**Home Task**

Write the algorithm of queue mechanism

1. Algorithm queue array alternative 1
2. IsEmpty

Function IsEmptyAlt1( Q : Queue ) → boolean

Kamus

Algoritma

→ ( Q.head = -1 and Q.tail = -1)

1. IsFull

Function IsFullAlt1( Q : Queue ) → boolean

Kamus

Algoritma

→ ( Q.head = 0 and Q.tail = NMax-1)

1. EnQueue

Procedure EnQueueAlt1 (input/output Q:Queue, input X:integer)

{IS. Queue mungkin kosong atau penuh, dan P berisi data

FS. X di-enqueue kedalam queue dengan Alternative 1, tampilan penuh apabila queue penuh}

Kamus

Algoritma

If (Q.head = 0 and Q.tail = Nmax-1) then //queue penuh

output(‘penuh’)

else

if (Q.head = -1 and Q.tail = -1) then

Q.head ← Q.head + 1 //queue kosong, head nambah 1

{end if}

Q.tail ← Q.tail + 1 //queue kosong dan tidak kosong, tail nambah 1

Q.info [Q.tail] ← X //data ditambahkan

{end if}

1. DeQueue

Procedure DeQueueAlt1 (input/output Q:Queue)

{IS. Queue mungkin kosong atau penuh

FS. Queue di-dequeue dengan Alternative 1 }

Kamus

I = integer

Algoritma

If (Q.head = 0 and Q.tail = -1) then //kondisi kosong

output(‘Stack kosong’)

else //tidak kosong

i ← 0 //dimajukan satu langkah kedepan

while ( i < Q.Tail ) do

Q.info[i] ← Q[i + 1]

I++

{end while}

Q.tail ← Q.tail – 1 //update tailnya, yaitu berkurang 1

If ( Q.tail = -1 ) then

Q.Head -1

{end if}

{end if}

1. Algorithm queue array alternative 2
2. IsEmpty

Function IsEmptyAlt2( Q : Queue ) → boolean

Kamus

Algoritma

→ ( Q.head = -1 and Q.tail = -1)

1. IsFull

Function IsFullAlt2( Q : Queue ) → boolean

Kamus

Algoritma

→ ( Q.head = 0 and Q.tail = NMax-1)

1. EnQueue

Procedure EnQueueAlt2(input/output Q: Queue, input P : infotype)

{ IS. Queue mungkin kosong atau penuh, dan P berisi data

FS. P di-enqueue kedalam Queue, dengan Alternative 2 }

Kamus

i,j : integer

Algoritma

if ( Q.head = 0 and Q.tail = NMax-1 ) then // Queue Penuh

output ('Queue Penuh')

else if ( Q.head = -1 and Q.Tail = -1 ) then // Queue Kosong

Q.head ← Q.head + 1

Q.tail ← Q.tail + 1

Q.info[Q.tail] <- P

else if ( Q.tail = NMax-1 ) then // Kondisi Khusus, jika tail ada di ujung kanan, dan tidak penuh

i ← Q.head // maka seluruh antrian dimajukan ke posisi paling depan semua

j ← 0

while ( i < Q.tail ) do

Q.info[j] <- Q.[i]

i ← i + 1

j ← j + 1

{end while}

Q.head ← 0 // maka head menjadi 0 dan tail sama dengan j

Q.tail ← j

Q.info[Q.tail] ← P // data di enqueue di posisi tail

else

Q.tail ← Q.tail + 1 // kondisi biasa

Q.info[Q.tail] ← P

{end if}

1. DeQueue

Procedure DeQueueAlt2(input/output Q : queue)

{ IS. Queue mungkin kosong atau penuh

FS. Queue di-dequeue , dengan Alternative 2 }

Kamus

Algoritma

if ( Q.head = -1 and Q.Tail = -1 ) then //queue kosong

output('Stack kosong')

else // queue tidak kosong

p <- Q.info[q.head] // data yang di head disimpan di p

if ( Q.head = Q.tail ) then // jika 1 elemen dan posisi dimanapun dan tidak kosong

Q.head <- -1 //maka head dan tail pasti jadi -1 semua} Q.tail -1

else // kondisi biasa, headnya nambah 1

Q.head ++

{end if}

{end if}

1. Algorithm queue array alternative 3
2. IsEmpty

Function IsEmptyAlt3( Q : Queue ) → boolean

Kamus

Algoritma

→ ( Q.head = -1 and Q.tail = -1)

1. IsFull

Function IsFullAlt2( Q : Queue ) → boolean

Kamus

Algoritma

→ ( Q.head = 0 and Q.tail = NMax-1) or (Q.head = (Q.tail + 1))

1. EnQueue

Procedure EnQueueAlt3(input/output Q:queue, input p : infotype)

{ IS. Queue mungkin kosong atau penuh, dan P berisi data

FS. P di-enqueue kedalam Queue, dengan Alternative 3}

Kamus

Algoritma

if (Q.head = 0 and Q.Tail = NMax-1) or (Q.head = ( Q.tail + 1)) then // Queue penuh

output('Queue penuh')

else if ( Q.head = -1 and Q.Tail = -1 ) then // Queue kosong

Q.head ← Q.head + 1

Q.tail ← Q.tail + 1

if ( Q.tail = NMax-1 ) then // Kondisi khusus, tail sudah di ujung kanan dan tidak penuh

Q.tail ← 0

else // Kondisi biasa

Q.tail ← Q.tail + 1

{end if}

Q.info[Q.tail] ← p // data di-enqueue di posisi tail

{end if}

1. DeQueue

Procedure DeQueueAlt3(input/output Q:queue)

{ IS. Queue mungkin kosong atau penuh

FS. Queue di-dequeue dengan Alternative 3}

Kamus

Algoritma

if ( Q.head = -1 and Q.Tail = -1 ) then // Queue kosong

output('Stack kosong')

else

if Q.Tail = Q.Head then // data cuman 1 elemenm berada dimanapun

Q.head ← -1

Q.tail ← -1

else if ( Q.Head <- nMax-1 and Q.tail <> Q.head ) then // Kondisi khusus, head ada di ujung kanan dan tidak penuh

Q.head <- 0

else // Kondisi biasa

Q.head ← Q.head + 1

{end if}

{end if}