LAPORAN PRAKTIKUM XI

QUEUE

UEUE

Tujuan

Setelah mempelajari bab ini diharapkan mahasiswa akan mampu :

1. Menjelaskan pengertian queue
2. Memahami pemahaman tipe data abstrak antrian

DASAR TEORI

QUEUE

1.1 Definisi Queue

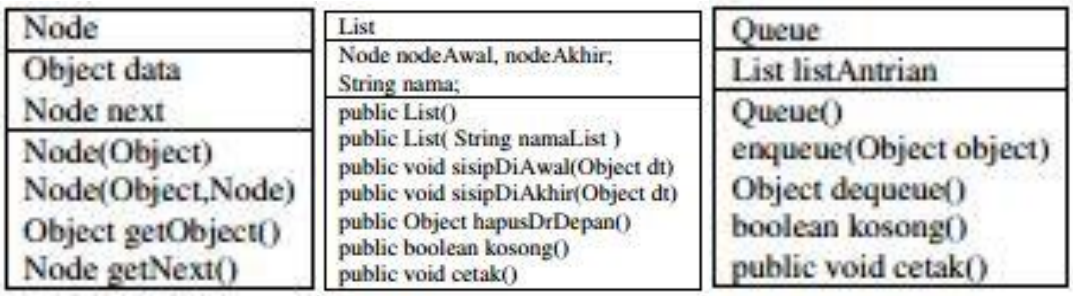
QUEUE adalah suatu kumpulan data yang mana penambahan data atau elemen hanya dapat dilakukan pada sisi belakang sedangkan penghapusan atau pengeluaran elemen dilakukan pada sisi depan. Antrian (Queue) dapat diartikan sebagai suatu kumpulan data yang seolah-olah terlihat seperti ada data yang diletakkan di sebelah data yang lain seperti pada gambar 01. Pada gambar, data masuk melalui lorong di sebelah kanan dan masuk dari terowongan sebelah kiri. Hal ini membuat antrian bersifat FIFO (First In First Out), beda dengan stack yang berciri LIFO.

Gambar 01

Antrian dapat dibuat baik dengan array maupun dengan struct. Pada pembuatan antrian dengan array, antrian yang disajikan bersifat statis. Ini disebabkan oleh jumlah maksimal array sudah ditentukan sejak deklarasi awal.

1.2 ADT Antrian

Dari ilustrasi gambar di atas ADT antrian dapat direpresentasikan sebagai berikut:

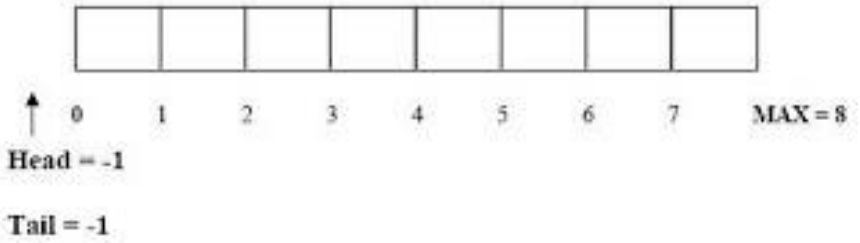


1.3 Manfaat Queue

- Digunakan sistem operasi untuk mengatur eksekusi taskdalam suatu sistem untuk mencapai perlakuan yang "adil" (seringkali queuedisebutwaiting line)
- Untuk mailbox dalam komunikasi antar proses
- Untuk buffer dalam mekanisme printspooler, komunikasi data
- Untuk simulasi dan modeling (misalnya simulasi sistem pengendali lalu lintas udara) dalam memprediksi performansi

1.4 Queue Dengan Liniear Array:

Terdapat satu buah pintu masuk di suatu ujung dan satu buah pintu keluar di ujung Satunya Sehingga membutuhkan variabel Head dan Tail.



1.4.1 Create

- Untuk menciptakan dan menginisialisasi Queue
- Dengan cara membuat Head dan Tail

1.4.2 IsEmpty

- Untuk memeriksa apakah Antrian sudah penuh atau belum
- Dengan cara memeriksa nilai Tail, jika Tail = -1 maka empty

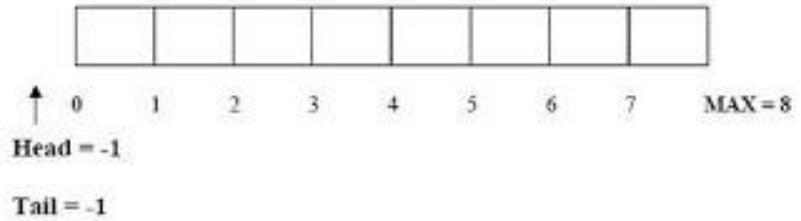
- Kita tidak memeriksa Head, kerena Head adalah tanda untuk kepala antrian (elemen pertama dalam antrian) yang tidak akan berubah-ubah
- Pergerakan pada Antrian terjadi dengan penambahan elemen antrian kebelakang, yaitu menggunakan nilai Tail.

