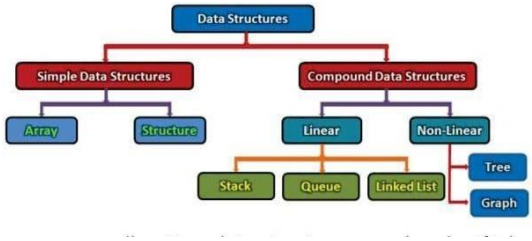
Data Structures and Operations

Definition

Data structure is a particular way of organising logically related data items which can be processed as a single unit. **Classification of data structures**



Depending upon memory allocation, data structures may be classified as **static data structures** and **dynamic data structures**. Memory allocation is fixed for static data structures (eg: arrays) and the **size cannot be changed** during execution. Memory is allocated during execution for dynamic data structures (eg: linked list) and the **size changes according to the addition or deletion** of data items.

Operations on Data Structures

The operations performed on data structures are traversing, searching, inserting, deleting, sorting and merging.

1. Traversing

The process of visiting each elements in a data structure is called traversing.It starts from first element to the last element.

2. Searching

The process of finding the location of a particular element in a data structure is called searching.It is based on a condition. **C. Insertion**

The process of adding a new data at a particular position in a data structure is called inserting.The position is identified first and insertion is then done.

4. Deletion

The process of removing a particular element from the data structure is called deletion.

5.Sorting

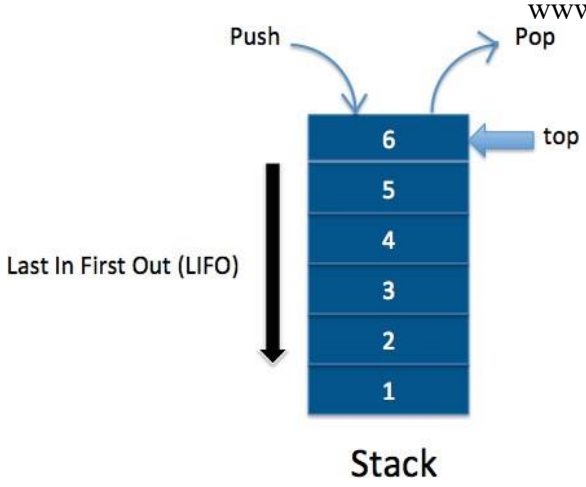
The process of arranging elements of a data structure in an order(ascending or descending) is called sorting.

6. Merging

The process of combining elements of two data structures is called merging.

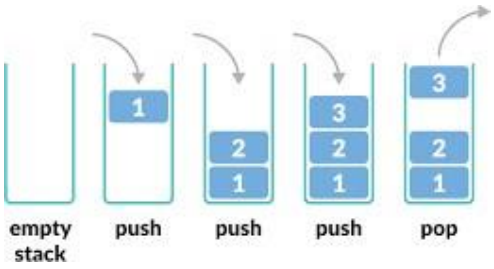
Stack and its Operations

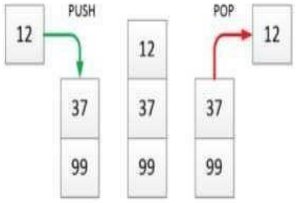
Stack is a linear data structure that follows **LIFO (Last In First Out)** principle. It is an ordered list of items in which all insertions and deletions are made at one end, usually called Top.



Push Operation: It is the process of inserting a new data item into the stack at Top position. Once the stack is full and if we attempt to insert an item, an impossible situation arises, known as **stack overflow**.
Pop Operation: It is the process of deleting an element from the top of a stack. I f we try to delete an item from an empty stack, an unfavourable situation arises, known as s**tack underflow.**

Algorithm for Push:
Assume that STACK[N] is an array of stack with size N and TOS denotes the top position of the stack. Let VAL contains the data to be added into the stack.





