9주차 1차시 큐와 선형큐

【학습목표】

1. 자료구조 큐에 대한 개념을 스택과 비교하여 설명할 수 있다.
2. 큐의 특징과 연산 방법에 대해 설명할 수 있다.

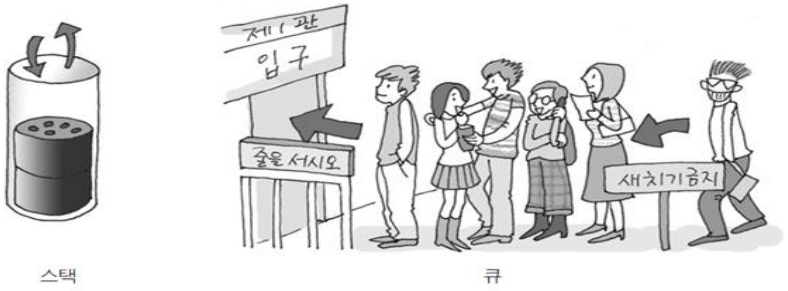
학습내용1 : 큐의 이해

1. 큐(Queue)

* 삽입과 삭제의 위치와 방법이 제한된 유한 순서 리스트
* 큐의 뒤에서는 삽입만 하고, 큐의 앞에서는 삭제만 할 수 있는 구조
- 삽입한 순서대로 원소가 나열되어 가장 먼저 삽입(First-In)한 원소는 맨 앞에 있다가 가장 먼저 삭제(First-Out)

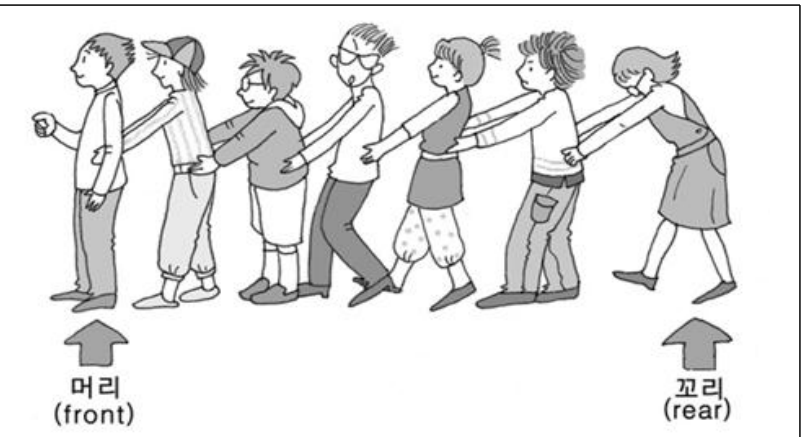
☞ 선입선출 구조(FIFO, First-In-First-Out)

2. 스택과 큐의 비교

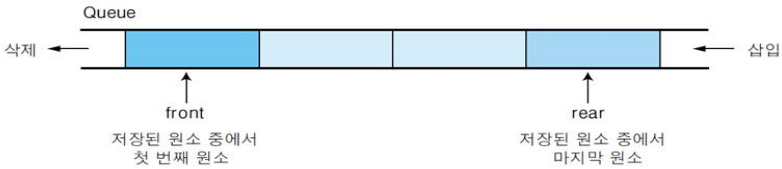


[그림 7-1] 스택과 큐의 구조 예

3. 큐의 구조



[그림 7-2] 꼬리잡기 놀이의 머리와 꼬리



[그림 7-3] 큐의 선입선출 구조

4. 큐의 연산

- 삽입 : enQueue
- 삭제 : deQueue
- 스택과 큐의 연산 비교

[표 7-1] 스택과 큐에서의 삽입과 삭제 연산 비교

항목 / 자료구조	삽입 연산		삭제 연산	
	연산자	삽입 위치	연산자	삭제 위치
스택	push	top	pop	top
큐	enQueue	rear	deQueue	front

