Pertemuan 9

Waktu : 135 menit

Tujuan Pembelajaran : Mahasiswa mampu menjelaskan teknik pemrograman menggunakan Queue.

Substansi Materi : Queue

Tabulasi Kegiatan Perkuliahan

No	Tahap Kegiatan	Kegiatan Pengajar	Kegiatan Mahasiswa	Media & Alat	Waktu
1	Pendahuluan	1. Membuka pertemuan 2. Mengulang materi pertemuan sebelumnya	Menyimak Bertanya	Papan Tulis	20 Menit
2	Penyajian Materi	1. Queue 2. Queue dengan array 3. Queue dengan circular array	Menyimak Bertanya Menjawab Pertanyaan	Papan Tulis	80 Menit
3	Penutup	1. Menyimpulkan materi pertemuan 2. Memberikan tugas kecil 3. Menutup pertemuan	Menyimak	Papan tulis	35 Menit

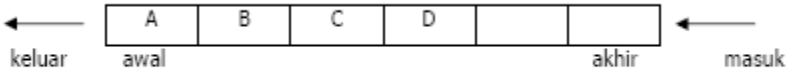
MATERI KULIAH

QUEUE / ANTRIAN

Secara harfiah queue dapat diartikan sebagai antrian. Queue merupakan kumpulan data dengan penambahan data hanya melalui satu sisi, yaitu belakang (tail) dan penghapusan data hanya melalui sisi depan (head). Berbeda dengan stack yang bersifat LIFO maka queue bersifat FIFO(First In First Out), yaitu data yang pertama masuk akan keluar terlebih dahulu dan data yang terakhir masuk akan keluar terakhir. Berikut ini adalah gambaran struktur data queue.



Gambar 1. Ilustrasi Queue 1



Gambar 2. Ilustrasi Queue 2

Elemen yang pertama kali masuk ke dalam queue disebut elemen depan (front/head of queue), sedangkan elemen yang terakhir kali masuk ke queue disebut elemen belakang (rear/tail of queue). Perbedaan antara stack dan queue terdapat pada aturan penambahan dan penghapusan elemen. Pada stack, operasi penambahan dan penghapusan elemen dilakukan di satu ujung. Elemen yang terakhir kali dimasukkan akan berada paling dekat dengan ujung atau dianggap paling atas sehingga pada operasi penghapusan, elemen teratas tersebut akan dihapus paling awal, sifat demikian dikenal dengan LIFO. Pada queue, operasi tersebut dilakukan di tempat yang berbeda. Penambahan elemen selalu dilakukan melalui salah satu ujung, menempati posisi di belakang elemen-elemen yang sudah masuk sebelumnya atau menjadi elemen paling belakang. Sedangkan penghapusan elemen dilakukan di ujung yang berbeda, yaitu pada posisi elemen yang masuk paling awal atau elemen terdepan. Sifat yang demikian dikenal dengan FIFO.

Operasi-operasi standar pada queue adalah:

1. membuat queue atau inisialisasi.

2. mengecek apakah queue penuh.

3. mengecek apakah queue kosong.

4. memasukkan elemen ke dalam queue atau InQueue (Insert Queue).

5. Menghapus elemen queue atau DeQueue (Delete Queue).

Implementasi Queue dengan Linear Array

Disebut juga queue dengan model fisik, yaitu bagian depan queue selalu menempati posisi pertama array. Queue dengan linear array secara umum dapat dideklarasikan sebagai berikut:

```
Const
        MaxQueue = {jumlah};
Type
        TypeQueue = byte;
Var
        Queue : Array[1..MaxQueue] of TypeQueuel
        Head, Tail      : Byte;
```

Operasi-operasi queue dengan linear array:

1. Create

   Procedure create berguna untuk menciptakan queue yang baru dan kosong yaitu dengan cara memberikan nilai awal (head) dan nilai akhir (tail) dengan nol (0). Nol menunjukan bahwa queue masih kosong.

```
Procedure Create;
Begin
    Head := 0; Tail := 0;
End;
```

2. Empty

   Function empty berguna untuk mengecek apakah queue masih kosong atau sudah berisi data. Hal ini dilakukan dengan mengecek apakah tail bernilai nol atau tidak, jika ya maka kosong.

```
Function Empty : Boolean;
Begin
    If Tail = 0 then
        Empty := true
    Else
        Empty := False;
End;
```

