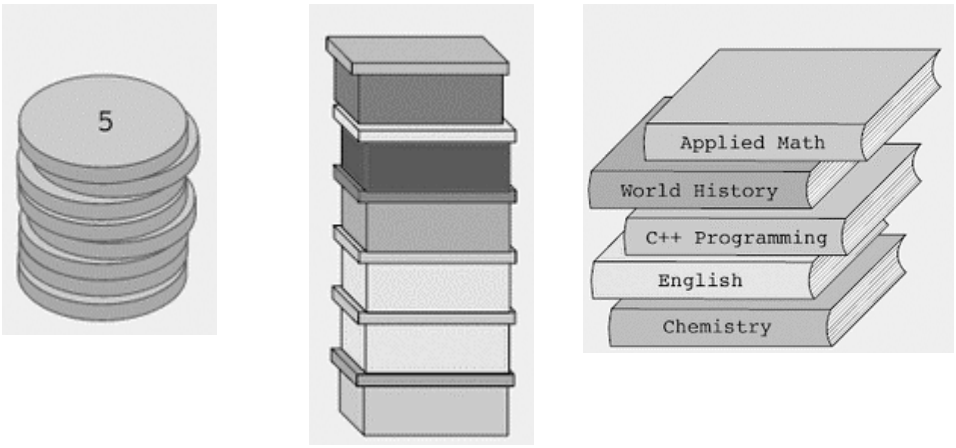
STACK
(TUMPUKAN)

Stack adalah suatu urutan elemen yang elemennya dapat diambil dan ditambah hanya pada posisi akhir (top) saja. Contoh dalam kehidupan sehari-hari adalah tumpukan piring di sebuah restoran yang tumpukannya dapat ditambah pada bagian paling atas dan jika mengambilnya pun dari bagian paling atas pula. Lihat gambar 1.



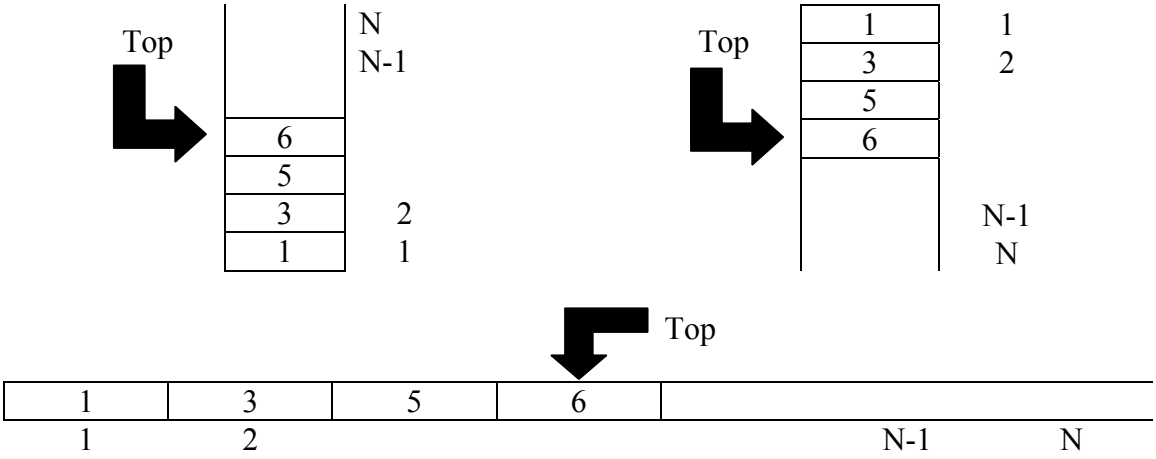
Tumpukan uang koin Tumpukan kotak Tumpukan Buku

Gambar 1. Macam-macam tumpukan

Ada 2 operasi paling dasar dari stack yang dapat dilakukan, yaitu :

