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}

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 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

Algortima
```
        { Bila Queue kosong }
    if Q.tail = 0 AND Q.head = 0 then
        Q.wadah[Q.head + 1] <- e
        Q.wadah[Q.tail + 1] <- e
    else
        Q.tail <- Q.tail + 1
        Q.wadah[Q.tail] <- e
```

Hapus?

createQueue nya kok hilang? slid 4.OK enqueue ada di belakang bukan? di tuker gitu ?

1

```
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

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

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

2

Kamus lokal

Algoritma
--> Q.Head

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

Kamus lokal

Algoritma
    --> Q.tail

3

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

4

```
kamus lokal
     cek: boolean

algoritma
    if(Q.tail = 0 AND Q.head = 0) then
       cek<-- true
   else
       cek<-- false
   -->(cek)
```

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

5

```
kamus lokal
    cek:boolean

algoritma
    if(Q.head = 1 AND Q.tail = 10) then
       cek<-- true
   else
      cek<-- false
   -->(cek)
```

6

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

```
Kamus lokal

Algoritma
   if not isEmptyQueue(Q) then
      -->Q.wadah[Head(Q)]
   Else {Q kosong}
      ---> ' '
```

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

Kamus lokal

Algoritma
    if not isEmptyQueue(Q) then
        --> Q.wadah[Tail(Q)]
    Else { Q kosong}
        --> ' '

'@'

7

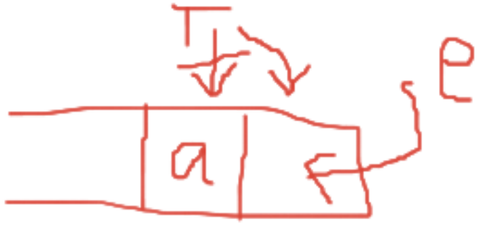
8

```
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 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}

kosong
tidak kosong
- tidak penuh
- penuh-->abaikan

Kamus lokal



Algoritma

```
if NOT isFullQueue(Q) then
    Q.tail <- Q.tail + 1
    Q.wadah[Q.tail] <- e
    if Q.tail = 1 then
        Q.head <-- 1
```

tidak penuh

bila sebelumnya kosong

Kamus lokal

Algoritma

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

```
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}

Kamus Lokal
   i : integer { iterator }

Algoritma
   if not(isEmptyQueue(Q)) then {Tidak kosong}
            Q.tail <- Q.tail - 1
      e <- Q.wadah[Q.head]
      i traversal[Q.head..Q.tail]
         Q.wadah[i] <- Q.wadah[i+1]
      Q.wadah[Q.tail+1] <- ' '
            if (Q.tail = 0) then {Kosong setelah dequeue}
                  Q.head <- 0
       else
            e <- ' '
```

enqueue diisi dong kasian jomblooo

yuk pindah enqueue wkwkkw

koreksi gais xixixi

kenapa disini paling rame sihh

traversal pake endtraversal?

ni dequeue

a b c d e f g

Setelah enqueue?

Urutannya ngaco semua wkwkwk

btw, deque kok di slide ini si wkwkwk

gbisa isNotEmptyQueue dong, wkwkwk

kenapa emang?

Kan ngga ada functionnya

cara mindahinnya gimana wkwk

harusnya dimana wkwk

janlup kalo tinggal 1 elemen, Q.headnya jadi 0

head = 0 berarti kosong kan?

jadinya gimana dong

Bentar coba

not(isEmptyQueue) kan jadinya kalo ngga kosong true

kasih endtraversal gais

okee hehe

Iya

iya, kalo ga kosong kan trus kehapus kan wkwk

dah bener blom si

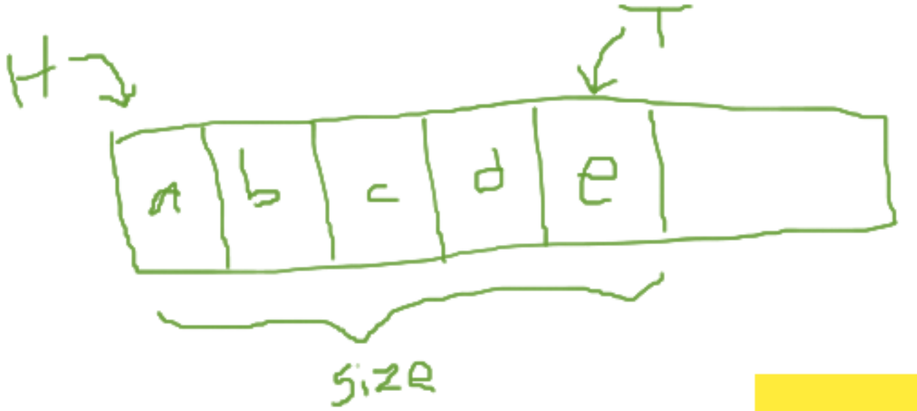
10

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

11

```
Kamus lokal :
i : integer

Algoritma
if(not isEmptyQueue(Q)) then
   i traversal [Q.head .. Q.tail]
       output(Q.wadah[i])
```

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



12

Kamus lokal
  i : integer
  size : integer

Algoritma
  if (not isEmptyQueue(Q)) then
    i traversal 1..Q.tail
      size <--ipemborosan assignment
  else
    size <- 0

  --> size

--> tail(Q) - head(Q) +1

5 - 1 + 1