Data Structures & Algorithms (CACS201)

III semester, BCA

Compiled by :

Ankit Bhattarai

Email: ankit.bca@kathford.edu.np

What will be discussed today ?

- Simple Instructions about Online Classes
- Syllabus
- Resources
- Chapter 1: Introduction to data structure

Instructions

- Try to **write or point down the question** so that you can ask them after the portion is completed.

- Both of us are new to the platform, we need to have good **coordination & communication** in order to make the platform effective. With time this shall will obviously be efficient.

Syllabus

Unit	Contents	Hours	Remarks
1.	Introduction to Data Structure	2	
2.	The Stack	3	
3.	Queue	3	
4.	List	2	
5.	Linked Lists	5	
6.	Recursion	4	
7.	Trees	5	
8.	Sorting	5	
9.	Searching	5	
10.	Graphs	5	
11.	Algorithms	5	

Credit : 3

Syllabus

Laboratory Works:

There shall be 10 lab exercises based on C or Java

1. Implementations of different operations related to Stack
2. Implementations of different operations related to linear and circular queues
3. Solutions of TOH and Fibonacci Series using Recursion
4. Implementations of different operations related to linked list: singly and doubly linked
5. Implementation of trees: AVL trees, Balancing of AVL
6. Implementation of Merge sort
7. Implementation of different searching technique: sequential, tree and binary
8. Implementations of Graphs: Graph Traversals
9. Implementation of Hashing
10. Implementation of Heap.

Resources

Resources

1. All chapters pdf form.
2. Question Collection (70-80 Questions)

Books

1. **Y. Langsam, M.J. Augenstein and A.M. Tenenbaum, "Data Structures using c and C++"**
2. G.W. Rowe, " Introduction to Data Structure and Algorithm with C and C++"
3. Robert Lafore, Data structures and Algorithms in Java (2nd Edition), Sams Publishing

Chapter 1

(2 hrs.)

