MAKALAH FUNGSI DAN QUEUE

BAHASA C



OLEH:

DIKI CANDRA

NIM 2022903430010

D4 TEKNOLOGI REKAYASA KOMPUTER JARINGAN

POLITEKNIK NEGERI LHOKSEUMAWE

2022/2023

**DAFTAR ISI**

[BAB I 4](#_Toc89291515)

[PENDAHULUAN 4](#_Toc89291516)

[A. Latar Belakang 4](#_Toc89291517)

[B. Rumusan Masalah 5](#_Toc89291518)

[C. TujuanPenulisan 5](#_Toc89291519)

[D. Manfaat penulisan 5](#_Toc89291520)

[BAB II 6](#_Toc89291521)

[PEMBAHASAN 6](#_Toc89291522)

[A. Pengertian Queue 6](#_Toc89291523)

[b. Karakteristik Queue 8](#_Toc89291524)

[c. Operasi Operasi Pada Queue 9](#_Toc89291525)

[d. Macam-Macam Queue dan Repsesentasinya 15](#_Toc89291526)

[BAB III 44](#_Toc89291527)

[PENUTUP 44](#_Toc89291528)

[a. Kesimpulan 44](#_Toc89291529)

# BAB I

# PENDAHULUAN

1. **[Latar](https://www.blogger.com/null) Belakang**

Dalam terminology pemrogaman berorientasi objek, pada umumnya kita mengenal suatu struktur data bentukan pengguna yang disebut sebagai ADT (Abstrak Data Type), yang merupakan pengganti struktur data primitive yang telah terdefinisi dalam Bahasa pemrograman tertentu (misalnya int,String, dan sebagainya). Dalam bab ini kita akan mempelajari beberapa ADT Statis (ADT yang menggunakan lokasi memory secara tetap) ADT yang kita pelajari bab ini adalah QUEUE atau ANTRIAN.

Antrian dalam ilmu komputer bermakna sama dengan antrian pada kehidupan nyata. Ada antrian konsumen di sebuah bank atau di supermarket, dan ada pula antrian kendaraan di pintu masuk tol. Sama halnya dengan komputer dapat mengirimkan permintaan cetak lebih cepat daripada kemampuan printer dalam mencetak, antrian dokumen sering kali terjadi pada sebuah printer. Aturan umum dalam memproses elemen – elemen sebuah antrian adalah bahwa elemen terdepan antrian dilayani lebih dulu dan elemen yang paling belakang akan dilayani setelah elemen – elemen lain di depannya diproses. Jadi, antrian adalah sebuah struktur data FIFO (First In, First Out). Antrian memiliki banyak aplikasi dalam ilmu komputer. Kapan pun sebuah sistem dimodelkan menggunakan prinsip FIFO, antrian pasti digunakan.

Queue merupakan kasus khusus ordered list. Dengan karakteristik terbatas itu maka kita dapat melakukan optimasi representasi ADT Queue untuk memperoleh kerja paling optimal. Namun sebagai mahasiswa bidang komputer, tentu kita harus bisa mengkomputerkan apapun dari dunia nyata. Untuk itu kita perlu memahami tentang struktur data queue, operasi – operasi pada struktur  data queue, dan bagaimanakah kita bisa mengimplementasikan operasi

– operasi struktur data queue tersebut kedalam bahasa pemograman, disini kita bahas bahasa pemograman C++.

1. **Rumusan Masalah**

Dari uraian latar belakang diatas, penulis merumuskan beberapa masalah yang akan dijelaskan melalui laporan ini, yaitu sebagai berikut:

1. Apakah struktur data queue itu.
2. Apa saja operasi-operasi stuktur data queue.
3. Bagaimana mengimplementasikan struktur data queue dalam bahasa C.
4. Apa saja contoh aplikasinya dalam kehidupan sehari-hari.
5. **TujuanPenulisan**

Tujuan pembuatan laporan ini adalah agar mahasiswa memahami tentang struktur data queue, dan mengerti bagaimana cara mengimplemetasikan dalam bahasa pemograman. Dan sebagai syarat pemenuhan tugas dalam mata kuliah Struktur Data, sekaligus bisa memberikan pendidikan dasar bagi penulis agar lebih banyak tahu tentang struktur data queue.

## D. Manfaat penulisan

Penulisan makalah ini diharapkan dapat memberikan manfaat kepada semua pihak, khususnya kepada mahasiswa untuk menambah pengetahuan dan wawasan. Adapaun manfaat penulisan makalah ini adalah sebagai berikut :

1. Dapat memahami pengertian struktur data queue

2. Dapat memahami cara pengoperasian struktur data queue

3. Dapat mengimplementasikan struktur data queu ke dalam bahasa C

4. Dapat mengetahui aplikasi struktur data queue dalam kehidupan.

# BAB II

# PEMBAHASAN

1. **Pengertian Queue**

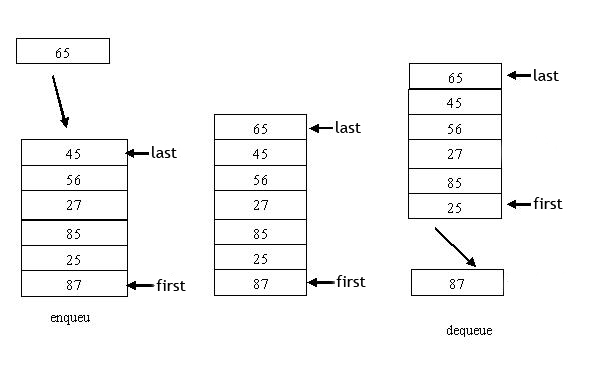
Queue jika diartikan secara harfiah, queue berarti antrian, Queue merupakan suatu struktur data linear. Konsepnya hampir sama dengan Stack, perbedaannya adalah operasi penambahan dan penghapusan pada ujung yang bebeda. Pada Stack atau tumpukan menggunakan prinsip “Masuk terakhir keluar pertama”atau LIFO (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.

