Data Structures
Circular Queue – Array Implementation

Team Emertxe

Algorithm - dequeue

Data Structure –Array Implementation

dequeue(queue,element)

Input Specification:

queue	:	Pointer that contains address of structure variable (queue_t)
element	:	Pointer that contains address of integer variable

Output Specification:

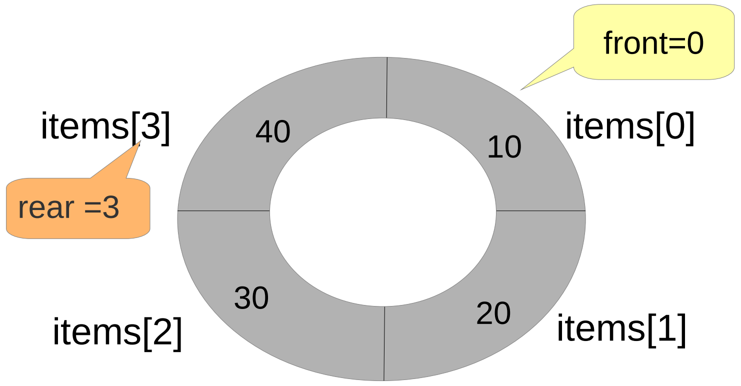
Status : e_true / e_false

Data Structure –Array Implementation

dequeue(queue,element)

capacity = 4

count = 4

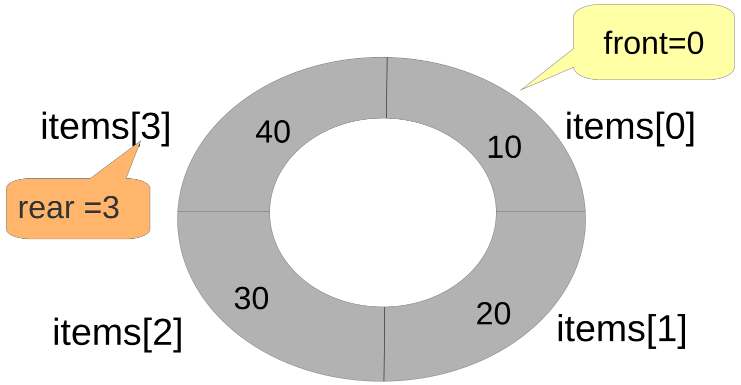


Data Structure –Array Implementation

dequeue(queue,element)

capacity = 4

count = 4



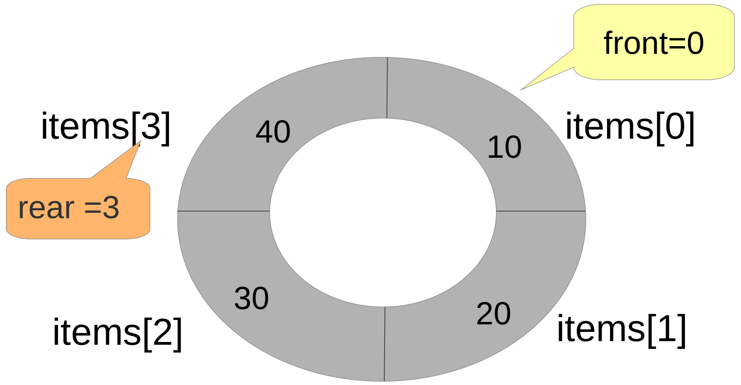
Data Structure –Array Implementation

dequeue(queue,element)

capacity = 4

count = 4

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue⟶ capacity )
--(queue⟶ count)
return e_true
```



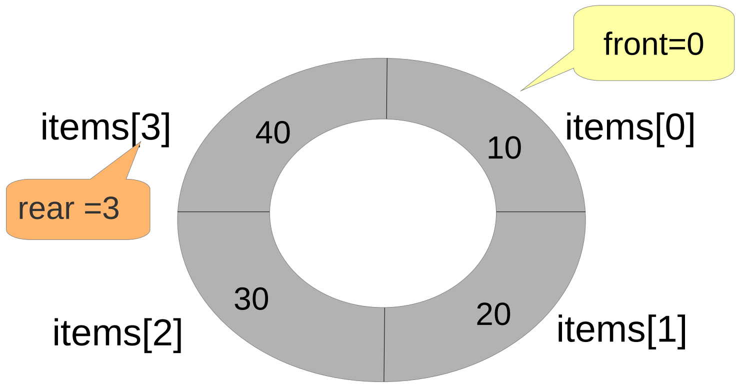
Data Structure –Array Implementation

dequeue(queue,element)

capacity = 4

count = 4

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue⟶ capacity )
--(queue⟶ count)
return e_true
```