Introduction to Data Structure

- Definition
- Abstract Data Type (ADT)
- Importance of Data Structure

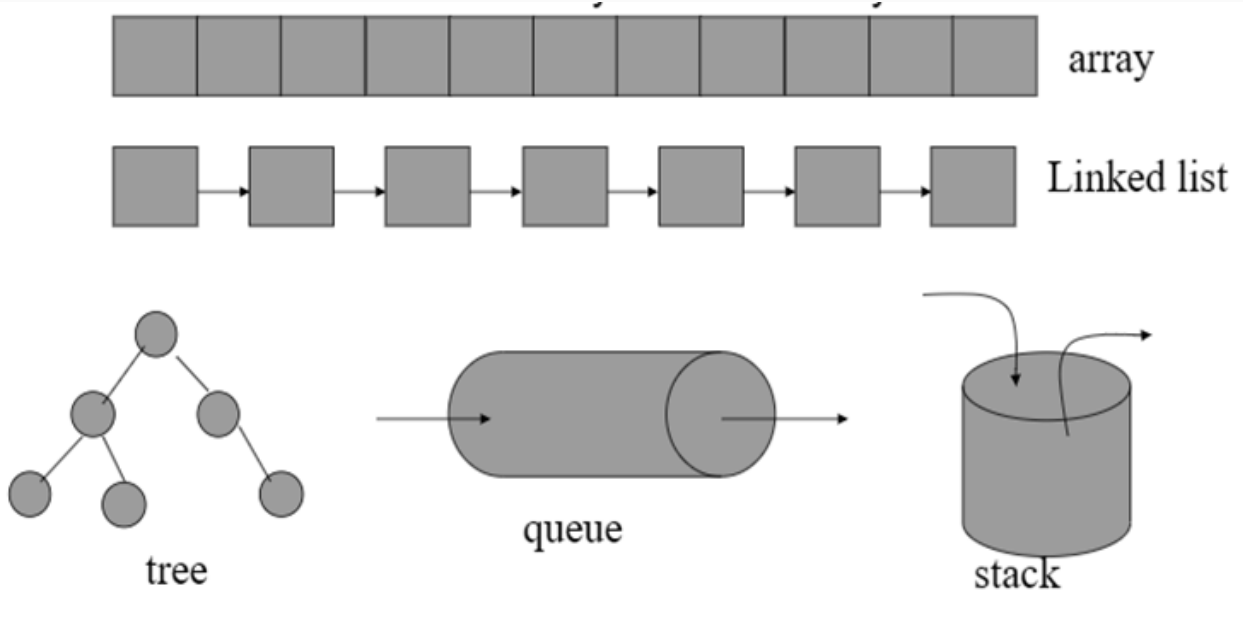
Data Structure

Definition:

- **Data structure** refers to the organization of data in computer memory or the way in which data efficiently stored, processed and retrieved is called data structure.
- **Data structure** is representation of logical relationship existing between individual elements of data.
- A **data structure** is a systematic way of storing and organizing data in a computer's memory such that it can be handled efficiently and effectively.
- Example: you can use graphs to represent routes between places; a stack can be used to represent history on web browser, etc.

Data Structure

- Different data structures that we are going to cover are demonstrated in the figure below:



Importance of Data Structure

- A data structure helps us to understand the relationship of one element with the other and organize it within a memory.

- Study of data structures includes:
  - **Logical description** of the data structures (Organization of data)
  - **Implementation** of data structure
  - **Quantitative analysis** of the data structure which includes determining the amount of memory needed to store the data structure and the time required to process it.

Importance of Data Structure

- Data structures are the building blocks of a program. And hence the selection of a particular data structure focus on

  - ✓ The data structures must be rich enough in structure to reflect the **relationship existing between the data.**

  - ✓ And the structure should be simple so that we can **process data effectively** whenever required.
- Data structure affects the design of both structural & functional aspects of a program.

Algorithm + Data structure = Program

So, to develop a program of an algorithm we should select an appropriate data structure for that algorithm.

Importance of Data Structure

- **Advantages of data structure**
  - ✓ It helps to store information securely and efficiently in the computer system.
  - ✓ It helps to use and process data on an application
  - ✓ It makes all processes involving itself very quick and efficient.
- **Disadvantages of data structure**
  - ✓ Advanced IT/Computer expertise is required in order to manipulate data structure.
  - ✓ The choice of data structure for particular problem must be spot on otherwise the data storing, processing and retrieving methods will not be efficient.

