**Difference between Stack and Queue Data Structures**

Stacks

The Stack is one of the most important data structures in computer science. Elements can be added to or removed from a stack at only one end. Since we view this end as the "top" of the stack, we use the term "push" for add, and the term "pop" for remove. Notice that the element popped from a stack is always the last one pushed onto it (among those still on it). Therefore, a stack is referred to as a Last-In-First-Out (LIFO) list.

Examples of stacks in "real life":

* The stack of trays in a cafeteria;
* A stack of plates in a cupboard;
* A driveway that is only one car wide. [Riley, p. 290]

Examples of stacks in computing:

* Back/Forward stacks on browsers;
* Undo/Redo stacks in Excel or Word;
* Activation records of method calls;

The Stack ADT

Contains the methods push, pop, empty, and peek (often called top). See the [java.util.Stack](http://java.sun.com/j2se/1.4/docs/api/java/util/Stack.html) API.

Two Implementations of Stacks

* [Using LinkdList](https://cse.buffalo.edu/~shapiro/Courses/CSE116/Demos/Lists/LStack.java) ([API](https://cse.buffalo.edu/~shapiro/Courses/CSE116/Demos/Lists/JavaDoc/LStack.html))
* [Using ArrayList](https://cse.buffalo.edu/~shapiro/Courses/CSE116/Demos/Lists/AStack.java) ([API](https://cse.buffalo.edu/~shapiro/Courses/CSE116/Demos/Lists/JavaDoc/AStack.html))

[java.util.Stack](http://java.sun.com/j2se/1.4/docs/api/java/util/Stack.html) extends Vector, and so, is less restricted than the "official" stack, but see the Go menu on a browser.

Queues

Elements can be added only at one end, the "rear", Elements can be removed only at the other end, the "front".  
We call adding to a queue "enqueueing", and removing from a queue "dequeueing".  
Since the element dequeued is always the first one enqueued (among those still on it), a queue is referred to as a First-In-First-Out (FIFO) list.

Examples of queues in "real life":

* A ticket line;
* An escalator;

## Applications of Stack in Real Life

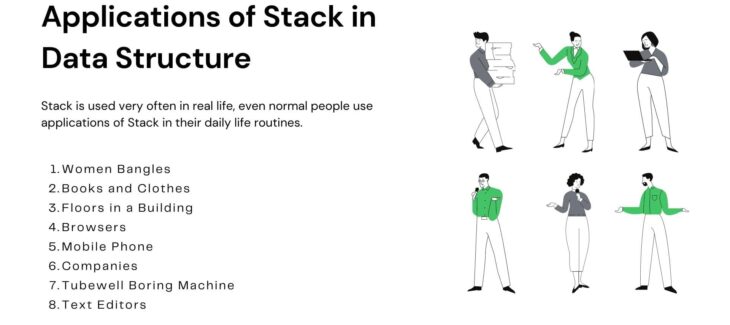
Stack is used very often in real life, even normal **people** use [applications of Stack](https://tutorialsbookmarks.com/applications-of-stack-in-data-structure/) in their daily life routines. Here is some example of the stack in real-life. Below are the 10 Applications of Stack in Real Life.

1. **Women Bangles:** Women wear a bangle one by one and to pull the first one they have to first pull out the last one.
2. **Books and Clothes:** Piled on top of each other is a great example of the stack.
3. **Floors in a Building:** A person is living on a top floor and wants to go outside, he/she first need to land on the ground floor.
4. **Browsers:** Web browsers use the stack to keep track of the history of web sites if you click back then the previous site opens immediately.
5. **Mobile Phone:** Call log in mobiles uses the stack, to get a first-person call log you have to scroll.
6. **Companies:** When a company want to reduce its workforce. Typically they follow “last hired, first fired”
7. **Garage:** If a garage is not wide enough. To remove the first car we have to take out all the other cars in after it.
8. **Tubewell Boring Machine:** Tubewell boring machine use stack to pull or push same as stack
9. **Text Editors:** Undo or Redo mechanism in the Text Editors(Excel, Notepad or WordPad etc.)
10. The CD/DVD stand

### Application of Stack in Data Structure

Apart from the real-life example of stack here are some different applications of the stack in Data Structure. The functionality will be the same else we can’t say it stack.

1. Memory management
2. Function Call(recursive functions.)
3. String Reversal
4. Parenthesis Checking
5. [Backtracking](https://www.geeksforgeeks.org/backtracking-introduction/)
6. Syntax Parsing
7. Reversing a String
8. Matching HTML Tags in Web Developing.
9. Arithmetic Expression Evaluation
10. Java compiler uses postfix notation
11. Java virtual machine uses a stack
12. Expression Conversion or Expression Evaluation
    1. infix to prefix
    2. infix to postfix
    3. prefix to infix
    4. prefix to postfix
    5. postfix to infix
    6. postfix to infix



[**https://www.geeksforgeeks.org/introduction-to-stack-data-structure-and-algorithm-tutorials/**](https://www.geeksforgeeks.org/introduction-to-stack-data-structure-and-algorithm-tutorials/)

