Stack

(TIB11 – Struktur Data)

Pertemuan 15, 16

Sub-CPMK

- Mahasiswa mampu menggunakan linked list dan array untuk membuat stack beserta operasi-operasinya (C3, A3)

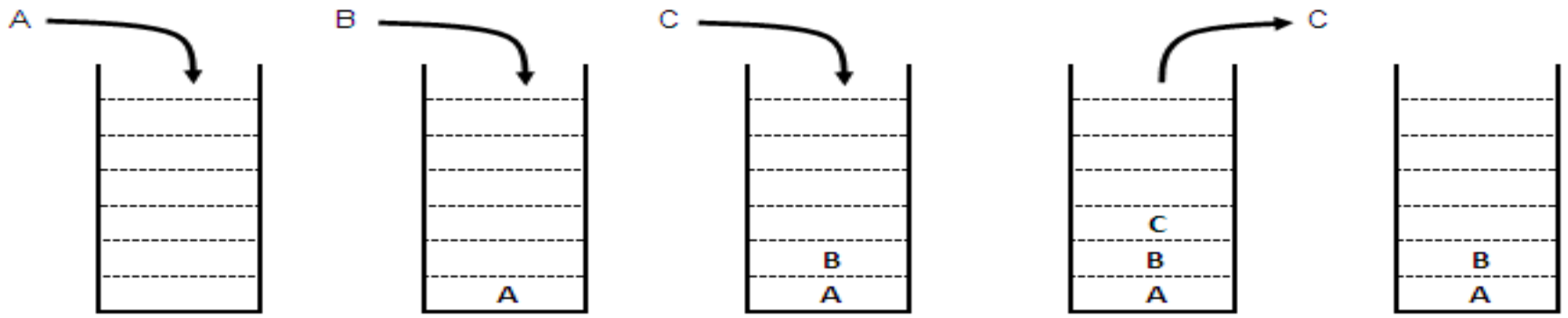
Materi:

1. Pengertian *Stack*
2. *Array Base Stack*
3. *Linked-List Base Stack*

1. Pengertian *Stack*

1.1. Stacks / Tumpukan

- Struktur Data Linear yang dapat diakses hanya pada satu sisi yaitu pada akhir (*top*) untuk menyimpan dan mengambil data. (*Last In First Out*)

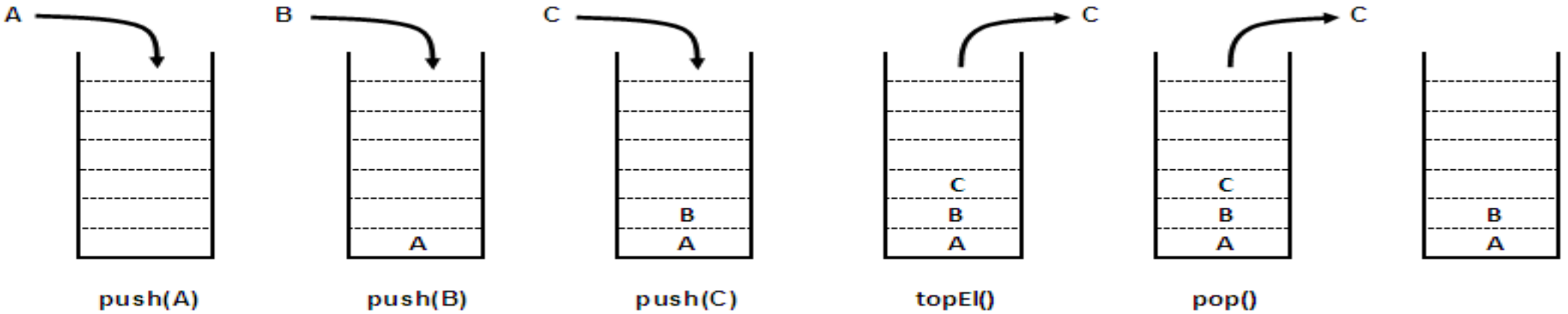


1.2. *Stacks Operation*

- Clear() : membersihkan semua isi *Stack*
- isEmpty() : memeriksa apakah *stack* dalam kondisi kosong
- push(el) : meletakkan *element* el pada *top stack*
- pop() : mengambil *element* dari *stack* pada *topmost*
- topEl(): membaca *topmost element* dari *stack* tanpa menghapusnya
- Seringkali adapula orang yang menambahkan operasi memeriksa size *stack*

