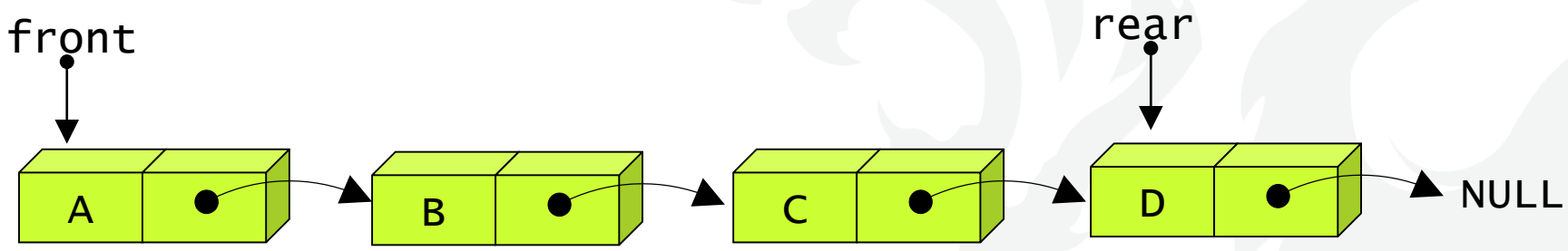
CSE2010 자료구조론

Week 5: Queue 2

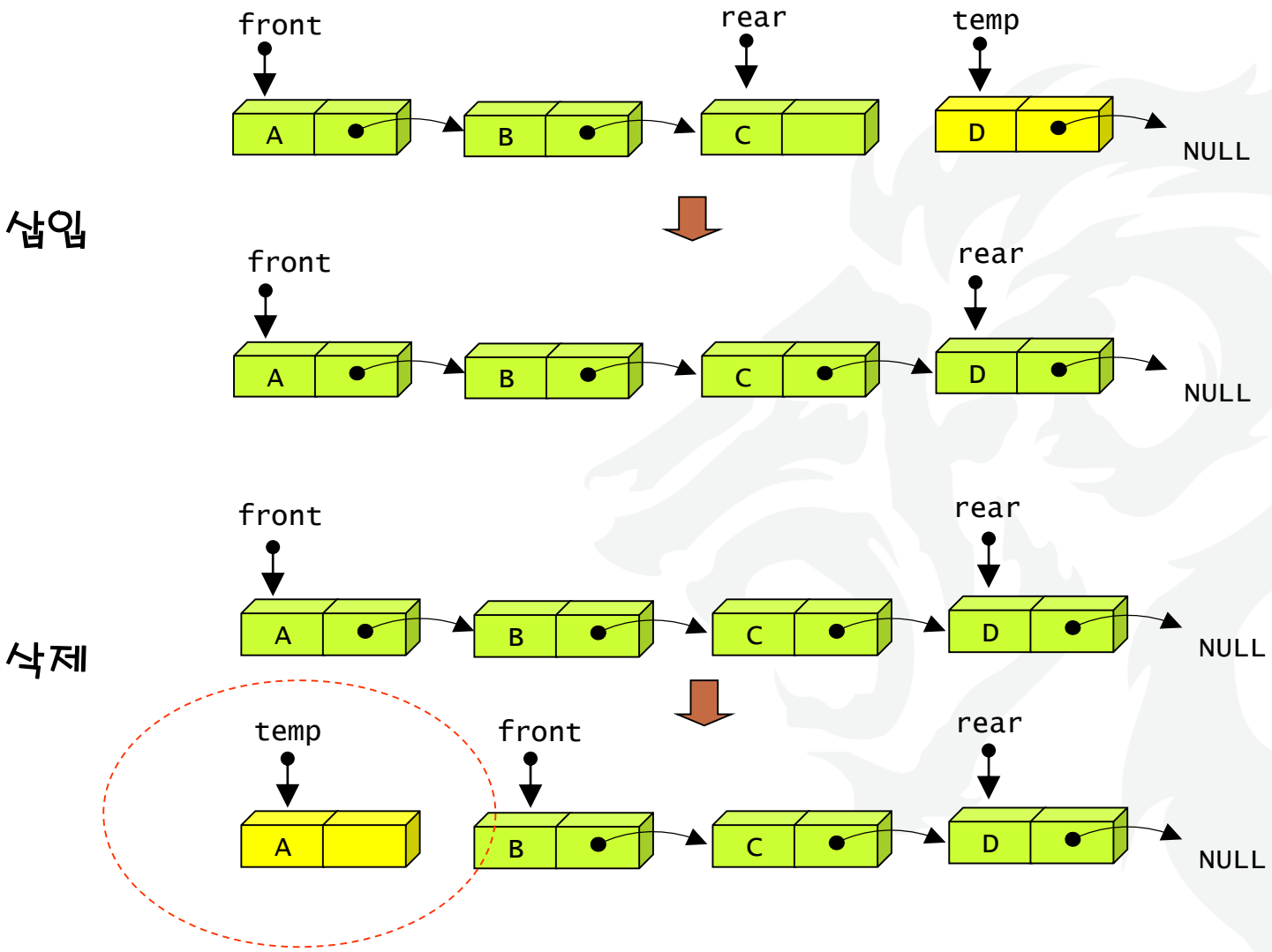
ICT융합학부 한진영

연결리스트로 구현된 큐

- 연결된 큐(linked queue): 연결리스트로 구현된 큐
  - front 포인터는 삭제와 관련되며 rear 포인터는 삽입
  - front는 연결 리스트의 맨 앞에 있는 요소를 가리키며, rear 포인터는 맨 뒤에 있는 요소를 가리킴
  - 큐에 요소가 없는 경우에는 front와 rear는 NULL



연결된 큐에서의 삽입과 삭제



연결된 큐 구현(1)

```
typedef int element;       //요소의 타입
typedef struct QueueNode { // 큐의 노드의 타입
          element item;
          struct QueueNode *link;
         } QueueNode;

typedef struct {          //큐 ADT 구현
       QueueNode *front, *rear;
      } QueueType;