Data Structure –Array Implementation

dequeue(queue,element)

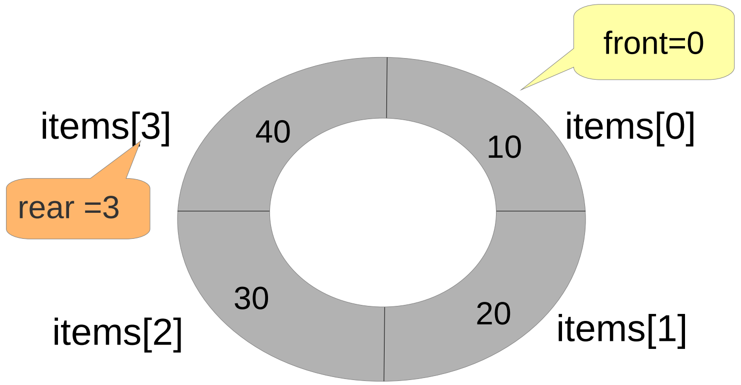
capacity = 4

count = 4

```
if(is_queue_empty(queue))
   return e_false
element = queue → items[queue → front ]
queue → front = (queue → front + 1 ) % (queue→ capacity )
--(queue→ count)
return e_true
```

is_queue_empty(queue)

```
if(queue → count = 0 )
        return e_true
else
    return e_false
```



Data Structure –Array Implementation

dequeue(queue,element)

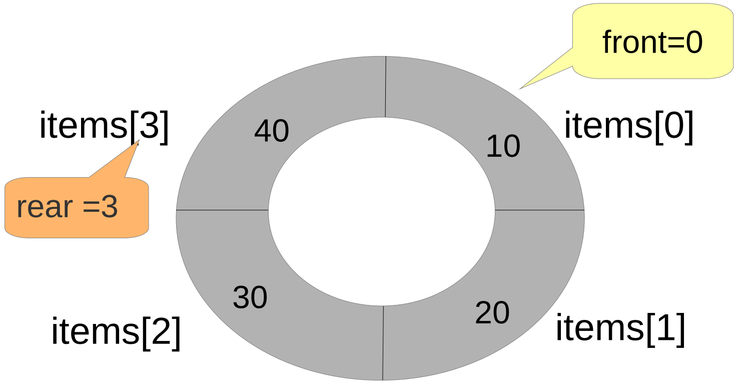
capacity = 4

count = 4

```
if(is_queue_empty(queue))
   return e_false
element = queue → items[queue → front ]
queue → front = (queue → front + 1 ) % (queue → capacity )
--(queue → count)
return e_true
```

is_queue_empty(queue)

```
if(queue → count = 0 )
        return e_true
else
    return e_false
```



Data Structure –Array Implementation

dequeue(queue,element)

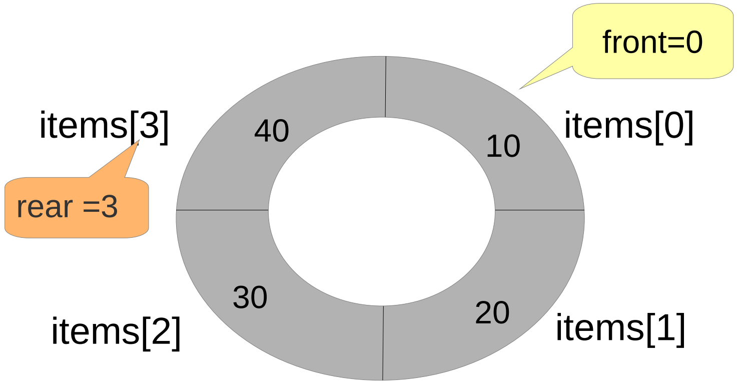
capacity = 4

count = 4

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue⟶ capacity )
--(queue⟶ count)
return e_true
```

is_queue_empty(queue)

