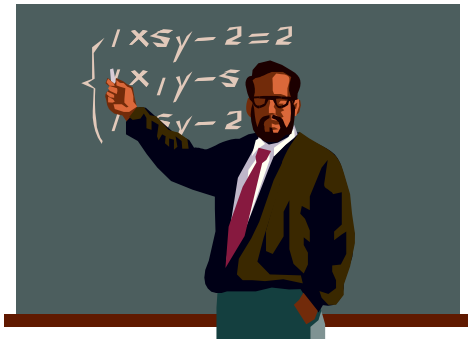
7장 연결 리스트 II

Q & A

- 언제라도 질문하세요
  모르면 외우면 되지만 코딩 안되면 꼭 질문
  ➔ 기말 오픈북 코딩시험, 인터넷차단
  1. 한성e-class 질의응답게시판
  2. 온라인 코딩 라운지 (자세한 내용은 공지사항 게시판)
- 강사의 1번 선생님은 여러분들의 질문
- 매주 정답이 있는 연습문제 홀수문제 꼭 해결해보세요



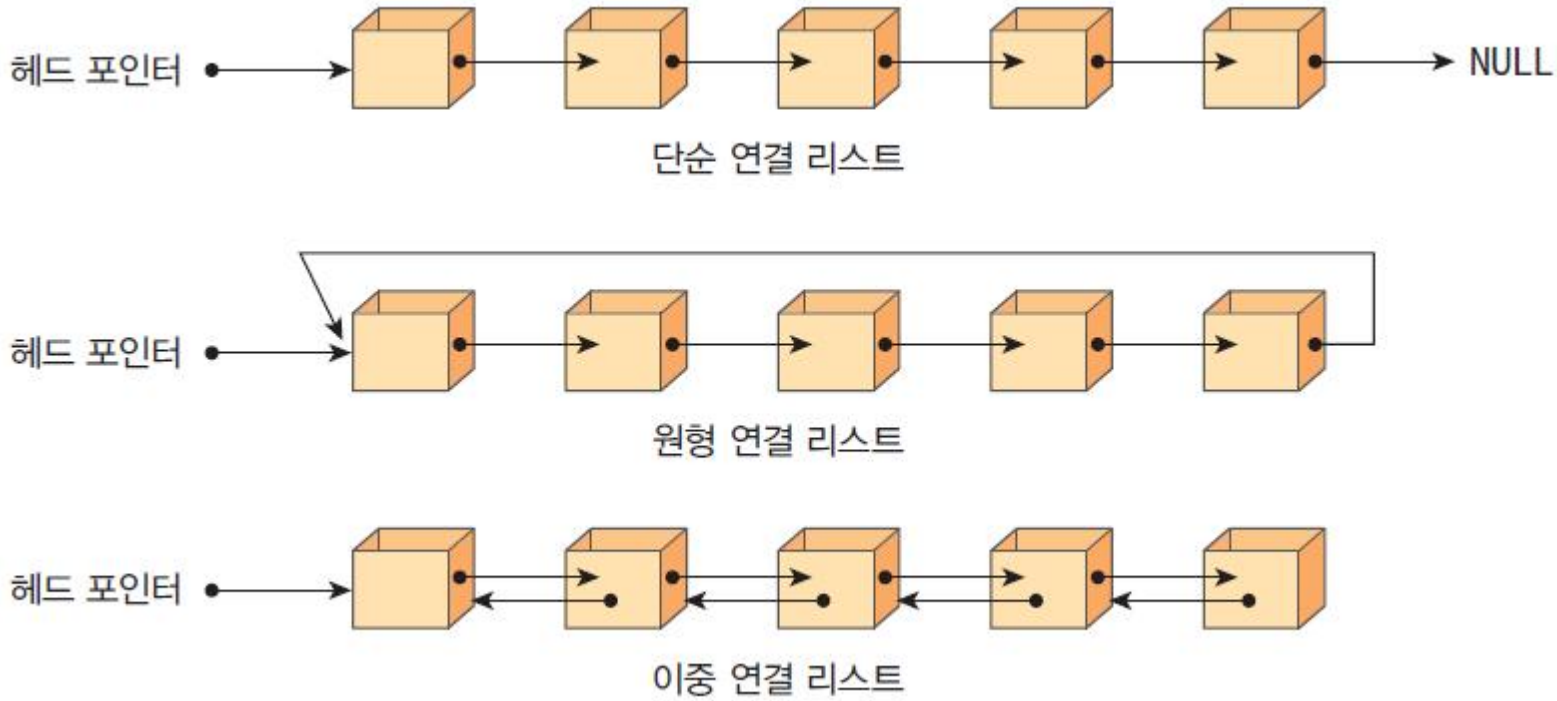
앞으로 나는 할 수 있다 (7장 목표)

1. 나는 리스트 자료구조를 원형연결리스트로 구현할 수 있다
2. 나는 리스트 자료구조를 이중연결리스트로 구현할 수 있다.

3. 나는 스택 자료구조를 연결 리스트로 구현할 수 있다
4. 나는 큐 자료구조를 연결 리스트로 구현할 수 있다

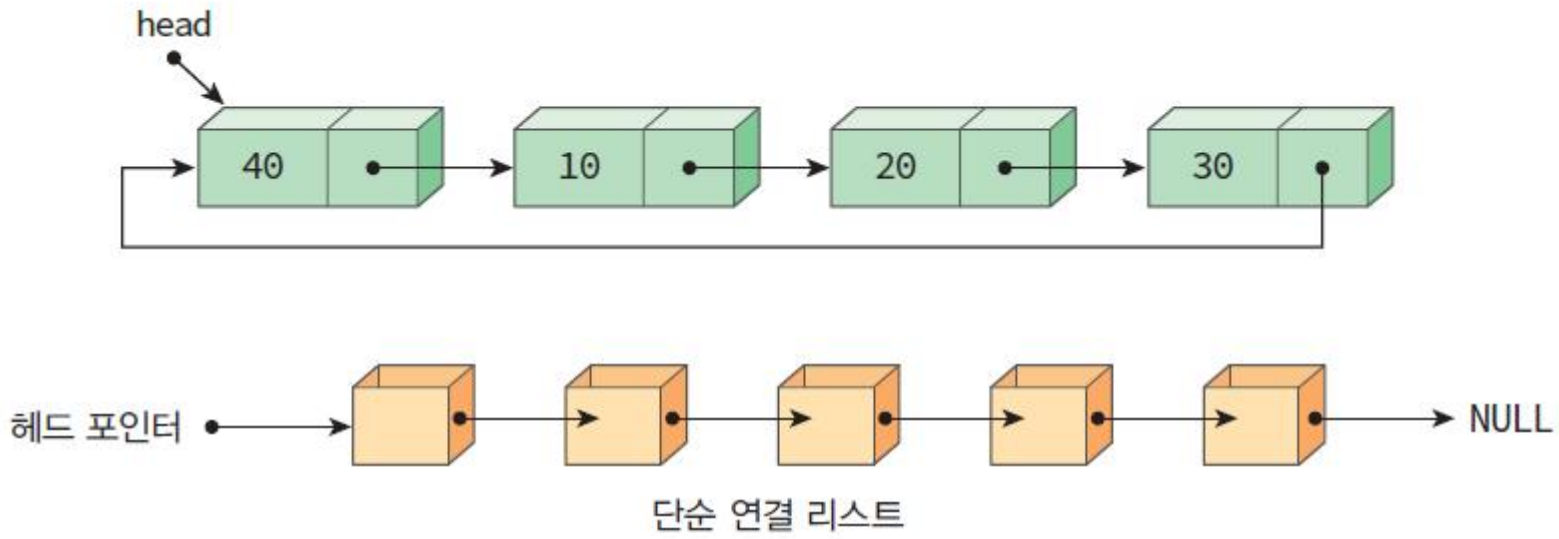
- 리스트와 연결리스트와의 관계는 햄과 햄스터와의 관계
  아무런 관계가 없음
- 리스트 자료구조는 배열 또는 연결리스트로 구현할 수 있다
- 리스트는 n개의 element로 구성된 순서 있는 모임
  예: My To-Do list, My bucket list

연결 리스트의 종류 3가지



원형 연결 리스트

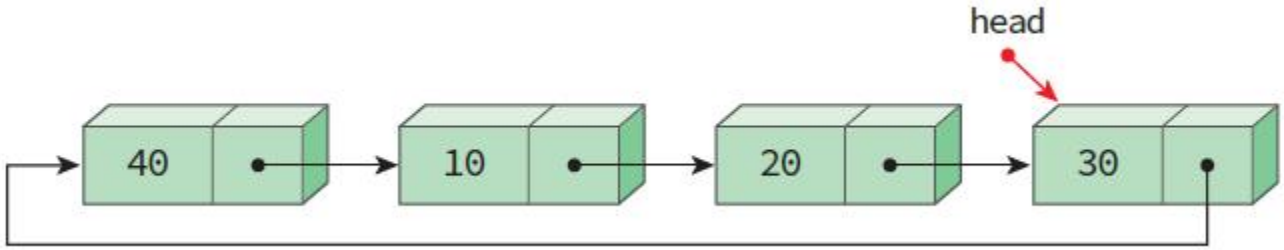
- 마지막 노드의 링크가 첫 번째 노드를 가리키는 리스트
- 한 노드에서 다른 모든 노드로의 접근이 가능
- 단순연결리스트는 항상 헤드포인트에서 시작해야 함



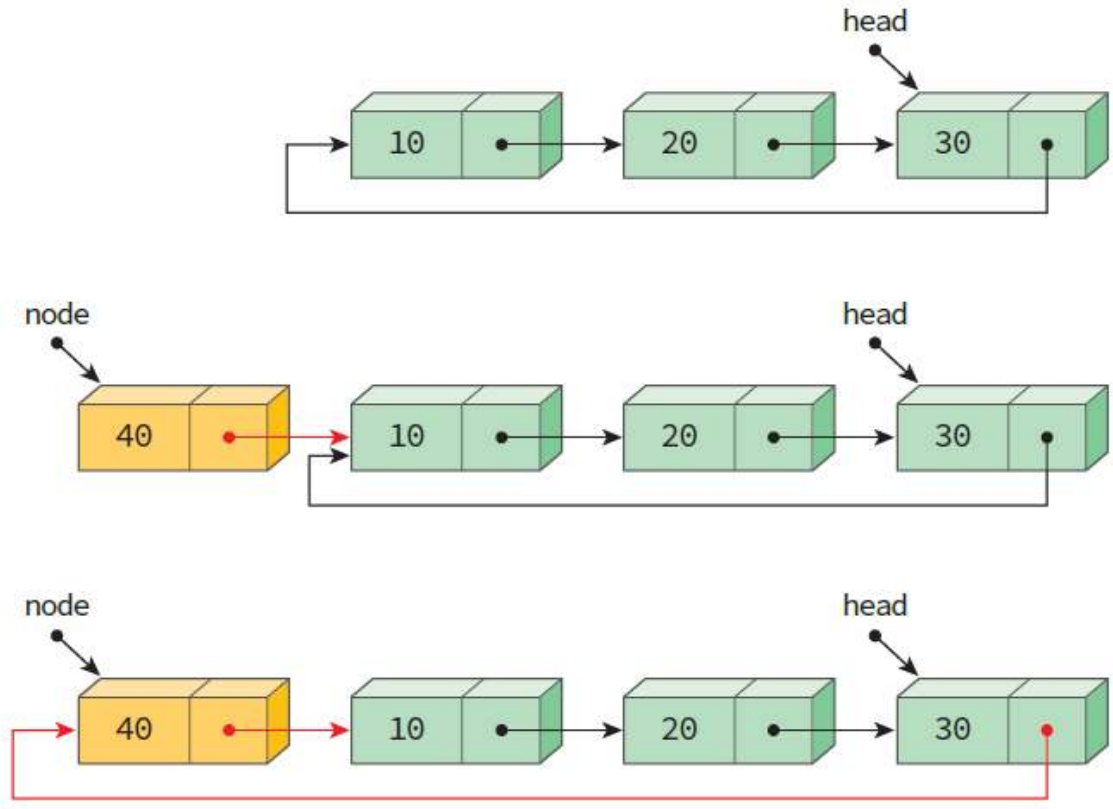
단순 연결 리스트

원형 연결 리스트

- 보통 헤드포인터가 마지막 노드를 가리키게끔 구성하면 리스트의 처음이나 마지막에 노드를 삽입하는 연산이 단순 연결 리스트에 비하여 용이



원형 연결 리스트의 처음에 삽입 insert_first()