```

연결된 큐 구현(2)

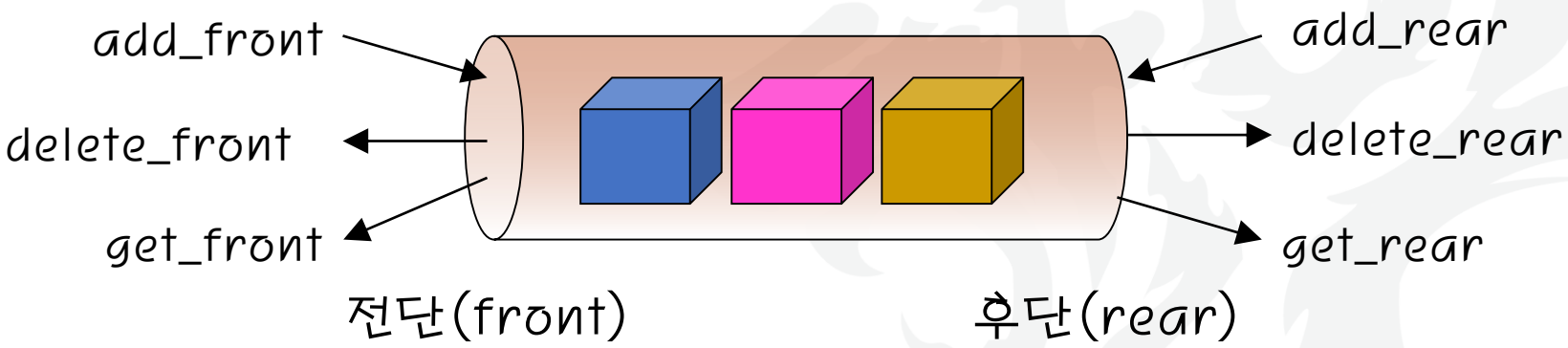
```
// 삽입 함수
void enqueue(QueueType *q, element item)
{
      QueueNode *temp = (QueueNode *)malloc(sizeof(QueueNode));
      if(temp == NULL)
        error("메모리를 할당할 수 없습니다.");
      else {
            temp → item = item;    //데이터 저장
            temp → link = NULL;    // 링크 필드를 NULL로 설정
            if(is_empty(q)){
                  q → front = temp;
                  q → rear = temp;
             }
            else {
                  q → rear → link = temp; //순서 중요!!
                  q → rear = temp;
             }
      }
}
```