1.2. *Stacks Operation (Lanj.)*

Illustrations:



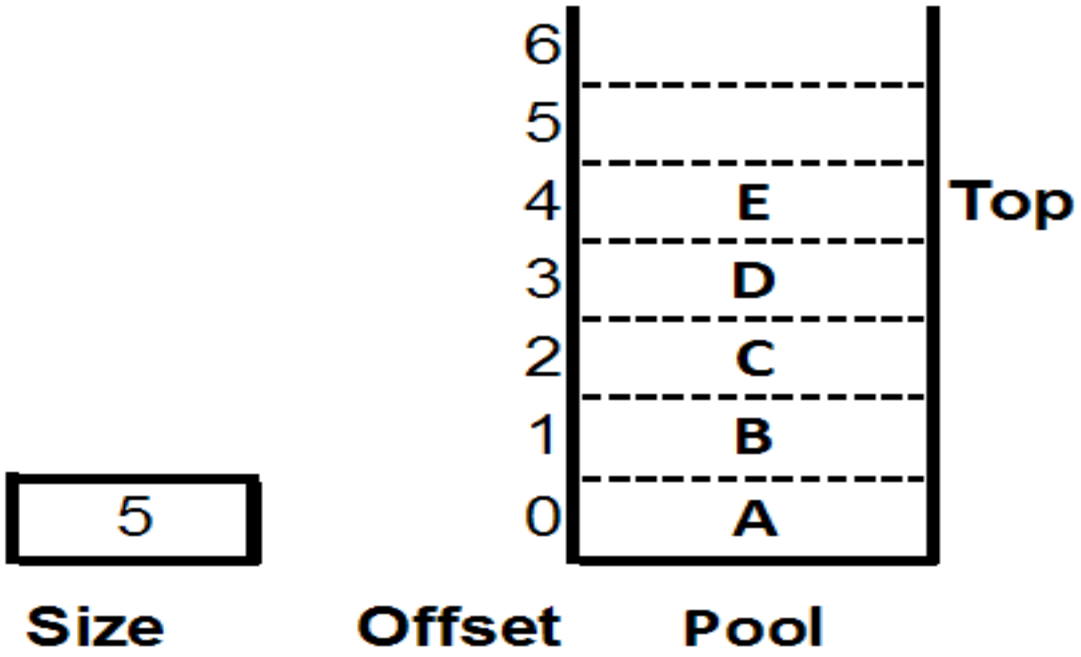
1.3. Implementasi *Stack*

Dapat diterapkan dengan dua cara

- *Linked-List Base Stack* → implementasi *stack* menggunakan *linked-list*
- *Array Base Stack* → implementasi *stack* menggunakan *array*

2. *Array Base Stack*

2.1. Implementasi Stack Dengan Array



Variable yang diperlukan

- *Sets of array as pool*
- *PoolSize*
  - Jumlah sel yang berisi pada *stacks*
  - Dapat digunakan untuk mendapatkan informasi dari *top* stack

    Top = PoolSize - 1

2.2. Operasi Stack

- Clear()
  - *Set* semua nilai pada *sets of array* dengan null
  - *Set* PoolSize dengan 0
- isEmpty()

  Periksa PoolSize Value
  - PoolSize Value = 0 → empty
  - PoolSize Value > 0 → not empty

