DATA STRUCTURE AND ALGORITHM

Dr. Khine Thin Zar
Professor
Computer Engineering and Information Technology Dept.
Yangon Technological University

Basic course information

- Course Title
  - Data Structure and Algorithm
- Textbook and Reference materials
  - Clifford A. Shaffer “Data Structures & Algorithm Analysis in C++”, , 3rd Edition, (2011)
  - Michael T. Goodrich, Roberto Tamassia, David M. Mount "Data Structures and Algorithms in C++", 2nd Edition, (2011)
  - Mark Allen Weiss "Data Structures and Algorithm Analysis in C++", 4th Edition, (2014)
- Course Duration
  - 14 Weeks

Course Outcomes

- To be able to understand the fundamental data structures and algorithms for representing data
- To be able to design appropriate data structures based on the efficient algorithms for solving computing problems
- To be able to implement common algorithms for working with advanced data structures and recognize which data structure is the best to use to solve a particular problem.
- To be able to design an algorithm that makes efficient use of the computer's resources
- To be able to evaluate the space and time efficiency of data organization and algorithms
- To be able to develop the abstract data types for various data structures in C++ programming

Course Schedule

No	Topic	Week
1	Overview of Data Structures and Algorithms	Week 1
2	Mathematical Preliminaries	Week 2
3	Algorithm Analysis	Week 3
4	Lists, Stacks and Queues	Week 4
5	Binary Trees	Week 5
6	Non-Binary Trees	Week 6
7	Internal Sorting	Week 7
8	File processing and External Sorting	Week 8
9	Searching	Week 9
10	Indexing	Week 10
11	Graphs	Week 11
12	Advanced Tree Structures	Week 12
13	Algorithm Design Techniques	Week 13
14	Applications	Week 14

Grading Policy

- Exam → 80%
- Tutorial / Assignment → 20%

Lecture 1

Overview of Data Structures and Algorithms

Outlines of Class (Lecture 1)

- Introduction
- Data Structure
- Algorithm
- Abstract Data Type (ADT)

Goals

- To learn the commonly used data structures
- To learn the concept that every data structure has costs and benefits
- To learn how to measure the effectiveness of a data structure or algorithm

Introduction

- The main goals of the computer program design:
  - ✓ To design an algorithm that is easy to understand, code, and debug.
  - ✓ To design an algorithm that makes efficient use of the computer's resources.
- Resource Constraints
  - ✓ Total space available to store data
  - ✓ The time allowed to perform each subtask