Pada saat kita mengantri, setiap kali ada orang yang datang, otomatis orang tersebut akan berada di belakang antrian bukan? Dan siapakah orang yang pertama kali dilayani?. Orang yang pertama kali datang atau orang yang berada di sisi depan antrian. Dalam antrian, ada beberapa poin penting, diantaranya adalah siapa orang yang berada paling depan, siapa yang berada di paling belakang, serta jumlah orang yang mengantri ada berapa.

Nah, queue atau antrian pada struktur data juga sama, hanya saja orang diganti menjadi data/elemen. Maka queue adalah sekumpulan elemen yang jika ada elemen baru yang ditambahkan, maka elemen tersebut akan berada di bagian belakang antrian Jika ada elemen yang harus dihapus atau keluar dari antrian, maka elemen yang keluar adalah elemen yang berada di sisi depan antrian. Atau konsep ini sering juga disebut dengan konsep FIFO (First In First Out).



Queue (antrian) adalah salah satu list linier dari struktur data yang beroperasi dengan cara First In First Out (FIFO) yaitu elemen pertama yang masuk merupakan elemen yang pertama keluar. Contohnya, ialah dalam sebuah antrian pembelian tiket bagi yang pertama masuk maka dia pulalah yang pertama keluar/selesai. Untuk penyisipan (INSERT) hanya dapat dilakukan pada satu sisi yaitu sisi belakang (REAR), sedangkan untuk penghapusan (REMOVE) pada sisi depan (FRONT) dari 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) menunjuk ke 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.

1. **Karakteristik Queue**

Karakteristik penting antrian 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).
5. Status/kondisi antrian.

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

* Penuh

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

* Kosong

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

1. **Operasi Operasi Pada Queue**

Sebuah queue di dalam program komputer dideklarasikan sebagai sebuah tipe bentukan baru, di dalam Bahasa C, biasa disebut struct. Sebuah struktur data dari sebuah queue setidaknya harus mengandung tiga variabel, yakni:

1. Variabel head yang akan berguna sebagai penanda bagian depan antrian,

2. Variabel tail yang akan berguna sebagai penanda bagian belakang antrian.

3. Array data dari yang akan menyimpan data-data yang dimasukkan ke dalam queue tersebut.

Berikut adalah syntax untuk mendeklarasikan struktur data dari sebuah queue:

typedef struct {

int HEAD, TAIL;

int data[max];

}Queue;

Queue antrian;

Dimana, nilai MAX didefinisikan sebagai jumlah tumpukan maksimum yang dapat disimpan dalam queue. Setelah struktur data dari queue didefinisikan dengan syntax di atas, maka setelah itu dapat dibuat variabel-variabel baru yang mengacu pada tipe data Queue di atas, misalkan membuat sebuah variabel bernama antrian yang bertipe Queue: Queue antrian; 8.

Berikut adalah ilustrasi dari sebuah queue kosong dengan ukuran nilai MAX = 8:

Pada dasarnya, operasi-operasi dasar pada queue mirip dengan operasi-operasi dasar pada stack. Perbedaannya hanya pada prosedur push dan pop saja. Pada queue, prosedur yang berfungsi untuk memasukkan data/ nilai ke dalam antrian adalah enqueue, sedangkan prosedur untuk mengeluarkan data/ nilai dari antrian adalah dequeue. Berikut adalah operasi – operasi dasar dalam Queue:

1. Prosedur Create

Sama pada stack, prosedur ini berfungsi untuk mengosongkan queue dengan cara meletakkan HEAD dan TAIL pada indeks array ke-(-1). Berikut deklarasi prosedur create pada queue:

void create()

{

   antrian.HEAD = -1;

   antrian.TAIL = -1;

}

2. Fungsi IsEmpty

Sama seperti fungsinya pada stack, fungsi ini berfungsi untuk melakukan pengecekan terhadap queue, apakah queue tersebut kosong atau tidak. Jika queue tersebut kosong (artinya, HEAD dan TAIL berada pada posisi -1, atau bisa juga ketika HEAD > TAIL), maka fungsi akan mengembalikan nilai 1 (true), tetapi jika queue tersebut tidak kosong/ berisi (artinya, HEAD dan TAIL tidak berada pada posisi -1), maka fungsi akan mengembalikan nilai 0 (false).

Berikut deklarasi fungsi IsEmpty:

int IsEmpty(){

   if(antrian.HEAD=antrian.TAIL==-1)

return 1;

   else

return 0;

}

3. Fungsi IsFull

Fungsi ini berfungsi untuk melakukan pengecekan terhadap queue, apakah queue tersebut penuh atau tidak. Jika queue tersebut penuh (artinya, TAIL berada pada posisi MAX), maka fungsi akan mengembalikan nilai 1 (true), tetapi jika queue tersebut tidak penuh (artinya, TAIL tidak berada pada posisi MAX), maka fungsi akan mengembalikan nilai 0 (false).

Berikut deklarasi fungsi IsFull dalam Bahasa C:

int IsFull(){

   if (antrian.TAIL == max-1)

return 1;

   else

return 0;

}

4. Prosedur EnQueue

Prosedur ini digunakan untuk memasukkan sebuah data / nilai ke dalam queue. Sebelum sebuah data / nilai dimasukkan ke dalam queue, maka prosedur ini terlebih dahulu melakukan pengecekan terhadap posisi HEAD dan TAIL. Jika posisi HEAD dan TAIL masih berada pada indeks ke-(-1) (artinya queue masih kosong), maka prosedur ini akan menempatkan HEAD dan TAIL pada indeks ke-0 terlebih dahulu, baru setelah itu memasukkan data / nilai ke dalam array data queue. Namun, jika posisi HEAD dan TAIL tidak berada pada posisi ke-(-1), maka posisi TAIL yang akan dinaikkan satu level. Jadi, pada proses enqueue, TAIL-lah yang berjalan seiring masuknya data baru ke dalam antrian, sedangkan HEAD akan tetap pada posisi ke-0.

