STACK

Sunu Wibirama

Jadwal ujian MID

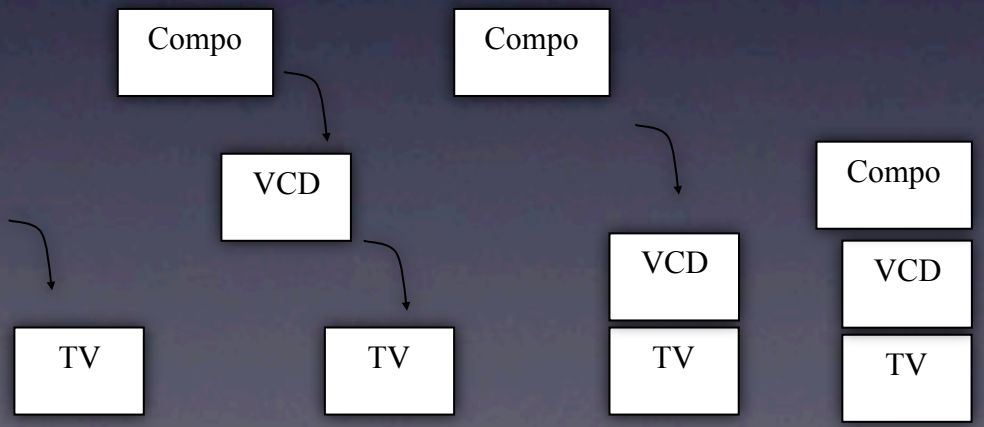
- Hari Rabu, 6 April 2011
- Pkl. 07.30 - 09.30 WIB
- Ruangan E3
- Open Book, tapi tidak diperkenankan menggunakan internet, membuka laptop, atau handphone

Bahan ujian

- Logika
- Pseudocode dan Flowchart
- Konsep Pointer dan Array
- Stack

Stack

- Stack bersifat LIFO (*Last In First Out*)
- "Benda yang **terakhir masuk** ke dalam stack akan menjadi yang **pertama keluar** dari stack



Ilustrasi Stack

- Compo di posisi terakhir, maka Compo akan menjadi elemen teratas dalam tumpukan.
- Kita menumpuk Televisi pada saat pertama kali, maka elemen Televisi menjadi elemen terbawah dari tumpukan.
- Jika kita mengambil elemen dari tumpukan, maka secara otomatis akan terambil elemen teratas, yaitu Compo.
- **Operasi-operasi/fungsi Stack**
  - **Push :** digunakan untuk menambah item pada stack pada tumpukan paling atas
  - **Pop :** digunakan untuk mengambil item pada stack pada tumpukan paling atas
  - **Clear :** digunakan untuk mengosongkan stack
  - **IsEmpty :** fungsi yang digunakan untuk mengecek apakah stack sudah kosong
  - **IsFull :** fungsi yang digunakan untuk mengecek apakah stack sudah penuh

Stack dengan Struct Array

- Definisikan Stack dengan menggunakan struct
- Definisikan konstanta MAX_STACK untuk menyimpan maksimum isi stack
- Buatlah variabel array data sebagai implementasi stack
- Deklarasikan operasi-operasi/function di atas dan buat implemetasinya

Program Stack

- **Deklarasi MAX_STACK**

```
#define MAX_STACK 10
```

- **Deklarasi STACK dengan struct dan array data**

```
typedef struct STACK{
   int top;
   char data[10][10];
};
```

