Warteschlangen (Queues) als Datenstrukturen

Implementierung als Vektor oder
als Liste (einfach verkettet)

FIFO - first in first out

als Vektor pq mit $n > 0$ Elementen: int pq[n]

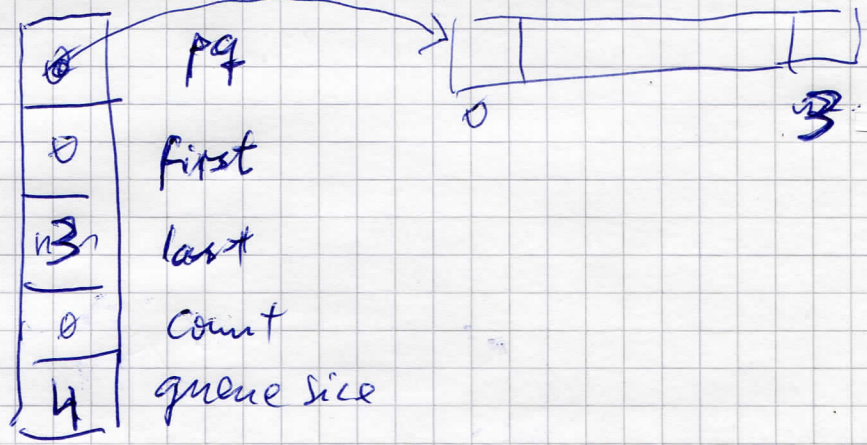
```
int first     // Kopf
int last      // Ende
int count     // Anzahl in Queue *q
int quenesize // max. Laenge (nur bei Vektor)
                 = n  (pq[n])
```

Operationen über Queue: (vgl. queue.pdf)

```
void init_quene(int n, quene *q);
    // Initialisierung q mit n Elementen
// Eintragen Element x am Ende:
    enquene(quene *q, int x);
// Erstes Element verläßt quene:
    int dequene(quene *q);
// Frage, ob quene leer?
    int empty(quene *q);
// Ausgabe quene:
    void printquene(quene *q);
```

①

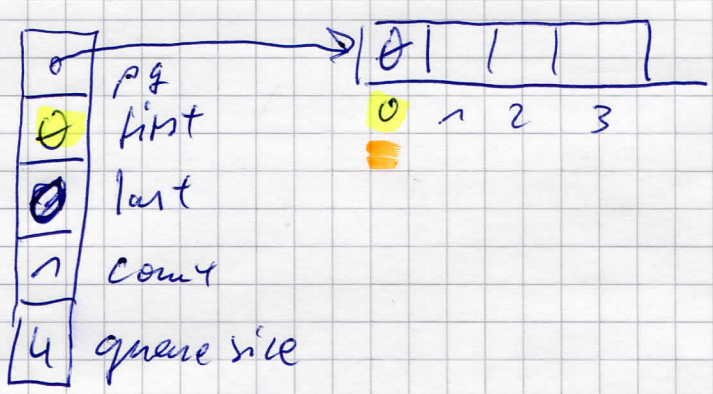
queue q



n=4 , init_queue(4,&q)

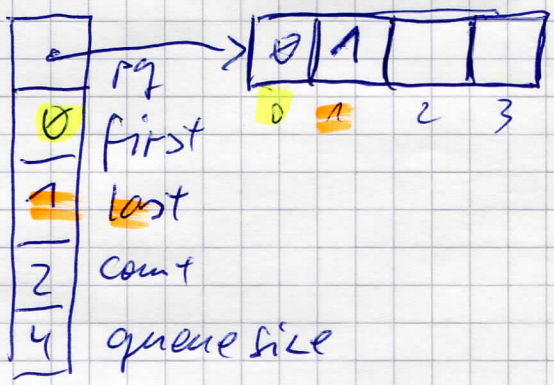
enqueue(&q, 0)

queue q



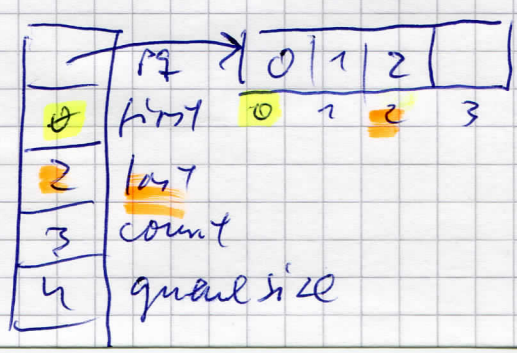
enqueue(&q, 1)

queue q



enqueue(&q, 2)

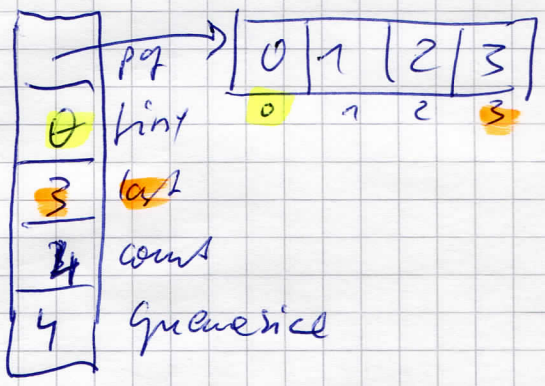
queue q



(2)

