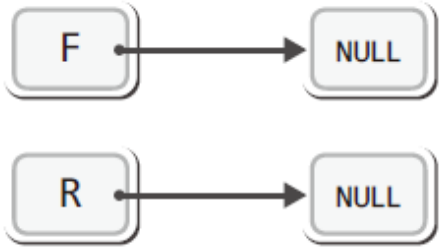
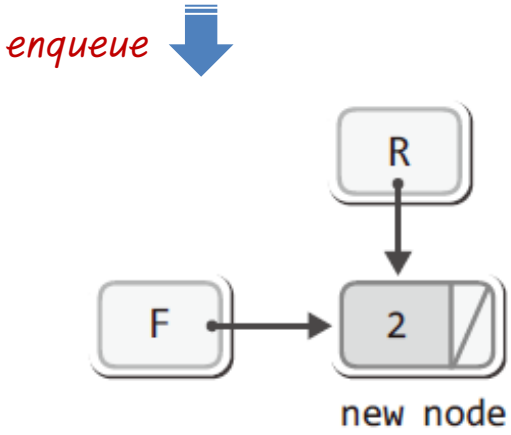
Chapter 07.
큐

11주차

큐의 연결리스트 기반 구현
(초기상태와 Enqueue)

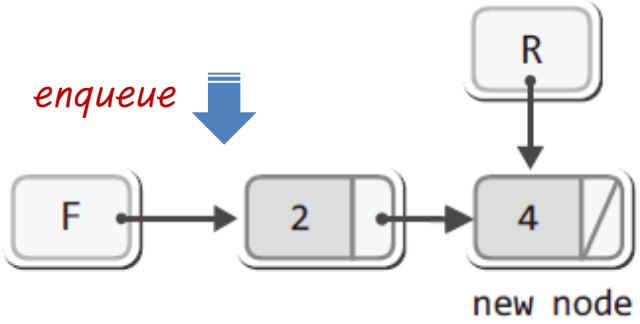


큐의 초기 상태



첫 노드 enqueue할 때 :

F와 R의 변경이 요구됨



두 번째 이후 노드 enqueue할 때 :

R의 변경만 요구됨

큐의 연결리스트 기반 구현
(Dequeue)

F가 다음 노드를 가리키게 하고, F가 이전에 가리키던 노드를 소멸시키기

dequeue의 구현을 하나로 통일하기 위해 R은 그냥 내버려 둔다!

Empty 체크할 때 front만 보게 했으므로 OK.

큐의 연결리스트 기반 구현
(구조체 정의)

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>

#define TRUE      1
#define FALSE     0

typedef int Data;
typedef struct _node {
        Data data;
        struct _node * next;
} Node;

typedef struct _lQueue {
        Node * front;
        Node * rear;
} Queue;
```

노드용 구조체와
큐용 구조체가 필요!

큐의 연결리스트 기반 구현
(초기화, empty 체크)

```
void QueueInit(Queue * pq)
{
        pq->front = NULL;
        pq->rear = NULL;
}

int QIsEmpty(Queue * pq)
{
        if (pq->front == NULL)
                return TRUE;
        else
                return FALSE;
}
```

front로써 체크.

큐의 연결리스트 기반 구현
(enqueue)

```
void Enqueue(Queue * pq, Data data)
{
	Node * newNode = (Node*)malloc(sizeof(Node));
	newNode->next = NULL;
	newNode->data = data;

	if (QIsEmpty(pq))
	{
		pq->front = newNode;
		pq->rear = newNode;
	}
	else
	{
		pq->rear->next = newNode;
		pq->rear = newNode;
	}
}
```

Full check :
malloc 결과가 NULL인지 체크한다.

큐의 연결리스트 기반 구현
(dequeue)

```
Data Dequeue(Queue * pq)
{
        Node * delNode;
        Data retData;

        if (QIsEmpty(pq))
        {
                printf("Queue Memory Error!");
                exit(-1);
        }

        delNode = pq->front;
        retData = delNode->data;
        pq->front = pq->front->next;

        free(delNode);
        return retData;
}
```

front 전진시킨 후 이전 front를 free 하려면 별도로 저장해야 함.

delNode를 free 시키고 delNode->data를 반환하려면 별도로 저장해야 함.

큐의 연결리스트 기반 구현
(QPeek)

```
Data QPeek(Queue * pq)
{
	if (QIsEmpty(pq))
	{
		printf("Queue Memory Error!");
		exit(-1);
	}

	return pq->front->data;
}
```

원형큐의 리스트기반 구현
(main 함수 : 배열기반 구현의 main함수와 같음.)

```
int main(void) {
        Queue q;
        QueueInit(&q);

