FAKULTAS
TEKNOLOGI
INFORMASI
Universitas Advent Indonesia

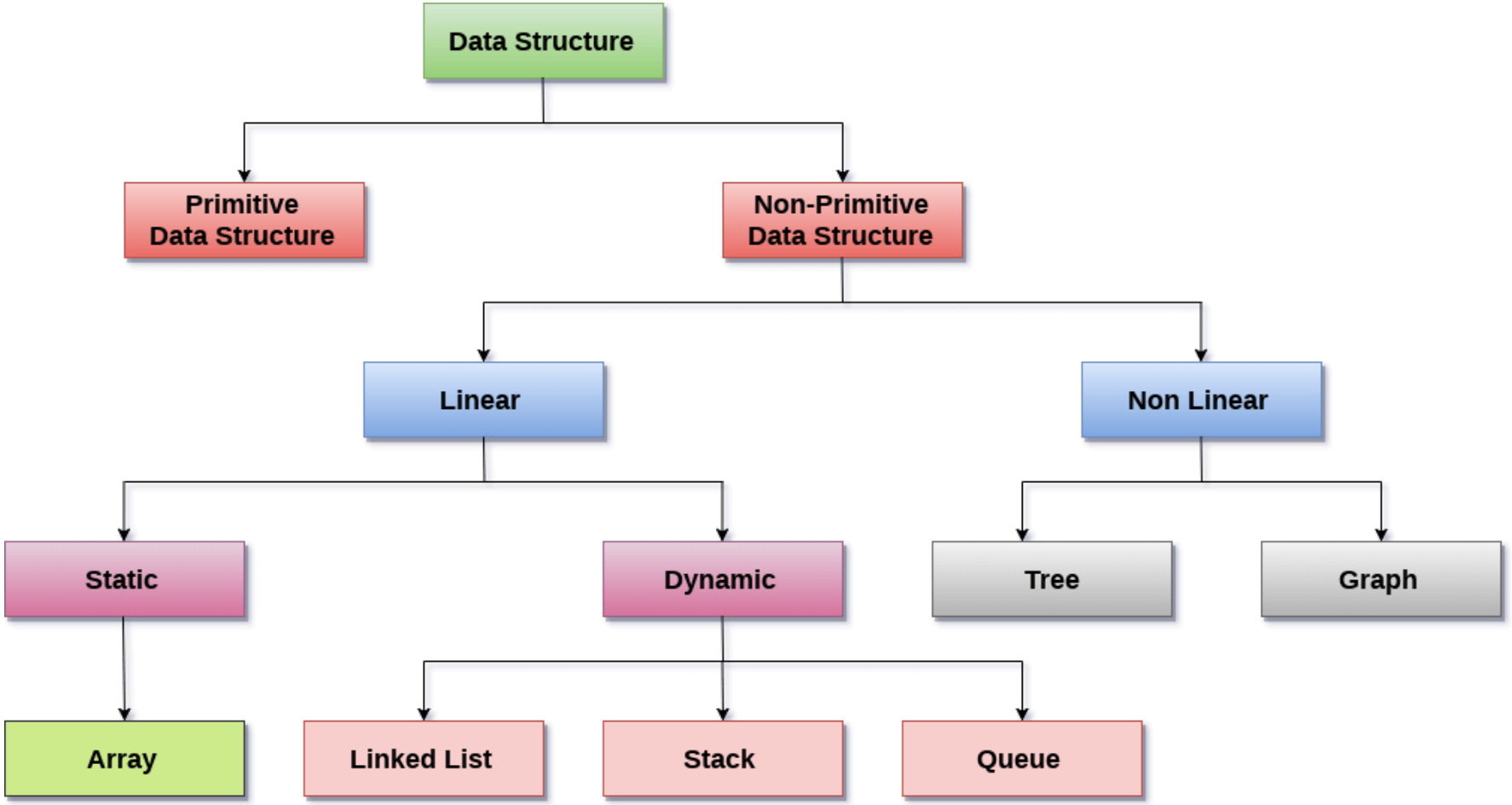
Lecture 07 – ADT Stack

Friday, October 20, 2023

Data Structure

fti.unai.edu

Data Structures Classifications

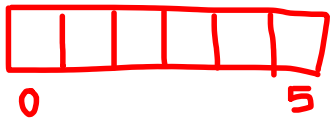


Introduction

- **Abstract Data Types (ADTs)** adalah cara mengklasifikasikan struktur data berdasarkan kegunaannya (bagaimana mereka digunakan) dan perilaku (behaviors) yang diberikan.