- **Deklarasi/buat variabel dari struct**

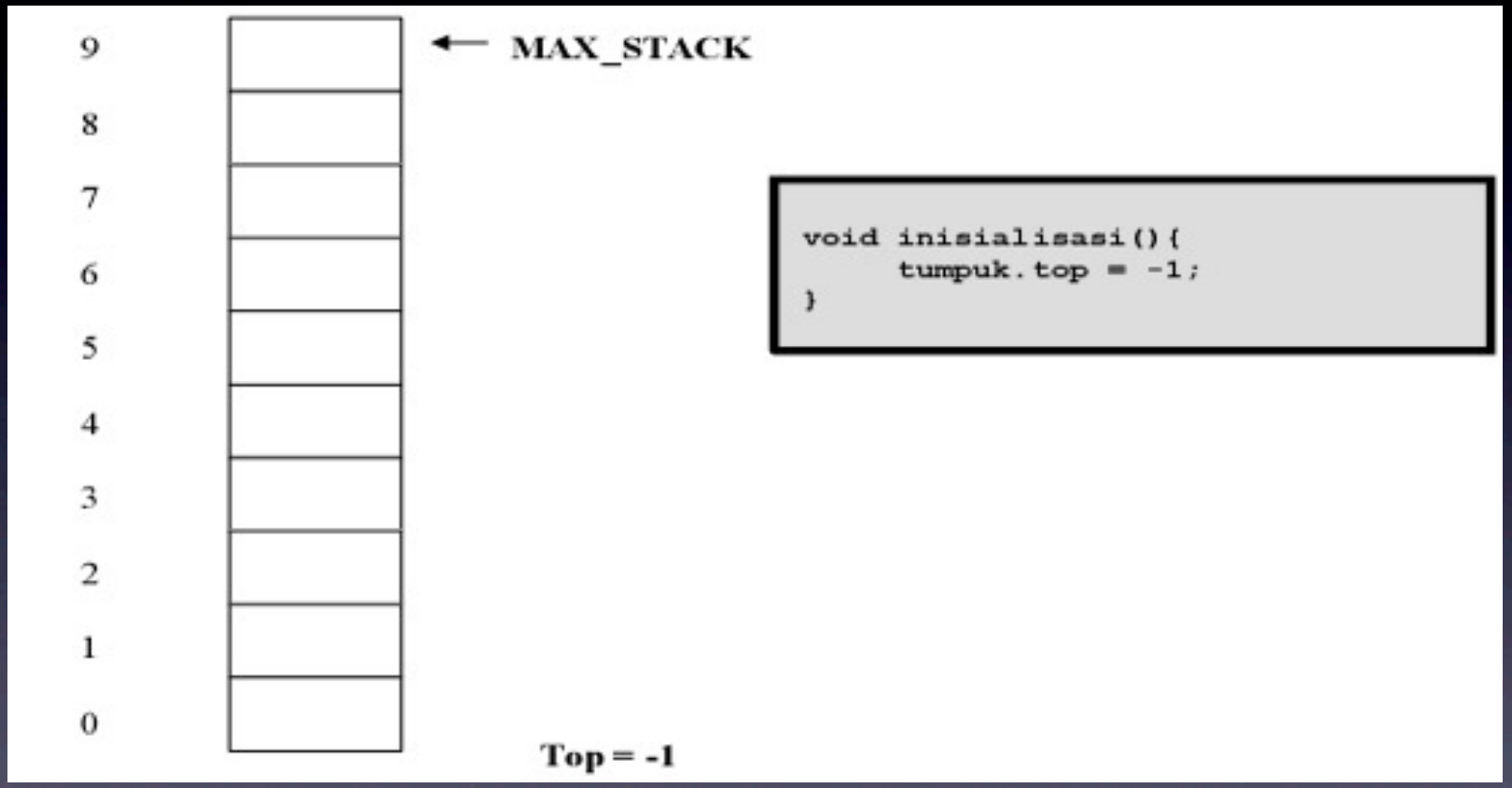
```
STACK tumpuk;
```

Program Stack (2)

Inisialisasi Stack

- Pada mulanya isi top dengan -1, karena array dalam C dimulai dari 0, yang berarti stack adalah KOSONG!
- Top adalah suatu variabel penanda dalam STACK yang menunjukkan elemen teratas Stack sekarang. Top Of Stack akan selalu bergerak hingga mencapai MAX of STACK sehingga menyebabkan stack PENUH!

Program Stack (2)




```
void inisialisasi(){
     tumpuk.top = -1;
}
```

Ilustrasi Stack pada saat inisialisasi!