Data Structure

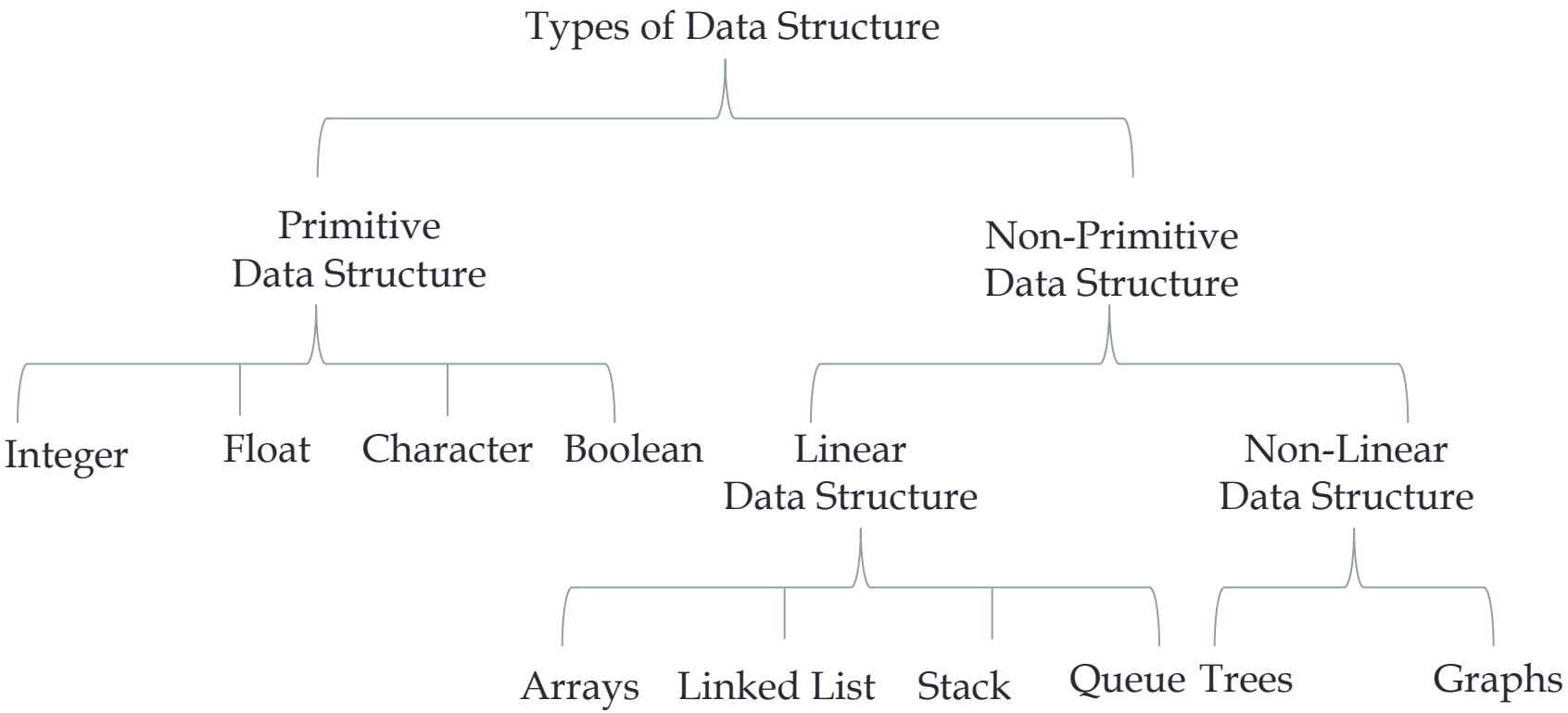
- A data structure is a particular way of organizing data in the memory and it is a systematic way to organize data in order to use it efficiently.
- The main aim of a data structure is to group similar/related items together and bundle them, hence making it easier and efficient to execute the program.
- Example: Library
  - ✓ is composed of elements (books)
  - ✓ Accessing a desired book needs to know the arrangement of the books
  - ✓ Access books only through the librarian

Selecting a Data Structure

Select a data structure as follows:

- Analyze the problem to determine the basic operations.
- Quantify the resource constraints for each operation.
- Select the data structure that best meets these requirements.

Types of Data Structure



Linear and Non-linear Data Structure

- Linear Data Structure:
  - A data structure is said to be linear if its elements form any sequence.
  - ✓ e.g. , Linked lists, Priority queues, Arrays, Queues, Stacks
- Non-linear Data Structure:
  - This structure is mainly used to represent data containing a hierarchical relationship between elements.
  - ✓ e.g. graphs, trees, hash tables
- The main difference between linear and non linear data structures is that linear data structures arrange data in a sequential manner while nonlinear data structures arrange data in a hierarchical manner, creating a relationship among the data elements.

Linear and Non-linear (cont.)

- Linear:
  - ✓ Linked list: Variable size
  - ✓ Priority queue: Delete the highest priority, Add anywhere
  - ✓ Array: Fixed-size
  - ✓ Queue: First in First out
  - ✓ Stack: Last in First out

- Non-linear:
  - ✓ Graph: A general branching structure, with less strict connection
  - ✓ Tree: Data is organized in branches like a tree
  - ✓ Hash tables: Use a hash function to insert and search

Differences between Linear and Non-linear Data Structures

Linear Data Structure	Non-linear Data Structure
Elements are linked one after another.	Elements are hierarchically related.
The data elements can be traversed directly.	The data elements can be traversed hierarchically.
Waste the memory.	Utilization of memory.
Easy to implement.	Implementation is complex.
Examples: Array, Queue, Stack, Linked List	Examples: Trees, graphs

Characteristics of a Data Structure

- **Correctness** − Data structure implementation should implement its interface correctly.
- **Time Complexity** − Running time or the execution time of operations of data structure must be as small as possible.
- **Space Complexity** − Memory usage of a data structure operation should be as little as possible.