2.2. Operasi Stack (Lanj.)

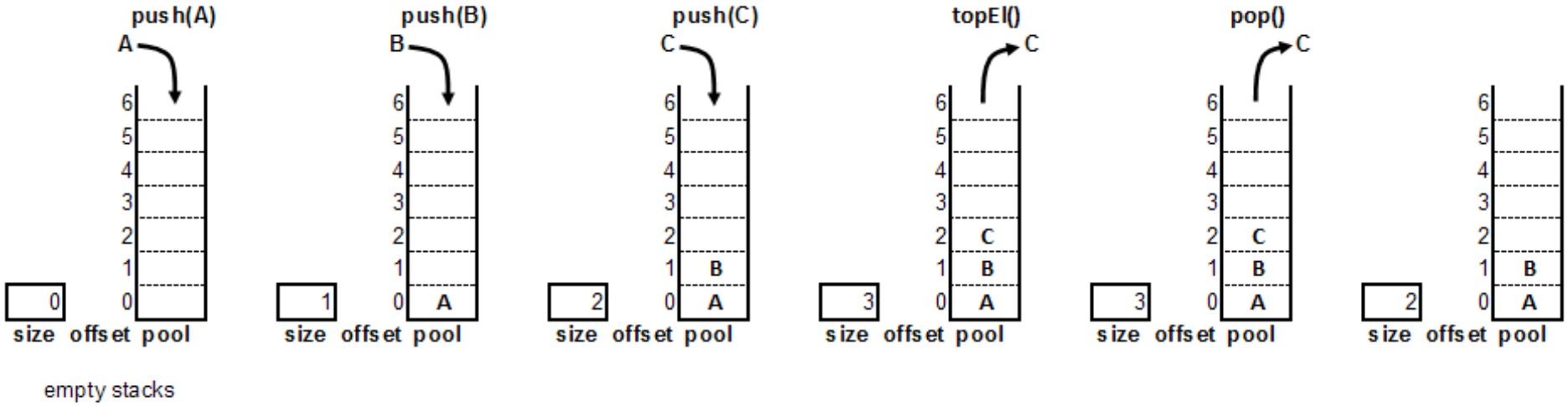
- push(el) :
  - Tambahkan nilai PoolSize dengan 1
  - Ambil data el, dan tulis pada sel yang ditunjuk oleh indeks PoolSize
- pop() :
  - Ambil topmost *element* dari *stack* dan masukkan ke *variable* yang digunakan.
  - Kurangi nilai PoolSize dengan 1
- topEl():
  - Ambil nilai topmost *element* dari *stack* tanpa menghapusnya, masukkan ke variabel

2.2.Operasi Stack (Lanj.)

Yang harus diingat pada *stacks* menggunakan *array*:

- Ketika melakukan *push data* ke *stacks*, harus diperiksa untuk memastikan *array* belum penuh.
- Untuk mengindikasikan *array* sudah penuh atau belum, dilakukan dengan memeriksa jumlah data yang terisi pada *array*, jika data *size* sesuai dengan *array size* maka berarti *array* sudah penuh. Batalkan *push data* ke *array stacks*.

2.2. *Stacks Array Operation* (Lanj.)



- Push memasukkan elemen pada top stack,
- topEl() hanya melihat isi dari top stack,
- sdangkan pop() mengeluarkan elemen pada top dari stack()

3. *Linked-List Base Stack*

3.1. *Implementation Stacks* dengan *Linked List*

Data store yang dibutuhkan

- *Single Linked List* dengan *Head* Node
- *Head* berfungsi sebagai *top Stacks*
- (saran: dapat juga menggunakan *Tail* Node sebagai *top stack*, dengan demikian pengisian selalu dilakukan pada *Tail*)