Berikut deklarasi prosedur EnQueue:

void enqueue()

{

   if (IsEmpty()){

     antrian.HEAD=antrian.TAIL=0;

     cout<<”Masukkan data : “;

     cin>>antrian.data[antrian.TAIL];

   }

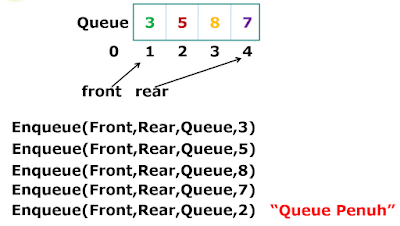
   else

   if(IsFull()){

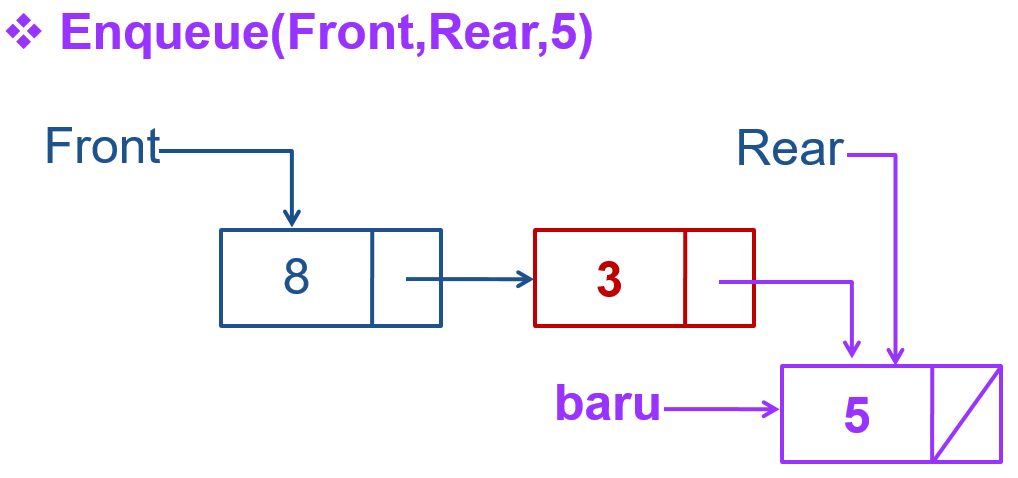
     cout<<”ANTRIAN SUDAH PENUH!”;

   }

}



Illustrasi Enqueue (Array Statis)



Illustrasi Enqueue (Linked List)

5. Prosedur DeQueue

Prosedur ini digunakan untuk mengeluarkan atau membuang sebuah data/ nilai yang paling awal masuk (yang berada pada posisi HEAD, yakni yang paling depan dari antrian) ke dalam queue. Pekerjaan yang dilakukan oleh prosedur ini adalah menaikkan nilai HEAD satu level. Jadi, setiap satu kali data dikeluarkan, maka posisi HEAD naik bertambah satu level. Misalkan HEAD berada pada indeks ke-0, maka ketika akan mengeluarkan/ menghapus data pada posisi paling depan (pada posisi HEAD), prosedur ini akan menaikkan posisi HEAD ke indeks array ke-1. Atau dengan cara menggeser semua elemen antrian kedepan dan mengurangi TAIL dengan 1 penggeseran dilakukan dengan menggunakan looping.

Berikut deklarasi prosedur DeQueue:

int Dequeue(){

   int i;

   int e = antrian.data[antrian.HEAD];

   for(i=antrian.HEAD;i<=antrian.Tail-1;i++){

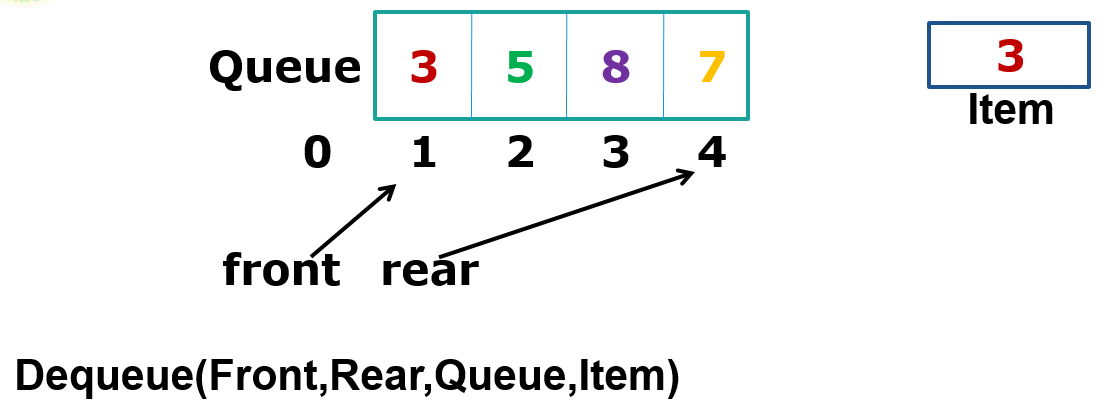
          antrian.data[i]=antrian.data[i+1];

   }

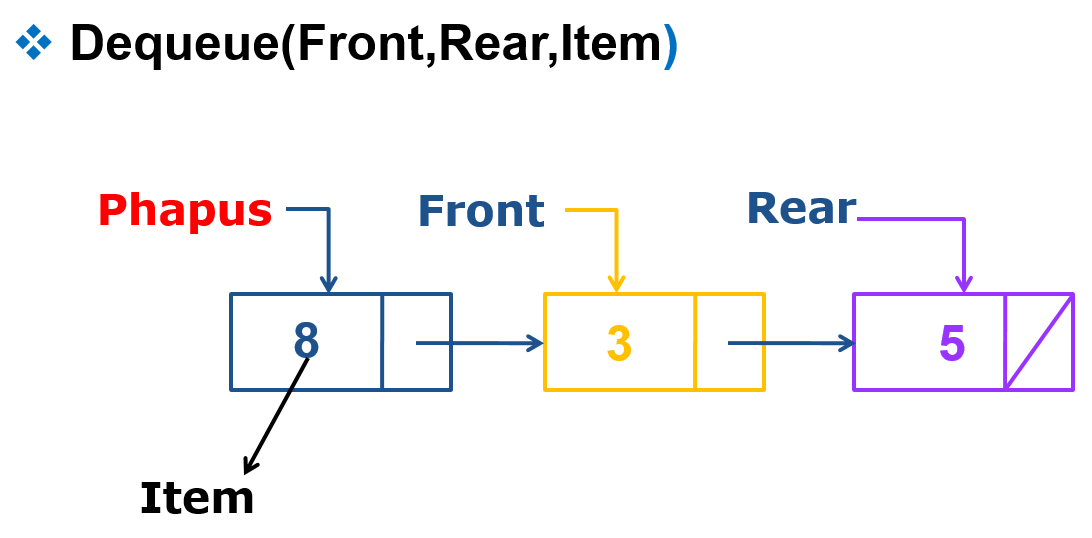
   Antrian.TAIL--;

