UCS1302: DATA STRUCTURES

Introduction to data structures

Session Meta Data

Author	**Dr. B. Bharathi**
Reviewer	
Version Number	1.2
Release Date	26 June 2019

Revision History

Revision Date	Details	Version no.
22 September 2017	1. New SSN template applied	1.2

Session Objectives

- To study the introduction about data structures
- To understand Abstract Data Type

Session Outcomes

- At the end of this session, participants will be able to
  - Understand the concepts of data structures
  - Different types of data structures and its application

Agenda

- Introduction
- Linear data structure
- Non linear data structure

Introduction to Data structures

Dr. B. Bharathi

SSNCE

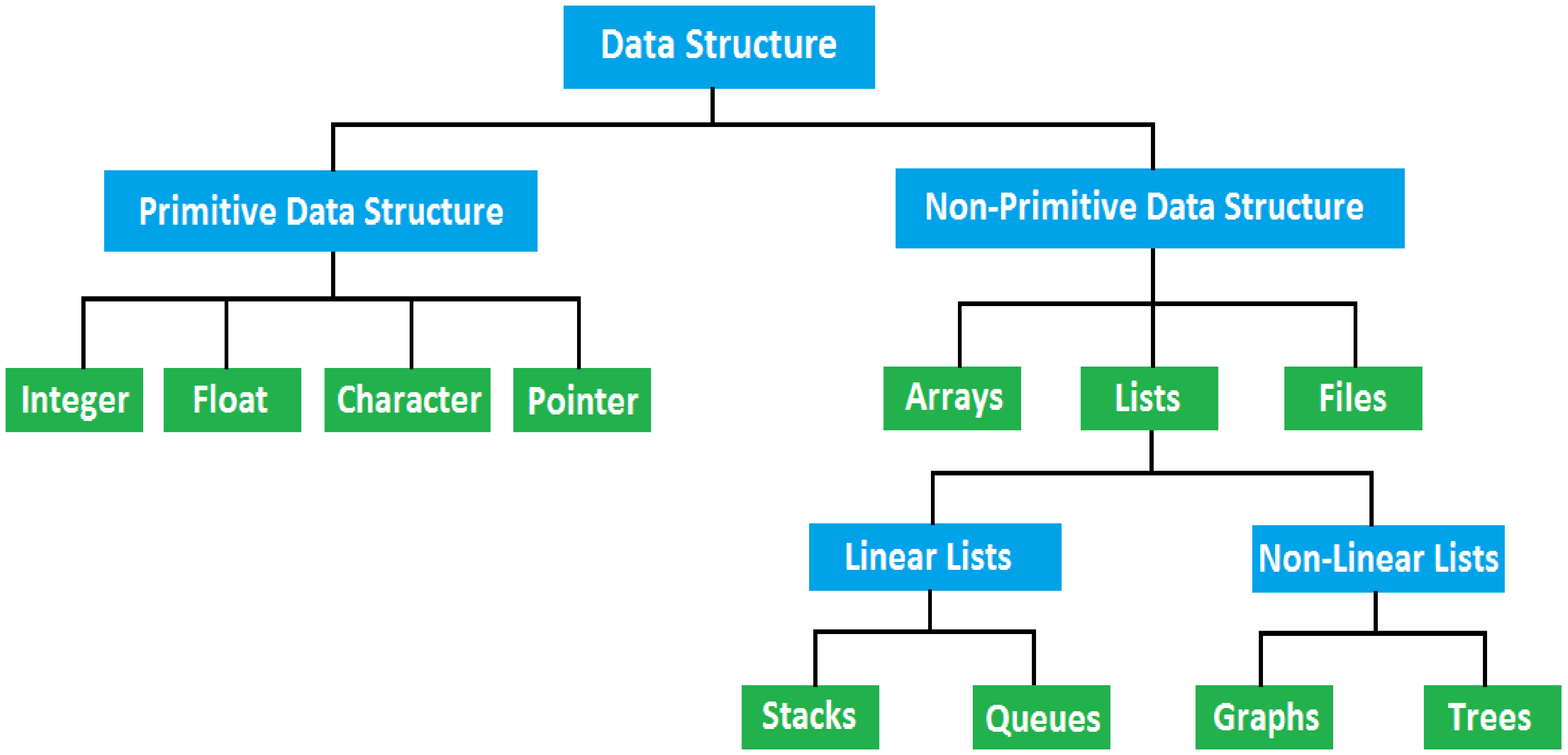
June 26, 2019

Introduction

- Data structure
    - Data – are simply a value or set of values of different type which is called data types called string, integer, char
    - Structure – Way of organizing the information so that it is easier to use.

