BAB III
QUEUE (ANTRIAN)

3.1 Pengertian Antrian

Antrian (Queue) merupakan kumpulan data yang mana penambahan elemen hanya bias dilakukan pada suatu ujung yaitu *rear* /*tail* / belakang, dan pengha[usan dilakukan melalui ujung yang lainnya yaitu *front* / *head* / depan. Antrian disebut FIFO (*First In First Out*) yaitu elemen yang lebih dulu disisipkan merupakan elemen yang akan lebih dulu diambil.

Operasi-operasi dasar dari sebuah queue adalah :

1. Enqueue : proses penambahan elemen di posisi belakang
2. Dequeue : proses pengambilan elemen di posisi depan

Selain operasi dasar di atas, ada pula operasi-operasi lain yang dapat dilakukan terhadap sebuah queue yaitu :

1. Operasi pemeriksaan queue kosong (fungsi kosong)
2. Operasi pemeriksaan queue penuh (fungsi penuh).
3. Operasi inisialisasi queue (fungsi inisialisasi)

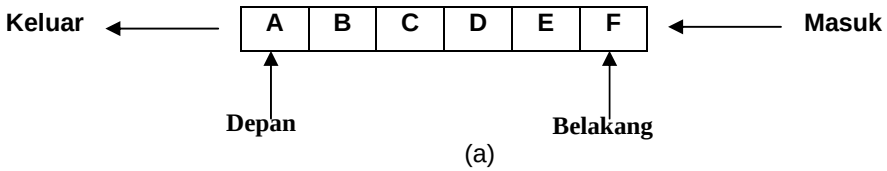
3.2 Karakteristik Antrian

Karakteristik antrian sebagai berikut :

1. Elemen antrian yaitu item-item data yang terdapat di elemen antrian
2. Front (elemen terdepan dari antrian)
3. Tail (elemen terakhir dari antrian)
4. Count (jumlah elemen pada antrian)
5. Status antrian apakah penuh atau kosong.
   - Penuh, jika elemen pada antrian mencapai kapasitas maximum antrian. Pada kondisi ini, tidak mungkin dilakukan panambahan ke antrian.
   - Kosong, jika tidak ada elemen pada antrian. Pada kondisi ini, tidak mungkin dilakukan pengambilan elemen dari antrian.

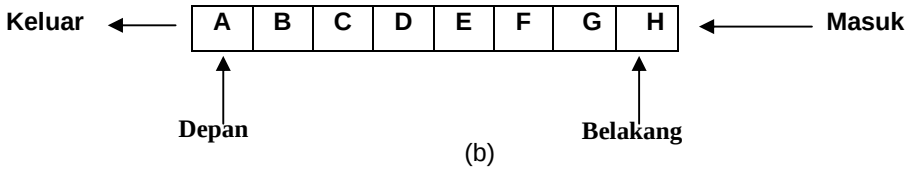
3.3 Implementasi Antrian

3.3.1 Dengan menggunakan Array Statis

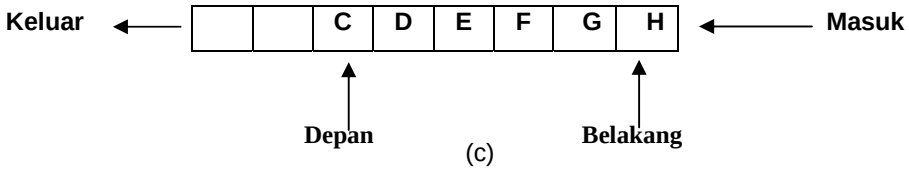


(a)

Jika ada elemen baru yang akan masuk pada gambar (a), maka ia akan diletakkan disebelah kanan F (gambar (b)). Jika ada elemen yang akan dihapus, maka A akan dihapus lebih dulu (gambar (c)).