   Return e;

}



Illustrasi Dequeue (Array Statis)



Illustrasi Dequeue (Linked List)

6. Prosedur Clear

Untuk menghapus elemen-elemen queue dengan cara membuat Tail dan Head= -1. Penghapusan elemen-elemen queue sebenarnya tidak menghapus arraynya, namun hanya mengeset indeks pengaksesan-nya ke nilai -1 sehingga elemen-elemen queue tidak lagi terbaca. Berikut deklarasi prosedur clear:

void clear(){

   antrian.HEAD = -1;

   antrian.TAIL = -1

   cout<<"Antrian sudah dikosongkan! ";

}

7. Prosedur Tampil

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

 void Tampil(){

if(IsEmpty()){

       cout<<"Antrian kosong! ";

     }

     else {

        for(i=antrian.HEAD;i<=antrian.TAIL;i++)

         cout<<"Data pada antrian ke "<<i<<" = " <<antian.data[i]<<endl;

      }

}

1. **Macam-Macam Queue dan Repsesentasinya**
2. Queue dengan Linear Array

Linear array adalah suatu array yang dibuat seakan-akan merupakan suatu garis lurus dengan satu pintu masuk dan satu pintu keluar. Berikut ini diberikan deklarasi kelas QueueLinear sebagai implementasi dari Queue menggunakan linear array. Dalam prakteknya, kita dapat menggantinya sesuai dengan kebutuhan kita. Data dikses dengan field data, sedangkan indeks item pertama dan terakhir disimpan dalam variabel Head dan Tail. Konstruktor akan menginisialisasi nilai Head dan Tail dengan -1 untuk menunjukkan bahwa antrian masih kosong dan mengalokasikan data sebanyak MAX\_QUEUE yang ditunjuk oleh data. Destruktor akan mengosongkan antrian kembali dan mendealokasikan memori yang digunakan oleh antrian.

hapus elemen 1 2 …….. ke – n sisip elemen

head tail

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D |  |  |  |  | …. |  |

Di bawah ini diperlihatkan suatu queue yang akan menempati N elemen array memori, serta cara pengurangan (delete) dan penambahan (added) elemen pada queue tersebut. Head :1

Tail :4

1 2 3 4 5 6 7 ..... N

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | B | C | D |  |  |  |  | …. |  |

REMOVE (Q)

Head : 2

Tail : 4

1 2 3 4 5 6 7 ..... N

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | B | C | D | E | F |  |  | …. |  |

INSERT(INSERT(E),F)

Head : 2

Tail : 6

1 2 3 4 5 6 7 ..... N

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | C | D | E | F |  |  | …. |  |

REMOVE(Q)

Head : 3

Tail : 6

1 2 3 4 5 6 7 ..... N

Dapat dilihat bahwa setiap terjadi penghapusan elemen pada queue nilai (index) dari Head bertambah satu (1) ; dapat ditulis HEAD = HEAD+1. Begitu pula bila terjadi penambahan elemen pada queue nilai (index) Tail bertambah satu (1); dapat ditulis TAIL = TAIL + 1.

Akan terjadi ketidakefisienan bila penambahan elemen sudah pada posisi index N (Tail = N) maka tidak dapat lagi dilakukan penambahan elemen, sedangkan dilokasi memori yang lain (nilai di bawah N) masih terdapat memori array yang kosong. Untuk mengatasi hal tersebut maka kita bayangkan bahwa memori dari queue tersebut berbentuk melingkar dimana kedua ujungnya saling bertemu atau disebut juga dengan Circular Queue.

Operasi-Operasi Queue dengan Linear Array

* 1. IsEmpty

Fungsi IsEmpty berguna untuk mengecek apakah queue sudah terisi atau masih kosong. hal ini dilakukan dengan mengecek apakah tail bernilai -1 atau tidak. Nilai -1 menandakan bahwa queue masih kosong.

* 1. . IsFull Fungsi

IsFull berguna untuk mengecek apakah queue sudah penuh atau masih bias menampung data dengan cara mengecek apakah nilai tail sudah sama dengan jumlah maksimal queue. Jika nilai keduanya sama, berarti antrian sudah penuh.

* 1. EnQueue

Fungsi EnQueue berguna untuk memasukkan sebuah elemen dalam sebuah antrian.

* 1. DeQueue

Fungsi DeQueue berfungsi untuk mengambil sebuah elemen dari antrian. Operasi ini sering disebut juga serve. Hal ini dilakukan dengan cara memindahkan sejauh satu langkah ke posisi di depannya sehingga otomatis elemen yang paling depan akan tertimpa dengan elemen yang terletak di belakangnya.

* 1. Clear Fungsi

Clear berguna untuk menghapus semua lemen dalam queue dengan jalan mengeluarkan semua elemen tersebut satu per satu hingga queue kosong dengan memanfaatkan fungsi DeQueue.

