**LinkedList의 ADT**

5645640 제갈건 0401

1. 실행코드

#include<stdio.h>

#include <stdlib.h>

#include<malloc.h>

typedef struct ListNode {

int data = 0; // 실제 자료, size = 전체크기

struct ListNode\* link = NULL; // 다음 노드를 가리키는데 사용하는 포인터

}ListNode;

//맨 처음에 노드 삽입(head에 삽입?)

ListNode\* insert\_first(ListNode\* head, int value) {

ListNode\* p = (ListNode\*)malloc(sizeof(ListNode));

p->data = value; // p 생성

p->link = head; // head가 가리키는 값을 p로 전환

head = p; //p의 주소를 head에 연결

return head; // head리턴

}

//pre뒤에 새로운 노드 삽입(pre를 모른다는 문제점이 있음, 우리는 head만 알고있다.)

ListNode\* insert(ListNode\* head, ListNode\* pre, int value) {

ListNode\* p = (ListNode\*)malloc(sizeof(ListNode));

p->data = value;

p->link = pre->link; //pre가 가리키는 값을 p로 전환(pre가 가리키는 다음 값을 p에 연결)

pre->link = p; //p의 주소를 pre에 연결(pre가 가리키는 다음 값을 p로 전환)

return head; // pre가 head면 head값이 바뀌므로 head 리턴

}

void insert\_last(ListNode\* head, int value) {

ListNode\* p2= head;

ListNode\* lst = (ListNode\*)malloc(sizeof(ListNode));

lst->data = value;

for (ListNode\* p = head; p != NULL; p = p->link) {

p2 = p; // 마지막 노드 들고오기

}

p2->link = lst;

lst->link = NULL;

/\*while (p->link != null) {

\* p = p->link;

\* }

\* p2 = p;

\*/

}

ListNode\* delete\_first(ListNode\* head) {

//empty체크 필요

ListNode\* removed;

if (head == NULL) {

printf("NULL : No element in the list!!\n");

exit(0);

}

removed = head;

head = removed->link;

free(removed);

return head;

}

void Delete\_last(ListNode\* head) {

//empty체크 필요

if (head == NULL) {

printf("NULL : No element in the list!!\n");

exit(0);

}

ListNode\* removed = (ListNode\*)malloc(sizeof(ListNode));

ListNode\* lst = (ListNode\*)malloc(sizeof(ListNode));

for (ListNode\* p = head; p ->link != NULL; p = p->link) {

lst = p; // 마지막의 바로 앞 노드 들고오기

removed = p->link; // 마지막 노드 들고오기

}

lst->link = NULL;

free(removed);

}

void print\_list(ListNode\* head) {

if (head == NULL) {

printf("NULL : No element in the list!!\n");

exit(0);

}

for (ListNode\* p = head; p != NULL; p = p->link) printf("%d->", p->data); // 검색 방식

printf("NULL\n");

// for (ListNode\* p = head; p->data == item; p = p->link) // 데이터를 입력받는 경우

}

bool is\_empty(ListNode\* head) {

return 1;

}

bool is\_full(ListNode\* head) {

return 0;

}

int main() {

ListNode\* head = NULL;

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

head = insert\_first(head, i\*10);

print\_list(head);

}

insert\_last(head, 60);

print\_list(head);

for (int i = 0; i < 4; i++) {

head = delete\_first(head);

print\_list(head);

}

Delete\_last(head);

print\_list(head);

return 0;

}

1. 실행화면

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 고찰

Insert\_last()와 delete\_last() 메서드는 쉽게 구현할 수 있었습니다. 하지만 delete\_last()를 실행한 이후의 검증 작업에서 막혔습니다. 이전에는 delete\_last()를 실행한 후 print\_list()를 호출하면 마지막 노드에서 쓰레기 값이 출력되었습니다. 이후 긴 시간 고민 끝에 마지막 노드의 앞 노드를 가리키는 방법을 찾아냈습니다.

for (ListNode\* p = head; p != NULL; p = p->link) //아래의 방식으로 변형

for (ListNode\* p = head; p->link != NULL; p = p->link)

마지막 노드의 앞 노드를 가져와 해당 노드의 link를 NULL로 바꾸어 해결하였습니다.