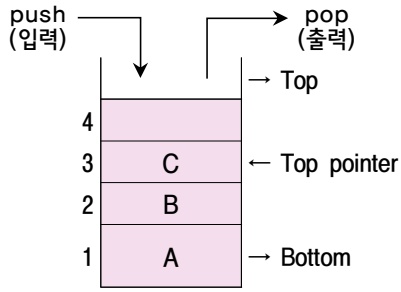
21. 제한구조

1. 스택(stack)



- 스택은 항아리와 같은 구조이다.
- 먼저 들어간 자료가 늦게 출력된다.

- 스택에서는 **입력**을 push
- **출력**을 pop이라 한다.

// 스택 구조 및 특성

① 입출력이 리스트의 한쪽 끝에서만 이루어지는 제한구조(바닥이 막힌 구조)
② LIFO(Last In First Out) 방식으로 작동한다.
③ Top : 입출력이 이루어지는 리스트의 끝
④ Bottom : Top의 반대쪽
⑤ Top Pointer : 입출력되는 자료의 위치를 가리킨다.
⑥ 함수 호출할 때 복귀할 주소 보관, 수식 계산, 트리 구현 등에 사용된다.

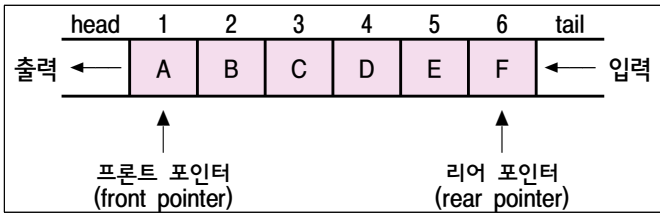
// 스택 알고리즘(스택 크기가 4인 경우)

초기상태(공백상태)	top = 0	

push
(입력)
	If top >= 4 Then
	  Stack_full(overflow 발생)
	Else
	  top++
	  자료입력
	EndIf
		top 포인터 증가 후, 자료입력

pop
(출력)
	If top <= 0 Then
	  Stack_empty(underflow 발생)
	Else
	  자료삭제
	  top--
	EndIf
		자료삭제 후, top 포인터 감소

2. 순차큐(queue)



• 큐는 표를 사려고 줄서 있는 것과 같은 구조로 먼저 입력된 자료가 먼저 출력된다.

• front를 head, rear를 tail이라고도 한다.

// 큐의 구조 및 특성

① 입출력이 서로 **반대쪽**에서 이루어지는 제한구조이다.

② FIFO(First In First Out) 방식 ~ 입력된 순서대로 출력된다.

③ 2개의 포인터가 필요

front pointer	리스트에서 **출력**시킬 자료의 위치를 가리킨다.
rear pointer	리스트에서 **입력**되는 자료의 위치를 가리킨다.

// 순차큐 알고리즘(큐 크기가 6인 경우)

초기상태(공백상태)	front = rear = 0	

입력(insert)
rear++, 자료입력
	If rear>= 6 Then
	  Queue_full(overflow 발생)
	Else
	  rear++
	  자료입력
	EndIf
		rear값을 증가시킨 후 자료 입력

출력(delete)
front++, 자료삭제
	If front = rear Then
	  Queue_empty(underflow 발생)
	Else
	  front++
	  자료삭제
	EndIf
		front값을 증가시킨 후 자료 출력

[Tip] 순차큐의 문제점

• 큐를 단순하게 구현하면 자료가 제거된 빈 장소가 발생한다.

