Modul T_Queue

```
      DEKLARASI/DEFINISI&SPESIFIKASI TIPE & PROTOTIPE
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}
```

```
      DEKLARASI/DEFINISI&SPESIFIKASI TIPE & PROTOTIPE
Procedure Enqueue(input/output Q:TQueue, input e:charac
{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:charac
{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 koso
Function sizeQueue(Q:TQueue) -> integer
{mengembalikan panjang/banyak elemen}
```

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

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}

{asumsi Q mungkin kosong}


Kamus Lokal
    -

Algoritma
    if Q.head = 0 then
        -> '_'
    else
        -> Q.wadah[Q.head]
```

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

{asumsi Q mungkin kosong}


Kamus Lokal
    -

Algoritma
    if Q.tail = 0 then
        -> '_'
    else
        -> Q.wadah[Q.tail]
```

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


 Kamus Lokal
      -

 Algoritma
      if Q.tail = 0 then    AND Q.head = 0
           -> True
      else
           -> False
```

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


Kamus Lokal
    -

Algoritma
    if Q.tail = 10 then   AND Q.head = 1
        -> True
    else
        -> False
```

```
Procedure CreateQueue(output Q:TQueue)
{I.S: - ; F.S: Q terdefinisi}
{Proses: mengisi elemen wadah dengan ‘ ’,
head 0, tail 0}

Kamus lokal
    i : integer {iterator}

Algoritma
    i traversal 1..10
        Q.wadah[i] <-- '_'
    Q.head <-- 0
    Q.tail <-- 0
```

```
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}


Kamus Lokal

Algoritma
if not(isFullQueue(Q)) then
   if IsEmptyQueque(Q) then
      Q.head <-- 1
   Q.tail <-- Q.tail +1
   Q.wadah[Q.tail] <-- e
```

Procedure Dequeue(input/output Q:TQueue, output e:character)
{I.S: Q,e terdefinisi, Q 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 }

```
Kamus Lokal
i : integer {iterator}
Algoritma
If NOT(IsEmptyQueue(Q)) then
  e <-- Q.wadah[Q.head]
  i traversal 1..(Q.tail - 1)
     Q.wadah[i] <-- Q.wadah[i+1]
  Q.wadah[Q.tail] <-- '_'
  Q.tail <--  Q.tail -1
  if Q.tail = 0 then
     Q.head <-- 0
```

```
Kamus Lokal
i : integer {iterator}
Algoritma
if(IsEmpty(Q)) then
  e <-- '_'
else
   if(Head(Q) > 1) then
      e<-- InfoHead(Q)
      Q.tail <-- Q.tail -1
      i traversal 1..Q.tail
         Q.wadah[i] <-- Q.wadah[i+1]
      Q.wadah[i+1] <-- '_'
   else
      e<-- InfoHead(Q)
      Q.wadah[Q.head] <-- '_'
      Q.tail <-- Q.tail -1
      Q.head <-- 0
```

```
DEKLARASI/DEFINISI&SPESIFIKASI TIPE & PROTOTIPE
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 ViewQueue (Input Q:Tqueue)
{I.S : - }
{F.S : -}
{Proses : menampilkan info elemen tak
kosong}
Kamus lokal
 i : integer {iterator}
Algoritma
 if not(IsEmpty(Q)) then
    i traversal 1..Q.Tail
       output Q.wadah[i]
```

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

Algoritma
if(IsEmpty(Q)) then
   --> 0
else
     --> Q.tail - Q.head + 1
```

```
Procedure PrintQueque (input Q:TQueque)
{I.S : -; F.S : -; Proses : Menampilkan kondisi awal Q}

    Kamus Lokal
    i : integer {iterator}
    Algoritma
    i <-- 0
    while(i > 0) do
        output(Q.wadah[i]
         i <-- i +1
    {endwhile}
```

```
Procedure ViewQueque(input Q:TQueque)
{I.S :-; F.S:-; Proses : Menampilkan info elemen ta kosong}

Kamus Lokal
i : integer

Algoritma
i <-- 0
i traversal 1 . . Q.Tail
      if (Q.wadah[i] NOT = '_') then
          output Q.wadah[i]
```

```
function sizeQueqie (Q:TQueque )--> integer
{mengembalikan panjang/banyak elemen}

Kamus Lokal
a : integer
i : integer

Algoritma
if NOT isEmpthyQueque(Q) then
      i traversal 1 .. Q.Tail
            if Q.wadah[i] NOT = '_' then
                  a <-- a + 1
 output a
```

KAMUS

Kamus