SISTEM ANTRIAN CUSTOMER BENGKEL TOYOTO DENGAN LINKED LIST ADT

Laporan ini disusun guna memenuhi salah satu syarat untuk mengikuti mata kuliah Struktur Data Teori

Disusun Oleh :

Kelompok4

Fathan Suprayogi	**5160411090**
Yohanes Irvan D A	**5160411091**
Muklis Adi S	**5160411104**
Akpin Gilang S	**5160411102**

PROGRAM STUDI S-1 INFORMATIKA

FAKULTAS INFORMASI TEKNOLOGI DAN ELEKTRO

UNIVERSITAS TEKNOLOGI YOGYAKARTA

2018

KATA PENGANTAR

Puji syukur diucapkan ke hadirat Allah SWT atas segala rahmat, hidayah, dan inayah-Nya, sehingga laporan yang berjudul Antrian customer bengkel Toyoto dengan linked list adt dapat diselesaikan. Adapun tujuan penulisan laporan ini untuk memenuhi salah satu tugas kelompok mata kuliah struktur data teori, salah satu mata kuliah yang ada di prodi informatika.

Laporan ini dibuat sebagai sumber wawasan bagi mahasiswa untuk mempelajari materi struktur data, khususnya queue/antrian. Di dalam laporan ini dipaparkan beberapa ulasan tentang salah satu source code antrian dengan Linked list ADT dan penjelasan tentang program tersebut. Untuk itu laporan ini diharapkan berguna bagi pembaca dalam memperdalam materi antrian.

Dalam penyusunan Laporan ini tidak terlepas dari bimbingan dari berbagai pihak, untuk itu diucapkan terima kasih kepada Bapak Noprianto yang senantiasa membimbing penyusunan laporan ini. Dalam penyusunan Laporan ini, disadari masih banyak kekurangan. Untuk itu diperlukannya kritik dan masukkan yang membangun demi kesempurnaan penyusunan Laporan ini. Demikian laporan ini disusun, semoga bermanfaat bagi semua pihak.

Yogyakarta, 05 Januari 2018

Penyusun

DAFTAR ISI

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Dalam kehidupan sehari-hari, sesuatu yang paling membuat seseorang merasa kesal adalah mengantri. Mengapa demikian?seseorang akan mendapat jatah atau tujuan dalam mengantri tergantung urutan yang didapat orang tersebut. Orang yang mendapat nomor antrian pertama, maka dialah yang akan dipanggil terlebih dahulu. Dalam hal ini dikenal dengan istilah FIFO(First In First Out).

Begitu juga dengan Pemrograman terstruktur. Di dalam pemrograman antrian dikenal dengan istilah Queue. Queue banyak digunakan dalam aplikasi seperti antrian tiket, pemesanan obat, tol, nasabah bank, dll. Sebuah Queue dapat disajikan dengan beberapa metode, seperti queue dengan Linked list,queue dengan array, dan queue ADT. Dalam hal ini akan dibahas mengenai Queue secara detail dengan metode-metode tersebut

B. Rumusan Masalah

Dari uraian latar belakang masalah diatas maka dapat di cari rumusan masalah sebagai berikut:

1. Bagaimana konsep Antrian customer bengkel toyoto dengan linked list ADT?
2. Apa saja operasi yang ada pada Antrian customer bengkel toyoto dengan linked list ADT?

C. Tujuan

Berdasarkan rumusan masalah diatas maka tujuan dibuatnya laporan ini adalah:

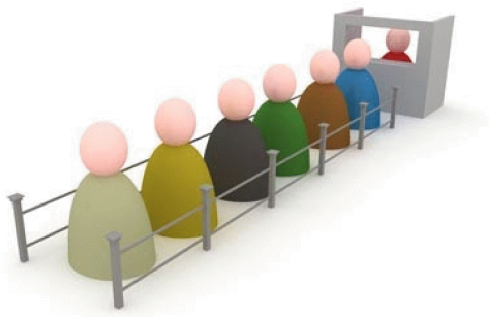
1. Mengetahui Antrian customer bengkel toyoto dengan linked list ADT.
2. Mengetahui operasi yang ada pada Antrian customer bengkel toyoto dengan linked list ADT.

BAB II

LANDASAN TEORI

A. Queue

Queue jika diartikan secara harfiah, *queue* berarti antrian, *Queue* merupakan suatu struktur data linear. Konsepnya hampir sama dengan *Stack*, perbedaannya adalah operasi penambahan dan penghapusan pada ujung yang bebeda. Pada *Stack* atau tumpukan menggunakan prinsip "Masuk terakhir keluar pertama"atau *LIFO (Last In First Out)*, Maka pada *Queue* atau antrian prinsip yang digunakan adalah "Masuk Pertama Keluar Pertama" atau *FIFO (First In First Out)*. Data-data di dalam antrian dapat bertipe *integer*, *real, record* dalam bentuk sederhana atau terstruktur(Santi dalam http://blognyonyait.blogspot.com/2017/05/makalah-sistem-antrian-queue-dengan.html ,diakses tanggal 27 Desember 2017 pukul 18.00)



Gambar1. Ilustrasi antrian/Queue
(Sumber: http://www.indonesiaantri.com/2014/03/pengaruh-antrian-terhadap-kepuasan.html ,diakses tanggal 27 Desember 2017, pukul 20.00 WIB.)