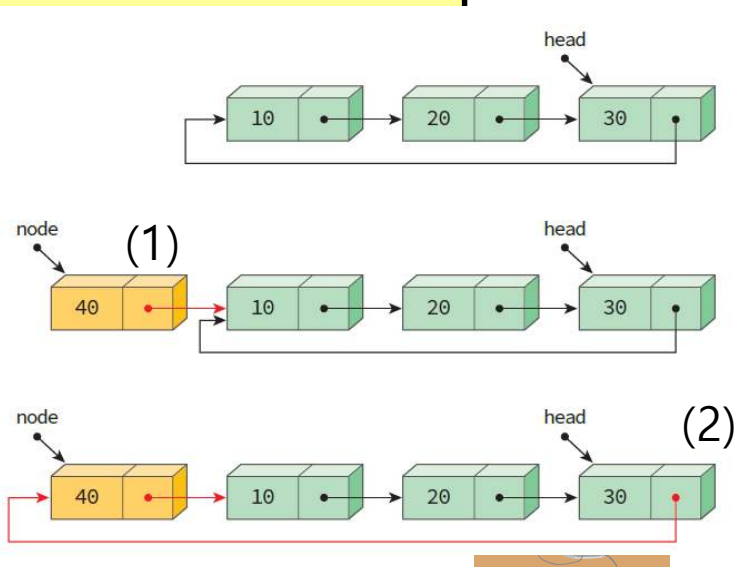
원형 연결 리스트의 처음에 삽입

학생> 교수님~ 코드 보아도 이해가 되지 않습니다.

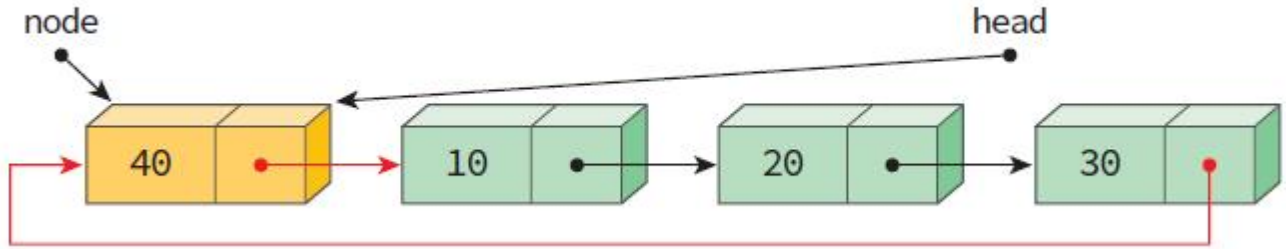
```
ListNode* insert_first(ListNode* head, element data)
{
        ListNode *node = (ListNode *)malloc(sizeof(ListNode));
        node->data = data;
        if (head == NULL) {
                head = node;
                node->link = head;
        }
        else {
                node->link = head->link;        // (1)
                head->link = node;              // (2)
        }
        return head;// 변경된 헤드 포인터를 반환한다.
}
```

원형 연결 리스트의 처음에 삽입

```
ListNode* insert_first(ListNode* head, element data)
{
        ListNode *node = (ListNode *)malloc(sizeof(ListNode));
        node->data = data;
        if (head == NULL) {
                head = node;
                node->link = head;
        }
        else {
                node->link = head->link;        // (1)
                head->link = node;              // (2)
        }
        return head;// 변경된 헤드 포인터를 반환한다.
}
```



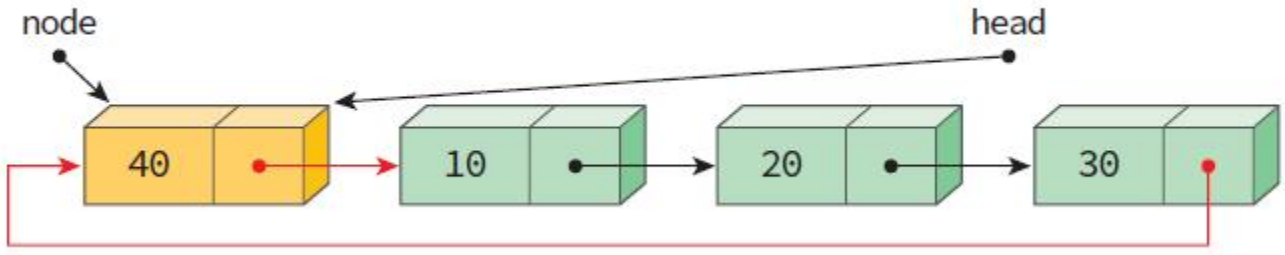
리스트의 끝에 삽입



리스트의 끝에 삽입 (insert_last())

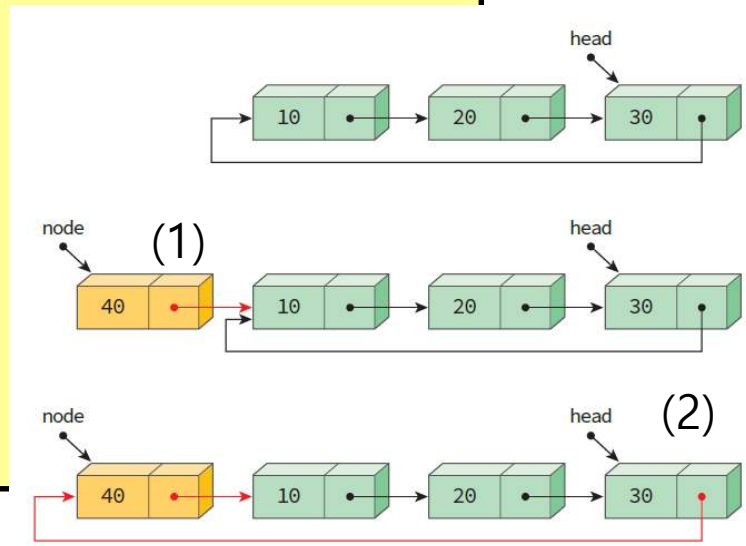
학생> 교수님~ (3)단계 그림만 있어요. (1) (2)단계 그림은?

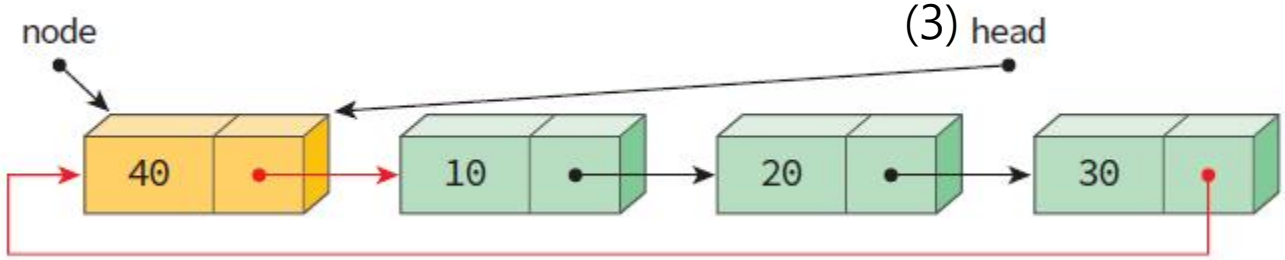
```
ListNode* insert_last(ListNode* head, element data)
{
        ListNode *node = (ListNode *)malloc(sizeof(ListNode));
        node->data = data;
        if (head == NULL) {
                head = node;
                node->link = head;
        }
        else {
                node->link = head->link;// (1)
                head->link = node;      // (2)
                head = node;            // (3)
        }
        return head;// 변경된 헤드 포인터를 반환한다.
}
```



리스트의 끝에 삽입 (insert_last())

```
ListNode* insert_last(ListNode* head, element data)
{
        ListNode *node = (ListNode *)malloc(sizeof(ListNode));
        node->data = data;
        if (head == NULL) {
                head = node;
                node->link = head;
        }
        else {
                node->link = head->link;// (1)
                head->link = node;  // (2)
                head = node;        // (3)
        }
        return head;// 변경된 헤드 포인터를 반환한다.
}
```





테스트 프로그램

```
#include <stdio.h>
#include <stdlib.h>

typedef int element;
typedef struct ListNode {      // 노드 타입
        element data;
        struct ListNode *link;
} ListNode;

// 리스트의 항목 출력
void print_list(ListNode* head)
{
        ListNode* p;

        if (head == NULL) return;
        p = head->link;
        do {
                printf("%d->", p->data);
                p = p->link;
        } while (p != head);
        printf("%d->", p->data); // 마지막 노드 출력
}
```

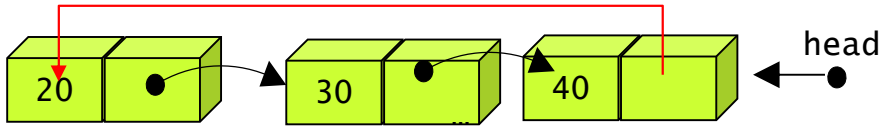
테스트 프로그램

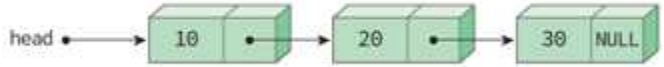
```
int main(void)
{
        ListNode *head = NULL;

        // list = 10->20->30->40
        head = insert_last(head, 20);
        head = insert_last(head, 30);
        head = insert_last(head, 40);
        head = insert_first(head, 10);
        print_list(head);
        return 0;
}
```

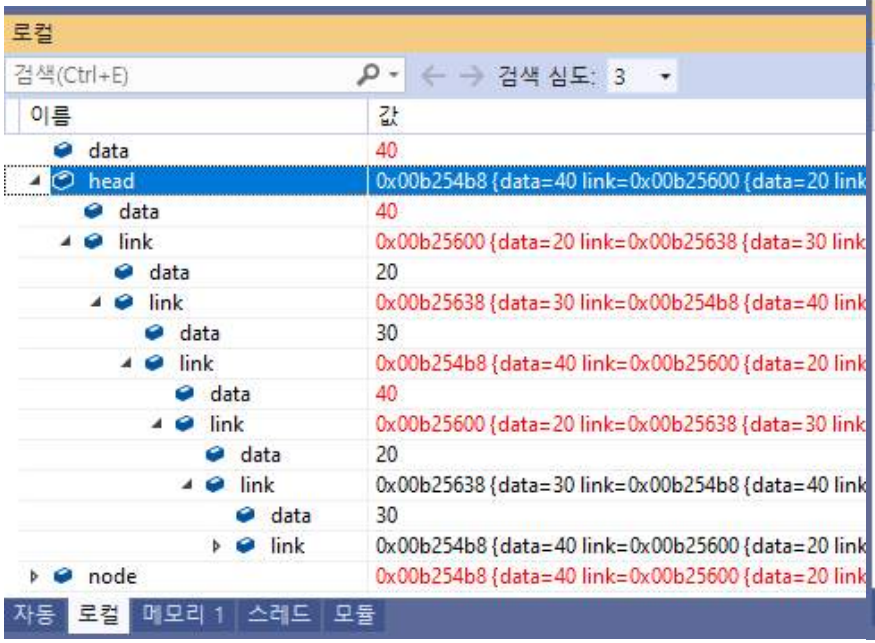
```
10->20->30->40->
```

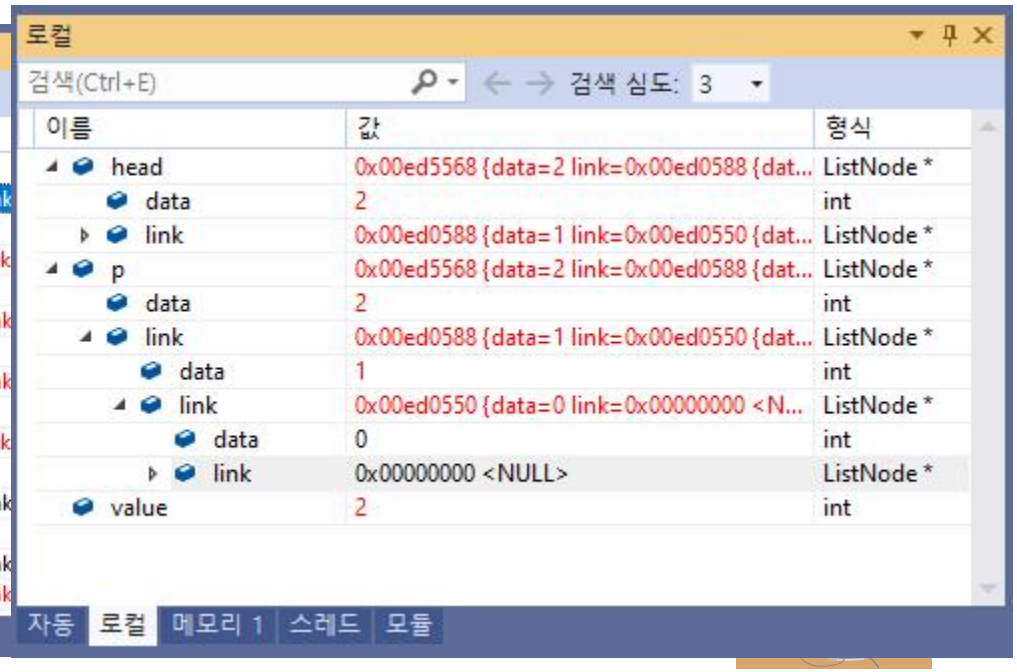
VS로 원형연결리스트 자료구조 보기 (여러줄실행)

