# 03. 기본 자료구조와 확장형 자료구조2

스택

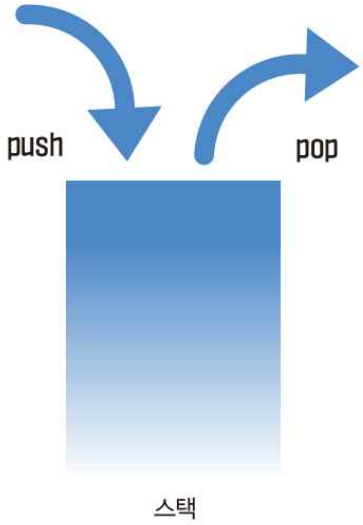
큐

# 스택과 큐

- 스택(Stack)과 큐(Queue)
  - 선형 자료구조에 속해 데이터가 저장될 때 순서를 가지고 저장되도록 하는 구조를 가지므로, 순서에 맞추어 데이터에 접근, 수정, 삭제가 가능

# 스택

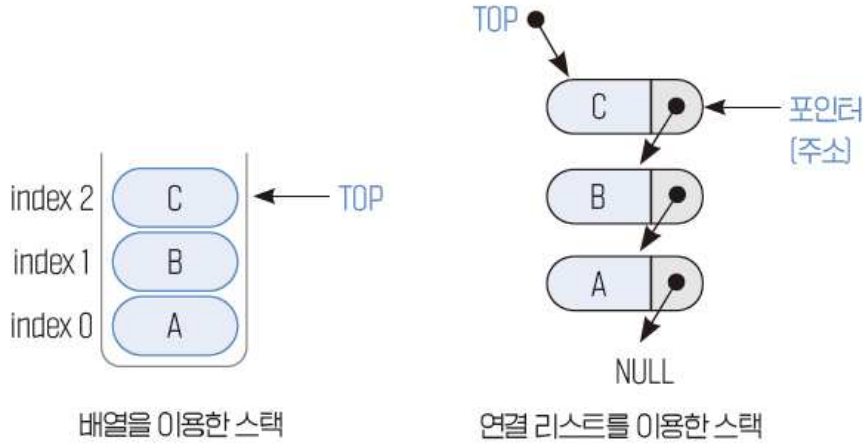
- 스택 자료구조
  - 새로운 데이터 삽입 시 항상 가장 마지막에 저장(push 연산)
  - 데이터 삭제 시 항상 가장 마지막에 있는 데이터 제거(pop 연산)
  - 따라서 데이터 삽입과 삭제가 한 쪽에서만 일어남(top이 가리킴)
  - LIFO(Last In First Out) 구조




스택

# 스택

- 스택의 구현
  - 배열을 이용하여 스택 구현
    - 스택의 크기를 미리 정해 놓아 메모리 효율성이 낮음
    - 데이터 삽입과 삭제 연산이 빠름
  - 연결 리스트를 이용하여 스택 구현
    - 스택의 크기가 동적으로 변하여 메모리 효율성이 높음
    - 데이터 삽입과 삭제 연산이 빠름




배열을 이용한 스택 연결 리스트를 이용한 스택

# 스택을 이용한 문제 해결

- 괄호 짝 맞추기 문제
  - 일반적으로 컴파일러는 프로그램에 작성된 괄호들이 짝이 맞는가를 검사
  - 괄호 짝 맞추기는 스택을 이용하여 좌에서 우로 괄호를 읽어가며 다음과 같이 괄호들을 검사
    - 왼쪽 괄호를 만나면 스택에 push 한다.
    - 오른쪽 괄호를 만나면 스택에서 pop을 수행한다.
    - pop된 왼쪽 괄호와 바로 읽었던 오른쪽 괄호가 다른 종류이면 에러 처리하고, 같은 종류이면 다음 괄호를 읽는다.
    - 스택이 비어 있는데 pop을 수행하면 에러 처리한다.
    - 모든 괄호를 처리한 후
      - 스택이 empty이면, 모든 괄호의 짝이 맞는 것이다.
      - 스택이 empty가 아니면, 짝이 맞지 않는 괄호가 스택에 남아 있으므로 에러 처리한다.

- 괄호 짝 맞추기 문제 예
  - 입력: { ( ) { ( ) } }
  - { : push( { ) : a
  - ( : push( ( ) : b
  - ) : pop() : b와 )를 비교: 짝이 맞음
  - { : push( { ) : c
  - ( : push( ( ) : d
  - ) : pop() : d와 )를 비교: 짝이 맞음
  - } : pop() : c와 }를 비교: 짝이 맞음
  - } : pop() : a와 }를 비교: 짝이 맞음
  - 입력을 다 읽고 스택이 empty이므로 괄호의 짝이 맞음

- 수식의 표현 방법
  - 중위 표기법(infix notation)
    - 연산자를 피연산자들 사이에 두는 표기법
    - A+B
  - 전위 표기법(prefix notation)
    - 연산자를 피연산자들 앞에 두는 표기법
    - +AB
  - 후위 표기법(postfix notation)
    - 연산자를 피연산자들 뒤에 두는 표기법
    - AB+

