Queue

(TIB11 – Struktur Data)

Pertemuan 17, 18

Sub-CPMK

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

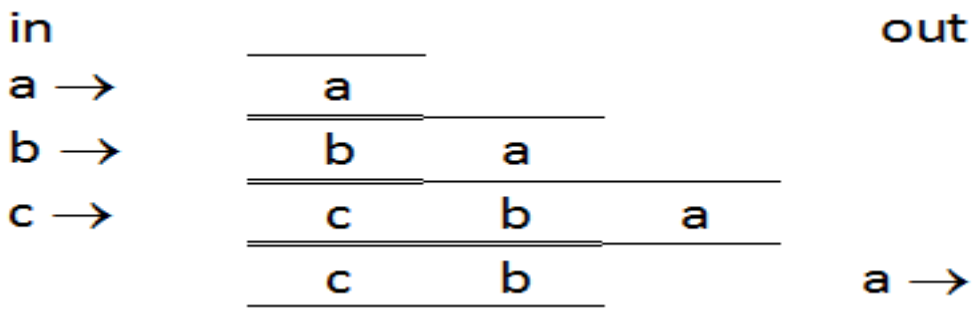
Materi:

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

1. Pengertian *Queue*

1.1. *Queue*

- Penambahan elemen dengan cara penambahan *elements* pada akhir dan mengeluarkan *element* dari depan
- *First In First Out*

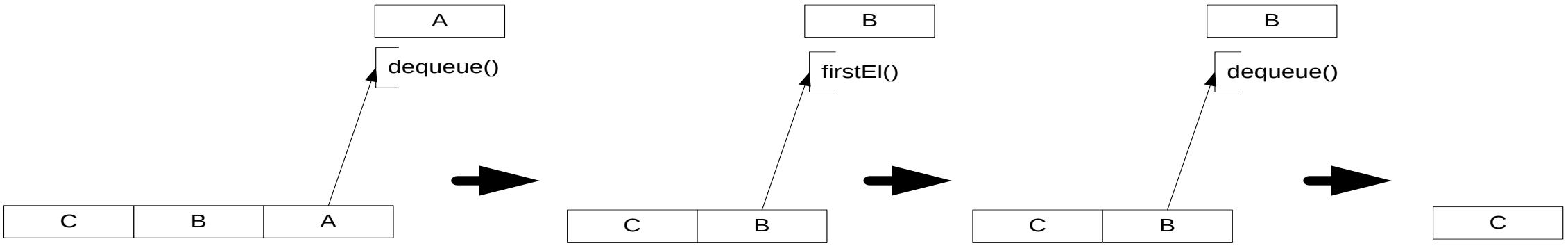


1.2. *Queue Operation*

- clear() → menghapus / membersihkan *queue*
- isEmpty() → memeriksa apakah *queue* kosong
- enqueue(el) → memasukkan *element* el pada akhir *queue*
- dequeue → Mengambil *element* pertama dari *queue*
- firstEl() → Membaca first *element* dari *queue* tanpa menghapusnya

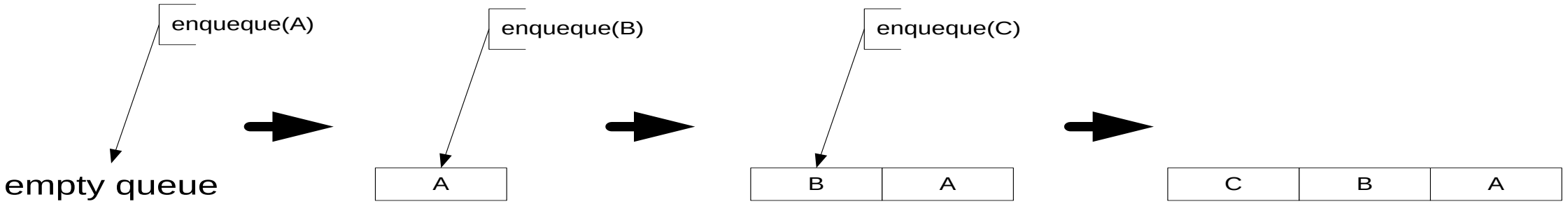
1.2. *Queue Operation*

- clear() → menghapus / membersihkan *queue*
- *Mulai dari last element, lakukan dequeue() sampai seluruh elemen habis*



1.2. *Queue Operation* (Lanj.)