Contoh program:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct antrian{

int value;

struct antrian \*next, \*prev;

}\*head, \*tail, \*current;

void print(){

if(head!=NULL){

current=head;

while(current!=NULL){

printf("| %d ", current->value);

current=current->next;

}

}else{

printf("Data Kosong");

}

printf("\n\n");

};

void pushHead(int value){

current = (struct antrian\*)malloc(sizeof (struct antrian));

current->value=value;

current->next = current->prev = NULL;

if(head==NULL){

head=tail=current;

}else{

head->prev=current;

current->next=head;

head=current;

}

}

void popTail(){

if(tail==NULL){

printf("Data Kosong");

}else if(tail==head){

current=tail;

head=tail=NULL;

free(current);

}else{

current=tail;

tail=tail->prev;

tail->next=NULL;

free(current);

}

}

void main(){

int menu;

int antrianke=1;

print();

do{

do{

system("clear");

print();

printf("1. Queue\n");

printf("2. DeQueue\n");

printf("3. Keluar\n");

printf("Pilih Menu di Atas : "); scanf("%d", &menu); fflush(stdin);

}while(menu<1 || menu>3);

switch(menu){

case 1 :

pushHead(antrianke);

antrianke++;

break;

case 2 :

popTail();

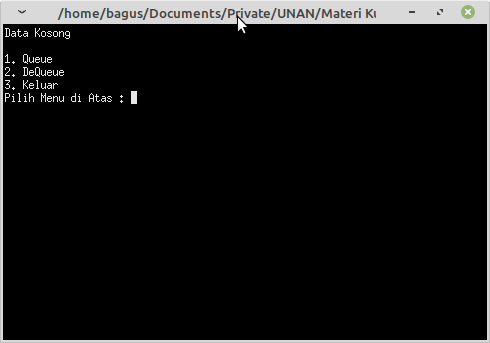
break;

}

}while(menu!=3);

}

Outputnya :



1. Queue dengan Circular Array

Circular array adalah suatu array yang dibuat seakan-akan merupakan sebuah lingkaran dengan titik awal (head) dan titik akhir (tail) saling bersebelahan jika array tersebut masih kosong. 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. Kelebihan jenis ini adalah alokasi penyimpanan data yang optimal dan dinamis. Hal ini disebabkan penambahan maupun pengurangan data/ item antrian yang baru selalu menempati pos kosong yang disediakan sistem.

Queue dengan circular array dapat dibayangkan sebagai berikut:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 6 | 7 | 9 |  |

Front = 1

Tail = 4

1 2 3 4 5

Aturan-aturan dalam queue yang menggunakan circular array adalah:

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

Front dan Tail akan bergerak maju, jika:

* Untuk penambahan.

Tail sudah mencapai elemen terakhir array akan memakai elemen pertama array yang telah dihapus.

* Untuk pngahapusan.

Front telah mencapai elemen terakhir array, maka akan menuju elemen pertama jika antrian masih berisi elemen.

Circular Queue menggunakan fungsi Counter. Counter merupakan jumlah pengantri yang ada dalam antrian.

- Pada proses awal, counter = 0

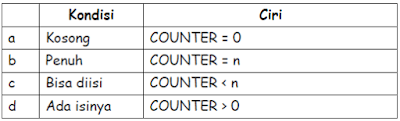
- Pada proses insert, counter++

- Pada proses delete, counter—

Proses Circular Queue :

* AWAL (Inisialisasi)
* INSERT (Sisip, Masuk, Simpan, Tulis)
* DELETE (Hapus, Keluar, Ambil, Dilayani)

Kondisi Antrian :

[](https://1.bp.blogspot.com/-TgGtoA2-E14/XuhzVKHlhGI/AAAAAAAADJg/eZPupG4sfMg0gzrq8u5xh5qq332j4TQlACLcBGAsYHQ/s1600/circular.png)

Operasi-Operasi Queue dengan Circular Array

* 1. IsEmpty Fungsi

IsEmpty berguna untuk mengecek apakah queue sudah terisi atau masih kosong. Hal ini dilakukan dengan mengecek apakah tail lebih besar dari head dan tail masih terletak bersebelahan dengan head. Jika benar demikian, maka queue masih kosong.

* 1. IsFull Fungsi

IsFull berguna untuk mengecek apakah queue sudah penuh atau masih bias menampung data dengan cara mengecek apakah tempat yang masih kosong tinggal satu atau tidak (untuk membedakan dengan empty dimana semua tempat kosong). Jika benar berarti queue penuh.

* 1. EnQueue Fungsi

EnQueue berfungsi untuk memasukkan sebuah elemen ke dalam queue tail dan head awal bernilai nol (0).

* 1. DeQueue

DeQueue berguna untuk mengambil sebuah elemen dari queue. Hal ini dilakukan dengan cara memindahkan posisi head satu posisi ke belakang.

Contoh program:

#include<stdio.h>  
#include<conio.h>  
#include<stdlib.h>  
using namespace std;  
int main() {  
  int queue[10], i, head=-1, tail=-1, enqueue, pil, urut=0, tmp;

do {

system("pause");

system("cls");

printf("1. INSERT");

printf("2. DELETE");

printf("3. CETAK ANTRIAN");

printf("4. QUIT");

printf("Silahkan masukkan pilihan anda : ");

scanf("%d", &pil);

printf("");

if(pil==1) {if(tail==9) {printf("Antrian Penuh");}  
         else if(tail==-1) {head++;tail++;  
                         printf("Masukkan nilai : ");  
                         scanf("%d", &enqueue);  
                         queue[tail]=enqueue;  
                         urut++;  
                         printf("Nomor urut : %d", urut);}

else {tail++;

printf("Masukkan nilai : ");

scanf("%d", &enqueue);

queue[tail]=enqueue;

urut++;

printf("Nomor urut : %d", urut);

printf("Anda harus menunggu %d antrian lagi", tail);}}

else if(pil==2) {if(tail==-1) {printf("Antrian kosong");}  
                 else {printf("Data dengan nilai %d sudah dilayani", queue[head]);  
                 tmp=queue[head];  
                 for(i=head;i<=tail;i++) {queue[i]=queue[i+1];}  
                 queue[tail]=tmp;  
                  urut++;  
                  printf("Data dengan nilai %d masuk antrian", queue[tail], urut);  
                  if(tail==0) {printf("Yang bersangkutan langsung dilayani");}  
                  else {printf("Yang bersangkutan harus menunggu %d antrian lagi", tail);}}}

else if(pil==3) {if(tail==-1) {printf("Antrian kosong");}

else {for(i=head;i<=tail;i++) {printf("Antrian ke-%d : %d", i+1, queue[i]);}

printf("");}}

else if(pil==4) {printf("Anda telah selesai menggunakan program Queue Circular");}

else {printf("Pilihan yang anda masukkan tidak valid");}

} while(pil!=4);