- Dalam hal ini, struktur data diklasifikasikan bukan dari bagaimana diimplementasikan namun hanya menyediakan antarmuka minimal yang diharapkan serta dengan perilakunya.

- Pembahasan mengenai struktur data sebagai ADT, berarti kita hanya akan membahas mengenai **fitur** atau **operasi-operasi** yang disediakan oleh struktur data tersebut. → **model logika**

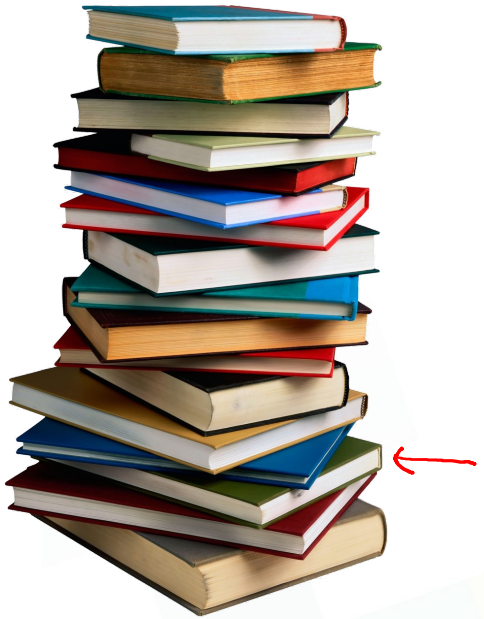


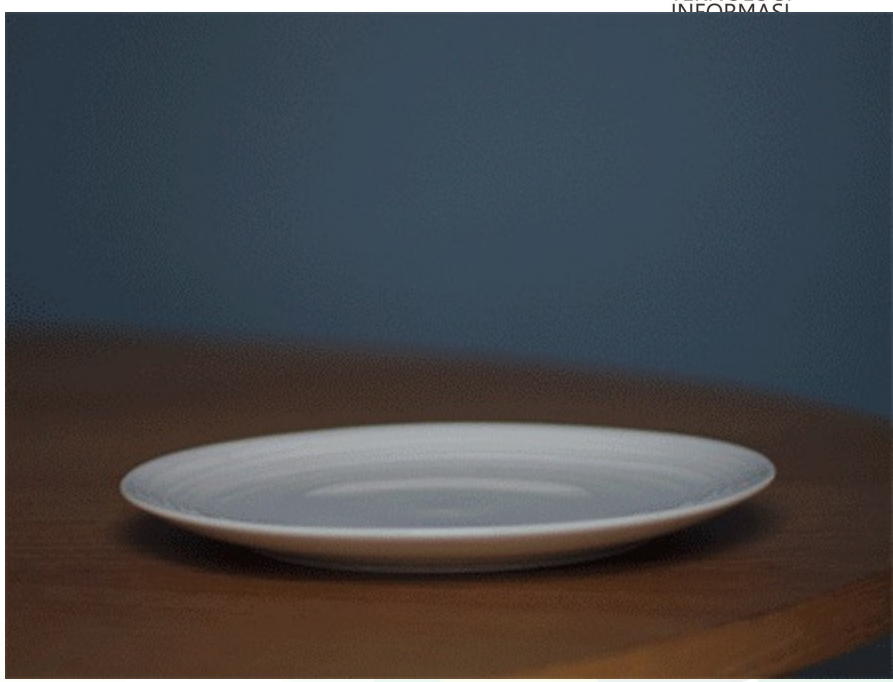
Stack as Abstract Data Types

- **Stack** dapat didefinisikan juga sebagai kumpulan elemen sama seperti Array dan List. Hal yang membedakan stack dari array atau list yakni elemen-elemen di dalam stack hanya dapat dimanipulasi dari satu bagian/ujung. Bagian/ujung ini umumnya dikenal dengan `top of stack`.