Why Data Structure Needed?

- Goal: to organize data efficiently
- There can be three common problems that applications face as applications are getting complex and data growth.
  - ✓ **Searching Data** − Consider as a 2 million items of a store. If the application is to search an item in a store, it must search an item in 2 million times slowing down the search. As data grows, searching will become slower.
  - ✓ **Processor Speed** − Although the processor speed is very high, falls limited if the data grows to billion records.
  - ✓ **Multiple Requests** − As many users can search data simultaneously on a web server, even the fast server fails while searching the data.
- To solve the above-mentioned problems, data can be organized in a data structure in such a way that all items can be accessed and searched easily.

Data Structure Operations

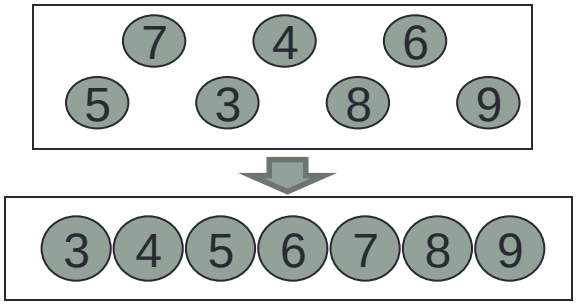
- **Traversing:** accessing each record/node exactly once so that certain items in the record may be processed. (This accessing and processing is sometimes called "**visiting" the record.)**

- **Searching:** Finding the location of the desired node with a given key value, or finding the locations of all such nodes which satisfy one or more conditions.

- **Inserting:** Adding a new node/record to the structure.

- **Deleting:** Removing a node/record from the structure.

What is an Algorithm?



- An algorithm is a method or a process followed to solve a problem, written in order, to accomplish a certain predefined task. Algorithm is not the complete code or program, it is just the core logic(solution) of a problem, which can be expressed either as a **pseudocode** or using a **flowchart**.
- Every Algorithm must satisfy the following properties:
  - ✓ **Input**- There should be 1 or more inputs supplied externally to the algorithm.
  - ✓ **Output**- There should be at least 1 output obtained.
  - ✓ **Definiteness**- Every step of the algorithm should be clear and well defined.
  - ✓ **Finiteness**- The algorithm should have finite number of steps.
  - ✓ **Correctness**- Every step of the algorithm must generate a correct output.

What is an Algorithm? (cont.)

- An algorithm is said to be efficient and fast, if it takes less time to execute and consumes less memory space.
- The performance of an algorithm is measured on the basis of following properties :
  - ✓ Space Complexity
  - ✓ Time Complexity

Pseudocode and Flowchart

Pseudocode

- Before writing an algorithm in a programming language, it is described in a higher level language called pseudo-language.
- It is a simpler version of a programming code in plain English which uses short phrases to write code for a program before it is implemented in a specific programming language.
- It is composed by a set of instructions to describe how the algorithm works.

Flowchart

- A flowchart is a strategic diagram of an algorithm or a stepwise process, showing the steps as simple geometric shapes, by connecting these with arrows.
- Flowcharts are used in designing or documenting a process or program how a process is performed from start to finish.
- The flowchart represents the body of the algorithm in a sequential order and the flow of algorithms as well as the links.

