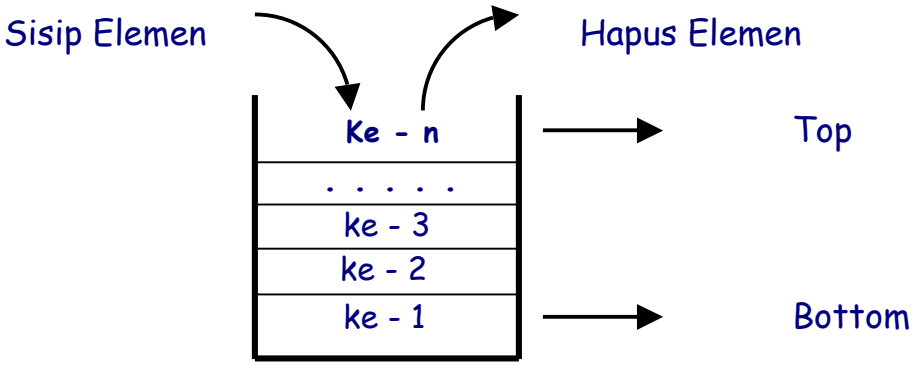
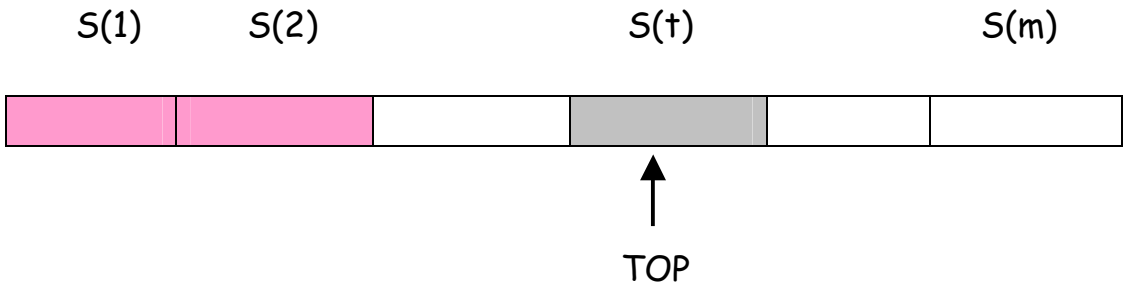
# S T A C K

STACK adalah suatu list yang mempunyai susunan yang terakhir masuk pertama kali keluar. Maka Stack dapat juga disebut LIFO List (last in first out list), dimana elemen yang terakhir masuk (disisipkan) merupakan elemen yang pertama keluar (dihapus).
Jadi sisip dan hapus elemen hanya dapat dilakukan pada satu sisi yg disebut TOP, yaitu pointer sebagai gambaran dari STACK

Ilustrasi dari STACK




Penerapan secara sequential




Ukuran STACK maksimum terdiri dari m elemen, dimana sudah terisi dengan t elemen. Pointer TOP menunjuk ke elemen S[t].

Representasi sequential ini mempunyai sifat statis dimana alokasi storage harus dalam jumlah tetap [m] meskipun ada sebagian saja yang digunakan [t]. Kerugiannya adalah adanya *overflow* dimana penyisipan dilakukan setelah t > m. Sebaliknya kondisi *underflow* terjadi bila dilakukan operasi hapus elemen terhadap STACK yang kosong (t=0).

Operasi dasar pada STACK

Telah disebutkan operasi dasar pada STACK terdiri dari :

1. Sisip elemen (PUSH DOWN)

```
algoritma :   if top = n (maximum) then overflow
                 top := top + 1
                 Stack[Top] := Item
              Return
```

2. Hapus elemen (POP UP)

```
algoritma :   if Top = 0 then Underflow
                 Item := stack[Top]
                 Top := Top – 1
              Return
```

Operasi lainnya pada Stack.

- Create(S)
  adalah operator yang menyebabkan stack menjadi suatu stack hampa.
  NOEL(CREATE(S)) = 0
  TOP(CREATE(S)) = TAK TERDEFINISI

- ISEMPTY(S)
  adalah operator yang mendesak apakah stack S hampa atau tidak.
  Jika S adalah hampa, maka ISEMPTY(S) = TRUE
  NOEL(S) = 0

  Jika S tidak hampa, maka ISEMPTY(S) = False

Contoh :

STACK : A, B , C , D , --- , --- , --- ,---

dialokasikan dalam N = 8 sel memory.

Tentukan susunan STACK setelah dilakukan operasi secara beranting

a. POP (Stack, ITEM)

b. PUSH (Stack, D)

c. POP (Stack, ITEM)

d. PUSH (POP(Stack, ITEM) , L)

e. POP (PUSH(Stack, M) ITEM)

Jawab :

**a. STACK :**
**b. STACK :**
**c. STACK :**
**d. STACK :**
**STACK :**
**e. STACK :**
**STACK :**

## Aplikasi Stack merubah penulisan dari notasi infix ke postfix

Pada bagian ini, akan dibahas ekspresi aritmatika dalam notasi *Infix* dan *Postfix*. Dalam merubah ke bentuk postfix diperlukan STACK, dalam hal ini yg diolah dalam STACK adalah *operatornya* saja.
(Prosedure pengolahan Operatornya pada stack dapat dilihat pada buku DATA STRUCTURES:Seymour Lipschutz, halaman 172 )

Contoh :

1. Infix String : A * B ↑ C - (D * E/F)
   dibentuk ke postfix String.
   Jadi Postfix String : A B C ↑ * D E * F / -

2. Infix String : ((A+B) * C - (D-E)) ↑ (F+G)
   dibentuk ke postfix string
   Jadi Postfix Stringnya : A B + C * D E - - F G + ↑

Catatan :
Hirarki operator :

1. ↑ Eksponensial
2. * dan / , perkalian dan pembagian
3. + dan - , penjumlahan dan pengurangan