/\*

2404015 이중 연결 리스트

\*/

1. **리스트를 하나로 결합하는 함수 구현**

#include <stdio.h>

#include <stdlib.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma warning(disable:4996)

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);

}

// 이중 연결 리스트 테스트 프로그램

int main(void)

{

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

init(head);

printf("추가 단계\n");

for (int i = 0; i < 5; i++) {

// 헤드 노드의 오른쪽에 삽입

dinsert(head, i);

print\_dlist(head);

}

printf("\n삭제 단계\n");

for (int i = 0; i < 5; i++) {

print\_dlist(head);

ddelete(head, head->rlink);

}

free(head);

return 0;

}

Mp3 play 리스트

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma warning(disable:4996)

typedef char element[100];

typedef struct DListNode { // 이중연결 노드 타입

element data;

struct DListNode\* llink;

struct DListNode\* rlink;

} DListNode;

DListNode\* current;

// 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) {

if (p == current)

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

else

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

}

printf("\n");

}

// 노드 newnode를 노드 before의 오른쪽에 삽입한다.

void dinsert(DListNode\* before, const element data)

{

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

strcpy(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);

}

// 이중 연결 리스트 테스트 프로그램

int main(void)

{

char ch;

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

init(head);

dinsert(head, "Mamamia");

dinsert(head, "Dancing Queen");

dinsert(head, "Fernando");

current = head->rlink;

print\_dlist(head);

do {

printf("\n명령어를 입력하시오(<, >, q): ");

ch = getchar();

if (ch == '<') {

current = current->llink;

if (current == head)

current = current->llink;

}

else if (ch == '>') {

current = current->rlink;

if (current == head)

current = current->rlink;

}

print\_dlist(head);

getchar();

} while (ch != 'q');

// 동적 메모리 해제 코드를 여기에

}