Ilustrasi Queue

By
Entin Martiana, S.Kom.

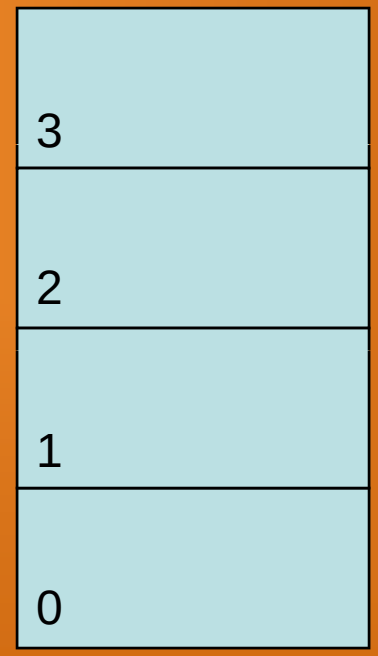
Deklarasi

```
struct Queue{
        int    Count;
        int    Front;
        int    Rear;
        int    Item[MAXQUEUE];
   };
```

Kondisi awal

MAX = 4

n = 0

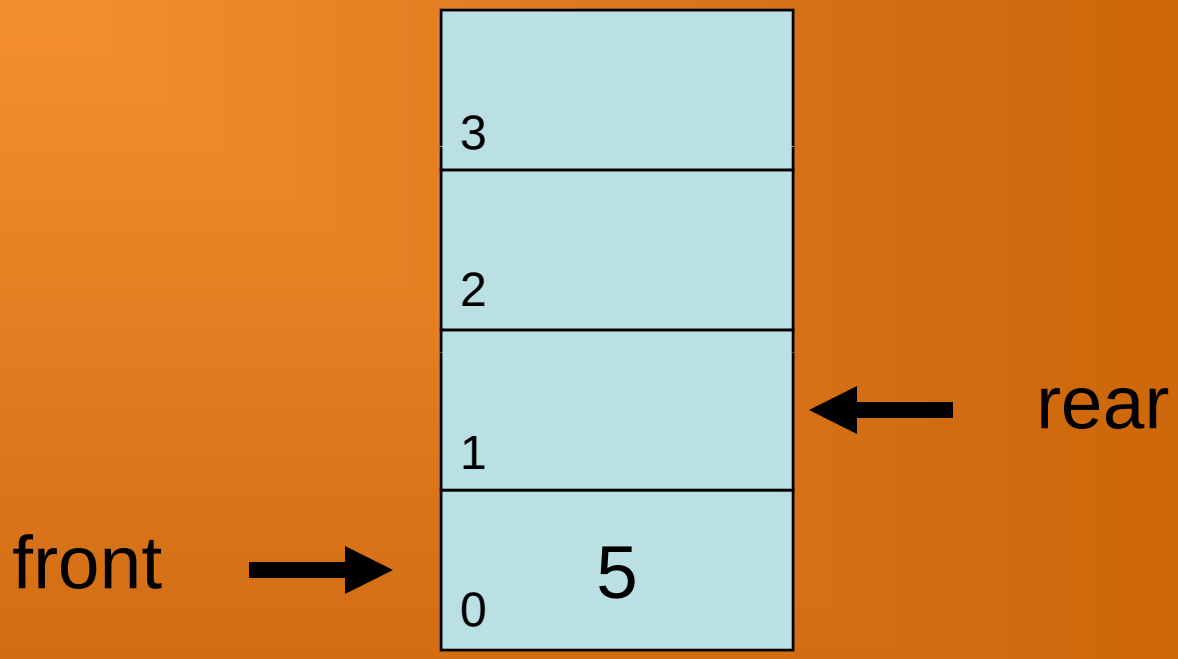


front → 0 ← rear