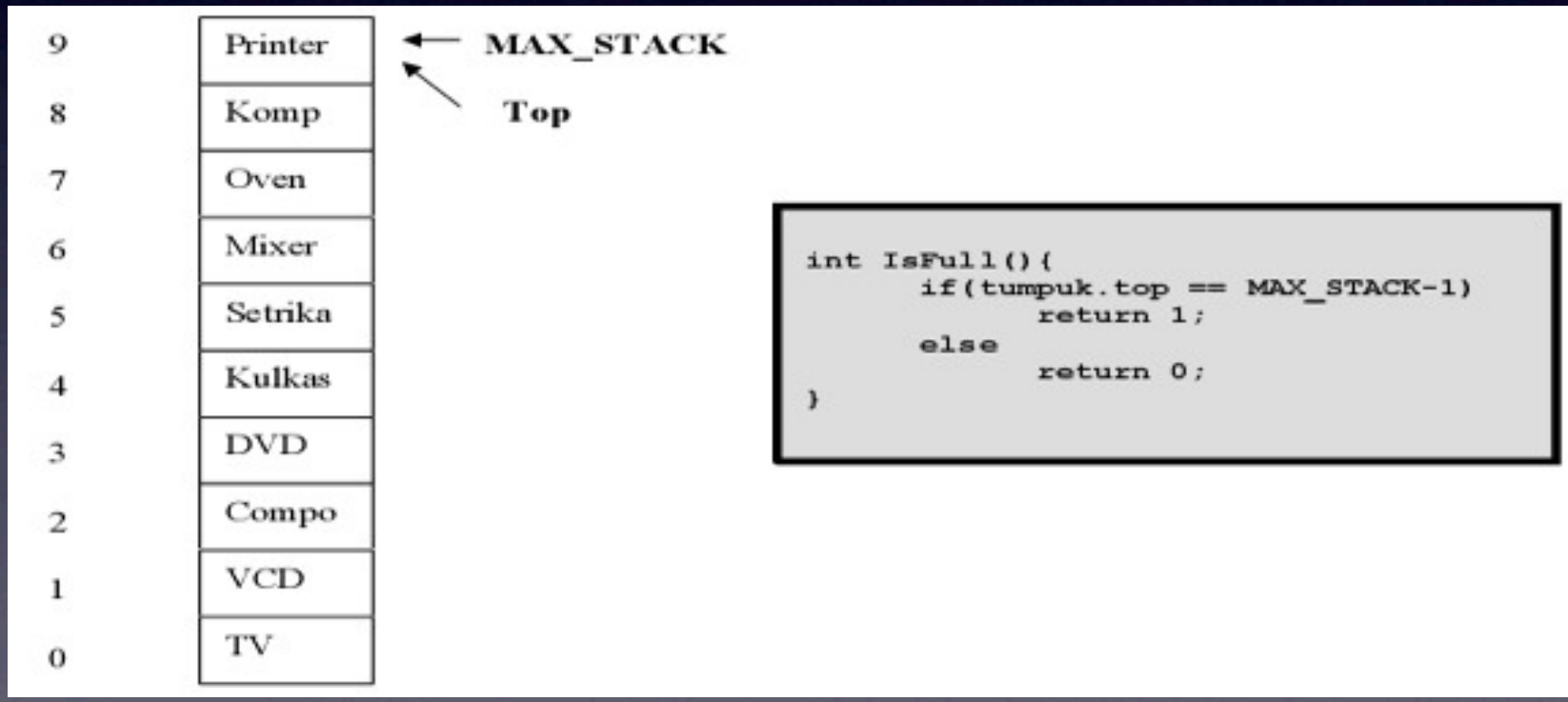
Program Stack (3)

Fungsi IsFull

- Untuk memeriksa apakah stack sudah penuh?
- Dengan cara memeriksa top of stack, jika sudah sama dengan MAX_STACK-1 maka full, jika belum (masih lebih kecil dari MAX_STACK-1) maka belum full

Program Stack (4)

- Ilustrasi Stack pada kondisi Full




```
int IsFull(){
      if(tumpuk.top == MAX_STACK-1)
            return 1;
      else
            return 0;
}
```

Program Stack (5)

Fungsi IsEmpty

- Untuk memeriksa apakah stack masih kosong?
- Dengan cara memeriksa top of stack, jika masih -1 maka berarti stack masih kosong!

