جامــــعة المــــستــقـبـل
AL MUSTAQBAL UNIVERSITY

كلية التقنيات الطبية والصحية
قــســـــــــم الانــظـــمـــة الـــطـبـيـة الــذكــــيــة

Lecture: (1)

Introduction to Data Structures

Subject: Data Structure
Class: Second
Lecturer: Asst. Prof. Mehdi Ebady Manaa

Introduction to Data Structures

Data Structure is an arrangement of data in a computer's memory (or sometimes on a disk). Data structures include arrays, linked lists, stacks, binary trees, and hash tables, among others. Algorithms manipulate the data in these structures in various ways, such as searching for a particular data item and sorting the data.

What is Data Structure ?

- A scheme for organizing related pieces of information
- A way in which sets of data are organized in a particular system
- An organized aggregate of data items
- A computer interpretable format used for storing, accessing, transferring and archiving data
- The way data is organized to ensure efficient processing: this may be in lists, arrays, stacks, queues or trees

Overview of Data Structures

Another way to look at data structures is to focus on their strengths and weaknesses. Table 1.1 shows the advantages and disadvantages of the various data structures.

TABLE 1.1 Characteristics of Data Structures

Data Structure	Advantages	Disadvantages
Array	Quick insertion, very fast access if index known.	Slow search, slow deletion, fixed size.

Ordered array	Quicker search than unsorted array	Slow insertion and deletion, fixed size.
Stack	Provides last-in, first-out access.	Slow access to other items.
Queue	Provides first-in, first-out access.	Slow access to other items.
Linked list	Quick insertion, quick deletion.	Slow search.
Binary tree	Quick search, insertion, deletion (if tree remains balanced).	Deletion algorithm is complex.
Red-black tree	Quick search, insertion, deletion. Tree always balanced.	Complex.
2-3-4 tree	Quick search, insertion, deletion. Tree always balanced. Similar trees good for disk storage.	Complex.
Hash	Table Very fast access if key known. Fast insertion	Slow deletion, access slow if key not known, inefficient memory usage.
Heap	Fast insertion, deletion, access to largest item.	Slow access to other items.
Graph	Models real-world situations.	Some algorithms are slow and complex.

The data structures shown in Table 1.1, except the arrays, can be thought of as Abstract Data Types, or ADTs.

A data type consists of

- a domain (= a set of values)
- A set of operations.

Example 1: *Boolean* or *logical* data type provided by most programming languages.

- Two values: true, false.
- Many operations, including: AND, OR, NOT, etc.

Abstract Data Type (**ADT**): The basic idea behind an abstract data type is the **separation of the use of the data type from its implementation** (i.e., what an abstract data type does can be specified separately from how it is done through its implementation)

The advantages of using the **ADT** approach are:

1. The implementation of **ADT** can change without affecting those method that use the **ADT**
2. The complexity of the implementation are hidden

For example, an **abstract stack data structure** could be defined by two operations: **push**, that inserts some data item into the structure, **and pop**, that extracts an item from it.

Advantages of data structures

The major advantages of data structures are:
• It gives different level of organizing data.
• It tells how data can be stored and accessed in its elementary level

Operation on Data Structures

The operations that can be performed on data structures are:
Creation: This operation creates a data structure. The declaration statement causes space to be created for data upon entering at execution time.

Destroy: This operation destroys the data structure and aids in the efficient use of memory.

Selection: This operation is used to access data within a data structure. The form of selection depends on the type of data structure being accessed.