Enqueue : 5

MAX = 4

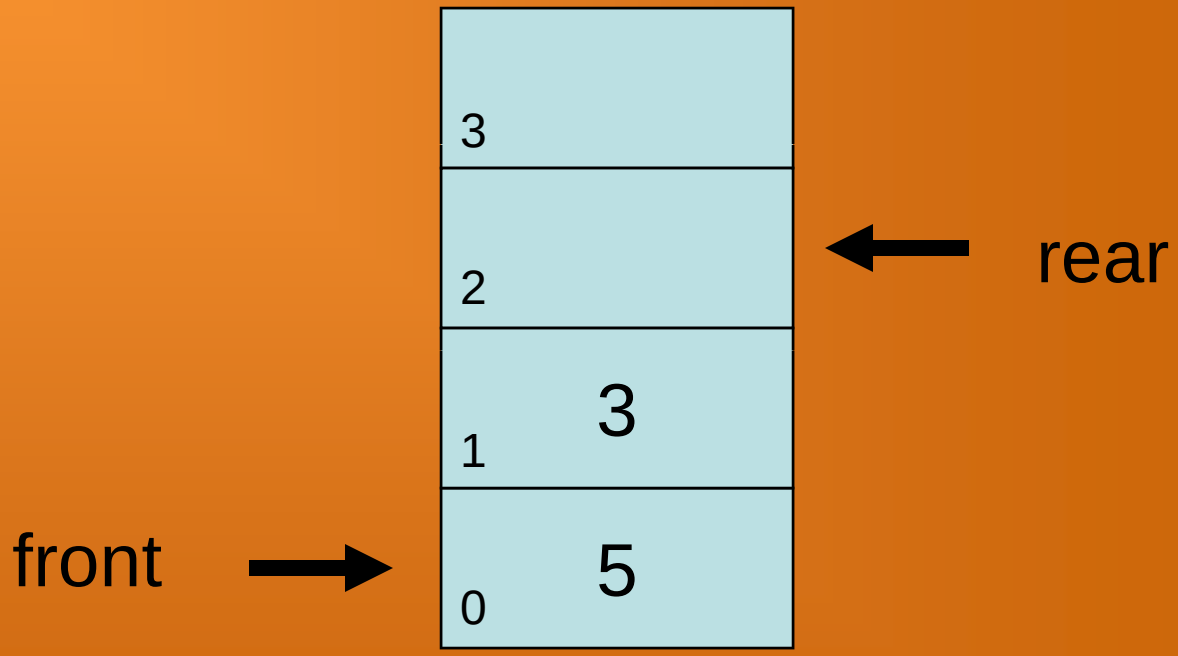
n = 1



Enqueue : 3

MAX = 4

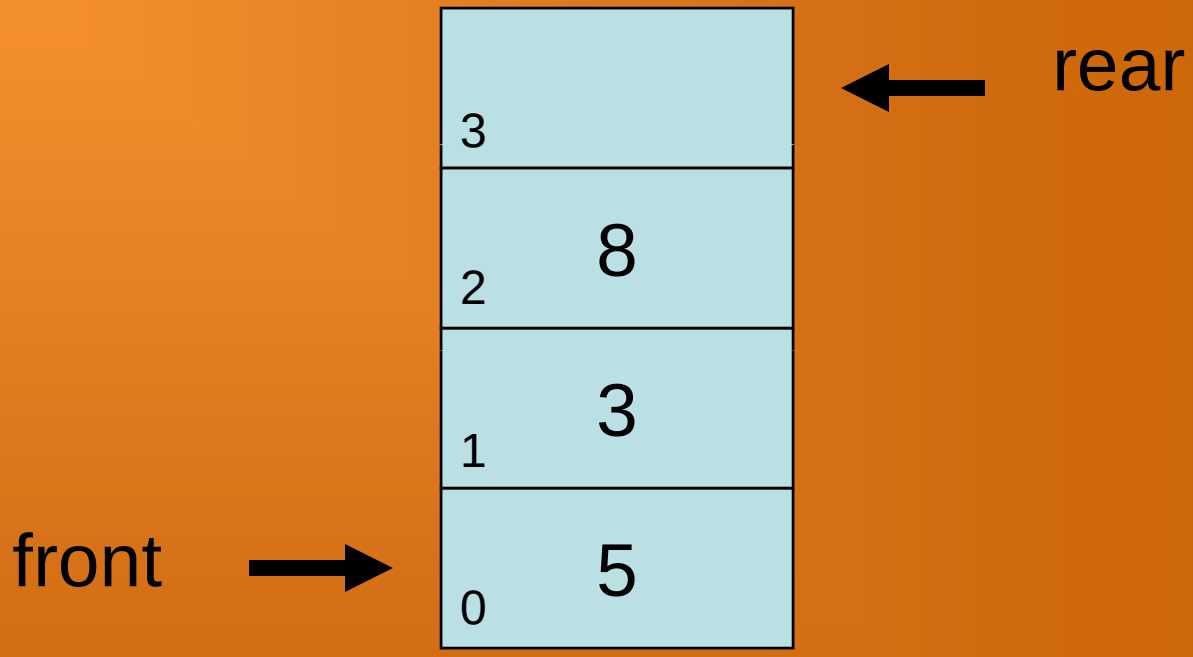
n = 2



Enqueue : 8

MAX = 4

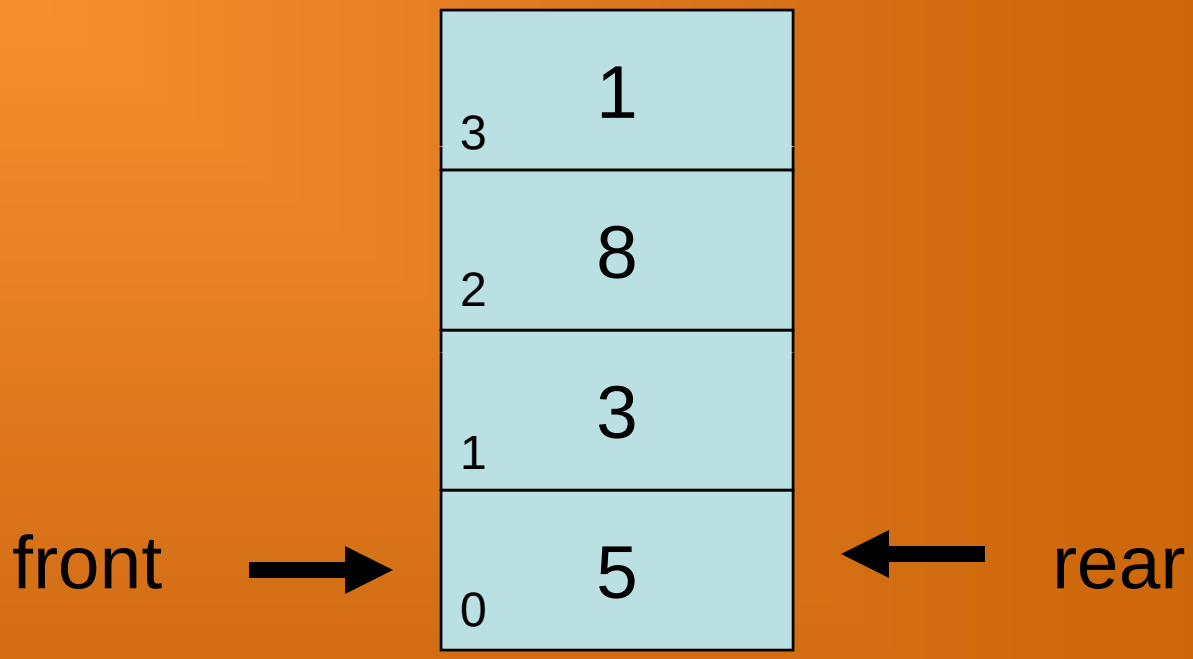
n = 3



Enqueue : 1

MAX = 4

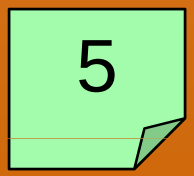
n = 4

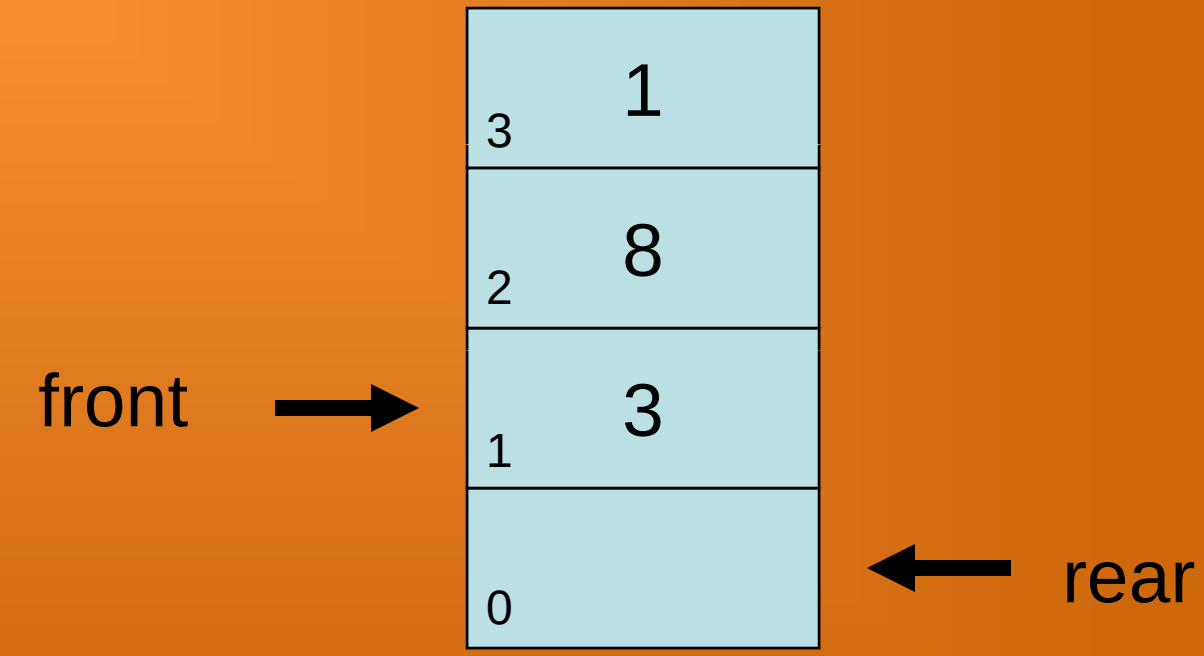


