PEMOGRAMAN WEB II

Program Studi PJJ INFORMATIKA

Sesi 5 – STACK

CATUR NUGROHO, S.KOM., M.KOM

STACK

Mahasiswa mampu :

Mahasiswa mampu menguasai struktur data array dan algoritma untuk tumpukan (*stack*)

STACK

Stacks, *queues, deques* dan *lists* merupakan contoh koleksi data yang item-itemmnya terurut (*linear*), bergantung pada bagaimana item tersebut ditambahkan atau dihapus. Begitu suatu item ditambahkan, ia mendiami posisi relatif terhadap elemen lain yang hadir sebelum dan setelahnya. Koleksi demikian dinamakan struktu data linier. Struktur linier mempunyai dua ujung, kadang dinamakan “kiri” dan “kanan”, “depan” dan “belakang”, juga “top” dan “base.” .

STACK

Stack (disebut pula "*push-down stack*") adalah koleksi berurut dari item-item dimana penambahan item baru dan penghapusan item yang telah ada selalu terjadi di ujung yang sama. Ujung ini dinamakan sebagai "*top*." Ujung berlawanan dari top dikenal sebagai "*base*."

Stack diurutkan mengikuti konsep LIFO (last in first out).
LIFO (Last In First Out) : elemen yang terakhir kali masuk akan menjadi elemen yang pertama kali keluar.

KONSEP STACK

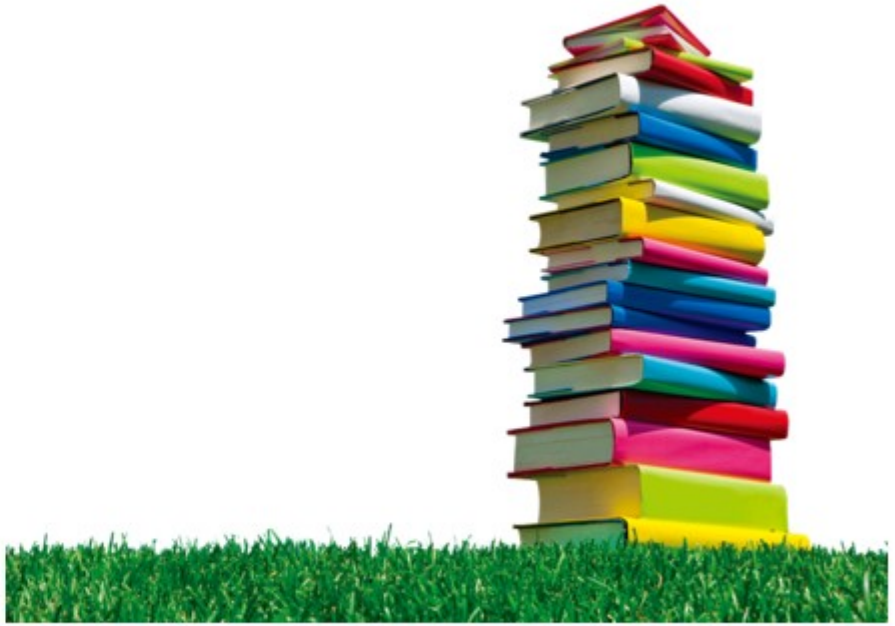
- **Analogi:**
  - tumpukan piring di kafetaria,
  - tumpukan koin yang harus dibayar, atau
  - tumpukan kotak,
  - tumpukan kemeja yang terlipat rapi
- **Menambahkan item**
  - Disebut sebagai mendorongnya ke tumpukan
- **Menghapus item**
  - Disebut sebagai meletuskannya dari tumpukan
- Stack dapat diimplementasikan menggunakan *array* atau *linked list*

KONSEP STACK

"Benda yang terakhir masuk ke dalam stack akan menjadi yang pertama keluar dari stack