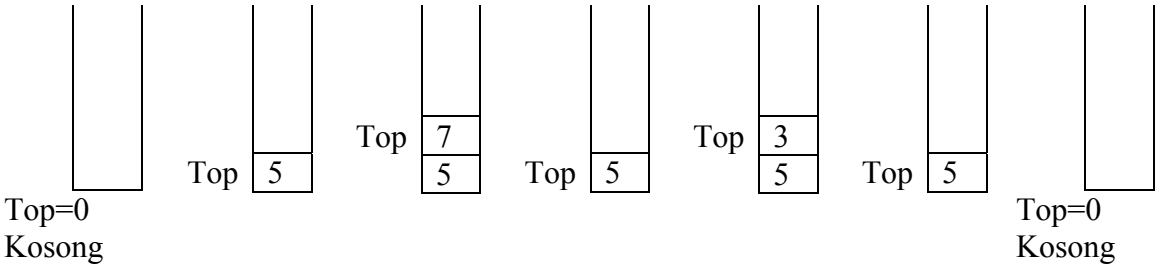
1. Operasi **push** yaitu operasi menambahkan elemen pada urutan terakhir (paling atas).
2. Operasi **pop** yaitu operasi mengambil sebuah elemen data pada urutan terakhir dan menghapus elemen tersebut dari stack.

Sebagai contoh, misalkah ada data sebagai berikut : 1 3 5 6, maka data tersebut dapat tersimpan dalam bentuk sebagai berikut :



Gambar 2. Asumsi-asumsi penyimpanan stack

Contoh lain adalah ada sekumpulan perintah stack yaitu push(5), push(7), pop, push(3), pop,pop. Jika dijalankan, maka yang akan terjadi adalah :



Push(5) Push(7) Pop Push(3) Pop Pop

Gambar 3. Proses operasi stack

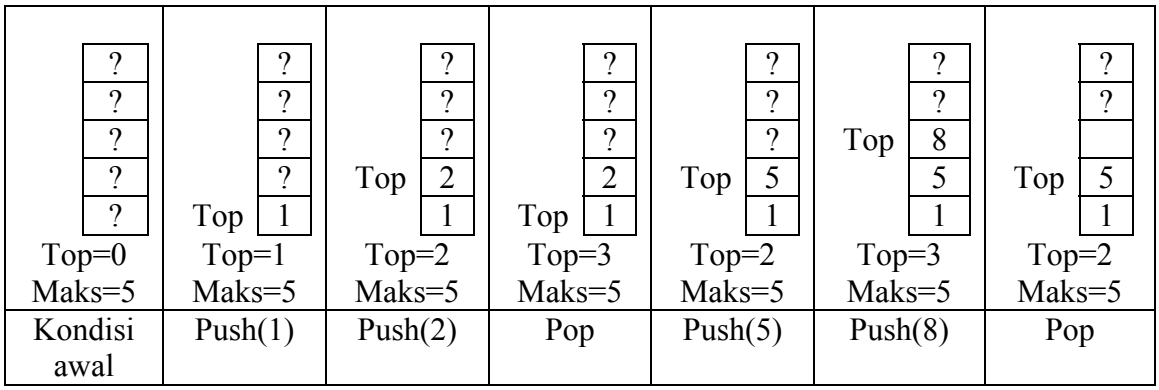
Selain operasi dasar stack (push dan stack), ada lagi operasi lain yang dapat terjadi dalam stack yaitu :

Representasi stack dalam pemrograman, dapat dilakukan dengan 2 cara yaitu :

1. Representasi stack dengan array
2. Representasi stack dengan single linked list

Sebagai contoh representasi kedua cara tersebut dengan operasi yang dilakukan adalah push(1), push(2), pop, push(5), push(8), pos. Untuk lebih detail, perhatikan gambar di bawah ini :

