

Type TQueue = <wadah:array[1..10] of character,  
                  head:integer,  
                  tail:integer >

{Queue model I, kondisi head 0 atau 1}

{pergeseran maju pada elemen ketika dequeue}

Procedure CreateQueue(output Q:TQueue)

{I.S: - ; F.S: Q terdefinisi}

{Proses: mengisi elemen wadah dengan ' ',  
          head 0, tail 0}

Function Head(Q:TQueue) -> integer

{mengembalikan posisi elemen terdepan}

Function Tail(Q:TQueue) -> integer

{mengembalikan posisi elemen terakhir}

Function InfoHead(Q:TQueue) -> character

{mengembalikan nilai elemen terdepan}

Function InfoTail(Q:TQueue) -> character

{mengembalikan nilai elemen terakhir}

Function isEmptyQueue(Q:TQueue) -> boolean

{mengembalikan true bila Q kosong}

Function isFullQueue(Q:TQueue) -> boolean

{mengembalikan true bila Q penuh}

Procedure Enqueue(input/output Q:TQueue, input  
e:character)

{I.S: Q,e terdefinisi, Q mungkin kosong }

{F.S: Q tetap, atau infoTail(Q)=e }

{Proses menambah elemen e ke ekor Q bila belum  
penuh}

Procedure Dequeue(input/output Q:TQueue, output  
e:character)

{I.S: Q terdefinisi, mungkin kosong }

{F.S: Q tetap, atau e berisi infoHead(Q) lama }

{Proses menghapus elemen e dari head Q bila belum  
kosong}

{lalu geser maju 1 langkah semua elemen di belakang  
head}

Procedure PrintQueue(input Q:TQueue)

{I.S:-; F.S:-; Proses: menampilkan kondisi wadah Q }

Procedure ViewQueue(input Q:TQueue)

{I.S:-; F.S:-; Proses: menampilkan info elemen tak  
kosong Q}

Function sizeQueue(Q:TQueue) -> integer

{mengembalikan panjang/banyak elemen}

Procedure CreateQueue(output Q:TQueue)

{I.S: - ; F.S: Q terdefinisi}

{Proses: mengisi elemen wadah dengan ' ', head 0, tail 0}

Kamus Lokal

i : integer

Algoritma

Q.head <-- 0

Head(Q) <-- 0

Q.tail <-- 0

i traversal [1..10]

Q.wadah[i] <-- ' '

Function Head(Q:TQueue) -> integer  
{mengembalikan posisi elemen terdepan}

Kamus lokal  
{ - }

Algoritma  
~~--> TQueue.head~~

Kamus lokal

Algoritma  
-> Q.head

Function Tail(Q:TQueue) -> integer  
  {mengembalikan posisi elemen terakhir}

Kamus lokal

{ - }

Algoritma

--> Q.tail

Function InfoHead(Q:TQueue) -> character  
{mengembalikan nilai elemen terdepan}

Kamus lokal

{ - }

Algoritma

if NOT isEmpty(Q) then  
--> Q.wadah[Q.head]

else  
--> ''

Function InfoTail(Q:TQueue) -> character  
{mengembalikan nilai elemen terakhir}

Kamus lokal

{ - }

Algoritma

```
if NOT isEmpty(Q) then
    --> Q.wadah[Q.tail]
else
    --> ''
```



Function isEmptyQueue(Q:TQueue) -> boolean  
{mengembalikan true bila Q kosong}

Kamus Lokal

empty: boolean

i : integer

Algoritma

```
empty <-- true
i traversal [1..10]
  if ( Q.wadah[i] != ' ' ) then
    empty <-- false
--> empty
```

-> Head(Q) = 0 AND Tail(Q) = 0

```
empty <-- true
i = 1
while ( empty = true AND i <= 10 ) do
  if ( Q.wadah[i] != ' ' )
    empty <-- false
  i = i + 1
--> empty
```

Function isFullQueue(Q:TQueue) -> boolean  
{mengembalikan true bila Q penuh}

Kamus Lokal

full : boolean

i : integer

Algoritma

full <-- true

i traversal [1..10]

if ( Q.wadah[i] = ' ' ) then

full <-- false

--> full

-> Head(Q) = 1 AND Tail(Q) = 10



Procedure Enqueue(input/output Q:TQueue, input e:character)

