Data Structures

Lecture Two

Second Stage

First Course - 2024-2025

ALI MUWAFAQ SHABAN

Master of Computer Engineering

Data Structures

Introduction to Data Structures

A data structure is a way of storing data in a computer so that it can be used efficiently and it will allow the most efficient algorithm to be used. The choice of the data structure begins from the choice of an abstract data type (ADT). A well-designed data structure allows a variety of critical operations to be performed, using as few resources, both execution time and memory space, as possible. Data structure introduction refers to a scheme for organizing data, or in other words it is an arrangement of data in computer's memory in such a way that it could make the data quickly available to the processor for required calculations. A data structure should be seen as a logical concept that must address two fundamental concerns:

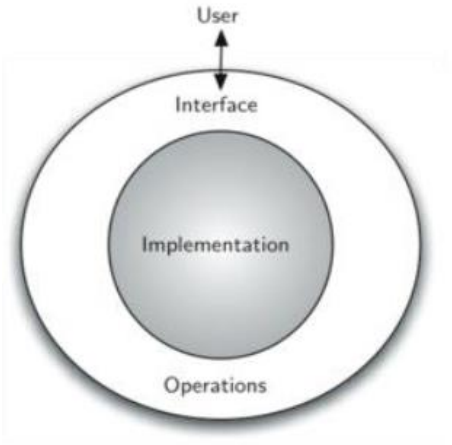
1. First, how the data will be stored, and

2. Second, what operations will be performed on it.

As data structure is a scheme for data organization so the functional definition of a data structure should be independent of its implementation. The functional definition of a data structure is known as ADT (Abstract Data Type) which is independent of implementation. The way in which the data is organized affects the performance of a program for different tasks. Computer programmers decide which data structures to use based on the nature of the data and the processes that need to be performed on that data. Some of the more commonly used data structures include lists, arrays, stacks, queues, heaps, trees, and graphs.

Abstract Data Type

An abstract data type, sometimes abbreviated ADT, is a logical description of how the data and the operations view are allowed without regard to how they will be implemented. This means the concern is only with what data is representing and not with how it will eventually be constructed. By providing this level of abstraction, we are creating an encapsulation around the data. The idea is that by encapsulating the details of the implementation, we are hiding them from the user's view. This is called information hiding. The implementation of an abstract data type, often referred to as a data structure, will require that we provide a physical view of the data using some collection of programming constructs and primitive data types. Examples of ADTs include list, stack, queue, set, tree, graph, etc.



Abstract Data Type (ADT): An ADT is a set of elements with a collection of well defined operations.

- The operations can take as operands not only instances of the ADT but other types of operands or instances of other ADTs.
- Similarly results need not be instances of the ADT
- At least one operand or the result is of the ADT type in question.

Data Structures: An implementation of an ADT is a translation into statements of a programming language,

– the declarations that define a variable to be of that ADT type

– the operations defined on the ADT (using procedures of the programming language)

OR

Data structure: A collection of data elements whose organization is characterized by accessing (Mechanism for organizing information), operation that are used to store and retrieve the individual of elements

Program: An implementation of an algorithm in some programming language

Data: Set of possible values for variables, characters, numbers, pictures, videos or mixed.

The Principles Data Structure

For each set of data there are more than one method to organization depend on:

1. Data size
2. Required storage space
3. The time required to retrieve the data
4. Programing language manner

Data Types

Data type is a way to classify various types of data such as integer, string, etc. which determines the values that can be used with the corresponding type of data, the type of operations that can be performed on the corresponding type of data. There are two data types –

- Built-in Data Type
- Derived Data Type

Built-in Data Type: Those data types for which a language has built-in support are known as Built-in Data types. For example, most of the languages provide the following built-in data types.