- Stack bekerja dengan prinsip **LIFO** (**Last In First Out**). Elemen terakhir yang dimasukkan ke dalam stack akan menjadi elemen pertama yang dikeluarkan.

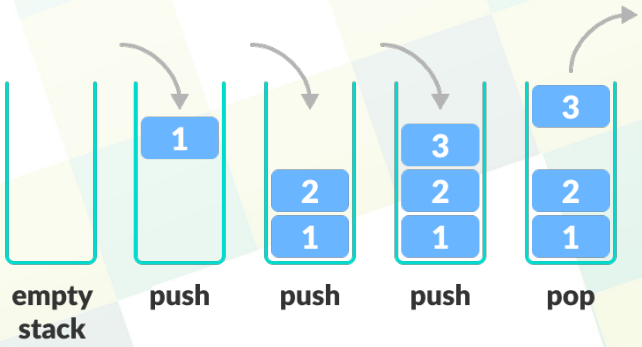
Stack as Abstract Data Types





Stack Operations

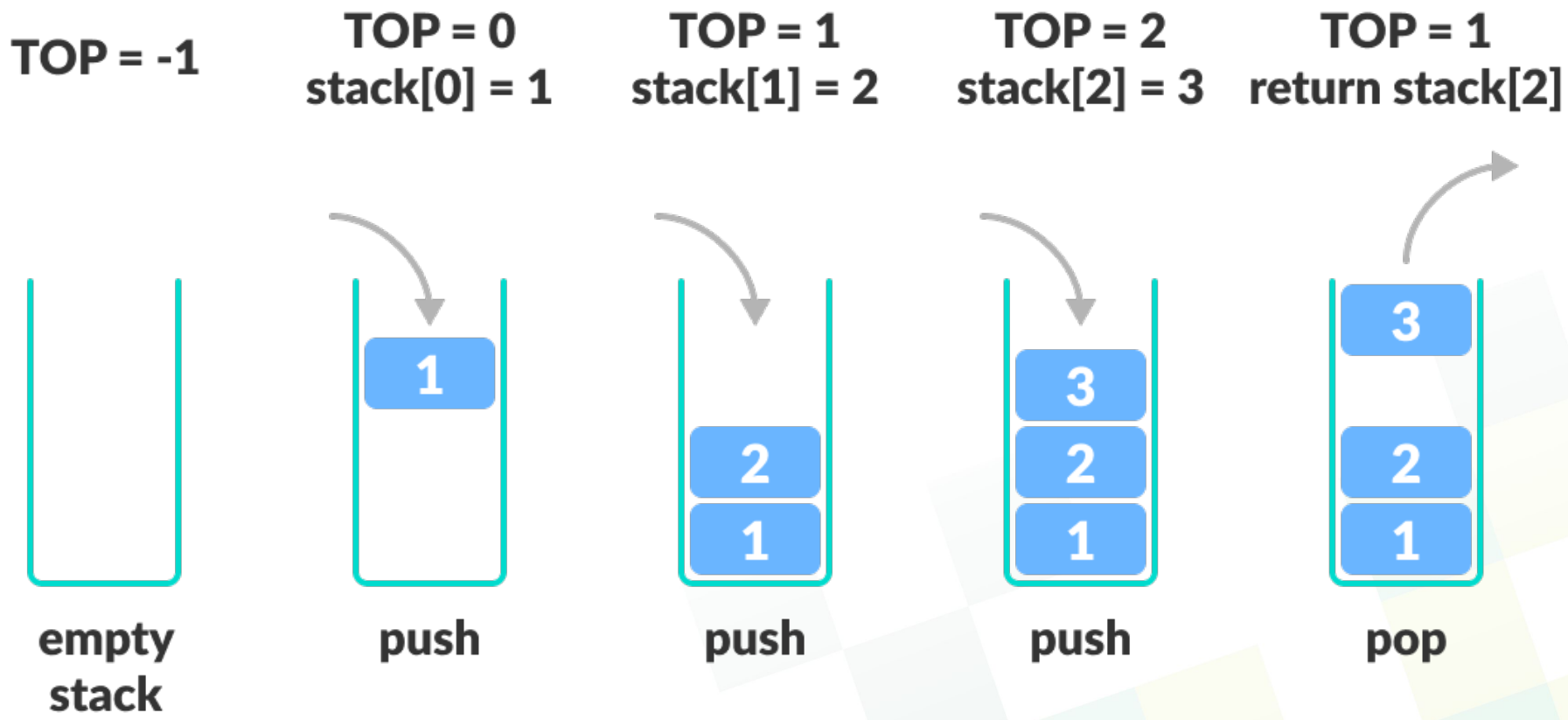
- Dalam pemrograman, istilah yang digunakan untuk meletakkan item di atas sebuah tumpukan dikenal dengan **“push”** dan mengeluarkan/mengambil satu item dari tumpukan dikenal dengan **“pop”**.
- Operasi lainnya yang mungkin dilakukan dengan stack yakni:
  - **IsEmpty** : mencari tahu apakah stack kosong
  - **IsFull** : mencari tahu apakah stack sudah penuh
  - **Peek/Top of Stack** : mengambil nilai dari elemen teratas dari stack tanpa mengeluarkan atau menghapus elemen tersebut.



