자료구조 Data Structure | 조행래

스택과 큐

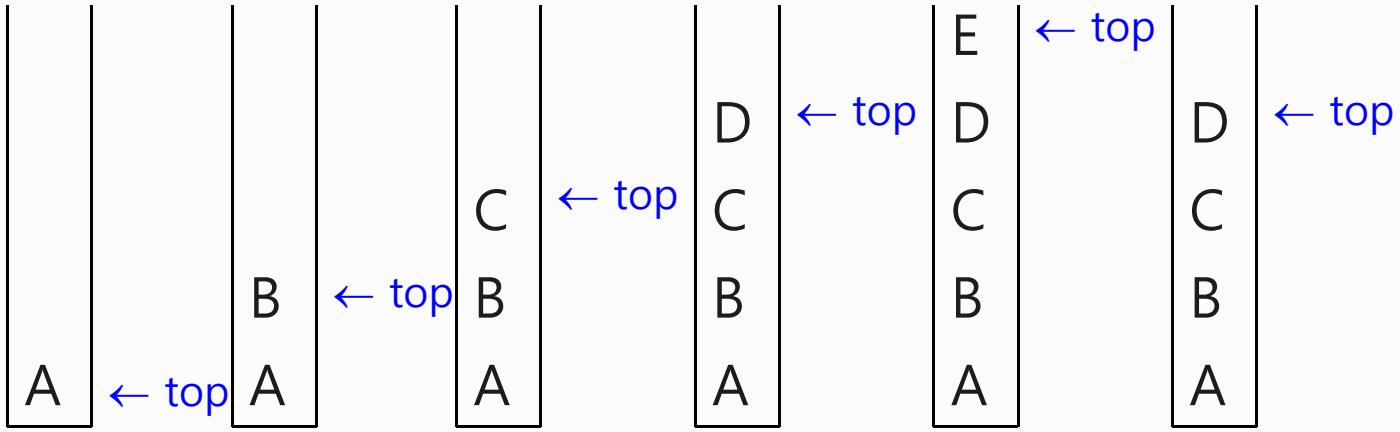
스택과 큐의 개념 및 배열을 이용한 구현

학습 목표

- 스택의 개념을 이해한다.
- 큐의 개념을 이해한다.
- C-언어에서 배열을 이용하여 스택과 큐를 구현할 수 있다.

1. 스택(Stack)의 개념

- **스택의 정의**
  - 삽입과 삭제가 "top"이라 불리는 한쪽 끝 지점에서 발생하는 순서 리스트
  - 후입 선출: Last-In-First-Out (LIFO)



- **스택의 예**
  - 출입문이 하나인 버스

스택 ADT

ADT Stack

객체: 0개 이상의 유한 개 원소로 구성된 순서 리스트

연산: for all stack ∈ Stack, item ∈ element, max_size ∈ 양의 정수

Stack **CreateS(max_size)** ::=
    max_size만큼의 원소를 저장할 수 있는 빈 스택 생성

Boolean **IsFull(stack, max_size)** ::=
    if (스택에 저장된 원소 수 == max_size)
        return TRUE
    else return FALSE

Stack **Push(stack, item)** ::=
    if (IsFull(stack)) stack_full
    else 스택의 top에 item을 저장한 후 **return**

Boolean **IsEmpty(stack)** ::=
    if (stack == CreateS(max_size)) **return** TRUE
    else return FALSE

Element **Pop(stack)** ::=
    if (IsEmpty(stack)) stack_empty
    else 스택 top의 item을 제거해서 반환

2. 배열을 이용한 스택의 구현

```
Stack CreateS(max_stack_size) ::=
        #define MAX_STACK_SIZE 100
        int stack[MAX_STACK_SIZE];        // 배열로 스택 구현
        int top = -1;      // 전역 변수
```

```
Boolean IsEmpty(Stack) ::= top < 0;
```

```
Boolean IsFull(Stack) ::= top >= MAX_STACK_SIZE - 1;
```

