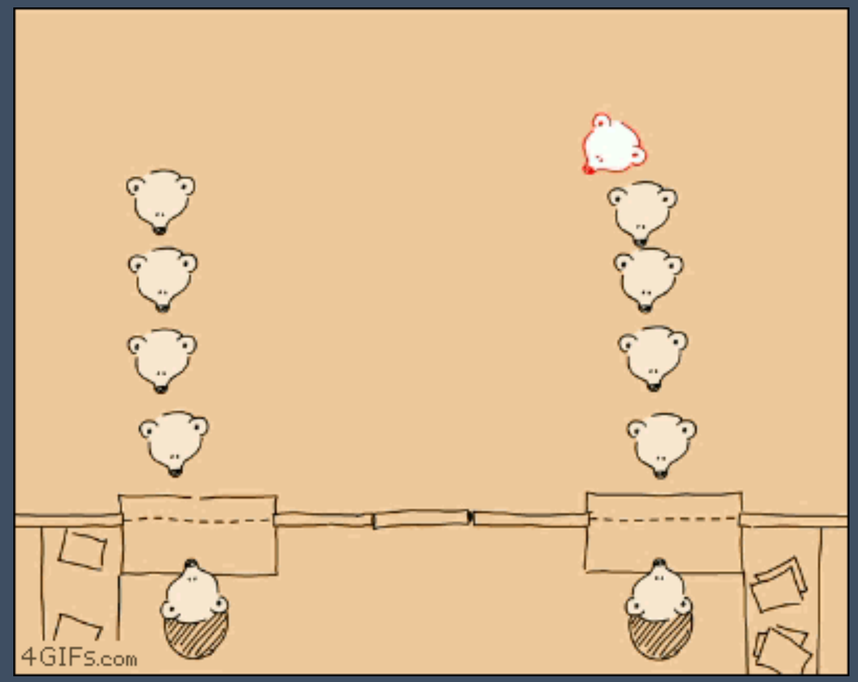
큐 Queue

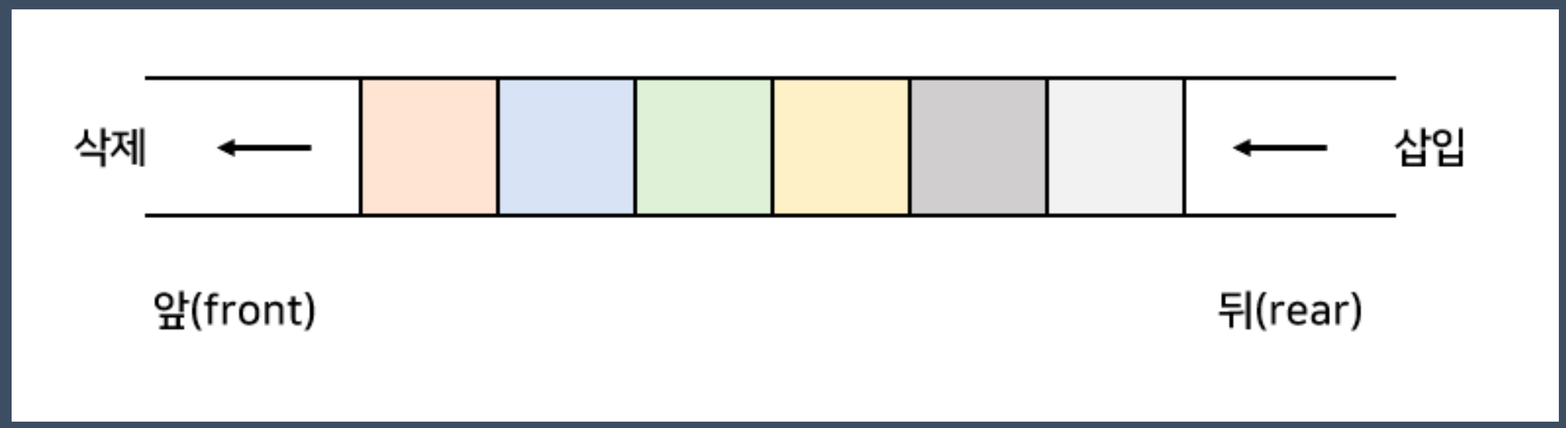
- 큐 개념설명
- 큐 구현방법
- 유형 설명

First In First Out (FIFO 선입선출)
먼저 들어간 자료가 먼저 나온다.

rear에 자료를 넣고 front로 빼는 자료구조
데이터가 들어오고 나가는 방향이 다르다.

