대구대학교 DAEGU UNIVERSITY

일상에서 찾아보는

자료구조의 원리

04주차 순서대로 처리하기 - 큐

1차시

순서대로 처리하는 큐

학습목표

- 일상에서 줄서기의 예와 줄서기가 사용되는 이유를 설명할 수 있다.
- 순서대로 처리하는 선형 큐의 장단점을 설명할 수 있다.

학습내용

- 순서대로 처리하기
- 큐의 동작과 응용
- 큐의 구현
- 선형 큐의 구현

순서대로 처리하기

» 일상 속 순서대로 처리하기

- 일상에서 많은 일들은 순서대로 처리하고 있음
- “먼저 들어온 것이 먼저 나간다”는 규칙을 따름

☑ 놀이공원

- 놀이공원에서 관람차를 타기 위해서 사람들이 먼저 온 순서대로 줄을 섬
- 가장 앞에 있는 사람이 먼저 타고, 그 다음 사람, 또 그 다음 사람 순서대로 탐
- 새로 온 사람은 줄의 맨 뒤에 서게 됨

순서대로 처리하기

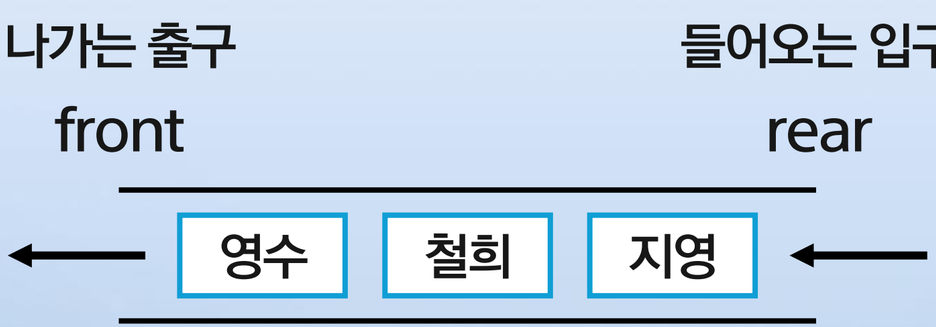
» 큐(Queue)

- 줄을 서서 순서대로 처리하는 자료구조
- 처리 시간과 도착 시간의 차이가 있다면 큐를 활용 할 수 있음

순서대로 처리하기

큐(Queue)

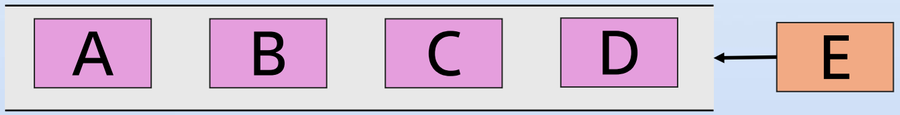
- FIFO (First-In, First-Out)
- 자료를 순서대로 사용하기 위해 모아두는 일시적인 저장 용도의 기능



큐의 동작과 응용

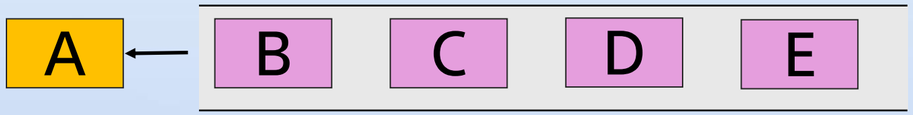
» 큐의 동작

- 들어가기(Enqueue)
  - 놀이공원에서 새로운 사람이 도착하면 맨 뒤에 줄을 섬
  - 새로운 자료는 맨 뒤에 들어옴
  - E가 줄에 들어옴
  - 줄:



큐의 동작과 응용

» 큐의 동작

- 나가기(Dequeue)
  - 놀이공원에서 맨 앞에 선 사람이 줄에서 먼저 나감
  - 먼저 들어온 순서대로 자료가 처리되어 나감
  - A가 줄에서 나감
  - 줄: 

큐의 동작과 응용

» 큐의 응용

- 큐의 일상 사용 예
  - 식당 대기 줄: 자리가 나기를 기다리는 손님들
  - 콜센터의 고객 대기열: 먼저 전화를 한 사람 순서대로 대기
  - 교통 신호 시스템 : 각 방향에서 대기하고 일정 시간 후 통과