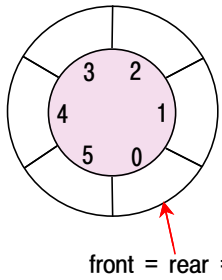
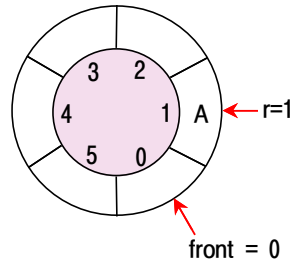
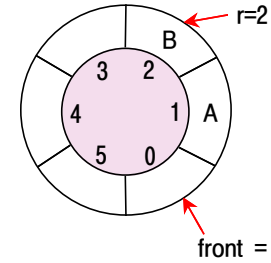
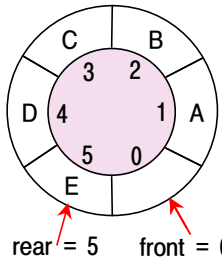
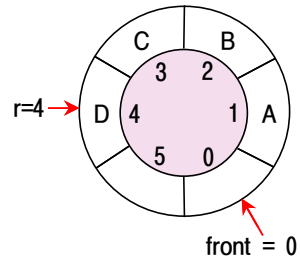
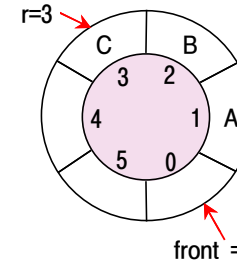
• 즉 큐에 저장된 자료의 양에 관계없이 입력이 큐의 크기보다 커지면 Overflow가 발생한다.

• 이를 해결하기 위한 방법으로 **이동큐**와 **원형큐**가 있다.

3. 원형큐(circular queue)

// 원형큐 구현 – C에서 char q[6];로 배열 정의

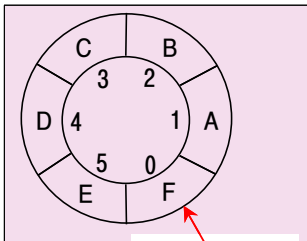
- 입력 : rear값을 증가시킨 후 자료 입력
- 출력 : front값을 증가시킨 후 자료 출력

초기 상태 : front = rear = 0;	입력 A	입력 B
		
입력 E	**입력 D**	**입력 C**
		

↓

↓ 위와 같은 상태에서 자료를 하나 더 입력하면, 다음처럼 rear=0이 되어야 한다.

↓



// 큐의 공백상태와 포화상태를 구별할 수 없게 된다.

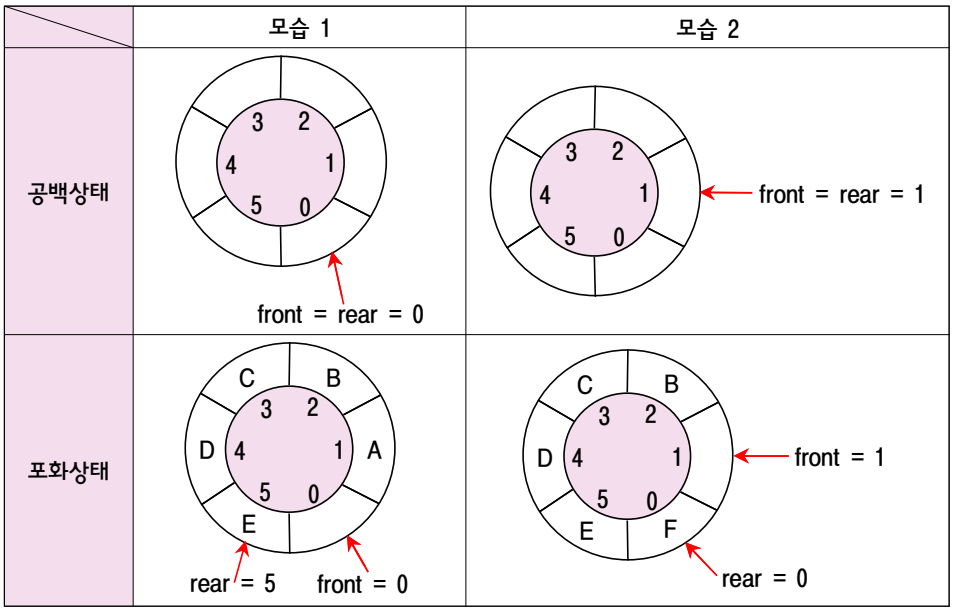
- 공백상태 : front == rear
- 포화상태 : front == rear

↓ **해결 방법**

- 별도의 **flag** 변수를 사용 → **처리속도가 느려 사용하지 않는다.**
- **공백조건의 원형큐** : 크기 n의 배열에 **n-1**개의 자료만은 저장

- **공백조건의 원형큐** : 기억장소 하나를 잃어버리지만 처리속도는 빠르다.(키보드에 이용됨)

// 공백조건의 원형큐



• 공백상태 : front == rear
• 포화상태 : front == rear + 1

// 원형큐에서 front, rear값 증가

• 크기 n인 원형큐에서 포인터 값을 증가시키는 것을 C로 표현하면 다음과 같다.
• 원형큐에서는 mod(%) 연산으로 front, rear값을 결정한다.