연결된 큐 구현(3)

```
// 삭제 함수
void dequeue(QueueType *q)
{
      QueueNode *temp = q → front;
      element item;
      if(is_empty(q))
            error("큐가 비어있습니다.");
      else {
            item = temp → item; // 데이터를 꺼냄
            q → front = q → front → link; //front를 다음 노드를
                                          //가리키도록 한다.
            if(q→front ==NULL)            //공백상태면
                q → rear = NULL;
            free(temp);                   //노드 메모리 해제
            return item;                  //데이터 반환
      }
}
```

덱(deque)

- 덱(deque)은 double-ended queue의 줄임말로 큐의 전단(front)와 후단(rear)에서 모두 삽입과 삭제가 가능한 큐
  - 덱은 스택과 큐의 연산을 모두 가지고 있음



덱(deque) ADT

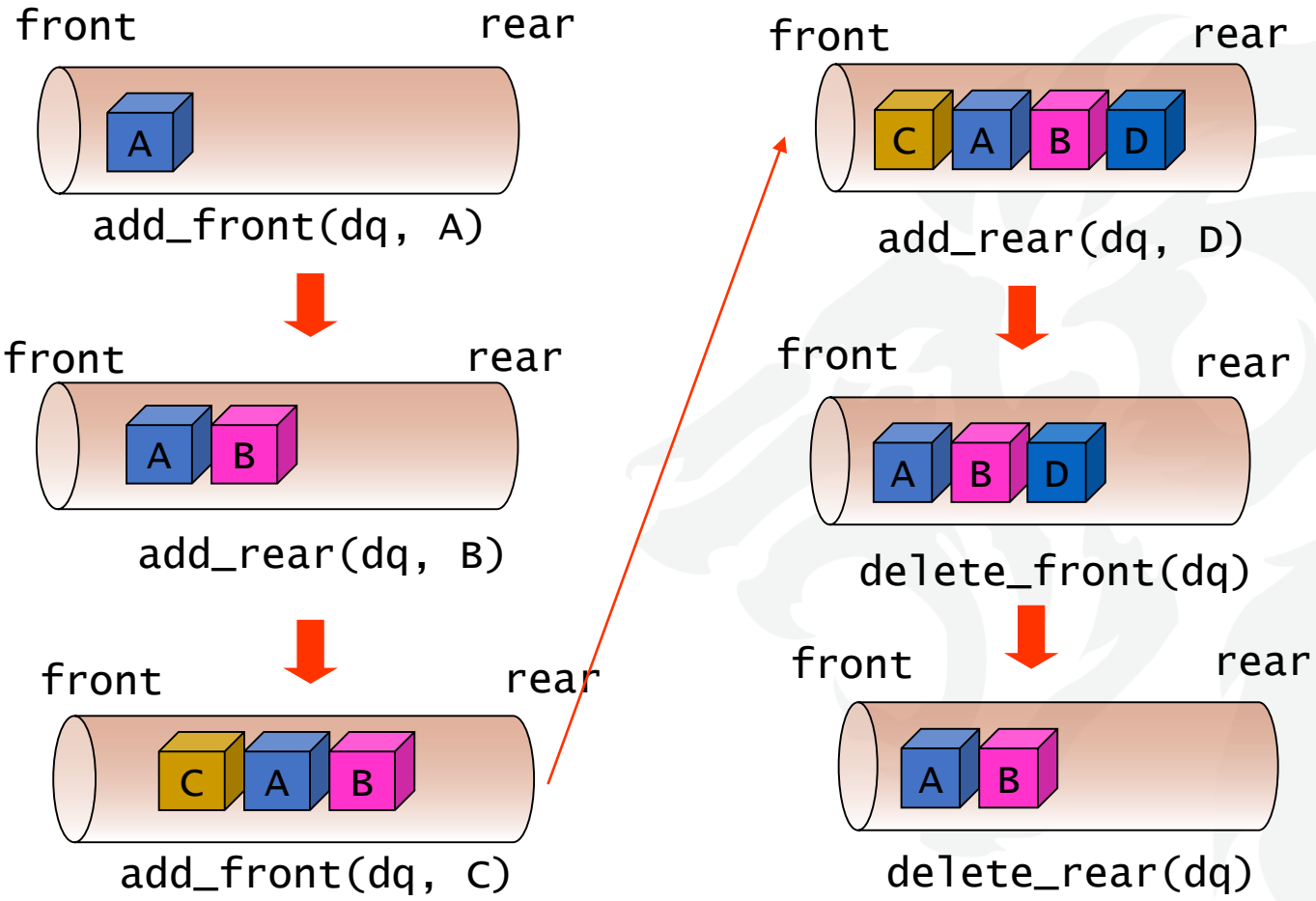
.객체: n개의 element형으로 구성된 요소들의 순서있는 모임

