자료구조
구현(C언어)

01

프로젝트 포인트 키워드

KEYWORD 1

노드에 대한 이해

KEYWORD 2

노드의 연결에 대한 이해

KEYWORD 3

본격 구현
(C언어)

06

5단 목록형 레이아웃

01 노드 삽입

02 노드 삭제

03 노드 뒤집기

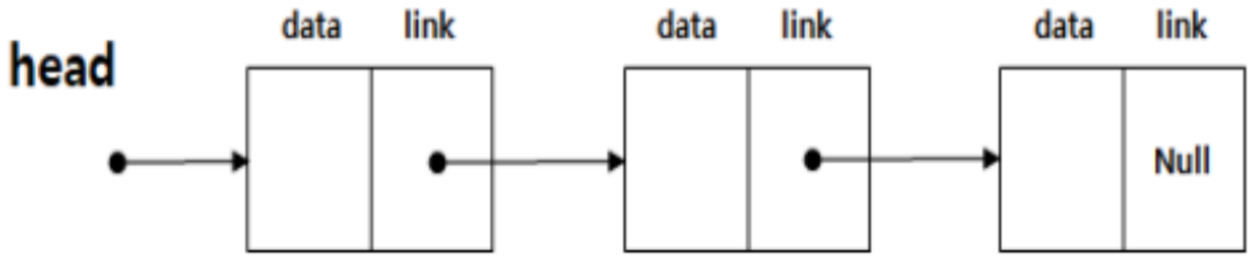
04 노드 오름차순 정렬

05 노드 할당 해제

노드의 구조체 선언

```
typedef struct Node {
    int data;
    struct Node* link;
}Node;
```

struct {data, link} Node



01

노드 삽입

```
void NODE_INSERT(Node** header, int n) { //앞에 삽입
    Node* pi = (Node*)malloc(sizeof(Node));
    if (pi==NULL) {
        printf("메모리 할당 실패₩n");
        return;
    }
    pi->data = n;
    pi->link = *header;
    *header = pi;
    printf("%d 삽입 완료! ₩n", n);
}
```

KEY POINTS

동적 할당(MEMORY-ALLOCATION)

malloc : 메모리의 부족으로 실패할 가능성이 있다.
실패를 어떻게 아는가? malloc함수는
메모리를 사용할 수 없으면 NULL 값을 반환한다.
pi->data : pi라는 포인터변수가 가리키는 구조체의 data값에 n대입
pi->link : pi라는 포인터변수가 가리키는 구조체의 link에 head에 NULL값 저장
head 는 첫 노드 삽입 시 NULL값임. (전역변수로 설정 해둠.)
그리고 head를 pi가 가리키는 노드를 가리키게 함.

02

노드 삭제

```
void NODE_DELETE(int m) {  //문제 발생 1,2,3,4,5를 입력하고 어떤 수를 단 하나 건너뛰고 삭제하면 이전 수도 삭제
    //ex)1 2 3 4 5 삭제:2 > 3 4 5 출력
    Node* temp =head;
    Node* prev = NULL; //제거하고 연결을 위해 이전 노드를 기억할 변수가 필요 null일때는?

    while (temp->data != m) {//삭제할 노드 찾는 반복문
        prev = temp;
        temp = temp->link;
    } //찾으면 제거 하고 연결 해야 하는데???? 안에서는 하면 안되고
    if (temp == NULL) { //값이 없을 경우
        printf("%d 값을 찾을 수 없습니다.₩n", m);
        return;
    }
    if (prev == NULL) { //삭제할 노드가 첫 번째 일 때 while문을 건너뛰기 때문에 prev은 null값임 아직.
        head = temp->link; //head부터 옮기고 free 해야 함
    }
    else {
        prev->link = temp->link; //첫번째가 아니면 전 노드의 링크와 삭제할 노드의 링크를 연결
    }
    free(temp);
    printf("%d 값을 지닌 노드 제거 완료 !₩n", m);
}
```

KEY POINTS

초기화 값 중요

삭제할 값을 입력받고 그 값을 지닌 노드를 찾아 삭제하고 연결하는 함수

temp는 head에 prev는 NULL로 초기화 하는 것이 좋았다.

temp는 값을 비교하며 다음 노드로 넘어가고

prev는 temp이전 노드를 기억하며 한 칸 전에서 따라간다.

1. 찾고자 하는 값이 없을 경우 temp는 NULL에 도착해 있을 것이다.
2. 삭제할 노드가 첫 번째 일 때 head를 옮기고 free(temp) > 순서 중요
   순서를 반대로 하면 head가 찾아갈 수 없음. (논리적으로 실제는 모름)
3. 첫 번째가 아니면 prev의 링크와 삭제할 노드의 링크를 연결 후 free

while문 : 값이 일치하지 않음을 확인하고 temp는 temp->link를 타고
다음 노드로 향하고 prev는 temp가 가리키는 노드를 가리킴.