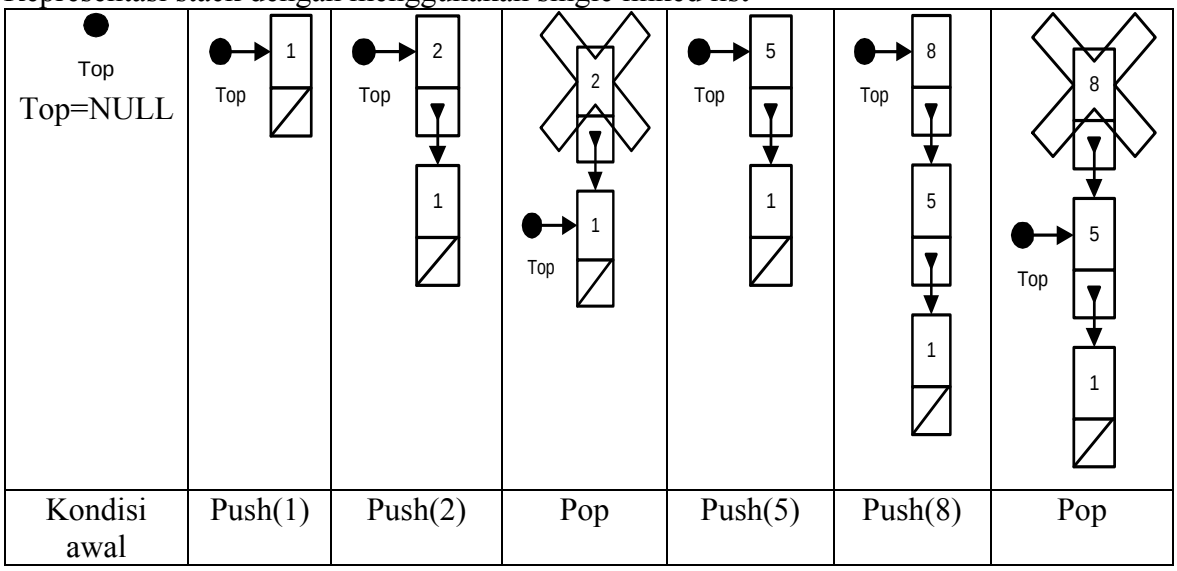
Representasi stack dengan menggunakan array dengan maksimal data 5 adalah



Elemen berisi ? berarti nilai elemen tidak diketahui.

Gambar 4. Representasi stack dengan menggunakan array

Representasi stack dengan menggunakan single linked list



Gambar 5. Representasi stack dengan menggunakan single linked list

Operasi-operasi stack secara lengkap adalah sebagai berikut :

1. Pendeklarasian stack

Proses pendeklarasian stack adalah proses pembuatan struktur stack dalam memori. Karena stack dapat direpresentasikan dalam 2 cara, maka pendeklarasian stack pun ada 2 yaitu :

a. Pendeklarasian stack yang menggunakan array.

```
Const
    Maxstack=……..
Type
     Nama_Stack= array [1..maxstack] of typedata
Stack  : Nama_Stack
Top    : Integer
```

Suatu stack memiliki beberapa bagian yaitu

- **top** yang menunjuk posisi data terakhir (top)
- **elemen** yang berisi data yang ada dalam stack. Bagian ini lah yang berbentuk array.
- **maks_elemen** yaitu variable yang menunjuk maksimal banyaknya elemen dalam stack.

b. Pendeklarasian stack yang menggunakan single linked list

Adapun stack yang menggunakan single linked list, hanya memerlukan suatu pointer yang menunjuk ke data terakhir (perhatikan proses di halaman sebelumnya). Setiap elemen linked list mempunyai 2 field yaitu elemen datanya dan pointer bawah yang menunjuk posisi terakhir sebelum proses push.

```
Type
    Nama_pointer      = ↑simpul
    Simpul            = Record
        <info   : typedata,
         Next   : Nama_pointer>
     Endrecord

  Var_pointer  : Nama_pointer
```

2. Operasi Push

Operasi **push** adalah operasi dasar dari stack. Operasi ini berguna untuk menambah suatu elemen data baru pada stack dan disimpan pada posisi top yang akan mengakibatkan posisi top akan berubah. Langkah operasi ini adalah :

a. Operasi push pada stack yang menggunakan array.