3.2. *Stacks-Linked List Operation*

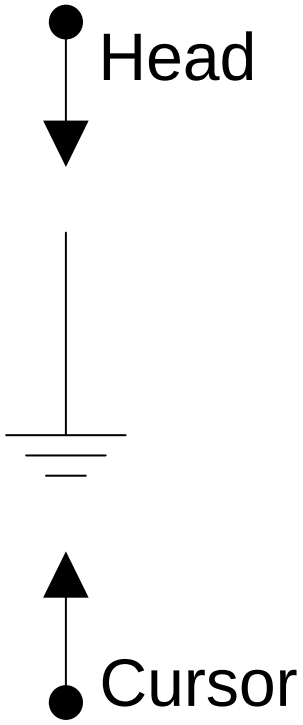
- Clear()
  - *Set Variabel Head* dengan Null
- isEmpty()

  Check isi simpul *Head*
  - Head Value = NULL → empty
  - Head Value Not NULL → not empty

3.2. *Stacks-Linked List Operation* (Lanj.)

- push(el) :
  - Buat Node baru
  - *Set Next Link* dari Node baru ke node *Head*
  - *Set* Node baru sebagai *Head*
  - Ambil data el dan isi ke node baru
- pop() :
  - Ambil *Head*->data dan masukkan ke *variable* yang akan digunakan
  - Isi variabel *Head* dengan *Head->Next*
- topEl()
  - Ambil *Head*->data dan masukkan ke *variable* yang akan digunakan

3.3. *Stacks-Linked List Operation*: isEmpty()

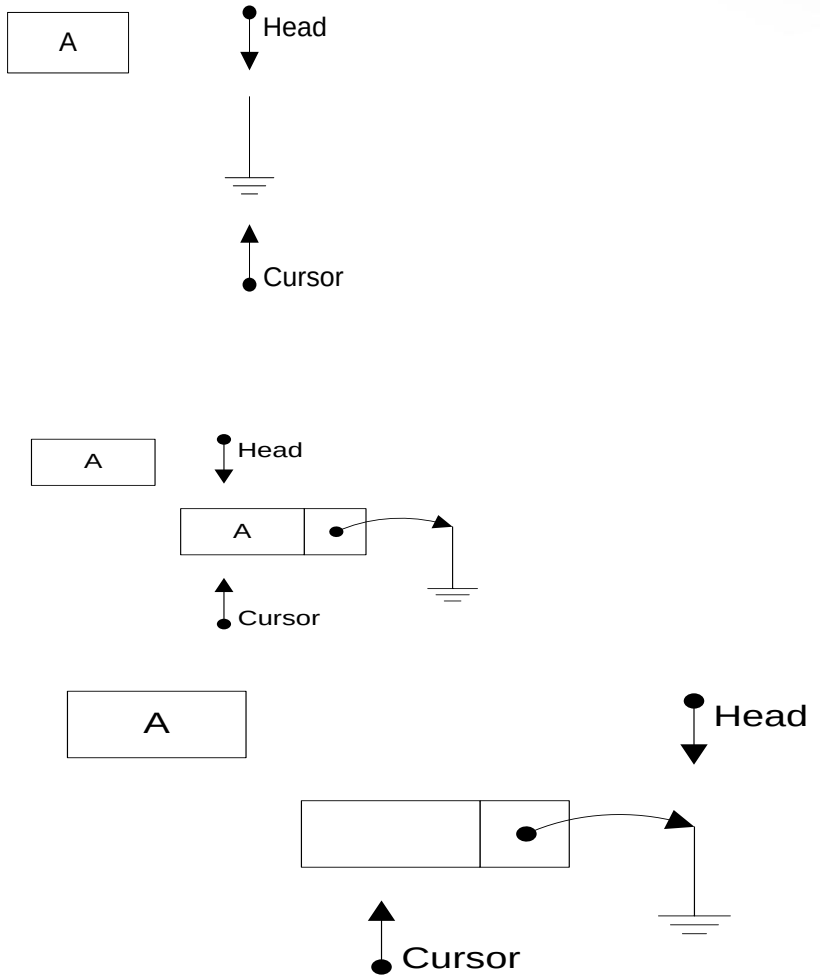


- Jika Cursor atau pointer berada pada Head, atau top, dan sama-sama berisi NULL, berarti stack dalam kondisi kosong

