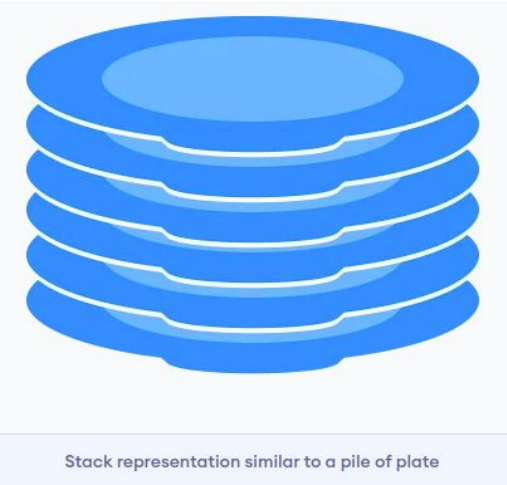
Stack & Queue

Ar. Gör. Elif Ece Erdem

Stack

A stack is a linear data structure that follows the principle of Last In First Out (LIFO). This means the last element inserted inside the stack is removed first.

You can think of the stack data structure as the pile of plates on top of another.

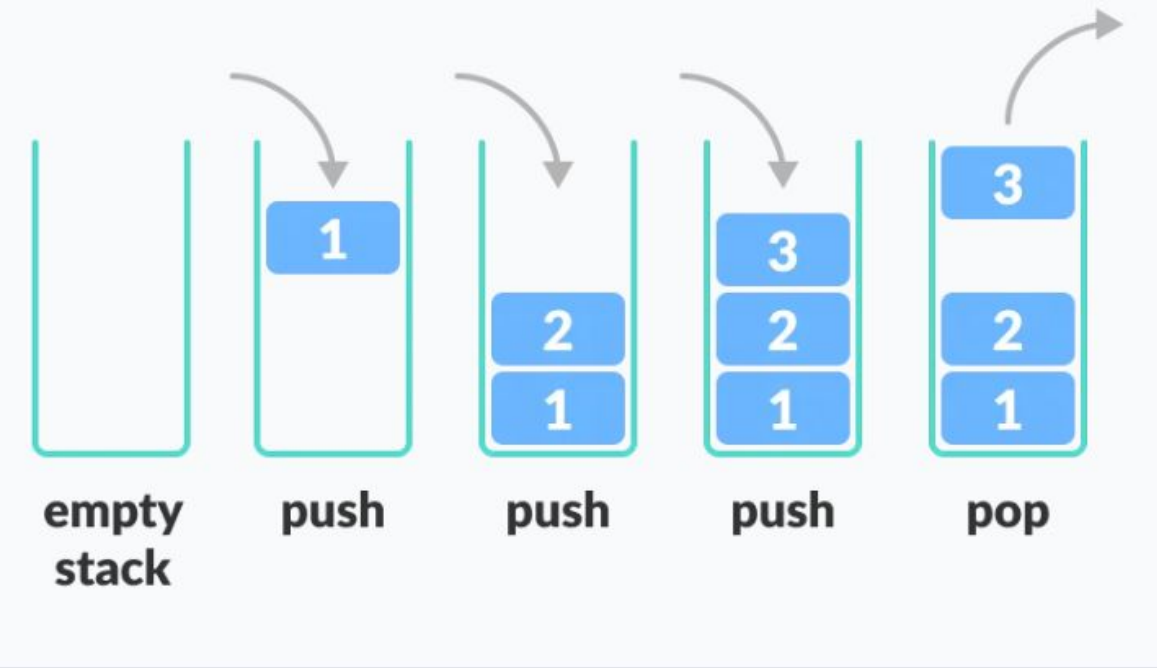


Stack representation similar to a pile of plate

```
// A structure to represent a
stack using linked lists
struct sNode {
    char data;
    struct sNode* next;
};
```

```
// A structure to represent a
stack using arrays
struct Stack {
    int top;
    unsigned capacity;
    int* array;
};
```