{I.S: Q,e terdefinisi, Q mungkin kosong }

{F.S: Q tetap, atau infoTail(Q)=e }

{Proses menambah elemen e ke ekor Q bila belum penuh}

cek 4 kondisi:

- penuh

- kosong ✓

- berisi

- 1 elemen ✗

if not isFullQueue(Q) then

if isEmptyQueue(Q) then

head (Q) <-- 1      Head(Q) <-- Head(Q) + 1

tail(Q) <-- tail(Q)+1

Q.wadah[tail(Q)]<--e

else

Procedure Dequeue(input/output Q:TQueue, output e:character)

{I.S: Q terdefinisi, mungkin kosong }

{F.S: Q tetap, atau e berisi infoHead(Q) lama }

{Proses menghapus elemen e dari head Q bila belum kosong}

{lalu geser maju 1 langkah semua elemen di belakang head}

if IsEmptyQueue(Q) then  
 $e \leftarrow \text{'-'}$   
 else  
 $e \leftarrow \text{InfoHead}(Q) \quad \{ Q.wadah_{Q.head} \}$   
 for traversal 1 ... (Tail(Q)-1)  
 $Q.wadah_i \leftarrow Q.wadah_{i+1}$   
 $Q.wadah_{\text{Tail}(Q)} \leftarrow \text{'-'}$  //  $\text{InfoTail}(Q) \leftarrow \text{'-'}$   
 $\text{Tail}(Q) \leftarrow \text{Tail}(Q) - 1$   
 if  $\text{Tail}(Q) = 0$  then  
 $\text{Head}(Q) \leftarrow 0$

if IsEmptyQueue(Q) then  
 $e \leftarrow \text{'-'}$   
 else  
 $e \leftarrow \text{InfoHead}(Q) \quad \{ Q.wadah_{Q.head} \}$   
 for traversal 1 ... (Tail(Q)-1)  
 $Q.wadah_i \leftarrow Q.wadah_{i+1}$   
 $Q.wadah_{\text{Tail}(Q)} \leftarrow \text{'-'}$  //  $\text{InfoTail}(Q) \leftarrow \text{'-'}$   
 $\text{Tail}(Q) \leftarrow \text{Tail}(Q) - 1$   
 if  $\text{Tail}(Q) = 0$  then  
 $\text{Head}(Q) \leftarrow 0$

Diagram illustrating the dequeue operation on a queue. The queue is represented as an array of 4 slots. Initially, the queue contains elements A, B, C, and D. The head pointer (H) points to the first slot (index 1) and the tail pointer (T) points to the last slot (index 4). The operation involves shifting all elements one position to the left, resulting in the queue containing B, C, D, and an empty slot. The head pointer (H) now points to the second slot (index 2) and the tail pointer (T) points to the third slot (index 3). The element A is removed from the queue.

Queue index	Queue value	Tail(Q)	InfoTail
1	A	4	D
2	B	4	B
3	C	4	<del>C</del>
		3	C

Procedure PrintQueue(input Q:TQueue)  
{I.S:-; F.S:-; Proses: menampilkan kondisi wadah Q }

Kamus Lokal

i : integer

Algoritma

i traversal [1..10]  
    output (Q.wadah[i])



Procedure ViewQueue(input Q:TQueue)  
{I.S:-; F.S:-; Proses: menampilkan info elemen tak  
kosong Q}

Kamus Lokal

i : integer

Algoritma

if (Q.head = 1) then

    i traversal [1..Q.tail]

        output (Q.wadah[i])

else

    output "antrean kosong"

Function sizeQueue(Q:TQueue) -> integer  
{mengembalikan panjang/banyak elemen}

Kamus lokal

~~i: integer~~

~~P: integer~~

Algoritma

→ Q.tail ✓

Kamus lokal

i : integer

P: integer

Algoritma

P ← 0

if(Q.head = 1) then

  i traversal [1 .. T.tail]

~~if(Q.wadah[i] != ' ')~~

~~P = P + 1~~

    else

      P = P

→ P