Dequeue

MAX = 4

n = 3



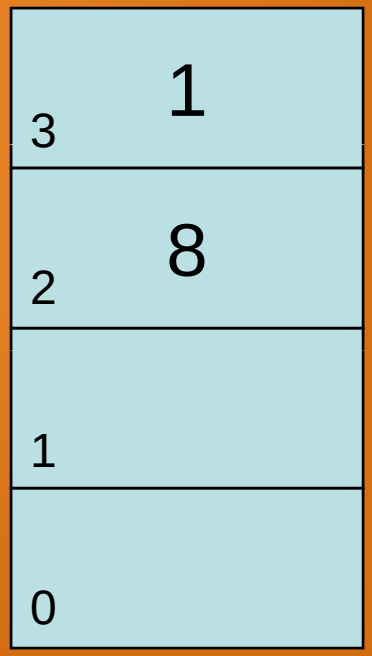


Dequeue

MAX = 4

n = 2

front



rear

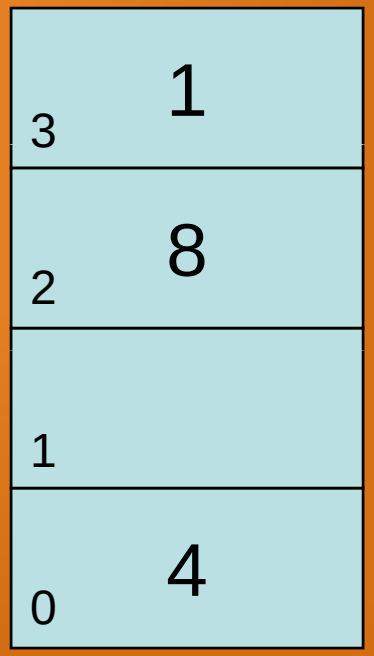
3

Enqueue : 4

MAX = 4

n = 3

front



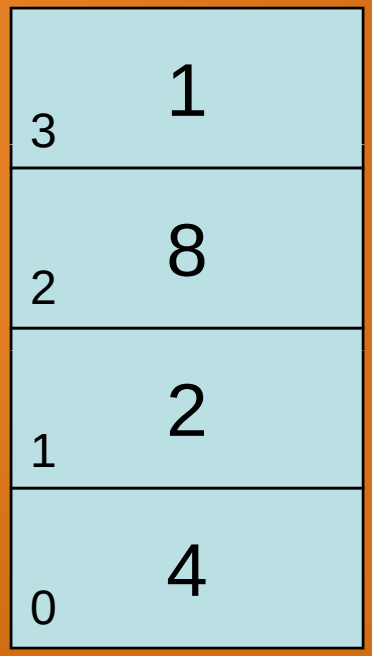
rear

Enqueue : 2

MAX = 4

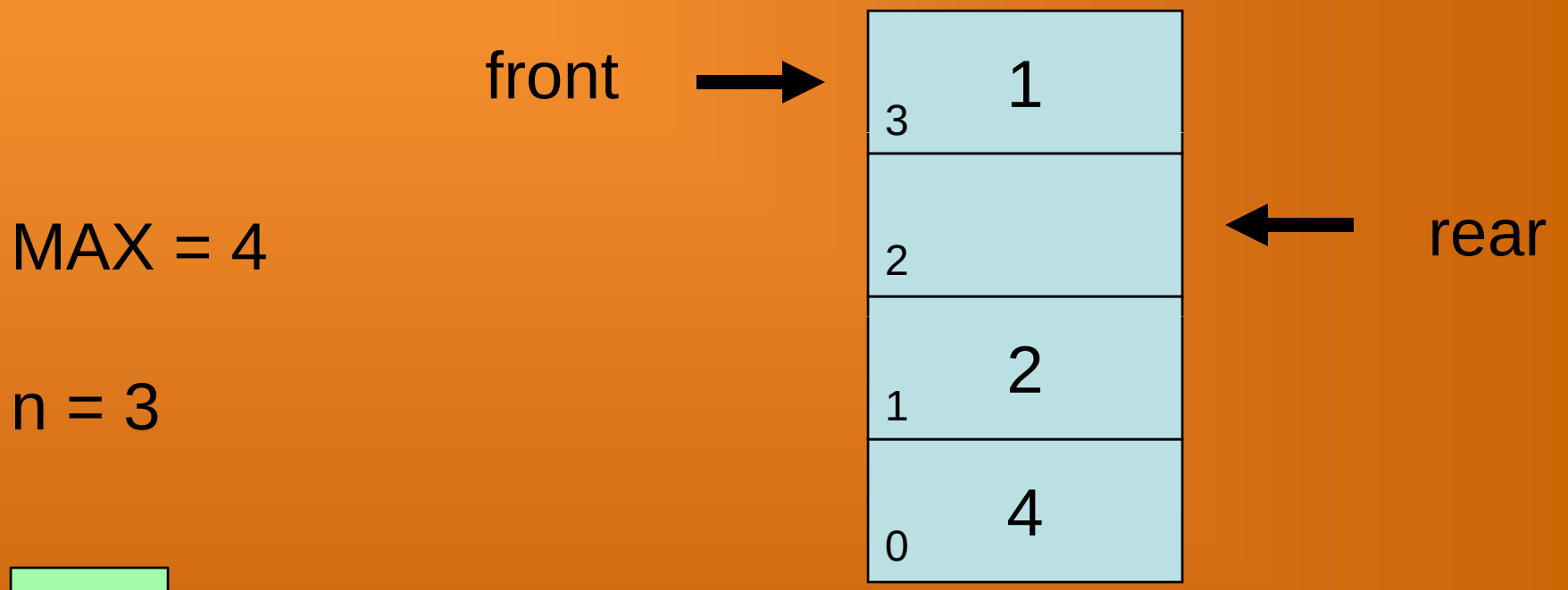