(은행 대기열, 또는 터널과 같은 방식)





큐
Queue

rear(끝)에서 자료를 넣고(**enqueue**)
front(앞)에서 자료를 빼는(**dequeue**) 자료구조

1 기본연산

" enqueue (삽입) "

큐의 **맨 뒤**에 데이터를
추가하는 연산

기본연산 2

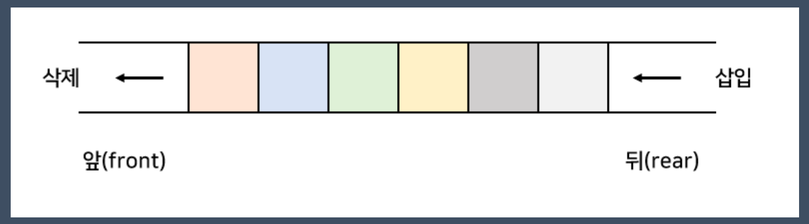
" dequeue (삭제) "

큐의 **맨 앞**에 있는 데이터를
삭제하는 연산

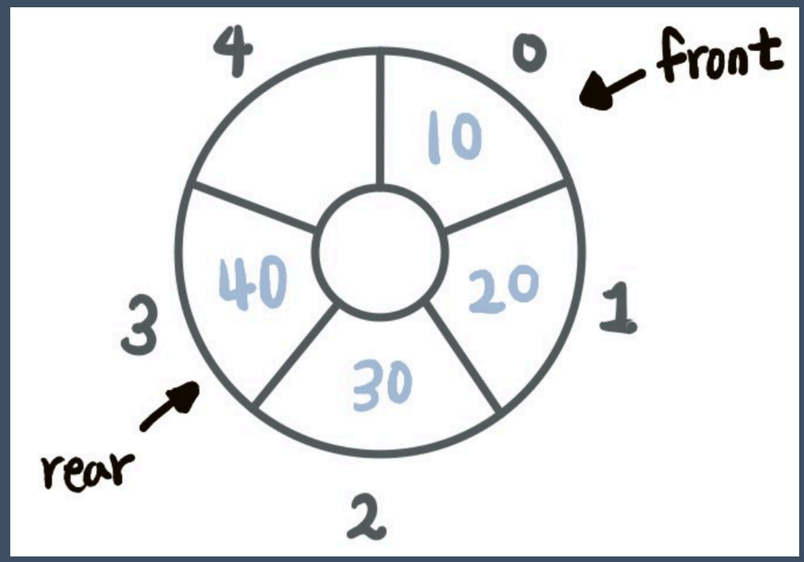
큐
Queue

큐의 종류

- 선형 큐 (linear queue)



- 환형(원형) 큐 (circular queue)



—

큐
Queue

큐 구현

enqueue(int value)
: 큐에 넣는 함수

dequeue()
: 큐에서 빼는 함수

front()
: 맨 앞에 있는 원소 출력

rear()
: 가장 늦게 넣은 원소 출력

큐 구현

1. 배열

배열을 원형으로 사용하여 큐를 구현

2. 링크드 리스트

선형 큐를 구현

배열을 사용한 큐

문제:
https://github.com/Landvibe-DataStructure-2023Study/Prob/blob/main/22%20%EC%8B%A4%EC%8A%B5%20%EB%AC%B8%EC%A0%9C/4%EC%A3%BC%EC%B0%A8/prob-W4_P1_2.pdf

코드:
https://github.com/Landvibe-DataStructure-2023Study/LimJumin/blob/main/22%20%EC%8B%A4%EC%8A%B5%20%EC%BD%94%EB%93%9C/%ED%81%90/22_w4_p1-2.cpp

