Stack (Tumpukan)

IF1210 – Algoritma dan Pemrograman 1
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Stack

Stack: list linier yang:

- dikenali elemen puncaknya (TOP),
- aturan penyisipan dan penghapusan elemennya tertentu:
  - Penyisipan selalu dilakukan "di atas" TOP,
  - Penghapusan selalu dilakukan pada TOP.

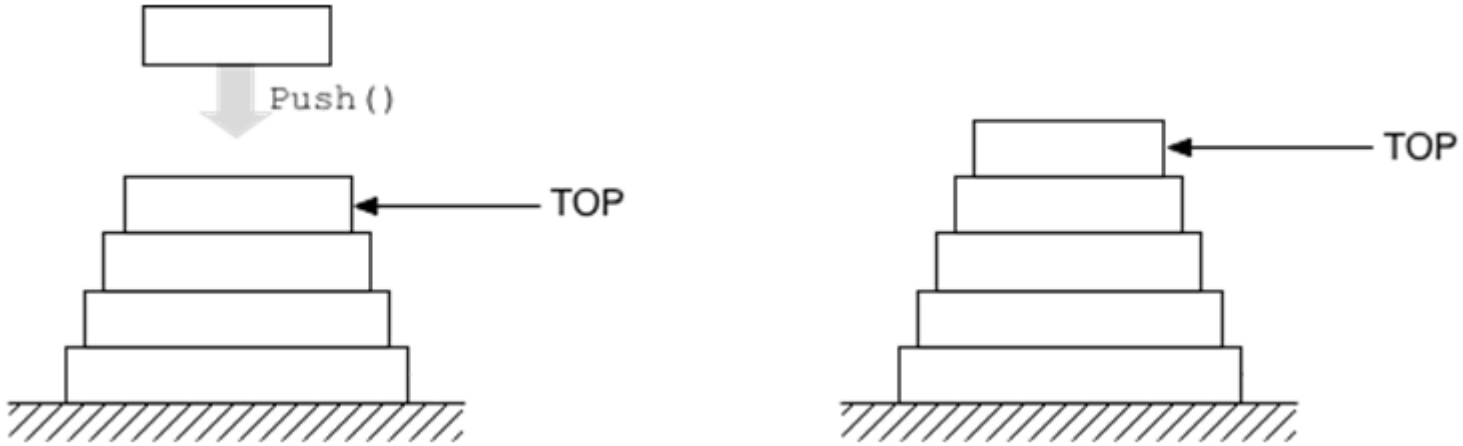
TOP adalah satu-satunya alamat tempat terjadi operasi.

Elemen Stack tersusun secara LIFO (*Last In First Out*).