n = 4

front



rear

Dequeue



MAX = 4

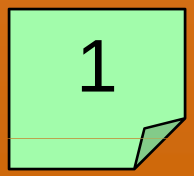
n = 3

8

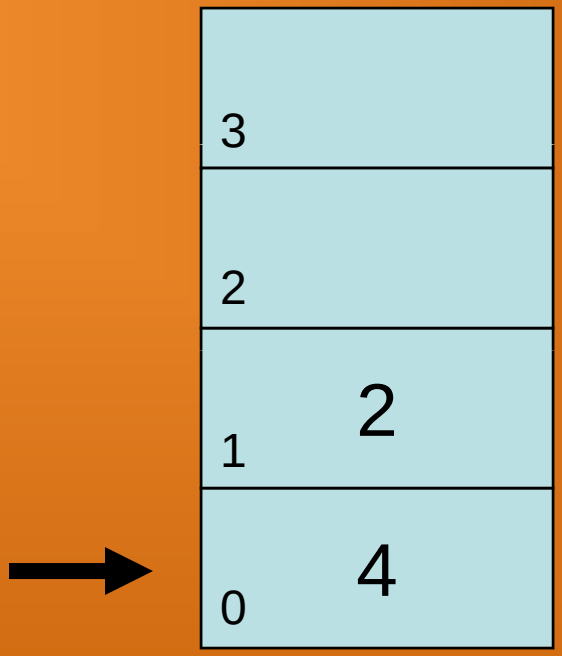
Dequeue

MAX = 4

n = 2



front

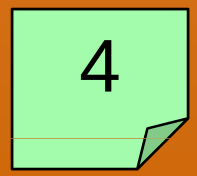


rear

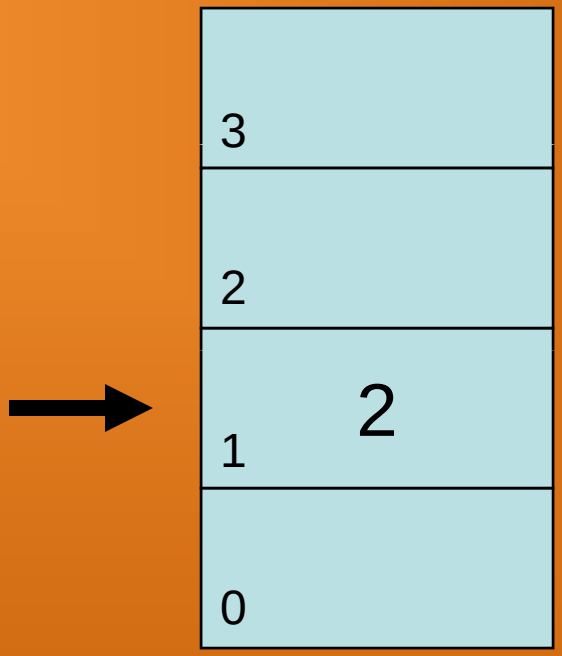
Dequeue

MAX = 4

n = 1



front