구현 1

배열을 사용한 큐

```
int n; // 입력받을 큐 Q의 크기를 전역변수로 선언(클래스 내에서도 사용해줄 거라서)

class Queue {
public:
    int arr[10000]; // 최대 크기 만큼 배열 선언
    int frontIdx = 0; // 데이터가 들어올 인덱스 0으로 초기화
    int rearIdx = -1; // 데이터가 나갈 인덱스 -1로 초기화(아무것도 없을 때)
    int size = 0; // 사이즈 0으로 초기화

    void printSize() { cout << size << "\n"; }
    bool isEmpty() { return size == 0; }
```

실습시간 내 빠른 구현과 편의를 위해
그냥 모두 public으로 선언하고, 생성자 따로 구현 없이 선언과 동시에 초기화 해줌ㅜ

구현 1

배열을 사용한 큐

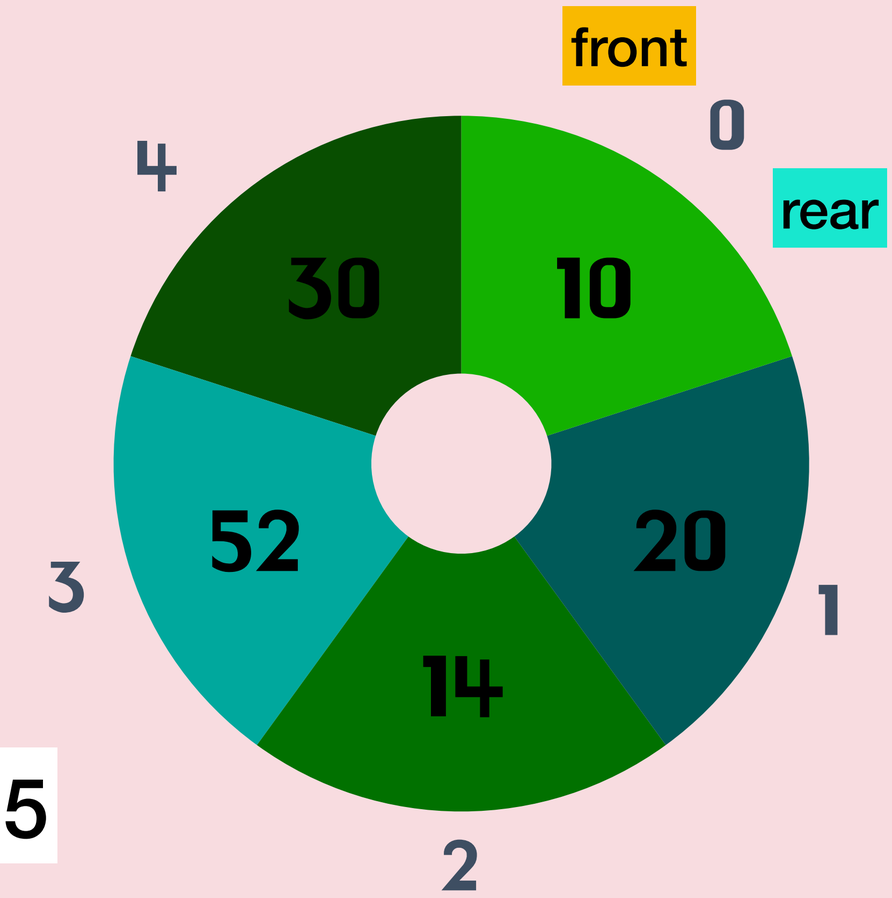
enqueue(int value)