    - In simple words we can define the data structures as
        - It’s a way of organizing data in such a way so that data can be easier to use.

Classification of data structure



Data structure

Primitive data structure are basic data structures that directly operate upon the machine instructions. They have different representation on different computers.

Eg. Integer, float, character etc

Non primitive data structure:

- These are more sophisticated data structures
- These are derived from primitive data structures
- They emphasize on grouping of homogenous or heterogeneous data items

Eg. Array, List, File

Data structure

Linear data structure: A linear data structure traverses the data elements sequentially, in which only one data element can directly be reached.

Eg. Stacks, queues and linked list

Non linear data structure: Every data item is attached to several other items in a way that is specific for reflecting relationships. The data items are not arranged in sequential structure.

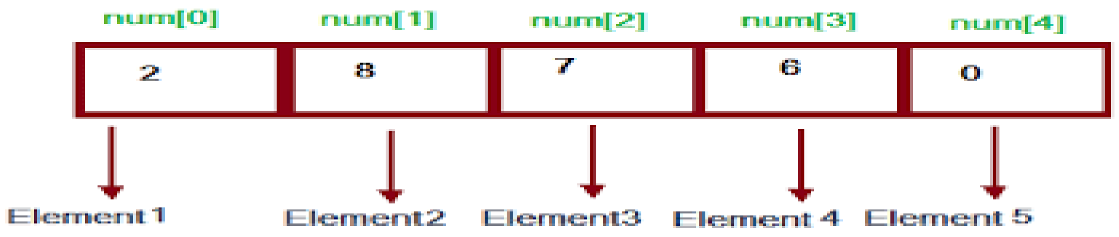
Eg. Trees, Graphs

Operations on linear or non linear data structure

- Add an element
- Delete an element
- Display the elements
- Sort the elements
- Search for an element

Arrays

- An array is defined as a set of finite number of homogeneous elements or same data items.
- It means an array can contain one type of data only, either all integer, all float-point number or all character.

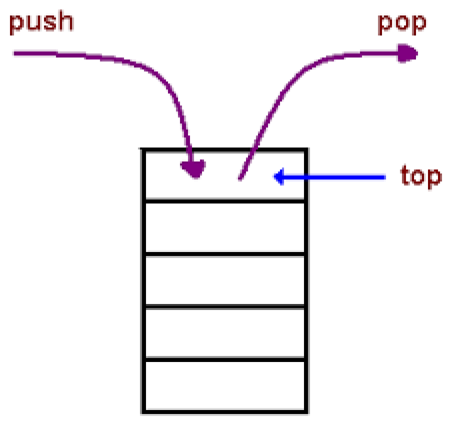


Stack

- A stack is also an ordered collection of elements like arrays, but it has a special feature that deletion and insertion of elements can be done only from one end called the top of the stack (TOP)
- Due to this property it is also called as last in first out type of data structure (LIFO).

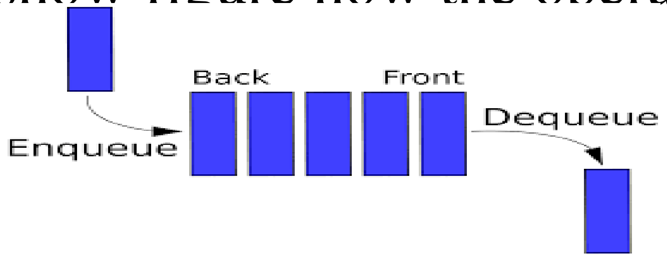
Stack

- When an element is inserted into a stack or removed from the stack, its base remains fixed where the top of stack changes.
- Insertion of element into stack is called PUSH and deletion of element from stack is called POP.
- The bellow show figure how the operations take place on a stack:



