
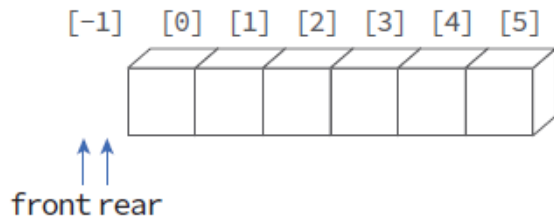


6 주차 실습

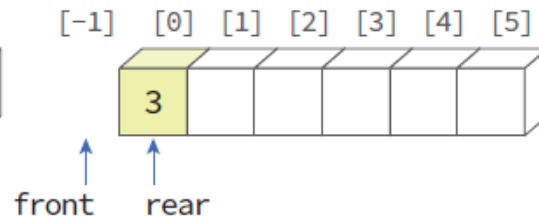




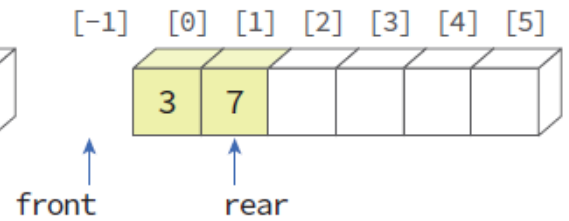
일반(선형) 큐



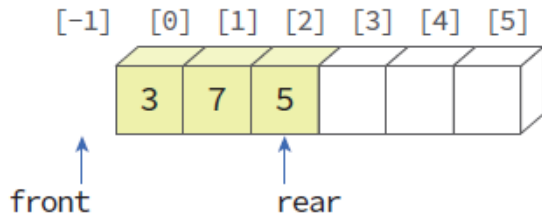
(a) 초기상태



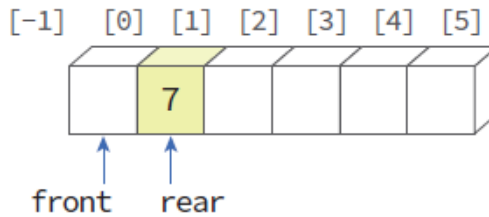
(b) enqueue(3)



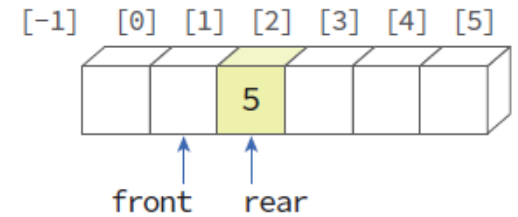
(c) enqueue(7)



(d) enqueue(5)



(e) dequeue()



(f) dequeue()



일반(선형) 큐

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_QUEUE_SIZE 5
typedef int element;
typedef struct { // 큐 타입
    int front;
    int rear;
    element data[MAX_QUEUE_SIZE];
} QueueType;
void error(char* message){
    fprintf(stderr, "%s\n", message);
    exit(1);
}
void init_queue(QueueType* q){
    q->rear = -1;
    q->front = -1;
}
void queue_print(QueueType* q){
    for (int i = 0; i < MAX_QUEUE_SIZE; i++) {
        if (i <= q->front || i > q->rear)
            printf("    | ");
        else
            printf("%d | ", q->data[i]);
    }
    printf("\n");
}
```

```
int is_full(QueueType* q){
    if (q->rear == MAX_QUEUE_SIZE - 1)
        return 1;
    else
        return 0;
}