Types/Classification of Data Structure

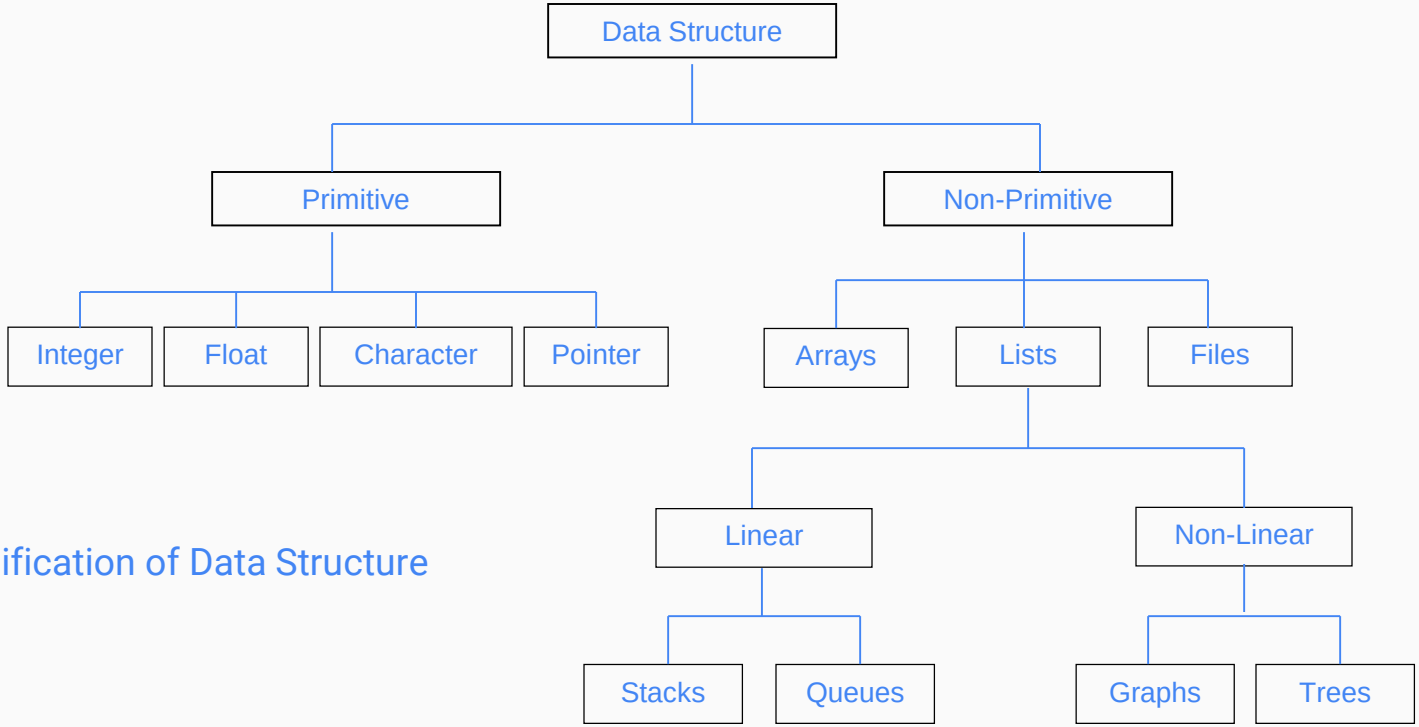


Fig: Classification of Data Structure

Types/Classification of Data Structure

1. **Primitive and Non primitive Data Structures**

i. **Primitive data structure:** They are the basic data structure and are directly operated by the machine instruction which is in primitive level.

Example: integers, floating point numbers, characters, string constants, etc.

ii. **Non primitive data structure**

They are more complicated data structure emphasizing on the structuring of group of homogeneous & heterogeneous data items.

They are generally derived from the primitive data structures.

Example: Arrays, Lists, Linked lists, trees , graphs etc.

Types/Classification of Data Structure

2. According to Nature of size

i. Static data structure

ii. Dynamic Data Structure

i. **Static data structure**

A data structure is said to be static if we can store data up to a fix number. E.g. array

ii. **Dynamic data structure**

In this type of data structure, it allows the programmers to change its size during program execution to add or delete data. For example: linked list, tree, graph, etc.

Types/Classification of Data Structure

3. **According to its Occurrence**
    i. Linear Data Structure
    ii. Non Linear Data Structure

i. **Linear data structure:** In this data structure, the data is stored in consecutive memory location or in sequential form i.e. every element in the structure has a unique predecessor and unique successor. E.g. array, linked list, queue, etc.

ii. **Non Linear data structure:** The data is stored in non consecutive memory location or data. There is no specific predecessor or successor. E.g. tree, graphs, etc.