[**https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/**](https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/)

[**https://www.geeksforgeeks.org/queue-data-structure/#intro**](https://www.geeksforgeeks.org/queue-data-structure/#intro)

[**https://www.geeksforgeeks.org/circular-queue-set-1-introduction-array-implementation/**](https://www.geeksforgeeks.org/circular-queue-set-1-introduction-array-implementation/)

[**https://www.geeksforgeeks.org/circular-queue-set-2-circular-linked-list-implementation/**](https://www.geeksforgeeks.org/circular-queue-set-2-circular-linked-list-implementation/)

[**https://www.javatpoint.com/circular-queue**](https://www.javatpoint.com/circular-queue)

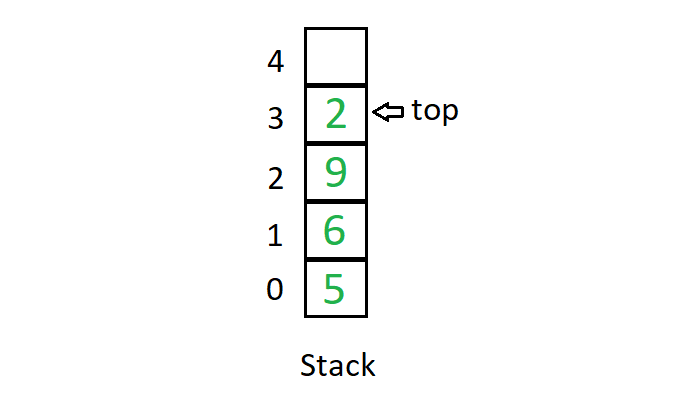
* Difficulty Level : [Medium](https://www.geeksforgeeks.org/medium/)
* Last Updated : 15 Sep, 2022

 Read

 Discuss

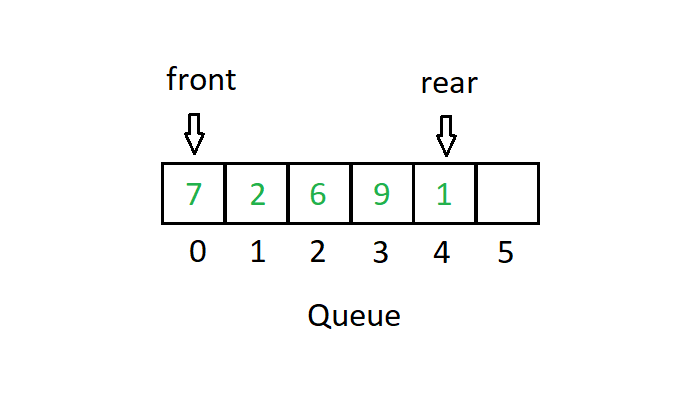
**Stack** A [stack](https://www.geeksforgeeks.org/stack-data-structure-introduction-program/) is a linear data structure in which elements can be inserted and deleted only from one side of the list, called the **top**. A stack follows the **LIFO** (Last In First Out) principle, i.e., the element inserted at the last is the first element to come out. The insertion of an element into the stack is called **push** operation, and the deletion of an element from the stack is called **pop** operation. In stack, we always keep track of the last element present in the list with a pointer called **top**.

The diagrammatic representation of the stack is given below:



[Queue](https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/) is a linear data structure in which elements can be inserted only from one side of the list called **rear**, and the elements can be deleted only from the other side called the **front**. The queue data structure follows the **FIFO** (First In First Out) principle, i.e. the element inserted at first in the list, is the first element to be removed from the list. The insertion of an element in a queue is called an **enqueue** operation and the deletion of an element is called a **dequeue** operation. In queue, we always maintain two pointers, one pointing to the element which was inserted at the first and still present in the list with the **front** pointer and the second pointer pointing to the element inserted at the last with the **rear** pointer.

The diagrammatic representation of the queue is given below:



Difference between Stack and Queue Data Structures are as follows:

| **Stacks** | **Queues** |
| --- | --- |
| Stacks are based on the LIFO principle, i.e., the element inserted at the last, is the first element to come out of the list. | Queues are based on the FIFO principle, i.e., the element inserted at the first, is the first element to come out of the list. |
| Insertion and deletion in stacks takes place only from one end of the list called the top. | Insertion and deletion in queues takes place from the opposite ends of the list. The insertion takes place at the rear of the list and the deletion takes place from the front of the list. |
| Insert operation is called push operation. | Insert operation is called enqueue operation. |
| Delete operation is called pop operation. | Delete operation is called dequeue operation. |
| In stacks we maintain only one pointer to access the list, called the top, which always points to the last element present in the list. | In queues we maintain two pointers to access the list. The front pointer always points to the first element inserted in the list and is still present, and the rear pointer always points to the last inserted element. |
| Stack is used in solving problems works on recursion. | Queue is used in solving problems having sequential processing. |
| Stack does not have any types. | Queue is of three types – 1. Circular Queue 2. Priority queue 3. double-ended queue. |
| Can be considered as a vertical collection visual. | Can be considered as a horizontal collection visual. |