How Stack works

1. Sebuah variabel, **top**, digunakan untuk melacak elemen teratas dari stack.
2. Inisialisasikan stack dengan mengatur nilai stack menjadi -1 sehingga dapat melakukan pengecekan apakah stack **kosong (isEmpty)** dengan cara membandingkan **top == -1**.
3. Sebelum melakukan push, cek apakah stack **penuh (isFull)**.
4. Ketika elemen dimasukkan (push) ke dalam stack, nilai **top** akan bertambah kemudian elemen baru akan ditempatkan di posisi teratas.
5. Sebelum melakukan pop, cek apakah stack sudah **kosong (isEmpty)**.
6. Ketika mengeluarkan (pop) elemen dari stack, nilai dari elemen teratas (**top)** akan dikembalikan (return) dan nilai **top** akan berkurang.

How Stack works



Stack Implementations

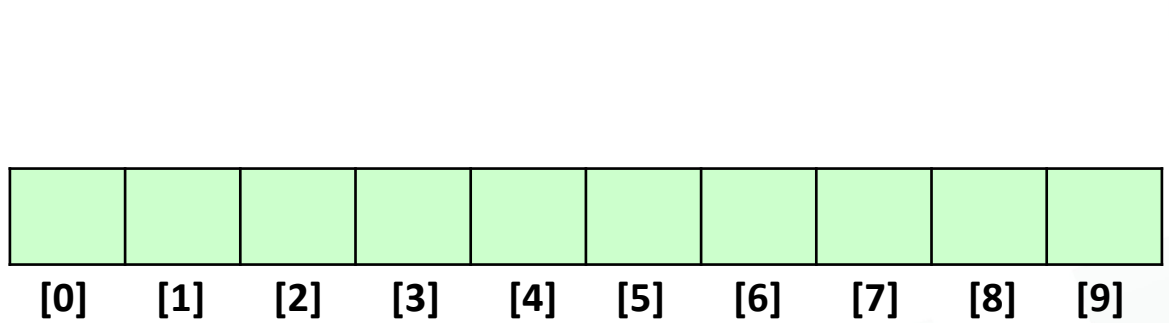
Stack dapat diimplementasikan dengan dua cara:

1. Implementasi menggunakan **Array**
2. Implementasi menggunakan **Linked List**

Stack Implementation using Array

```
int  a[10];
top ← -1 //empty
stack

Push (x)
{
   top ← top + 1
   a[top] ← x
}
```