Types/Classification of Data Structure

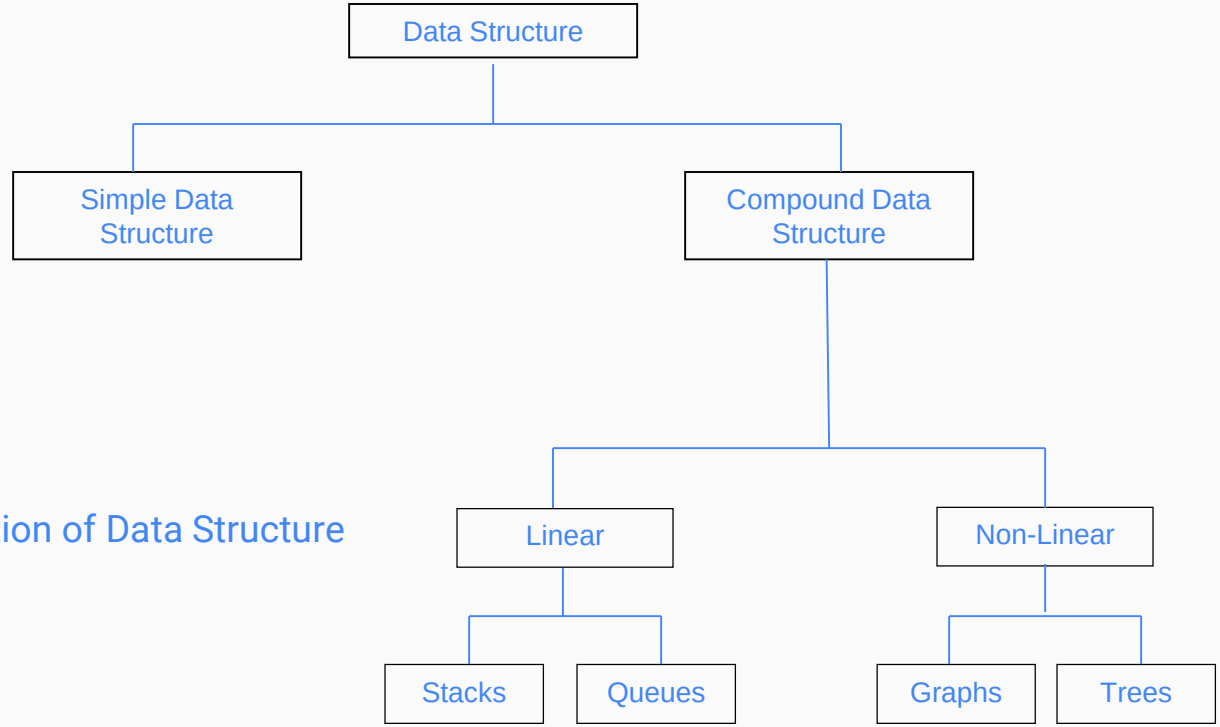


Fig: Classification of Data Structure

Types/Classification of Data Structure

i. **Simple data structure:** A primitive data structure helps in the construction of simple data structure. A primitive data is used to represent the standard data types of any one of the computer languages.

Array, float, structure etc. are the example of simple data structure.

ii. **Compound data structure:** It can be constructed with the help of any one of the primitive data structure and the specific functionality. It is of two types : linear and non linear data structures.

Operations on Data Structure

- Insertion : adding a new record to the structure
- Deletion : removing a record from the structure
- Traversing: accessing each record exactly once so that certain item in the record may be processed.
- Searching: finding the location of the record with a given key value .
- Sorting: arranging the data in some logical order for example in numerical order or alphabetically
- Merging: combining the data of two different sorted files into a single sorted file.

Data Type

- A **data type** is a collection of values and a set of operations on those values. Lets us take an example of integer datatype.

- The type int in C consist of the values

  { 0, +1, -1, +2, -2,……., MAXINT, MININT}

- The operation in int includes the arithmetic operators.

  +, -, * and / along with operator for testing equality and inequality.

Abstract Data Type (ADT)

- **Abstraction**: It can be defined as breaking down the problem into its most fundamental parts and describes this parts in a simple precise language.
- An ADT is a **mathematical model of data type** that specifies the typed data stored. The operation supported on them and the type of parameters of the operations.
- This ADT is made of with primitive datatypes but operation logic are generally hidden.
- This model **contrasts with the data structures**, which are concrete representations of data and are the point of view of an implementer, not a user.