큐의 동작과 응용

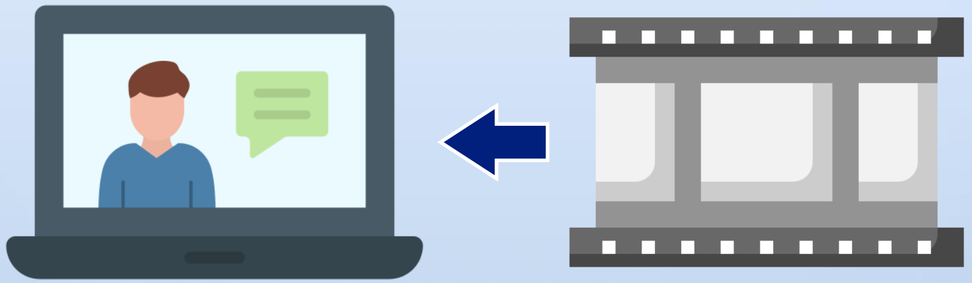
» 큐의 응용

- 버퍼(Buffer)
  - 데이터를 주고받을 때 장치들 사이에 존재하는 속도의 차이
  - 임시로 자료를 모아둠
  - 먼저 들어온 자료를 먼저 처리
  - 예 : 프린터와 컴퓨터 사이의 작업 큐

큐의 동작과 응용

» 큐의 응용

- 비디오 스트리밍에서의 버퍼
  - 일정 시간동안 비디오프레임을 모아서 재생
  - 프레임을 모아주는 저장 장소로 '큐' 사용



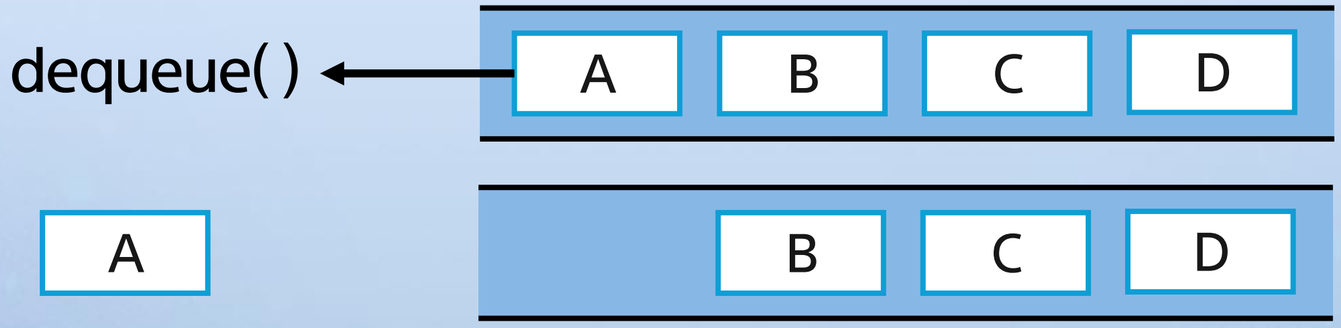
큐의 동작과 응용

» 큐의 동작

- 자료를 삽입, 삭제

삽입

삭제



큐의 구현

» 선형 큐

- 리스트를 사용하여 큐의 동작 원리를 구현
  - enqueue(N)
  - dequeue()

append(N)

pop(0)

큐의 구현

» 큐의 추상 자료형(Queue ADT)

- 자료 저장과 동작 함수들의 모임

enqueue(X)	원소 X를 큐의 맨 뒤에 추가
dequeue()	큐의 맨 앞에 있는 원소를 꺼내 반환
isEmpty()	큐가 비어 있으면 True, 아니면 False를 반환
size()	큐에 저장된 원소들의 개수를 반환
clear()	큐를 공백 상태로 만듦

선형 큐의 구현

```
class LinearQueue :
    def __init__(self):
        self.q_list = [] #큐 저장 공간을 빈 리스트로 생성한다.
    def enqueue(self, data): #enqueue는 삽입할 데이터(값)를 필요로 한다.
        self.q_list.append(data)
    def dequeue(self):
        if not self.isEmpty(): #빈 상태가 아니라면
            return self.q_list.pop(0)#가장 처음 삽입된 데이터를 반환하여 삭제한다.
    def isEmpty(self):
         return len(self.q_list) == 0  #q_list의 길이가 0이면 빈 상태를 의미한다.
    def size(self) :
          return len(self.q_list)
    def clear(self) :
          self.q_list = []
    def display(self):
        print("Queue state:", self.q_list)

#선형 큐 테스트
lq = LinearQueue()
lq.enqueue(1)
lq.enqueue(2)
lq.enqueue(3)
lq.display()  # Queue state: [1, 2, 3, None, None]
print(lq.dequeue())  # 1
lq.display()  # Queue state: [None, 2, 3, None, None]
```

