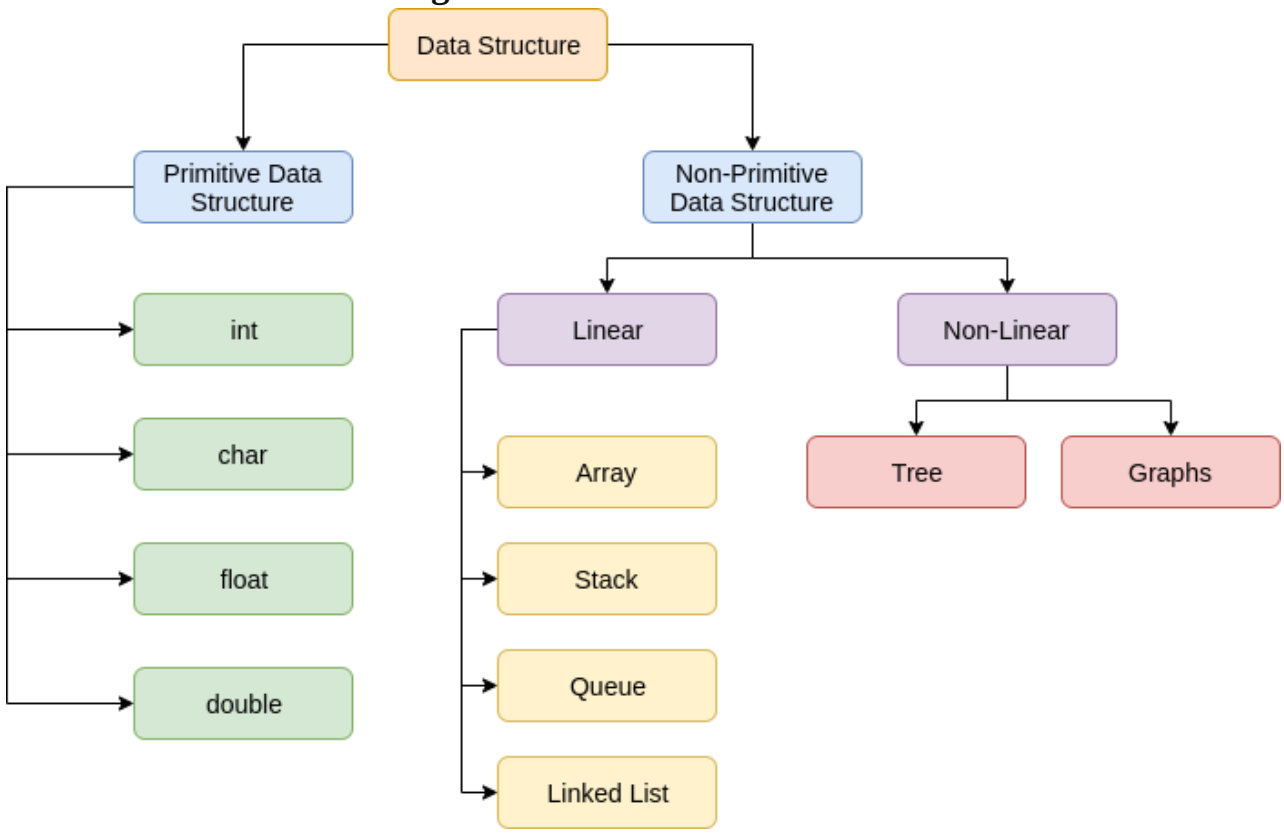
Experiment No. 20

Aim: Case study on Data Structure in C

What are Data Structures using C?

- It is a way to arrange data in computer
  List of Data Structures using C



- Array
- Linked List
- Stack
- Queue
- Binary Tree
- Hashing
- Graph

1.Array

- Linear Data Structures using C
- Elements are stored in contiguous memory locations

- Can access elements randomly using index
- Stores homogeneous elements i.e, similar elements
- Syntax:
- Array declaration
    - Datatype varname [size] ;
- Can also do declaration and initialization at once
    - Datatype varname [] = {ele1, ele2, ele3, ele4};

Advantages

- Random access
- Easy sorting and iteration
- Replacement of multiple variables

Disadvantages

- Size is fixed
- Difficult to insert and delete
- If capacity is more and occupancy less, most of the array gets wasted
- Needs contiguous memory to get allocated

2. Linked List

- Linear Data Structure
- Elements can be stored as per memory availability
- Can access elements on linear fashion only
- Stores homogeneous elements i.e, similar elements
- Dynamic in size
- Easy insertion and deletion
- Starting element or node is the key which is generally termed as the head.

Advantages

- Dynamic in size
- No wastage as capacity and size is always equal
- Easy insertion and deletion as 1 link manipulation is required
- Efficient memory allocation

Disadvantages

- If the head node is lost, the linked list is lost
- No random access possible

2. Stack

- It is a type of Linear Data Structures using C
- Follows LIFO: Last In First Out
- Only the top elements are available to be accessed
- Insertion and deletion takes place from the top
- Eg: a stack of plates, chairs, etc
- 4 major operations:
    - push(ele) – used to insert element at top
    - pop() – removes the top element from stack
    - isEmpty() – returns true is stack is empty
    - peek() – to get the top element of the stack
- All operation works in constant time i.e, O(1)

Advantages

- Maintains data in a LIFO manner
- The last element is readily available for use
- All operations are of O(1) complexity

Disadvantages

- Manipulation is restricted to the top of the stack
- Not much flexible

3. Queue

- Linear Data Structures using C
- Follows FIFO: First In First Out
- Insertion can take place from the rear end.
- Deletion can take place from the front end.
- Eg: queue at ticket counters, bus station
- 4 major operations:
    - enqueue(ele) – used to insert element at top

    - dequeue() – removes the top element from queue
    - peekfirst() – to get the first element of the queue
    - peeklast() – to get the last element of the queue
- All operation works in constant time i.e, O(1)

Advantages

- Maintains data in FIFO manner
- Insertion from beginning and deletion from end takes O(1) time

Applications

- Scheduling
- Maintaining playlist
- Interrupt handling

3. Binary Tree

- Hierarchical Data Structures using C
- Topmost element is known as the root of the tree
- Every node can have at most 2 children in the binary tree
- Can access elements randomly using index
- Eg: File system hierarchy
- Common traversal methods:
    - preorder(root) : print-left-right
    - postorder(root) : left-right-print
    - inorder(root) : left-print-right

Advantages

- Can represent data with some relationship
- Insertion and search are much efficient

Disadvantages

- Sorting is difficult
- Not much flexible

4. Hashing

- Uses special Hash function
- A hash function maps element to an address for storage
- This provides constant-time access
- Collision is handled by collision resolution techniques
- Collision resolution technique
    - Chaining
    - Open Addressing

Advantages

- The hash function helps in fetching element in constant time
- An efficient way to store elements

Disadvantages

- Collision resolution increases complexity

7 Graph

- Basically it is a group of edges and vertices
- Graph representation
    - G(V, E): where V(G) represents a set of vertices and E(G) represents a set ofedges
- A graph can be directed or undirected.A graph can be connected or disjoint

Advantages

- finding connectivity
- Shortest path
- min cost to reach from 1 pt to other
- Min spanning tree

Disadvantages

- Storing graph(Adjacency list and Adjacency matrix) can lead to complexities