QUEUE

TUJUAN UMUM

KONSEP QUEUE DAN KARAKTERISTIKNYA

TUJUAN KHUSUS

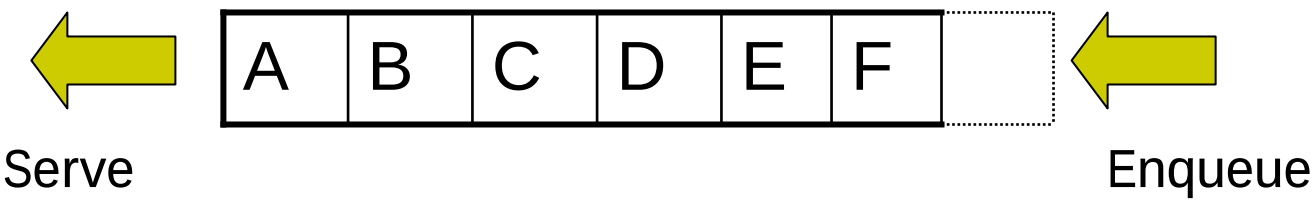
- Operasi dalam Queue
- Proses Enqueu
- Proses Dequeu
- Circular Queue
- Implementasi Queu dalam linked list
- Priority Queue
- Representasi Priority Queue

QUEUE

Sistem penyimpanan data dengan mekanisme First In First Out(FIFO).

Queue merupakan tipe data abstrak yang banyak digunakan dalam dalam printer spooler dan berbagai algoritma untuk simulasi.

Bentuk Umum Stack:



A merupakan elemen yang pertama masuk dan akan menjadi elemen yang pertama keluar

Operasi dalam QUEUE:

1. Create() Menciptakan queue baru dalam keandaan kosong
2. Enqueue(e) Memasukkan data baru dari variabel ke dalam Queue
3. Serve(*e) atau dequeue(*e) Mengambil data dari queue untuk disimpan di variabel e.

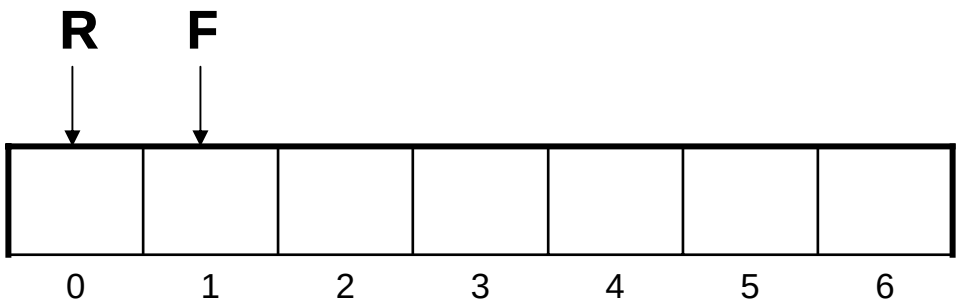
4. Empty( ) Memeriksa apakah queue dalam keadaan kosong
5. Full( ) Memeriksa apakah queue dalam keadaan penuh.
6. Clear( ) Menghapus semua data yang ada dalam Queue.

Implementasi Queue dengan Array

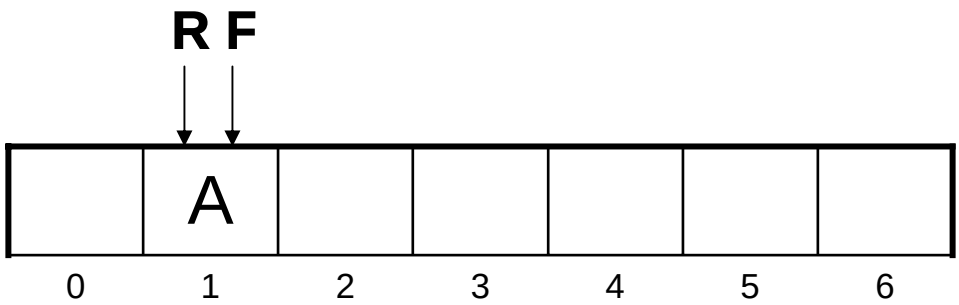
```
#define pj_max 7
typedef int elemen type;
elemen_type Queue[pj_max};
int FRONT;REAR;
void create( )
{
FRONT=1;REAR=0;
}
int full( )
{if (REAR==pj_max-1) return 1;else return 0;
}
int empty( )
{
if (REAR<FRONT) return 1;else return 0;
}
void enqueue(elemen_type e)
{
if (!full){REAR++;Queue[REAR]=e;};
}
void dequeue(elemen_type *e)
{
if (!empty){*e=queue[FRONT];FRONT- -;};
}
void clear( )
{
FRONT=1;REAR=0;
}
```