큐 클래스 정의

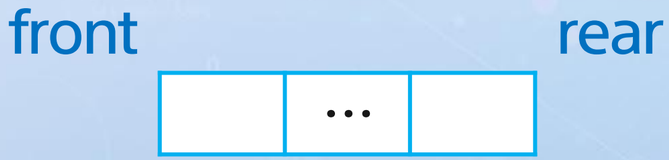
큐 객체 생성 테스트

선형 큐의 구현

» 선형 큐 클래스

```
class LinearQueue :
    def __init__(self):
        self.q_list = []
    def enqueue(self, data):
        self.q_list.append(data)
    def dequeue(self):
        if not self.isEmpty():
            return self.q_list.pop(0)
    def isEmpty(self):
         return len(self.q_list) == 0
```

- 생성자: 자료 저장 공간 생성
  - 파이썬 리스트를 사용하여 배열 구조로 구현
    - ➡ 빈 리스트 생성

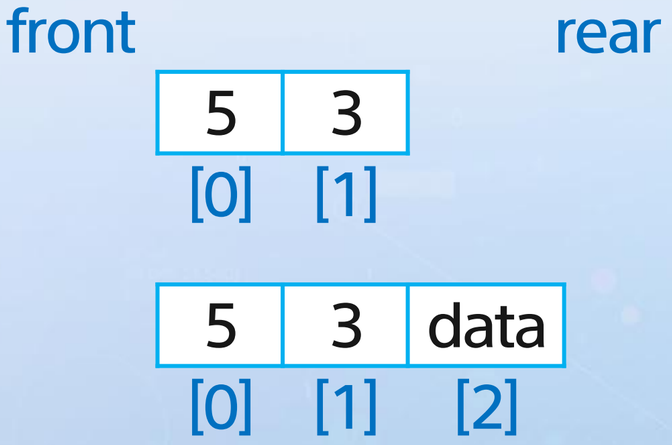


선형 큐의 구현

» 선형 큐 클래스

```
class LinearQueue :
    def __init__(self):
        self.q_list = []
    def enqueue(self, data):
        self.q_list.append(data)
    def dequeue(self):
        if not self.isEmpty():
            return self.q_list.pop(0)
    def isEmpty(self):
         return len(self.q_list) == 0
```

- enqueue(self, data)
  : 자료 넣기

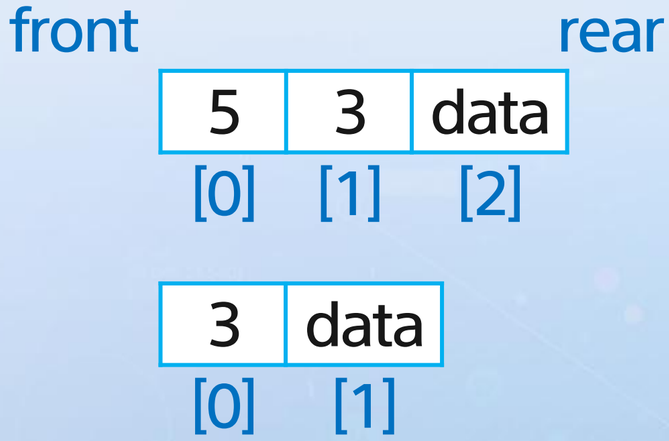


선형 큐의 구현

» 선형 큐 클래스

```
class LinearQueue :
    def __init__(self):
        self.q_list = []
    def enqueue(self, data):
        self.q_list.append(data)
    def dequeue(self):
        if not self.isEmpty():
            return self.q_list.pop(0)
    def isEmpty(self):
         return len(self.q_list) == 0
```

- dequeue(self): 자료 꺼내기
  리스트의 맨 앞



선형 큐의 구현

» 선형 큐 클래스

```
class LinearQueue :
    def __init__(self):
        self.q_list = []
    def enqueue(self, data):
        self.q_list.append(data)
    def dequeue(self):
        if not self.isEmpty():
            return self.q_list.pop(0)
    def isEmpty(self):
        return len(self.q_list) == 0
```

- isEmpty(self): 큐가 비었는가?