3.4. *Stacks-Linked List Operation*: *Push Stack* pada *empty stacks*

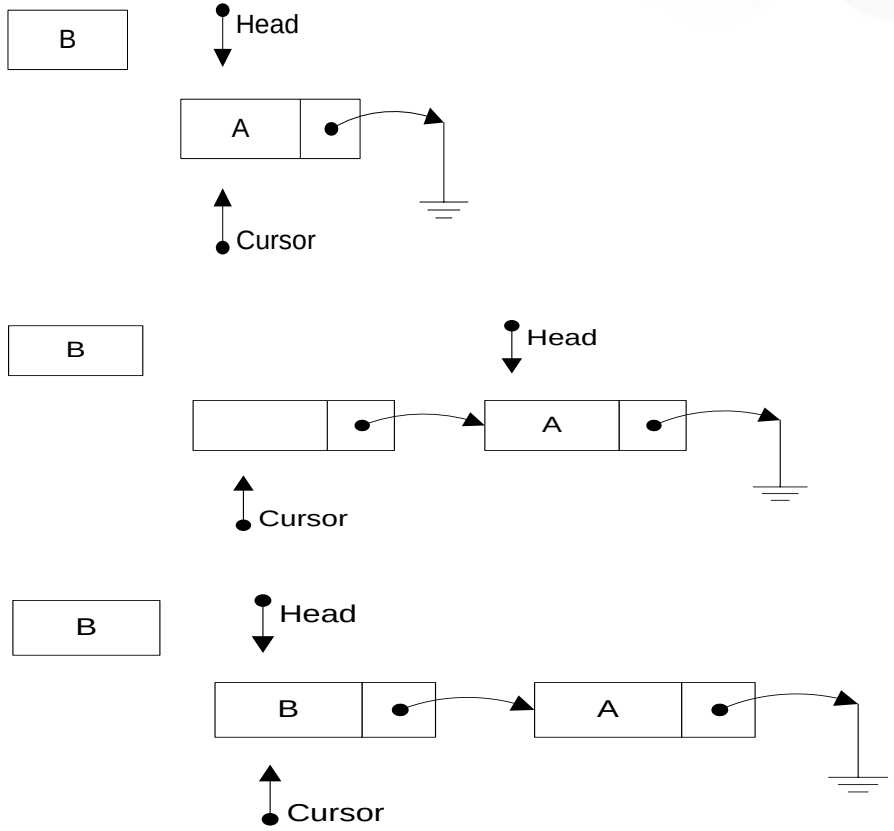
push(A)

- Simpul baru dibentuk,
- elemen A diisi pada simpul baru
- Head dan cursor / pointer menunjuk ke simpul baru