Examples of Pseudocode

```
Start Program
    If student's mark is greater than or equal to 50
        Print “pass"
    else
        Print "fail“
End Program
```

Student's Grade Result

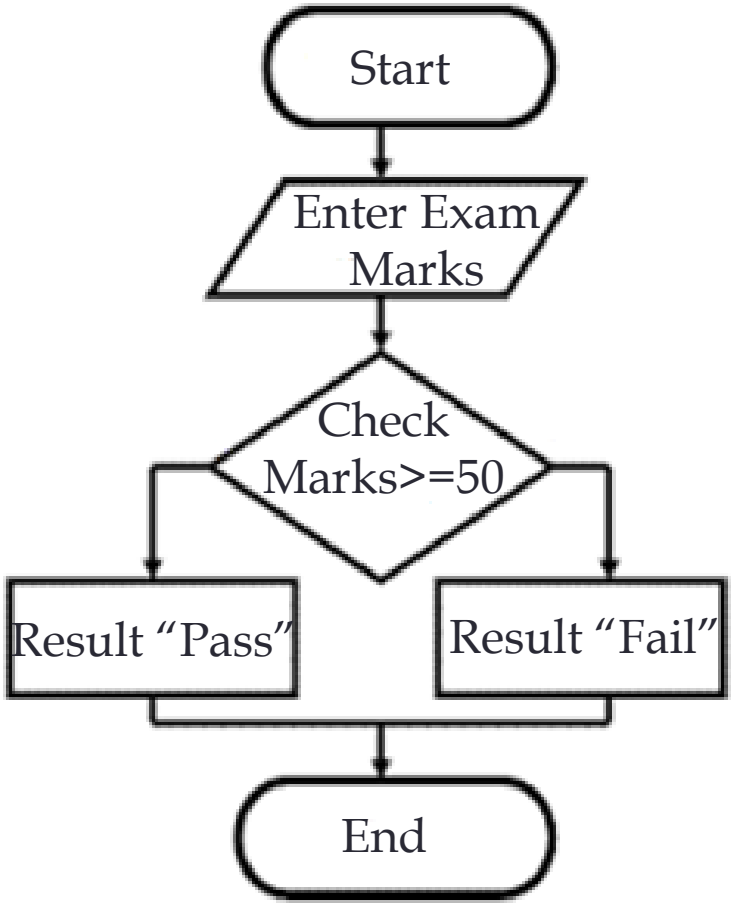
```
If you are happy Then
        smile
Else If you are angry
        glare
Else
        keep face plain
Endif
```

Human Emotion

```
Start
        Enter three numbers, A, B, C
        Add the numbers together
        Print Sum
End
```

Addition of Three Numbers

Example of Flowchart



Space Complexity

- It is the amount of memory space required by the algorithm, during the course of its execution. Space complexity must be taken seriously for multi-user systems and in situations where limited memory is available.
  - ✓ **Instruction Space:** It is the space required to store the executable version of the program. This space is fixed, but varies depending upon the number of lines of code in the program.
  - ✓ **Data Space:** It is the space required to store all the constants and variables(including temporary variables) value.
  - ✓ **Environment Space:** It is the space required to store the environment information needed to resume the suspended function.

Space Complexity (cont.)

- The memory space we consider is the space of primary memory.
- Whenever a program executes, the program is to be loaded on the primary memory. Then during execution, how much space will be occupied it will be decided by the space complexity.
- Obviously, in case of space complexity, the number of instruction getting executed, number of instruction residing in the algorithm that will decide the space complexity.