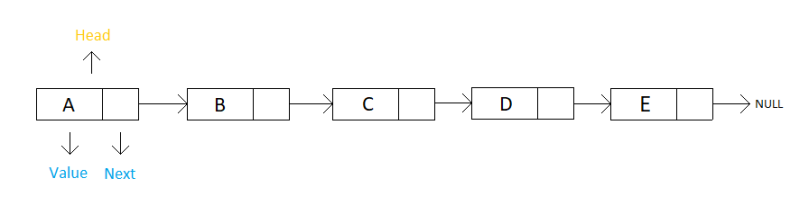
getch();

}

1. Queue dengan Linked-List

Selain menggunakan array, queue juga dapat dibuat dengan linked list. Metode linked list yang digunakan adalah double linked list.

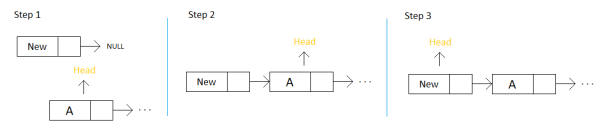
Berikut adalah sebuah gambaran linked list sederhana :



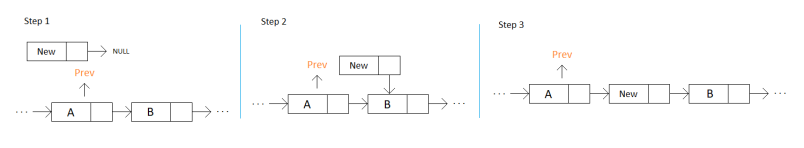
* Head, adalah sebuah pointer atau penanda dari awal sebuah linked list.
* Value, adalah sebuah data yang disimpan dalam sebuah item.
* Next, adalah sebuah pointer yang menghubungkan dengan item selanjutnya.

Kemudian berikut adalah ilustrasi dari operasi – operasi yang umum terdapat dalam linked list sederhana yaitu :

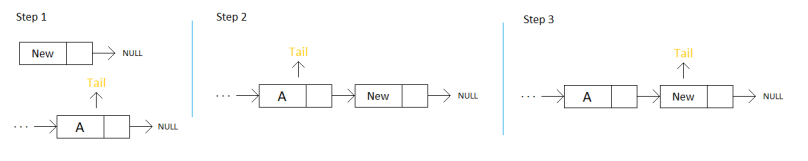
Add First:



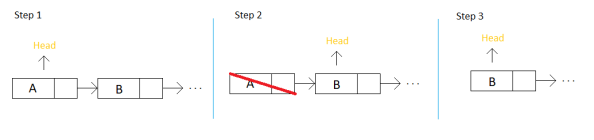
Add After:



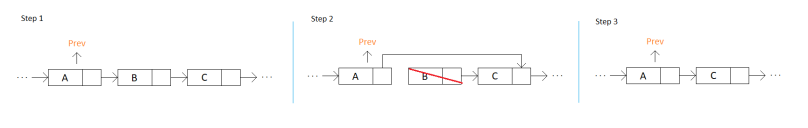
Add Last:



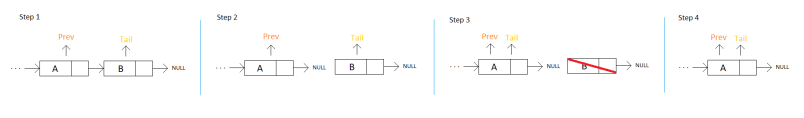
Remove First:



Remove After:



Remove Last:



Operasi-operasi Queue dengan Double Linked List:

* 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.

* 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.

* EnQueue

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

* DeQueue

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

Contoh program:

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

struct Mahasiswa{

char nim[15];

char nama[30];

float ipk;

};

struct Elemen{

Mahasiswa mhs;

Elemen \*next;

};

struct List{

Elemen \*first;

Elemen \*last;

};

void addFirst(char nim[], char nama[], float ipk, List \*L);

void addAfter(Elemen \*prev, char nim[], char nama[], float ipk, List \*L);

void addLast(char nim[], char nama[], float ipk, List \*L);

void deleteFirst(List \*L);

void deleteAfter(Elemen \*prev, List \*L);

void deleteLast(List \*L);

void showData(List \*L);

Elemen \*searchData(char nim[], List \*L);

Mahasiswa inputMahasiswa();

char \*searchNim();

int main(){

char option;

List L;

L.first = NULL;

Mahasiswa data;

Elemen \*prev;

int menu = 0;

int pos = 0;

do{

system("cls");

printf("+++ LINKED LIST +++\n");

printf("1. Input Data (Depan)\n");

printf("2. Input Data (Setelah)\n");

printf("3. Input Data (Akhir)\n");

printf("4. Hapus Data (Depan)\n");

printf("5. Hapus Data (Setelah)\n");

printf("6. Hapus Data (Akhir)\n");

printf("7. Tampilkan Data\n");

printf("8. Search Data\n");

printf("9. Exit\n");

printf("Pilih Menu : ");

scanf("%d", &menu);

// fflush(stdin)

switch(menu){

case 1:

data = inputMahasiswa();

addFirst(data.nim, data.nama, data.ipk, &L);

break;

case 2:

prev = searchData(searchNim(), &L);

if(prev != NULL){

data = inputMahasiswa();

addAfter(prev, data.nim, data.nama, data.ipk, &L);