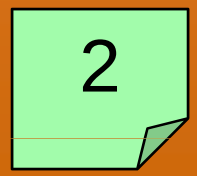


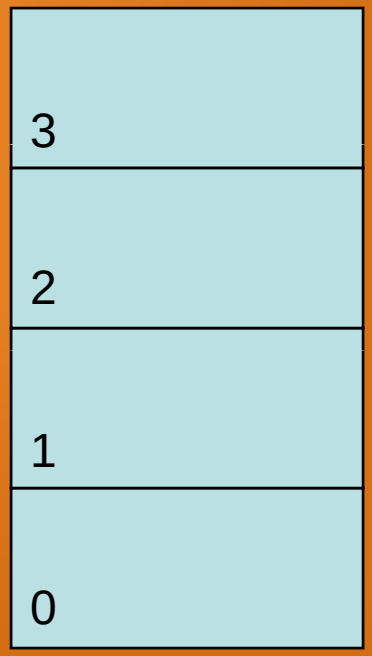
rear

Dequeue

MAX = 4

n = 0





front → 0 ← rear

Inisialisasi

```
void Inisialisasi(struct Queue *q)
{
  q->Front = q->Rear = 0;
  q->Count = 0;
}
```

Full

```
int Full(struct Queue *q)
{
  return(q->Count == MAXQUEUE);
}
```

Empty

```
int Empty(struct Queue *q)
{
  return(q->Count == 0);
}
```

Coding : Enqueue

```
void Enqueue(int x, Queue *Q)
{
  if (Full)
      printf("Tidak dapat memasukkan data! Queue
  Penuh!");
  else {
            Q->Item[Q->Rear] = x;
            Q->Rear = (Q->Rear + 1) % MAXQUEUE;
            ++(Q->Count);
      }
}
```