Proses Enqueue

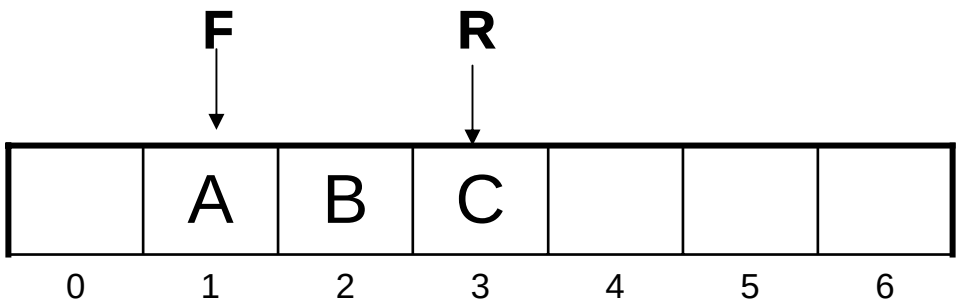
Keadaan awal setelah create( )



Enqueue(‘A’);

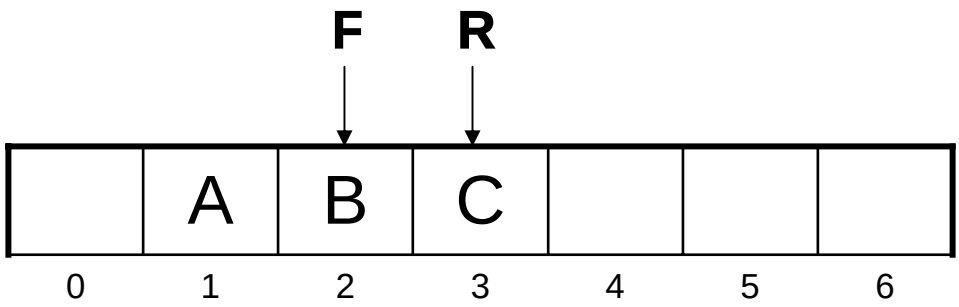


Enqueue(‘B’); Enqueue(‘C’);

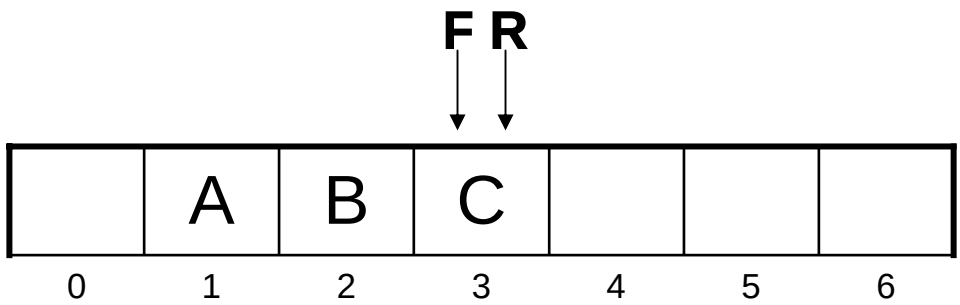


Proses Dequeue

Dequeue(&e)

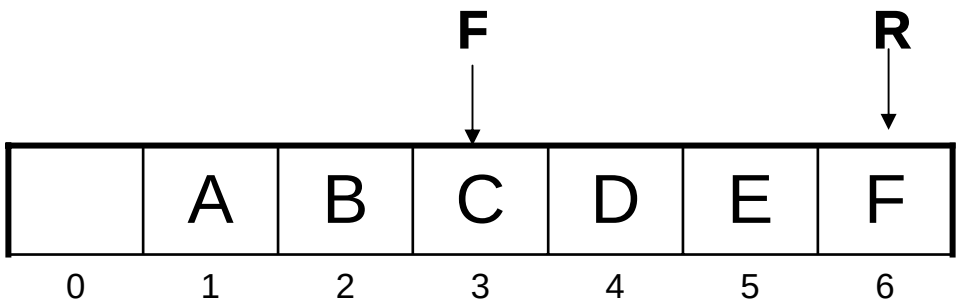


Dequeue(&e)



Kemudian dilakukan:

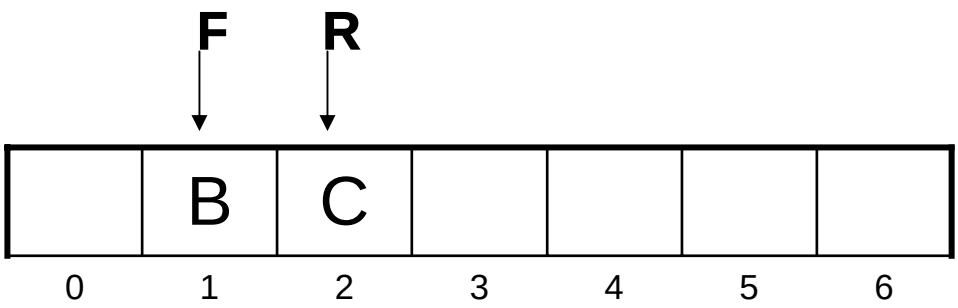
Enqueue('D'); Enqueue('E'); Enqueue('F');



Pada posisi ini Enqueue('G') tidak dapat dilaksanakan karena Queue sudah dianggap penuh.

Untuk memperbaiki keadaan di atas, maka pada setiap operasi dequeue, seluruh elemen digeser kearah posisi 1, dengan demikian posisi pointer FRONT tetap berada di posisi 1.

Dequeue(&e);




```
void dequeue(elemen_type *e)
{
if (!empty)
    {
        *e = Queue[FRONT];
        for (i=1;i<=R;i++)
            queue[i] = queue[i+1];
    };
REAR--;};
}
```

Kelemahan dari cara ini , adalah proses Serve atau dequeue menjadi lambat.

LATIHAN

1. Alternatif yang dapat dilakukan unatuk memperbaiki masalah kosongnya posisi dibagian depan Queue, adalah dengan menetapkan langkah-langkah berikut:Pergeseran elemen ke arah paling depan (posisi 1) ditunda hingga posisi REAR menyentuh posisi belakang(pj_max-1). Begitu REAR mencapai pj_max –1, maka seluruh elemen digeser ke arah paling depan.

   Bila pj_max=7, elemen_type = integer.

   untuk uarutan operasi:

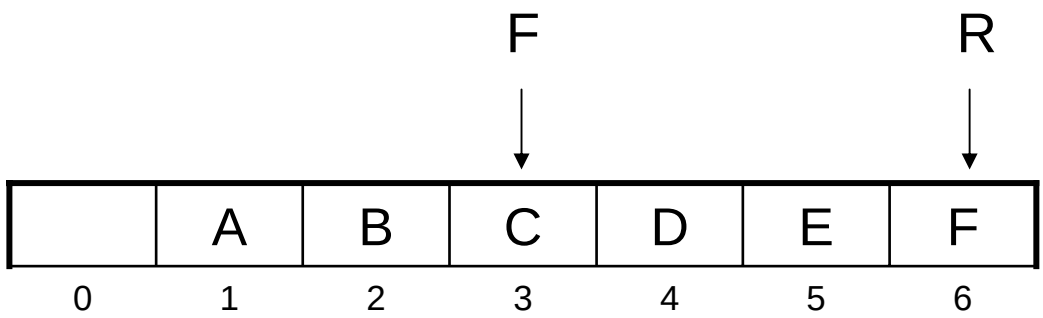
a. create( );
b. enqueue(23);
c. enqueue(21);
d. enqueue(45);
e. dequeue(&e);cout«e;
f. dequeue(&e);cout«e;
g. enqueue(17);
h. enqueue(16);
i. enqueue(15);
j. enqueue(14);
k. enqueue(13);
l. dequeue(&e);cout«e;
m. dequeue(&e);cout«e;

Gambarkan proses pergeseran elemen dan tuliskan output yang dicetak.

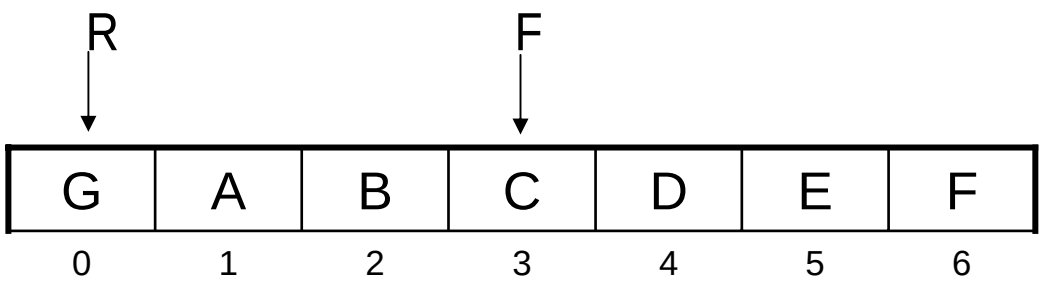
2. Apa kelemahan dari cara pergeseran disebutkan diatas?