```
if(queue ⟶ count = 0 )
        return e_true
else
    return e_false
```



Data Structure –Array Implementation

dequeue(queue,element)

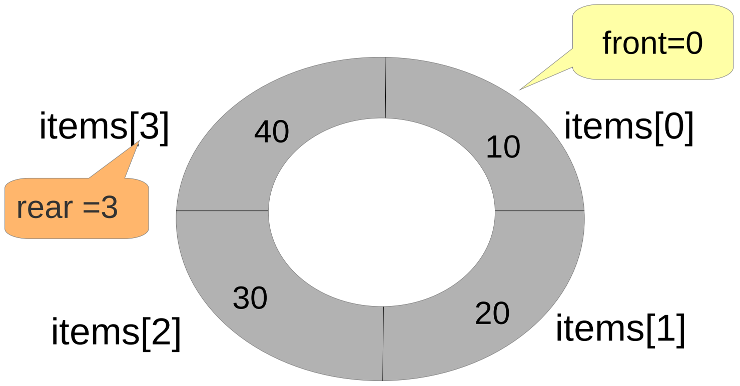
capacity = 4

count = 4

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue⟶ capacity )
--(queue⟶ count)
return e_true
```

is_queue_empty(queue)

```
    if(queue ⟶ count = 0 )
                return e_true
    else
        return e_false
```



Data Structure –Array Implementation

dequeue(queue,element)

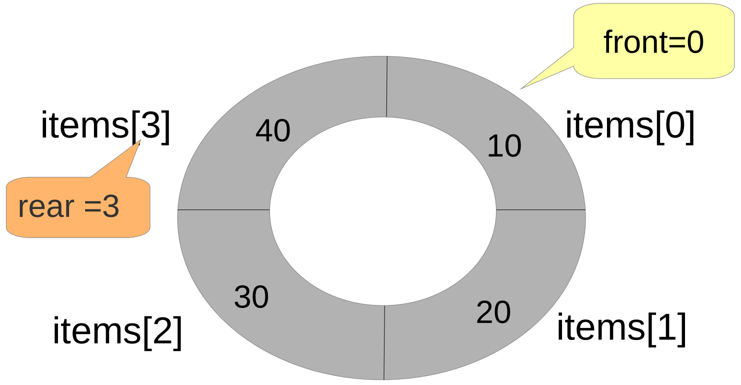
capacity = 4

count = 4

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue⟶ capacity )
--(queue⟶ count)
return e_true
```

is_queue_empty(queue)

```
if(queue ⟶ count = 0 )
        return e_true
else
    return e_false
```



Data Structure –Array Implementation

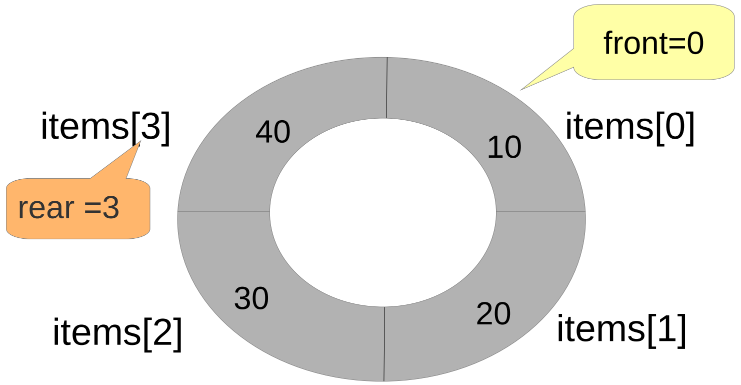
dequeue(queue,element)

capacity = 4

count = 4

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue⟶ capacity )
--(queue⟶ count)
return e_true
```

element = 10



is_queue_empty(queue)

```
if(queue ⟶ count = 0 )
        return e_true
else
   return e_false
```

Data Structure –Array Implementation

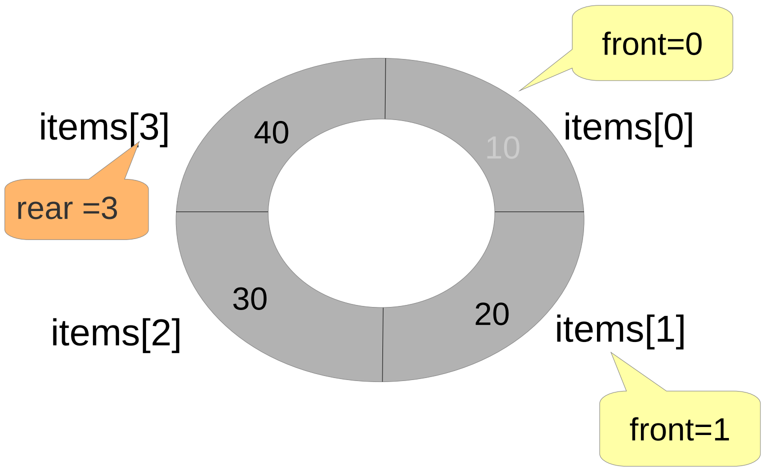
dequeue(queue,element)

capacity = 4

count = 4

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue⟶ capacity )
--(queue⟶ count)
return e_true
```

element = 10



is_queue_empty(queue)