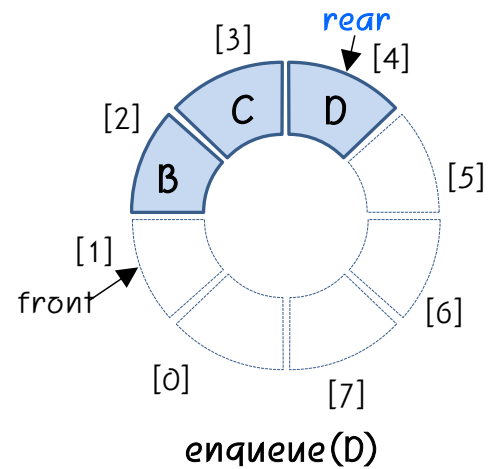
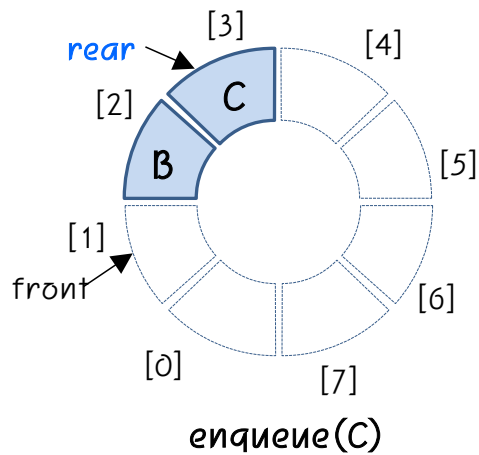
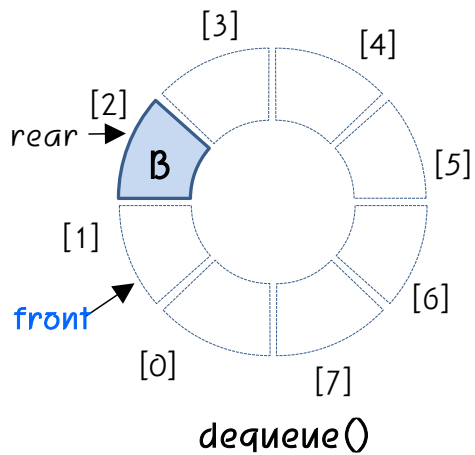
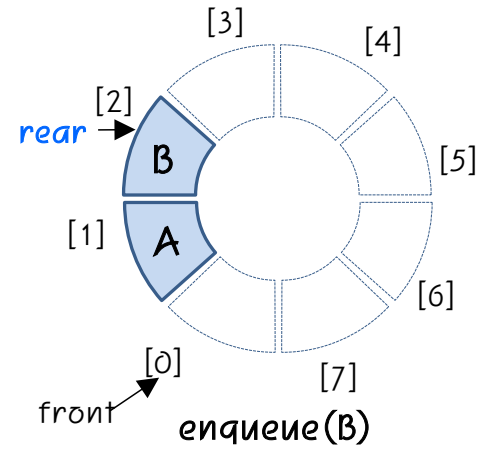
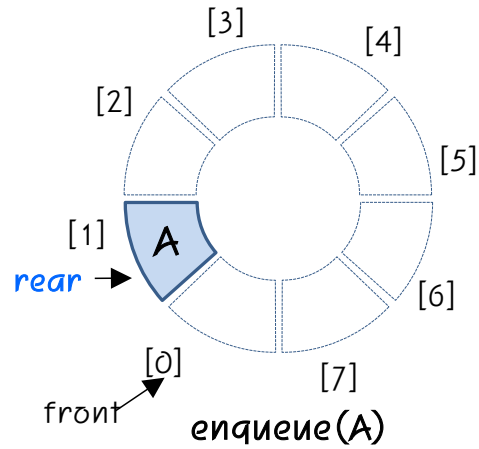
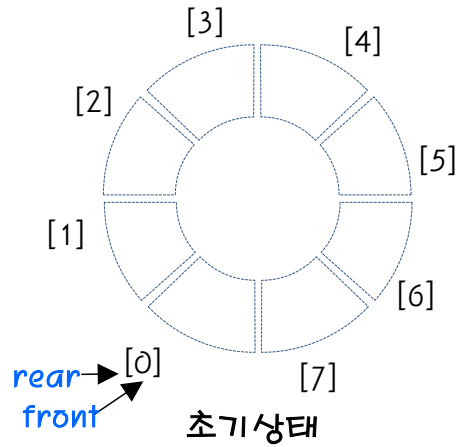
int is_empty(QueueType* q){
    if (q->front == q->rear)
        return 1;
    else
        return 0;
}

void enqueue(QueueType* q, int item){
    if (is_full(q)) {
        error("큐가 포화상태입니다.");
        return;
    }
    q->data[++(q->rear)] = item;
}