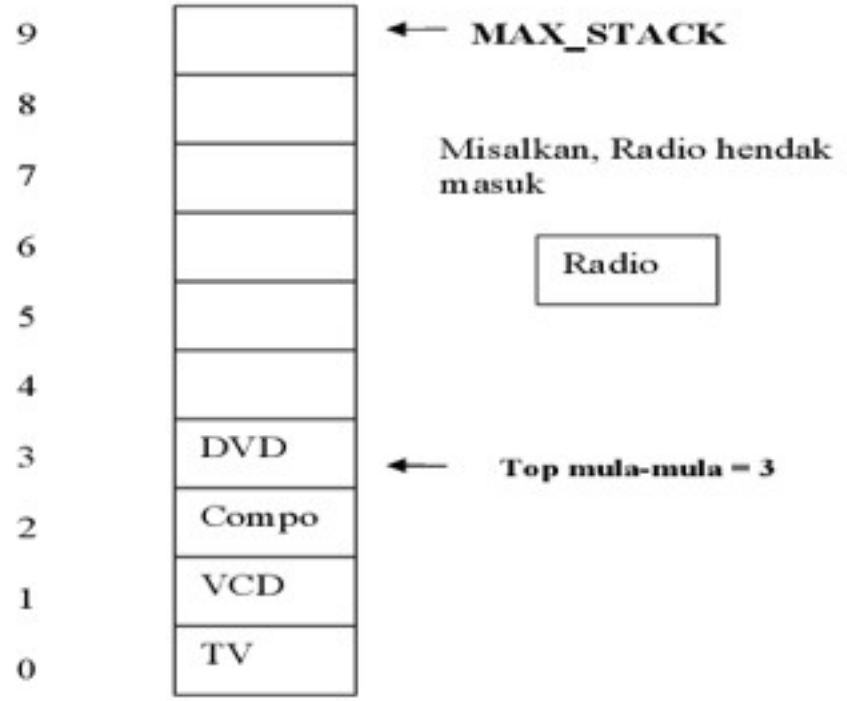
```
int IsEmpty(){
  if(tumpuk.top == -1)
       return 1;
  else
       return 0;
}
```

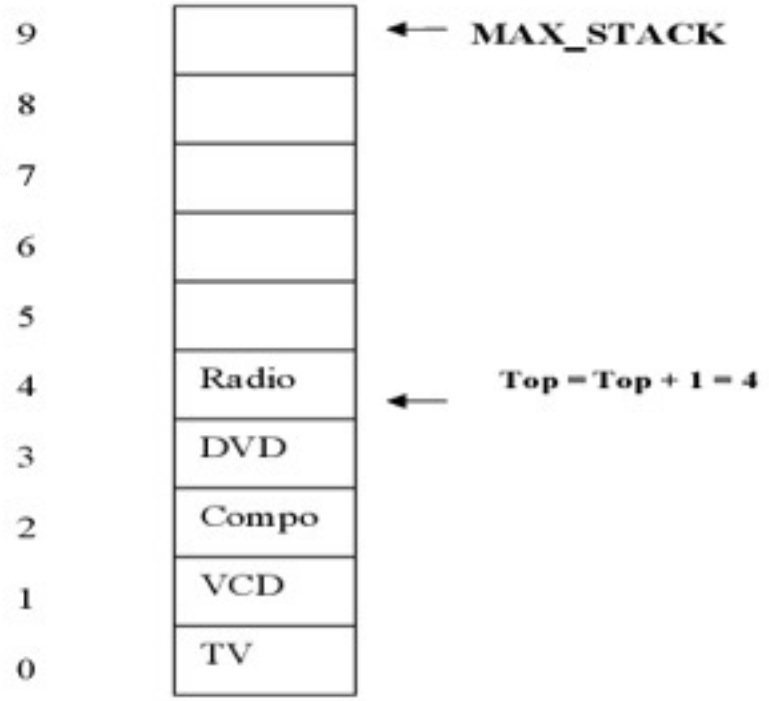
Program Stack (6)

Fungsi Push

- Untuk memasukkan elemen ke stack, selalu menjadi elemen teratas stack (yang ditunjuk oleh TOS)
- Tambah satu (increment) nilai top of stack lebih dahulu setiap kali ada penambahan elemen stack.
- Asalkan stack masih belum penuh, isikan data baru ke stack berdasarkan indeks top of stack setelah diincrement sebelumnya.