(b)



(c)

Contoh deklarasi antrian :

Const max = 5

Var antrian : array [1..max] of char;

Belakang, depan : integer; x : char;

Dengan menggunakan array, maka overflow dapat terjadi jika antrian telah penuh , sementara masih ingin menambah elemen ke dalam antrina. Dengan mengabaikan adanya overflow, maka penambahan elemen baru yang dinyatakan oleh var x dapat diimplementasikan dengan statemen :

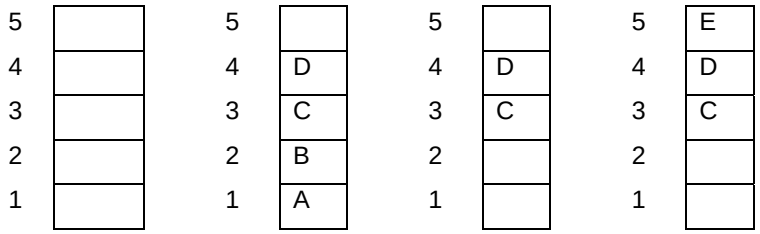
belakang := belakang + 1;

antrian [belakang] := x;

Operasi penghapusan bias diimplementasikan dengan ;

x := antrian [depan];

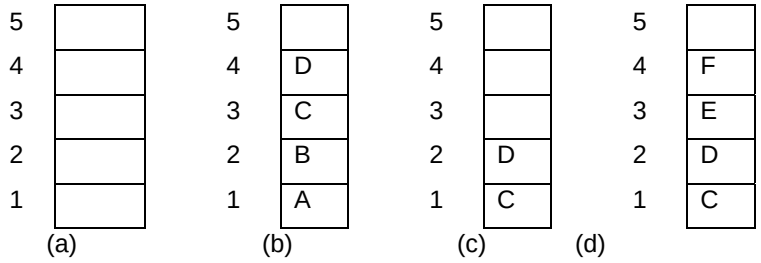
depan := depan + 1;



Pada gambar (a) posisi depan = 1 dan belakang = 0. Pada gambar (b) keadaan setelah penambahan empat buah elemen dimana posisi depan = 1 dan belakang = 4. Pada gambar (c) keadaan setelah penghapusan dua buah elemen dimana posisi depan = 3 dan belakang = 4. Pada gambar (d) keadaan setelah penambahan dua buah elemen diamana posisi depan = 3 dan belakang = 5.

Jika akan ditambah elemen baru, maka nilai belakang harus ditambah satu menjadi 6. Sedangkan larik antrian tersebut hanya terdiri dari 6 elemen sehingga tidak bias ditambah lagi meskipun sebenarnya larik tersebut masih kosong di dua tempat. Oleh karena itu dilakukan dengan metoda penggeseran dimana jika ada elemen yang dihapus, maka semua elemen lain digeser sehingga antrian selalu dimulai dari depan = 1.

```
x := antrian [1];
for i := 1 to belakang-1 do
begin
  antrian [ i] := antrian [i +1];
end;
```



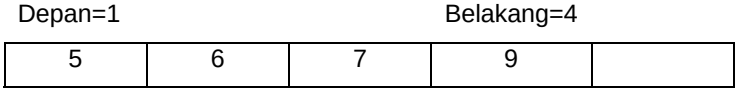
Pada gambar (a) posisi depan = 1 dan belakang = 0. Pada gambar (b) keadaan setelah penambahan empat buah elemen dimana posisi depan = 1 dan belakang = 4. Pada gambar (c) keadaan setelah penghapusan dua buah elemen dimana posisi depan = 1 dan belakang = 2. Pada gambar (d) keadaan setelah penambahan dua buah elemen dimana posisi depan = 1 dan belakang = 4.

Cara penggeseran elemen tidak efisien untuk larik berukuran besar. Oleh karena itu dilakukan dengan larik yang menyimpan elemen antrian sebagai larik memutar (*circular*).

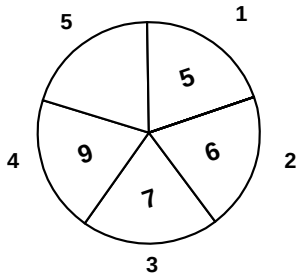
Queue Dengan Circular Array

Jika menggunakan array untuk queue seperti di atas, maka ketika ada proses pengambilan (dequeue) ada proses pergeseran data. Proses pergeseran data ini pasti memerlukan waktu apalagi jika elemen queue-nya banyak. Oleh karena itu solusi agar proses pergeseran dihilangkan adalah dengan metode circular array.