Designed by lifeforstock / Freepik

Lakeside 8206 Two Stack Plate Dispenser

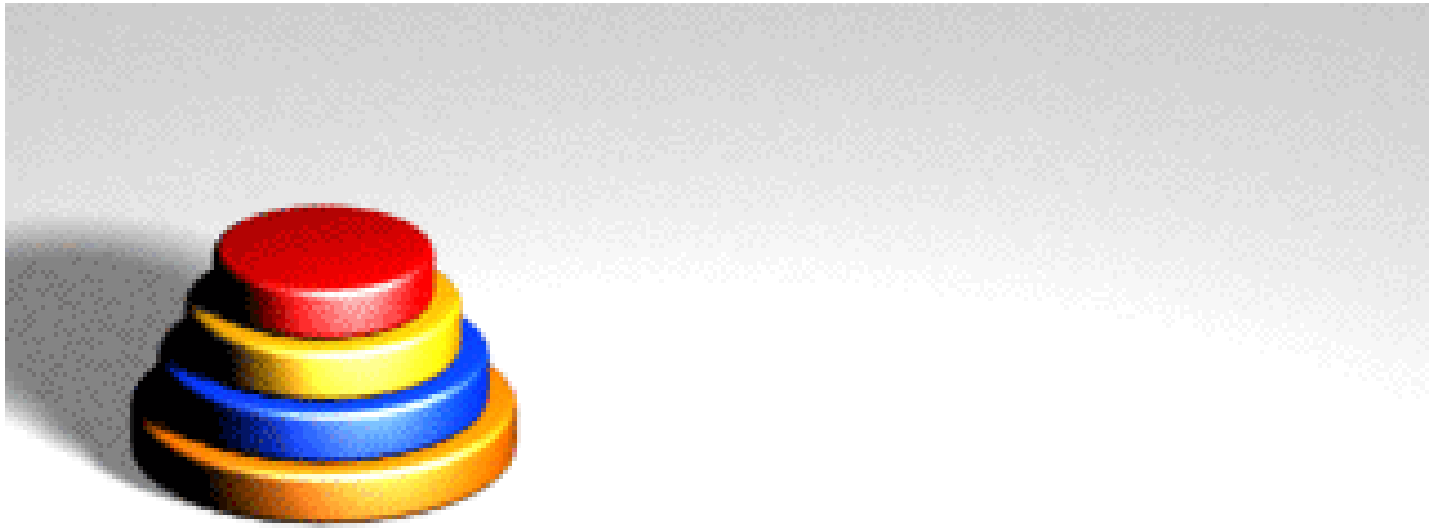
Stack



Stack seperti sebuah List dengan batasan lokasi penambahan & pengurangan elemen.

- Pada **Stack**: operasi penambahan dan pengurangan hanya dilakukan di **salah satu** “ujung” list.
- Pada **List**: operasi **boleh di manapun**.

Tower of Hanoi



https://en.wikipedia.org/wiki/File:Tower_of_Hanoi_4.gif

Pemakaian Stack

- Pemanggilan prosedur
- Perhitungan ekspresi aritmatika
- Rekursivitas
- *Backtracking*

dan algoritma lanjut yang lain

Definisi operasi

Jika diberikan *S* adalah Stack dengan elemen *ElmtS*

CreateStack: $\rightarrow S$	{ Membuat sebuah tumpukan kosong }
top: $S \rightarrow$ *ElmtS*	{ Mengirimkan elemen teratas S saat ini }
length: $S \rightarrow$ *integer*	{ Mengirimkan banyaknya elemen S saat ini }
push: *ElmtS* $\times S \rightarrow S$	{ Menambahkan sebuah elemen ElmtS sebagai TOP, TOP berubah nilainya }
pop: $S \rightarrow S \times$ *ElmtS*	{ Mengambil nilai elemen TOP, sehingga TOP yang baru adalah elemen yang datang sebelum elemen TOP, mungkin S menjadi kosong }
isEmpty: $S \rightarrow$ *boolean*	{ Test stack kosong, true jika S kosong, false jika S tidak kosong }

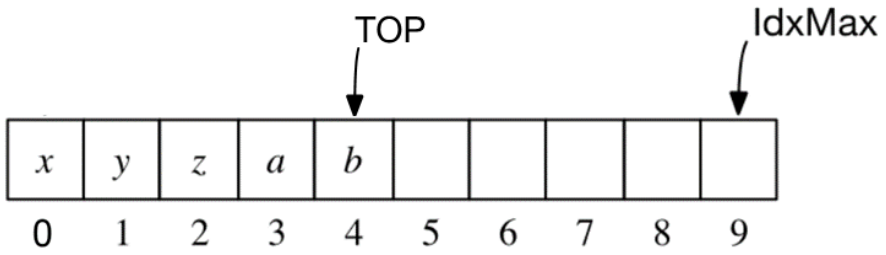
Axiomatic Semantics (fungsional)

1) new() returns a stack
2) pop(push(*v*, *S*)) = *S*
3) top(push(*v*, *S*)) = *v*

Di mana *S* adalah Stack dan *v* adalah value.

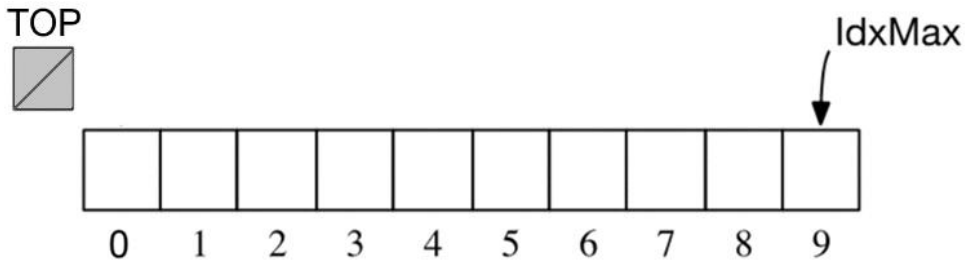
Implementasi Stack dengan List (array)

- Ilustrasi Stack tidak kosong, dengan 5 elemen:



*dengan IdxMax = CAPACITY-1

- Ilustrasi Stack kosong, maka idxTOP diset = IDX_UNDEF.