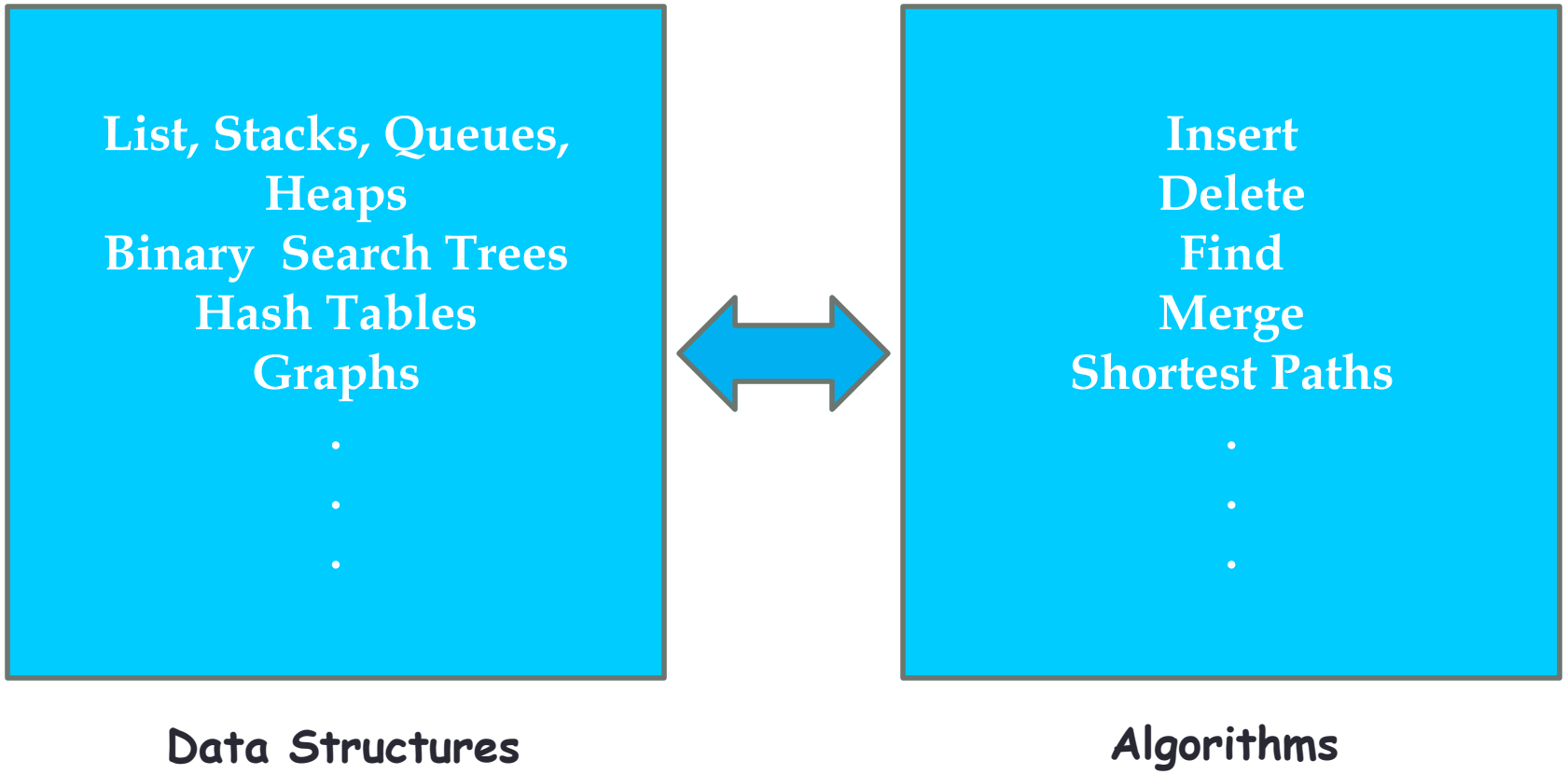
Time Complexity

- Time Complexity is a way to represent the amount of time required by the program to run till its completion.
- It's generally a good practice to try to keep the time required minimum, so that the algorithm completes it's execution in the minimum time possible.
- The time complexity is defined as the process of determining a formula for total time required towards the execution of that algorithm.
- This calculation will be independent of implementation details and programming language.
- So during the execution of an algorithm, the total time required that will be decided in the time complexity.
- While considering the time complexity of an algorithm, it will be irrespective of the programming language selected for the implementation of the algorithm.

Algorithms in our Daily Life

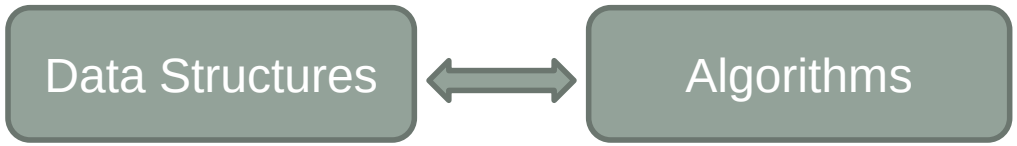
- I need to build a tower
  - ✓ The algorithm to follow here is the list of instructions that tell you how to make the tower.
- I need to make a pizza
  - ✓ Here the algorithm is how to make a pizza. You can find the algorithm to solve this problem in a cookbook.
- I can't find the university
  - ✓ The algorithm you need is a set of directions to get to the university. You can have different algorithms because there might be different ways to the university.

