**자료구조 실습 보고서**

**실습 7. 고급연결리스트**

**2016년 4월 29일**

**학번: 201404051**

**이름: 정 용 석**

**1. 실습 문제 소개**

강의 자료에 나와 있는 dnode와 deque 구조체를 바탕으로 이중연결리스트를 기반으로 한, 덱 자료구조를 구현하는 문제이다.

**2. 소스 코드**

void printDeque(deque \*deq)

{

dnode \*temp = deq->front;

printf("Deque: ");

if (isEmpty(deq))

printf("\n");

else {

//덱의 Front부터 Rear까지 출력

while (temp != NULL) {

printf("[%c] ", temp->value);

temp = temp->next;

}

printf("\n");

}

}

int isEmpty(deque \*deq) //덱이 EMPTY면 1 출력

{

if (deq->size == 0)

return 1;

else

return 0;

}

void insertFront(deque \*deq, int value)

{

dnode\* newNode = getNode();

newNode->value = value;

//덱이 EMPTY이면 FRONT와 REAR모두 newNode

if (isEmpty(deq))

{

deq->front = newNode;

deq->rear = newNode;

deq->size++;

}

//덱이 EMPTY가 아닐시, 알맞게 연결

else

{

newNode->next = deq->front;

deq->front->prev = newNode;

deq->front = newNode;

deq->size++;

}

}

int deleteFront(deque \*deq)

{

dnode\* oldNode = deq->front;

int value;

//덱이 EMPTY일 시, 프로그램 종료

if (isEmpty(deq)) { exit(0); }

//덱에 노드가 1개일 시, front와 rear모두 NULL

if (deq->size == 1)

{

value = oldNode->value;

deq->front = NULL;

deq->rear = NULL;

returnNode(oldNode);

deq->size--;

return value;

}

else {

value = oldNode->value;

deq->front = oldNode->next;

returnNode(oldNode);

deq->front->prev = NULL;

deq->size--;

return value;

}

}

void insertRear(deque \*deq, int value)

{

dnode\* newNode = getNode();

newNode->value = value;

//덱이 EMPTY일 시, front와 rear모두 newNode

if (isEmpty(deq))

{

deq->rear = newNode;

deq->front = newNode;

deq->size++;

}

else {

newNode->prev = deq->rear;

deq->rear->next = newNode;

deq->rear = newNode;

deq->size++;

}

}

int deleteRear(deque \*deq)

{

dnode \*oldNode = deq->rear;

int value;

//덱이 EMPTY일 시, 프로그램 종료

if (isEmpty(deq)) { exit(0); }

//덱의 노드가 1개일 시, rear와 front모두 NULL

if (deq->size == 1)

{

value = oldNode->value;

deq->rear = NULL;

deq->front = NULL;

returnNode(oldNode);

deq->size--;

return value;

}

else {

value = oldNode->value;

deq->rear = oldNode->prev;

deq->rear->next = NULL;

returnNode(oldNode);

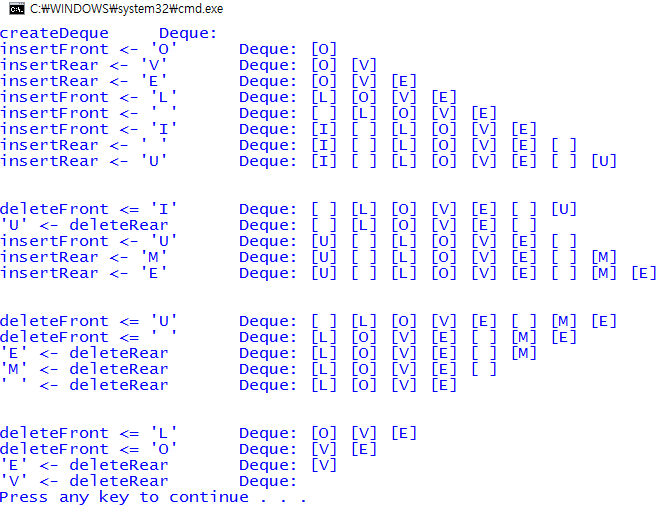
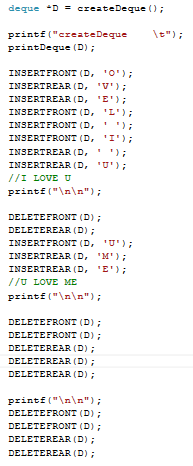
deq->size--;

return value;

}

}

**3. 테스트 결과**



**4. 작성자 코멘트**

고급 연결리스트라고 해봐야 사실 상 단순 연결리스트에서 선 하나 추가된 정도이다. 머리속으로는 쉽게 그려지지 않을 때도 있었지만, 그림만 그린다면 정말 간단하게 구현할 수 있는 자료구조가 아니었나 싶다. 문제점이 있었냐고 한다면 처음에 딱히 손으로 코딩하지 않고, 막연하게 코딩 중 예외 처리에 대한 부분을 모두 생략해서 추가하려고 하니, 코딩이 너무 읽기 힘들어서(직접 작성했음에도 불구하고) 처음부터 다시 작성해야 했던 문제점과 덱이 EMPTY 상태에서 DELETEREAR혹은 DELETEFRONT을 할 때 예외처리를 어떻게 할 것인가에 대해 고민했던 것 정도가 되겠다.

덱 자체가 이중 연결리스트의 자료구조를 가지고 있고, 차이점이라고 해봐야 노드 삭제 시, 데이터를 반환한다는 점 하나만 다르기 때문에 사실 상 이중 연결리스트의 개념을 잘 알고 있다면 이 문제 또한 굉장히 간단하다고 느낄 것 같다.