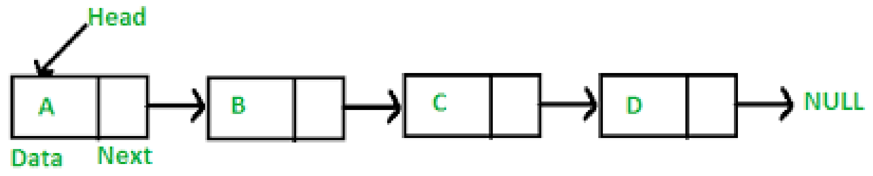
Queue

- Queue are first in first out type of data structure (i.e. FIFO)
- In a queue new elements are added to the queue from one end called REAR end and the element are always removed from other end called the FRONT end.
- The people standing in a railway reservation row are an example of queue.
- Each new person comes and stands at the end of the row and person getting their reservation confirmed get out of the row from the front end.
- The bellow show figure how the operations take place on a queue:



Lists

- A lists (Linear linked list) can be defined as a collection of variable number of data items.
- Lists are the most commonly used non-primitive data structures.
- An element of list must contain at least two fields, one for storing data or information and other for storing address of next element.
- As you know for storing address we have a special data structure of list the address must be pointer type.
- Each such element is referred to as a node.



Trees

- Tree is non-linear type of data structure.
- Tree represent the hierarchical relationship between various elements.
- There is a special data item at the top of hierarchy called the Root of the tree.
- The remaining data items are partitioned into number of mutually exclusive subset, each of which is itself, a tree which is called the sub tree.

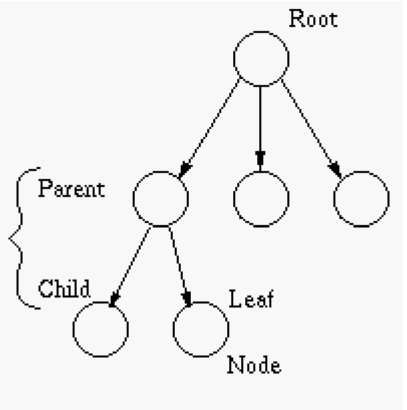
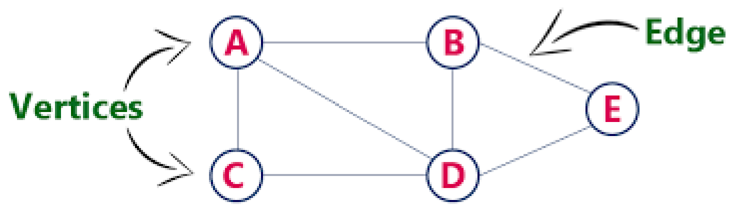


Figure: Tree data structure

Graph

- It is a set of items connected by edges. Each item is called as vertex or node.
- Definition: A graph G(V,E) is a set of vertices V and a set of edges E.
- An edge connects a pair of vertices and many have weight such as length or cost etc.
- Vertices on the graph are shown as point or circles and edges are drawn as arcs or line segment.