Queue dengan circular array dapat dibayangkan sebagai berikut :



Atau agar lebih jelas, array di atas dapat dibayangkan ke dalam bentuk seperti lingkaran di bawah ini.



depan = 1
belakang = 4

Aturan-aturan dalam queue yang menggunakan circular array adalah :

1. Proses penghapusan dilakukan dengan cara nilai depan (front) ditambah 1 : depan=depan + 1.

2. Proses penambahan elemen sama dengan linear array yaitu nilai belakang ditambah 1 : belakang=belakang + 1.
3. Jika depan = maks dan ada elemen yang akan dihapus, maka nilai depan = 1.
4. Jika belakang = maks dan depan tidak 1 maka jika ada elemen yang akan ditambahkan, nilai belakang=1
5. Jika hanya tinggal 1 elemen di queue (depan = belakang), dan akan dihapus maka depan diisi 0 dan belakang diisi dengan 0 (queue kosong).

Front dan Tail akan bergerak maju, jika ;

1. Untuk penambahan.
2. Tail sudah mencapai elemen terakhir array akan memakai elemen pertama array yang telah dihapus.
3. Untuk pngahapusan.
4. Front telah mencapai elemen terakhir array, maka akan menuju elemen pertama jika antrian masih berisi elemen.

Keunggulan representasi circular adalah seluruh kapasitas antrian bisa terpakai seleruhnya. Berdasarkan ilustrasi sebelumnya dapat disusun prosedur untuk menambah dan menghapus elemen antrian sebagai berikut ini :

```
Const max_elemen = 5;
Type antri = array [1..max_elemen] of char;
Var antrian : antri;
    depan, belakang : integer;
```

Nilai awal untuk depan dan belakang masing-masing 0.

```
Depan := 0; Belakang := 0;
```

Proses enqueue data hanya bisa dilakukan jika queue tidak kosong. Ada beberapa hal yang harus diperhatikan ketika akan melakukan enqueue pada circular array, diantaranya adalah :

- Kondisi ketika queue masih kosong. Jika ini terjadi, maka posisi depan dan belakang bernilai 0.
- Ketika ketika nilai belakang sama dengan maks_queue, maka posisi belakang adalah 0. Ini terjadi ketika posisi depan lebih besar dari 0 (pernah ada dequeue).
- Ketika nilai belakang masih lebih kecil dari maks_queue maka posisi belakang ditambah 1 : belakang=belakang+1;

Dari ketentuan-ketentuan di atas dapat diimplementasikan operasi Enqueue-nya dalam bahasa Pascal adalah sebagai berikut :

```
procedure Enqueue (var q : antri; x : char);
begin
    if belakang = max_elemen then belakang := 1
    else belakang := belakang + 1;
    if depan = belakang  then
```

```
    begin
        write (‘ antrian sudah penuh’);
         belakang := belakang – 1;
         if belakang = 0 then
           belakang :=max_elemen;
    end
    else q[belakang] := x;
end;
```

Sedangkan proses dequeue hanya bisa dilakukan jika queue dalam keadaan tidak kosong. Ada beberapa kondisi yang harus diperhatikan ketika dequeue elemen queue yaitu :

- Kondisi ketika posisi depan sama dengan posisi belakang (queue hanya memiliki 1 elemen) maka nilai depan dan belakang diisi dengan 0 (queue kosong).
- Jika posisi depan sama dengan posisi maks_queue maka posisi depan berpindah ke 1.
- Selain itu, posisi depan ditambah dengan 1 : depan=depan+1

  Impelementasinya dalam bahasa Pascal adalah :

```
function dequeue (var q : antri) : char;
begin
    if (depan=0) and 9belakang=0) then then
       write (‘ antrian kososng’);
    else
    begin
        dequeue := q [depan];
        if depan = max_elemen  then
           depan := 1
        else
           depan := depan + 1;
    end;
end;
```