- cir_list.c의 52번 라인에 중단점 설정하고 [계속]버튼 3회 하면 3개 노드의 내용(data+link)가 무한반복 표시됨
- 원형연결리스트(무한반복표시)
  단순연결리스트(NULL로 종료)









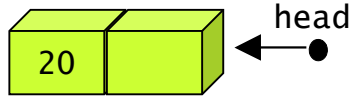
VS로 원형연결리스트 자료구조 보기 (한줄씩실행) insert_last() : 20

- cir_list.c의 43번 라인에 중단점 설정하고 한줄씩 실행

이름	값
data	20
head	0x00000000 <NULL>
data	<메모리를 읽을 수 없음>
link	<메모리를 읽을 수 없음>
node	0x009d5568 {data=20 link=0xcdc

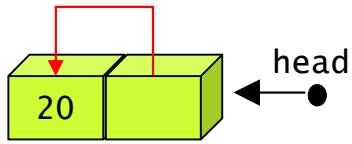
- head = node;

이름	값
data	20
head	0x009d5568 {data=20 link=0xcdcc
data	20
link	0xcdcdcdcd {data=??? link=??? }



- node->link = head;

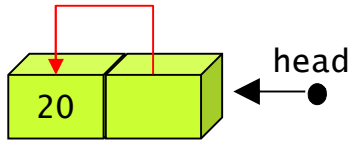
이름	값
data	20
head	0x009d5568 {data=20 link
data	20
link	0x009d5568 {data=20 link



VS로 원형연결리스트 자료구조 보기 (한줄씩실행)

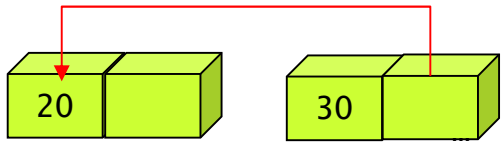
insert_last() : 20 , 30

이름	값
data	20
head	0x009d5568 {data=20 link
data	20
link	0x009d5568 {data=20 link



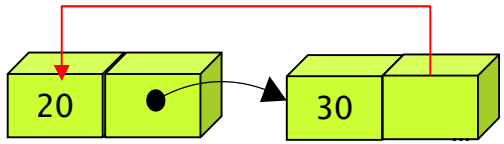
- node->link = head->link;

이름	값
data	30
head	0x009d5568 {da
data	20
link	0x009d5568 {da
data	20
link	0x009d5568 {da
node	0x009d55a0 {da
data	30
link	0x009d5568 {da



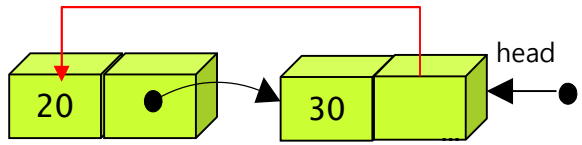
- head->link = node;

이름	값
data	30
head	0x009d5568 {da
data	20
link	0x009d55a0 {da
data	30
link	0x009d5568 {da
node	0x009d55a0 {da
data	30
link	0x009d5568 {da



- head = node;

이름	값
data	30
head	0x009d55a0 {da
data	30
link	0x009d5568 {da
data	20
link	0x009d55a0 {da
node	0x009d55a0 {da
data	30
link	0x009d5568 {da



학생> 교수님~ VS를 왜 보나요?

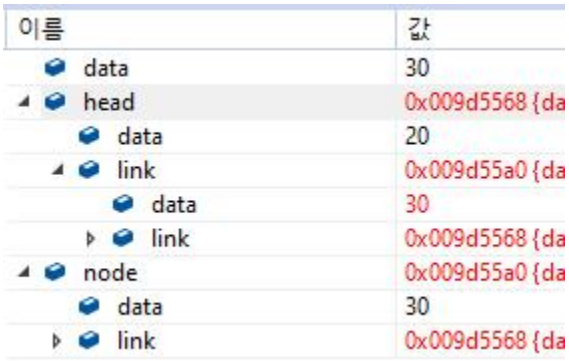
교수> VS로 코드 한 줄 씩 수행하면서 자료구조 변화를 볼 수 있습니다.

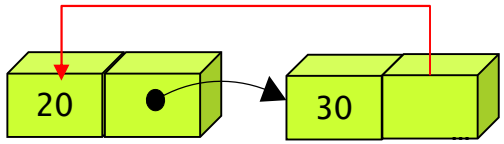
학생> 하지만 VS는 자료구조의 값(숫자)만 보여주어서 이해하기 어려워요

교수> 그래서 여러분이 자료구조 그림을 그리면 이해하기 쉽습니다.

천인국교수님은 교재에서 단계별 자료구조 그림을 많이 보여 주었기에 이제 여러분이 필요한 그림을 만들면 코드 이해하기 쉽습니다.

- head->link = node;





원형 연결 리스트의 응용: 멀티 플레이어 게임

- 현재 차례가 누구인가?

```
현재 차례=KIM
현재 차례=CHOI
현재 차례=PARK
현재 차례=KIM
현재 차례=CHOI
현재 차례=PARK
현재 차례=KIM
현재 차례=CHOI
현재 차례=PARK
현재 차례=KIM
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef char element[100];
typedef struct ListNode {       // 노드 타입
        element data;
        struct ListNode *link;
} ListNode;


ListNode* insert_first(ListNode* head, element data)
{
        ListNode *node = (ListNode *)malloc(sizeof(ListNode));
        strcpy(node->data, data);
        if (head == NULL) {
                head = node;
                node->link = head;
        }
        else {
                node->link = head->link;        // (1)
                head->link = node;              // (2)
        }
        return head;        // 변경된 헤드 포인터를 반환한다.
}
```

멀티 플레이어 게임 (3사람이 원형으로 무한 반복)

```
// 원형 연결 리스트 테스트 프로그램
int main(void)
{
        ListNode *head = NULL;

        head = insert_first(head, "KIM");
        head = insert_first(head, "PARK");
        head = insert_first(head, "CHOI");

                ListNode* p = head;
        for (int i = 0; i < 10; i++) {
                printf("현재 차례=%s \n", p->data);
                p = p->link;
        }
        return 0;
}
```

학생> 교수님~ warning 발생했어요

1. strcpy()에 클릭한후 F1
2. string.h 추가
3. strcpy_s () 변경
4. strcpy_s()에 클릭한 후 F1
5. strcpy_s(node->data, _countof(node->data), data); 변경

교수> 야호~ warning 0

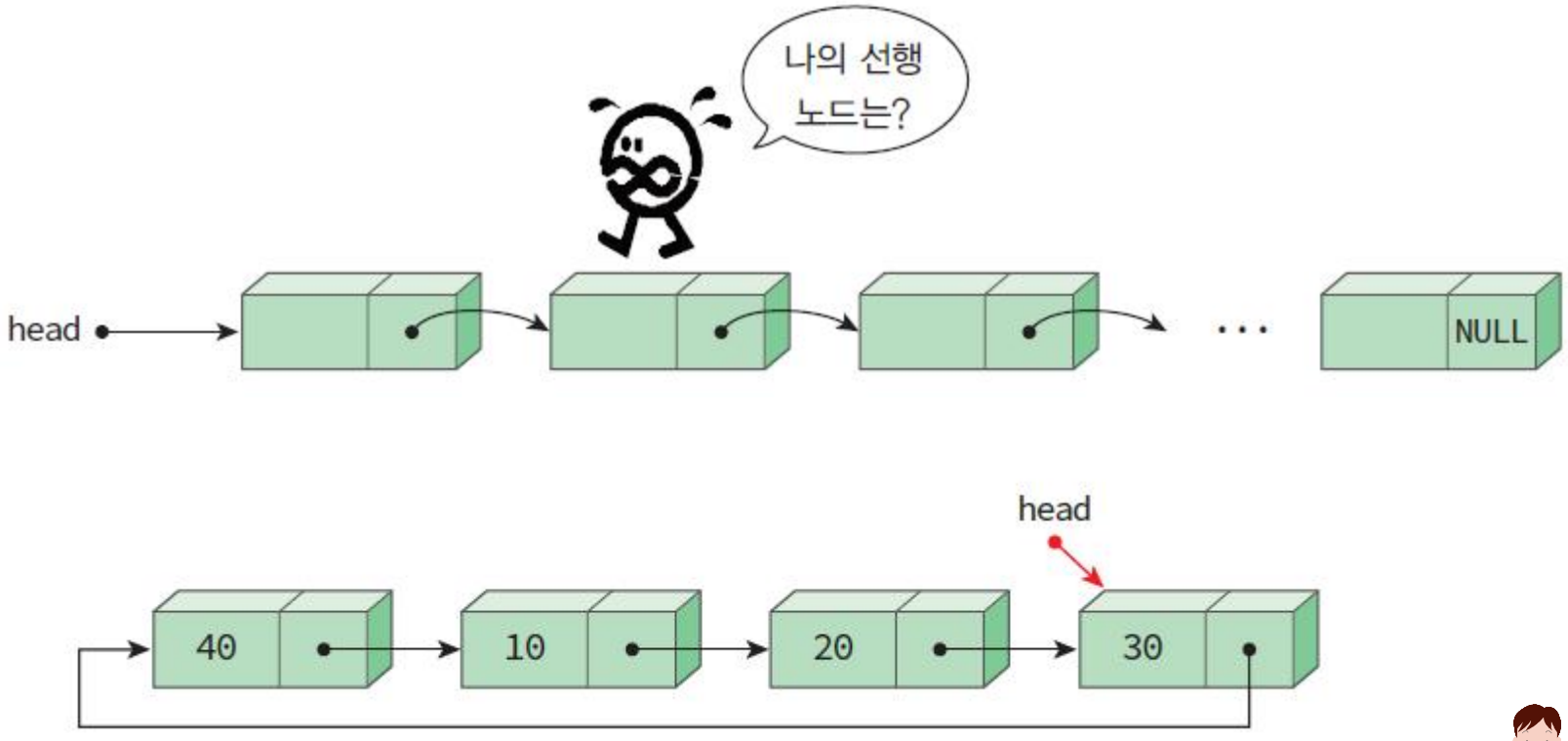
휴식

1. 나는 리스트 자료구조를 원형연결리스트로 구현할 수 있다

2. 나는 리스트 자료구조를 이중연결리스트로 구현할 수 있다.
3. 나는 스택 자료구조를 연결 리스트로 구현할 수 있다
4. 나는 큐 자료구조를 연결 리스트로 구현할 수 있다