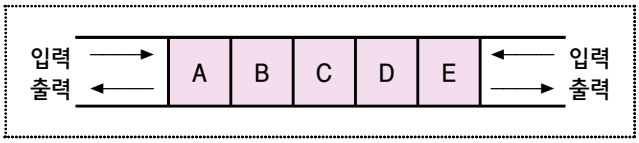
```
if (rear==n-1)
   rear = 0;
else
   rear++;
```

이를 간단히 → 표현하면

```
rear = (rear+1)%n;
```

0 % 6 = 0
1 % 6 = 1
2 % 6 = 2
3 % 6 = 3
4 % 6 = 4
5 % 6 = 5
6 % 6 = 0
7 % 6 = 1
:

4. 데크(deque; double ended queue)



데크는 리스트의 양쪽 끝에서 입출력이 허용되는 자유로운 제한구조이다.

- 데크(deque)는 double ended queue의 약어로 큐의 개념을 확장한 구조이다.
- 리스트의 양쪽 끝에서 모두 입출력이 허용되며, 다음 두 가지처럼 구현하기도 한다.

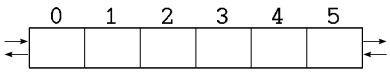
- **입력제한데크(scroll)** : 입력이 한쪽 끝으로만 제한된 구조
- **출력제한데크(shelf)** : 출력이 한쪽 끝으로만 제한된 구조

// 데크의 구현 – 하나의 원형 배열로 구현하는 경우

```
char deque[6];
char *left, *right;
```

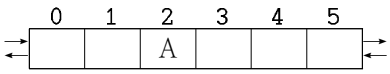
---

① 초기 상태 – 빈 상태



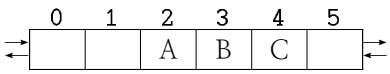
•left = right = null

② 왼쪽에서 A를 입력(배열의 중앙 지점에 적절히 넣는다 – Overflow 최소화를 위해)



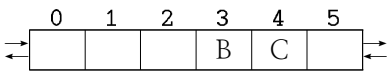
•left = 2
•right = null

③ 오른쪽에서 B, C를 입력



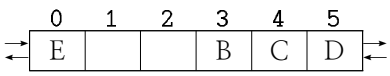
•left = 2
•right = 3, 4

④ 왼쪽에서 출력 – A가 출력됨



•left = 3 → 출력 후 left는 1증해서 3이 된다.
•right = 3, 4

⑤ 오른쪽에서 D, E를 입력



•left = 3
•right = 5, 0 → 원형 구조이므로 5 다음은 0이다.

기출문제 분석

1. 선형(linear) 자료구조로만 짝지어진 것은? [2014년 국회 9급]

① 배열, 리스트, 스택, 큐
② 배열, 트리, 스택, 큐
③ 배열, 리스트, 그래프, 큐
④ 배열, 리스트, 스택, 트리
⑤ 트리, 리스트, 스택, 큐

♤ **자료구조**

- 선형구조 : 배열, 리스트, 스택, 큐, 데크
- 비선형구조 : 트리. 그래프

정답 : ①

2. 스택의 입력으로 4개의 문자 D, C, B, A가 순서대로 들어올 때, 스택 연산 push와 pop에 의해서 출력될 수 없는 결과는? [2021년 국가 9급]

① ABCD ② BDCA ③ CDBA ④ DCBA

♤ **스택 입출력 - 출력 순서로 불가능한 것**

① ABCD : 스택에 모두 입력한 후 차례로 출력한 순서
② BDCA : **출력 순서로 불가능**(D가 C보다 먼저 출력 불가)
③ CDBA : push(D), push(C), pop(C), pop(D), push(B), pop(B), push(A), pop(A)
④ DCBA : push(D), pop(D), push(C), pop(C), push(B), pop(B), push(A), pop(A)

정답 : ②

3. 후입선출(LIFO : Last-In First-Out) 형태로서 자료의 삽입과 삭제가 한쪽 끝에서 이루어지는 자료구조는? [2014년 지방 9급]