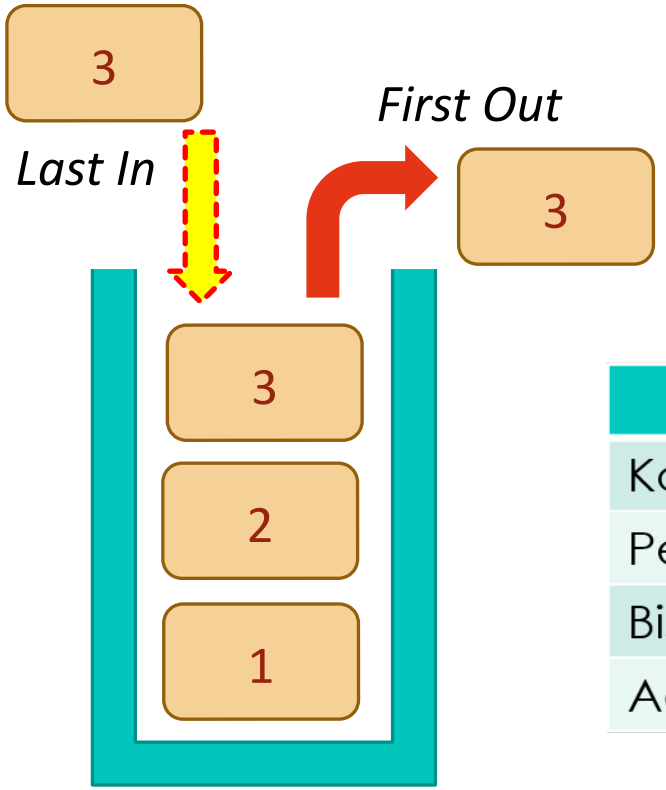
Tumpukan uang koin

Tumpukan kotak



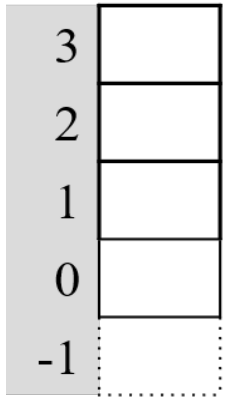
Tumpukan Buku

KONSEP STACK



Kondisi Stack dapat diketahui dari posisi (isi) Top-nya.

Kondisi Stack	Ciri (Posisi Top)
Kosong, tidak ada isinya.	Top = -1
Penuh, tidak bisa diisi lagi.	Top = n-1
Bisa diisi, lawan dari penuh.	Top < n-1
Ada isinya, lawan dari kosong.	Top > -1