ADT Stack (dengan array eksplisit-statik)

```
KAMUS UMUM
constant IDX_UNDEF: integer = -1
constant CAPACITY: integer = 10

type ElType: integer { elemen Stack }

{ Stack dengan array statik }
type Stack: < buffer: array [0..CAPACITY-1] of ElType, { penyimpanan elemen }
              idxTop: integer > { indeks elemen teratas }
```

ADT Stack – Konstruktor, akses, & predikat

```
procedure CreateStack(output s: Stack)
{ I.S. Sembarang
  F.S. Membuat sebuah Stack s yang kosong berkapasitas CAPACITY
       jadi indeksnya antara 0..CAPACITY-1
       Ciri Stack kosong: idxTop bernilai IDX_UNDEF }

function top(s: Stack) → ElType
{ Prekondisi: s tidak kosong.
  Mengirim elemen terdepan s, yaitu s.buffer[s.idxTop]. }
function length(s: Stack) → integer
{ Mengirim jumlah elemen s saat ini }

function isEmpty(s: Stack) → boolean
{ Mengirim true jika s kosong: lihat definisi di atas }
function isFull(s: Stack) → boolean
{ Mengirim true jika penyimpanan s penuh }
```

ADT Stack – Operasi

```
{ *** Menambahkan sebuah elemen ke Stack *** }
procedure push(input/output s: Stack, input val: ElType)
{ Menambahkan val sebagai elemen Stack s.
  I.S. s mungkin kosong, tidak penuh
  F.S. val menjadi TOP yang baru, TOP bertambah 1 }

{ *** Menghapus sebuah elemen Stack *** }
procedure pop(input/output s: Stack, output val: ElType)
{ Menghapus X dari Stack S.
  I.S. S tidak mungkin kosong
  F.S. X adalah nilai elemen TOP yang lama, TOP berkurang 1 }
```

ADT Stack (dengan array eksplisit-statik)

```
procedure CreateStack(output s: Stack)
{ I.S. Sembarang
  F.S. Membuat sebuah Stack s yang kosong berkapasitas CAPACITY
       jadi indeksnya antara 0..CAPACITY-1
       Ciri stack kosong: idxTop bernilai IDX_UNDEF }
KAMUS LOKAL
    -
ALGORITMA
    s.idxTop ← IDX_UNDEF
```

ADT Stack (dengan array eksplisit-statik)

```
function isEmpty(s: Stack) → boolean
{ Mengirim true jika Stack kosong: lihat definisi di atas }
KAMUS LOKAL
    -
ALGORITMA
    → s.idxTop = IDX_UNDEF


function isFull(s: Stack) → boolean
{ Mengirim true jika penyimpanan s penuh }
KAMUS LOKAL
    -
ALGORITMA
    → s.idxTop = CAPACITY-1
```

ADT Stack (dengan array eksplisit-statik)

```
procedure push(input/output s: Stack, input val: ElType)
{ (keterangan tidak ditulis untuk menghemat tempat) }
KAMUS LOKAL
    -
ALGORITMA
    s.idxTop ← s.idxTop + 1
    s.buffer[s.idxTop] ← val

procedure pop(input/output s: Stack, output val: ElType)
{ (keterangan tidak ditulis untuk menghemat tempat) }
KAMUS LOKAL
    -
ALGORITMA
    val ← top(s)
    s.idxTop ← s.idxTop - 1
```

Thought exercise

Sama seperti pada Queue, contoh-contoh sebelumnya menggunakan buffer yang terbatas dan statis.

Renungkan perubahan seperti pada kasus Queue:

- buffer menjadi dinamis (*s1*, *s2*: Stack;
  s1 dan *s2* bisa memiliki kapasitas yang berbeda)
- stack tidak boleh memiliki batas length
  (length bisa ∞ secara teoretis)

Apa konsekuensinya terhadap implementasi?
Apa bedanya (jika ada) dengan konsekuensi di ADT Queue?