        Enqueue(&q, 1);   Enqueue(&q, 2);
        Enqueue(&q, 3);   Enqueue(&q, 4);
        Enqueue(&q, 5);

        while (!QIsEmpty(&q))
                printf("%d ", Dequeue(&q));

        return 0;
}
```

Queue의 생성 및 초기화

데이터 넣기

데이터 꺼내기

Deque(덱)의 ADT
(front, rear에서 각각 add, remove 가능)

- `void DequeInit(Deque * pdeq);`
  - 덱의 초기화를 진행한다.
  - 덱 생성 후 제일 먼저 호출되어야 하는 함수이다.

- `int DQIsEmpty(Deque * pdeq);`
  - 덱이 빈 경우 TRUE(1)을, 그렇지 않은 경우 FALSE(0)을 반환한다.

Deque의 4가지 연산

- `void DQAddFirst(Deque * pdeq, Data data);`
  - 덱의 머리에 데이터를 저장한다. data로 전달된 값을 저장한다.
- `void DQAddLast(Deque * pdeq, Data data);`
  - 덱의 꼬리에 데이터를 저장한다. data로 전달된 값을 저장한다.
- `Data DQRemoveFirst(Deque * pdeq);`
  - 덱의 머리에 위치한 데이터를 반환 및 소멸한다.
- `Data DQRemoveLast(Deque * pdeq);`
  - 덱의 꼬리에 위치한 데이터를 반환 및 소멸한다.
- `Data DQGetFirst(Deque * pdeq);`
  - 덱의 머리에 위치한 데이터를 소멸하지 않고 반환한다.
- `Data DQGetLast(Deque * pdeq);`
  - 덱의 꼬리에 위치한 데이터를 소멸하지 않고 반환한다.

양방향 연결리스트로 Deque(덱) 구현
(구조체 정의)

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>

#define TRUE      1
#define FALSE     0

typedef int Data;
typedef struct _node {
        Data data;
        struct _node * next;
        struct _node * prev;
} Node;

typedef struct _dlDeque {
        Node * head;
        Node * tail;
} Deque;
```

양방향 연결리스트로 Deque(덱) 구현
(초기화, empty 확인)

```
void DequeInit(Deque * pdeq)
{
	pdeq->head = NULL;
	pdeq->tail = NULL;
}

int DQIsEmpty(Deque * pdeq)
{
	if (pdeq->head == NULL)
		return TRUE;
	else
		return FALSE;
}
```

양방향 연결리스트로 Deque(덱) 구현
(노드 추가)

```
void DQAddFirst(Deque * pdeq, Data data) {
        Node * newNode = (Node*)malloc(sizeof(Node));
        newNode->data = data;
        newNode->prev = newNode->next = NULL;

        if (pdeq->head == NULL)
                pdeq->head = pdeq->tail = newNode;
        else {
                pdeq->head->prev = newNode;
                newNode->next = pdeq->head;
                pdeq->head = newNode;
        }
}
```

양방향 연결리스트로 Deque(덱) 구현
(노드 추가)

```
void DQAddLast(Deque * pdeq, Data data) {
        Node * newNode = (Node*)malloc(sizeof(Node));
        newNode->data = data;
        newNode->prev = newNode->next = NULL;

        if (pdeq->tail == NULL)
                pdeq->head = pdeq->tail = newNode;
        else {
                pdeq->tail->next = newNode;
                newNode->prev = pdeq->tail;
                pdeq->tail = newNode;
        }
}
```

양방향 연결리스트로 Deque(덱) 구현
(맨 앞 노드 삭제)

```
Data DQRemoveFirst(Deque * pdeq) {
        Node * rnode = pdeq->head;
        Data rdata;

        if (DQIsEmpty(pdeq)) {
                printf("Deque Memory Error!");
                exit(-1);
        }
        rdata = pdeq->head->data;

        pdeq->head = pdeq->head->next;
        free(rnode);

        if (pdeq->head == NULL)
                pdeq->tail = NULL;
        else
                pdeq->head->prev = NULL;

        return rdata;
}
```

양방향 연결리스트로 Deque(덱) 구현
(맨 뒤 노드 삭제)

```
Data DQRemoveLast(Deque * pdeq) {
        Node * rnode = pdeq->tail;
        Data rdata;

        if (DQIsEmpty(pdeq)) {
                printf("Deque Memory Error!");
                exit(-1);
        }
        rdata = pdeq->tail->data;

        pdeq->tail = pdeq->tail->prev;
        free(rnode);

        if (pdeq->tail == NULL)
                pdeq->head = NULL;
        else
                pdeq->tail->next = NULL;

        return rdata;
}
```

양방향 연결리스트로 Deque(덱) 구현
(맨 앞 / 맨 뒤 노드 peek)

```
Data DQGetFirst(Deque * pdeq) {
        if (DQIsEmpty(pdeq)) {
                printf("Deque Memory Error!");
                exit(-1);
        }

        return pdeq->head->data;
}

Data DQGetLast(Deque * pdeq) {
        if (DQIsEmpty(pdeq)) {
                printf("Deque Memory Error!");
                exit(-1);
        }

        return pdeq->tail->data;
}
```

양방향 연결리스트로 Deque(덱) 구현
(main 함수)

```
int main(void) {
        Deque deq;
        DequeInit(&deq);

        DQAddFirst(&deq, 3); DQAddFirst(&deq, 2); DQAddFirst(&deq, 1);
        DQAddLast(&deq, 4); DQAddLast(&deq, 5); DQAddLast(&deq, 6);

        while (!DQIsEmpty(&deq))
                printf("%d ", DQRemoveFirst(&deq));
        printf("₩n");

        DQAddFirst(&deq, 3); DQAddFirst(&deq, 2); DQAddFirst(&deq, 1);
        DQAddLast(&deq, 4); DQAddLast(&deq, 5); DQAddLast(&deq, 6);

        while (!DQIsEmpty(&deq))
                printf("%d ", DQRemoveLast(&deq));
        return 0;
}
```

Deque의 생성 및 초기화

데이터 넣기 1차

데이터 꺼내기 1차

데이터 넣기 2차

데이터 꺼내기 2차