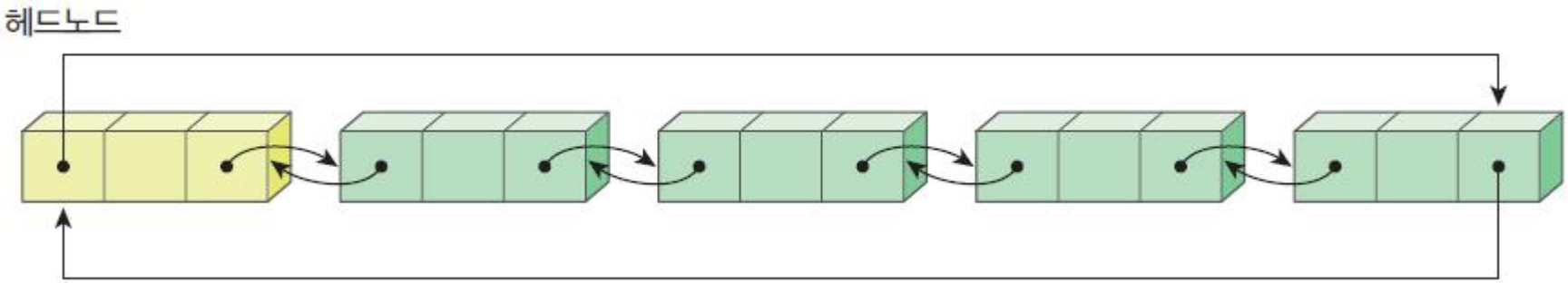
이중 연결 리스트

- 단순 연결 리스트의 문제점: 선행 노드를 찾기가 힘들다
  원형 연결 리스트도 선행 노드 찾기 힘들다

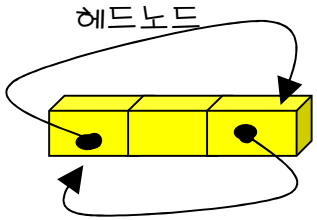


이중 연결 리스트

- 이중 연결 리스트: 하나의 노드가 선행 노드와 후속 노드에 대한 두 개의 링크를 가지는 리스트
- 단점은 공간을 많이 차지하고 코드가 복잡



- 헤드노드(head node): 데이터를 가지지 않고 단지 삽입, 삭제 코드를 간단하게 할 목적으로 만들어진 노드
  - 헤드 포인터와의 구별 필요
  - 공백상태에서는 헤드 노드만 존재

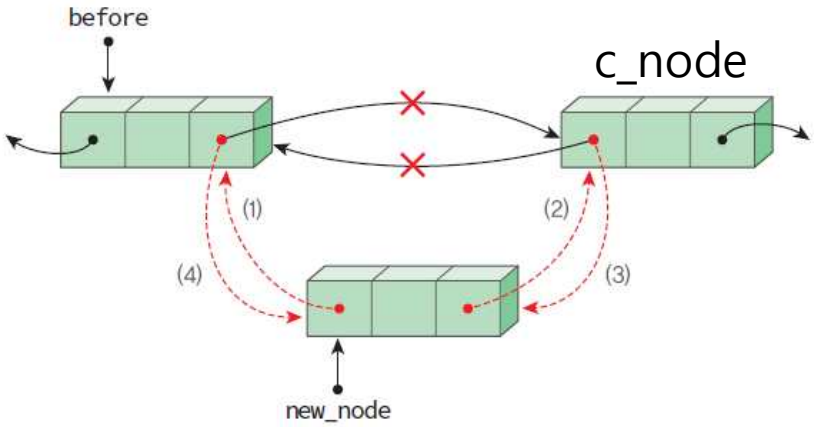


노드의 구조

- 이중연결리스트에서의 노드의 구조

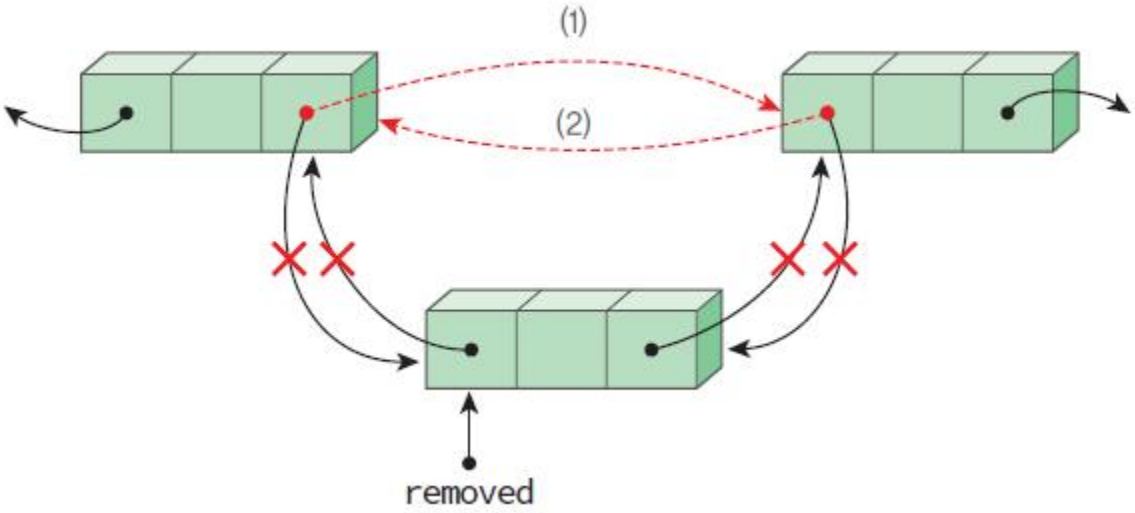
```
typedef int element;
typedef struct DlistNode {
        element data;
        struct DlistNode *llink;
        struct DlistNode *rlink;
} DlistNode;
```

삽입연산




```
// 새로운 데이터를 노드 before의 오른쪽에 삽입한다.
void dinsert(DListNode *before, element data)
{
        DListNode *newnode = (DListNode *)malloc(sizeof(DListNode));
        strcpy(newnode->data, data);
        newnode->llink = before; // (1)
        newnode->rlink = before->rlink; // (2)
        before->rlink->llink = newnode; // (3)
        before->rlink = newnode; // (4)
}
```

삭제연산




```
// 노드 removed를 삭제한다.
void ddelete(DListNode* head, DListNode* removed)
{
        if (removed == head) return;
        removed->llink->rlink = removed->rlink; // (1)
        removed->rlink->llink = removed->llink; // (2)
        free(removed);
}
```

테스트 프로그램

```
#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;
}
```

테스트 프로그램

```
// 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));
        strcpy(newnode->data, data);
        newnode->llink = before;
        newnode->rlink = before->rlink;
        before->rlink->llink = newnode;
        before->rlink = newnode;
}
```

테스트 프로그램

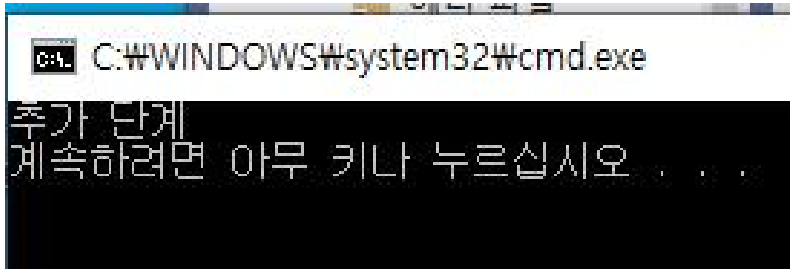
```
// 노드 removed를 삭제한다.
void ddelete(DListNode* head, DListNode* removed)
{
        if (removed == head) return;
        removed->llink->rlink = removed->rlink;
        removed->rlink->llink = removed->llink;
        free(removed);
}
```

테스트 프로그램

```
// 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;
}
```

실행 결과 (교수님~ 결과화면이 이상해요)

```
추가 단계
<-| |0| |->
<-| |1| |-> <-| |0| |->
<-| |2| |-> <-| |1| |-> <-| |0| |->
<-| |3| |-> <-| |2| |-> <-| |1| |-> <-| |0| |->
<-| |4| |-> <-| |3| |-> <-| |2| |-> <-| |1| |-> <-| |0| |->

삭제 단계
<-| |4| |-> <-| |3| |-> <-| |2| |-> <-| |1| |-> <-| |0| |->
<-| |3| |-> <-| |2| |-> <-| |1| |-> <-| |0| |->
<-| |2| |-> <-| |1| |-> <-| |0| |->
<-| |1| |-> <-| |0| |->
<-| |0| |->
```

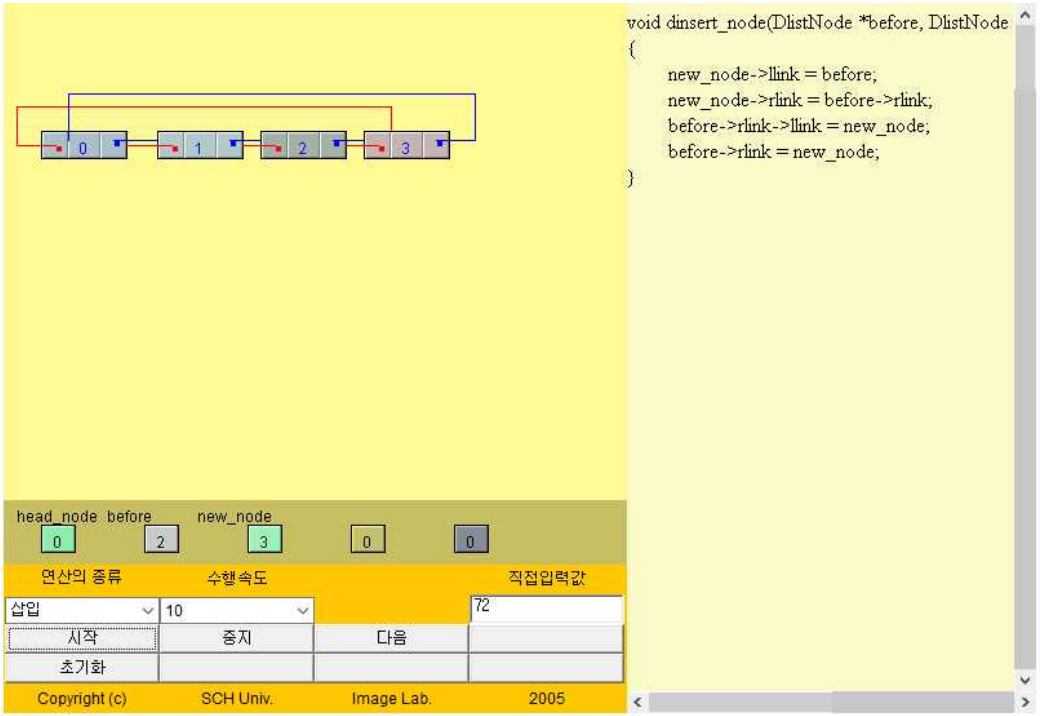


학생> 교수님~ 결과화면이 이상해요

1. data는 int 타입이기에 strcpy() 를 사용하면 안됨
2. 교재 234페이지에 있는대로 newnode->data = data; 변경

이중 연결 리스트 자바 애플릿

- 그림 방식의 자바 애플릿 vs. (학생이 그린 자료구조그림)과 VS



<연산>

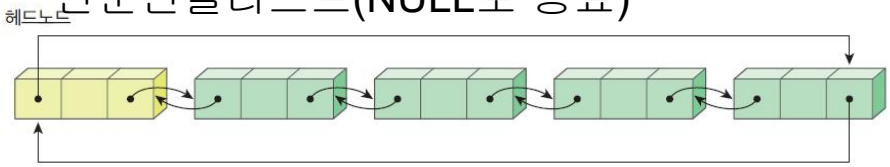
- dinsert(DListNode *before, element data)
- ddelete(DListNode* head, DListNode* removed)