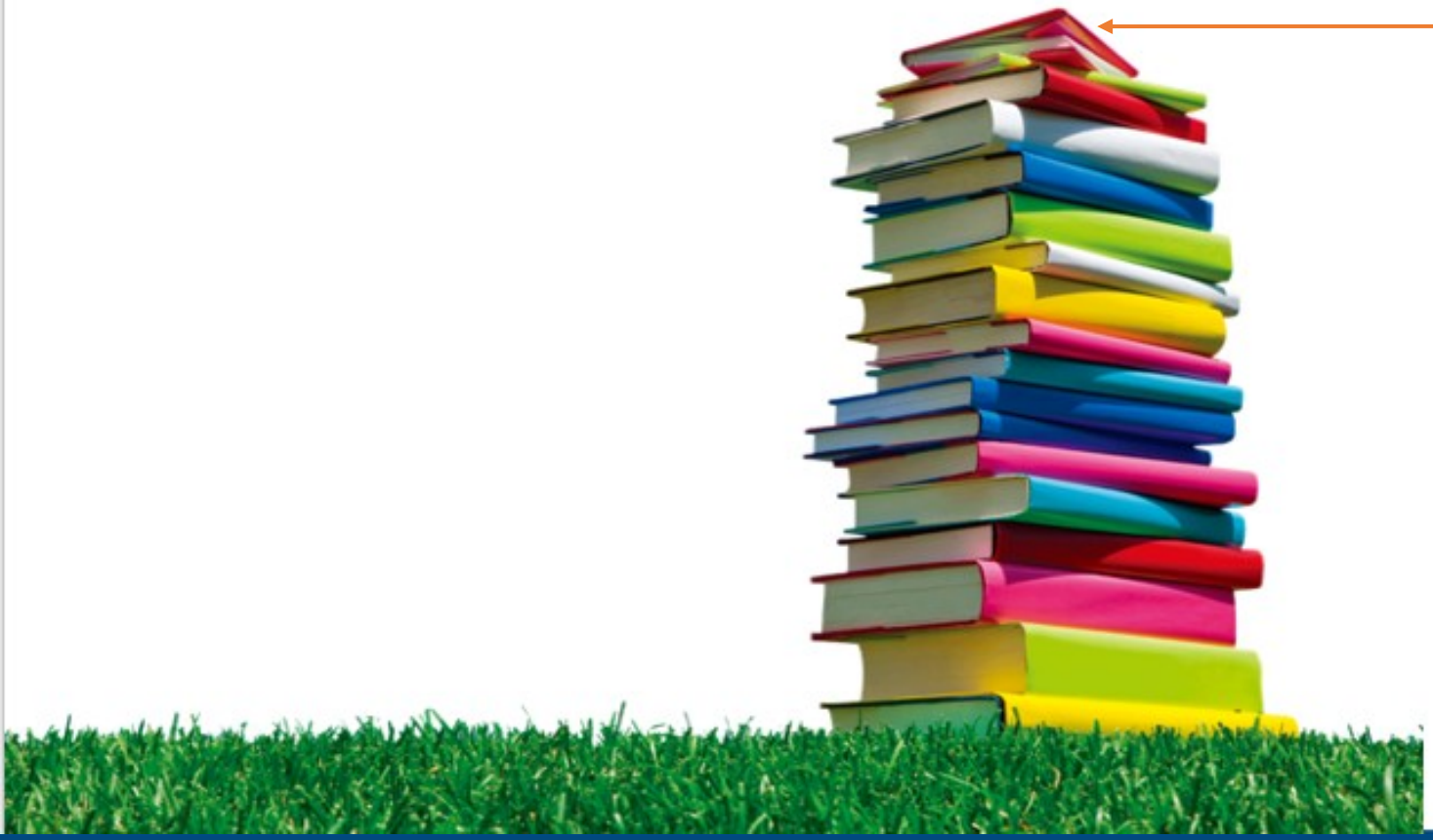


OPERASI PADA STACK

No	Operasi	Penjelasan
1	Push	Menambah data pada Stack
2	Pop	Mengambil data pada Stack
3	Top	Data paling atas pada Stack
4	Clear	Mengosongkan data pada Stack
5	IsFull	Periksa apakah data pada Stack sudah penuh
6	IsEmpty	Periksa apakah data pada Stack sudah kosong
7	Print	Mencetak data Stack

- Kondisi Stack, apakah kosong, penuh, bisa diisi atau ada isinya, ditentukan oleh posisi Top.
- Setiap statement diakhiri dengan character '**\0**' (NULL).
- Pop dan push sama-sama dilakukan pada item yang terakhir kali ditambahkan pada stack.

KONSEP STACK



Pointer TOP : digunakan untuk menunjuk
element paling akhir yang dimasukkan
kedalam stack.

KONSEP STACK

Jika top tidak menunjuk pada element manapun hal ini menunjukkan bahwa stack dalam kondisi kosong (*empty*).

Pada array : top akan menyimpan index element paling akhir masuk.
Pada linked list : top akan menyimpan alamat node yang paling akhir ditambahkan.

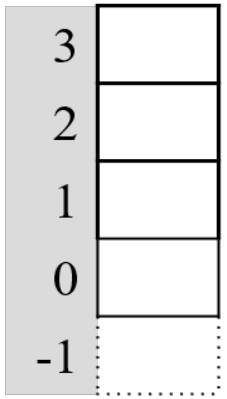
Representasi Array dari Stacks

- Stack memiliki dua variabel:
  - TOP yang digunakan untuk menyimpan alamat elemen paling atas dari tumpukan.
  - MAX yang digunakan untuk menyimpan jumlah maksimum elemen yang dapat ditampung oleh stack

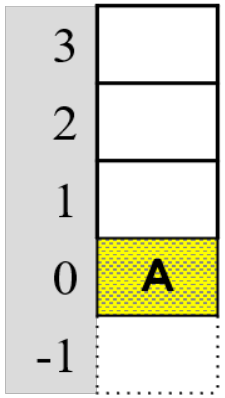
- IF TOP = NULL, maka menunjukkan bahwa stack kosong
- IF TOP = MAX - 1, maka tumpukan sudah penuh
- TOP = 4, penyisipan dan penghapusan akan dilakukan pada posisi ini.