1.4.3 IsFull

- Untuk mengecek apakah antrian sudah penuh atau belum
- Dengan cara mengecek nilai tail, jika Tail>=MAX-1 (karena MAX-1 adalah batas elemen array pada C) berarti sudah penuh

1.4.4 Clear

- Untuk menghapus elemen-elemen antrian dengan cara membuat Tail dan Head =-1
- Penghapusan elemen-elemen antrian sebenarnya tidak menghapus arraynya, namun hanya mengeset indeks pengaksesannya ke nilai-1 sehingga elemen-elemen antrian tidak lagi terbaca



1.4.5 Print

- Untuk menampilkan nilai-nilai elemen Antrian

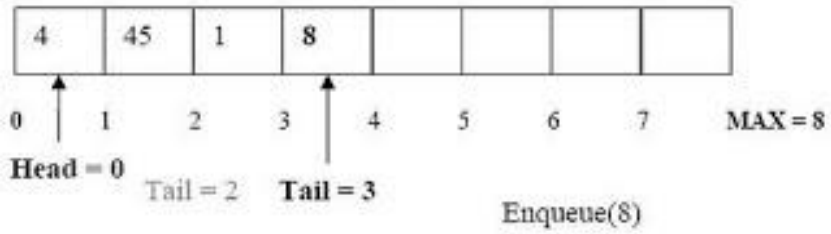
- Menggunakan looping dari head s/d tail

1.5 Operasi Queue

Dua operasi pada antrian yakni enQueue dan deQueue. Untuk menyisipkan data pada antrian menggunakan enQueue dan menghapus data dari antrian menggunakan deQueue.

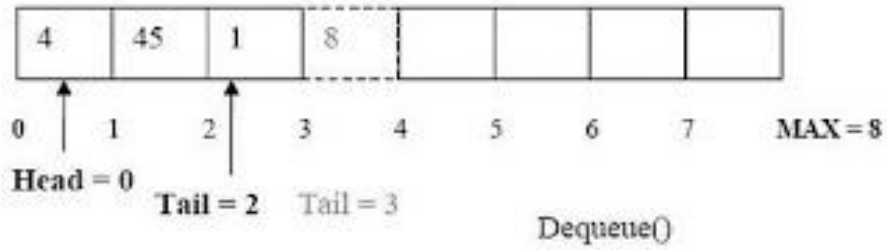
1.5.1 Enqueue(data)

- Untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu ditambahkan di elemen paling belakang
- Penambahan elemen selalu menggerakan variabel Tail dengan cara increment counter Tail



1.5.2 Dequeue()

- Digunakan untuk menghapus elemen terdepan/pertama dari Antrian
- Dengan cara mengurangi counter Tail dan menggeser semua elemen antrian kedepan
- Penggeseran dilakukan dengan menggunakan looping



1.6 Perbedaan Stack dan 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 palingawal atau elemen terdepan. Sifat yang demikian dikenal dengan FIFO.

Latihan 1

Pemrograman Java

Nama Program : package P11

Bahasa Pemrogramam : Java

Compiler : NetBeans IDE 8.2

Script program :

Class Node

```
package p11;

public class Node {
    Object data;
    Node next;

    Node(Object object){this( object,null);}

    Node( Object object, Node node){
        data = object;
        next = node;
    }

    Object getObject(){return data;}

    Node getNext(){return next;}
}
```

Class List

```
package p11;

public class List {
    private Node nodeAwal;
    private Node nodeAkhir;
    private String nama;
    public List(){  this( "list" ); }
    public List( String namaList ){
        nama = namaList;
nodeAwal = nodeAkhir = null;
    }
    public void sisipDiAwal( Object dt ){
        if (kosong()) nodeAwal = nodeAkhir = new Node( dt );
else nodeAwal = new Node( dt, nodeAwal );
    }
    public void sisipDiAkhir( Object dt ){
        if (kosong()) nodeAwal = nodeAkhir = new Node( dt );
        else nodeAkhir = nodeAkhir.next = new Node( dt );
    }

    public Object hapusDrDepan(){
        Object itemDihapus = null;
        if (!kosong()) {
            itemDihapus = nodeAwal.data;
```

```
            if ( nodeAwal == nodeAkhir )
                nodeAwal = nodeAkhir = null;
            else nodeAwal = nodeAwal.next;
        }
        return itemDihapus;
    }
    public boolean kosong(){return nodeAwal == null;}
    public void cetak(){
        if ( kosong() ){
            System.out.printf( "Kosong %s\n", nama );
            return;
        }
        System.out.printf( "Isi %s adalah : ", nama );
        Node kini = nodeAwal;
        while ( kini != null ){
            System.out.printf( "%s ", kini.data );
            kini = kini.next;
        }
        System.out.println( "\n" );
    }
}
```