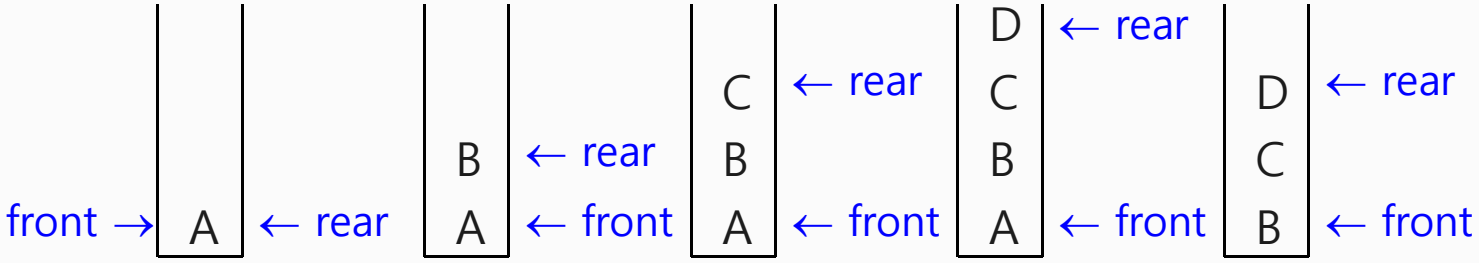
배열을 이용한 스택의 구현 (계속)

```
void push(element item)
{       // 스택에 새로운 항목을 추가
        if (top >= MAX_STACK_SIZE - 1) {
                stack_full();
                return;
        }
        stack[++top] = item;        // top은 -1로 초기화
}
```

```
element pop()
{       // 스택 top의 항목을 return
        if (top == -1)
                return stack_empty();
        return stack[top--];
}
```

3. 큐(Queue)의 개념

- **큐의 정의**
  - 삽입과 삭제가 다른 쪽에서 발생하는 순서 리스트
    - 삽입이 발생하는 위치: **rear**
    - 삭제가 발생하는 위치: **front**
  - 선입 선출: First-In-First-Out (FIFO)



큐 ADT

ADT Queue

객체: 0개 이상의 유한 개 원소로 구성된 순서 리스트

연산: for all queue ∈ Queue, item ∈ element, max_size ∈ 양의 정수

Queue **CreateQ(max_size)** ::=
max_size만큼의 원소를 저장할 수 있는 빈 큐를 생성

Boolean **IsFullQ(queue, max_size)** ::=
if (큐의 원소 수 == max_size)
return TRUE
else return FALSE

Queue **AddQ(queue, item)** ::=
if (IsFullQ(queue, max_size)) **return** queue_full
else 큐의 rear에 item을 삽입하고 큐를 반환

Boolean **IsEmptyQ(queue)** ::=
if (queue == CreateQ(max_size)) **return** TRUE
else return FALSE

Element **DeleteQ(queue)** ::=
if (IsEmptyQ(queue)) **return** queue_empty
else 큐의 front에 있는 item을 제거해서 반환

4. 배열을 이용한 큐의 구현

```
Queue CreateQ(max_queue_size) ::=
   #define MAX_Q_SIZE 100
   int queue[MAX_Q_SIZE];       // 배열로 큐 구현
   int rear = -1, front = -1;   // 전역 변수
```

```
Boolean IsEmptyQ(queue) ::= front == rear;
```

```
Boolean IsFullQ(queue) ::= rear == MAX_Q_SIZE - 1;
```

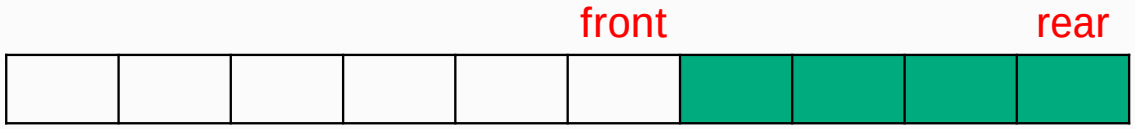
배열을 이용한 큐의 구현 (계속)

```
void addq(element item)
{       // Queue에 새로운 항목을 추가
        if (rear >= MAX_Q_SIZE – 1) {
                queue_full();
                return;
        }
        queue[++rear] = item;
}
```