- enqueue(el) → memasukkan *element* el pada akhir *queue*
- *Enqueue dilakukan pada first queue*



2. *Array Base Queue*

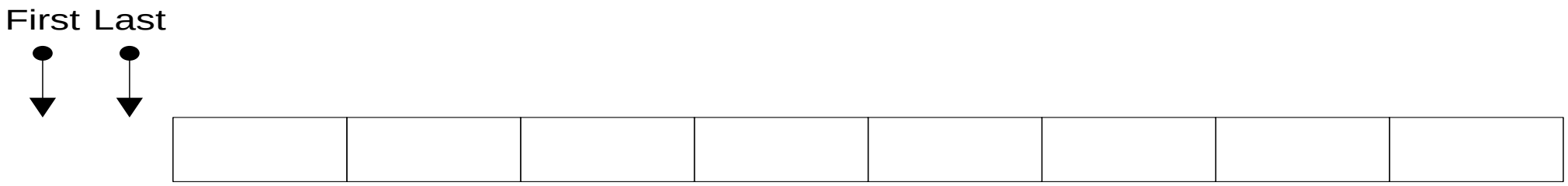
2.1. Implementasi *Queue* dengan *Array*

Variable yang dibutuhkan

- *Array* sebagai *queue pool*
- *Integer variable First* untuk menginformasikan *offset number* dari *array* yang menjadi *Queue* pertama
- *Integer variable Last* untuk menginformasikan *offset number* dari *array* yang menjadi *Queue* terakhir
- *Array* harus *circular* untuk mempermudah penerapan

2.2. Kondisi *First* dan *Last Variable* Pada *Circular Array*

- Pada kondisi awal, kedua *First* dan *Last variable* dapat di *set* -1 untuk mengindikasikan *queue is empty* dan data *size* harus di *set* dengan 0

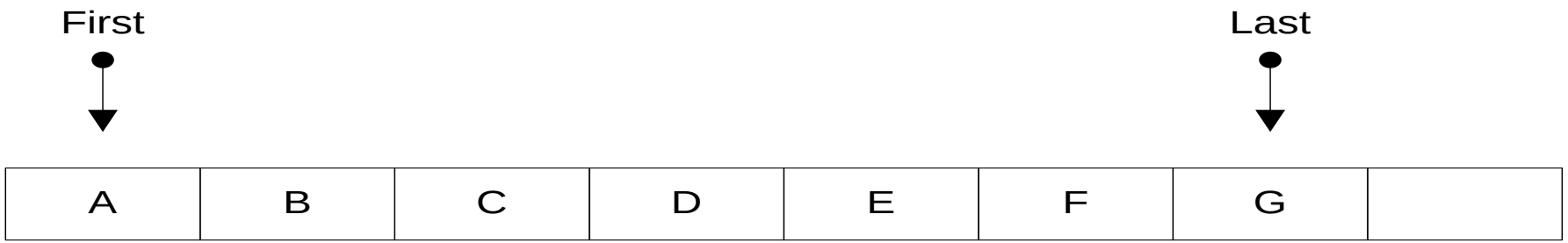


2.2. Kondisi *First* dan *Last Variable* Pada *Circular Array (lanj.)*

- Setelah *queue* terisi, *normally*

  If empty() then First=0 //set to the 1st array

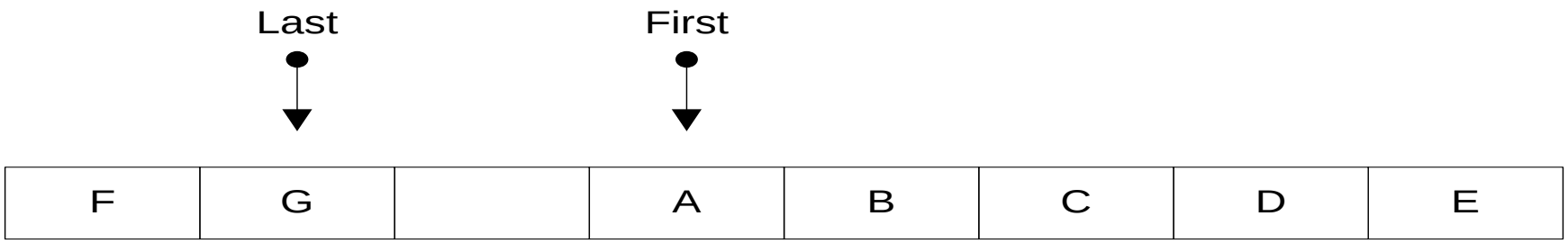
  Last = First + Data Size - 1



2.2. Kondisi *First* dan *Last Variable* Pada *Circular Array (lanj.)*

- *Last variable* dapat berisi nilai yang lebih rendah daripada *First* jika *First value* lebih dari 0 dan *First* + Data *Size* – 1 lebih besar dari *Array Size*.
- Kedua kondisi di atas dapat diterapkan dengan notasi

  Last = ((First + Data Size - 1) mod Array Size)



2.3. *Queue Operation With Array*

- clear() →
  - Isi *First* dan *Last* dengan -1 *value* dan data *size* dengan 0
- isEmpty() →
  - Periksa *First* or *Last* value, jika berisi -1 maka berarti *queue* kosong,
  - Atau periksa data *size*, jika berisi 0 maka *queue* kosong

2.3. *Queue Operation With Array* (Lanj.)

