4장: 이중연결 리스트를 이용한 리스트 ADT(개정 3판)

#include <stdio.h>

#include <stdlib.h>

typedef int element;

typedef struct DListNode { // 이중연결 노드 타입

element data;

struct DListNode\* llink;

struct DListNode\* rlink;

} DListNode;

// 이중 연결 리스트를 초기화

void init(DListNode\* phead)

{

*phead->llink = phead;*

*phead->rlink = phead;*

}

// 이중 연결 리스트의 노드를 출력

void print\_dlist(DListNode\* phead)

{

DListNode\* p;

for (p = *phead->rlink; p != phead; p = p->rlink* ) {

printf("<-| |%d| |-> ", p->data);

}

printf("\n");

}

// 새로운 데이터를 노드 before의 오른쪽에 삽입한다.

void dinsert(DListNode \*before, element data)

{

DListNode \*newnode = (DListNode \*)malloc(sizeof(DListNode));

newnode->data = data;

*newnode->llink = before;*

*newnode->rlink = before->rlink;*

*before->rlink->llink = newnode;*

*before->rlink = newnode;*

}

// 노드 removed를 삭제한다.

void ddelete(DListNode\* head, DListNode\* removed)

{

if (removed == head) return;

*removed->llink->rlink = removed->rlink;*

*removed->rlink->llink = removed->llink;*

*free(removed);*

}

void print\_reverse\_dlist(DListNode\* head)  
{

}  
DListNode\* search(DListNode \*head, element data)  
{

}// 이중 연결 리스트 테스트 프로그램

int main(void)

{

DListNode\* head = (DListNode \*)malloc(sizeof(DListNode));

init(head);

printf("추가 단계\n");

// 맨 앞에 노드를 삽입

dinsert(head, 10);

print\_dlist(head);

dinsert(head, 20);

print\_dlist(head);

dinsert(head, 30);

print\_dlist(head);

// 맨 뒤에 노드를 삽입하려면?

*dinsert(head->llink, 99);*

*print\_dlist(head);*

printf("\n삭제 단계\n");

// 맨 앞의 노드를 삭제

ddelete(head, head->rlink);

print\_dlist(head);

// 맨 뒤의 노드를 삭제하려면?

*ddelete(head, head->llink);*

*print\_dlist(head);*

free(head);

return 0;

}