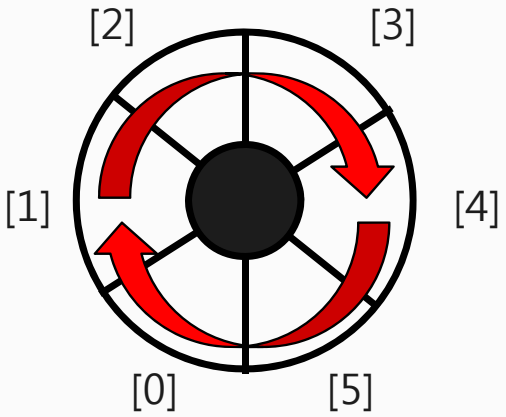
```
element deleteq()
{       // Queue의 항목을 return
        if (front == rear)      return queue_empty();
        return queue[++front];
}
```

5. 원형 큐(Circular Queue)

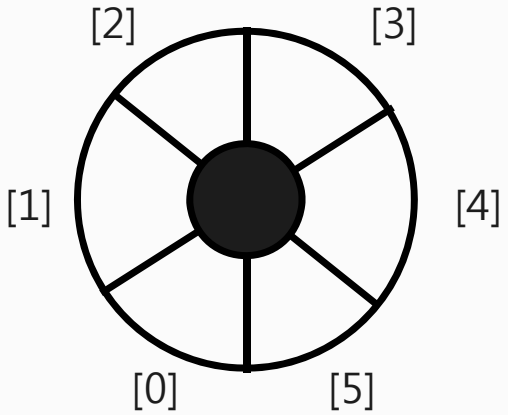
- **배열을 이용하여 큐를 구현할 때 발생하는 문제점**
  - QueueFull의 조건: rear == max_Q_SIZE – 1
  - 문제점: 큐에 저장된 원소의 수 < max_Q_SIZE



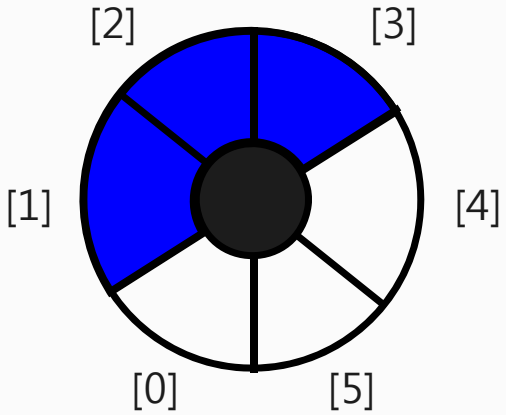
- **원형 큐의 개념**
  - 큐의 처음과 마지막을 연결
  - 나머지(%) 연산자 이용



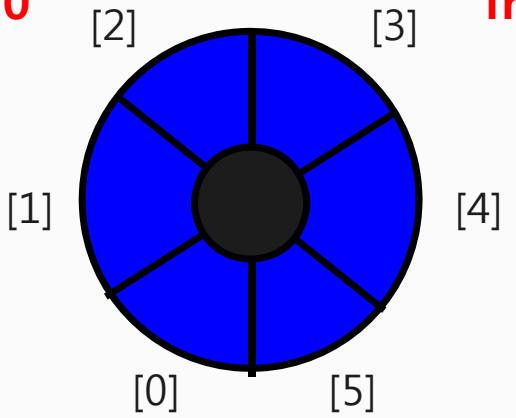
원형 큐의 개념



front = 0, rear = 0



front = 0, rear = 3



front = 0, rear = 0

최대 큐 이용률 =
MAX_Q_SIZE – 1

6. 원형 큐의 구현

```
void addq(element item)
{       // 원형 큐에 새로운 항목을 추가
        rear = (rear + 1) % MAX_Q_SIZE;
        if (rear == front) {
                queue_full(); return;
        }
        queue[rear] = item;
}
```

```
element deleteq()
{       // 원형 큐의 항목을 return
        if (front == rear)
                return queue_empty();
        front = (front + 1) % MAX_Q_SIZE;
        return queue[front];
}
```

요약 정리

- 스택의 개념을 이해
- 큐의 개념을 이해
- C-언어에서 배열을 이용하여 스택과 큐를 구현하는 방법 이해