Kode_Asisten_PJ_:_MTN Kelompok = A11.4301U

NIM :A11.2012.06758 NIM : A11.2012.06877

Nama :Muhammad Adhi D Nama : Dwi Lestari Aprilyani.

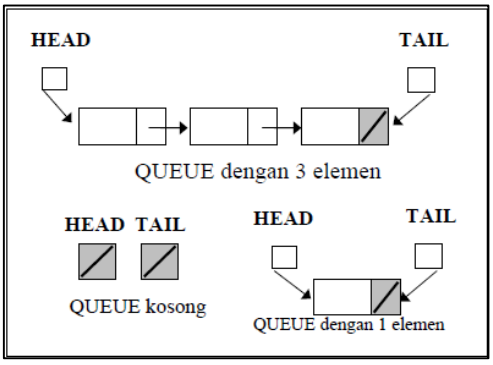
PSDA – 12.b (Priority Queue).

A. Pendahuluan Mengenai Priority Queue

Priority Queue merupakan "List Linier" yang dikenali elemen pertamanya (HEAD), dan elemen terakhirnya (TAIL) dan dikenali elemen Prioritasnya (PRIO).Aturan penambahan dan penghapusan elemen yaitu :

- Penambahan bisa dilakukan pada HEAD, TAIL , atau di tengah elemen, berdasarkan urutan dari Prioritasnya.
- Penghapusan selalu dilakukan pada HEAD. Dimana berisi prioritas paling tinggi atau rendah.

Elemen Queue tersusun secara FIFO (*First In First Out*).Queue dapat digambarkan sebagai sebuah lorong lurus, dimana terdapat 2 pintu, sebagai tempat masuk dan keluar dari elemen (seperti dalam antrian berdasarkan prioritas).



Gambar di samping menunjukan kondisi Priority Queue dengan 3 elemen , di mana setiap kali dilakukan penambahan elemen akan dilakukan pengecekan perbandingan prioritas dari setiap elemen. Otomatis priority yang ada pada Queue akan selalu terurut dengan urutan "Ascending" atau "Descending".

Kondisi kosong dipenuhi jika HEAD dan TAIL bernilai NULL.Sedangkan kondisi 1 elemen , nilai dari HEAD dan TAIL akan menunjuk pada elemen atau addres yang sama.

```
typedef struct tElemen
{
    address NEXT;
    int PRIORITY;
    InfoType INFO;
}Elemen;

typedef struct
{
    address HEAD;
    address TAIL;
}P_QUEUE;
```

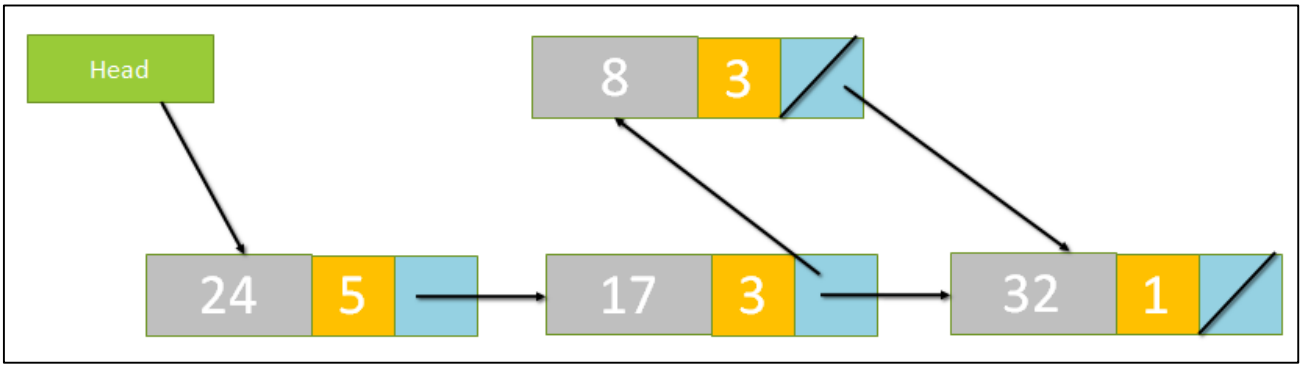
ADT disamping menunjukan struktur ADT pada Priority Queue . terlihat perbedaan hanya pada ADT dengan nama "elemen" . pada ADT "elemen" ditambahan informasi baru yaitu "PRIORITY". Dimana variabel tersebut nantinya akan berfungsi sebagai penampung nilai dari prioritas setiap elemen.

B. Penjelasan Fungsi Penambahan. *(Add)*

Aturan penambahan pada bab ini, berbeda dengan penambahan pada QUEUE biasa, perbedaanya adalah pada urutan prioritasnya , sehinnga saat dilakukan penyisipan elemen maka akan dilakukan perbandingan priority antar setiap elemen.

Algoritmanya adalah sebagai berikut :

- Cek apakah list kosong? ,jika ya lakukan penambahan elemen seperti biasa , dengan posisi TAIL dan HEAD menunjuk pada elemen yang baru.
- Kemudian , cek apakah jika Queue hanya terdiri 1 elemen?, jika ya maka , lakukan perbandingan nilai "Prioritas" pada elemen yang baru dengan elemen yang ada pada Queue (HEAD).
  - ✓ Kondisi 1 : jika *(Prio pada elemen baru >= Prio pada HEAD)* Maka lakukan penambahan elemen pada elemen pertama (Insert_First).
  - ✓ Kondisi 2 : jika *(Prio pada elemen baru > Prio pada HEAD) Maka lakukan penambahan elemen pada elemen terakhir (Inser_Last))*
- Cek apakah elemen pada Queue jumlahnya >= 2?, jika ya maka lakukan perbandingan nilai "Prioritas" pada elemen yang baru dengan elemen HEAD.
  - ✓ Kondisi 1 : jika *(Prio pada elemen baru >= Prio pada HEAD)* Maka lakukan penambahan elemen pada elemen pertama (Insert_First).
  - ✓ Kondisi 2 : lakukan perulangan dengan while selama kondisi yang ada pada Queue > Prio yang ada pada elemen baru . maka QUEUE maju 1 langkah .lakukan terus pengulangan tersebut sampai kondisi Prio(Queue) < Prio(P). Jika kondisi tersebut terpenuhi maka lakukan penyisipan elemen, missal.



C. Penjelasan Fungsi Penghapusan *(Del)*

Pada fungsi penghapusan ini system kerjanya sama dengan peghapusan yang ada pada QUEUE biasa. Dengan asumsi pada List Linier adalah fungsi Del_First.

Penghapusan selalu dilakukan pada elemen HEAD (elemen pertama), dengan prioritas tertinggi yang terletak pada HEAD.

```
void Del (P_QUEUE *Q,InfoType *X)
{
    address P;

    if (NEXT(HEAD(*Q))==NIL)
    {
        P = HEAD(*Q);
        *X = INFO(P);
        HEAD(*Q) = NIL;
        TAIL(*Q) = NIL;
        DEALOKASI(P);
    }
    else
    {
        P = HEAD(*Q);
        *X = INFO(P);
        HEAD(*Q) = NEXT(HEAD(*Q));
        DEALOKASI(P);
    }
}
```

. Pada gambar di samping dilakukan pengecekan kondisi apakah QUEUE tersebut hanya ada 1 elemen? , jika ya maka kosongan elemen tersebut dengan cara TAIL dan HEAD dikosongkan (NULL).

. Sedangkan kondisi ke 2, adalah hapus pada elemen HEAD, sehingga NEXT(HEAD) , akan menjadi HEAD yang baru.Kemudian dealokasikan address yang akan dihapus tadi.