}

break;

case 3:

data = inputMahasiswa();

addLast(data.nim, data.nama, data.ipk, &L);

break;

case 4:

deleteFirst(&L);

break;

case 5:

prev = searchData(searchNim(), &L);

if(prev != NULL){

deleteAfter(prev, &L);

}

break;

case 6:

deleteLast(&L);

break;

case 7:

showData(&L);

break;

case 8:

searchData(searchNim(), &L);

break;

case 9:

printf("Keluar Program!!");

break;

default:

printf("Invalid Input!!");

break;

}

printf("\n");

printf("Press any key to continue…\n");

getch();

}while(menu!=9);

return 0;

}

void addFirst(char nim[], char nama[], float ipk, List \*L){

Elemen \*baru = new Elemen;

strcpy(baru->mhs.nim, nim);

strcpy(baru->mhs.nama, nama);

baru->mhs.ipk = ipk;

if(L->first == NULL){

L->first = baru;

baru->next = NULL;

L->last = baru;

}else{

baru->next = L->first;

L->first = baru;

}

baru = NULL;

printf("\nBerhasil ditambahkan!");

}

void addAfter(Elemen \*prev, char nim[], char nama[], float ipk, List \*L){

Elemen \*baru = (Elemen\*) malloc (sizeof(Elemen));

strcpy(baru->mhs.nim, nim);

strcpy(baru->mhs.nama, nama);

baru->mhs.ipk = ipk;

baru->next = prev->next;

prev->next = baru;

if(baru->next == NULL){

L->last = baru;

}

baru = NULL;

printf("\nBerhasil ditambahkan!");

}

void addLast(char nim[], char nama[], float ipk, List \*L){

Elemen \*baru = (Elemen\*) malloc (sizeof(Elemen));

strcpy(baru->mhs.nim, nim);

strcpy(baru->mhs.nama, nama);

baru->mhs.ipk = ipk;

if(L->first == NULL){

L->first = baru;

L->last = baru;

baru->next = NULL;

}else{

L->last->next = baru;

L->last = baru;

baru->next = NULL;

}

baru = NULL;

printf("\nBerhasil ditambahkan!");

}

void deleteFirst(List \*L){

Elemen \*data = L->first;

L->first = data->next;

free(data);

printf("\nBerhasil dihapus!");

}

void deleteAfter(Elemen \*prev, List \*L){

Elemen \*data = prev->next;

prev->next = data->next;

if(prev->next == NULL){

L->last = prev;

}

free(data);

printf("\nBerhasil dihapus!");

}

void deleteLast(List \*L){

Elemen \*data = NULL;

Elemen \*prev = NULL;

Elemen \*pNow = L->first;

while(pNow != NULL){

prev = data;

data = pNow;

pNow = pNow->next;

}

if(prev == NULL){

L->first = NULL;

L->last = NULL;

}else{

prev->next = NULL;

L->last = prev;

}

free(data);

printf("\nBerhasil dihapus!");

}

void showData(List \*L){

Elemen \*pnow = L->first;

if(pnow == NULL){

printf("\nAnda belum menambahkan data!\n");

}else{

printf("\nIsi Data");

int no = 0;

while(pnow != NULL){

no++;

printf("\n\nMahasiswa ke-%d", no);

printf("\nNIM : %s", pnow->mhs.nim);

printf("\nNama : %s", pnow->mhs.nama);

printf("\nIpk : %.2f", pnow->mhs.ipk);

pnow = pnow->next;

}

}

}

Elemen \*searchData(char nim[], List \*L){

int no = 0;

Elemen \*pnow = L->first;

Elemen \*result = NULL;

if(pnow == NULL){

printf("\nAnda belum menambahkan data!\n");

}else{

while(pnow != NULL){

no++;

if(strcmp(pnow->mhs.nim,nim) == 0){

result = pnow;

pnow = NULL;

}else{

pnow = pnow->next;

}

}

}

if(result == NULL){

printf("Data tidak ditemukan!");

}else{

printf("Data ditemukan pada posisi ke-%d!",no);

}

return result;

}

Mahasiswa inputMahasiswa(){

Mahasiswa data;

printf("\nNIM : ");

scanf("%s",&data.nim);

printf("Nama : ");

scanf("%s",&data.nama);

printf("Ipk : ");

scanf("%f",&data.ipk);

return data;

}

char \*searchNim(){

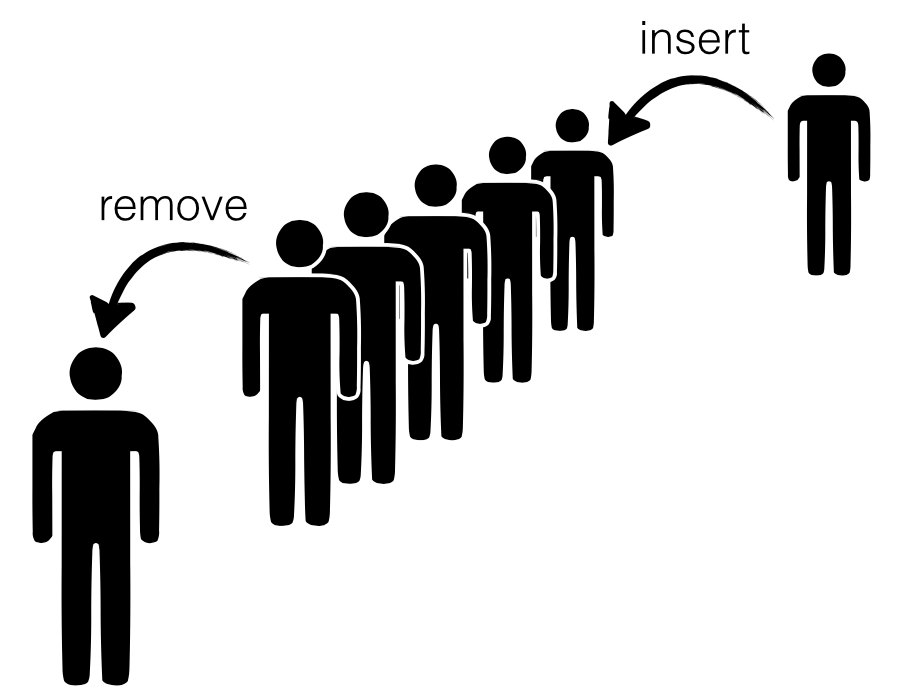
Mahasiswa data;

printf("\nCari data (NIM) : ");

scanf("%s",&data.nim);

return data.nim;

}



1. Penerepan Aplikasi Qeueu

Meski Queue sangat sederhana, namun Queue merupakan kakas dasar penyelesaian masalah-masalah besar. Penggunaan Queue yang utama adalah untuk simulasi fenomena antrian di dunia nyata, serta fenomena antrian di pengolahan data.