- Integers
- Boolean (true, false)
- Floating (Decimal numbers)
- Character and Strings

Derived Data Type: Those data types which are implementation independent as they can be implemented in one or the other way are known as derived data types. These data types are normally built by the combination of primary or built-in data types and associated operations on them. For example –

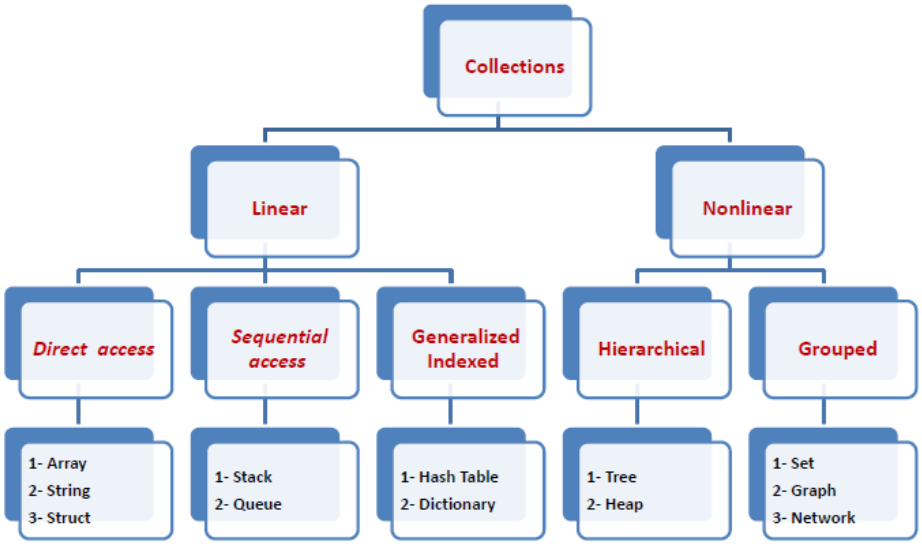
- List
- Array
- Stack
- Queue

Basic Operations of Data

The data in the data structures are processed by certain operations. The particular data structure chosen largely depends on the frequency of the operation that needs to be performed on the data structure.

- Insertion
- Deletion
- Sorting
- Merging
- Traversing
- Searching

Types of Data Structures:



Linear Data Structure:

Linear data structures can be constructed as a continuous arrangement of data elements in the memory. It can be constructed by using array data type. In the linear Data Structures the relationship of adjacency is maintained between the data elements. For example Stack, Queue, Tables, List, and Linked Lists.

Operations applied on linear data structure: The following list of operations applied on linear data structures

1. Add an element
2. Delete an element
3. Traverse
4. Sort the list of elements
5. Search for a data element

Non-linear Data Structure:

Non-linear data structure can be constructed as a collection of randomly distributed set of data item joined together by using a special pointer (tag). In non-linear Data structure the relationship of adjacency is not maintained between the data items. For example Tree, Decision tree, Graph and Forest.

Operations applied on non-linear data structures: The following list of operations applied on non-linear data structures.

1. Add elements
2. Delete elements

3. Display the elements

4. Sort the list of elements

5. Search for a data element

Operations on Data Structures

Design of efficient data structure must take operations to be performed on the data structures into account. The most commonly used operations on data structure are broadly categorized into following types

1) Create:- The create operation results in reserving memory for program elements. This can be done by declaration statement. Creation of data structure may take place either during compile-time or run-time. malloc() function of C language is used for creation.
2) Destroy:- Destroy operation destroys memory space allocated for specified data structure. free() function of C language is used to destroy data structure.
3) Selection:- Selection operation deals with accessing a particular data within a data structure.
4) Updation:- It updates or modifies the data in the data structure.
5) Searching:- It finds the presence of desired data item in the list of data items, it may also find the locations of all elements that satisfy certain conditions.
6) Sorting:- Sorting is a process of arranging all data items in a data structure in a particular order, say for example, either in ascending order or in descending order.
7) Merging:- Merging is a process of combining the data items of two different sorted list into a single sorted list.
8) Splitting:- Splitting is a process of partitioning single list to multiple list.
9) Traversal:- Traversal is a process of visiting each and every node of a list in systematic manner.

References:

- Frank Carrano, D.J. Henry: Data Abstraction and Solving with C++, 2012, 6th edition, Pearson Education, Inc.
- Mark Allen Weiss: Data Structures and Algorithm Analysis in C++, 2014, 4th edition, Pearson Education, Inc.