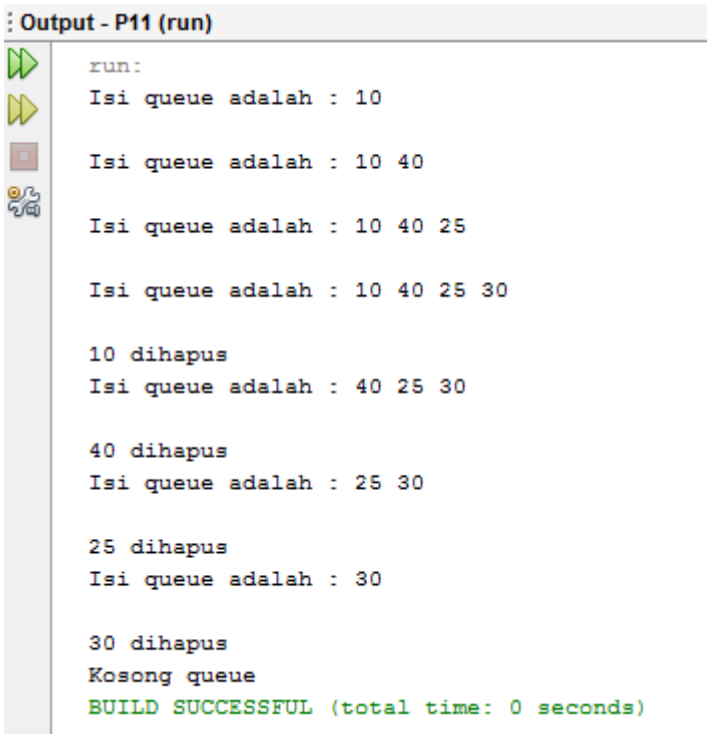
Class Queue

```
package p11;

public class Queue {
    private List listAntrian;
    public Queue() {
        listAntrian = new List( "queue" );
    }
    public void enqueue( Object object ){
        listAntrian.sisipDiAkhir( object );
    }
    public Object dequeue(){
        return listAntrian.hapusDrDepan();
    }
    public boolean kosong(){
        return listAntrian.kosong();
    }
    public void cetak(){listAntrian.cetak();}
    public static void main( String args[]){
        Queue q = new Queue();
q.enqueue( 10 );
q.cetak();
q.enqueue( 40 );
q.cetak();
q.enqueue( 25 );
q.cetak();
q.enqueue( 30 );
q.cetak();
```

```
Object dtHapus = null;
while(!q.kosong()){
    dtHapus = q.dequeue();
    System.out.printf("%s dihapus \n",dtHapus );
    q.cetak();
}
    }
}
```

Output Program :



Penjelasan Program :

Pada program ini praktikkan membuat 3 class public dengan nama class Node, class List dan class Queue. Class Queue digunakan sebagai main method, didalam class tersebut terdapat beberapa operasi seperti enqueue ini digunakan untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu ditambahkan di elemen paling belakang, sehingga urutan dari enqueue program diatas adalah 10, 40, 25, 30, karena queue menerapkan konsep FIFO atau First In First Out, maka data yang pertama masuk adalah data yang pertama keluar, pengertian sederhananya adalah penerapan antrian. Sehingga program diatas mengeluarkan atau menghapus data awal yaitu 10 berurutan menghapus data ke 40, 25, dan yang terakhir 30. Setelah itu program akan menampilkan bahwa queue telah kosong.

Latihan 2

Pemrograman Java

Nama Program : package P11_2

Bahasa Pemrogramam : Java

Compiler : NetBeans IDE 8.2

Script program :

Class Node

```
package p11_2;

public class Node {
   Object data;
    Node next;

    Node(Object object){this( object,null);}

    Node( Object object, Node node){
        data = object;
        next = node;
    }

    Object getObject(){return data;}

    Node getNext(){return next;}

}
```

