Chapter 1

INTRODUCTION TO DATA STRUCTURES

Objective

- To introduce:
  - Concept and basic terminologies
  - Operations in data structures

CONTENT

- Data Structure?
  - Type
  - Operations

Introduction

- Variable
  - To hold data (in computer science programming)
- Data types
  - System-defined data types (primitive data types) eg. integer, double. – int a;
  - User defined data types
    - Allow user to define their own data types.

Data Structures

- Once we have data in variables, we need some mechanism for manipulating that data to solve problems.
- Data structure is a particular way of storing and organizing data in computer.
- General data structure types: array, linked lists, stacks, queue, trees, graph etc.
- Computer programs need to organize data using data structure.

The Need for Data Structures

- One might think that with more powerful computers, *program efficiency* is becoming less important. But:
  - More powerful computers encourage us to attempt more complex problems.
  - More complex problems demand more calculations.
- A data structure is any data representation and its associated operations.

Efficiency

- A solution is said to be efficient if it solves the problem within the required resource constraints.
  - Space
  - Time
- The cost of a solution is the amount of resources that the solution consumes.
  - Most often, cost is measured in terms of **time**.

Data Structure Philosophy

- Each data structure has costs and benefits.
- Rarely is one data structure better than another in all situations.
- A data structure requires:
  - space for each data item it stores,
  - time to perform each basic operation,
  - programming effort.

Data Structures in real scenario

- Look around and you will see that people organize things:
  - Write TO-DO Lists
  - Queue at ATMs
  - Arrange stacks of books
  - Look-up words in dictionary
  - Organize directories in computers
  - Plan road map to go somewhere

Abstract Data Types (ADT)

- All primitive data types (int, double, etc) support basic operations such addition and subtraction – the system provide implementation for the primitive data types.
- For user-defined data types, we also need to define operations.
- The implementation for these operations can be done when we want to use them – user-defined data types are defined along with their operations.
- ADT is a specification that describes a data set and the operation on that data.
- Each ADT specifies WHAT data is stored and WHAT operations on the data
- ADT DOES NOT specify HOW to store or HOW to implement the operations, so ADT is independent of any programming language.

ADT in Object-oriented programming languages (Java)

- Object-oriented programming languages (such as Java) make it easy for programmers to use ADTs: each ADT corresponds to a **class** (or **Java interface**) and the operations on the ADT are the class/interface's **public methods**.
- The user, or client, of the ADT only needs to know about the method **interfaces** (the names of the methods, the types of the parameters, what the methods do, and what values they return (if any) BUT not the actual implementation (how the methods are implemented, the private data members, private methods, etc.).

Cont’d

- Type of data structures in Java
  - Array
  - Class
  - String
- Data structures that have to be created:
  - Lists
  - Stacks
  - Queues
  - Trees
  - Graph

Type of Data Structures

- **1. Array**
  - Is the simplest data structure.
  - Can be 1 dim. (linear) and multidim.

Index	Student
0	Zaki
1	Dollah
2	Mamat
3	Zack
4	Ali

Type of Data Structures

- **2. Linked-list**
  - A dynamic data structure where size can be changed.
  - Using a pointer

Student	Pointer
Alfie	3
Daud	1
Zack	2
Dollah	2
Ali	3

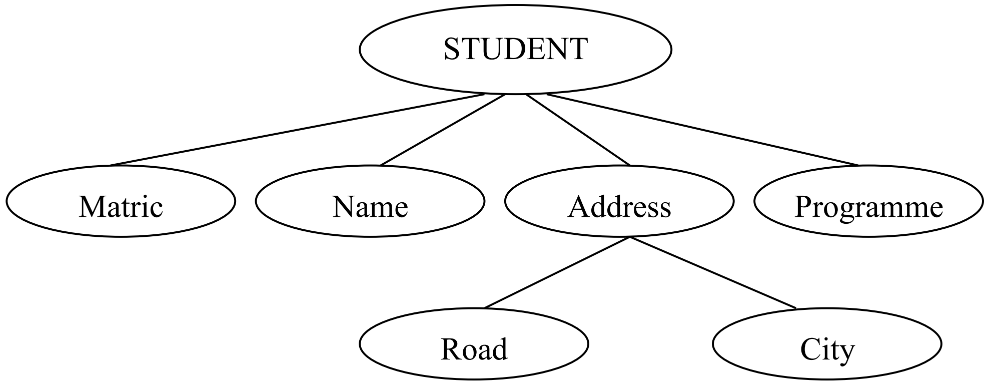
PA	Pointer
Dr Azim	1
Dr Ramlan	2
Dr Ali	3

Type of Data Structures

- **3. Stack**
  - Addition and deletion always occur on the top of the stack. Know as - LIFO (Last In First Out).
  - Example: a stack of plates
- **4. Queue**
  - Adding at the back and deleting in front. Known as – FIFO (First In First Out).
  - Example: queuing for the bus, taxi, etc.
- **5. Recursive**
  - Function calling itself to perform repetition.
  - Example: Factorial, Fibonacci, etc.

Type of Data Structures

- **6. Tree**
  - Data in a hierarchical relationship.



- **7. Graph**
  - Like a tree but the relationship can be in any direction.
  - Example: Distance between two or more towns.

Operations in a Data Structure

- Operations
  - Traversing – to access every record at least once.
  - Searching – to find the location of the record using key
  - Insertion – adding a new record.
  - Deletion – remove a record from the structure.
- Combination
  - Updating
  - Sorting
  - Merging

Problem – Definition

- A Problem is a task to be performed.
- Some problems can be viewed as functions in the mathematical sense.
  - A function is a matching between inputs (the domain) and outputs (the range).
  - A particular input must always result in the same output every time the function is computed.

Algorithms and Programs

- An algorithm is a method to solve a problem.
  - A recipe.
- An algorithm takes the input to a problem (function) and transforms it to the output.
  - A mapping of input to output.
- A problem can have many algorithms.
- But a given algorithm solves only one problem.
- A program is an instance of an algorithm in a specific programming language.

Algorithms and Programs

- At the heart of computer program design there are two (sometimes conflicting) goals.
  - Goal 1. To design an algorithm that is easy to understand, code, debug.
  - Goal 2. To design an algorithm that makes efficient use of the computer's resources.
- Goal 1 is the concern of Software Engineering.
- Goal 2 is the concern of data structures and algorithm analysis.