Queue - Antrian

4.1. Definisi *Queue*

Antrian (*Queue*) adalah struktur data yang penting dan sering ditemukan dalam mayoritas perangkat lunak mulai dari mode pengguna hingga mode kernel aplikasi. *Queue* juga termasuk *Abstract Data Type* (ADT) seperti halnya *Stack*, namun jika pada *Stack* salah satu "ujung" seolah-olah tertutup sedangkan *Queue* kedua "ujung" tempat penyimpanan datanya terbuka sehingga memungkinkan terjadi operasi dikedua ujung. *Queue* menerapkan prinsip *First In First Out* (FIFO) dimana elemen/data yang pertama masuk ke dalam antrian akan menjadi yang pertama diakses, elemen/data kedua ditambahkan ke antrian akan menjadi yang kedua diakses dan seterusnya. Contoh sebuah antrian dalam kehidupan sehari-hari diantaranya antrian kendaraan diterminal, deretan pembeli tiket pertunjukan atau nasabah sebuah bank yang menunggu dilayani seperti pada gambar 4.1 berikut ini

Gambar 4.1 Contoh Antrian di kehidupan sehari-hari

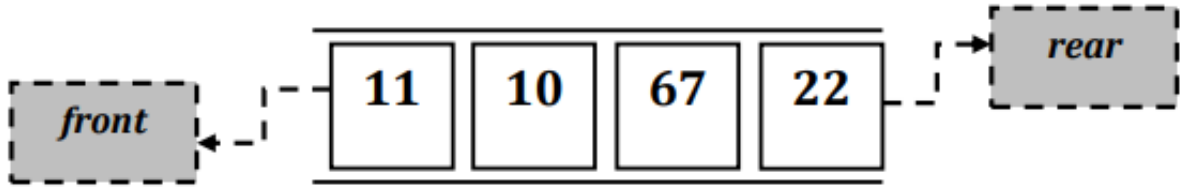
Sumber: http://rumahfilsafat.com/2013/09/21/antri-donk-pak-bu-mas-mbak-2/

Penerapan *Queue* dalam sistem komunikasi dan komputer diantaranya:

1. Melayani permintaan layanan sumber daya bersama tunggal (*single shared resource*) seperti printer, *CPU task scheduling* (penjadwalan).
2. Telepon *Call Center* akan menggunakan antrian, untuk menahan orang-orang panggilan dan melayani sesuai urutan.
3. Penanganan interupsi dalam sistem *real-time*.

Antrian dapat diimplementasikan menggunakan *Array atau Linked List*. Selayaknya *stack* penerapan *Queue* menggunakan *linked list* akan mengakibatkan ukurannya dinamis dibandingkan dengan menggunakan *array*. Operasi yang umum terdapat dalam sebuah antrian ada dua, yaitu ***Enqueue*** operasi menempatkan elemen baru di belakang antrian dan ***Dequeue*** untuk mengambil elemen di depan antrian

serta menghapusnya dari antrian. Di sistem antrian (*Queue*) untuk menyatakan ujung data lazimnya disebut dengan istilah *rear* untuk data bagian belakang atau yang terakhir masuk dan *front* untuk data yang masuk pertama.



Gambar 4.2 Ilustrasi Antrian (*Queue*)

Listing Program 4.1 menunjukan pada kita bagaimana sebuah antrian (*queue*) dibuat. Deklarasi variabel `front` dan `rear` sama-sama diberikan nilai NULL.

```
/* Membuat queue */
void create()
{
    front = rear = NULL;
}
```