Program Stack (7)





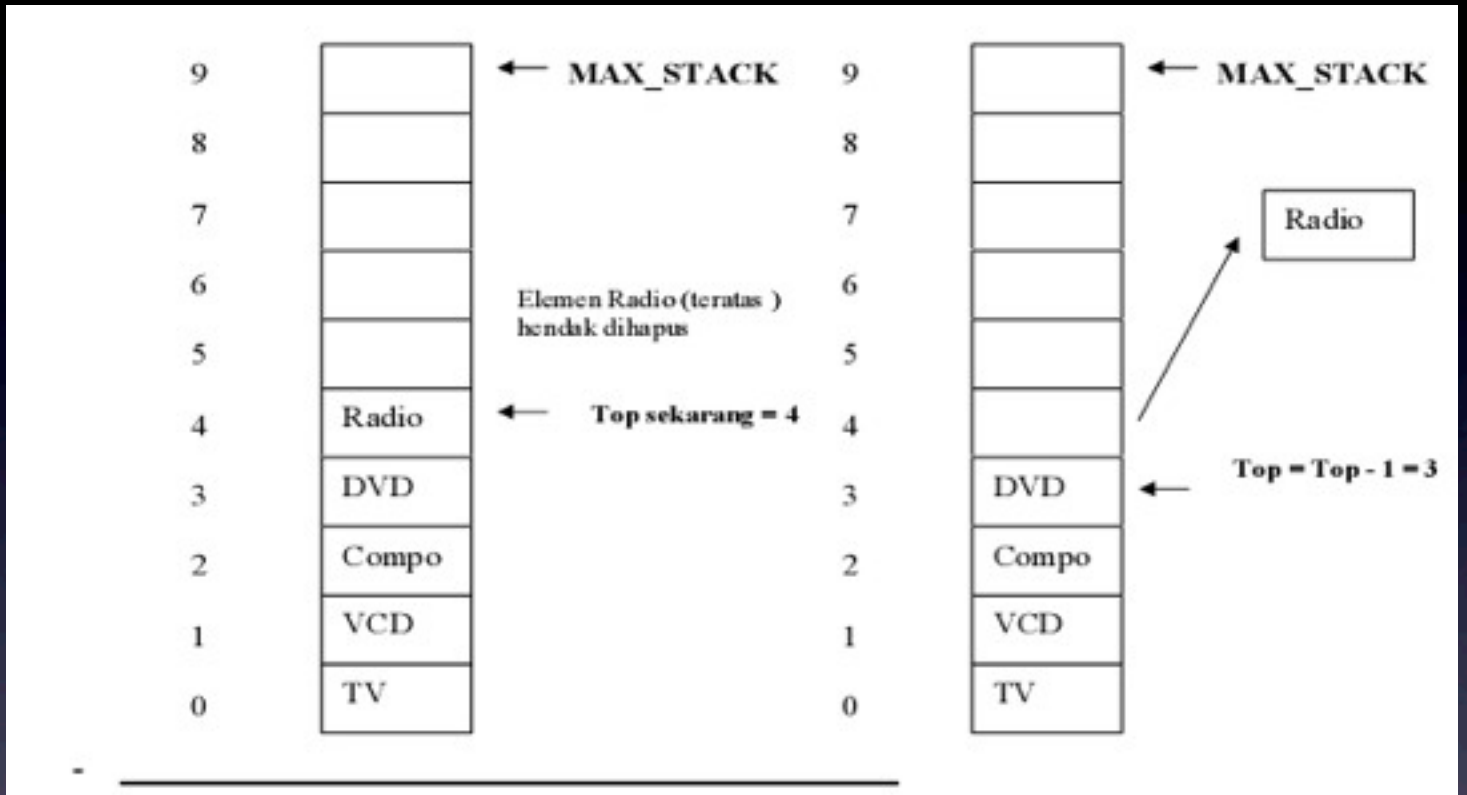

```
void Push(char d[10]){
  tumpuk.top++;
  strcpy(tumpuk.data[tumpuk.top],d);
}
```