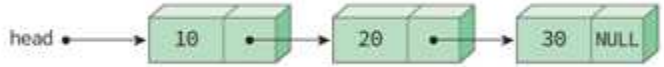
<자료구조>

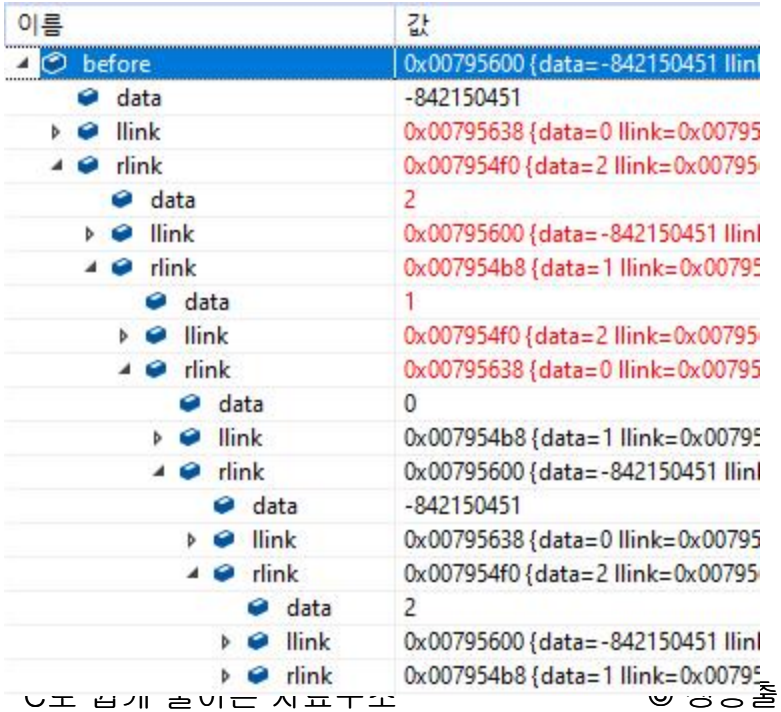
```
typedef struct DListNode {
  element data;
  struct DListNode* llink;
  struct DListNode* rlink;
} DListNode;
```

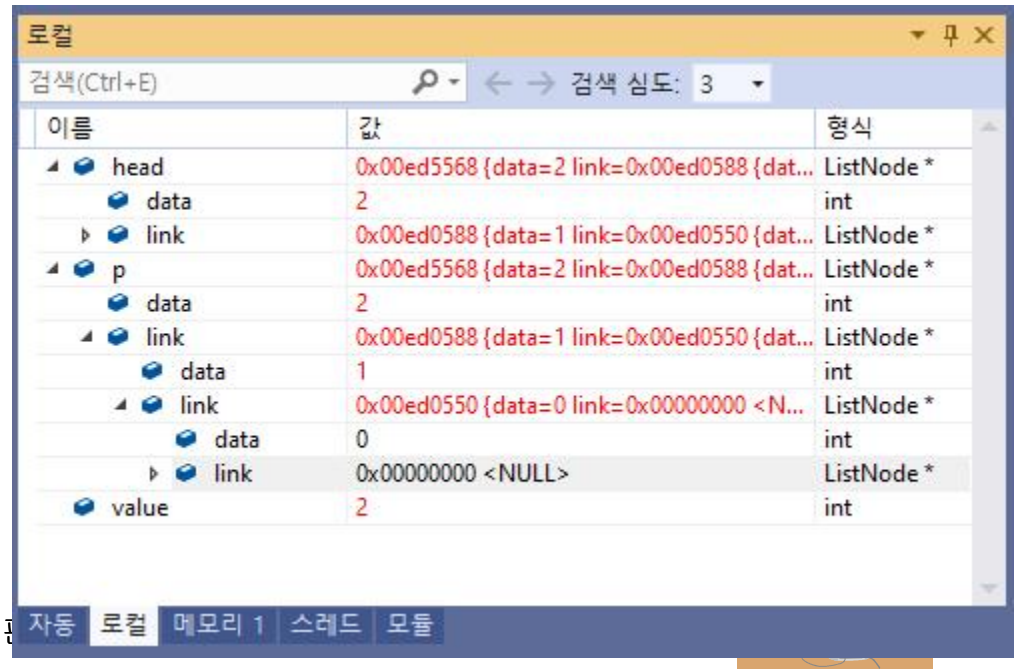
VS로 이중연결리스트 자료구조 보기 (여러줄실행)

- dlinkedlist.c의 36번 라인에 중단점 설정하고 [계속]버튼 3회하면 3개 노드의 내용(data+link)가 무한반복 표시됨
- 이중연결리스트(무한반복표시, left link, right link)
  단순연결리스트(NULL로 종료)



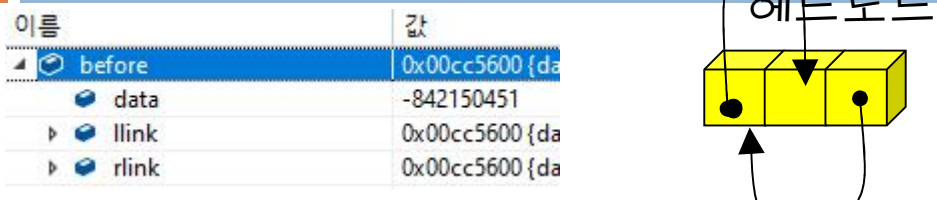




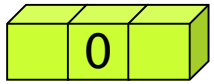


VS로 이중연결리스트 자료구조 보기 (한줄씩실행)

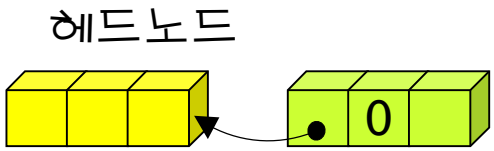
dinsert() : 0 dlinkedlist.c의 31번 라인에 중단점 설정



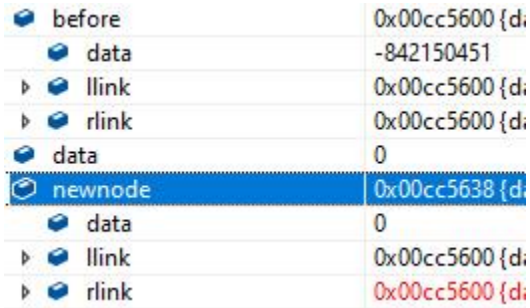
- newnode->data = data;

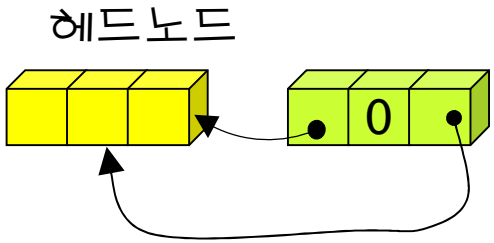


- newnode->llink = before;



- newnode->rlink = before->rlink;

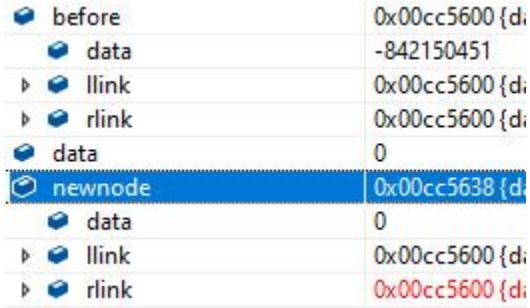


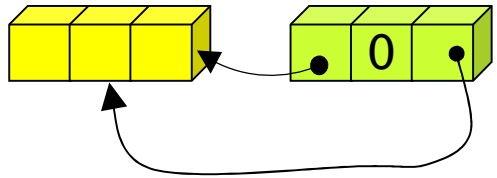


VS로 이중연결리스트 자료구조 보기 (한줄씩실행)

dinsert() : 0

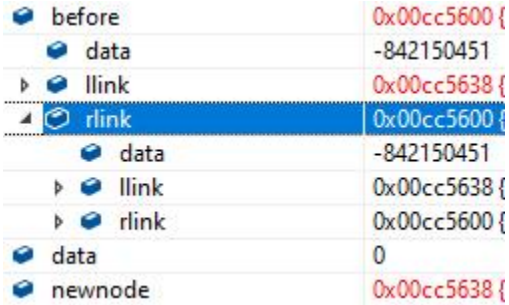
- newnode->rlink = before->rlink;

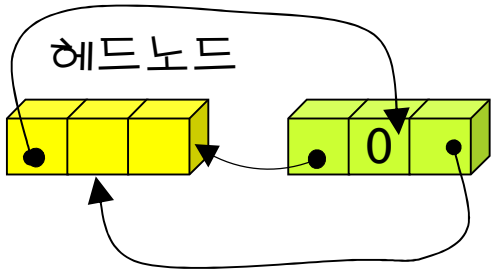




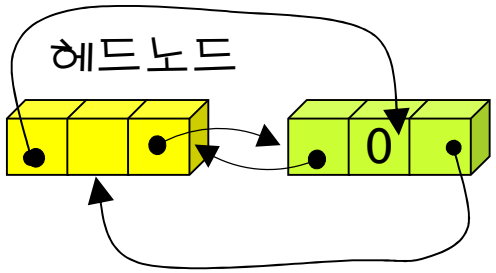
헤드노드

- before->rlink->llink = newnode;





- before->rlink = newnode;



VS로 이중연결리스트 자료구조 보기 (한줄씩실행)

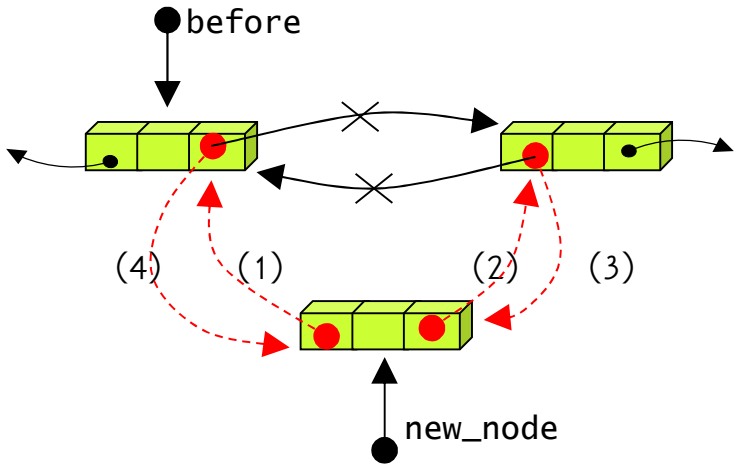
dinsert(DListNode *before, element data)

1. newnode->llink = before;
2. newnode->rlink = before.rlink;
3. before->rlink->llink = newnode
4. before->rlink = newnode;

3단계에서 before->llink로 하지 않고 before->rlink->llink로 한 이유는?