Relation between Data Structure and Algorithm



What is a Program?

- A program is the expression of an algorithm in a programming language.
- A set of instructions which the computer will follow to solve a problem.



Difference between Algorithm, Pseudocode and Program

- **Algorithm :** Systematic logical approach which is a well-defined, step-by-step procedure that allows a computer to solve a problem.

- **Pseudocode :** It is a simpler version of a programming code in plain English which uses short phrases to write code for a program before it is implemented in a specific programming language.

- **Program :** It is exact code written for problem following all the rules of the programming language.

Abstract Data Type (ADT) vs Data Structure

- Abstract Data Type (ADT)
  - ✓ The realization of a data type as a software component
  - ✓ The interface of the ADT is defined in terms of a type and a set of operations on that type.
  - ✓ An ADT is a collection of data and associated operations for manipulating that data and it is a logical description of how we view the data.
- Data Structure
  - ✓ Physical implementation of an ADT (Concrete implementation)
  - ✓ Data structures are used to implement the Abstract Data Type provided in a primitive/ built-in language or user-defined language.
  - ✓ Data structures are part of an ADT's implementation.

Abstract Data Type (ADT) vs Data structure (cont.)

- ADTs provide abstraction, encapsulation and information hiding.
- Abstraction focuses only on relevant data of an object and emphasizes on the details of how to do a collection of data by defining their data and operations.
- The principle of hiding the used data structure, restricting the use and operation that can be performed on that data structure is known as encapsulation.
- Designing a method without any need the detail knowledge of the code is called information hiding.