① 스택(stack) ② 큐(queue) ③ 트리(tree) ④ 그래프(graph)

♤ **제한구조**

- 스택(stack) : 후입선출(lifo, last-in first-out)
- 큐(queue) : 선입선출(fifo. first in first out)

정답 : ①

4. 다음 중 자료구조 큐(queue)에 대한 설명으로 옳은 것은? [2016년 국회 9급]

① 후입선출(last-in first-out) 특성을 갖는 자료구조이다.
② 데이터를 넣는 위치는 뒤(rear)이고, 데이터를 꺼내는 위치는 앞(front)인 선형리스트이다.
③ 프로그램 실행 시 함수 호출과 복귀를 위한 처리에 유용하다.
④ 인덱스를 이용하여 지정된 임의 위치에서 직접 데이터를 저장하거나 읽기 위한 자료구조이다.
⑤ 하나의 노드가 최대 2개의 자식노드를 가질 수 있는 구조이다.

♤ **자료구조 큐**

① 후입선출(last-in first-out) 특성을 갖는 자료구조이다.**(×)**
→ FIFO(First In First Out) 방식 ~ 입력된 순서대로 출력된다.
③ 프로그램 실행 시 함수 호출과 복귀를 위한 처리에 유용하다.**(×)**
→ 함수 호출과 복귀는 스택이 유용하다.
④ 인덱스를 이용하여 지정된 임의 위치에서 직접 데이터를 저장하거나 읽기 위한 자료구조이다.**(×)**
→ 임의 위치에서 직접 데이터를 저장하거나 읽을 수는 없다.
⑤ 하나의 노드가 최대 2개의 자식노드를 가질 수 있는 구조이다.**(×)**
→ 큐는 자식노드 개념이 없다.

정답 : ②

5. 다음 중 큐가 컴퓨터 시스템에서 이용되는 경우는? [2014년 서울 9급]

① 부프로그램을 처리할 때 레지스터들의 내용 및 복귀주소를 저장할 때
② 순환적 프로그램(recursive program)을 처리할 때
③ 다중 프로그래밍의 운영체제가 대기하고 있는 프로그램들에게 처리기를 할당할 때
④ 그래프를 컴퓨터 내부에 나타낼 때
⑤ 후위 표기방식으로 표현된 수식을 계산할 때

♤ **자료구조 이용 분야**

① 부프로그램을 처리할 때 레지스터들의 내용 및 복귀주소를 저장할 때 → 스택
② 순환적 프로그램(recursive program)을 처리할 때 → 스택
③ 다중 프로그래밍 운영체제가 대기하고 있는 프로그램들에게 처리기를 할당할 때 → 큐
④ 그래프를 컴퓨터 내부에 나타낼 때 → 배열 또는 연결리스트
⑤ 후위 표기방식으로 표현된 수식을 계산할 때 → 스택

정답 : ③

6. <보기>의 각 설명과 일치하는 데이터 구조로 바르게 짝지어진 것은? [2019년 서울 9급]

------<보기>----------
㈎ 먼저 추가된 항목이 먼저 제거된다.
㈏ 먼저 추가된 항목이 나중에 제거된다.
㈐ 항목이 추가된 순서에 상관없이 제거된다.

	㈎	㈏	㈐
①	큐	연결리스트	스택
②	스택	연결리스트	큐
③	스택	큐	연결리스트
④	큐	스택	연결리스트

♤ **데이터 구조**

㈎ 먼저 추가된 항목이 먼저 제거된다. → 큐
㈏ 먼저 추가된 항목이 나중에 제거된다. → 스택
㈐ 항목이 추가된 순서에 상관없이 제거된다. → 연결리스트, 배열 등

정답 : ④

7. 선형 자료구조에 해당하지 않는 것은? [2018년 지방 9급]

① 큐
② 스택
③ 이진트리
④ 단순연결리스트

♤ **자료구조**

선형구조	• **기본 선형구조** : 배열, 연결리스트(단순, 원형, 이중) • 기본 선형구조는 자료의 삽입, 삭제가 임의 위치에서 이루어지는 구조이다. • **제한 선형구조** : 스택(순차 스택, 연결 스택), 큐(순차 큐, 원형 큐, 연결 큐), 데크 • 제한 선형구조는 자료의 삽입, 삭제가 정해진 위치에서 이루어지는 구조이다.
비선형구조	트리(계층구조), 그래프(망구조)