B. Operasi

Selain menggunakan array, queue juga dapat dibuat dengan linked list. Metode linked list yang digunakan adalah double linked list. Operasi-operasi Queue dengan Double Linked List:

1. IsEmpty

Fungsi IsEmpty berguna untuk mengecek apakah queue masih kosong atau sudah berisi data. Hal ini dilakukan dengan mengecek apakah head masih menunjukkan pada Null atau tidak. Jika benar berarti queue masih kosong.

2. IsFull

Fungsi IsFull berguna untuk mengecek apakah queue sudah penuh atau masih bias menampung data dengan cara mengecek apakah Jumlah Queue sudah sama dengan MAX_QUEUE atau belum. Jika benar maka queue sudah penuh.

3. EnQueue

Fungsi EnQueue berguna untuk memasukkan sebuah elemen ke dalam queue (head dan tail mula-mula meunjukkan ke NULL).

4. DeQueue

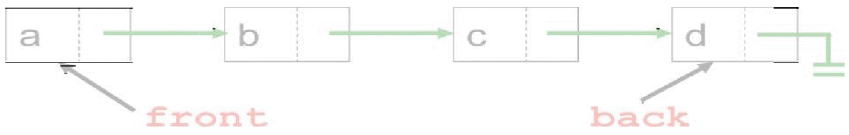
Procedure DeQueue berguna untuk mengambil sebuah elemen dari queue. Hal ini dilakukan dengan cara menghapus satu simpul yang terletak paling depan (head).

BAB III

PEMBAHASAN

A. Queue dengan Linked List

Queue merupakan sebuah antrian. Untuk mengimplementasikannya dapat melalui beberapa metode salah satunya dengan Linked List. Untuk dapat menggunakan Queue dengan metode linked list, maka yang digunakan adalah double linked list. Menggunakan 2 variabel reference objek yakni front dan back.Queue dikatakan koso ng apabila front=back=null. Proses enqueue dilaku kan dengan cara membuat node bar u N, kemudian memasukkan data baru X ke dala mN. Jika queue dalam keadaan kosong, mengatur nilai front=back =N. Jika queue tidak kosong, maka ditambahkan N dan diupdate nilai back. Prosess dequeue dilakukan dengan cara menghapus elemen pe rtama (yang diacu oleh variabel front).



Gambar 1. Il ustrasi Implementasi Queue dengan Linked List

Queue/antrian adalah *ordered list* dengan penyisipan di satu ujung, sedang penghapusan di ujung lain. Ujung penyisipan biasa disebut *rear/tail*, sedang ujung penghapusa disebut *front/head*. Fenomena yang muncul adalah elemen yang lebih dulu disisipkan akan juga lebih dulu diambil. *Queue* merupakan kasus khusus *ordered list*. Dengan karakteristik terbatas itu maka kita dapat melakukan optimasi representasi *ADT Queue* untuk memperoleh kerja paling optimal.

Misalnya *Queue* Q= (a1,a2,a3…,an), maka:

1. Elemen a1 adalah elemen paling depan
2. Elemen ai adalah diatas elemen ai-1, di mana 1<i<n.
3. Elemen an adalah elemen paling belakang.

Head (Front) menunjukke awal antrian Q (elemen terdepan), sedangkan *tail (rear)*menunjuk akhir antrian Q (elemen paling belakang). Disiplin

FIFO pada *Queue* berimplikasi jika elemen A, B, C, D, E dimasukkan ke *Queue*, maka penghapusan/ pengambilan elemen akan terjadi dengan urutan A, B, C, D, E.

Sebagai gambaran, cara kerja *queue* dapat disamakan pada sebuah antrean di suatu loket dimana berlaku prinsip 'Siapa yang duluan antre dia yang akan pertama kali dilayani', sehingga dapat dikatakan prinsip kerja *queue* sama dengan prinsip sebuah antrean.Untuk itu Queue memiliki beberapa karakteristik sebagai berikut:

1. Elemen antrian yaitu item-item data yang terdapat di elemen antrian.
2. *Head/Front* (elemen terdepan dari antrian).
3. *Tail/Tear* (elemen terakhir dari antrian).
4. Jumlah elemen pada antrian (*count*).

Ada beberapa kondisi yang bisa kita temukan dalam queue. Kondisi antrian yang menjadi perhatian adalah:

1. Penuh

Bila elemen di antrian mencapai kapasitas maksimum antrian.Pada kondisi ini, tidak mungkin dilakukan penambahan ke antrian.Penambahan elemen menyebabkan kondisi kesalahan *Overflow*.

2. Kosong

Bila tidak ada elemen di antrian.Pada kondisi ini, tidak mungkin dilakukan pengambilan elemen dari antrian.Pengambilan elemen menyebabkan kondisi kesalahan *Underflow*.