Class List

```
package p11_2;

public class List {
    private Node nodeAwal;
    private Node nodeAkhir;
    private String nama;
    public List(){  this( "list" ); }
    public List( String namaList ){
        nama = namaList;
nodeAwal = nodeAkhir = null;
    }
    public void sisipDiAwal( Object dt ){
        if (kosong()) nodeAwal = nodeAkhir = new Node( dt );
        else nodeAwal = new Node( dt, nodeAwal );
    }
    public void sisipDiAkhir( Object dt ){
        if (kosong()) nodeAwal = nodeAkhir = new Node( dt );
        else nodeAkhir = nodeAkhir.next = new Node( dt );
    }
```

```
public Object hapusDrDepan(){
    Object itemDihapus = null;
    if (!kosong()) {
        itemDihapus = nodeAwal.data;
        if ( nodeAwal == nodeAkhir )
            nodeAwal = nodeAkhir = null;
        else nodeAwal = nodeAwal.next;
    }
    return itemDihapus;
}
public boolean kosong(){return nodeAwal == null;}
public int size(){
    Node kini = nodeAwal; int count = 0;
    while(kini!= null){
        count++;
        kini = kini.next;
    }
    return count;
}
    public void cetak(){
        if ( kosong() ){
            System.out.printf( "Kosong %s\n", nama );
            return;
        }
        Node kini = nodeAwal;
        while ( kini != null ){
            System.out.printf( "%s ", kini.data );
            kini = kini.next;
        }
        System.out.println( "\n" );
    }
}
```

Class Queue

```
package p11_2;

public class Queue {
    private List listAntrian;
    public Queue() {
        listAntrian = new List( "queue" );
    }
    public void enqueue( Object object ){
        listAntrian.sisipDiAkhir( object );
    }
    public Object dequeue(){
        return listAntrian.hapusDrDepan();
    }
    public boolean kosong(){
        return listAntrian.kosong();
    }
```

```
    public void cetak(){listAntrian.cetak();
    }
    public int size(){
        return listAntrian.size();
    }
   public static void main( String args[]){
       Queue q = new Queue();
       q.enqueue( 10 );
       q.enqueue( 20 );
       q.enqueue( 30 );
       q.enqueue( 40 );
       q.enqueue( 50 );
       System.out.println("Element : ");
       q.cetak();
       System.out.println("Ukuran Queue : "+q.size()+"\n");
       System.out.printf("Hapus Angka : %s\n",q.dequeue());
       System.out.println("Ukuran Queue : "+q.size()+"\n");
       System.out.println("Tambahkan Data : 60");
       q.enqueue(60);
       System.out.println("Ukuran Queue : "+q.size());
       System.out.print("Cetak Queue : ");
       q.cetak();
   }
}
```

Output Program :

```
Output - P11_2 (run)
run:
Element :
10 20 30 40 50

Ukuran Queue : 5

Hapus Angka : 10
Ukuran Queue : 4

Tambahkan Data : 60
Ukuran Queue : 5
Cetak Queue : 20 30 40 50 60

BUILD SUCCESSFUL (total time: 0 seconds)
```

Penjelasan Program :

Pada program ini praktikkan membuat 3 class public dengan nama class Node, class List dan class Queue. Class Queue digunakan sebagai main method, didalam class tersebut terdapat beberapa operasi seperti enqueue ini digunakan untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu ditambahkan di elemen paling belakang, sehingga

urutan dari enqueue program diatas adalah 10, 20, 30, 40, 50 karena queue menerapkan konsep FIFO atau First In First Out, maka data yang pertama masuk adalah data yang pertama keluar, pengertian sederhananya adalah penerapan antrian. Sehingga program diatas mengeluarkan atau menghapus data awal yaitu 10, setelah menghapus kemudian program ini menambahkan data 60. Dalam praktikum kali ini menggunakan ukuran queue atau jumlah antrian yang berada pada queue dengan mendeklarasikan fungsi public int size() yaitu node awal 0, kemudian menggunakan perulangan while dimana terdapat kondisi apabila nilai dari variabel kini tidak sama dengan null maka count++ atau ditambah dengan 1, Setelah itu program akan menampilkan bahwa queue telah kosong.

KESIMPULAN

Dari praktikum kali ini, kami dapat menyimpulkan bahwa :

- ADT Queue (antrian) adalah struktur data dimana proses pengambilan dan penambahan element dilakukan pada ujung yang berbeda dengan mengikuti konsep FIFO (First In First Out). Queue berguna untuk menyimpan pekerjaan yang tertunda.
- Contoh penggunaan atau implementasi dari ADT Queue misalnya aplikasi untuk antrian di rumah sakit, *mailbox* dalam komunikasi antar proses, simulasi dan modeling (misalnya simulasi sistem pengendali lalu lintas udara) dalam memprediksi performansi, *Waiting Line* pada Sistem Operasi

DAFTAR RUJUKAN

Annisa Puspa Kirana, S.Kom, M.Kom. 2017. Modul Praktikum Algoritma dan Struktur Data. Malang : Universitas Negeri Malang.

Arif. 2014. Queue, http://kabularif17.blogspot.co.id/2014/06/queue.html, diakses pada tanggal 19 April 2017, pukul 18:24.