Circular Queue

Ketika keadaan penuh – tetapi sebetulnya belum penuh – dialami oleh queue, maka degan sedikit modifikasi persoalan tersebut dapat dipecahkan dengna cara memberikan jalan pada pointer REAR untuk melingkar sehingga dapat menjapai posisi 0.



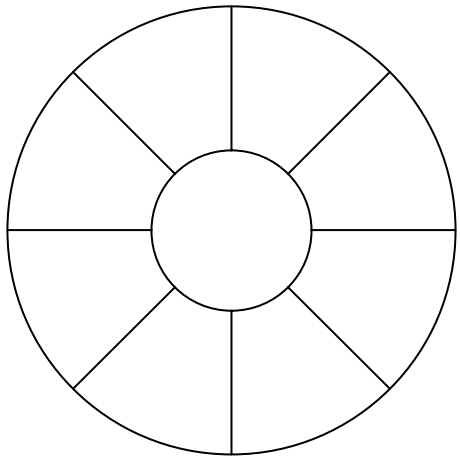
Dengan demikian Enqueue('G') tidak mendapat masalah.



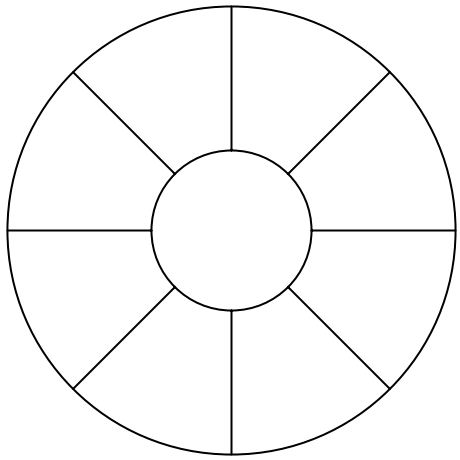
Implementasi Circular Queue

```
#define pj_max 8
typedef int elemen_type;
elemen_type Queue[pj_max];
int FRONT, REAR;
void create( );
{
FRONT = 0; REAR = pj_max – 1;
}
void enqueue(elemen_type e)
{
if (full( )) cout «”Queue Penuh”; else
      {
      REAR++
      REAR=REAR%pj_max;
      queue[REAR]=e;
      }
}
void dequeue(elemen_type *e)
{
if (empty( )) cout «”Queue Kosong”); else
      {
      *e = Queue[REAR] – e;
      FRONT++;
      FRONT = FRONT % pj_max;
      }
}
```

Keadaan Circular Queue setelah create( );



Enqueue: 21,34,68,17,53,80, dan 71



Keadaan seperti ini dilaporkan sebagai queue penuh

Diposisi manapun FRONT dan REAR

Apabila REAR berada dibelakang FRONT sebanyak 1 posisi dalam arah jarum jam, maka Queue Kosong.

Apabila REAR berada dibelakang FRONT sebanyak 2 posisi dalam arah jarum jam, maka Queue Penuh.

Latihan

1. Untuk circular queue dengan pj_max=10, dan dengan elemen_type int, gambarkan keadaan Queue untuk setiap langkah atau group langkah atau group langklah berikut:

a. create( );
b. enqueue(43);
c. enqueue(57); enqueue(21); enqueue(18);
d. enqueue(70); enqueue(15); enqueue(36);
e. enqueue(13); enqueue(29); enqueue(84);
f. dequeue(x);cout«x;endl;
g. dequeue(x);cout«x; endl;
h. dequeue(x);cout«x; endl;
i. enqueue(30); enqueue(93); enqueue(42);
j. dequeue(x);cout«x; endl;

2. Bila diketahui suatu circular Queue dengan pj-max = 20

a. Berapa kapasitas Circular Queue
b. Bila posisi FRONT = 6, posisi REAR = 5, berapa jumlah elemen queue
c. Bila posisi FRONT = 9, posisi REAR = 7, berapa jumlah elemen queue
d. Bila posisi FRONT = 8, posisi REAR = 8, berapa jumlah elemen queue
e. Bila jumlah elemen queue = 15 dan posisi REAR =19,dimana posisi FRONT