.연산:

- create() ::= 덱을 생성
- init(dq) ::= 덱을 초기화
- is_empty(dq) ::= 덱이 공백상태인지를 검사
- is_full(dq) ::= 덱이 포화상태인지를 검사
- add_front(dq, e) ::= 덱의 앞에 요소를 추가
- add_rear(dq, e) ::= 덱의 뒤에 요소를 추가
- delete_front(dq) ::= 덱의 앞에 있는 요소를 반환한 다음 삭제
- delete_rear(dq) ::= 덱의 뒤에 있는 요소를 반환한 다음 삭제
- get_front(q) ::= 덱의 앞에서 삭제하지 않고 앞에 있는 요소를 반환
- get_rear(q) ::= 덱의 뒤에서 삭제하지 않고 뒤에 있는 요소를 반환

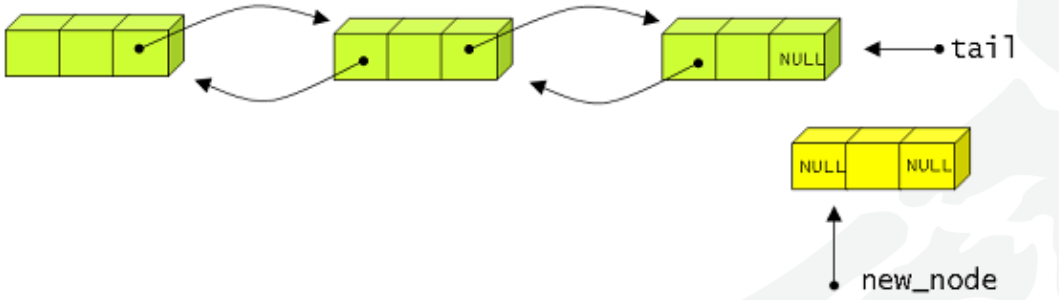
- 덱의 삽입 연산을 push_front와 push_back이라 하기도 함
- 덱의 삭제 연산은 pop_front, pop_back으로 불리기도 함

덱(deque) 연산 예

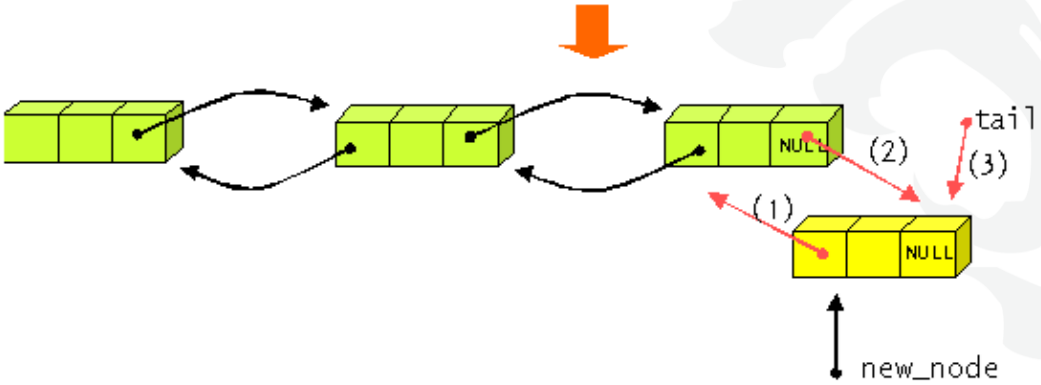

front
rear
A
add_front(dq, A)
front
rear
A
B
add_rear(dq, B)
front
rear
C
A
B
add_front(dq, C)
front
rear
C
A
B
D
add_rear(dq, D)
front
rear
A
B
D
delete_front(dq)
front
rear
A
B
delete_rear(dq)