Program Stack (8)

Fungsi Pop

- Untuk mengambil elemen teratas (data yang ditunjuk oleh TOS) dari stack.
- Ambil dahulu nilai elemen teratas stack dengan mengakses top of stack, tampilkan nilai yang akan dipop, baru dilakukan decrement nilai top of stack sehingga jumlah elemen stack berkurang

Program Stack (9)



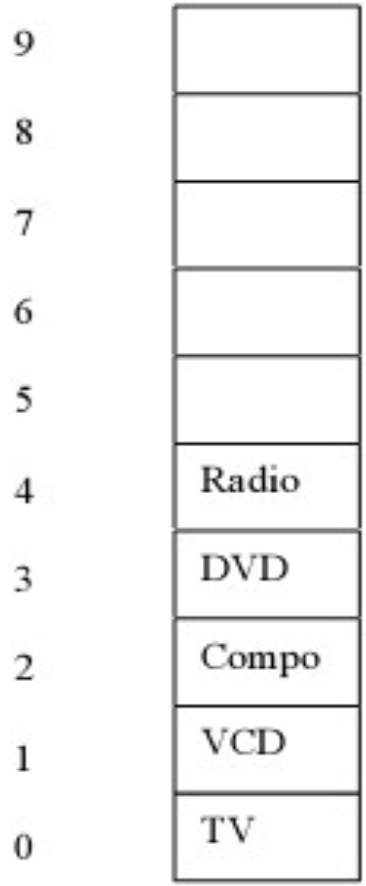
Programnya:

```
void Pop(){
   printf("Data yang terambil = %s\n",tumpuk.data[tumpuk.top]);
   tumpuk.top--;
}
```

Program Stack (10)

- **Fungsi Print**
- Untuk menampilkan semua elemen-elemen stack
- Dengan cara looping semua nilai array secara terbalik, karena kita harus mengakses dari indeks array tertinggi terlebih dahulu baru ke indeks yang kecil!

Program Stack (11)



← **MAX_STACK**

Jika kita ingin mengeprint elemen stack, kita harus membaca elemen array dari indeks yang ada isinya tertinggi, yaitu dari Top downto 0

Sehingga kita akan menampilkan elemen:
Radio, DVD, Compo, VCD, dan TV

← **Top**

Program:

```
void TampilStack(){
     for(int i=tumpuk.top;i>=0;i--){
             printf("Data : %s\n",tumpuk.data[i]);
     }
}
```

Infix-Postfix-Prefix

- Perhatikan kalimat aritmatika A*(B+C)
- Infix: "in" , operator berada di dalam operand. Contoh: A + B
- Postfix: operator berada setelah operand. Contoh: A B +
- Prefix: operator berada sebelum operand. Contoh: + A B

Contoh Konversi

Infix	Postfix	Prefix
A + B	A B +	+ A B
A + B + C	A B + C +	+ + A B
A + B * C	A B C * +	+ A * B C
(A + B) * C	A B + C *	* + A B C
A + B + C * D	A B + C D * +	+ + A B * C D
A + (B+C) * D	A B C + D * +	+ A * + B C D

Studi Kasus Stack

- Perhitungan dua buah stack
- Pembuatan Kalkulator SCIENTIFIC
  - Misalkan operasi: 3 + 2 * 5
  - Operasi di atas disebut notasi infiks, notasi infiks tersebut harus diubah lebih dahulu menjadi
  - notasi postfix 3 + 2 * 5
  - notasi postfiksnya adalah 2 5 * 3 +

Studi Kasus Stack (2)