* 1. Penerapan Queue dalam Kehidupan Sehari – hari
* Pada Pembelian Tiket

Dalam kehidupan sehari-hari kita bisa dapati melalui penerapan pembelian tiket kereta api, tiket pesawat, tiket kapal laut, pembayaran tiket tol, pembayaran listrik, pembayaran air, dan lain sebagainya. Saat mengantri di loket untuk membeli tiket. Istilah yang cukup sering dipakai seseorang masuk dalam sebuah antrian adalah enqueue. Dalam suatu antrian, yang datang terlebih dahulu akan dilayani lebih dahulu. Istilah yang sering dipakai bila seseorang keluar dari antrian adalah dequeue. Walaupun berbeda implementasi, tapi pada prinsipnya sama saja. Aplikasi dalam pembelian tiket kereta api:

1. Enqueue : Seseorang membeli tiket melalui tempat pembayaran tiket yang

disediakan.

1. Dequeue : Setelah membeli tiket, langsung menuju tempat penungguan kereta,

dengan sebelumnya petugas memeriksa cek tiket tersebut.

1. Clear : Pembeli tiket tersebut telah terhapus dari antrian karena sudah melewati

pembayaran administrasi tersebut.

1. IsEmpty : Petugas tiket Kereta melihat tidak ada lagi yang ingin membeli tiket

kereta.

1. IsFull : Petugas Tiket Kereta melihat masih ada pembeli tiket kereta.

* Antrian Mobil di Pintu Tol



Ketika sebuah mobil datang, dari belakang akan menuju kedepan dari antrian. Setelah mobil mendapatkan karcis tol, antrian yang berada didepan akan maju. Pada saat menempatkan data pada ujung (tail) dari queue disebut dengan enqueue. Pada saat memindahkan data dari kepala (head) sebuah queue disebut dengan dequeue.

* 1. Penerapan Aplikasi Qeueu Dalam Aplikasi Komputer
* Pada Aplikasi Download Manager IDM

Queue juga dipakai sebagai salah satu fitur dari Internet Download Manager atau yang biasa disebut dengan IDM. Fitur ini sangat membantu bagi para pecinta download. Fitur ini akan membantu pengguna untuk mendownload file yang anda pilih satu-persatu, jadi sebanyak apapun anda mendownload, tetapi akan tetap dibuat antrian atau istilahnya queueing. Sistem yang diterapkan dalam fitur ini adalah pada saat download file, maka IDM akan mendownload satu per satu, hingga download file selesai maka baru akan secara otomatis mendownload file berikutnya.

* Printer Sharing

Pada suatu jaringan terdapat beberapa aplikasi yang dapat membantu kita untuk mempermudah pekerjaan kita untuk melakukan cetak hanya dengan menggunakan 1 printer yang dapat dipakai oleh banyak orang yaitu dengan cara printer sharing. Aplikasi ini menggunakan cara kerja queue atau juga disebut dengan antrian, untuk mencetak suatu document yang hanya menggunakan 1 printer yang dapat diakses oleh orang banyak, penggunaan queue diperlukan untuk mengatur sistem jaringan agar tidak terjadi error, dengan cara siapa yang terlebih dahulu mencetak suatu dokumen, dia akan dilayani terlebih dahulu untuk mencetak dokumen dan jika ada seseorang yang ingin mencetak dokumen dia akan dimasukan kedalam antirian dan menunggu untuk dapat mencetak sebuah document.

* 1. Penerapan Aplikasi Queue dalam Sistem Produksi

Dalam sistem produksi terdapat banyak sekali aplikasi antrian, misalnya pada mesin pengisi (filling) botol minuman otomatis di pabrik. Mesin ini digunakan agar mempermudah pekerjaan, meminimalisir waktu, dan masih banyak lagi manfaat lainnya. Cara kerja dari mesin filling otomatis ini yaitu :

1. Beberapa botol disiapkan pada tempatnya.

2. Kemudian mesin akan mengisi air kedalam botol sesuai dengan posisinya.

3. Botol yang diletakkan paling depan akan diisi oleh mesin paling awal, setelah

botol paling depan terisi baru akan berjalan ke botol yang dibelakangnya, dan seterusnya.

Prinsip queue atau antrian digunakan dalam mesin ini, karena FIFO (First In First Out). Jadi botol yang paling awal/depan sampai pada pengisi air, maka botol itu juga yang akan pertama keluar/diisi.

# BAB III

## PENUTUP

## Kesimpulan

Queue (antrian) adalah salah satu list linier dari struktur data yang beroperasi dengan cara First In First Out (FIFO) yaitu elemen pertama yang masuk merupakan elemen yang pertama keluar. 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.

Sebuah queue dalam program setidaknya harus mengandung tiga variabel, yakni: head untuk penanda bagian depan antrian, tail unttuk penanda bagian belakang antrian, dan array data untuk menyimpan data-data yang dimasukkan ke dalam queue tersebut.

Pada queue ada operasi – operasi dasar, 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.

Ada beberapa queue, yaitu: queue dengan linear array yaitu queue yang dibuat seakan-akan merupakan suatu garis lurus dengan satu pintu masuk dan satu pintu keluar, queue dengan circular array yaitu queue yang dibuat seakan-akan merupakan sebuah lingkaran dengan titik awal (head) dan titik akhir (tail) saling bersebelahan jika array tersebut masih kosong, dan queue dengan linked-list.