

```

#define _CRT_SECURE_NO_WARNINGS //scanf 사용가능한 코드
#include <stdio.h> //printf 함수 사용가능한 코드
#include <stdlib.h> //malloc 함수 라이브러리 불러오기

#define MAX_QUEUE_SIZE 5 //크기 큐 세팅 정의 = 5
typedef int element; //이러 자료형을 element 자료형으로 제한 가능.
typedef struct { //구조체 선언
    element data[MAX_QUEUE_SIZE]; //element 자료형의 data 배열 선언 (크기=5)
    int front, rear; //이러 자료형의 front, rear 변수 선언
} QueueType; //구조체 이름 QueueType

void error(const char* message) //이러 표기 함수
{
    fprintf(stderr, "%s\n", message); //오류 메시지 출력
    exit(1); //이러로 인한 강제종료
}

void init_queue(QueueType* q) //큐 초기화 함수
{
    q->front = q->rear = 0; //큐의 front와 rear를 0으로 설정 (공백상태)
}

int is_empty(QueueType* q) //공백 여부 함수
{
    return (q->front == q->rear); //큐의 front와 rear를 같게 해서 반환함 (공백상태)
}

int is_full(QueueType* q) //포화상태 여부 함수
{
    return ((q->rear + 1) % MAX_QUEUE_SIZE == q->front); //큐의 rear+1을 큐의 리저렉션(5)
    //로 나누었을 때의 나머지가 == 큐의 front
    //반환
}

```



```

void queue_print (QueueType *q) // 큐 출력 함수
{
    printf("QUEUE (front=%d rear=%d) = ", q->front, q->rear); // 큐의 front와 rear 출력.
    if (!is_empty(q)) { // 만약 큐가 비어있지 않다면
        int i = q->front // 큐의 front를 i에 대입
        do { // do while 문
            i = (i+1) % (MAX_QUEUE_SIZE); //  $i = (q \rightarrow front + 1) \% MAX\_QUEUE\_SIZE$  대입
            printf("%d | ", q->data[i]); // 큐의 data의 i번째에 요소 출력 (i번재 요소)
            if (i == q->rear) // 만약 i가 rear와 요소라면 (rear까지)
                break; // 탈출
        } while (i != q->front); // i가 q->front가 아닌 동안 무한 반복
    }
    printf("\n"); // 새 라인에 출력
}

```

```

void enqueue (QueueType *q, element item) // 큐 삽입 함수

```

```

{
    if (is_full(q)) // 큐가 꽂을 수 없음이라면
        error("큐가 꽂을 수 없습니다."); // 이리 메시지 출력 후 프로그램 종료
    q->rear = (q->rear + 1) % MAX_QUEUE_SIZE; // 큐의 rear 한칸 뒤
    q->data[q->rear] = item; // data의 rear번째 요소에 value 삽입.
}

```

```

element dequeue (QueueType *q) // 큐 삭제 함수

```

```

{
    if (is_empty(q)) // 큐가 공백상태라면
        error("큐가 공백상태입니다."); // 이리 메시지 출력 후 프로그램 종료
    q->front = (q->front + 1) % MAX_QUEUE_SIZE; // 큐의 front 한칸 이동.
    return q->data[q->front]; // 큐 data의 front번째 값 반환 (삭제됨)
}

```

```

element peek (QueueType *q) { // peek 함수.

```

```

    if (is_empty(q)) // 만약 큐가 공백상태라면.
        error("큐가 공백상태입니다."); // 이리 메시지 출력 후 프로그램 종료
    return q->data[(q->front + 1) % MAX_QUEUE_SIZE]; // data의 front + 1 번째
}

```



```

int main(void) //메인함수
{
    QueueType queue; // QueueType의 Queue 선언
    int element; // int 자료형 element 변수 선언

    init_queue(&queue); // 큐 초기화
    srand(time(NULL)); // 난수 생성

    for (int i = 0; i < 100; i++) { // 100번 반복하는 for문
        if (rand() % 5 == 0) { // 난수가 5의 배수라면
            enqueue(&queue, rand() % 100); // 난수를 큐에 삽입.
        }
        queue_print(&queue); // 큐 프린트
        if (rand() % 10 == 0) { // 난수가 10의 배수라면
            int data = dequeue(&queue); // 큐에서 요소 하나 삭제, data 삽입
        }
        queue_print(&queue); // 큐 프린트
    }
    return 0; // 0 반환
}