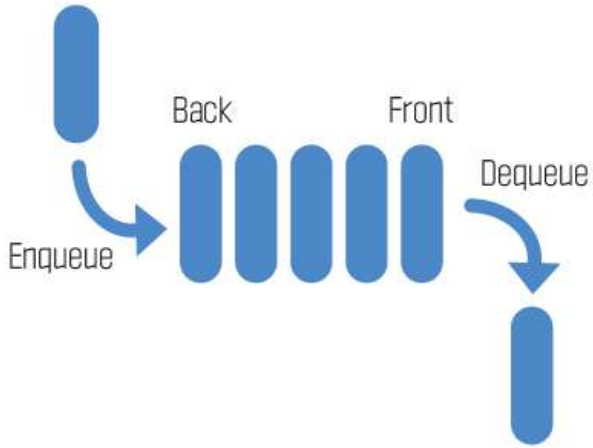
## 후위 표기 수식의 계산 문제

- 컴퓨터는 프로그램에 작성된 중위표기법 수식을 후위표기법 수식으로 바꾸어 계산
- 후위표기법 수식을 계산하기 위해 스택 자료구조를 이용
- 후위 표기법 수식 계산 알고리즘
  - 입력된 수식을 좌에서 우로 문자를 한 개씩 읽는다.
  - 피연산자를 만나면 스택에 push 한다.
  - 연산자를 만나면 스택에서 pop을 2회 수행한다.
    - 먼저 pop된 피연산자가 A이고, 나중에 pop된 피연산자가 B라면, (B op A)를 수행하여 그 결과 값을 다시 스택에 push 한다.
  - 모든 입력을 읽었을 때, 스택을 pop하여 계산 결과를 얻는다.

- 후위 표기 수식의 계산 문제 예
  - 입력: 8 3 2 + 1 - /
  - 8 : push( 8 )
  - 3 : push( 3 )
  - 2 : push( 2 )
  - + : pop(): 2, pop(): 3, push( 3 + 2 )
  - 1 : push( 1 )
  - - : pop(): 1, pop(): 5, push( 5 - 1 )
  - / : pop(): 4, pop(): 8, push( 8 / 4 )
  - 모든 입력을 읽었으므로 pop(): 2 (연산 결과)

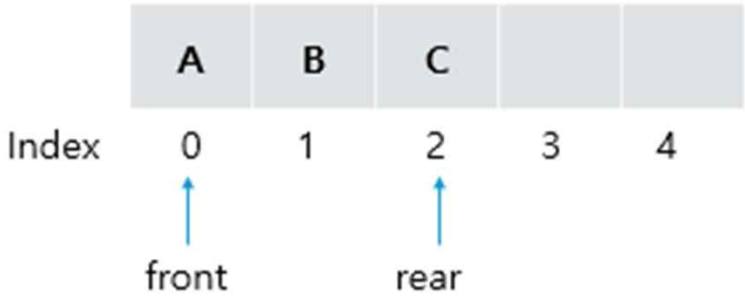
## 큐 자료구조

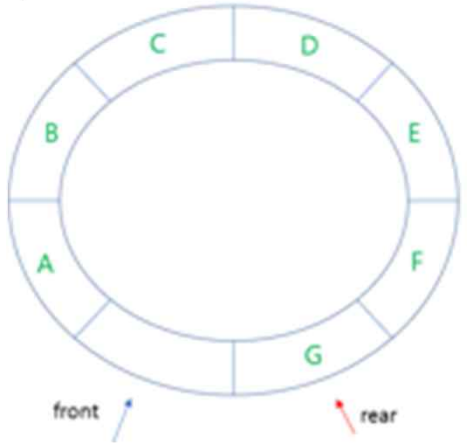
- 새로운 데이터 삽입 시 항상 가장 마지막에 저장
  - add 연산 또는 enqueue
- 데이터 삭제 시 항상 가장 처음에 들어간 데이터 제거
  - remove 연산 또는 dequeue
- 따라서 데이터 삽입과 삭제가 서로 다른 쪽 끝에서만 일어남
  - 삽입 위치는 rear, 삭제 위치는 front가 가리킴
- FIFO(First In First Out) 구조




큐

- 큐의 구현
  - 배열을 이용하여 큐 구현
    - 큐의 크기를 미리 정해 놓아 메모리 효율성이 낮음
    - 데이터 삭제 연산 시 빈 공간 발생 또는 데이터의 이동이 필요
      - 이를 해결하기 위해 원형(순환) 큐를 구현




  - 연결 리스트를 이용하여 큐 구현
    - 스택의 크기가 동적으로 변하여 메모리 효율성이 높음
    - 데이터 삽입과 삭제 연산이 빠름

- 큐의 응용
  - 우선순위가 같은 작업의 예약
  - 은행 업무
  - 콜센터 고객 대기시간 관리
  - 프로세스 관리

끝