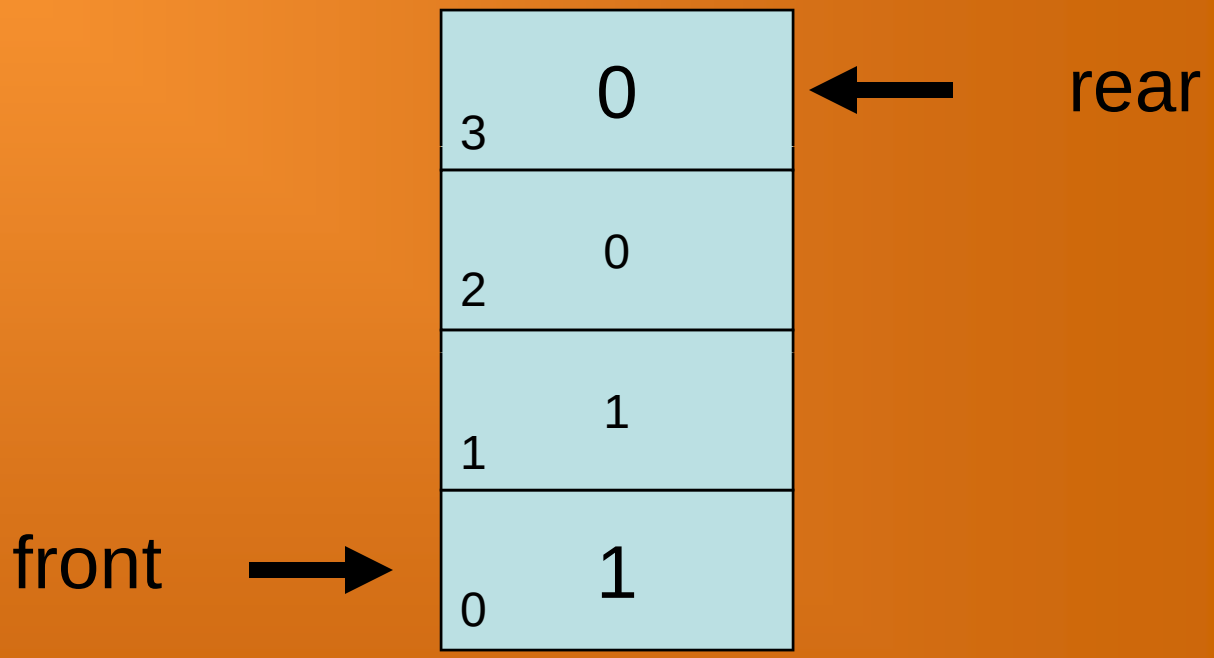
Coding : Dequeue

```
int Dequeue(Queue *Q)
{
  int temp;
  if (Empty)
     printf("Tidak dapat mengambil data! Queue
  Kosong!");
  else {
     temp = Q->Item[Q->Front];
     Q->Front = (Q->Front + 1) % MAXQUEUE;
     --(Q->Count);
     return temp;
  }
}
```

Kondisi awal

MAX = 4

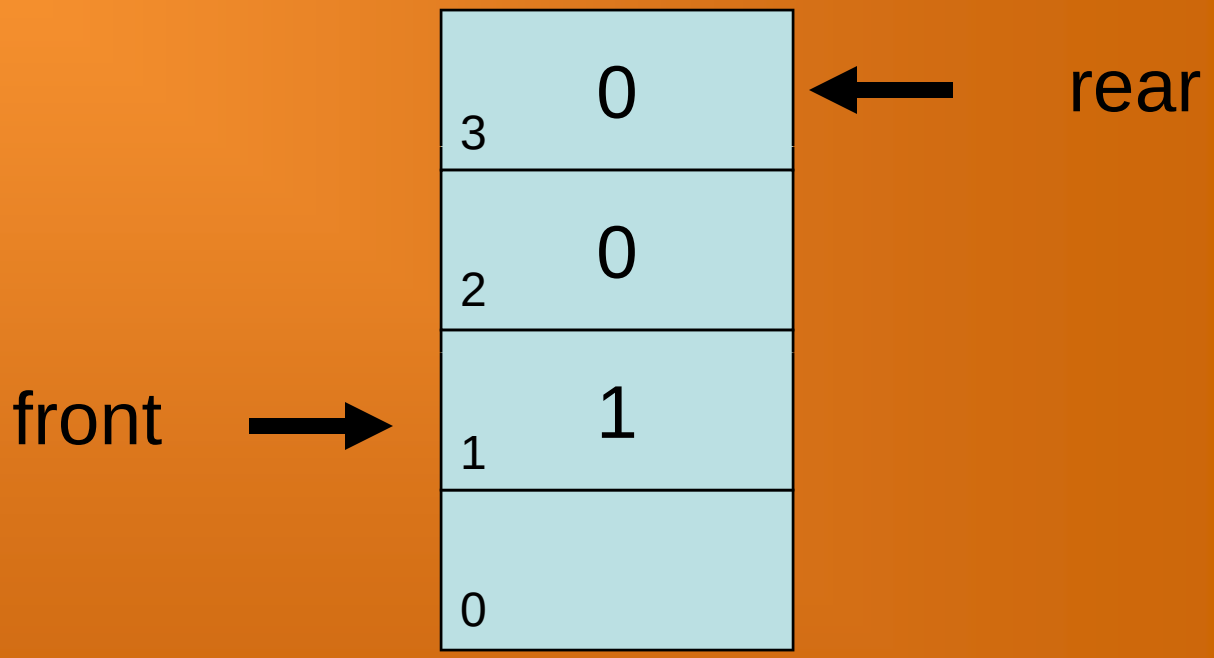
n = 4



Shift 1x(Dequeue)