- enqueue(el) →
  - *Set First value* dengan 0 jika kosong.
    if isEmpty() then First = 0;
  - *Increment Data Size value*
    Inc(Data Size);
  - *Set Last value* dengan
    Last = (First + Data Size - 1) mod Array Size
  - Kemudian masukkan *element* el ke sel *array* pada *offset Last*
    Pool[Last] = el

2.3. *Queue Operation With Array* (Lanj.)

- dequeue →
  - Ambil *value* dari Pool[First]
  - Hapus data pada Pool[First]
  - *Decrement DataSize*
  - *Set next value* dari variabel *First*

    First = (First + 1) mod ArraySize
- firstEl() →
  - Ambil data yang terdapat Pool[First]

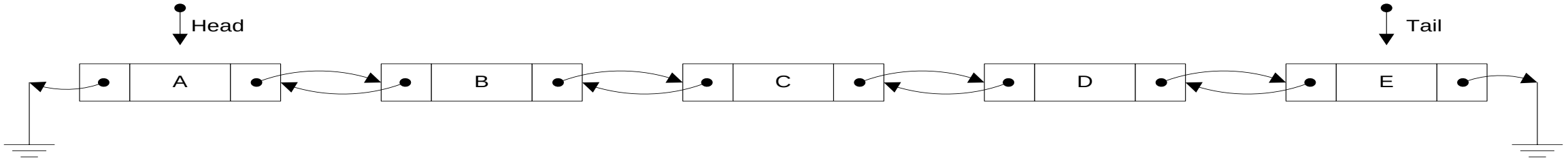
3. *Linked-List Base Queue*

3.1. Penerapan *Queue* dengan *Linked List*

- Hanya diperlukan *Double Linked List* dan cara yang mudah untuk mendapatkan *queue* pertama dan terakhir
- (jika menggunakan *single linked list*, akan sedikit lebih repot pada saat melakukan dequeue, pada first step harus menyimpan *current head* ke *temporary variable*, untuk membebaskan *current head* dengan mudah)
- *First queue* dapat diterapkan dengan *head*
- *Last queue* dapat diterapkan dengan *tail*

  (atau kebalikannya: *head* sebagai *Last queue* dan *tail* sebagai *First queue*, tapi harus punya *pointer* untuk menunjuk ke *previous tail* – dapat diterapkan dengan *double linked list* or *single list* dengan informasi *Previous Tail*)

3.1. Penerapan *Queue* dengan *Linked List* (Lanj.)



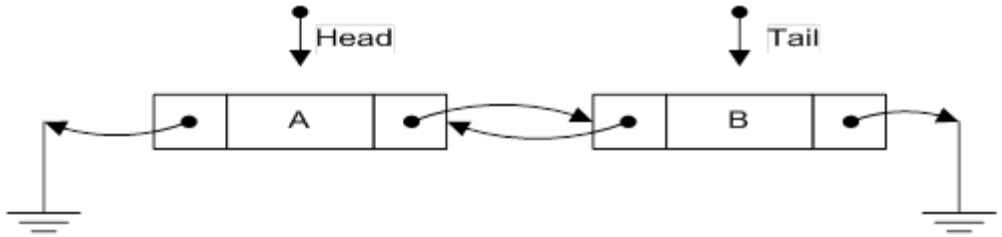
3.2. *Queue Operation* dengan *Linked List*

- clear() →
  - Hancurkan setiap node
  - set *Head* dengan NULL
- isEmpty() →
  - Periksa jika *Head* == NULL, maka queue is empty
- enqueue(el) →
  - Buat node baru pada *tail*,
  - Isi *element* el pada node baru.
- Dequeue →
  - Ambil *element* dari node yang ditunjuk oleh *Head*,
  - *set next* Node sebagai *Head*
  - *destroy* node yang semula *Head*
- firstEl() →
  - Ambil *element* dari node *Head*.

3.2. *Queue Operation* dengan *Linked List* (Lanj.)