3. Full

```
Function Full : Boolean;
Begin
    If Tail = MaxQueue then
        Full := true
    Else
        Full := False
End;
```

4. EnQueue

Procedure EnQueue berguna untuk memasukkan 1 elemen ke dalam queue.

```
Procedure Enqueue(Elemen : Byte);
Begin
    If Empty then
    Begin
        Head := 1;
        Tail := 1;
        Queue[head] := Elemen;
    End
    Else
    If Not Full then
    Begin
        Inc(Tail);
        Queue[tail] := Elemen;
    End;
End;
```

5. DeQueue

   Procedure DeQueue berguna untuk mengambil 1 elemen dari Queue, operasi ini sering disebut juga SERVE. Hal ini dilakukan dengan cara memindahkan semua elemen satu langkah ke posisi di depannya, sehingga otomatis elemen yang paling depan akan tertimpa dengan elemen yang terletak dibelakangnya.

```
Procedure DeQueue;
Var I : Byte;
Begin
If Not Empty then
        Begin
        For I := Head to Tail-1 do
                Queue[I] := Queue[I+1];
                Dec(Tail);
        End;
End;
```

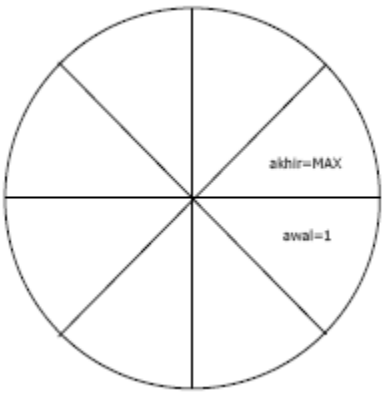
6. Clear

   Procedure clear berguna untuk menghapus semua elemen dalam queue dengan jalan mengeluarkan semua elemen tersebut satu per satu sampai kosong dengan memanfaatkan procedure DeQueue.

```
Procedure Clear;
Begin
        While Not Empty then
                DeQueue;
End;
```

Implementasi Queue dengan Circular Array

Salah satu variasi array adalah array melingkar (circular array), artinya array dapat diakses mulai dari sembarang indeks (indeks awal) ke arah indeks terakhir (maksimum array), lalu memutar ke indeks pertama hingga kembali ke indeks awal. Circular array adalah array yang dibuat seakan-akan merupakan sebuah lingkaran dengan titik awal dan titik akhir saling bersebelahan jika array tersebut masih kosong. Jumlah data yang dapat ditampung oleh array ini adalah besarnya ukuran array dikurangi 1. Misalnya besar array adalah 8, maka jumlah data yang dapat ditampung adalah 7.



Gambar 3. Ilustrasi Circular Array

Dengan circular array, meskipun posisi terakhir telah terpakai, elemen baru tetap dapat ditambahkan pada posisi pertama jika posisi pertama dalam keadaan kosong. Jika akhir=MAX dan awal=1, nilai head dan tail mencapai maksimum, maka akan dikembalikan ke posisi awal. Operasi-operasi yang terdapat pada circular array tidak jauh berbeda dengan operasi pada linear array.

Operasi-operasi queue dengan circular array:

1. Create

```
Procedure Create;
Begin
    Head := 1;
    Tail := MaxQueue;
End;
```

2. Empty

```
Function Empty : Boolean;
    Begin
        If (Tail Mod Max_Queue ) + 1  = Head then
            Empty := true
        Else
            Empty := False;
    End;
```

3. Full

```
Function Full : Boolean;
    Var
    X : 1 .. Max_queue;
    Begin
        X := (Tail Mod Max_queue)+1;
        If (x Mod Max_queue) + 1 = head then
            Full := True;
        Else
```

```
                Full := False;
        End;
```

4. EnQueue

```
        Procedure EnQueue (Elemen : TypeElemen);
        Begin
                If Not Full Then
                        Begin
                        Tail := (Tail Mod Max_queue) + 1 ;
                        Queue[Tail] := Elemen;
                        End;
        End;
```

5. DeQueue

```
        Procedure Dequeue;
        Begin
                If Not Empty then
                Head := (Head Mod Max_queue) + 1;
        End;
```