size 넘어가면 문제 조건따라 에러처리

1. rear인덱스 1증가
+) ?
2. arr[rear인덱스]에 값 할당
3. size 1증가

원형큐 활용

선형큐 활용할시 -> 요소 추가할 때마다
모든 원소를 옆으로 이동해줘야하는
번거로운 연산이 필요함



size=5

배열을 사용한 큐

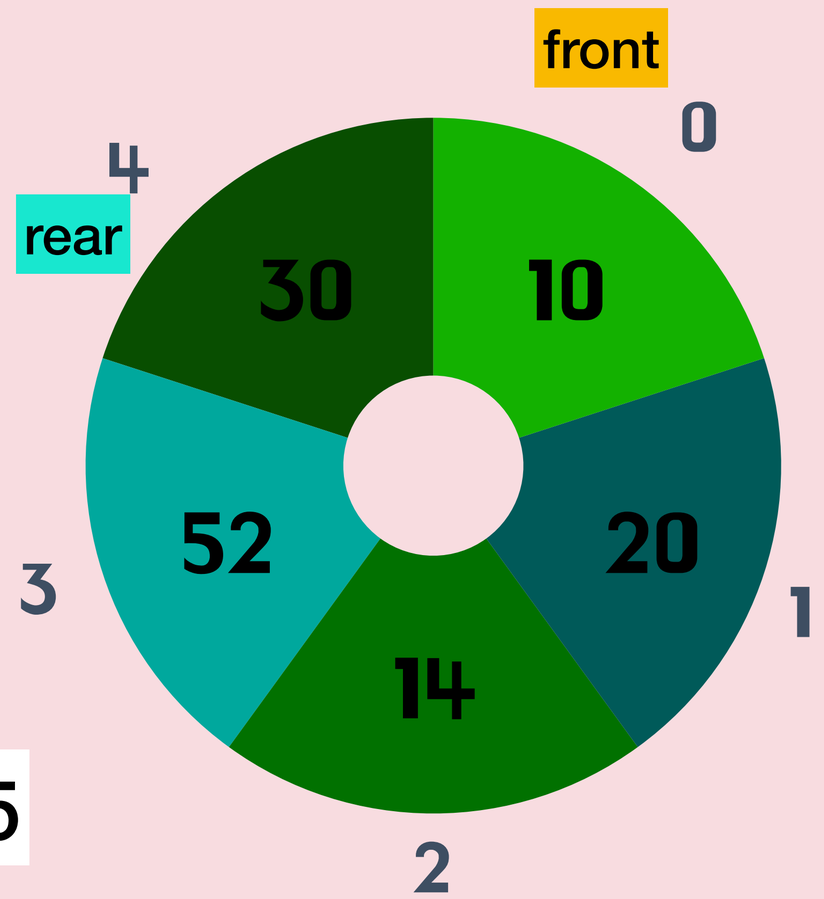
dequeue()

size 0이면 문제 조건따라 에러처리

1. arr[front인덱스] 값 출력
2. front인덱스 1증가

+) ?

3. size 1감소



배열을 사용한 큐

enqueue(int value)

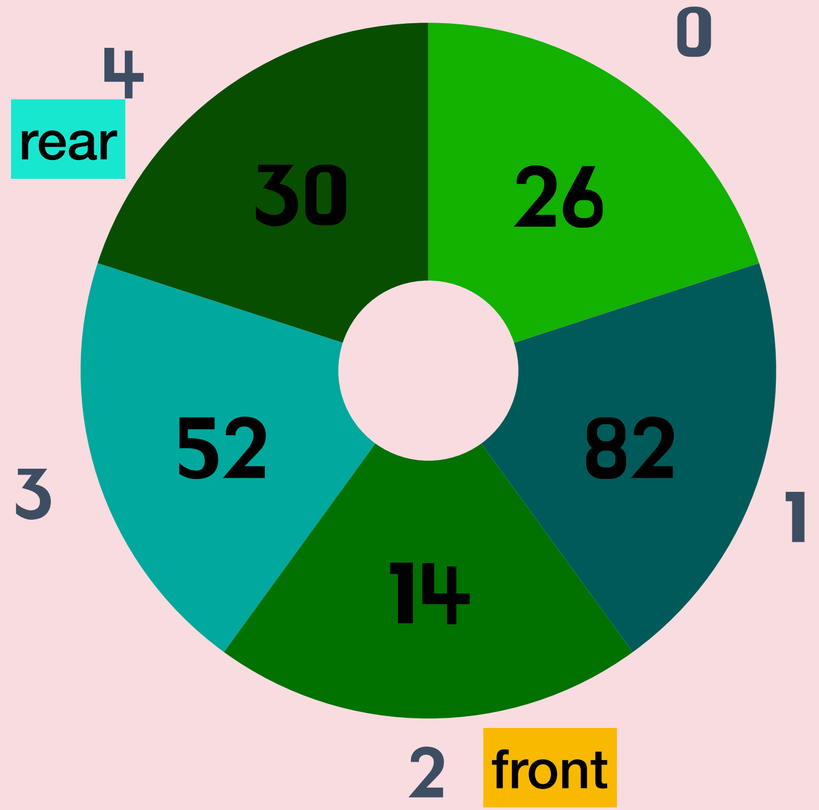
size 넘어가면 문제 조건따라 에러처리

1. rear인덱스 1증가
+) updateIndex() 실행
2. arr[rear인덱스]에 값 할당
3. size 1증가

updateIndex()

