Islamic University of Technology (IUT)

Organization of Islamic Cooperation (OIC)

Department of Electrical and Electronic Engineering (EEE)

COURSE NO	:	EEE 4416
LAB NO	:	02 (Part A)
TOPIC	:	INTRODUCTION TO DATA STRUCTURES

DATA STRUCTURES

A data structure is a way of organizing and storing data so that it can be accessed and modified efficiently. They are the building blocks of programming. It provides the means to manage large amounts of data for use in different applications. If the data is not organized properly, it is very difficult to perform any task on it.

Types of Data Structures

Data structures can be broadly classified into two categories:

1. Primitive Data Structures

These are the basic data types provided by most programming languages:

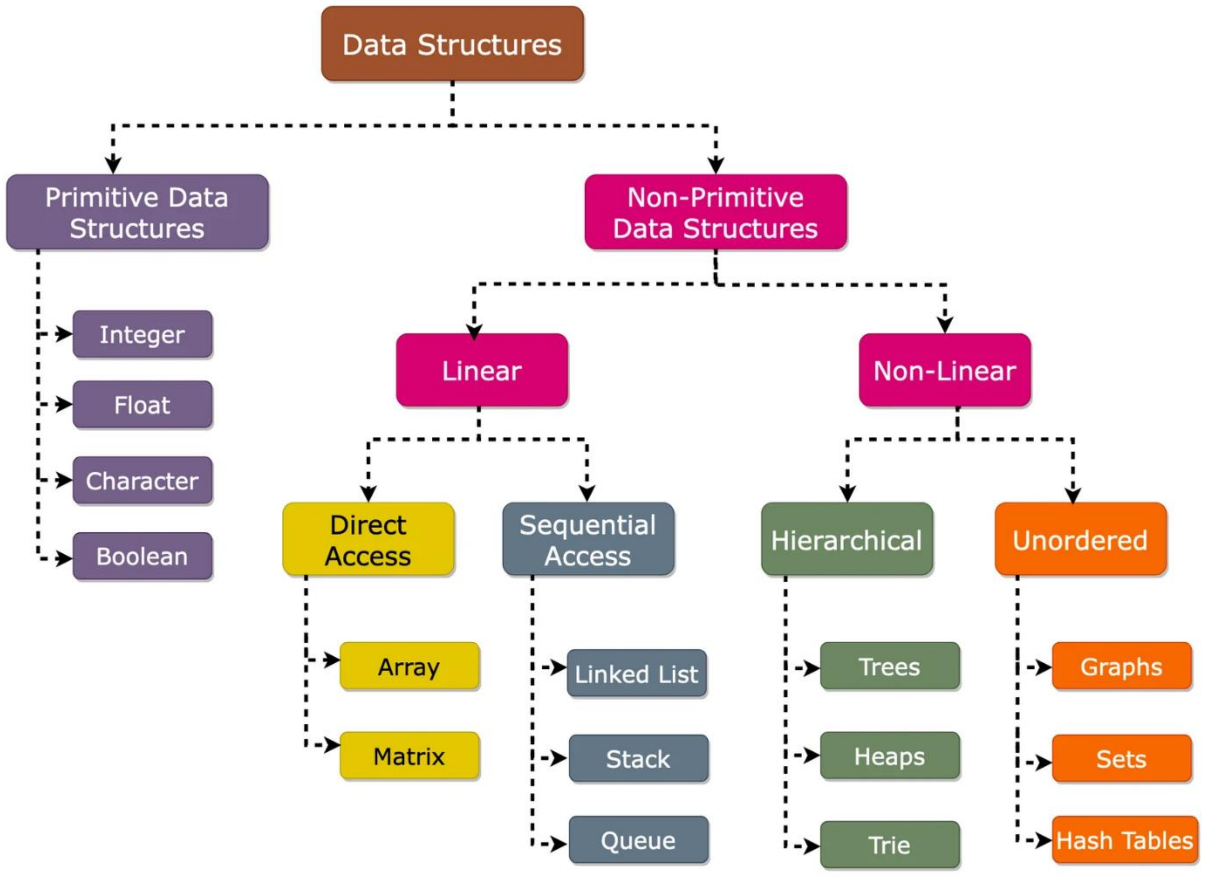
- Integer
- Float
- Character
- Boolean

2. Non-Primitive Data Structures

These are more complex and are built using primitive types. They can be further classified into two categories.

- Linear Data Structures: Elements are arranged sequentially. Examples include –
  - Array
  - Linked List
  - Queue
  - Stack
- Non-Linear Data Structures: Elements are not in a sequential order. Examples include –
  - Trees
  - Graphs
  - Hash Table
  - Heaps

Prepared by: Asif Newaz
Lecturer, EEE, IUT



Why need different data structures

Different problems have different requirements in terms of how data should be stored, accessed, modified, and processed. No single data structure is optimal for all situations — choosing the right one can dramatically improve the efficiency, readability, and scalability of your solution.

An array is a collection of elements stored in contiguous memory locations. Suppose you need to store the IDs of students in a class. A one-dimensional array is ideal for this — it allows indexed access to all the student IDs, meaning you can quickly retrieve or update the ID at a particular position.

Now, imagine you're managing a viva or enrollment session, where students arrive one by one and must be processed in the exact order of arrival. In this case, a queue data structure is more appropriate. A queue follows the FIFO (First-In, First-Out) principle, ensuring that students are handled fairly and in sequence.