(-> 가 2개나 있음)

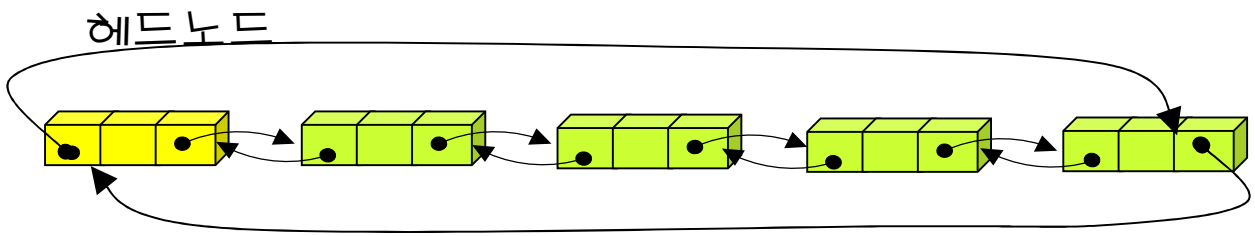
< before의 오른쪽에 insert >

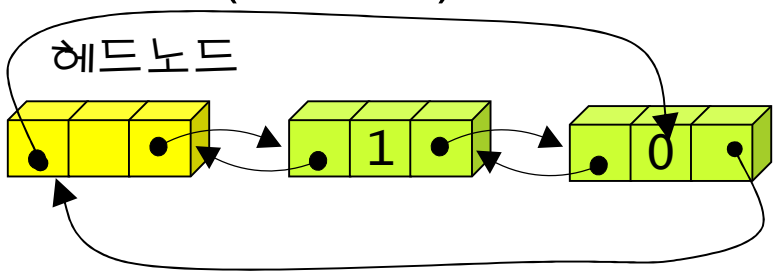


< 프로그램 7.7 >

dinsert (head, 0);

dinsert (head, 1);





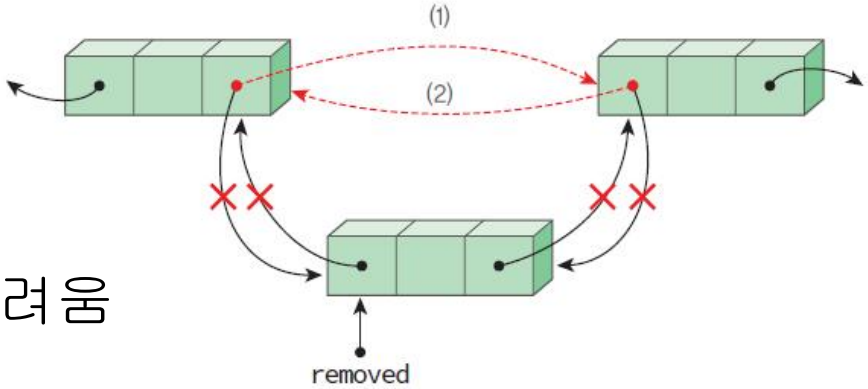
VS로 이중연결리스트 자료구조 보기 (한줄씩실행)

ddelete(DListNode* head, DListNode* removed)

1. removed->llink->rlink = removed->rlink;
2. removed->rlink->llink = removed->llink;

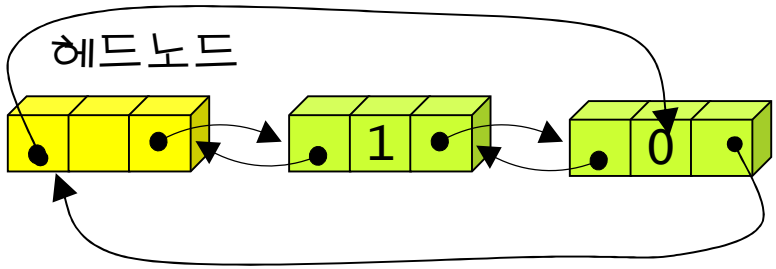
(-> 가 2개나 있음)
그림없이 C코드만으로 이해하기 어려움

< head의 오른쪽 delete >



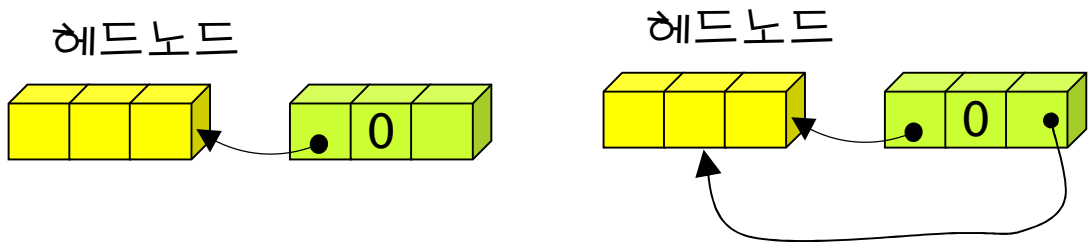
< 프로그램 7.7 >

```
ddelete (head, head->rlink);
  // 1 제거
Ddelete (head, head->rlink);
  // 0 제거
```



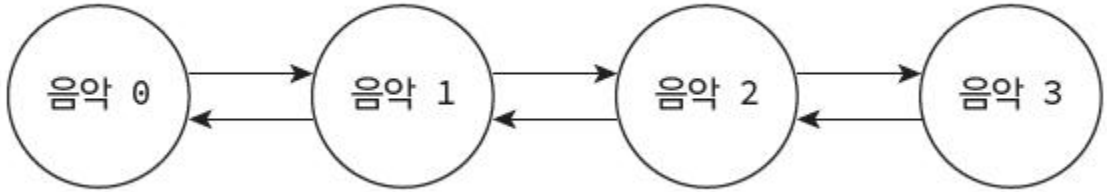
휴식 (이제 나는 할 수 있다)

1. 나는 리스트 자료구조를 원형연결리스트로 구현할 수 있다
2. 나는 리스트 자료구조를 이중연결리스트로 구현할 수 있다.
3. 나는 원형연결리스트와 이중연결리스트 내용변화를 C코드 한줄씩 VS와 내가 그린 자료구조그림으로 보여줄 수 있다



4. 나는 스택 자료구조를 연결 리스트로 구현할 수 있다
5. 나는 큐 자료구조를 연결 리스트로 구현할 수 있다

mp3 재생 프로그램 만들기 (이중 연결리스트)




```
<-| #Fernando# |-> <-| Dancing Queen |-> <-| Mamamia |->

명령어를 입력하시오(<, >, q): >
<-| Fernando |-> <-| #Dancing Queen# |-> <-| Mamamia |->

명령어를 입력하시오(<, >, q): >
<-| Fernando |-> <-| Dancing Queen |-> <-| #Mamamia# |->

명령어를 입력하시오(<, >, q): <
<-| Fernando |-> <-| #Dancing Queen# |-> <-| Mamamia |->

명령어를 입력하시오(<, >, q):
```

학생> 교수님~ 에러나요

학생> 7장 multigame.c에서도 strcpy_s() 고쳐보았기에 이제는 제가 고칠수 있어요.

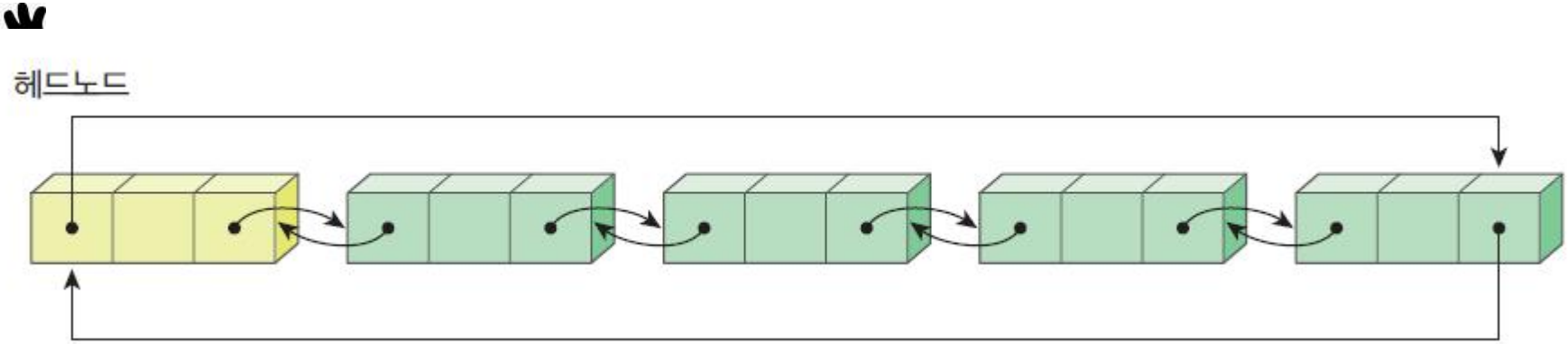
1. strcpy_s(newnode->data, _countof(newnode->data), data); 변경

테스트 프로그램

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

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;
}
```

테스트 프로그램

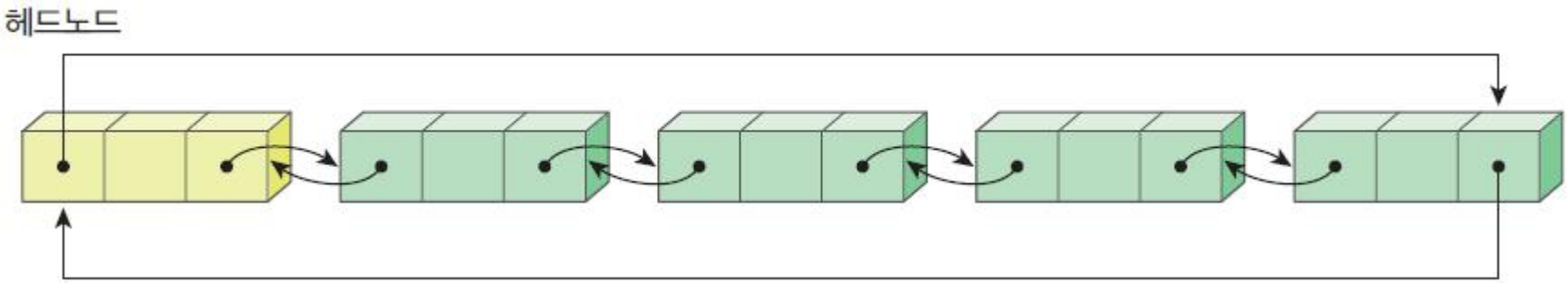
```
// 이중 연결 리스트 테스트 프로그램
int main(void)
{
        char ch;
        DListNode* head = (DListNode *)malloc(sizeof(DListNode));
        init(head);

        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) // 교수님~ 이 라인은 왜 있나요?
                                current = current->llink;
                }
                else if (ch == '>') {
                        current = current->rlink;
                        if (current == head) // 교수님~ 이 라인은 왜 있나요?
                                current = current->rlink;
                }
                print_dlist(head);
                getchar();
        } while (ch != 'q');
  // 동적 메모리 해제 코드를 여기에 (dlinkedlist.c의 삭제단계)
}
```

교수> 자료구조 그림 없이 코드설명 불가능합니다.

교수> 여러분도 코드가 이상하면 자료구조 그림 그리면 됩니다.

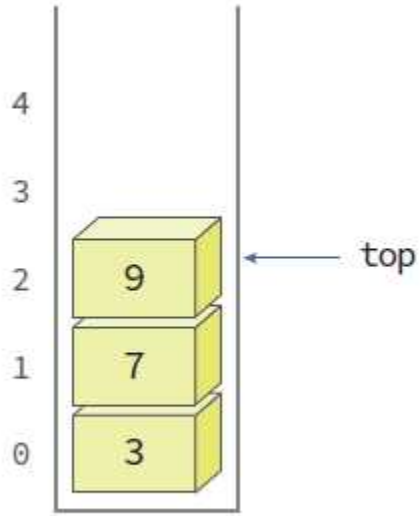