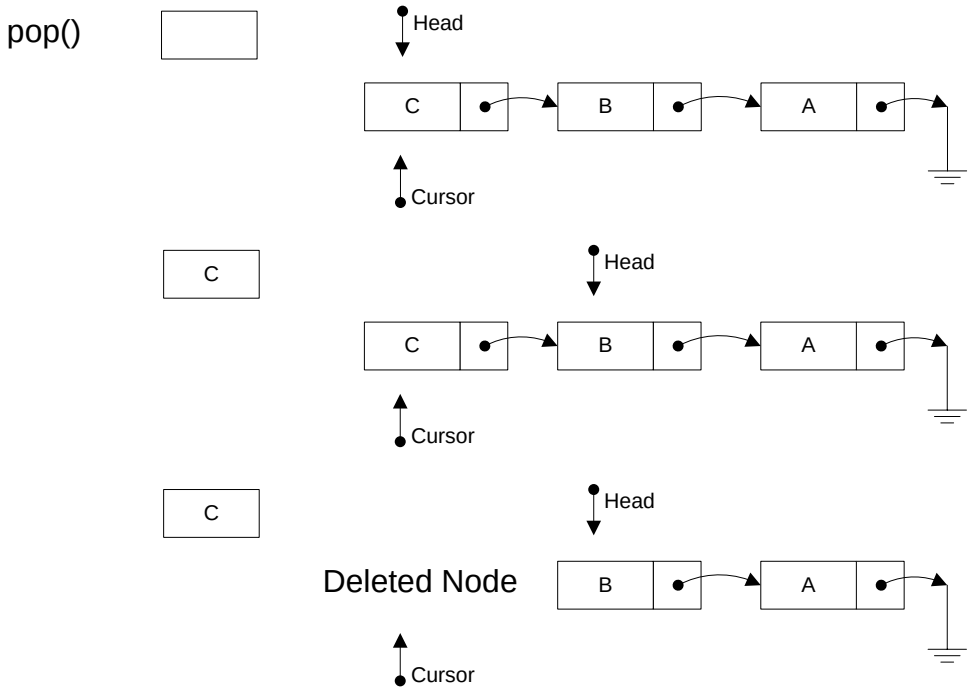


3.5. *Stacks-Linked List Operation*: *Push Stack* pada *stacks* yang berisi

- push(B)
- Simpul baru yang akan diisi elemen B dibuat dan dilink nya diarahkan ke Head
- Selanjutnya Head dipindahkan ke simpul baru

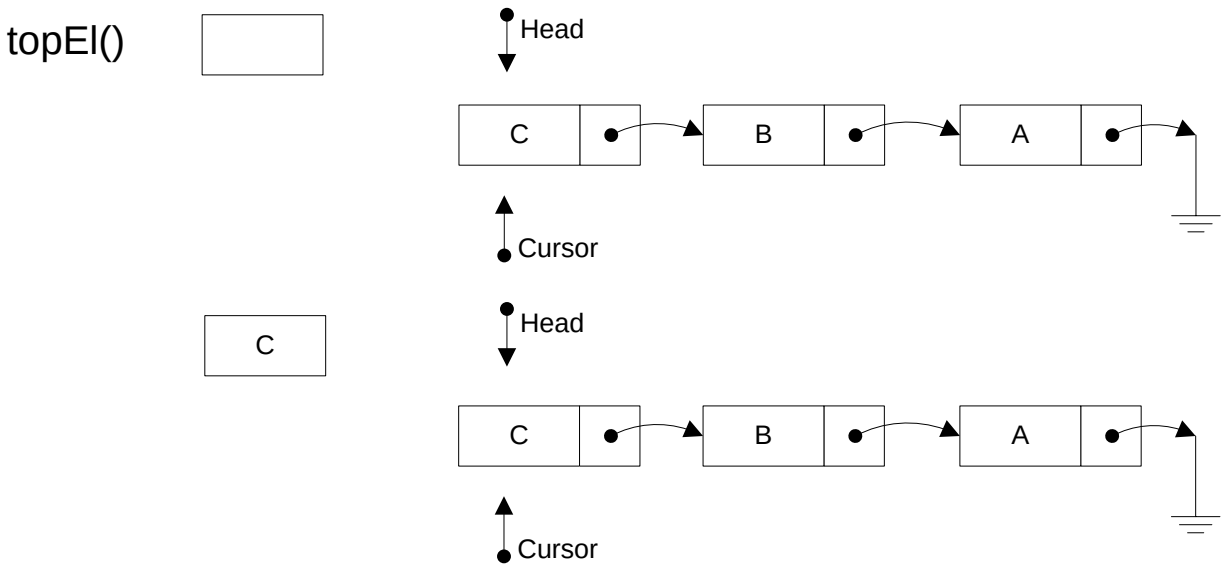


3.6. *Stacks-Linked List Operation*: pop()



- Ptr diarahkan ke Head
- Data / elemen pada top stack ditampung pada variabel
- Pindahkan Head
- Hapus node yang ditunjuk oleh cursor/ptr