Ilustrasi PADA STACK

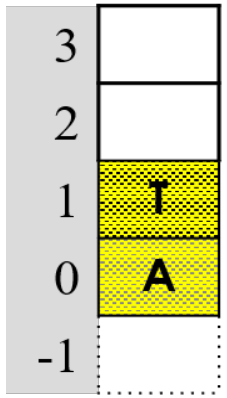
1) Stack kosong: Top = -1



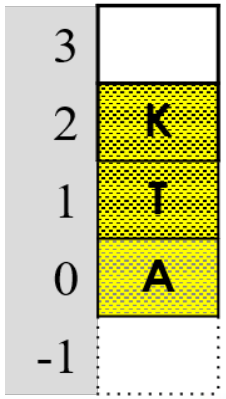
2) Push ‘A’: Top = 0



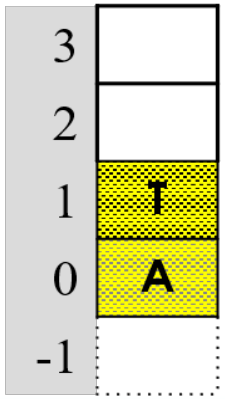
3) Push ‘T’: Top = 1



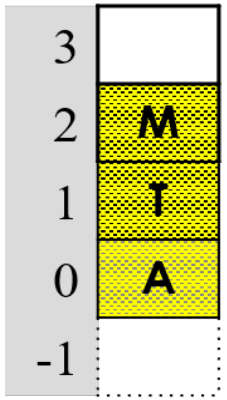
4) Push ‘K’: Top = 2



5) Pop : Top = 1



6) Push ‘M’: Top = 2



UNGKAPAN ARITMATIKA

1. **Prefix** adalah metode penulisan dengan meletakkan *operator di depan operand* dan tanpa menuliskan tanda kurung.
   Contoh : +AB, -+ABC, *+AB – CD.

2. **Infix** adalah metode penulisan dengan meletakkan *operator di antara dua operand*. Pemakaian tanda kurung sangat menentukan hasil operasi.
   Contoh : A+B, A+B-C, (A+B)*(C-D)

3. **Postfix** adalah metode penulisan dengan meletakkan *operator setelah operand* dan tanpa menuliskan tanda kurung.
   Contoh : AB+

KETENTUAN KONVERSI

Derajat Operator:

No	Notasi Operator	Fungsi (penjelasan)
1	(……)	Tanda kurung
2	^	Pangkat
3	* atau /	Kali atau Bagi
4	+ atau -	Tambah atau Kurang

Contoh:

Prefix	Infix	Postfix
+AB	A+B	AB+
++ABC	A+B+C	AB+C+
+A*BC	A+B*C	ABC*+
*+ABC	(A+B)*C	AB+C*
++AB*CD	A+B+C*D	AB+CD*+

Tanda kurung "(…)" untuk mengelompokkan suatu nilai sekaligus mengatur urutan operasi, namun dalam proses konversi tanda tsb tidak dicetak (tidak menjadi output).

KETENTUAN KONVERSI

Telusuri isi array (statement) character per character sampai ditemui character ‘\0’ (NULL).