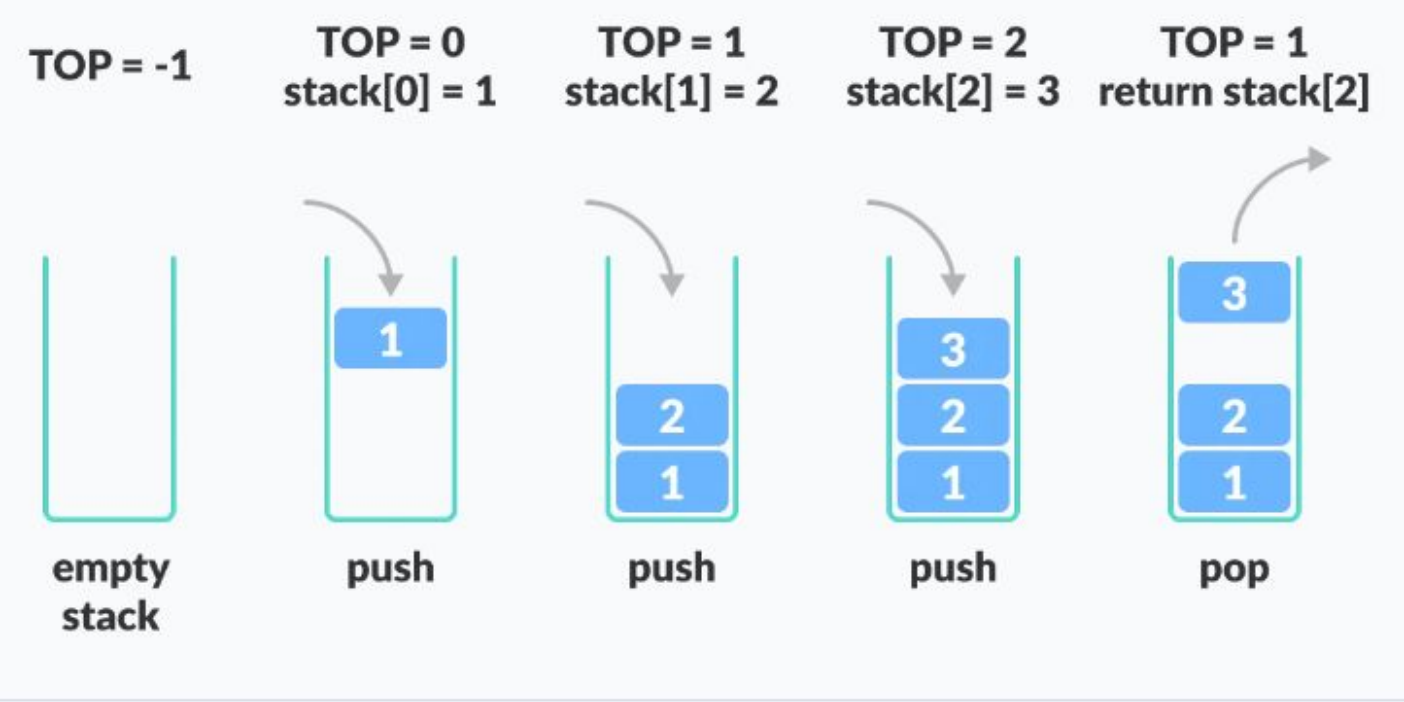
Stack (LIFO)



Stack Push and Pop Operations

- `Push: Add an element to the top of a stack`
- `Pop: Remove an element from the top of a stack`
- `IsEmpty: Check if the stack is empty`
- `IsFull: Check if the stack is full`
- `Peek: Get the value of the top element without removing it`

Stack



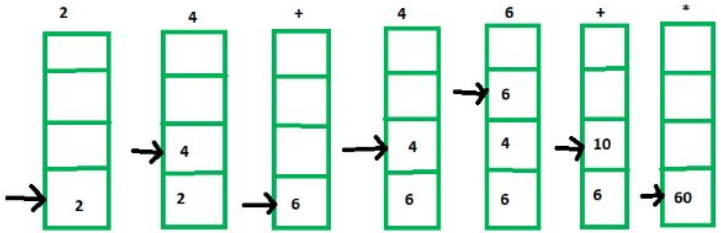
Working of Stack Data Structure

Infix, Prefix, Postfix Notations:

Infix expression: The expression of the form “a operator b” (a + b) i.e., when an operator is in-between every pair of operands.
Prefix expression (Polish Notation): It refers to the notation in which the operator is placed before its two operands. Here no parentheses are required, i.e., +ab
Postfix expression: The expression of the form “a b operator” (ab+) i.e., When every pair of operands is followed by an operator.

```
Infix notation: (2+4) * (4+6)
Post-fix notation: 2 4 + 4 6 + *
Result: 60
```

The stack operations for this expression evaluation is shown below:

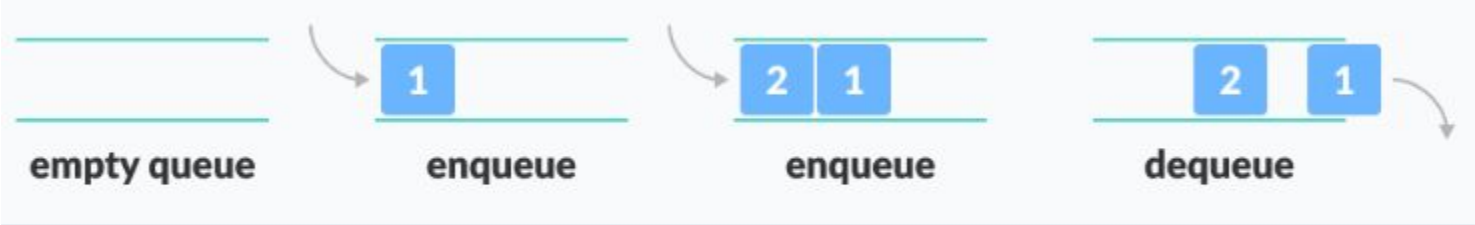


Stack operations to evaluate (2+4)*(4+6)

Queue

A queue is a useful data structure in programming. It is similar to the ticket queue outside a cinema hall, where the first person entering the queue is the first person who gets the ticket.