Langkah operasi push dalam array adalah dengan :

- Stack dapat ditambah jika stack belum penuh
- Proses push-nya sendiri adalah dengan menambah field top dengan 1, kemudian elemen pada posisi top diisi dengan elemen data baru.

b. Operasi push pada stack yang menggunakan single linked list

Operasi push pada stack yang menggunakan single linked list adalah sama dengan proses tambahawal pada operasi linked list. Langkah-langkahnya adalah :

- Proses push-nya sendiri adalah dengan cara mengalokasikan suatu elemen linked list (disebut variable baru) dan memeriksa apakah stack kosong / tidak (lihat gambar 5 halaman 3)

3. Operasi Pop

Operasi pop adalah salah satu operasi paling dasar dari stack. Operasi ini berguna untuk mengambil elemen terakhir (top) dan kemudian menghapus elemen tersebut

sehingga posisi top akan berpindah. Operasi ini biasanya dibuat dalam bentuk function yang me-return-kan nilai sesuai data yang ada di top.

a. Operasi pop pada stack yang menggunakan array.

   Langkah operasi pop pada stack yang menggunakan array adalah

   - Stack dapat mengeluarkan elemennya jika stack tidak kosong
   - Elemen yang dikeluarkan disimpan pada suatu variable
   - Nilai top berkurang 1

b. Operasi pop pada stack yang menggunakan single linked list

   Langkah operasi pop pada stack yang menggunakan single linked list adalah sama dengan proses hapusawal pada operasi single linked list. Prosesnya adalah :

   - Periksa apakah.stack kosong (isempty), jika kosong maka proses pop tidak bisa dilakukan. Jika stack tidak kosong maka proses pop dijalankan.
   - Proses pop-nya sendiri adalah mengambil elemen yang ditunjuk oleh pointer stack kemudian simpan dalam variable data. Kemudian buat variable pointer **bantu** yang diisi dengan pointer penunjuk stack yang nantinya akan dihapus dari memori. Kemudian pointer penunjuk stack dipindahkan ke posisi yang ditunjuk oleh field pointer bawah dari variable **bantu**.

Contoh Implementasi Stack

Notasi Polish (Polish Notation)

Dalam operasi aritmatika, dikenal 3 jenis notasi, yaitu :

1. Notasi Infix (menempatkan operator di antara 2 operand)
   Contoh : A+B atau C-D atau E * F atau G / H
2. Notasi Prefix (menempatkan operator di depan / sebelum ke-2 operandnya)
   Contoh : +AB atau –CD atau *EF atau /GH
3. Notasi Postfix (menempatkan operator di belakang/setelah operandnya)
   Contoh : AB+ atau CD- atau EF* atau GH/

Komputer umumnya hanya mengenal ekspresi matematika yang ditulis dalam notasi postfix. Ekspresi matematika yang ditulis dalam notasi infix agar dikenal oleh komputer harus diubah dengan memperhatikan :

1. Mengubah notasi infix menjadi postfix, kemudian menghitungnya
2. Menggunakan stack sebagai penampung sementara operator dan operandnya.

1. Mengubah Notasi Infix menjadi Notasi Postfix

Dimisalkan Q adalah ekspresi matematika yang ditulis dalam notasi infix dan P adalah penampung ekspresi matematika dalam notasi postfix, maka algoritmanya adalah :

a. Push tanda "**(**" ke stack dan tambahkan tanda "**)**" di sentinel di **Q**.
b. Scan **Q** dari kiri ke kanan, kemudian ulangi langkah **c** s.d **f** untuk setiap elemen **Q** sampai stack **Q** kosong.
c. Jika yang discan adalah operand, maka tambahkan ke P
d. Jika yang discan adalah "**(**" maka push ke stack
e. Jika yang discan adalah "**)**" maka pop isi stack sampai ditemukan tanda "**(**", kemudian tambahkan ke P sedangkan tanda "**(**" tidak disertakanke P.

f. Jika yang discan adalah operator, maka :
   - Jika elemen paling atas dari stack adalah operator yang mempunyai tingkatan sama atau lebih tinggi dari operator yang discan, maka pop operator tersebut dan tambahkan ke P.
   - Push operator tersebut ke stack.