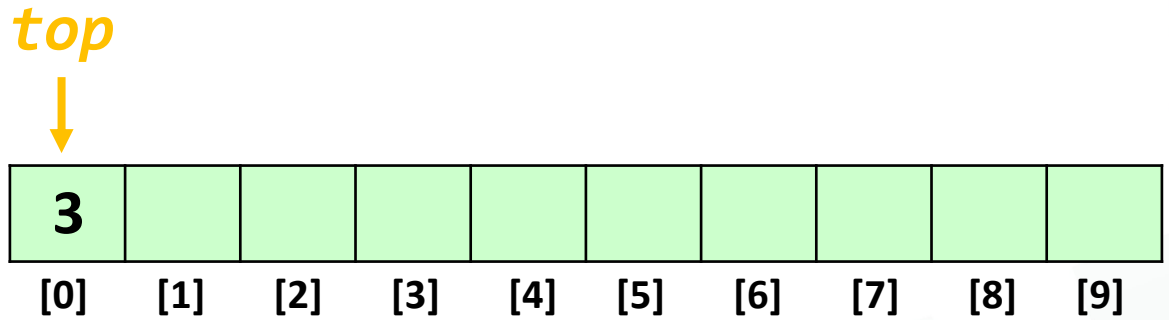


Push (3)

Stack Implementation using Array

```
int  a[10];
top ← -1 //empty
stack

Push (x)
{
   top ← top + 1
   a[top] ← x
}
```

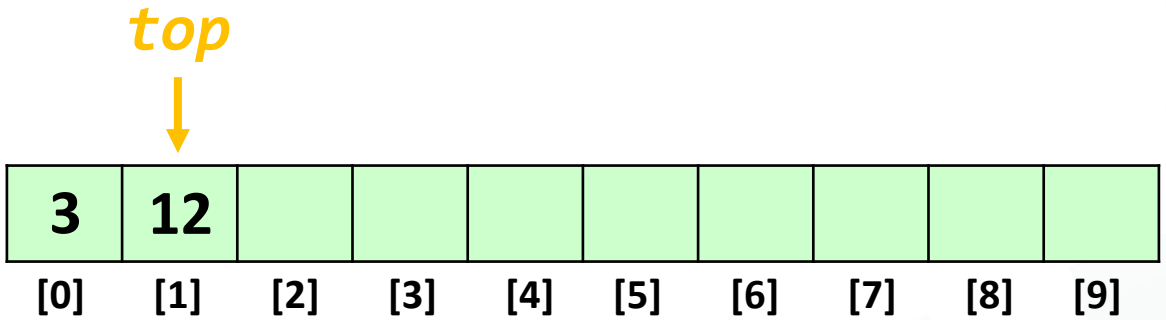


Push (3)

Stack Implementation using Array

```
int  a[10];
top ← -1 //empty
stack

Push (x)
{
   top ← top + 1
   a[top] ← x
}
```

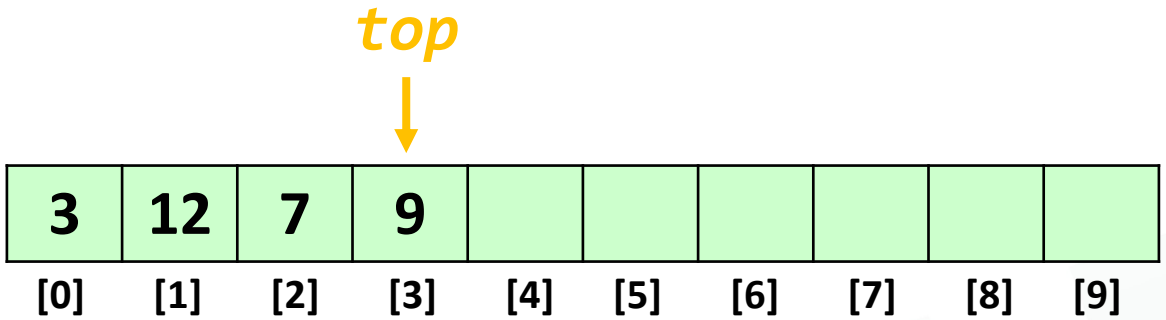


Push (3)

Push (12)