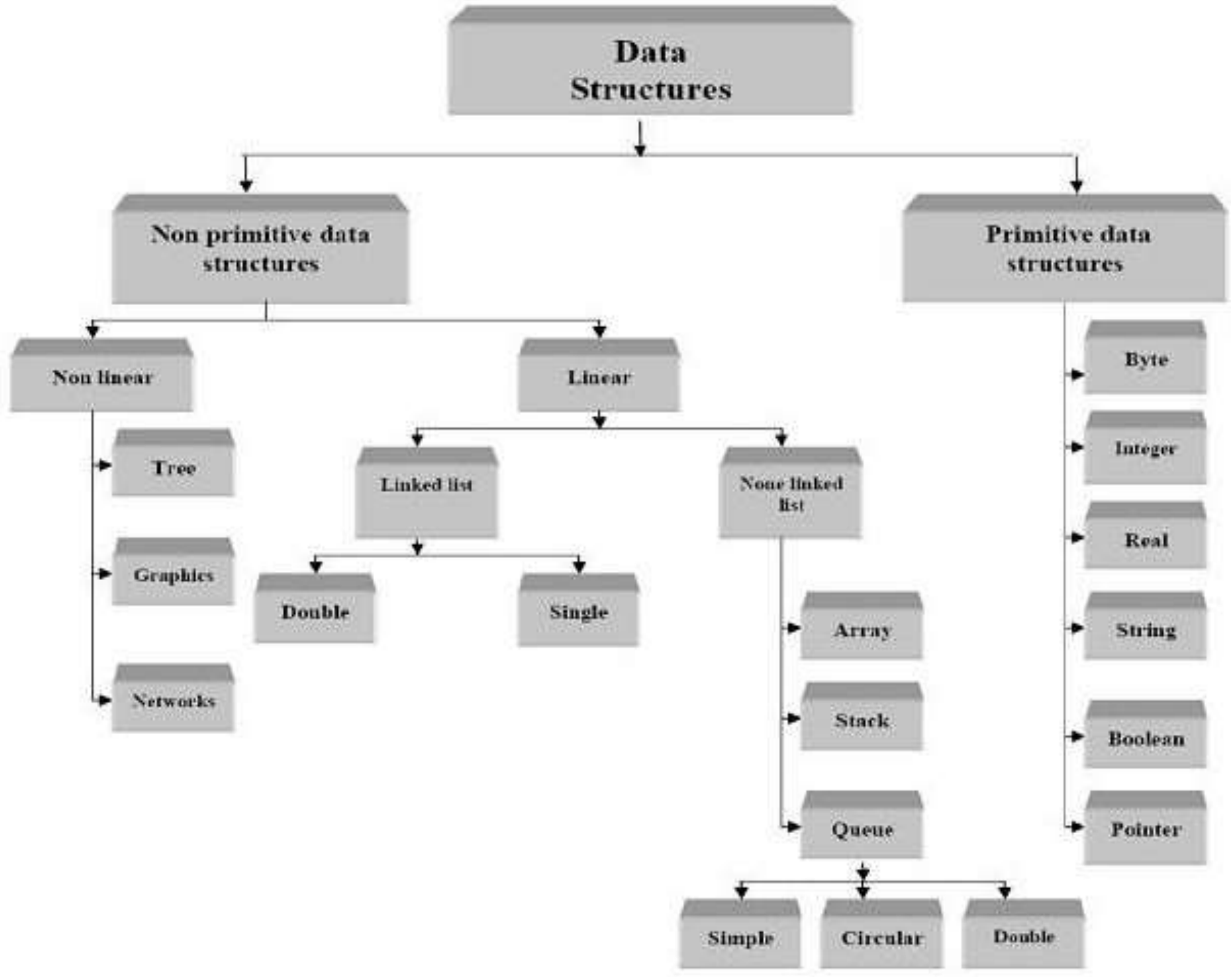
Update: This operation changes or modifies the data in the data structure and it is an important property in selection operation.

Types of Data Structures

Linear Data Structure: Stacks, Queues, Linked Lists, etc.

Non-linear Data Structure: Trees, Graphs, etc .

As illustrated in block diagram below.



Block diagram of Data Structures

Difference between Linear and Nonlinear Data Structures

Main difference between linear and nonlinear data structures lie in the way they organize data elements. In linear data structures, data elements are organized sequentially and therefore they are easy to implement in the computer's memory. In nonlinear data structures, a data element can be attached to several other data elements to represent specific relationships that exist among them. Non-Linear data structure is that if one element can be connected to more than two adjacent element then it is known as non-linear data structure.

Certainly! Here are three questions based on the lecture "Introduction to Data Structures":

Questions

Question 1:

What is the fundamental difference between linear and non-linear data structures? Provide examples of each type and explain how they organize data elements.*

Question 2

Describe the concept of an Abstract Data Type (ADT) and its advantages. How does the ADT approach separate the use of a data type from its implementation? Provide an example of an ADT and explain its operations.*

Question 3

In the context of data structures, explain the significance of the operations: Creation, Destruction, Selection, and Update. Provide examples of each operation and discuss their roles in manipulating data structures for efficient use.