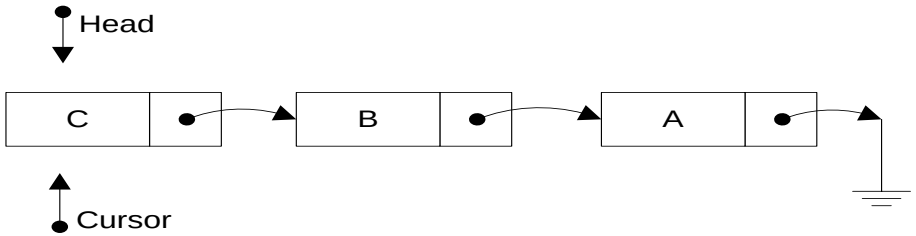
3.7. *Stacks-Linked List Operation*: topEl()



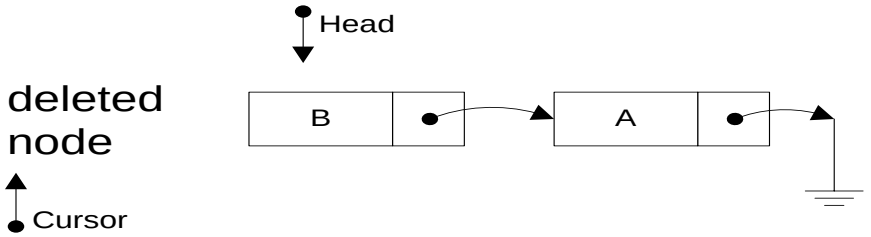
- TopEl hanya membaca data pada top Stack
- Cursor / Ptr arahkan ke Head
- Copykan isi / elemen yang ditunjuk oleh Head ke variabel penampungnya

3.8. *Stacks-Linked List Operation*: clear()

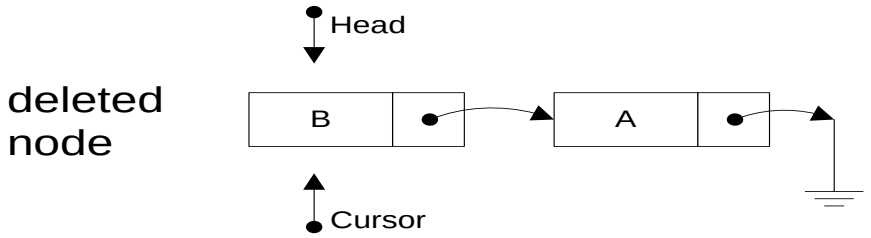
1.



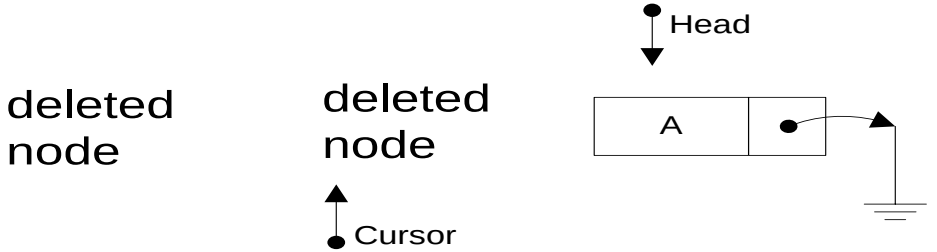
2.



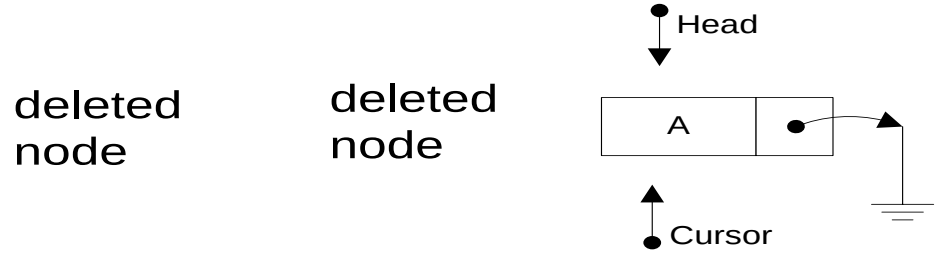
3.