정답 : ③

8. 다음은 정수를 저장할 수 있는 스택을 Java로 구현한 것이다. ㉠과 ㉡에 넣을 문장으로 옳은 것은? [2019년 국회 9급]

```
------------------------------
public class StackInt {
  int size, top;
  int buf[];
  public StackInt(int s) {
      buf = new int[s];
      size = s;
      top = -1;
  }
  public void push(int x) {
      ____㉠____;
  }
  public int pop() {
      ____㉡____;
  }
}
------------------------------
```

	㉠	㉡
①	buf[++top] = x	return buf[--top]
②	buf[top] = x	return buf[top]
③	buf[--top] = x	return buf[top++]
④	buf[++top] = x	return buf[top--]
⑤	buf[top + 1] = x	return buf[top - 1]

♤ 스택

```
public void push(int x)
{
  ㉠ buf[++top] = x;      //top 포인터 증가 후, 자료 x를 입력
}
public int pop()
{
  ㉡ return buf[top--];  //자료 출력 후, top 포인터 감소
}
```

정답 : ④

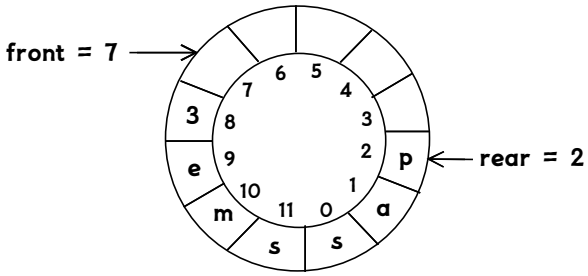
9. 다음과 같은 코드로 동작하는 원형 큐의 front와 rear의 값이 각각 7과 2일 때, 이 원형 큐가 가지고 있는 데이터의 개수는? (단, MAX_QUEUE_SIZE는 12이고, front와 rear의 초깃값은 0이다) [2016년 지방 9급]

```
int queue[MAX_QUEUE_SIZE];
int front, rear;
void enqueue(int item) {
  if( (rear + 1) % MAX_QUEUE_SIZE == front ) {
    printf("queue is full\n");
    return;
  }
  rear = (rear + 1) % MAX_QUEUE_SIZE;
  queue[rear] = item;
}
int dequeue() {
  if( front == rear ) {
    printf("queue is empty\n");
    return -1;
  }
  front = (front + 1) % MAX_QUEUE_SIZE;
  return queue[front];
}
```

① 5 ② 6 ③ 7 ④ 8

♤ 원형 큐(queue)

• 현재, 원형 큐의 상태는 다음과 같다.(데이터 개수 = 7)



• 공백 조건의 원형 큐에
rear 값을 증가시키고, 어떤 자료를 삽입하고
front 값을 증가시키고, 가리키는 곳의 자료를 삭제하는 알고리즘이다.
• 원형 큐를 한바퀴 이상 돈 상태이다.

정답 : ③

10. 다음은 배열로 구현한 스택 자료구조의 push() 연산과 pop() 연산이다. ㉠과 ㉡에 들어갈 코드가 옳게 짝지어진 것은? [2017년 지방 9급]

```
-------------------------------------------------
#define ARRAY_SIZE 10
#define IsFull() ((top == ARRAY_SIZE-1) ? 1: 0)
#define IsEmpty() ((top == -1) ? 1: 0)
int a[ARRAY_SIZE];
int top = -1;
void push(int d)
{
  if( IsFull() )
    printf("STACK FULL\n");
  else
    [        ㉠        ]
}
int pop()
{
  if( IsEmpty() )
    printf("STACK EMPTY\n");
  else
    [        ㉡        ]
}
-------------------------------------------------
```

	㉠	㉡
①	a[++top] = d;	return a[--top];
②	a[++top] = d;	return a[top--];
③	a[--top] = d;	return a[++top];
④	a[top--] = d;	return a[top++];

♤ 스택 자료구조의 push() 연산과 pop() 연산

- push(int d) : top 포인터 증가 후 자료 입력 : a[++top] = d;
- pop() : 자료 출력 후 top 포인터 감소 : return a[top--];

정답 : ②