각 인덱스를 size로 나눈 나머지

front인덱스 = front인덱스 % size
rear인덱스 = rear인덱스 % size



배열을 사용한 큐 정리

enqueue(int value)

size 넘어가면 문제 조건따라 에러처리

1. rear인덱스 1 증가
2. updateIndex() 실행
3. arr[rear인덱스]에 값 할당
4. size 1 증가

dequeue()

size 0이면 문제 조건따라 에러처리

1. arr[front인덱스] 값 출력
2. front인덱스 1 증가
3. updateIndex() 실행
4. size 1 감소

updateIndex()

각 인덱스를 size(n)으로 나눠준 나머지

front인덱스 %= n
rear인덱스 %= n

링크드 리스트를 사용한 큐

문제:
https://github.com/Landvibe-DataStructure-2023Study/Prob/blob/main/22%20%EC%8B%A4%EC%8A%B5%20%EB%AC%B8%EC%A0%9C/4%EC%A3%BC%EC%B0%A8/prob-W4_P1_1.pdf

코드:
https://github.com/Landvibe-DataStructure-2023Study/LimJumin/blob/main/22%20%EC%8B%A4%EC%8A%B5%20%EC%BD%94%EB%93%9C/%ED%81%90/22_w4_p1-1.cpp

구현 2

링리를 사용한 큐

```
class Node {
public:
    Node *next = nullptr;
    int value;
};
```

```
class Queue {
public:
    Node* head = nullptr;
    Node* tail = nullptr;
    int size = 0;

    int getSize() { return size; }
    bool isEmpty() { return size==0; }
```

실습시간 내 빠른 구현과 편의를 위해
그냥 모두 public으로 선언하고,
생성자 따로 구현 없이 선언과 동시에 초기화 해줌
여러분들의 correct를 위해 양해 하세요~ㅜ

링리를 사용한 큐

enqueue(int value)

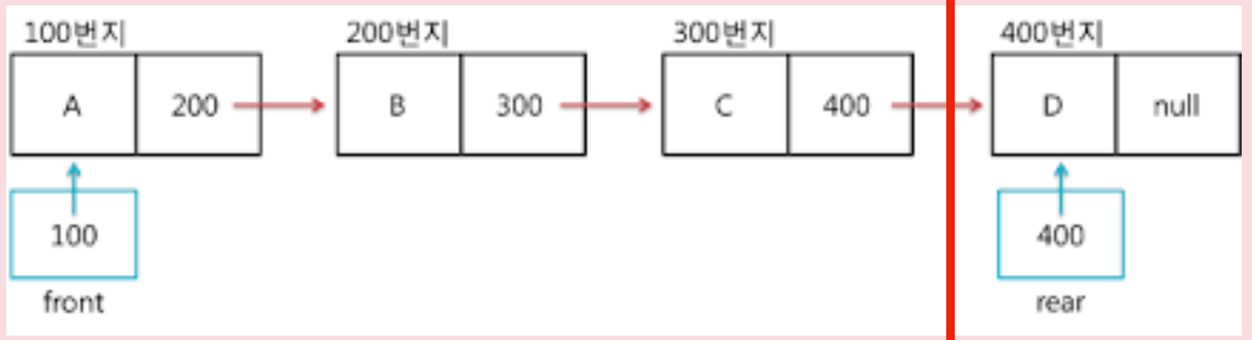
size 넘어가면 문제 조건따라 에러처리

1. 새 노드 생성 후 값 할당

2-1. 빈 경우
head와 tail 모두 새 노드로 할당
사이즈 1 증가

2-2. 안 비었던 경우(else)
Tail의 next를 새 노드로 할당
tail을 새 노드로 업데이트
사이즈 1 증가

tail(rear)에 추가하는 경우만 존재



링리를 사용한 큐

dequeue()