Abstract Data Type (ADT)

- An ADT specifies what operation does but not how it does that. ADT is also called programmer defined data type. ADT consists of two parts:
    i. Value definition
    ii. Operator definition (Operations)

i. **Value definition**: The value definition parts defined the values of an instances of ADT can have.
ii. **Operator definition**: It is defined the set of operation that can be applied to the instance of that ADT.

Abstract Data Type (ADT)

Defining an abstract data type:

- ADTs are theoretical concept.
- It can be used in the design and analysis of algorithms, data structure and software systems & they do not correspond to specific features of computer languages.
- As it is a mathematical model of the data objects that make up a data type as well as functions that operate on these objects.
- So, there is **no standard conventions** on defining them and depends upon the implementer (programmers) point of view.

Example: Rational Number as an ADT

Rational number can be specified as an ordered pair of integer p and q where p is the numerator and q is the denominator in which q is not equal to zero.

i.e. $\frac{p}{q}$ Where, $q \neq 0$

Values:

For all a, b, c € Rational, p and q € integer. T, F € Boolean and *, +, == are usual integer operations.

Example: Rational Number as an ADT

Operator definition:

1. Make rational(**p1,q1**): Rational (return type)

   It creates and returns a new rational '**a**' with its numerator initialize to **p1** and denominator initialize to **q1**.

Pre: q1 ≠ 0

Post: numerator (a) = p1

denominator (a) = q1

So, **a = p1 / q1**

Similarly, **b = p2 / q2**

Example: Rational Number as an ADT

Operator definition:

2. Add rational(**a , b**): Rational

Add two rational number a and b and returns a new rational number c.

i.e. c = a + b

Or, c = (p1 /q1) + (p2 / q2) where, a = (p1 / q1) and b = (p2 / q2)

Or, c = (p1 * q2 + p2 * q1) / (q1 * q2)

Pre: q1 ≠ 0, q2 ≠ 0

Post: **numerator (c) = p1 * q2 + p2 * q1** = numerator(a) * denominator (b) +

numerator(b) * denominator(a)

denominator (c) = q1 * q2 = denominator (a) * denominator (b)

Example: Rational Number as an ADT

Operator definition:

3. Multiply (a, b) : rational

Multiply two rational number **a** and **b** and return a new rational number **c**.

i.e. c = a * b = (p1/q1) * (p2/q2) = (p1 * p2) / (q1* q2)

Pre: q1 ≠ 0, q2 ≠ 0

Post : **numerator(c)** = (p1 * p2) = numerator(a) * numerator (b)

denominator (c) = (q1* q2)= denominator (a) * denominator (b)

Example: Rational Number as an ADT

Operator definition:

4. Equal(a, b) : Boolean

Determines if two rational number are equal or not.

i.e. a = b

Or, (p1/q1) = (p2/q2)

Therefore, **p1 * q2 = p2 * q1**

Pre: q1 ≠ 0, q2 ≠ 0

Post : numerator(a) * denominator (b) = numerator(b) * denominator (a)

ADT Vs Data Structure

Abstract Data Type	Data Structure
ADT is logical picture of data and operation that manipulates them.	It is systematic collection of data elements and operations that store & retrieve individual element and also manipulate its physical properties.
ADT is implementation independent.	It is implementation dependent
Time and space complexity does not matter.	Time and space complexity matter.
It is just theoretical concept.	It is practical oriented with real world implementation

QUESTIONS TO FOCUS

1. Difference between ADT and data structure.
2. What do you mean by data structure ? Do you think every program needs to be in the concept of data structure?
3. Define data type. Explain the need of data structure in the development of program.
4. What are the pros & cons of data structures ?
5. Explain linear & non linear data structure with its examples.
6. Explain natural number as an ADT.
7. Write short notes on:
    i. Simple Vs Compound Data Structure
    ii. Primitive Vs Non Primitive Data Structure
    iii. ADT

THANK YOU

Any Queries ?

ankit.bca@kathford.edu.np