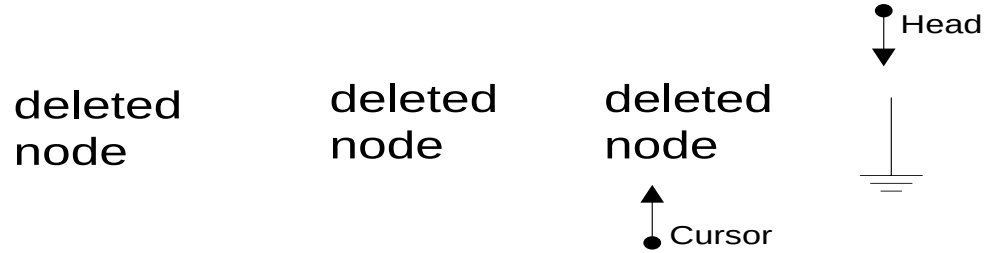
4.



5.



6.



3.8. *Stacks-Linked List Operation*: clear() (lanj.)

Sesuai dengan yang diperagakan pada gambar,

- Arahkan cursor ke Head, kemudian pindahkan Head ke simpul berikutnya
- Hapus simpul yang ditunjuk oleh cursor
- Ulangi langkah dari langkah pertama sampai Cursor dan Head menunjuk NULL

Ringkasan

- *Stack* adalah Struktur Data Linear yang dapat diakses hanya pada satu sisi yaitu pada akhir (*top*) untuk menyimpan dan mengambil data. (*Last In First Out*) berisi operasi: Clear(), isEmpty(), push(el), pop() dan topEl().
- *Stack* dapat diterapkan dengan dua cara
  - *Linked-List Base Stack* --> implementasi stack menggunakan *linked-list*
  - *Array Base Stack* --> implementasi *stack* menggunakan *array*

Contoh Program

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <string.h>
#include <iomanip.h>

//Record Definition
struct TheCell
{
  int dat;
  struct TheCell *berikut;
};

struct TheCell *ptrCell=NULL;
struct TheCell *kepala=NULL;
```

Contoh Program (Lanj.)

```
void push(int isi)
{
  struct TheCell *baru;
  baru=(struct TheCell *) malloc(sizeof(struct TheCell));
  if (kepala!=NULL) //untuk mengecek stack kosong atau tidak. atau bisa juga pakai
isEmpty()
  {  //kalau tidak kosong, baru-> diarahkan ke kepala
    baru->berikut = kepala;
  }
  else
  {
    baru->berikut = NULL;
  }
  kepala = baru;
  kepala->dat = isi;
}
```

Contoh Program (Lanj.)

```
int top()
{
  return(kepala->dat);
}


bool isEmpty()
{
  if (kepala==NULL)
    return(true); //Berarti benar kosong
  else
    return(false); //Berarti bukan kosong
}
```

Contoh Program (Lanj.)

```
int pop()
{
  if (kepala!=NULL)
  {
    int getData;
    getData = kepala->dat;
    ptrCell = kepala;
    kepala = kepala->berikut;
    free(ptrCell);
    return(getData);
  }
  else
  {
    return(NULL);
  }
}
```

Contoh Program (Lanj.)

```
//program utama
void main()
{
  //deklarasi variable
  int i; int bilRandom;
  //pengisian bilangan random ke dalam stack
  for (i=1;i<=10;i++)
  {
    bilRandom = rand();
    push(bilRandom);
  }
}
```

Terimakasih

TUHAN Memberkati Anda

Teady Matius Surya Mulyana (tmulyana@bundamulia.ac.id)