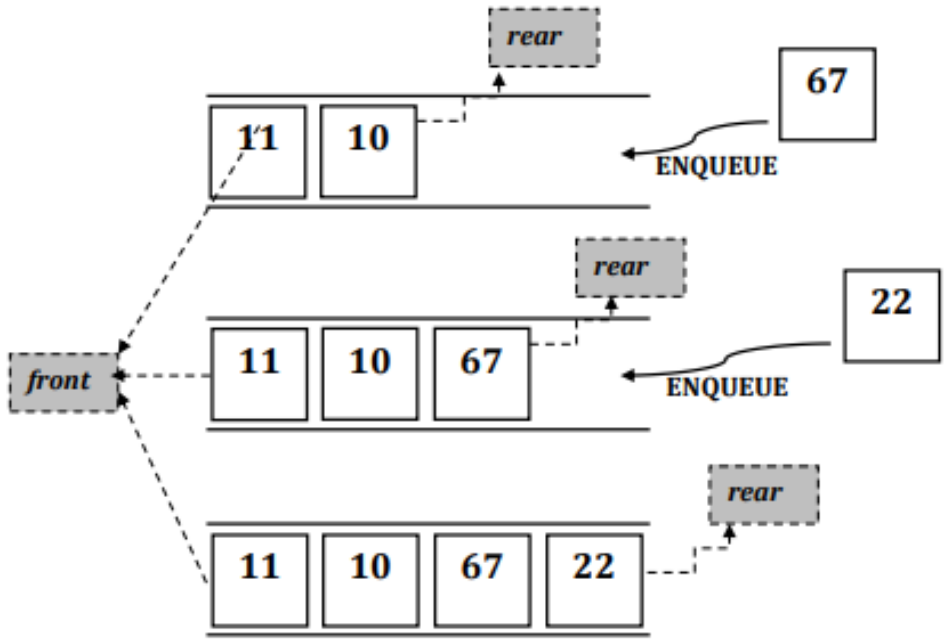
Listing Program 4.1 Pembuatan Queue

4.2. Operasi *Enqueue*

Enqueque merupakan perintah dasar untuk menambahkan elemen/data ke dalam antrian. Berikut ini langkah-langkah operasi *Enqueue*

a. Periksa apakah antrian penuh .
b. Jika antrian penuh, berikan tanda *overflow error* dan keluar .
c. Jika antrian tidak penuh, arahkan pointer *next untuk menunjuk ruang kosong di sebelah .
d. Tambahkan elemen data ke lokasi antrian, lalu pindahkan *rear* ke lokasi elemen baru.
e. kembali ke langkah awal.

Perhatikan Gambar 4.3, di dalam Antrian (*queue*) dikenal istilah ***front*** untuk menandai data yang pertama masuk (depan) dan ***rear*** sebagai penanda data terakhir yang masuk. Selama operasi yang berlangsung adalah operasi ENQUEUE maka variabel ***front*** akan tetap menunjuk pada data yang sama, sehingga terjadi perubahan hanya pada variabel ***rear***. Diawal *queue* telah memiliki dua data yaitu 11 dan 10, maka ***rear*** menunjuk data 10 karena data tersebut sebagai data terakhir yang masuk. Selanjutnya ketika data baru 67 dimasukan ke dalam Queue dengan operasi ENQUEUE, maka ***rear*** berubah menunjuk data 67, begitu seterusnya.



Gambar 4.3 Ilustrasi Enqueue

Berikutnya mari kita lihat Listing Program 4.2 tentang implementasi operasi ENQUEUE untuk linked list.

```
/* Memasukan data dalam queue */

void enq(int data)
{
    if (rear == NULL)
    {
        rear = (struct node *)malloc(1*sizeof(struct node));
        rear->ptr = NULL;
        rear->info = data;
        front = rear;
    }

    else
    {
        temp=(struct node *)malloc(1*sizeof(struct node));
        rear->ptr = temp;
        temp->info = data;
        temp->ptr = NULL;

        rear = temp;
    }
    count++;
}
```