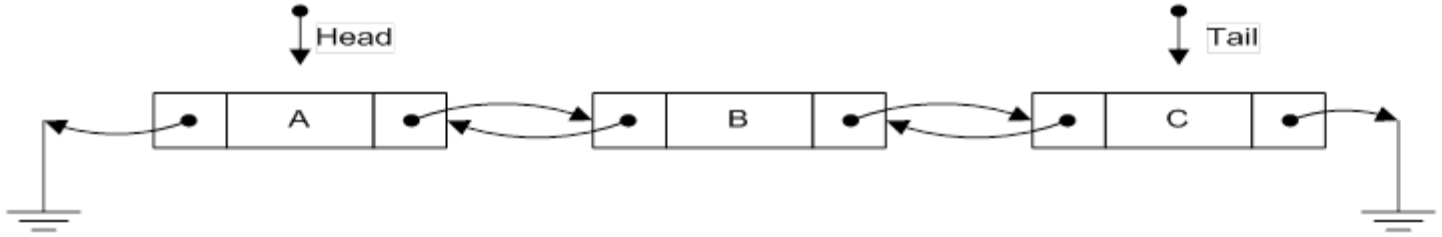
enqueue()

- Buat node baru pada *tail*,
- Isi *elem*



After enqueue(A)

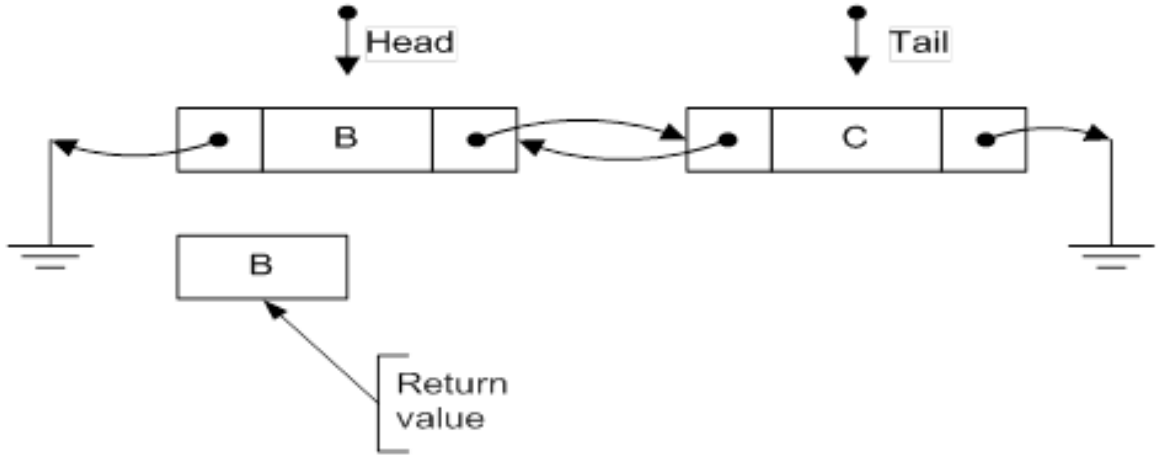
After enqueue(B)



After enqueue(C)

3.2. *Queue Operation* dengan *Linked List* (Lanj.)