선형 큐의 구현

» 테스트

- 큐 클래스의 객체를 생성하고 삽입과 삭제를 실행하여 큐의 원리대로 동작하는지 테스트

```
print("선형 큐 테스트")
Queue = Linearqueue()
for i in range(3):
    queue.enqueue(i)

for i in range(3):
    print(queue.dequeue(), end='  ')
```

[0,1,2]

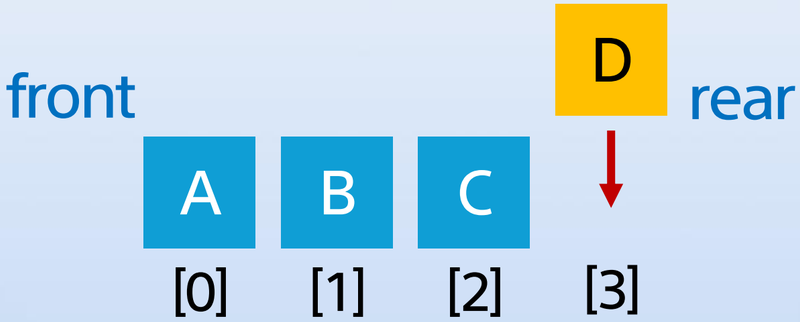
0	1	2
[0]	[1]	[2]

```
선형 큐 테스트
 0 1 2
```

선형 큐의 구현

» 선형 큐의 구현 시 고려해야 할 점

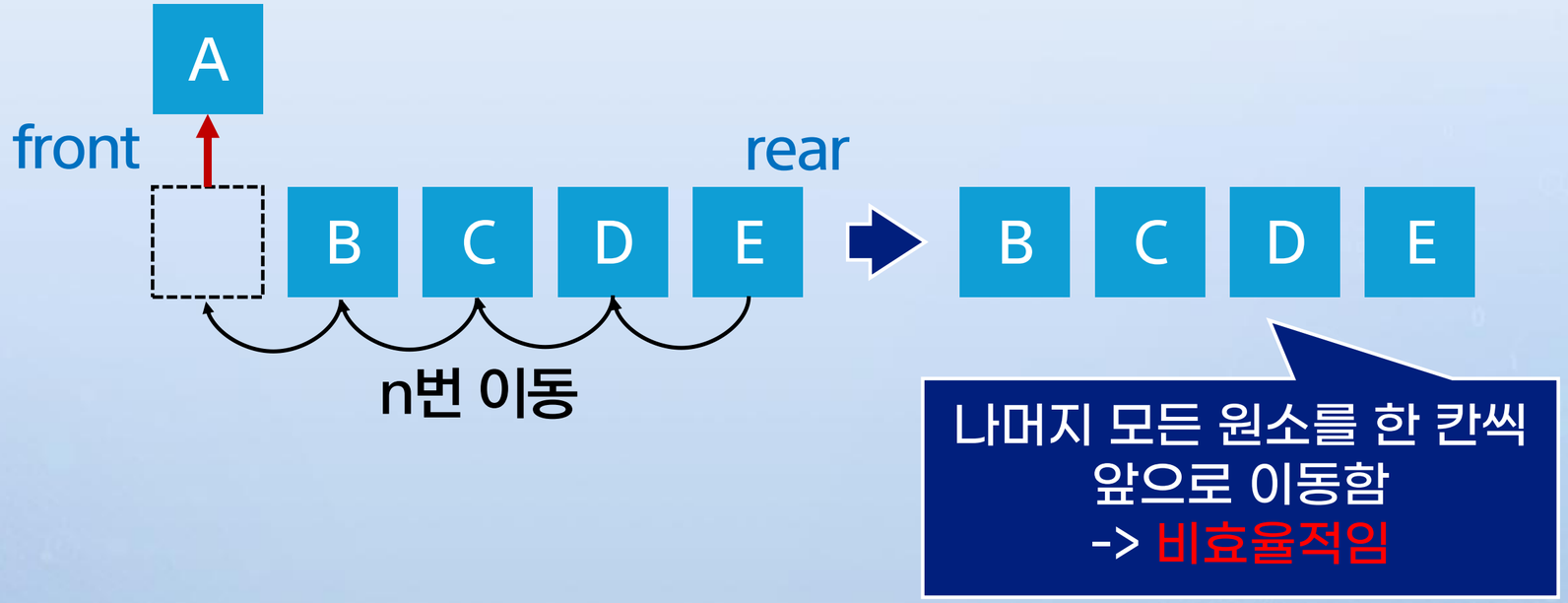
- 삽입: rear에서 일어남
  - 한 번에 실행



선형 큐의 구현

» 선형 큐의 구현 시 고려해야 할 점

- 삭제: front에서 일어남
  - 삭제 후 뒤쪽의 원소를 하나씩 이동해야 함



학습정리

큐(Queue)

- 먼저 들어온 것을 먼저 처리하는 구조
- 컴퓨터 프로세스 작업 대기열 등에 사용되는 구조
- 선형의 리스트로 구현할 수 있음
  - 구현이 간단함
  - 큐의 원소를 제거하면 빈 공간을 없애기 위한 원소 이동이 필요
  - 삽입이 늘어날 수록 공간의 사용양도 늘어남