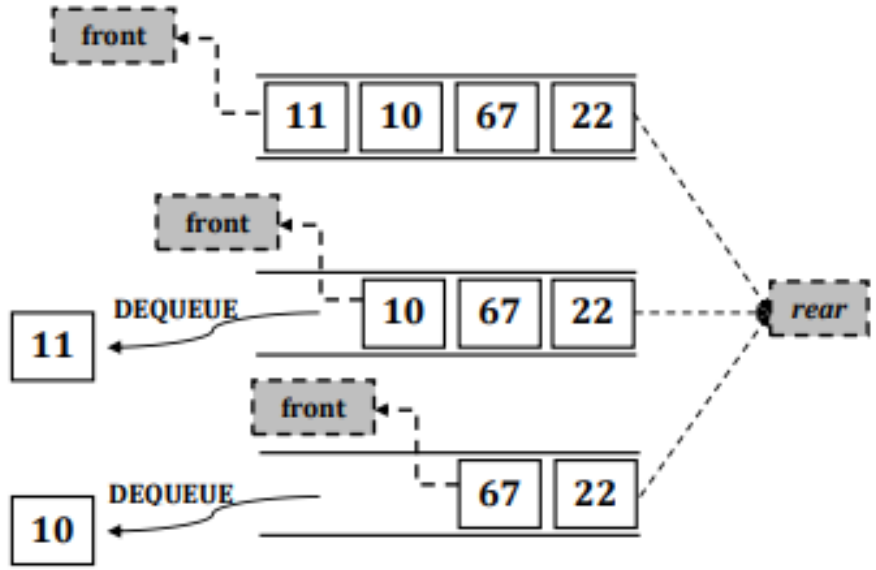
Listing Program 4.2 Operasi Enqueue

4.3. Operasi *Dequeue*

Mengakses data dengan operasi DEQUEUE adalah proses yang menangani dua tugas sekaligus, yaitu merubah orientasi variabel *front* untuk mengakses data selanjutnya dan menghapus data yang diakses. Langkah-langkah berikut ini yang dilakukan oleh operasi dequeue:

a. Periksa apakah antrian kosong .
b. Jika antrian kosong , tampilkan pesan *error* dan keluar .
c. Jika antrian tidak kosong , akses data yang ditunjuk variabel *front* depan.
d. Lalu tambahkan 1 (naik 1) pada variabel *front* agar menunjuk elemen data yang berikutnya .
e. kembali sukses .

Perhatikan Gambar 4.4 sebagai ilustrasi operasi DEQUEUE, di dalam Antrian (*queue*). Diawal *queue* telah memiliki empat data yaitu 11, 10, 67 dan 22 maka ***rear*** menunjuk data 22 karena data tersebut sebagai data terakhir yang masuk. Selanjutnya ketika akan melakukan operasi DEQUEUE, maka variabel ***front*** akan menunjuk data 10, data 11 dikelurkan dari antrian (*Queue*), begitu seterusnya. Selama operasi yang berlangsung adalah operasi DEQUEUE maka variabel ***rear*** akan tetap menunjuk pada data yang sama, sehingga terjadi perubahan hanya pada variabel ***front.***



Gambar 4.4 Ilustrasi Dequeue

Listing Program 4.3 merupakan baris-baris kode untuk Operasi DEQUEUE. Hal pertama yang harus dilakukan dalam proses ini adalah menduplikasi variabel `front`, Misal kita membuat variabel baru `front1`. Perlu dilakukan pengecekan apakah Queue kosong atau ada isinya, jika kosong maka keluar. Jika Queue telah terisi, maka proses mengeluarkan data dilakukan seperti pada penjelasan untuk Gambar 4.4 diatas.

```
/* Mengeluarkan data dari Queue */
void deq()
{
    frontl = front;

    if (frontl == NULL)
    {
        printf("\n Tidak ada data dalam queue ");
        return;
    }
    else
        if (frontl->ptr != NULL)
        {
            frontl = frontl->ptr;
            printf("\n Nilai yang di-Deque : %d", front->info);
            free(front);
            front = frontl;
        }
        else
        {
            printf("\n Nilai yang di-Deque : %d", front->info);
            free(front);
            front = NULL;
            rear = NULL;
        }
        count--;
}
```

Listing Program 4.3 Operasi Dequeue

4.4. Display Queue

Selain Fungsi untuk Operasi Enqueue dan Dequeue, ada fungsi lain yang penting dalam teknik QUEUE, yaitu fungsi yang digunakan untuk menampilkan data yang ada dalam antrian. Hal ini diperlukan agar kita dapat mengetahui isi dalam antrian. Listing Program 4.4 merupakan kode program dalam Bahasa C untuk menampilkan

isi Antrian. Alur program hampir sama dengan operasi Dequeue hanya saja yang dilakukan adalah perintah `printf` dan tidak dilakukan perintah `free(front);` agar data tidak terhapus dalam memori.

```
/* Menampilkan data dalam Queue */
void display()
{
    frontl = front;

    if ((frontl == NULL) && (rear == NULL))
    {
        printf("Queue Kosong");
        return;
    }
    while (frontl != rear)
    {
        printf("%d ", frontl->info);
        frontl = frontl->ptr;
    }
    if (frontl == rear)
        printf("%d", frontl->info);
}
```

Listing Program 4.4 Menampilkan data dalam Queue