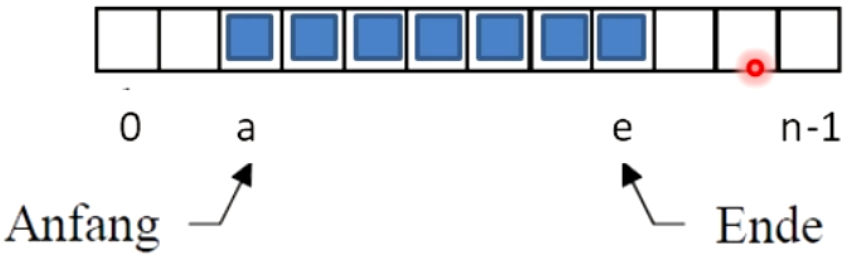
Eigenschaften

- implementiert mit [[Array]]
- FIFO Prinzip
  - first in first out



Operationen

Queue (Schlange):

Q[0...n-1]

Init: anz←0; e←-1; a←0

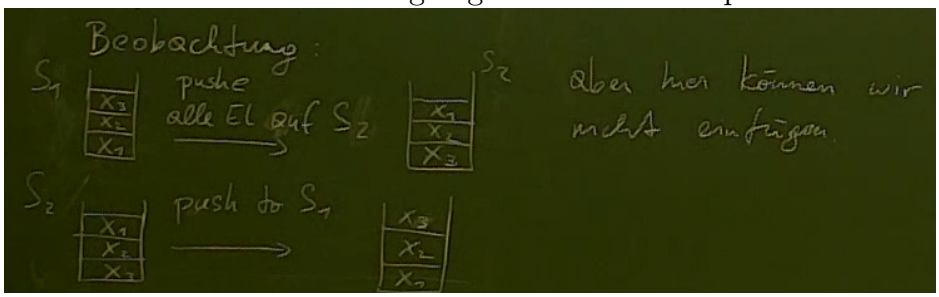
Einfügen:

```
PUT(Q,x)
1:   IF anz=n THEN „overflow“
2:   ELSE anz ← anz+1
3:        IF e<n-1 THEN e ← e+1
4:             ELSE e ← 0
5:        Q[e] ← x
```

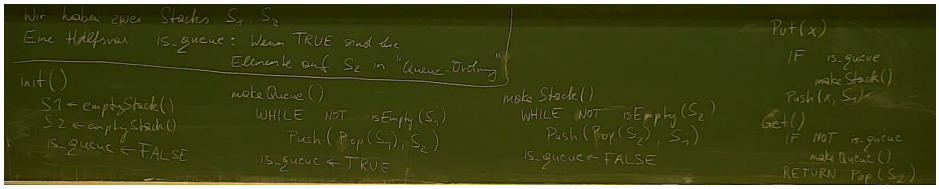
- O(1) für alle Operationen

Queue Implementation mittels [[Stack]]

- verwendet zwei Stacks S_1, S_2
  - S1 zum Einfügen
  - S2 zum Auslesen/Löschen
- Elemente von S1 wenn Auslesen/Löschen gefordert nach S2 pushen
- Elemente von S2 wenn Einfügen gefordert nach S1 pushen



- Algorithmus/Funktionen

- 
Wir haben zwei Stacks S_1, S_2
Eine Hilfsvar. is_queue: Wenn TRUE sind die
Elemente auf S_2 in "Queue-Order"
init()
$S_1 \leftarrow$ emptyStack()
$S_2 \leftarrow$ emptyStack()
is_queue $\leftarrow$ FALSE
makeQueue()
WHILE NOT isEmpty(S_1)
Push(Pop(S_1), S_2)
is_queue $\leftarrow$ TRUE
makeStack()
WHILE NOT isEmpty(S_2)
Push(Pop(S_2), S_1)
is_queue $\leftarrow$ FALSE
Put(x)
IF is_queue
makeStack()
Push(x, S_1)
Get()
IF NOT is_queue
makeQueue()
RETURN Pop(S_2)