding a specified data item.

# The Need for Data Structures

2. Quantify the resource constraints for each operation.

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

# The Need for Data Structures

The following three questions, which you should ask yourself whenever you must choose a data structure:

1. Are all data items inserted into the data structure at the beginning, or are insertions interspersed with other operations?
2. Can data items be deleted?
3. Are all data items processed in some well-defined order, or is search for specific data items allowed?

# Costs and Benefits

Each data structure has associated costs and benefits. In practice, it is hardly ever true that one data structure is better than another for use in all situations. If one data structure or algorithm is superior to another in all respects, the inferior one will usually have long been forgotten. For nearly every data structure and algorithm presented in this book, you will see examples of where it is the best choice. Some of the examples might surprise you.

# Costs and Benefits

A data structure requires a certain amount of space for each data item it stores, a certain amount of time to perform a single basic operation, and a certain amount of programming effort. Each problem has constraints on available space and time. Each solution to a problem makes use of the basic operations in some relative proportion, and the data structure selection process must account for this. Only after a careful analysis of your problem's characteristics can you determine the best data structure for the task.

# Abstract Data Types and Data Structures

- A type is a collection of values. For example, the Boolean type consists of the values true and false. The integers also form a type. An integer is a simple type because its values contain no subparts. A bank account record will typically contain several pieces of information such as name, address, account number, and account balance. Such a record is an example of an aggregate type or composite type. A data item is a piece of information or a record whose value is drawn from a type. A data item is said to be a member of a type.

# What is Data Structures?

Data may be organized in many different ways; the logical or mathematical model of a particular organization of data is called a Data Structure.

# What is an Array?

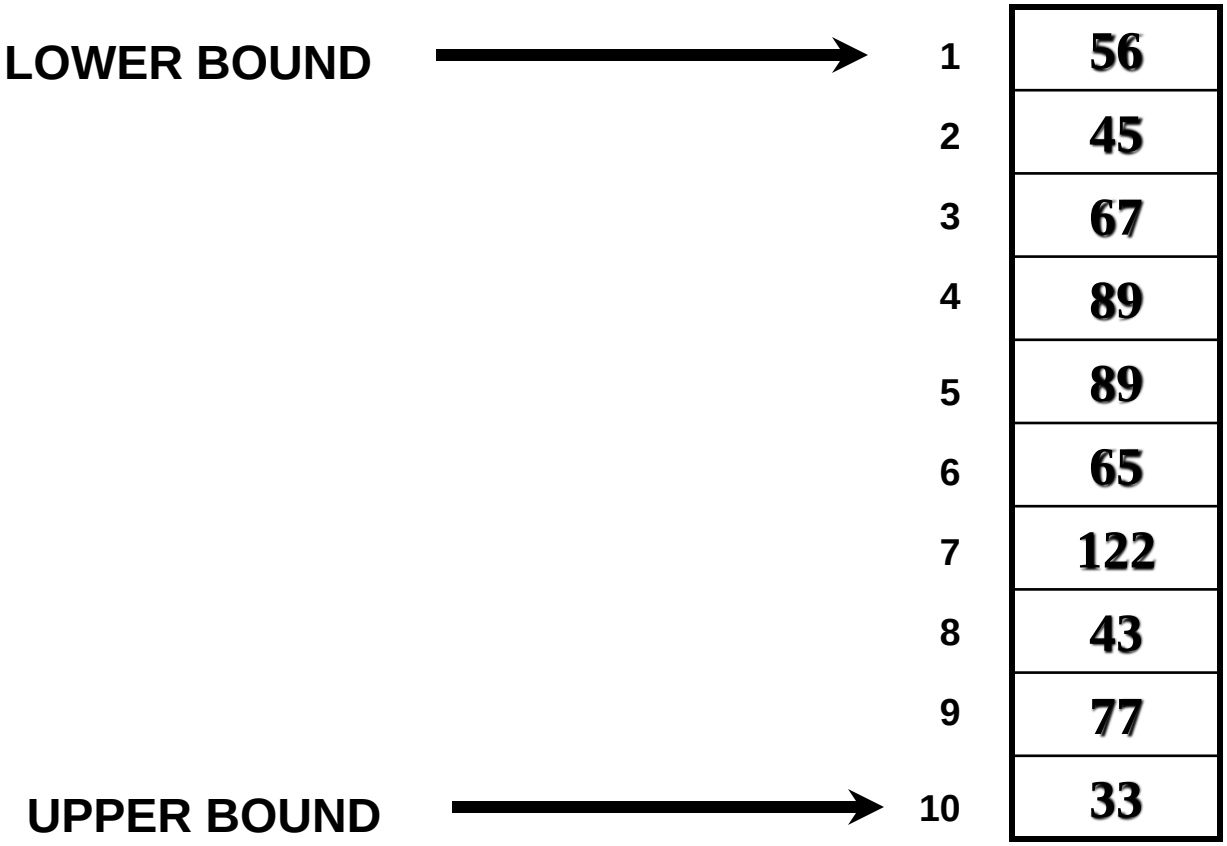
A block of memory divided into separate storage locations. Each storage location is capable of holding a single value of a specific type. All values in the array must be of the same type.

