
Data Structure

Trainer : Nisha Dingare

Email : nisha.dingare@sunbeaminfo.com



Queue :

Queue: It is a collection/list of logically related similar type of elements into which elements can be added from one end referred as **rear** end, whereas elements can be deleted from another end referred as a **front** end.

- In this list, element which was inserted first can be deleted first, so this list works in **first in first out** manner, hence this list is also called as **FIFO list/LILO list**.

- Two basic operations can be performed on queue in $O(1)$ time.

1. **Enqueue:** to insert/push/add an element into the queue from rear end.

2. **Dequeue:** to delete/remove/pop an element from the queue which is at front end.

- There are different types of queue:

1. **Linear Queue** (works in a fifo manner)

2. **Circular Queue** (works in a fifo manner)

3. **Priority Queue:** it is a type of queue in which elements can be inserted from rear end randomly (i.e. without checking priority), whereas an element which is having highest priority can only be deleted first.

- Priority queue can be implemented by using linked list, whereas it can be implemented efficiently by using **binary heap**.

4. **Double Ended Queue (deque)** : it is a type of queue in which elements can added as well as deleted from both the ends.



Applications of Queue :

Applications of Queue:

- Queue is used to implement OS data structures like **job queue, ready queue, message queue, waiting queue** etc...
- Queue is used to implement OS algorithms like **FCFS CPU Scheduling, Priority CPU Scheduling, FIFO Page Replacement** etc...
- Queue is used to implement an advanced data structure algorithms like **BFS: Breadth First Search** Traversal in tree and graph.



Stack :

- It is a basic/ linear /utility data structure.
- It is a collection/list of logically related similar type elements into which data elements can be added as well as deleted from only one end referred as “**top**” end.
- In this collection, element which was inserted last only can be deleted first, so this list works in “**last in first out /first in last out**” manner, and hence it is also called as **LIFO list/FILO list**.
- We can perform basic three operations on stack in $O(1)$ time: Push, Pop & Peek.
 - **Push** : to insert/add an element onto the stack at top position
 - step1: check stack is not full.
 - step2: increment the value of top by 1.
 - step3: insert an element onto the stack at top position.
 - **Pop** : to delete/remove an element from the stack which is at top position.
 - step1: check stack is not empty.
 - step2: decrement the value of top by 1.
 - **Peek** : to get the value of an element which is at top position without push & pop.
 - step1: check stack is not empty.
 - step2: return the value of an element which is at top position.
- Stack Empty : $top == -1$
- Stack Full : $top == SIZE-1$



Stack :

- Applications of Stack :
 - Stack is used by an OS to control of flow of an execution of program.
 - In recursion internally an OS uses a stack.
 - undo & redo functionalities of an OS are implemented by using stack.
 - Stack is used to implement advanced data structure algorithm like DFS: Depth First Search traversal in tree & graph.
 - Stack is used in algorithms to covert given infix expression into its equivalent postfix and prefix, and for postfix expression evaluation.
- Time complexity :
 - All the operations push, pop and peek take $O(1)$ time.
- .dynamic implementation of stack by using linked list (dll):
 - push : add_last()
 - pop : delete_last()
 - OR
 - push : add_first()
 - pop : delete_first()



Stack Applications :

- Stack Application Algorithms: -
 - To evaluate postfix expression.
 - To evaluate prefix expression.
 - To convert given infix expression into its equivalent postfix expression.
 - To convert given infix expression into its equivalent prefix expression.
- What is an expression?
 - An expression is a combination of an operands and operators.
there are 3 types of expression:
 1. infix expression : $a+b$
 2. prefix expression : $+ab$
 3. postfix expression : $ab+$



Thank You !