Queue follows the First In First Out (FIFO) rule - the item that goes in first is the item that comes out first.



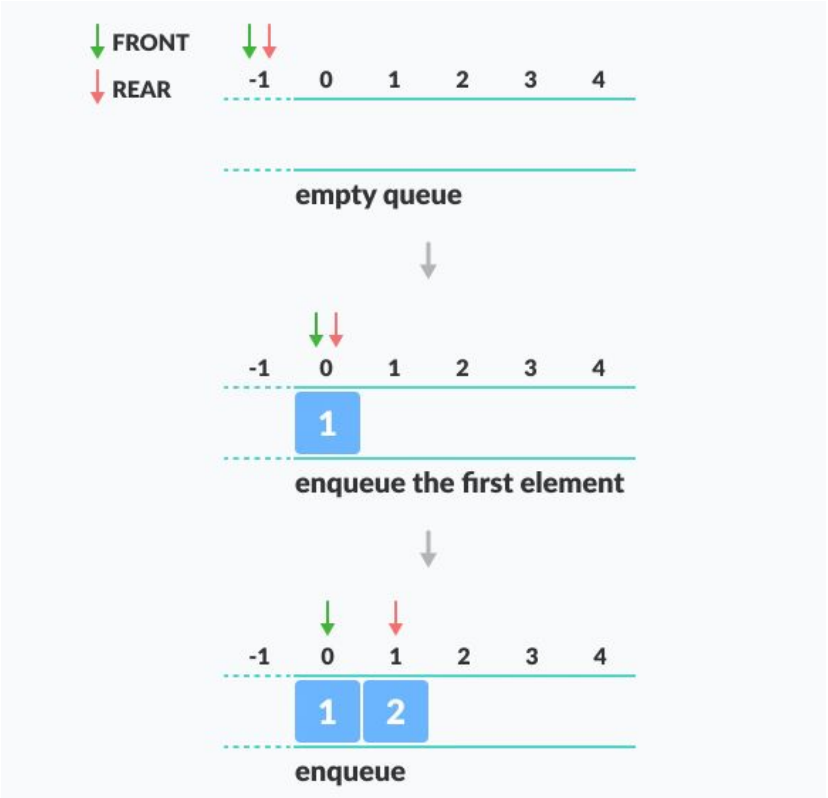
FIFO Representation of Queue

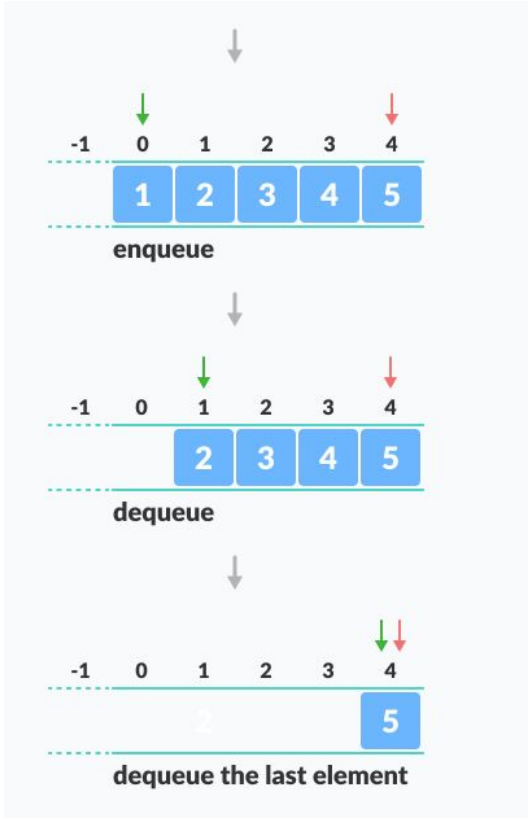
Queue

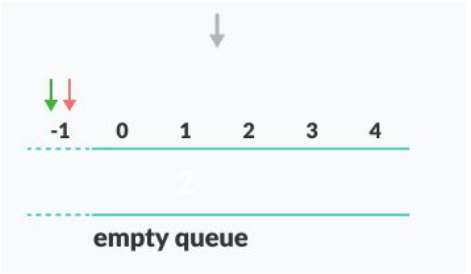
```
// A linked list (LL) node to store
a queue entry
struct QNode {
    int key;
    struct QNode* next;
};

// The queue, front stores the
front node of LL and rear
// stores the last node of LL
struct Queue {
    struct QNode *front, *rear;
};
```

```
// A structure to represent a queue
using array
struct Queue {
    int front, rear, size;
    unsigned capacity;
    int* array;
};
```

Queue


FRONT
REAR
-1 0 1 2 3 4
empty queue
-1 0 1 2 3 4
1
enqueue the first element
-1 0 1 2 3 4
1 2
enqueue


-1 0 1 2 3 4
1 2 3 4 5
enqueue
-1 0 1 2 3 4
2 3 4 5
dequeue
-1 0 1 2 3 4
5
dequeue the last element


-1 0 1 2 3 4
empty queue