g. Keluar

Contoh kasus :

Q : A + (B * C - (D / E ^ F) * G) * H

Tambahkan "**(**" ke stack dan tambahkan tanda "**)**" ke sentinel Q sehingga Q menjadi

Q : A + (B * C - (D / E ^ F) * G) * H)

Dari Q, terdapat 20 simbol yaitu :

Q :	**A**	**+**	**(**	**B**	*****	**C**	**-**	**(**	**D**	**/**	**E**	**^**	**F**	**)**	*****	**G**	**)**	*****	**H**	**)**
No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Urutan operasinya adalah :

No	Simbol	Stack	Ekspresi P
		(	
1	**A**	**(**	**A**
2	**+**	**(+**	**A**
3	**(**	**(+(**	**A**
4	**B**	**(+(**	**AB**
5	*****	**(+(***	**AB**
6	**C**	**(+(***	**ABC**
7	**-**	**(+(-**	**ABC***
8	**(**	**(+(-(**	**ABC***
9	**D**	**(+(-(**	**ABC*D**
10	**/**	**(+(-(/**	**ABC*D**
11	**E**	**(+(-(/**	**ABC*DE**
12	**^**	**(+(-(/^**	**ABC*DE**
13	**F**	**(+(-(/^**	**ABC*DEF**
14	**)**	**(+(-**	**ABC*DEF^/**
15	*****	**(+(-***	**ABC*DEF^/**
16	**G**	**(+(-***	**ABC*DEF^/G**
17	**)**	**(+**	**ABC*DEF^/G*-**
18	*****	**(+***	**ABC*DEF^/G*-**
19	**H**	**(+***	**ABC*DEF^/G*-H**
20	**)**		**ABC*DEF^/G*-H*+**

Dari proses di atas didapatkan bahwa postfixnya adalah **ABC*DEF^/G*-H*+**

2. Menghitung Ekspresi Matematika yang disusun dalam Notasi Postfix

Diasumsikan P adalah ekspresi matematika yang ditulis dalam notasi postfix dan variable **value** sebagai penampung hasil akhir.

Algoritmanya adalah :

a. Tambahkan tanda ")" pada sentinel di P
b. Scan P dari kiri ke kanan, ulangi langkah **c** dan **d** untuk setiap elemen P sampai ditemukan sentinel.
c. Jika yang discan adalah operand, maka push ke stack.
d. Jika yang discan adalah operator (sebut **opr1**), maka
   - Pop 1 buah elemen teratas dari stack, simpan dalam variable **var1**.
   - Pop 1 buah elemen teratas dari stack, simpan dalam variable **var2**.
   - Hitung variable (**var2 opr1 var1**), simpan hasil di variable **hitung**.
   - Push variable **hitung** ke stack.
e. Pop isi stack dan simpan di variable **value**.
f. Keluar.

Contoh : P : 5, 2, 6, +, *, 12, 4, /, -

Tambahkan tanda "**)**"pada sentinel P sehingga

P : 5, 2, 6, +, *, 12, 4, /, -,)

Dari P, didapatkan 10 simbol yaitu :

P :	**5**	**2**	**6**	**+**	*****	**12**	**4**	**/**	**-**	**)**
No	1	2	3	4	5	6	7	8	9	10

Urutan operasinya adalah :

No	Simbol	Stack	Operasi Perhitungan
1	**5**	**5**	
2	**2**	**5, 2**	
3	**6**	**5, 2, 6**	
4	**+**	**5, 8**	**var1=6,var2=2,hitung=2 + 6 =8**
5	*****	**40**	**var1=8,var2=5,hitung=5 * 8 =40**
6	**12**	**40, 12**	
7	**4**	**40, 12, 4**	
8	**/**	**40, 3**	**var1=4,var2=12,hitung=12 / 4 = 3**
9	**-**	**37**	**Var1=3,var2=40,hitung=40 – 3 = 37**
10	**)**	**Perulangan selesai karena telah mencapai sentinel dan data dalam stack adalah 37 (hasil akhir).**	

Jadi hasil operasi tersebut adalah : 37