Start
1: If (TOS < N-1) Then //Space availability checking (Overflow)
2: TOS = TOS + 1
3: STACK[TOS] = VAL
4: Else
5: Print "Stack Overflow "
6: End of If
Stop

Algorithm for Pop:
Assume that STACK[N] is an array of stack with size N and TOS denotes the top position of the stack. Let VAL be a variable to store the popped item.

Start
1: If (TOS > -1) Then //Empty status checking (Underflow)
2: VAL = STACK[TOS]

```
3:              TOS = TOS - 1
4:          Else
5:              Print "Stack Underflow "
6:          End of If
Stop
```

Implementation of Stack

A Stack can be implemented in two ways **1)Using an array.**

In implementing a stack using an array,the number of elements is limited to the size of array.When an element is added to the stack top of stack(TOS) is incremented by1.In array implementation a stack can towards end index of the array.The array implementation has following dis-advantages

1. Memory space is wasted.
2. The size of stack is fixed.
3. Insertion and deletion operation involved more data movement.

2)Using a Linked list.

In Linked list implementation of a stack the first element inserted points to second element,second element points to third element and so on.Here stack is not of fixed size.

Applications of Stack

The following are the applications of a stack

- Decimal to binary conversion.
- Reversing a string.
- Infix to postfix conversion.

Queue and its Operations

Queue is a data structure that follows the **FIFO (First In First Out)** principle. A queue has two end points - **Front and Rear**. Insertion of a data item will be at the **rear end** and deletion will be at the **front end**.

Insertion is the process of adding a new item into a queue at the rear end. One the value of Rear equals the last position and if we attempt an insertion, queue **overflow** occurs.

Deletion is the process of removing the item at the front end of a queue. If we attempt a deletion from an empty queue, **underflow** occurs.

Algorithm for Insertion:

Assume that Q[N] is an array of queue with size N and FRONT and REAR denote the front and rear positions of the queue. Let VAL contains the data to be added into the queue.

```
Start
1: If (REAR == -1) Then
2:      FRONT = REAR = 0
3:      Q[REAR] = VAL
4: Else If (REAR < N-1) Then
5:      REAR = REAR + 1
6:      Q[REAR] = VAL
7: Else
```

8: Print "Queue Overflow "
9: End of If Stop

Algorithm for Deletion:

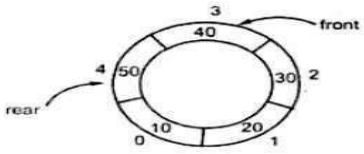
Assume that Q[N] is an array of queue with size N and FRONT and REAR denote the front and rear positions of the queue. Let VAL be a variable to store the deleted data.

Start
1: If (FRONT > -1) Then // Empty status checking
2: VAL = Q[FRONT]
3: FRONT = FRONT + 1
4: Else
5: Print "Queue Underflow "
6: End of If
7: If (FRONT > REAR) Then // Checking the deletion of last element
8: FRONT = REAR = -1
9: End of If
Stop

Applications of Queue

- Job Scheduling.
- Resource scheduling.
- Handling of interrupts in real-time systems.

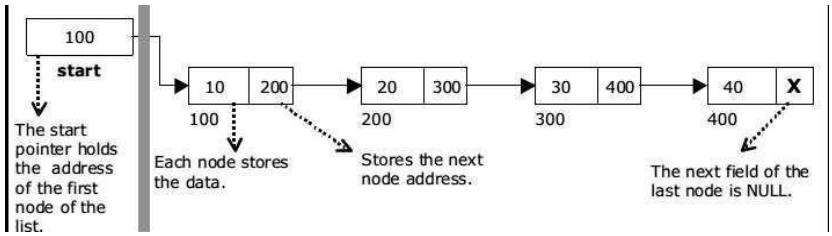
Circular Queue



A circular queue allows to store elements without shifting any data within the queue.The main advantage of circular queue is that we can utilize the space of queue fully.It allows to store data without shifting any data in the queue.

Linked List and Operations

Linked list is a collection of nodes, where each node consists of two parts – a **data** and a **link**. Link is a pointer to the next node in the list.