```



```

QUEUE(front=0 rear=0) =
QUEUE(front=0 rear=0) =
QUEUE(front=0 rear=0) =
QUEUE(front=0 rear=0) =
QUEUE(front=0 rear=0) =
QUEUE(front=0 rear=0) =
QUEUE(front=0 rear=1) = 67
QUEUE(front=0 rear=1) = 67
QUEUE(front=0 rear=1) = 67
QUEUE(front=0 rear=1) = 67
QUEUE(front=0 rear=1) = 67
QUEUE(front=0 rear=1) = 67
QUEUE(front=0 rear=1) = 67
QUEUE(front=0 rear=1) = 67
QUEUE(front=0 rear=1) = 67
QUEUE(front=0 rear=1) = 67
QUEUE(front=0 rear=2) = 67 89
QUEUE(front=0 rear=2) = 67 89
QUEUE(front=0 rear=3) = 67 89 26
QUEUE(front=0 rear=3) = 67 89 26
QUEUE(front=0 rear=4) = 67 89 26 0
QUEUE(front=0 rear=4) = 67 89 26 0
QUEUE(front=0 rear=4) = 67 89 26 0
QUEUE(front=1 rear=4) = 89 26 0
QUEUE(front=1 rear=4) = 89 26 0
QUEUE(front=1 rear=4) = 89 26 0
QUEUE(front=1 rear=4) = 89 26 0
QUEUE(front=1 rear=4) = 89 26 0
QUEUE(front=1 rear=4) = 89 26 0
QUEUE(front=2 rear=4) = 26 0
QUEUE(front=2 rear=4) = 26 0
QUEUE(front=2 rear=4) = 26 0
QUEUE(front=2 rear=0) = 26 0 41
QUEUE(front=2 rear=0) = 26 0 41
QUEUE(front=2 rear=0) = 26 0 41
QUEUE(front=2 rear=0) = 26 0 41
QUEUE(front=2 rear=0) = 26 0 41
QUEUE(front=2 rear=0) = 26 0 41
QUEUE(front=2 rear=1) = 26 0 41 80
QUEUE(front=2 rear=1) = 26 0 41 80
QUEUE(front=2 rear=1) = 26 0 41 80
QUEUE(front=2 rear=1) = 26 0 41 80
QUEUE(front=2 rear=1) = 26 0 41 80
QUEUE(front=2 rear=1) = 26 0 41 80
QUEUE(front=2 rear=1) = 26 0 41 80
QUEUE(front=3 rear=1) = 0 41 80
QUEUE(front=3 rear=1) = 0 41 80
QUEUE(front=3 rear=1) = 0 41 80
QUEUE(front=3 rear=1) = 0 41 80

```



찾기



Hi





```

QUEUE(front=1 rear=4) = 89 | 26 | 0 |
QUEUE(front=1 rear=4) = 89 | 26 | 0 |
QUEUE(front=1 rear=4) = 89 | 26 | 0 |
QUEUE(front=1 rear=4) = 89 | 26 | 0 |
QUEUE(front=1 rear=4) = 89 | 26 | 0 |
QUEUE(front=2 rear=4) = 26 | 0 |
QUEUE(front=2 rear=4) = 26 | 0 |
QUEUE(front=2 rear=4) = 26 | 0 |
QUEUE(front=2 rear=0) = 26 | 0 | 41 |
QUEUE(front=2 rear=0) = 26 | 0 | 41 |
QUEUE(front=2 rear=0) = 26 | 0 | 41 |
QUEUE(front=2 rear=0) = 26 | 0 | 41 |
QUEUE(front=2 rear=0) = 26 | 0 | 41 |
QUEUE(front=2 rear=0) = 26 | 0 | 41 |
QUEUE(front=2 rear=1) = 26 | 0 | 41 | 80 |
QUEUE(front=2 rear=1) = 26 | 0 | 41 | 80 |
QUEUE(front=2 rear=1) = 26 | 0 | 41 | 80 |
QUEUE(front=2 rear=1) = 26 | 0 | 41 | 80 |
QUEUE(front=2 rear=1) = 26 | 0 | 41 | 80 |
QUEUE(front=2 rear=1) = 26 | 0 | 41 | 80 |
QUEUE(front=2 rear=1) = 26 | 0 | 41 | 80 |
QUEUE(front=3 rear=1) = 0 | 41 | 80 |
QUEUE(front=3 rear=1) = 0 | 41 | 80 |
QUEUE(front=3 rear=1) = 0 | 41 | 80 |
QUEUE(front=3 rear=1) = 0 | 41 | 80 |
QUEUE(front=3 rear=1) = 0 | 41 | 80 |
QUEUE(front=3 rear=1) = 0 | 41 | 80 |
QUEUE(front=3 rear=1) = 0 | 41 | 80 |
QUEUE(front=3 rear=1) = 0 | 41 | 80 |
QUEUE(front=3 rear=1) = 0 | 41 | 80 |
QUEUE(front=3 rear=2) = 0 | 41 | 80 | 91 |
QUEUE(front=3 rear=2) = 0 | 41 | 80 | 91 |
QUEUE(front=3 rear=2) = 0 | 41 | 80 | 91 |
QUEUE(front=3 rear=2) = 0 | 41 | 80 | 91 |

```

큐가 포화상태입니다.

C:\Users\한상결\source\repos\testa\Debug\자료구조 실습.exe(프로세스 8056개)이(가)  
 디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅]  
 이 창을 닫으려면 아무 키나 누르세요...



찾기



바





```
do { // do while 문
```

```
i = (i + 1) % (MAX_QUEUE_SIZE); // i에 (i+1)%MAX
```

```
printf("%d | ", q->data[i]); // 큐의 가장 첫번
```

Microsoft Visual Studio 디버그 콘솔

```
QUEUE(front=0 rear=1) = 10
```

```
QUEUE(front=0 rear=1) = 10
```

```
QUEUE(front=0 rear=1) = 10
```

```
QUEUE(front=0 rear=1) = 10
```

```
QUEUE(front=0 rear=1) = 10
```

```
QUEUE(front=0 rear=1) = 10
```

```
QUEUE(front=0 rear=1) = 10
```

```
QUEUE(front=1 rear=1) =
```

```
QUEUE(front=1 rear=1) =
```

```
QUEUE(front=1 rear=1) =
```

```
QUEUE(front=1 rear=1) =
```

```
QUEUE(front=1 rear=1) =
```

```
QUEUE(front=1 rear=1) =
```

```
큐가 공백상태입니다.
```

```
C:\Users\한상결\source\repos\testa\Debug\자료구조 실습.exe
```

```
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [솔
```

```
하도록 설정합니다.
```

```
{ 이 창을 닫으려면 아무 키나 누르세요...
```