```
do {
        printf("\n명령어를 입력하시오(<, >, q): ");
        ch = getchar();
        if (ch == '<') {
                current = current->llink;
                if (current == head) // 교수님~ 이 라인은 왜 있나요?
                        current = current->llink;
        }
```

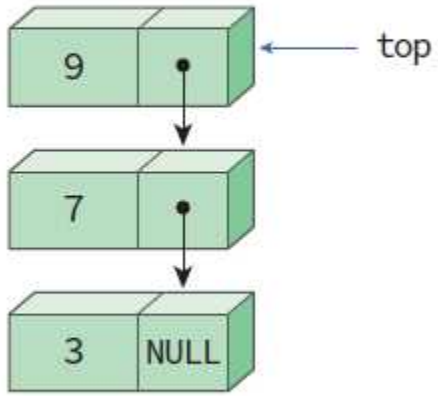
연결 리스트로 구현한 스택 (단순연결리스트)

학생> 교수님~ 그림 보니까 감 잡았습니다.

- **스택 : 후입선출(LIFO:Last-In First-Out)**: 가장 최근에 들어온 데이터가 가장 먼저 나감.
  미로탈출문제에서 스택자료구조 사용하지 않으면 미로탈출 불가능. 무한반복 (살려주세요~~)



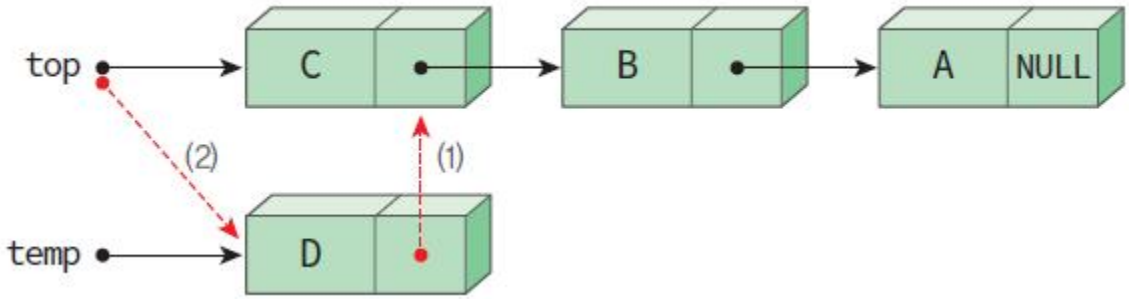
(a) 배열을 이용한 스택



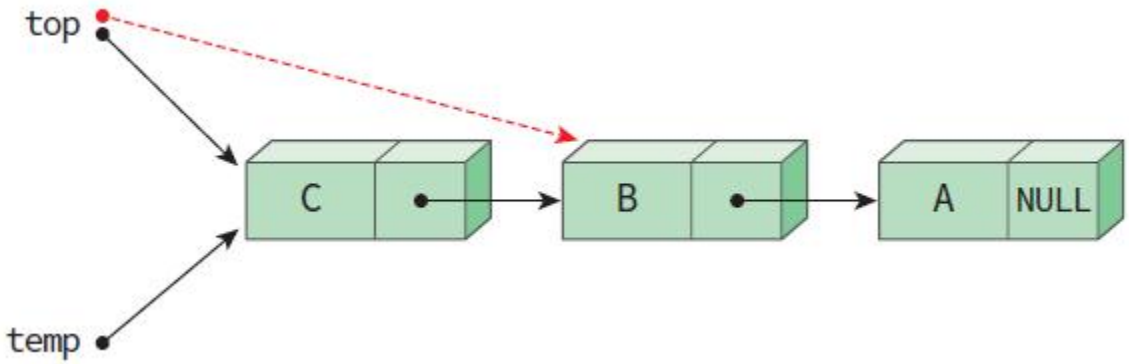
(b) 연결 리스트를 이용한 스택

```
typedef int element;
typedef struct StackNode {
        element data;
        struct StackNode *link;
} StackNode;

typedef struct {
        StackNode *top;
} LinkedStackType;
```

삽입 연산

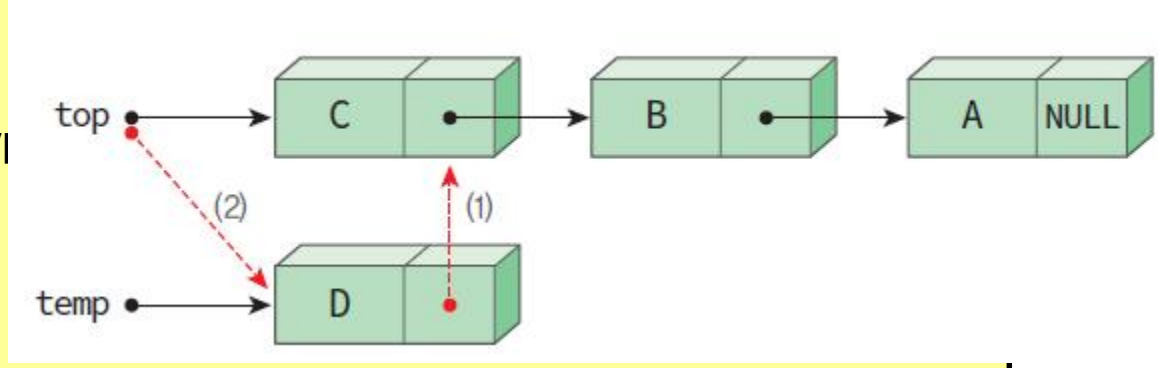


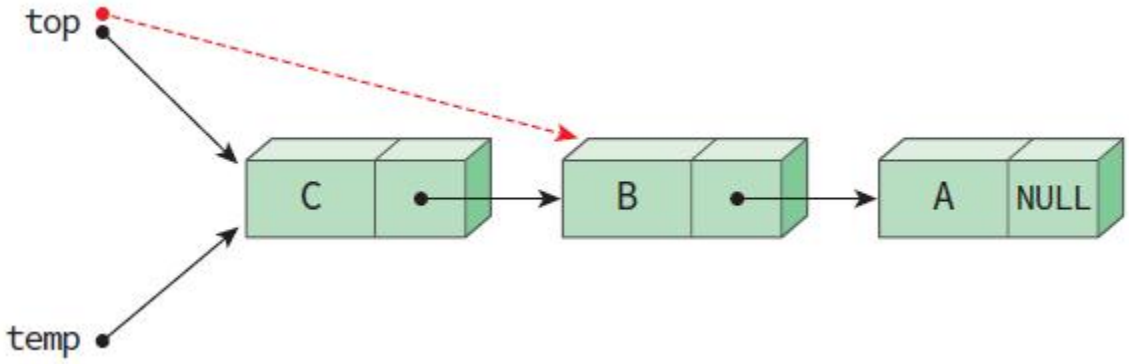
삭제 연산



```
#include <stdio.h>
#include <malloc.h>

typedef int element;
typedef struct StackNode {
        element data;
        struct StackNode *link;
} StackNode;

typedef struct {
        StackNode *top;
} LinkedStackType;
// 초기화 함수
void init(LinkedStackType *s)
{
        s->top = NULL;
}
```

```
// 공백 상태 검출 함수
int is_empty(LinkedStackType *s)
{
        return (s->top == NULL);
}
// 포화 상태 검출 함수
int is_full(LinkedStackTy
{
        return 0;
}
// 삽입 함수
void push(LinkedStackType *s, element item)
{
        StackNode *temp = (StackNode *)malloc(sizeof(StackNode));
        temp->data = item;
        temp->link = s->top; (1)
        s->top = temp; (2)
}
void print_stack(LinkedStackType *s)
{
        for (StackNode *p = s->top; p != NULL; p = p->link)
                printf("%d->", p->data);
        printf("NULL \n");
}
```





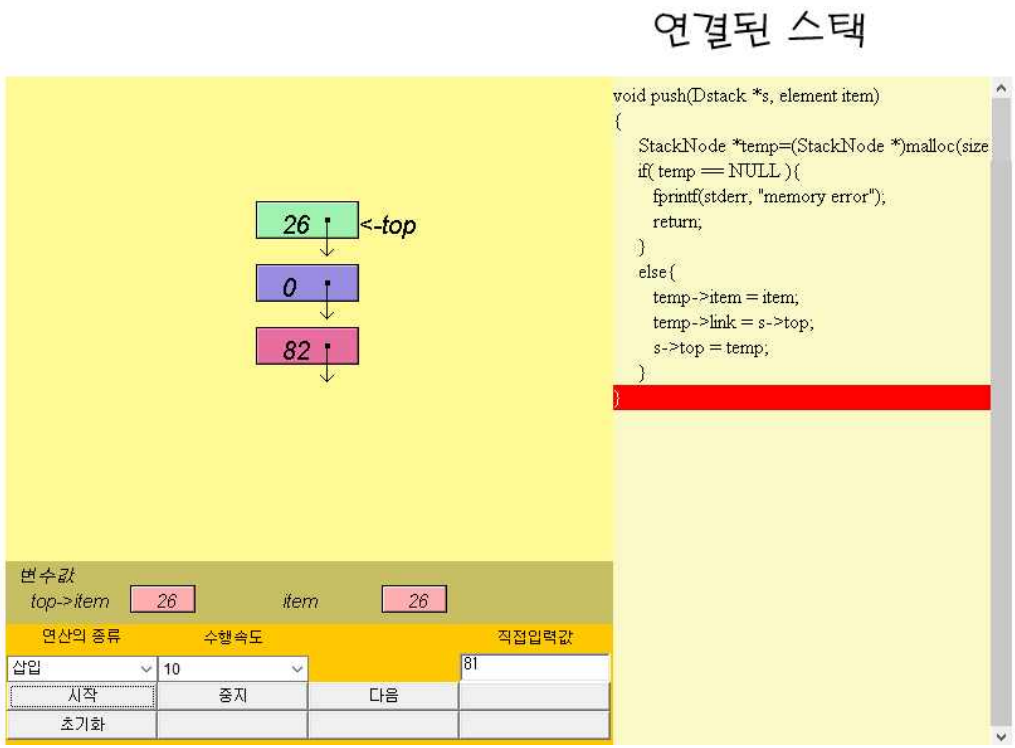

```
// 삭제 함수
element pop(LinkedStackType *s)
{
        if (is_empty(s)) {
                fprintf(stderr, "스택이 비어있음\n");
                exit(1);
        }
        else {
                StackNode *temp = s->top;
                int data = temp->data;
                s->top = s->top->link; (1)
                free(temp);
                return data;
        }
}
```

```
// 주 함수
int main(void)
{
        LinkedStackType s;
        init(&s);
        push(&s, 1); print_stack(&s);
        push(&s, 2); print_stack(&s);
        push(&s, 3); print_stack(&s);
        pop(&s); print_stack(&s);
        pop(&s); print_stack(&s);
        pop(&s); print_stack(&s);
        return 0;
}
```