The address of the first node is stored in a special pointer called START. Linked list is a dynamic data structure. Memory is allocated during run time. So there is no problem of overflow. **It grows as and when new data items are added, and shrinks whenever any data is removed.** Linked list is created with the help of self referential structures.

Implementing linked list

Linked list can be implemented in two ways,both as stack and queue ie,

- using array (static linked list)
- using self-referential structures(Dynamic linked list)

Difference between Array and Linked list

The following are the difference between array and linked list

- The size of array is fixed whereas a linked list can grow or shrink.
- Array elements are accessed randomly.Linked list elements are accessed sequentially.

Operations on linked list

The following operations can be performed on a linked list a.Creating a Linked list. b.Traversing a Linked list.
c.Inserting elements to a Linked list.
d.Deleting elements from a Linked lis

Creation of a Linked List:

Step 1: Create a node and obtain its address.
Step 2: Store data and NULL in the node.
Step 3: If it is the first node, store its address in START.
Step 4: If it is not the first node, store its address in the link part of the previous node. Step 5: Repeat the steps 1 to 4 as long as the user wants.

Traversing a linked list

Step 1: Get the address of the first node from START and store it in Temp.
Step 2: Using the address in Temp, get the data of the first node and store in Val.
Step 3: Also get the content of the link part of this node (i.e., the address of the next node) and store it in Temp. Step 4: If the content of Temp is not NULL, go to step 2; otherwise stop.

Insertion in a linked list

Step 1: Create a node and store its address in Temp.
Step 2: Store the data and link part of this node using Temp.
Step 3: Obtain the addresses of the nodes at positions (POS-1) and POS in the pointers PreNode and PostNode respectively, with the help of a traversal operation.
Step 4: Copy the content of Temp (address of the new node) into the link part of node at position (POS-1), which can be accessed using PreNode.
Step 5: Copy the content of PostNode (address of the node at position POS) into the link part of the new node that is pointed to by Temp.

Deletion in a linked list

Step 1: Obtain the addresses of the nodes at positions (POS-1) and (POS+1) in the pointers

PreNode and PostNode respectively, with the help of a traversal operation. Step 2: Copy the content of PostNode (address of the node at position POS+1) into the link part of the node at position (POS-1), which can be accessed using PreNode. Step 3: Free the node at position POS.

Questions

1. name the data structure where memory allocation is done only at the time of execution Answer : Dynamic datastructure
2. Write an algorithm to add new data item into a stack
3. What is datastructure?How are they classified?
4. Queue follows ..... principle(Answer:FIFO Rule)
5. How does stack overflow and underflow occurs?
6. Write a procedure to implement traversal operation in a linked lsit
7. Attempting to insert in an already full stack leads to ..... Answer: Stack overflow
8. Explain how push operations is done in a stack 9. Linked list do not have the problem of overflow.Discuss?
10. Name the data structure that follows LIFO principle.
    (a) stack (b) queue (c) array (d) linked list
11. Write an algorithm to perform insertion operation in a Queue.
12. Match the following:

A	B	C
1. stack	i. Front	a. Inserting a new item.
2. Queue	ii. Push	b. Elements are accessed by specifying its position.
3. Array	iii. Start	c. Contains the address of the first node.
4. Linked List	iv. Subscript	d. Removing an item.

13. Write down the full form of FIFO and LIFO
14. People waiting in a cinema theatre counter is an exmple for ..... (Answer:Queue)
15. A link list is a linear collection of data element is called.....(Answer:Node) 16. Placing glasses one above another can be considered similar to.....data structure
17. write a short not on circular queue.
18. Prepare a short notes about all the operations associated with datastructure
19. A linked list containing all the names of students in your class is to be created.Write its C++ structure to define the node
    Answer: struct node
    {

    ```
    char    name[64];
    node *link; }
    ```

20. Write a short notes about queue?
21. Define push and pop operation
22. Write the algorithm to add an item in to a queue which is not empty ?(March 2020)
23. Explain about the operations performed on stack data structure.(March 2020)