덱(deque) 삽입 연산

- 연결리스트의 연산과 유사
- 헤드포인터 대신 head와 tail 포인터 사용

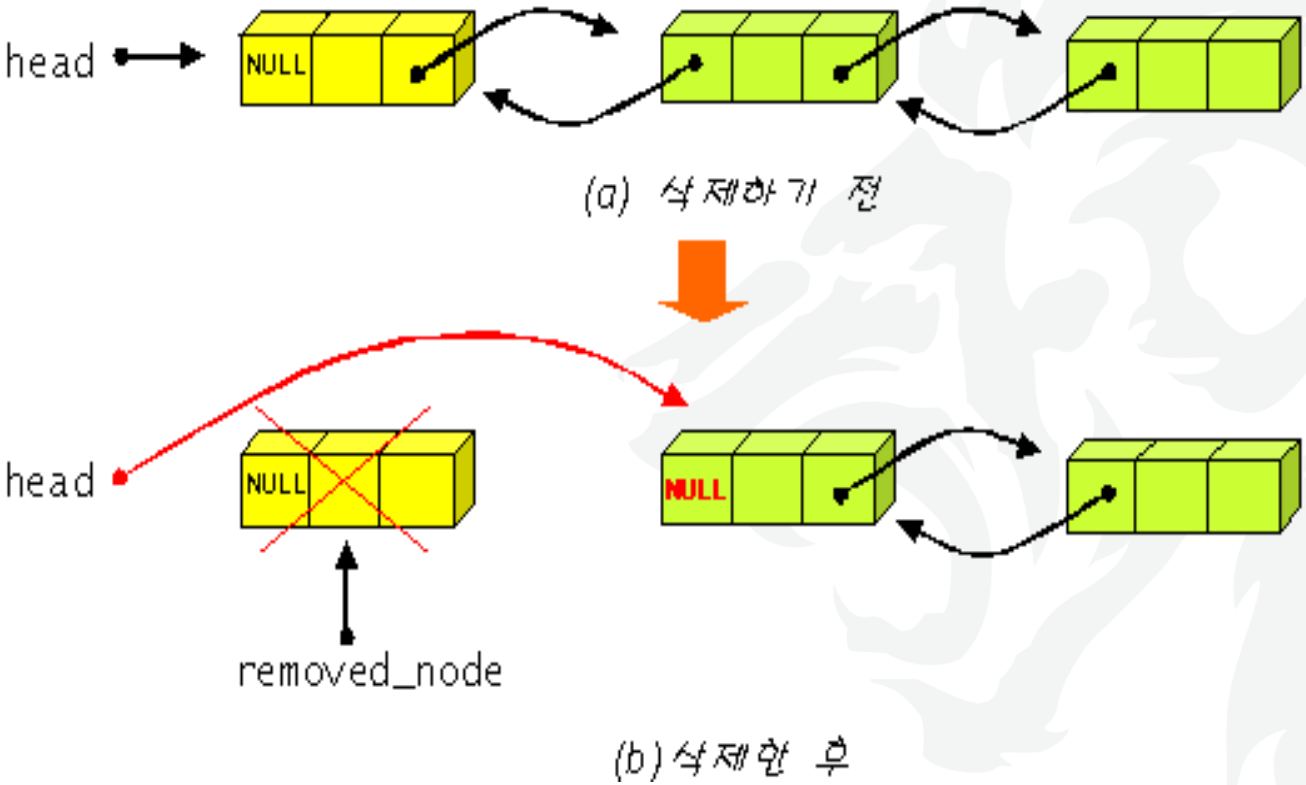


(a) 삽입하기 전



(b) 삽입한 후

덱(deque) 삭제 연산



Week 5: Queue 2