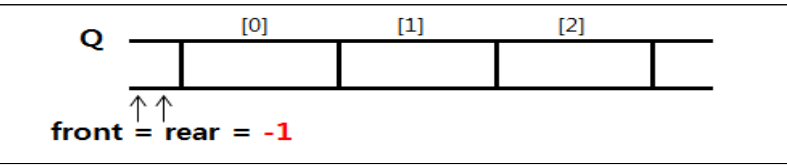
학습내용2 : 추상 자료형 큐

1. 큐에 대한 추상 자료형

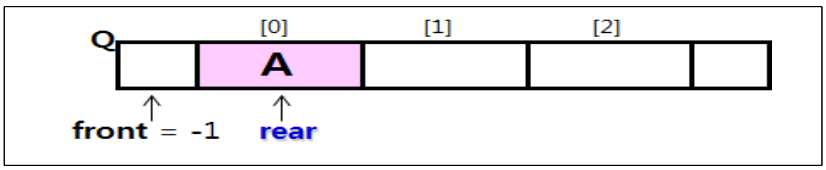
```
ADT Queue
   데이터 : 0개 이상의 원소를 가진 유한 순서 리스트
   연산 :
        Q∈Queue; item∈Element;
        createQueue() ::= create an empty Q;
                    // 공백 큐를 생성하는 연산
        enQueue(Q, item) ::= insert item at the rear of Q;
                    // 큐의 rear에 item(원소)을 삽입하는 연산
        isEmpty(Q) ::= if (Q is empty) then return true
                       else return false;
                    // 큐가 공백인지 아닌지를 확인하는 연산
        deQueue(Q) ::= if (isEmpty(Q)) then return error
                       else { delete and return the front item of Q };
                    // 큐의 front에 있는 item(원소)을 큐에서 삭제하고 반환하는 연산
        delete(Q) ::= if (isEmpty(Q)) then return error
                      else { delete the front item of Q };
                    // 큐의 front에 있는 item(원소)을 삭제하는 연산
        peek(Q) ::= if (isEmpty(Q)) then return error
                    else { return the front item of the Q };
                    // 큐의 front에 있는 item(원소)을 반환하는 연산
End Queue
```

2. 큐의 연산 과정

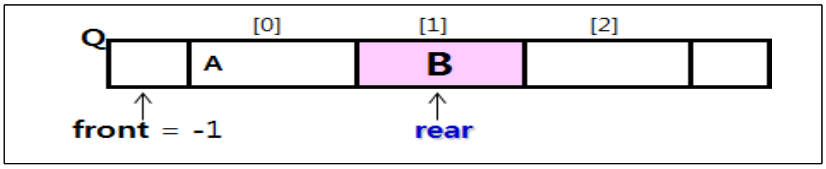
① 공백 큐 생성 : createQueue();



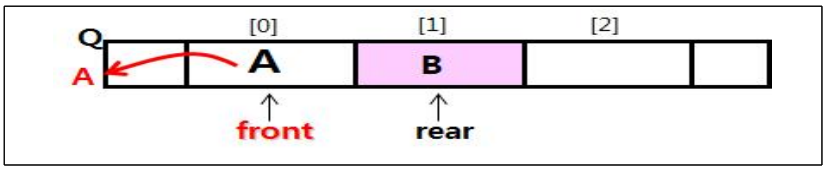
② 원소 A 삽입 : enQueue(Q, A);



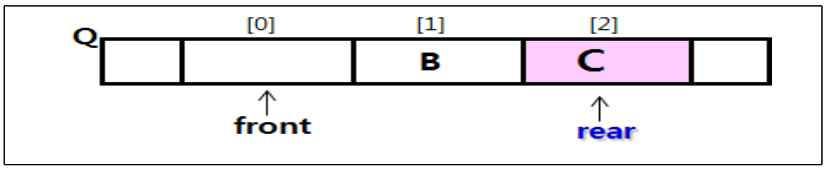
③ 원소 B 삽입 : enQueue(Q, B);



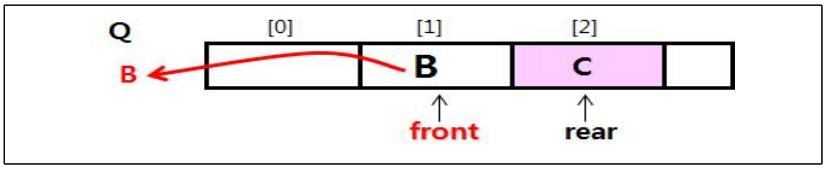
④ 원소 삭제 : deQueue(Q);



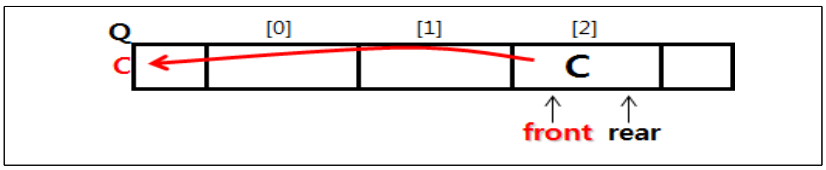
⑤ 원소 C 삽입 : enQueue(Q, C);



⑥ 원소 삭제 : deQueue(Q);



⑦ 원소 삭제 : deQueue(Q);



