5645640 제갈건 0321

1. 실행코드

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include <stdlib.h>

#include<malloc.h>

typedef int element;

// 이중 연결리스트 노드 타입

typedef struct DoubleListNode {

element data;

struct DoubleListNode\* Leftlink; //앞 노드를 가리키는 포인터

struct DoubleListNode\* Rightlink; // 뒷 노드를 가리키는 포인터

}DoubleListNode;

// 이중 연결리스트의 노드 출력

void print\_dlist(DoubleListNode\* phead) {

DoubleListNode\* p;

for (p = phead->Rightlink; p != phead; p = p->Rightlink) {

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

}

printf("head\n");

}

//첫 번째 노드에 노드 추가

DoubleListNode\* dinsert\_first(DoubleListNode\* head, element data) {

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

newnode->data = data;

if (head == NULL) {

newnode->Leftlink = newnode;

newnode->Rightlink = newnode;

}

else {

newnode->Leftlink = head->Leftlink;

newnode->Rightlink = head;

head->Leftlink->Rightlink = newnode;

head->Leftlink = newnode;

}

head = newnode; // head 포인터 변경

return head;

}

//원하는 위치의 노드 오른쪽에 삽입

void dinsert\_pos(DoubleListNode\* head, element data, int count) {

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

DoubleListNode\* before = head;

newnode->data = data; // 노드 생성

for (int i = 0; i < count && before->Rightlink != head; i++) { // before->Rightlink != head 노드의 끝에서 멈추는 코드

before = before->Rightlink; // Leftlink를 사용하면 반대 방향으로 이동

}

newnode->Leftlink = before; // 새 노드의 왼쪽 링크를 before노드와 연결

newnode->Rightlink = before->Rightlink; //새 노드의 오른쪽 링크를 before노드의 오른쪽 링크가 가리키는 노드와 연결

before->Rightlink->Leftlink = newnode; // before노드의 오른쪽 링크가 가리키는 노드의 왼쪽 링크를 새 노드와 연결

before->Rightlink = newnode; //before노드의 오른쪽 링크를 새 노드와 연결

//printf("left: 0x%p, me: 0x%p, right: 0x%p\n", newnode->Leftlink, newnode, newnode->Rightlink);

}

//마지막 노드에 새로운 노드 삽입

void dinsert\_last(DoubleListNode\* head, element data) {

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

DoubleListNode\* before = head;

newnode->data = data; // 노드 생성

while (before->Rightlink != head) { //노드끝까지 이동

before = before->Rightlink;

}

newnode->Leftlink = before; // 새 노드의 왼쪽 링크를 before노드와 연결

newnode->Rightlink = before->Rightlink; //새 노드의 오른쪽 링크를 before노드의 오른쪽 링크가 가리키는 노드와 연결

before->Rightlink->Leftlink = newnode; // before노드의 오른쪽 링크가 가리키는 노드의 왼쪽 링크를 새 노드와 연결

before->Rightlink = newnode; //before노드의 오른쪽 링크를 새 노드와 연결

//printf("left: 0x%p, right: %p\n" , before, before->Rightlink);

}

//원하는 위치의 노드 삭제

void ddelete\_pos(DoubleListNode\* head, int count) {

DoubleListNode\* before = head;

DoubleListNode\* removed = NULL;

for (int i = 0; i < count && before->Rightlink->Rightlink != head; i++) { // before->Rightlink != head 노드의 끝에서 멈추는 코드

before = before->Rightlink; // Leftlink를 사용하면 반대 방향으로 이동

}

removed = before->Rightlink;

if(removed == head) return;

before->Rightlink = removed->Rightlink;

removed->Rightlink->Leftlink = removed->Leftlink;

free(removed);

}

// 메인

int main(void) {

DoubleListNode\* head = NULL;

printf("추가 단계\n");

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

head = dinsert\_first(head, i); // head오른쪽에 insert

//printf("right: 0x%p\n", head->Rightlink);

print\_dlist(head);

}

dinsert\_pos(head, 44, 10); //10번째 노드(마지막 위치)에 44를 insert

print\_dlist(head);

dinsert\_last(head, 10); // 마지막 노드에 10 insert

print\_dlist(head);

ddelete\_pos(head, 2); // 2번째 노드 삭제

print\_dlist(head);

return 0;

}

1. 실행화면

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 고찰

insert함수와 delete함수는 기존의 방식과 유사하여 구현하는데 큰 어려움이 없었습니다. 하지만 강의 중 이중 포인터에 관한 내용이 나와 당황했습니다. 기존의 방식이 아닌 이중 포인터를 이용하여 insert함수와 다른 함수들도 연습을 할 필요성을 느꼈습니다.