The elements of the array are referenced respectively by an index set consisting of n consecutive numbers.

# TYPES OF ARRAYS

- SINGLE DIMENSION ARRAY
- DOUBLE DIMENSION ARRAY
- MULTI-DIMENSION ARRAY

- UB is the largest index called the  upper bound
- LB is the smallest index called the lower bound



# LINEAR ARRAY ALGORITHMS

# Algorithm-1

## Traversing a Linear Array using for loop

1. Repeat for K = LB to UB
2. [visit element] Apply process to LA(K)

   [End of loop]
3. Exit

# Algorithm-2

## Traversing a Linear Array using while loop

1. **[initialize counter]** set k:=LB
2. Repeat steps 3 and 4 while K ≤ UB
3. **[visit element]** Apply process to LA[K]
4. **[increase counter]** set k:=k+1

   [end of loop]
5. Exit

# Algorithm-3

## Inserting into a Linear Array

**INSERT(LA, N, K, ITEM)**

1. [initialize counter] set J:=N
2. Repeat steps 3 and 4 while J ≥ K
3. [move jth element downward] set LA[J+1]:=LA[J]
4. [decrease counter] set J:=J-1

   [end of loop]
5. [insert element] set LA[K]:=ITEM
6. [reset N] set N:=N+1
7. Exit

# Algorithm-4

## Deleting from a Linear Array

**DELETE(LA, N, K, ITEM)**

1. set ITEM=LA[K]
2. Repeat for J = K to N-1
   set LA[J]:=LA[J+1]
   [end of loop]
3. set N:=N-1
4. Exit

# What is Bubble Sort?

A multiple-pass sorting technique that starts by sequencing the first two items, then the second with the third, then the third with the fourth, and so on, until the end of the set has been reached. The process is repeated until all items are in the correct sequence.

# Algorithm-5

## Bubble Sort

**BUBBLE(DATA, N)**

1. Repeat steps 2 and 3 for K = 1 to N-1
2. Set PTR :=1
3. Repeat while PTR ≤ N-K
   a) if DATA[PTR] > DATA[PTR+1], then
      interchange DATA[PTR] and DATA[PTR+1]
   [end of if structure]
   b) set PTR :=PTR+1
   [end of innter loop]
   [end of outer loop]
4. Exit

# What is Linear Search?

Also known as a sequential search, linear search is a method of how a search is performed. With a linear search each item is examined, one at a time, in sequence, until a matching result is found.

# Algorithm-6

## Linear Search

**LINEAR(DATA, N, ITEM, LOC)**

1. [insert item at the end of DATA] set DATA[N+1] :=ITEM
2. Set LOC :=1
3. Repeat while DATA[LOC] ≠ ITEM

   set LOC :=LOC+1

   [end of loop]
4. Display “Value found at Location” LOC
5. Exit

# What is Binary Search?

Binary search is a searching technique for searching a set of sorted data for a particular value.

- Start search in the middle of array
- if x is less than the middle element, search in lower half
- if x is greater than the middle element, search in upper half

# Algorithm-7

## Binary Search

**BINARY(DATA, LB, UB, ITEM, LOC)**

1. [initialize variables]
   BEG :=LB, END :=UB and MID=INT((BEG+END)/2)
2. Repeat steps 3 & 4 while BEG ≤ END and DATA[MID] ≠ ITEM
3. if ITEM < DATA[MID], then:
   set END :=MID-1
   else:
   set BEG :=MID+1
   [end of if structure]
4. Set MID=INT((BEG+END)/2)
   [end of loop]
5. If DATA[MID] = ITEM, then:
   set LOC :=MID
   else:
   set LOC :=NULL
   [end of if structure]
6. Exit.


Q & A