03

노드 뒤집기

```
void NODE_REVERSE() {
    Node* p = head;
    Node* q = NULL;
    Node* r = NULL;
    while (p != NULL) {
        q = p->link;
        p->link = r;
        r = p;
        p = q;
    }
    head = r;
}
```

KEY POINTS

순서 매우 중요

q : 현재 진행 중인 노드 (떼어낸 노드 하나)

p : q를 떼어내고 남은 노드의 헤드

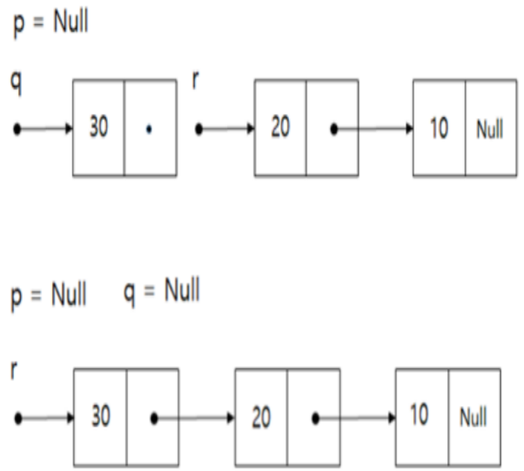
r : 역순 리스트의 헤드

03

노드 뒤집기

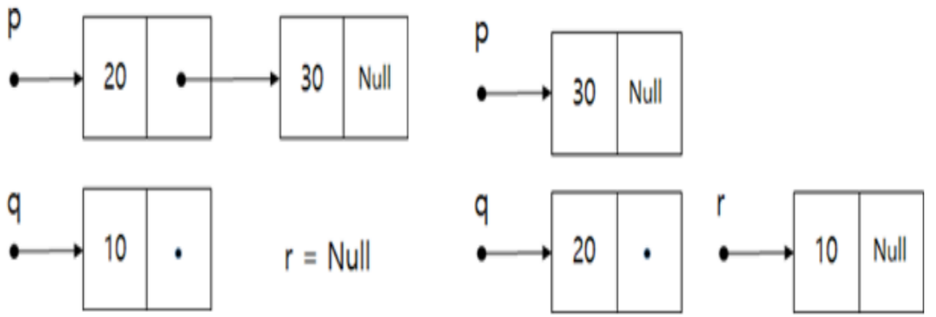
4 문제 해결을 위한 솔루션

- 문제 해결 과정



4 문제 해결을 위한 솔루션

- 문제 해결 과정



04

노드 출력

```
void NODE_PRINT(Node* header) {
    //머리에 연결해서 첫 노드를 찾음.
    Node* current = header;
    while (current != NULL) {
        printf("%d  -> ", current->data);
        current = current->link;
    }
    printf("NULL₩n");//while문 끝나고 연결 돼 있는 마지막 주소 == null
}
```

current 라는 구조체포인터를 만들고 current가 null이 아닐 때 까지 찾아가는 함수

current가 가리키는 구조체의 data를 출력하고 link를 타고 들어가서
다음 노드 다음 노드 다음 노드 출력

05

노드 메모리 할당 해제

```
void NODE_FREE(Node** header, Node** header2) {
    ***

    Node* temp = *header;
    Node* next = *header; //head를 그냥 사용해서 메모리 낭비를 줄 일 수 있음. 논의사항
    while (temp != NULL) {//temp->link !=NULL을 하면 마지막 남은 노드가 할당 해제 안 됨.
        next = temp->link;
        free(temp); //temp는 next이전 노드에 있으니 할당 해제 하고 next 따라가게
        temp = next;
    }
    temp = *header2;
    while (temp != NULL) {
        next = temp->link;
        free(temp);
        temp = next;
    }
    printf("모든 노드 할당 해제 성공 !₩n");
}
```

KEY POINTS

PREV (X) NEXT (O)

prev은 이전 노드를 기억하고 next는 다음 노드를 기억한다.

NODE_FREE()는 temp라는 변수가 노드들을 해제하며 next를 따라간다.

밑의 while문은 무시

44 과제를 똑바로 읽지 않은 자

기존 리스트의 오름차순 정렬

cur, next, asc, temp, prev라는 5개의 포인터가 필요..

asc : 오름차순 정렬할 리스트의 헤드포인터

temp, prev : asc를 따라가며 값을 비교하고 넣는 포인터

cur, next : 정렬 전 리스트를 기억하고 떼내는 포인터

2개의 오름차순 정렬된 리스트를 합병정렬

미구현

더블 포인터가 필요??!!

감사합니다


Microsoft Visual Studio 디버그
입력할 값 : 15
15 삽입 완료!
1: 삽입, 2: 삭제, 3: 뒤집기, 4: 출력, 5: 오름차순, 6:종료
선택 : 1
입력할 값 : 5
5 삽입 완료!
1: 삽입, 2: 삭제, 3: 뒤집기, 4: 출력, 5: 오름차순, 6:종료
선택 : 1
입력할 값 : 3
3 삽입 완료!
1: 삽입, 2: 삭제, 3: 뒤집기, 4: 출력, 5: 오름차순, 6:종료
선택 : 1
입력할 값 : 7
7 삽입 완료!
1: 삽입, 2: 삭제, 3: 뒤집기, 4: 출력, 5: 오름차순, 6:종료
선택 : 4
7 -> 3 -> 5 -> 15 -> 9 -> NULL
1: 삽입, 2: 삭제, 3: 뒤집기, 4: 출력, 5: 오름차순, 6:종료
선택 : 3
9 -> 15 -> 5 -> 3 -> 7 -> NULL
1: 삽입, 2: 삭제, 3: 뒤집기, 4: 출력, 5: 오름차순, 6:종료
선택 : 5
오름차순 정렬 완료 !
3 -> 5 -> 7 -> 9 -> 15 -> NULL
1: 삽입, 2: 삭제, 3: 뒤집기, 4: 출력, 5: 오름차순, 6:종료
선택 : 6
모든 노드 할당 해제 성공 !
C:\Users\psw01\source\repos\Project3\x64\Debug\Project3.exe(프로세스 10836)이(가) 0 코드(0x0)와 함께 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요...
7,3,5,15,9 입력
출력
뒤집기
오름차순
할당 해제
100%