```
if(queue ⟶ count = 0 )
        return e_true
else
    return e_false
```

Data Structure –Array Implementation

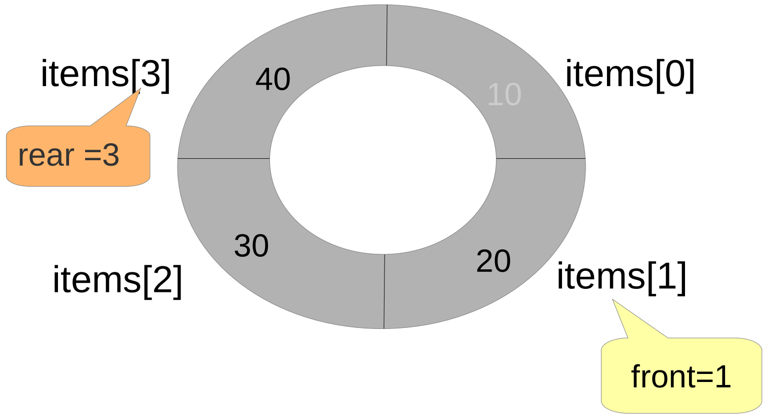
dequeue(queue,element)

capacity = 4

count = 3

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue ⟶ capacity )
--(queue ⟶ count)
return e_true
```

element = 10



is_queue_empty(queue)

```
if(queue ⟶ count = 0 )
        return e_true
else
   return e_false
```

Data Structure –Array Implementation

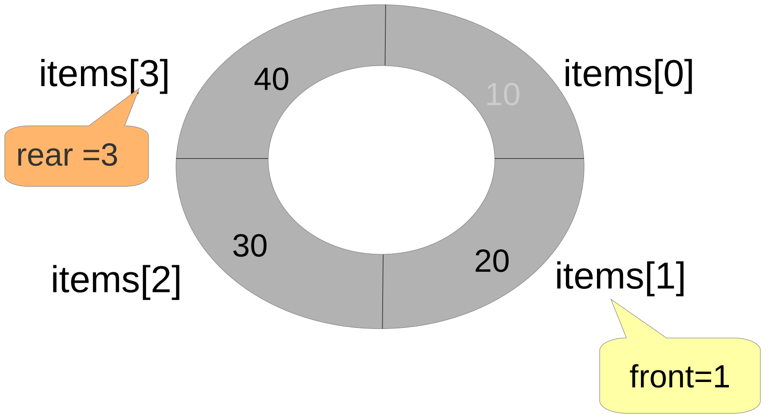
dequeue(queue,element)

capacity = 4

count = 3

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue⟶ capacity )
--(queue⟶ count)
return e_true
```

element = 10



is_queue_empty(queue)

```
if(queue ⟶ count = 0 )
            return e_true
else
    return e_false
```

Data Structure –Array Implementation

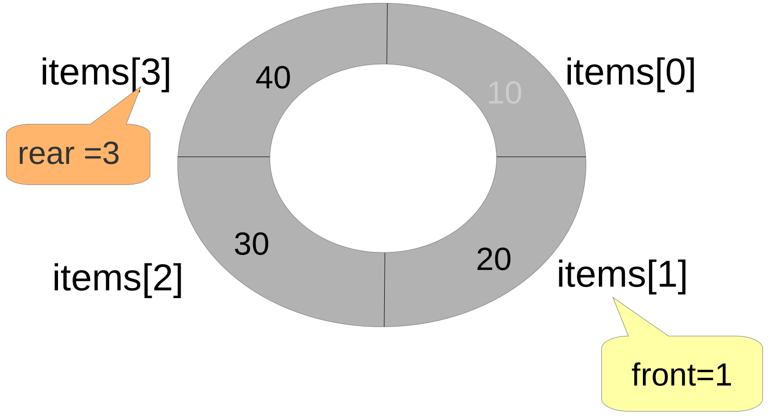
dequeue(queue,element)

capacity = 4

count = 3

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue⟶ capacity )
--(queue⟶ count)
return e_true
```

element = 20



is_queue_empty(queue)

```
if(queue ⟶ count = 0 )
          return e_true
else
   return e_false
```