Stack Implementation using Array

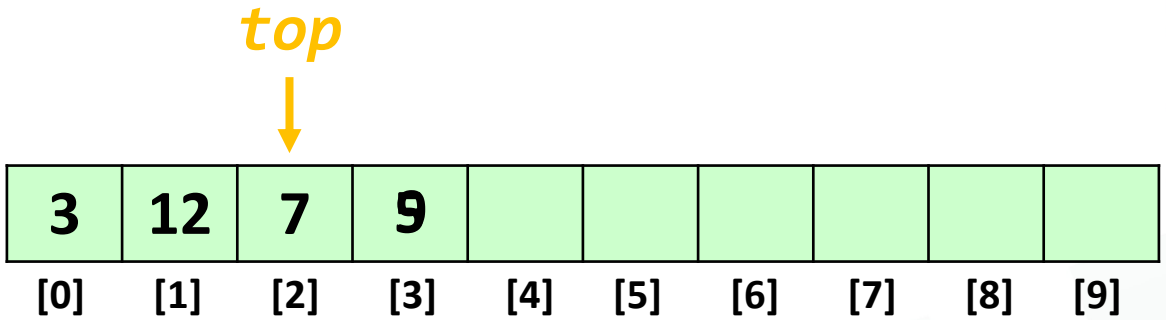
```
int  a[10];
top ← -1 //empty stack
Push (x)
{
   top ← top + 1
   a[top] ← x
}
Pop ()
{
   top ← top – 1
}
```




```
Push (3)
Push (12)
Push (7)
Push (9)
Pop ()
```

Stack Implementation using Array

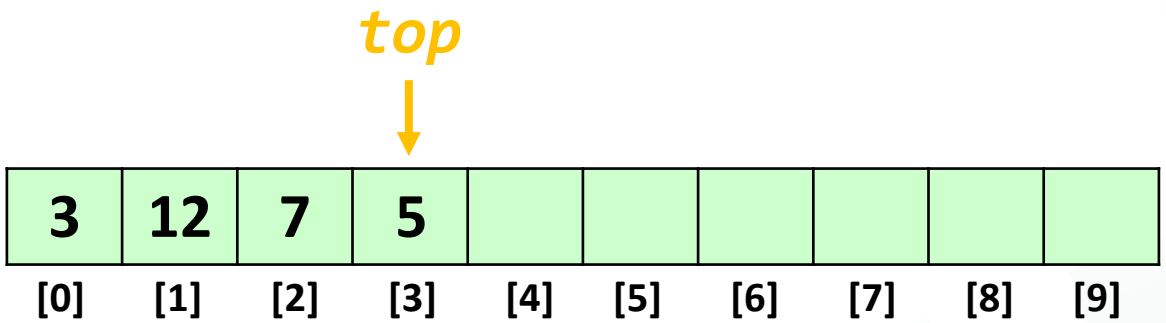
```
int  a[10];
top ← -1 //empty stack
Push (x)
{
   top ← top + 1
   a[top] ← x
}
Pop ()
{
   top ← top – 1
}
```




```
Push (3)
Push (12)
Push (7)
Push (9)
Pop ()
Push (5)
```

Stack Implementation using Array

```
int  a[10];
top ← -1 //empty stack
Push (x)
{
   top ← top + 1
   a[top] ← x
}
Pop ()
{
   top ← top – 1
}
```




```
Top ()
{
   return a[top];
}
isEmpty()
{
   if(top==-1)
      return true;
   else
      return false;
}
```

```
isFull()
{
   if(top==size-1)
      return true;
   else
      return false;
}
```

```
Push (3)
Push (12)
Push (7)
Push (9)
Pop ()
Push (5)
```

Stack Applications

Stack umumnya digunakan :

- Untuk membalik (reverse) kata
- Dalam compiler: untuk melakukan penghitungan untuk ekspresi-ekpresi aritmatika tertentu serta mengecek balanced parentheses.
- Dalam browser: tombol Kembali (back) di browser menyimpan semua URL yang telah user kunjungi sebelumnya dalam stack (tumpukan).
- Undo dalam image editor
- Function calls dan proses rekursif