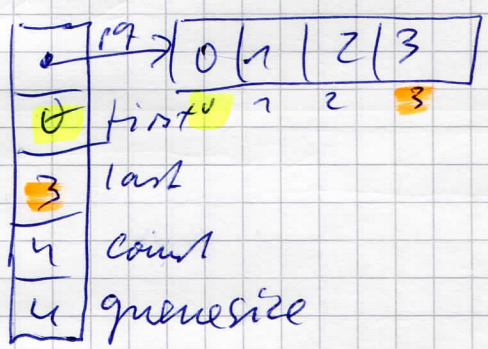
enqueue(&q, 3)

queue q



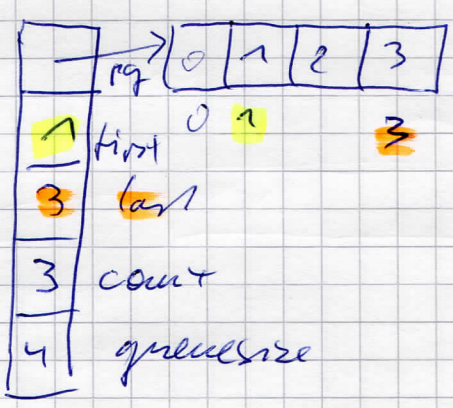
enqueue(&q, 4) q->count >= q->queuesize

queue q



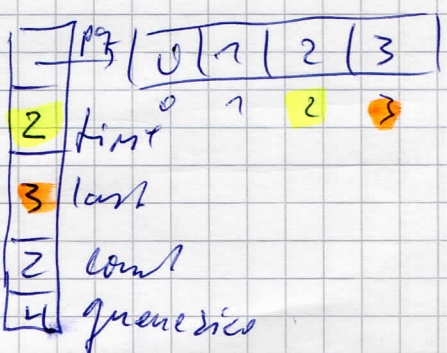
dequeue(&q) → x = q->pq[q->first] = 0

queue q



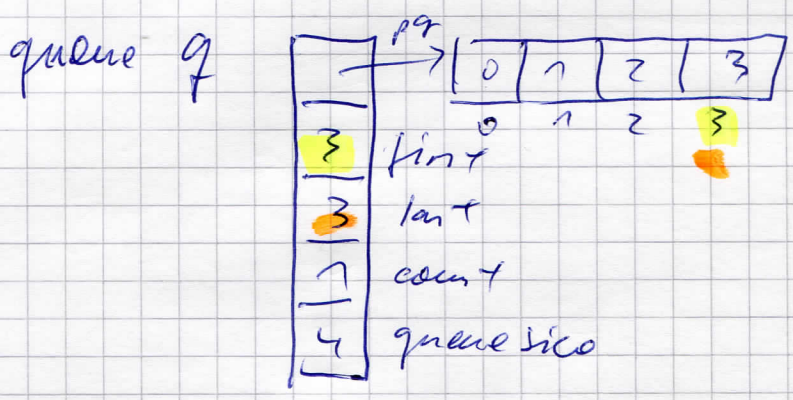
dequeue(&q) → x = pq[1] = 1

queue q

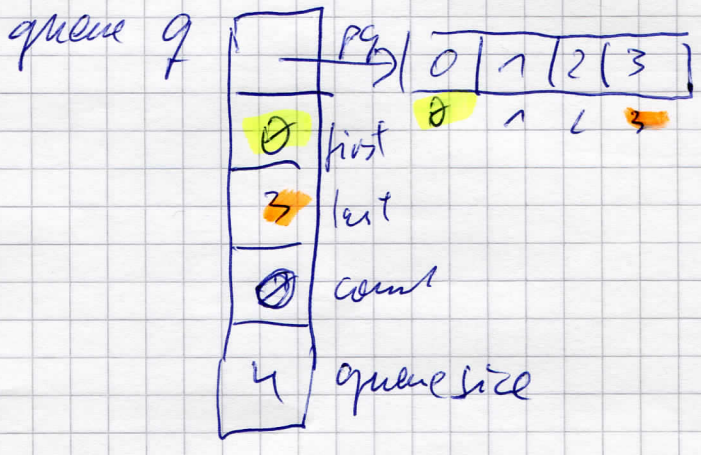


(3)

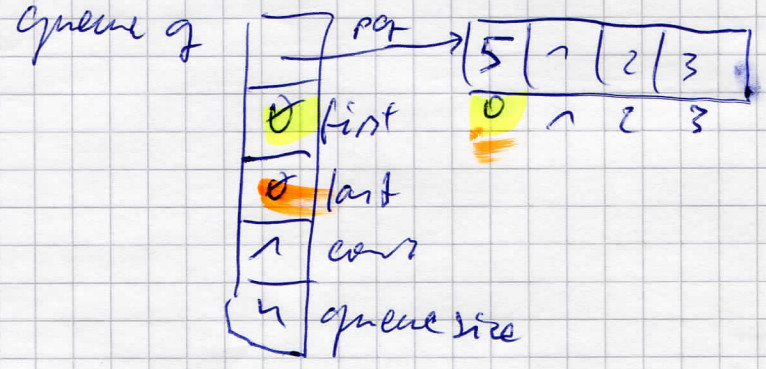
dequeue(&q) → x = pq[2] = 2



dequeu(&q) → x = pq[3] = 3



enqueue(&q, 5)



.
.
.