Let’s add another layer: you also need to store the marks students achieve in the viva. You could use a separate array to store these marks, but then comes a challenge — how do you link each mark to the correct student ID?

This is where a hash table (also known as a dictionary or map) comes in handy. A hash table allows you to create key-value pairs, where the student ID acts as the key and the viva mark is the value. Just like an Oxford dictionary allows you to quickly look up meanings based on a word, a hash table enables fast and direct access to a student's marks using their ID. This structure is widely used in student management systems for efficiently storing and retrieving information.

```
students = {
   1023: {"name": "Bruce", "gpa": 3.95},
   1024: {"name": "Affleck", "gpa": 3.65},
}
```

Data structures in MATLAB

Different programming languages provide various built-in data structures. It varies from language to language. However, the basic concept remains the same. MATLAB doesn't have "traditional" data structures like C++ or Java (e.g., linked list, stack, etc.) as built-in types, but it offers high-level, flexible structures that can mimic or replace most data structure behaviors.

Here are some common Data Structures in MATLAB –

1. Arrays (Vectors and Matrices)

- MATLAB is array-based. While other programming languages usually work with numbers one at a time, MATLAB operates on whole matrices and arrays.
- Numeric arrays are of homogeneous data types and can only contain numbers.
- A 3rd bracket [ ] is used to create a numeric array.

```
student_ids = [101, 102, 103];   % Row vector

marks = [85; 90; 88];          % Column vector

matrix = [101, 85; 102, 90; 103, 88];    % 2D array (matrix)
```

2. Character Array (char)

- A character array is just like a numeric array, but it stores characters.
- Single quotation ‘ ’ is used to store characters.
- Less flexible than a string array.

```
name = 'Jon Snow'

% This creates a 1×8 array where each letter has a unique
index value including the whitespace character.
```

3. String Array

- A string array is a newer and more flexible way to work with text data.
- Introduced in R2016b, string arrays use double quotes (" ") and offer powerful built-in functions for string manipulation, comparison, and formatting.

```
Given_name = "Alicient";          % Single string

Family_name = "Hightower";          % Single string

name = ["Alicient", "Hightower”]     % String array (1×2)
```

4. Cell array

- Allows storing heterogeneous data types.
- It is not possible to store both numbers and characters in a numeric or character array. They are homogeneous.
- Curly braces or 2nd bracket are used to create a cell array.

```
Info = {'Geralt', ‘male’, 100; 'Triss', ‘female’, 30};
```

5. Structures (struct)

- MATLAB’s version of key-value pairs — similar to a dictionary (Python).
- It can be used for grouping multiple properties.
- A dot ‘.’ is used after the variable name to create a struct datatype.

```
student(1).id = 101;
student(1).name = 'Sherlock';
student(1).gpa = 3.99;

student(2).id = 102;
student(2).name = 'Watson';
student(2).gpa = 3.8;
```

6. Tables

- An efficient way to store and manipulate tabular data.
- It is similar to how data is stored in a spreadsheet.
- You can easily import data from a CSV/Excel file into MATLAB in a TABLE datatype.

7. Categorical

- The categorical data type is used to represent discrete, finite sets of categories or groups, such as gender, grades, status, region, or labels.
- If you create a string array of students, whether they passed or failed, you can group them since MATLAB will treat each string separately.
- But if it is created using a categorical data type, then they can be easily grouped.

```
status = categorical({'Pass', 'Fail', 'Pass', 'Pass', 'pass', 'fail','fail'})

histogram(status)
```

8. Logical

- The logical data type represents Boolean values — true or false. It is primarily used in conditional indexing, bitwise operations, etc.
- *true* and *false* are reserved keywords in MATLAB. (not True with a capital T).
- *Logical 1 and 0* are used to show the values.

```
a = true;      % Logical 1 (TRUE)

b = false;     % Logical 0 (FALSE)
```

- Logical operators:

```
a = 5;
b = 10;

a == b   % false
a ~= b   % true
a < b    % true
a >= b   % false
```

- Logical array:

```
scores = [55, 72, 48, 90];
passed = scores >= 50

% passed = [1, 1, 0, 1]  %logical array
```

9. Date and Time

- MATLAB provides several specialized data types for handling date and time values.
- Dates and times can come in various formats. MATLAB's *datetime* data type allows us to handle all these different formats and perform any required calculations or modifications.
- For instance, the date is written in different formats.
  - 01 Jan, 2025
  - Jan 01, 2025
  - 01 Jan 2025
  - 01. 24. 2025
  - 01. 01. 2025

MATLAB’s datetime datatype makes it easier to process such differences.

- Using *duration* datatype, you can calculate the time differences from timestamps (time math).
- You can use the *timetable* datatype to store variables with a time-based row index.

Data Type	Description
`datetime`	Stores specific points in time (e.g., timestamps)
`duration`	Represents elapsed time in fixed-length units (e.g., hours, minutes)
`calendarDuration`	Represents calendar-based durations (e.g., months, years, days)
`timetable`	A table indexed by time, combining `datetime` with tabular data

```
datetime("now");        % Current date and time
```

```
datetime("now") + days(45)     % 01-Jul-2025 19:27:25
```

This lab introduces several of MATLAB's built-in data structures.

So far, you have already worked with the array data type in the first lab. In today’s session, you will explore additional data types that are commonly used in MATLAB programming. As you progress, you will gradually learn how to apply these data types effectively in different contexts and problem-solving scenarios.