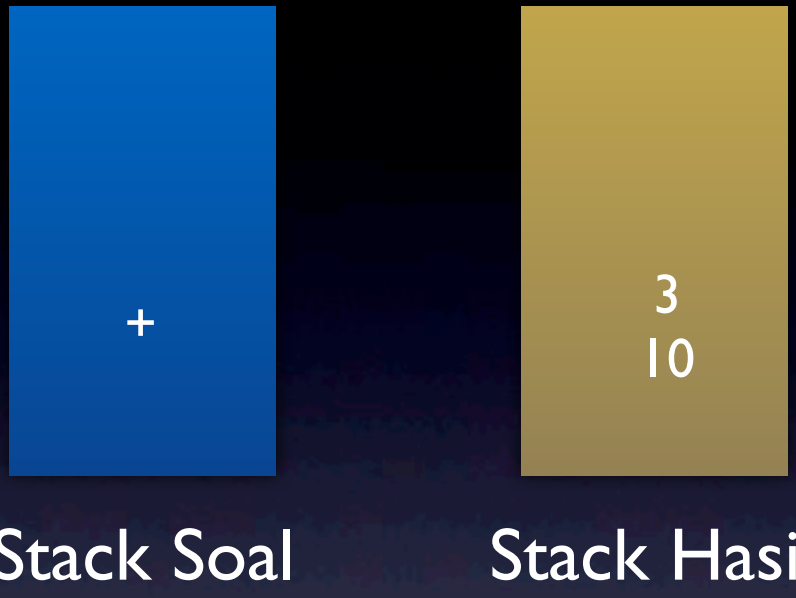
Stack Soal Stack Hasil

- Pop Stack Soal:
- Jika berupa operand, maka masukkan ke Stack hasil
- Jika berupa operator, maka:
  - Pop nilai pertama dari Stack Hasil
  - Pop nilai kedua dari Stack Hasil
  - Lakukan operasi sesuai dengan operator yang didapat.

Studi Kasus Stack (3)

- Operator * di pop dari Stack Soal.
- Pop Stack Hasil dua kali, yaitu 5 dan 2 kemudian, simpan 5 ke dalam variabel misalnya a, dan 2 ke dalam variabel misalnya b.
- Lakukan operasi sesuai dengan operatornya, b <operator> a
- Jadi b * a, yaitu 2 * 5 kemudian hasilnya disimpan lagi ke dalam Stack Hasil

Studi Kasus Stack (4)



- Kemudian lakukan langkah yang sama, sampai selesai.
- Pada contoh: operator + dipop dari Stack Soal
- Pop Stack Hasil dua kali, yaitu 3, disimpan pada variabel a, dan 10, disimpan pada variabel b. Kemudian lakukan operasi sesuai dengan operatornya, b <operator> a
- Jadi b + a, yaitu 10 + 3 = 13.

Ilustrasi Operasi Stack
3 + 2 * 5

Input	Operation	Stack
2	Push operand	2
5	Push operand	2, 5
*	Multiply	10
3	Push operand	10, 3
+	Add	13

Hasil akhir 13, tinggalkan pada *top stack*

Kasus Lain

- Soal: Selesaikan soal ((1+2)*4)+3 menggunakan operasi stack
- Langkah-langkah:
  - Operasi infix dirubah ke operasi postfix: 1 2 + 4 * 3 +
  - Buat ilustrasi operasi stack dengan tabel

Ilustrasi Operasi Stack ((1+2)*4)+3

Input	Operation	Stack
1	Push operand	1
2	Push operand	1, 2
+	Add	3
4	Push operand	3, 4
*	Multiply	12
3	Push operand	12, 3
+	Add	15

Hasil akhir 15, tinggalkan pada *top stack*

Kesimpulan Presentasi

- Secara umum sudah baik, penjelasan lebih jelas karena ada langkah-langkah dan animasinya
- Ada program riil untuk menunjukkan implementasi dari tema yang dibahas
- Disampaikan juga, kapan kita membutuhkan jenis struktur data / algoritma yang sedang dibahas (mis. Stack : untuk menghemat memori saat melakukan operasi aritmatika)

Terima Kasih