B. Penjelasan Operasi Antrian Customer Bengkel Toyoto Dengan Linked List ADT

Berikut adalah penjelasan dari operasi Queue dengan Linked list:

1. Empty

   Function Empty berguna untuk mengecek apakah QUEUE masih kosong atau sudah berisi data. Hal ini dilakukan dengan mengecek apakah head masih menunjuk pada nil atau tidak, jika ya maka kosong.

```
int isEmpty(queue *queue){
        if(queue->count==0) return 1;
        else return 0;
        }
```

3. Full

Function Full berguna untuk mengecek apakah QUEUE sudah penuh atau masih bias menampung data dengan cara mengecek apakah N (JumlahQueue) sudah sama dengan Max_Queue atau belum, jika ya maka penuh.

```
int isFull(queue *queue){
        if(queue->count==5) return 1;
        else return 0;
        }
```

4. EnQueue

Procedure EnQueue berguna untuk memasukkan 1 elemen ke dalam QUEUE. Head dan Tail mula-mula menunjuk ke NULL.

```
void mobil_masuk(queue *queue){
clrscr();
system("color 70");
    mobil *pNew;
    pNew=(mobil*)malloc(sizeof(mobil));
    if(pNew!=NULL){
    if(queue->count==5) {cout<<"Ruang Bengkel full"; getch();}
    else{
        pNew->next=NULL;

cout<<"|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxx|"<<endl;
        cout<<"|                MENU REGISTRASI CUSTOMER
|"<<endl;

cout<<"|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxx|"<<endl;
        cout<<"|Nama Pemilik    : "; scanf("%[^\n]",&pNew-
>pemilik);fflush(stdin);
```

```
        cout<<"|Plat Nomor       : "; scanf("%[^\n]",&pNew-
>platnomor);fflush(stdin);
        cout<<"|merk Mobil       : "; scanf("%[^\n]",&pNew-
>merk_mobil);fflush(stdin);
        cout<<"|Keluhan       : "; scanf("%[^\n]",&pNew-
>keluhan);fflush(stdin);

        if(queue->count==0){
        queue->head=pNew;
        }
        else{
        queue->tail->next=pNew;

        }
        queue->tail=pNew;
        queue->count=(queue->count)+1;
        }
        }
}
```

5. DeQueue

Procedure DeQueue berguna untuk mengambil 1 elemen dari QUEUE. Jal ini dilakukan dengan cara menghapus satu simpul yang terletak paling depan (head).

```
void mobilout(queue *queue){clrscr();
system("color 70");
     mobil *dltPtr;
    dltPtr = queue->head;
    if(queue->count == 1){
          queue->head = queue->tail =NULL;
    }
    else {
    queue->head=queue->head->next;
    }
    queue->count--;
    free(dltPtr);
}
```

6.Print

Untuk menampilkan nilai-nilai elemen *queue* menggunakan *looping* dari HEAD sampai dengan TAIL. Berikut deklarasi prosedur tampil:

```
        void cetak(queue *queue){
            mobil *pWalker;
            int i=1;
            pWalker=queue->head;
            if(queue->count==0){printf("\nBengkel kosong");getch();}
            else{ system("cls");
            mobil2an(queue);
            gotoxy(1,7);

cout<<"|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx|"<<endl;
            cout<<"|                              DAFTAR  ANTRIAN  CUSTOMER
|"<<endl;

cout<<"|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx|"<<endl;
            while(pWalker!=NULL){

                        cout<<"mobil ke "<<i<<endl;
                        cout<<"plat nomor:"<<pWalker->platnomor<<endl;
                        cout<<"Pemilik          :"<<pWalker->pemilik<<endl;
                        cout<<"keluhan          :"<<pWalker->keluhan<<endl;
                        cout<<endl;
                        pWalker=pWalker->next; i++;
                        }
                        getch();
                }
        }
```

BAB IV

PENUTUP

A. Kesimpulan

Queue 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 melakukan . Data-data di dalam antrian dapat bertipe *integer*, *real, record* dalam bentuk sederhana atau terstruktur. Queue dilakukan dengan cara penyisipan di satu ujung, sedang penghapusan di ujung lain. Ujung penyisipan biasa disebut *rear/tail*, sedang ujung penghapusa disebut *front/head*.

Terdapat operasi – operasi dasar pada queue, yaitu: prosedur *create* untuk membuat queue baru yang kosog, fungsi *IsEmpty* untuk mengecek queue tersebut kosong atau tidak, fungsi *IsFull* untuk mengecek queue tersebut penuh atau tidak, prosedur *EnQueue* untuk memasukkan data kedalam queue, prosedur *DeQueue* untuk mengeluarkan sebuah elemen pada posisi *head* dari queue, fungsi *clear* untuk menghapus elemen queue, dan prosedur *tampil* untuk menampilakn elemen yang ada pada queue.

B. Saran

Mungkin perlu diterangkan lebih lanjut tentang penerapan Queue pada program yang mengacu pada sebuah sistem, sehingga mahasiswa tidak hanya terpacu pada modul,dan lebih bagus lagi bila mahasiswa dihadapkan pada sebuah kasus yang berkaitan dengan stack,sehingga mahasiswa dapat berusaha setidaknya mencari tahu lebih dari sekedar menyalin sebuah contoh program instant.