1. Jika isinya ‘(‘ ➔ Push ‘(‘ tsb ke dalam Stack.
2. Jika isinya Operand ➔ Operand tersebut langsung dicetak (menjadi output).
3. Jika isinya Operator ➔ lakukan pemeriksaan :
   a. Jika Stack masih kosong (Top = -1) atau Stack yang paling atas berisi ‘(‘ ➔ Push, simpan Operator baru ke dalam Stack.
   b. Jika Stack ada isinya dan Stack yang paling atas isinya bukan ‘(‘ ➔ bandingkan derajat Operator baru dengan Operator yang ada pada posisi Top:
      i. Jika derajat Operator dalam Stack >= Operator baru ➔ keluarkan terus menerus dan cetak Operator dalam Stack, hingga ditemui isi Stack ‘(‘ atau hingga ditemui Operator yang derajatnya lebih rendah atau hingga isi Stack habis. Kemudian Push, simpan Operator baru ke dalam Stack.
      ii. Jika derajat Operator baru > Operator dalam Stack ➔ Push, simpan Operator baru ke dalam Stack.
4. Jika isinya ‘)‘ ➔ keluarkan (Pop) dan cetak isi Stack satu persatu hingga ditemui isi Stack ‘(‘. Setelah itu keluarkan isi Stack yang berupa ‘(‘ tsb dari dalam Stack.
5. Lakukan terus menerus hingga ditemui character ’\0’ dan keluarkan serta cetak isi Stack satu persatu hingga Stack kosong Kembali.

Beberapa Aplikasi Stacks

Stack banyak digunakan untuk :

- Membalik urutan data
- Ubah ekspresi infix menjadi postfix
- Ubah ekspresi postfix menjadi infix
- Tumpukan sistem digunakan di setiap fungsi rekursif
- Mengonversi bilangan desimal menjadi padanan binernya
- Mengurai (*Parsing*)HTML
- Menara Hanoi

Implementasi Stack : Notasi Polish

Menggubah notasi infix menjadi notasi postfix.
Contoh :

- A+B (infix)
- AB+ (postfix)

Misal :
Q = ekspresi matematika yang ditulis dalam notasi *infix*
P = penampung ekspresi matematika dalam notasi *postfix*

Algoritma

1. Push tanda “(“ ke stack dan tambahkan tanda “)” di sentinel di Q.
2. Scan Q dari kiri ke kanan, kemudian ulangi langkah c s.d f untuk setiap elemen Q sampai stack Q kosong.
3. Jika yang discan adalah operand, maka tambahkan ke P
4. Jika yang discan adalah “(“ maka push ke stack
5. Jika yang discan adalah “)” maka pop isi stack sampai ditemukan tanda “(“, kemudian tambahkan ke P sedangkan tanda “(“ tidak disertakanke P.

Algoritma

6. 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 tsb dantambahkan ke P.
   - Push operator tersebut ke stack.
7. Keluar

Algoritma

Contoh :

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

Algoritma

Penyelesaian :

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

setelah ditambahkan tanda ")" pada notasi sehingga terdapat 20 simbol sbb :

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

STACK

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*+

Tanda kurung "(…)" untuk mengelompokkan suatu nilai sekaligus mengatur urutan operasi, namun dalam proses konversi tanda tsb tidak dicetak (tidak menjadi output).

Tambahkan character '\0' sebagai tanda akhir statement.

Algoritma

Penyelesaian :

Hasil akhir :

Dari proses di atas didapatkan notasi postfix

Q = ABC*DEF^/G*-H*+

Notasi Polish (lanjutan)

Menghitung ekspresi matematika yang disusun dalam notasi *postfix.*

Contoh :

2,5,* (postfix)

Hasil : 10

Algoritma : Misal :

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

1. Tambahkan tanda ")" pada sentinel di P
2. Scan P dari kiri ke kanan, ulangi langkah c dan d untuk setiap elemen P sampai ditemukan sentinel.
3. Jika yang discan adalah operand, maka push ke stack.
4. 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.
5. Pop isi stack dan simpan di variable value.
6. Keluar.

Algoritma

Contoh Kasus :

Dari proses di atas didapatkan notasi postfix

P = 5, 2, 6, +, *, 12, 4, /, -

Tambahkan tanda ")"pada sentinel P sehingga

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

Didapatkan 10 simbol yaitu :

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

Penyelesaian

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).	

Hasil :

Didapatkan Bilangan 37

Operator Priority

- ^ Pangkat, akar
- /,* Pembagian, perkalian
- +, - Penambahan, pengurangan


TERIMA KASIH
FAIR &
RESPONSIBLE