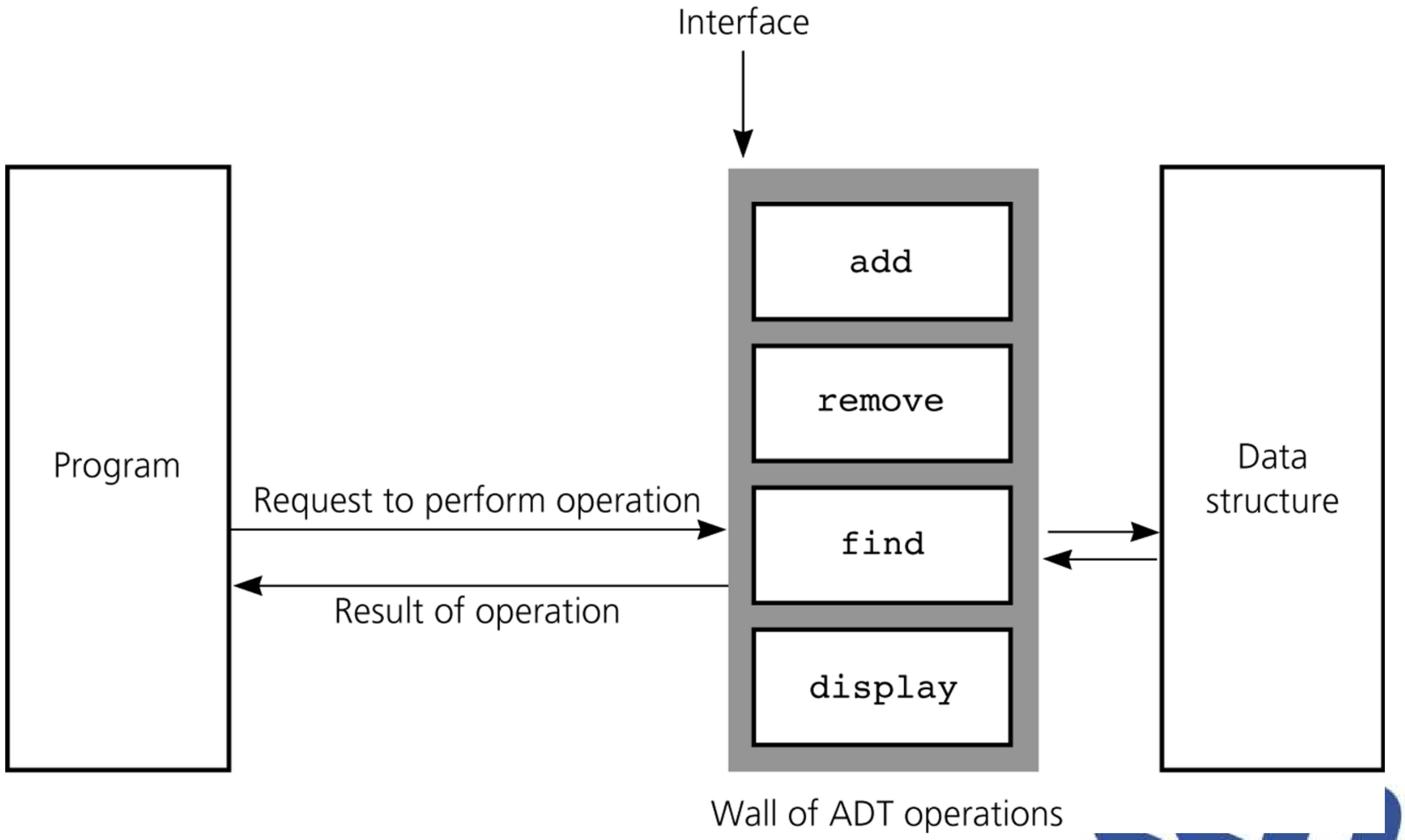
Abstract data type

- ADT is a mathematical model of a set of data items and operations on the data items.
- Implementation consists of
    - Storage **(data) structures**
    - **Algorithms** for basic operations
- ADT does not imply how these operations are implemented
- Typical ADTs are Lists, Stacks, Queues, Sets, Maps, Graphs and Priority Queues

Built-in Types as ADTs

- Definition of the ADT integer
  - Data items:
    - an integer value in the set {…,-3, -2, -1, 0, 1, 2, 3,…}
    - Maximum and minimum values are determined by the storage representation used
  - Operations:
    - Binary arithmetic operations: +, -, *, /, %
    - Unary arithmetic operations: +, -
    - Relational operations: ==, !=, <, <=, >, >=

Abstract data type



Abstract data type

- C is not object-oriented, but we can still manage to inject some object-oriented principles into the design of C code.
- For example, a data structure and its operations can be packaged together into an entity called an ADT.
- There's a clean, simple interface between the ADT and the program(s) that use it.
- The lower-level implementation details of the data structure are hidden from view of the rest of the program.
- The implementation details can be changed without altering the ADT interface.

Abstract data type

- This can be accomplished by creating the ADT in three different files:
  - One to hold the type and constant definitions.
  - One to hold the prototypes of the functions in the ADT’s (public) interface.
  - One to hold the implementations of the public and private functions.

LIST ADT

- Array implementation of LIST
- Linked list implementation of LIST

List operations

- Insert
- Delete
- Retrieve
- Isempty
- Getlength

Summary

- Introduction
- Linear data structure
- Non linear data structure
- ADT