int dequeue(QueueType* q){
    if (is_empty(q)) {
        error("큐가 공백상태입니다.");
        return -1;
    }
    int item = q->data[++(q->front)];
    return item;
}
```



일반(원형) 큐





일반(원형) 큐

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_QUEUE_SIZE 5
typedef int element;
typedef struct { // 큐 타입
    int front;
    int rear;
    element data[MAX_QUEUE_SIZE];
} QueueType;
void error(char* message){
    fprintf(stderr, "%s\n", message);
    exit(1);
}
void init_queue(QueueType* q){
    q->rear = 0;
    q->front = 0;
}
void queue_print(QueueType* q){
    printf("QUEUE(front=%d rear=%d) = ", q->front, q->rear);
    if (!is_empty(q)) {
        int i = q->front;
        do {
            i = (i + 1) % (MAX_QUEUE_SIZE);
            printf("%d | ", q->data[i]);
            if (i == q->rear)
                break;
        } while (i != q->front);
    }
    printf("\n");
}
```

```
int is_full(QueueType* q){
    return ((q->rear + 1) % MAX_QUEUE_SIZE == q->front);
}

int is_empty(QueueType* q){
    return (q->front == q->rear);
}

void enqueue(QueueType* q, element item){
    if (is_full(q))
        error("큐가 포화상태입니다");
    q->rear = (q->rear + 1) % MAX_QUEUE_SIZE;
    q->data[q->rear] = item;
}

element dequeue(QueueType* q){
    if (is_empty(q))
        error("큐가 공백상태입니다");
    q->front = (q->front + 1) % MAX_QUEUE_SIZE;
    return q->data[q->front];
}

element peek(QueueType* q){
    if (is_empty(q))
        error("큐가 공백상태입니다");
    return q->data[(q->front + 1) % MAX_QUEUE_SIZE];
}
```

실습 (partice_13.c)



큐를 활용한 피보나치 수열 계산

- 원형 큐를 활용하여 13 번째 까지의 피보나치 수열을 계산하고 이를 출력 해보시오.
 - 피보나치 수열
 - (0), (1), 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233
1. 초기 0, 1을 q에 집어넣는다.
 2. 이후 for문을 13번 반복하면서 디큐/인큐를 통해 피보나치 수열 계산

```
#define MAX_QUEUE_SIZE 5  
  
... 원형 큐 소스코드 ...  
void main() {  
    QueueType queue;  
    ...  
}
```

1 2 3 5 8 13 21 34 55 89 144 233 377



덱 큐

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_QUEUE_SIZE 5
typedef int element;
typedef struct { // 큐 타입
    element data[MAX_QUEUE_SIZE];
    int front, rear;
} DequeType;
void error(char* message){
    fprintf(stderr, "%s\n", message);
    exit(1);
}
void init_deque(DequeType* q){
    q->front = q->rear = 0;
}
int is_empty(DequeType* q){
    return (q->front == q->rear);
}
int is_full(DequeType* q){
    return ((q->rear + 1) % MAX_QUEUE_SIZE == q->front);
}
void add_rear(DequeType* q, element item){
    if (is_full(q))
        error("큐가 포화상태입니다");
    q->rear = (q->rear + 1) % MAX_QUEUE_SIZE;
    q->data[q->rear] = item;
}
element get_front(DequeType* q){
    if (is_empty(q))
        error("큐가 공백상태입니다");
    return q->data[(q->front + 1) % MAX_QUEUE_SIZE];
}
```

```
void deque_print(DequeType* q){
    printf("DEQUE(front=%d rear=%d) = ", q->front, q->rear);
    if (!is_empty(q)) {
        int i = q->front;
        do {
            i = (i + 1) % (MAX_QUEUE_SIZE);
            printf("%d | ", q->data[i]);
            if (i == q->rear)
                break;
        } while (i != q->front);
    }
    printf("\n");
}
void add_front(DequeType* q, element val){
    if (is_full(q))
        error("큐가 포화상태입니다");
    q->data[q->front] = val;
    q->front = (q->front - 1 + MAX_QUEUE_SIZE) % MAX_QUEUE_SIZE;
}
element get_rear(DequeType* q){
    if (is_empty(q))
        error("큐가 공백상태입니다");
    return q->data[q->rear];
}
element delete_rear(DequeType* q){
    int prev = q->rear;
    if (is_empty(q))
        error("큐가 공백상태입니다");
    q->rear = (q->rear - 1 + MAX_QUEUE_SIZE) % MAX_QUEUE_SIZE;
    return q->data[prev];
}
element delete_front(DequeType* q){
    if (is_empty(q))
        error("큐가 공백상태입니다");
    q->front = (q->front + 1) % MAX_QUEUE_SIZE;
    return q->data[q->front];
}
```

실습 (partice_14.c)



덱을 활용한 회문 체크

- 특정 문자열을 배열에 넣고 이를 덱에 넣고 회문인지 아닌지에 대한 것을 체크하는 프로그램을 작성하시오
- 이 때, 덱의 개수(get_count)를 구하는 함수를 추가하여 프로그램에 활용하시오

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
... 덱 큐 소스코드 ...

int get_count(DequeType* q){
    ...
}

void main() {
    DequeType queue;
    char string[100];
    printf("문자열을 입력해주세요: ");
    gets_s(string, 100);
    ...
}
```