MAX = 4

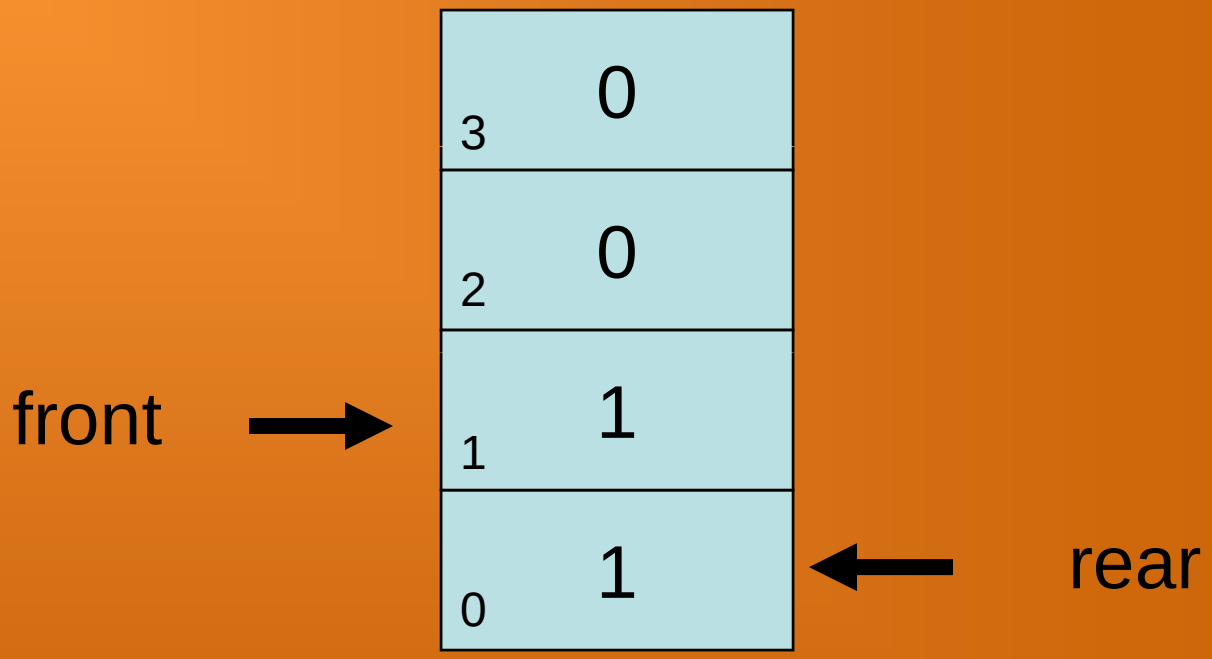
n = 3



Shift 1x(Enqueue)

MAX = 4

n = 4

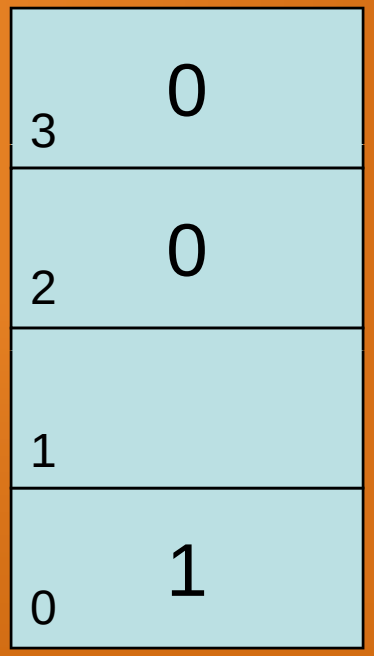


Shift 2x(Dequeue)

MAX = 4

n = 4

front



rear

Shift 2x(Enqueue)

MAX = 4

n = 4