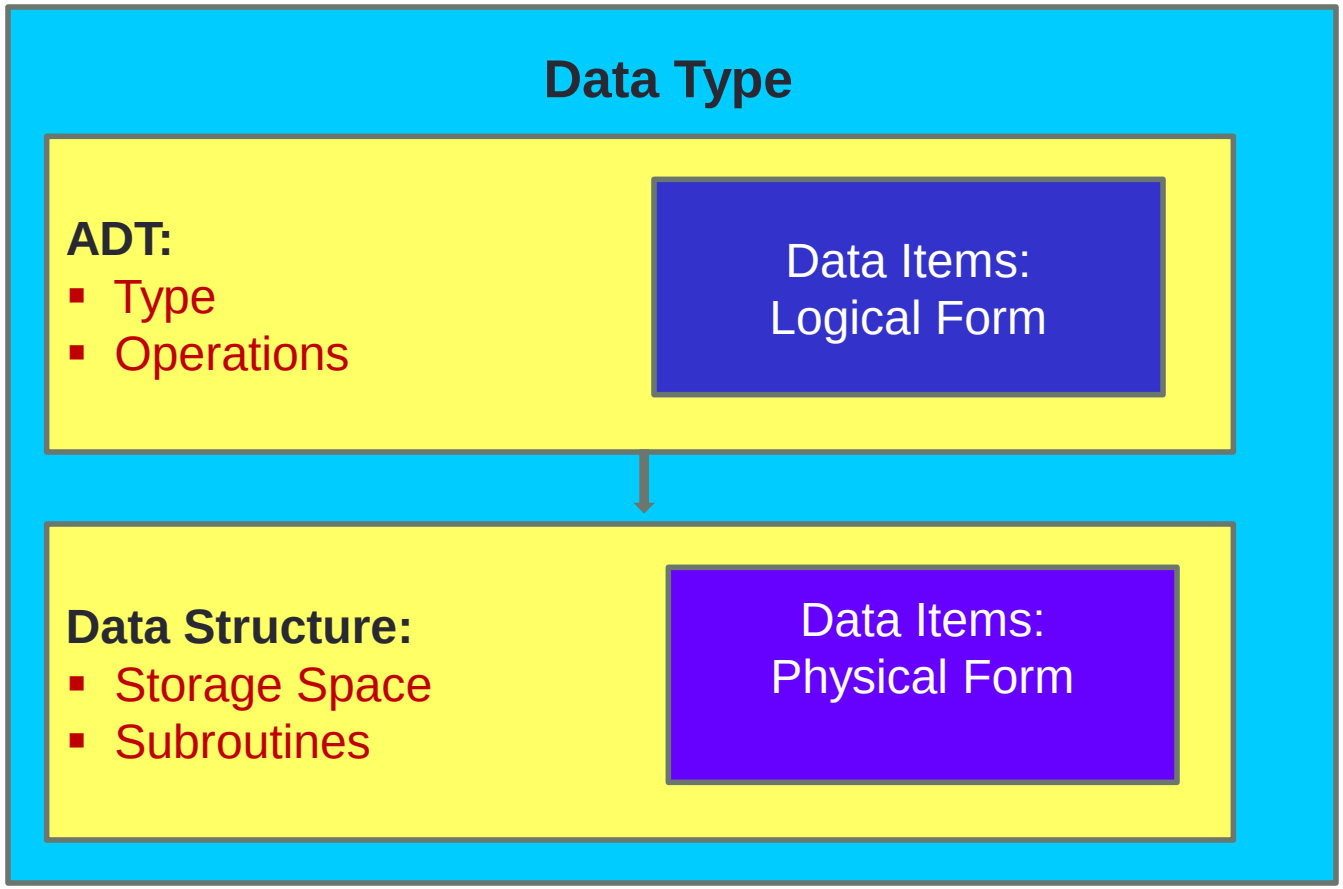
The Core Operations of ADT

- Every collection ADT should provide a way to:
  - ✓ insert a item
  - ✓ delete an item
  - ✓ search, retrieve, or access an item

Logical vs Physical Form

- Data items have both a logical and a physical form.
- Logical Form: definition of the data item within an ADT.
  - ✓ Eg: Integers in mathematical sense: +, -
- Physical Form: implementation of the data item within a data structure.
  - ✓ Eg: 16/32 bit integers, overflow.

Logical vs Physical Form (cont.)



Algorithm/ Data Structure Philosophy

- Each data structure requires:
  - ✓ space to store each item, including overhead
  - ✓ time to perform basic operations
  - ✓ programming effort
- Algorithms are closely related:
  - ✓ poor data structure choice can make higher complexity algorithm
  - ✓ good data structure choice can make the algorithm trivial

Why need Algorithm analysis?

- The reason for analyzing an algorithm is to discover its characteristics in order to evaluate its suitability for various applications or compare it with other algorithms for the same application.
- The more complicated the algorithm, the more difficult the analysis.
- The characteristics of interest are most often the primary resources of time and space.
- If the program is run on a large data set, then the running time becomes an issue.

Assignments

1. Define data structure. Name and explain different types if data structure. Explain operations that can be performed on theses data structures.

2. What is an algorithm? What is time and space analysis of an algorithm?

3. Write an algorithm to add three numbers entered by user.

4. Write an algorithm to find the larger between two different numbers entered by user.

Sample Answers for Algorithms

Write an algorithm to add two numbers entered by user.

Step 1: Start

Step 2: Declare variables num1, num2,num3 and sum.

Step 3: Read values num1, num2 and num3.

Step 4: Add num1, num2 and num3 and assign the result to sum.

sum←num1+num2+num3

Step 5: Display sum

Step 6: Stop

Write an algorithm to find the larger between two different numbers entered by user.

Step 1: Start

Step 2: Declare variables x and y.

Step 3: Read variables x and y.

Step 4: If x>y

Display x is larger than y.

Else

Display y is larger than x.

Step 5: Stop

Next Week Lecture (Week 2)

- ✓ Lecture 2: Mathematical Preliminaries

Thank you!