학습내용3 : 선형큐의 구현

1. 순차 자료구조를 이용한 큐의 구현

* 1차원 배열을 이용한 큐
- 큐의 크기 : 배열의 크기
- 변수 front : 저장된 첫 번째 원소의 인덱스 저장
- 변수 rear : 저장된 마지막 원소의 인덱스 저장

* 상태 표현
- 초기 상태 : front = rear = -1
- 공백 상태 : front = rear
- 포화 상태 : rear = n -1 (n:배열의 크기, n-1: 배열의 마지막 인덱스)

2. 선형 큐 알고리즘

① 초기 공백 큐 생성 알고리즘
- 크기가 n인 1차원 배열 생성
- front와 rear를 -1로 초기화

알고리즘 7-1 초기 공백 큐 생성 알고리즘

```
createQueue()
    Q[n];
    front ← -1;
    rear ← -1;
end createQueue()
```

② 공백 큐 검사 알고리즘과 포화상태 검사 알고리즘
- 공백 상태 : front = rear
- 포화 상태 : rear = n-1(n:배열의 크기, n-1: 배열의 마지막 인덱스)

③ 큐의 삽입 알고리즘

- 마지막 원소의 뒤에 삽입하여야 하므로
    - 마지막 원소의 인덱스를 저장한 rear의 값을 하나 증가시켜 삽입할 자리 준비
    - 그 인덱스에 해당하는 배열원소 Q[rear]에 item을 저장

④ 큐의 삭제 알고리즘

알고리즘 7-2 공백 큐 검사 알고리즘과 포화 상태 검사 알고리즘

```
isEmpty(Q)
    if(front=rear) then return true;
    else return false;
end isEmpty()

isFull(Q)
    if(rear=n-1) then return true;
    else return false;
end isFull()
```

알고리즘 7-3 큐의 삽입 알고리즘

```
enQueue(Q, item)
    if(isFull(Q)) then Queue_Full();
    else {
        rear ← rear+1;          // ❶
        Q[rear] ← item;         // ❷
    }
end enQueue()
```

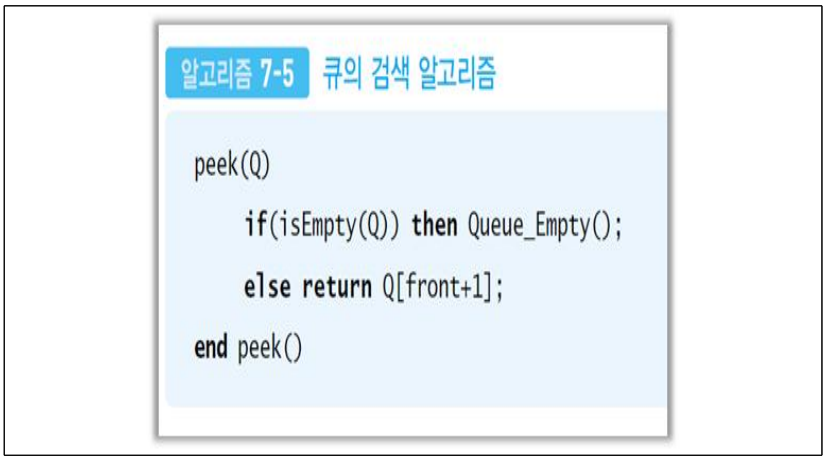
알고리즘 7-4 큐의 삭제 알고리즘

```
deQueue(Q)
    if(isEmpty(Q)) then Queue_Empty();
    else {
        front ← front+1;        // ❶
        return Q[front];        // ❷
    }
end deQueue()

delete(Q)
    if(isEmpty(Q)) then Queue_Empty();
    else front ← front+1;
end delete()
```

- 가장 앞에 있는 원소를 삭제해야 하므로
    - front의 위치를 한자리 뒤로 이동하여 큐에 남아있는 첫 번째 원소의 위치로 이동하여 삭제할 자리 준비
    - 그 자리의 원소를 삭제하여 반환

⑤ 큐의 검색 알고리즘



- 가장 앞에 있는 원소를 검색하여 반환하는 연산
  - 현재 front의 한자리 뒤(front+1)에 있는 원소, 즉, 큐에 있는 첫 번째 원소를 반환

【학습정리】

1. 큐는 리스트의 한쪽 끝(rear)에서 삽입 작업이 이루어지고 반대쪽 끝(front)에서는 삭제 작업이 이루어져서 삽입된 순서대로 삭제되는 선입선출(First-In-First-Out) 구조로 운영되는 리스트이다.

2. 큐를 순차 자료구조를 사용하여 구현하는 방법은 1차원 배열을 이용하는 것이다 .