Data Structure –Array Implementation

dequeue(queue,element)

capacity = 4

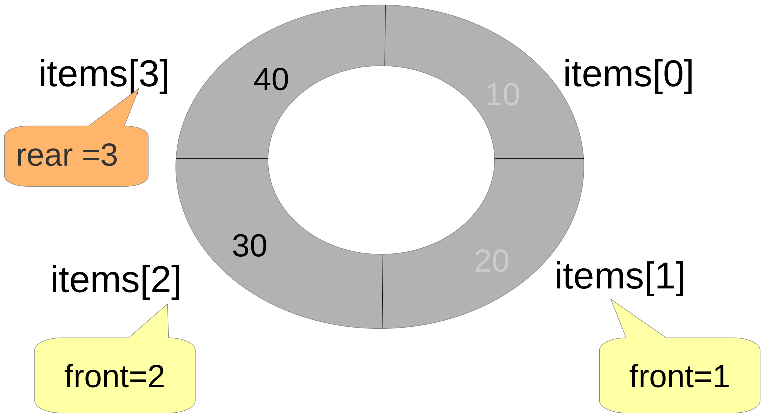
count = 3

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue⟶ capacity )
--(queue⟶ count)
return e_true
```

element = 20

is_queue_empty(queue)

```
if(queue ⟶ count = 0 )
            return e_true
else
    return e_false
```



Data Structure –Array Implementation

dequeue(queue,element)

capacity = 4

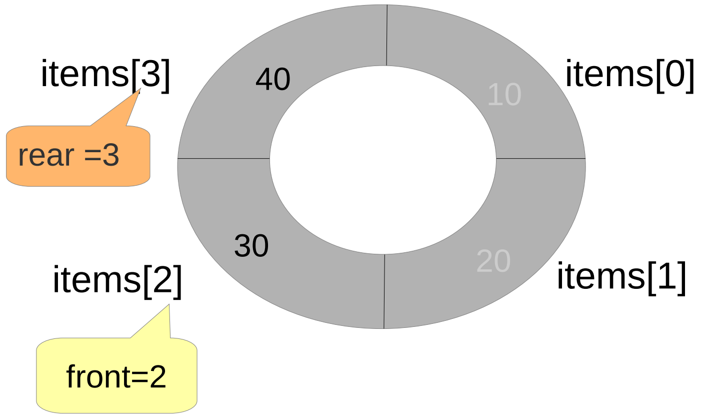
count = 2

```
if(is_queue_empty(queue))
   return e_false
element = queue → items[queue → front ]
queue → front = (queue → front + 1 ) % (queue → capacity )
--(queue → count)
return e_true
```

element = 20

is_queue_empty(queue)

```
if(queue → count = 0 )
                return e_true
else
    return e_false
```



Data Structure –Array Implementation

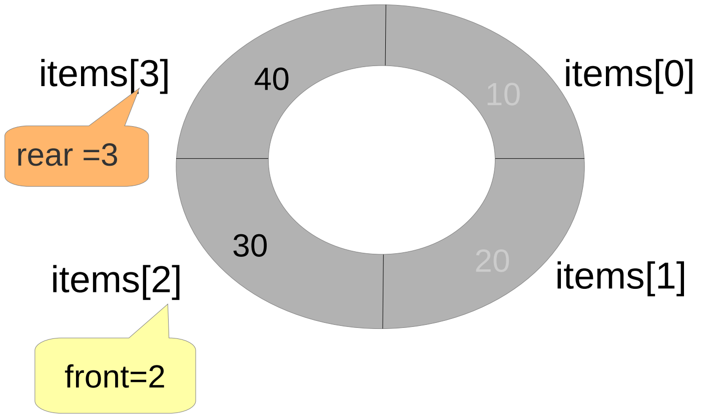
dequeue(queue,element)

capacity = 4

count = 2

```
if(is_queue_empty(queue))
   return e_false
element = queue ⟶ items[queue ⟶ front ]
queue ⟶ front = (queue ⟶ front + 1 ) % (queue⟶ capacity )
--(queue⟶ count)
return e_true
```

element = 20



is_queue_empty(queue)

```
if(queue ⟶ count = 0 )
            return e_true
else
    return e_false
```

Time Complexity = O(1)

Circular Queue – Array Implementation