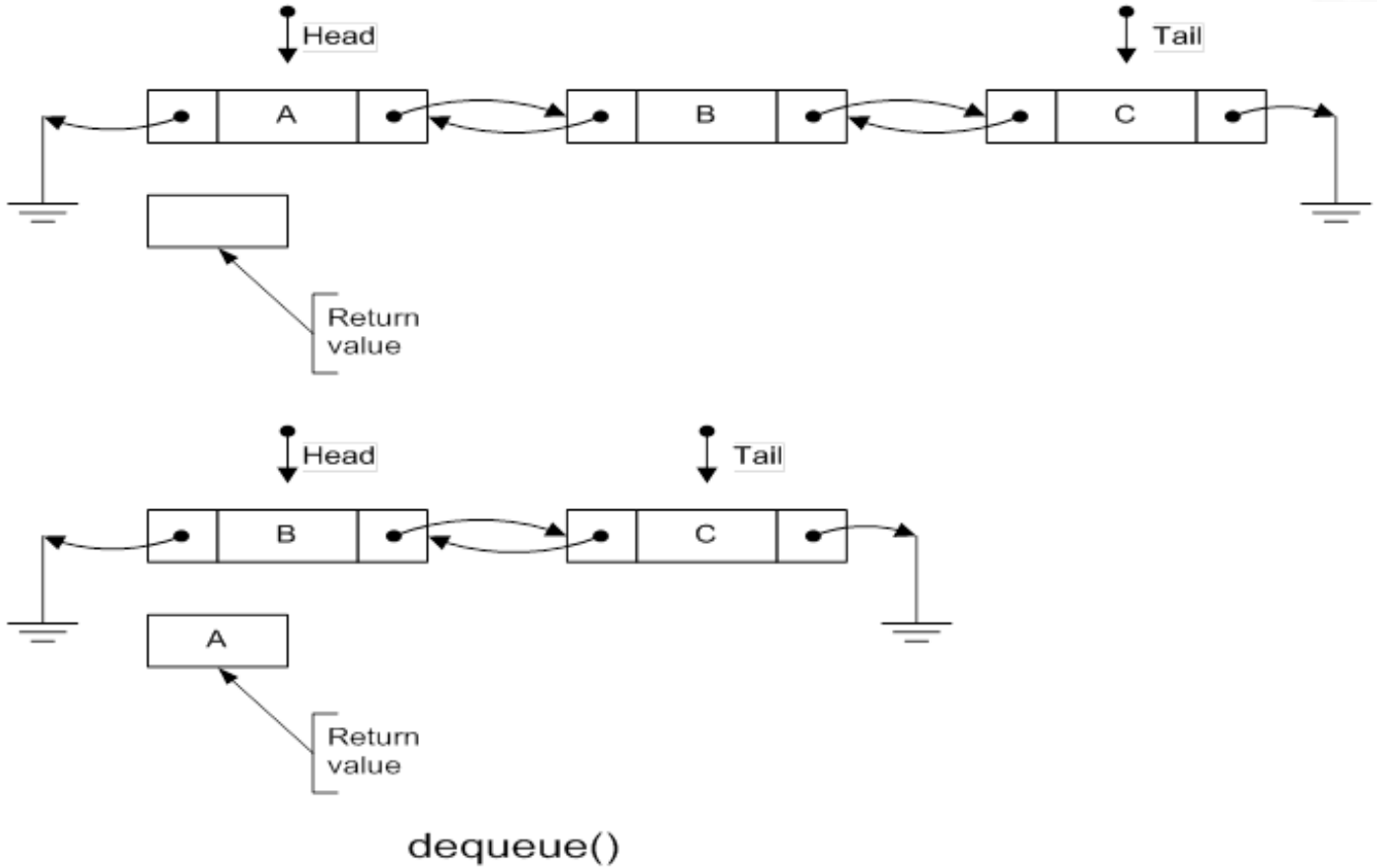
- firstEl() →
  - Ambil *element* dari node *Head*.



firstEl()

3.2. *Queue Operation* dengan *Linked List* (Lanj.)

- Dequeue →
  - Ambil *element* dari node yang ditunjuk oleh *Head*,
  - *set next* Node sebagai *Head*
  - *destroy* node yang semula *Head*



3.3. Tambahan

- Untuk penggunaan *Single List*, ketika melakukan dequeue,
  - anda harus menyimpan dahulu alamat *head* saat ini pada variabel temp.
  - Setelah data diambil dari *queue*, arahkan head ke node berikutnya.
  - Dan *free* kan node yang sebelumnya merupakan *head* yang sebelumnya yang sekarang ditunjuk oleh *variable* temp di atas
- Untuk penggunaan *double linked list*, anda cukup mengarahkan *head* ke node berikutnya, kemudian *free* kan node yang sebelumnya merupakan *head* yang sekarang ditunjuk oleh *Head*->Prev

Ringkasan

- *Queue* adalah Struktur Data Linear dimana penambahan elemen dilakukan pada akhir dan mengeluarkan *element* dari depan (*First In First Out*). berisi operasi : clear(), isEmpty(), enqueue(el), dequeue() dan firstEl()
- *Queue* dapat diterapkan dengan dua cara
  - *Linked-List Base Queue*--> implementasi *queue* menggunakan *linked-list*
  - *Array Base Queue* --> implementasi *queue* 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 *sebelum;
  struct TheCell *berikut;
};
struct TheCell *ptrCell=NULL;
struct TheCell *first=NULL;
struct TheCell *last=NULL;
```

Contoh Program (Lanj.)

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

Contoh Program (Lanj.)

```
int dequeue()
{
  if (first!=NULL)
  {
    int getData;
    getData = last->dat;
    ptrCell = last;
    last = last->sebelum;
    free(last->berikut);
    last->berikut=NULL
    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();
    enqueue(bilRandom);
  }
}
```

Terimakasih

TUHAN Memberkati Anda

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