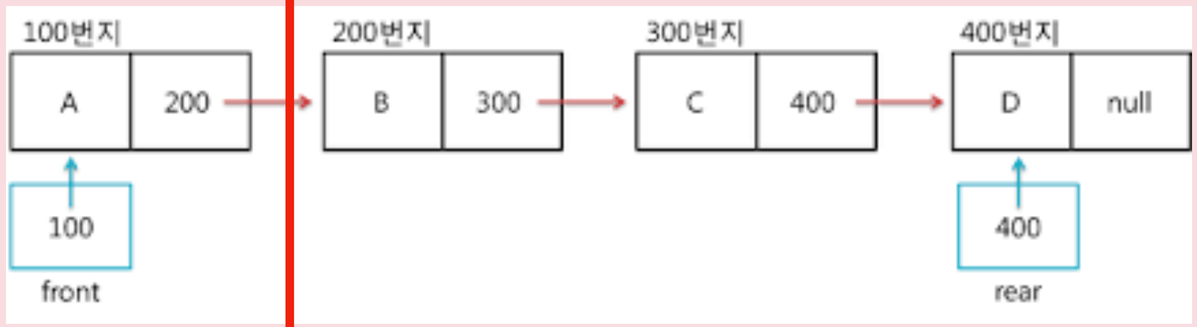
size 0이면 문제 조건따라 에러처리

1. 삭제할 노드를 저장할 노드 생성 후 head로 초기화
2. 삭제할 노드의 값 출력

3-1. size가 1일 경우(삭제하면 비게될 경우)
head와 tail 모두 nullptr로
삭제할 노드 delete
size 1 감소

3-2. 삭제해도 노드 존재할 경우(else)
head를 다음 노드로 업데이트
삭제할 노드 delete
size 1 감소

head(front) 삭제하는 경우만 존재



큐 응용

1. 카드 덱 문제

문제: https://github.com/Landvibe-DataStructure-2023Study/Prob/blob/main/21%20%EC%8B%A4%EC%8A%B5%20%EB%AC%B8%EC%A0%9C/Prob_W4/prob-W4_P2.pdf

코드: https://github.com/Landvibe-DataStructure-2023Study/LimJumin/blob/main/21%20%EC%8B%A4%EC%8A%B5%20%EC%BD%94%EB%93%9C/%ED%81%90/21_w4_p2.cpp

응용 1 : 카드 덱

사이즈 넘어가는 경우 X, 사이즈 0에서 삭제할 경우 없으므로 필요한 부분만 구현해도 괜찮
배열로 큐 구현해서 하면 enqueue와 dequeue 모두 한 줄로 구현하기 가능

Solve)

플레이어 수만큼 큐 하나씩 생성
각 플레이어마다 잔여체력과 승점을 저장할 변수 생성
각 플레이어마다 카드 수만큼 반복해서 카드 입력 받고 큐에 넣기

카드 수만큼 게임 반복 실행 {
  플레이어 1의 카드 = dequeue한 값 + 플레이어 1 잔여체력
  플레이어 2의 카드 = dequeue한 값 + 플레이어 2 잔여체력

  (플레이어 1 카드가 플레이어 2 카드보다 높은 경우) {
    플레이어 1에게 줄 잔여체력 = 플레이어 1 카드 - 플레이어 2 카드
    플레이어 2에게 줄 잔여체력 = 0
    플레이어 승점 1 추가
  } else if, else 써서 비슷하게 반복~
}

문제 조건대로 출력

이부분 반드시
문제 조건따라서

문제 풀어보기~

21년도 : https://github.com/Landvibe-DataStructure-2023Study/LimJumin/tree/main/21%20%EC%8B%A4%EC%8A%B5%20%EC%BD%94%EB%93%9C/%ED%81%90

22년도 : https://github.com/Landvibe-DataStructure-2023Study/LimJumin/tree/main/22%20%EC%8B%A4%EC%8A%B5%20%EC%BD%94%EB%93%9C/%ED%81%90

—

큐
Queue