```
1->NULL
2->1->NULL
3->2->1->NULL
2->1->NULL
1->NULL
NULL
```

연결된 스택 자바 애플릿

- 그림 방식의 자바 애플릿 vs. 텍스트 방식의 VS 디버깅 줄단위로 자료구조와 알고리즘 동시에 봄



<연산>

- void push(LinkedStackType *s, element item)
- element pop(LinkedStackType *s)

<자료구조>

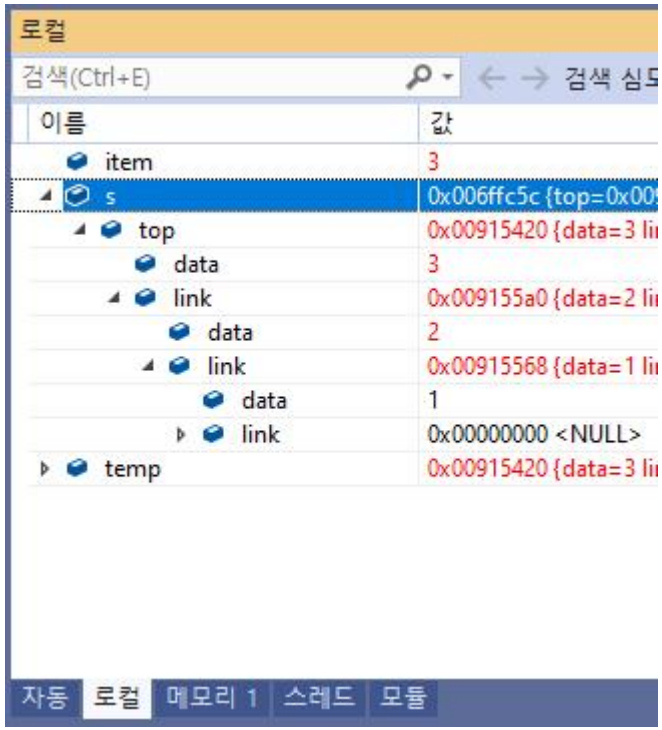
```
typedef int element;
typedef struct StackNode {
  element data;
  struct StackNode *link;
} StackNode;
```

VS로 stack 자료구조 보기

- 6장 연결리스트-1에서 VS로 단순연결리스트 자료구조를 보았음(6장.pdf의 42page)

VS로 stack 자료구조 보기 (여러줄 실행)

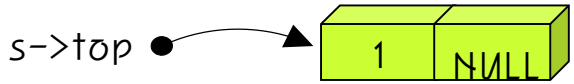
- linked_stack.c의 35번 라인에 중단점 설정하고 [계속]버튼 3회하면 3개 노드의 내용(data+link)가 표시됨



VS로 stack 자료구조 보기 (한줄씩실행) 31줄 중단점

void push(LinkedStackType *s, element item)

이름	값	형식
item	1	int
s	0x0133fcdc {top=0x00000000 <NULL> }	LinkedStackType *
temp	0xcccccccc {data=??? link=??? }	StackNode *
data	<메모리를 읽을 수 없음>	
link	<메모리를 읽을 수 없음>	



- StackNode *temp = (StackNode *)malloc(sizeof(StackNode));

이름	값	형식
item	1	int
s	0x0133fcdc {top=0x00000000 <NULL> }	LinkedStackType *
temp	0x01465600 {data=-842150451 link=0xc...	StackNode *
data	-842150451	int
link	0xcdcdcdcd {data=??? link=??? }	StackNode *

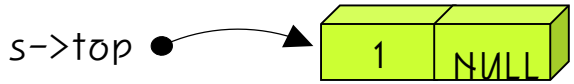
- temp->data = item;

이름	값	형식
item	1	int
s	0x0133fcdc {top=0x00000000 <NULL> }	LinkedStackType *
temp	0x01465600 {data=1 link=0xcdcdcdcd {...	StackNode *
data	1	int
link	0xcdcdcdcd {data=??? link=??? }	StackNode *

VS로 stack 자료구조 보기 (한줄씩실행)
void push(LinkedStackType *s, element item)

- temp->link = s->top

이름	값	형식
item	1	int
s	0x0133fcdc {top=0x00000000 <NULL> }	LinkedStackType *
temp	0x01465600 {data=1 link=0x00000000 <...	StackNode *
data	1	int
link	0x00000000 <NULL>	StackNode *



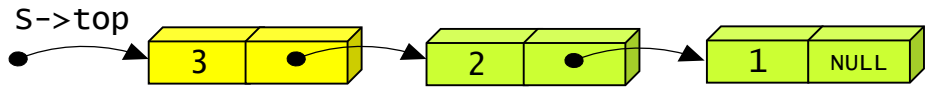
- s->top = temp;

이름	값	형식
item	1	int
s	0x0133fcdc {top=0x01465600 {data=...	LinkedStackType *
top	0x01465600 {data=1 link=0x0000000...	StackNode *
data	1	int
link	0x00000000 <NULL>	StackNode *
temp	0x01465600 {data=1 link=0x0000000...	StackNode *
data	1	int
link	0x00000000 <NULL>	StackNode *

VS로 linked list 자료구조 보기 (한줄씩실행) 50줄 중단점

element pop(LinkedStackType *s)

이름	값	형식
data	-858993460	int
s	0x006ff840 {top=0x00a254b8 {data=...	LinkedStackType *
top	0x00a254b8 {data=3 link=0x00a2563...	StackNode *
data	3	int
link	0x00a25638 {data=2 link=0x00a2560...	StackNode *
temp	0xcccccccc {data=??? link=??? }	StackNode *



- StackNode *temp = s->top

s	0x006ff840 {top=0x00a254b8 {data=...	LinkedStackType *
top	0x00a254b8 {data=3 link=0x00a2563...	StackNode *
data	3	int
link	0x00a25638 {data=2 link=0x00a2560...	StackNode *
temp	0x00a254b8 {data=3 link=0x00a2563...	StackNode *
data	3	int
link	0x00a25638 {data=2 link=0x00a2560...	StackNode *

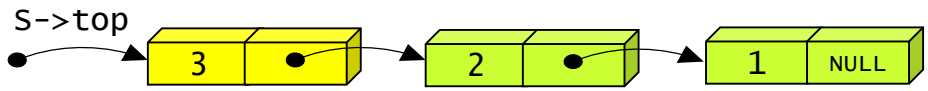
- int data = temp->data;

data	3	int
s	0x006ff840 {top=0x00a254b8 {data=...	LinkedStackType *
top	0x00a254b8 {data=3 link=0x00a2563...	StackNode *
data	3	int
link	0x00a25638 {data=2 link=0x00a2560...	StackNode *
temp	0x00a254b8 {data=3 link=0x00a2563...	StackNode *
data	3	int
link	0x00a25638 {data=2 link=0x00a2560...	StackNode *

VS로 linked list 자료구조 보기 (한줄씩실행)

element pop(LinkedStackType *s)

data	3	int
s	0x006ff840 {top=0x00a254b8 {data=...	LinkedStackType *
top	0x00a254b8 {data=3 link=0x00a2563...	StackNode *
data	3	int
link	0x00a25638 {data=2 link=0x00a2560...	StackNode *
temp	0x00a254b8 {data=3 link=0x00a2563...	StackNode *
data	3	int
link	0x00a25638 {data=2 link=0x00a2560...	StackNode *



- s->top = s->top->link

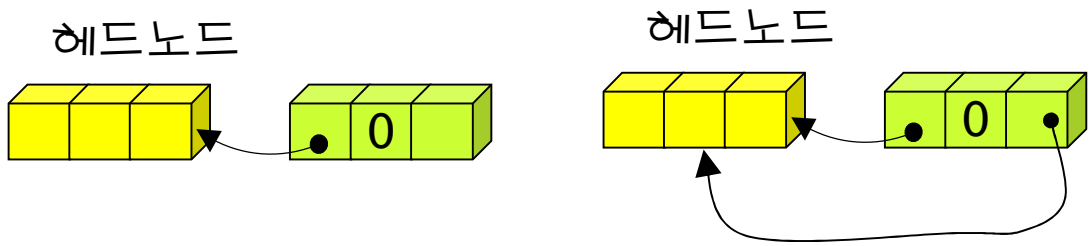
s	0x006ff840 {top=0x00a25638 {data=...	LinkedStackType *
top	0x00a25638 {data=2 link=0x00a2560...	StackNode *
data	2	int
link	0x00a25600 {data=1 link=0x0000000...	StackNode *

- free(tmp);

temp	0x00a254b8 {data=0 link=0x0000000...	StackNode *
data	0	int
link	0x00000000 <NULL>	StackNode *

휴식 (이제 나는 할 수 있다)

1. 나는 리스트 자료구조를 원형연결리스트로 구현할 수 있다
2. 나는 리스트 자료구조를 이중연결리스트로 구현할 수 있다.
3. 나는 원형연결리스트와 이중연결리스트 내용변화를 C코드 한줄씩 VS와 내가 그린 자료구조그림으로 보여줄 수 있다

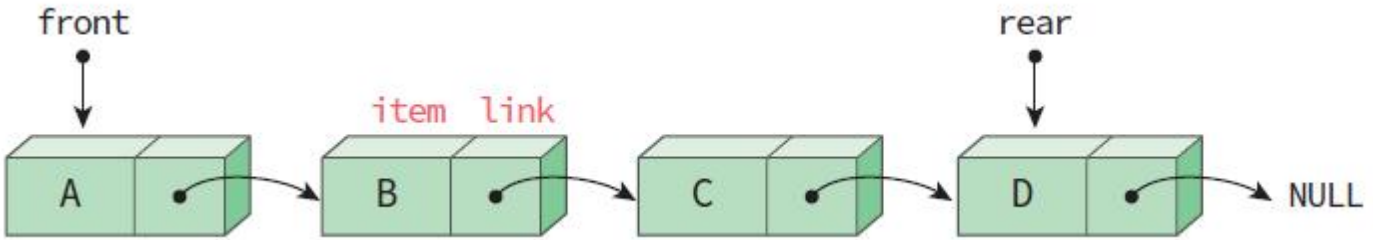


4. 나는 스택 자료구조를 연결 리스트로 구현할 수 있다

5. 나는 큐 자료구조를 연결 리스트로 구현할 수 있다

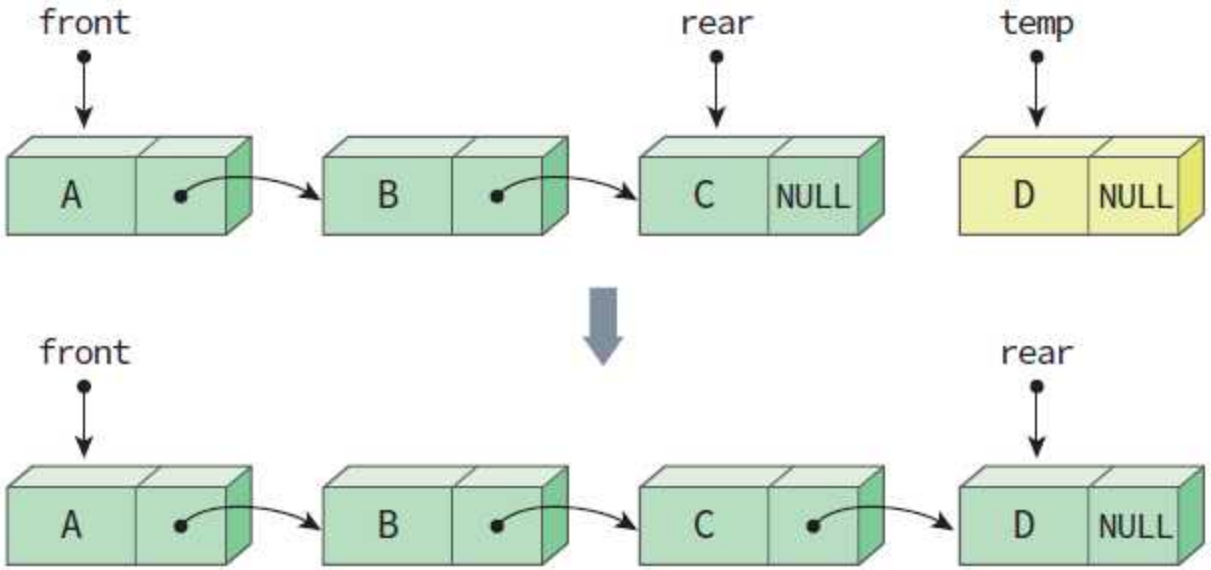
연결 리스트로 구현한 큐 (단순연결리스트)

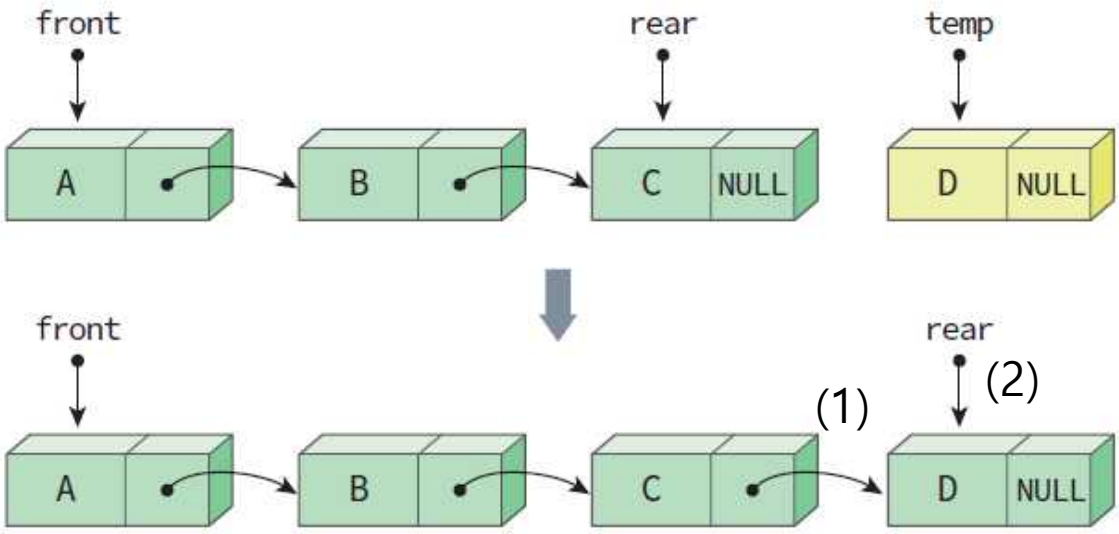
- 큐: 먼저 들어온 데이터가 먼저 나가는 자료구조
- 선입선출(**FIFO: First-In First-Out**)
- enqueue_rear(), dequeue_front()



```
typedef int element;              // 요소의 타입
typedef struct QueueNode {    // 큐의 노드의 타입
        element data;
        struct QueueNode *link;
} QueueNode;

typedef struct {                  // 큐 ADT 구현
        QueueNode *front, *rear;
} LinkedQueueType;
```

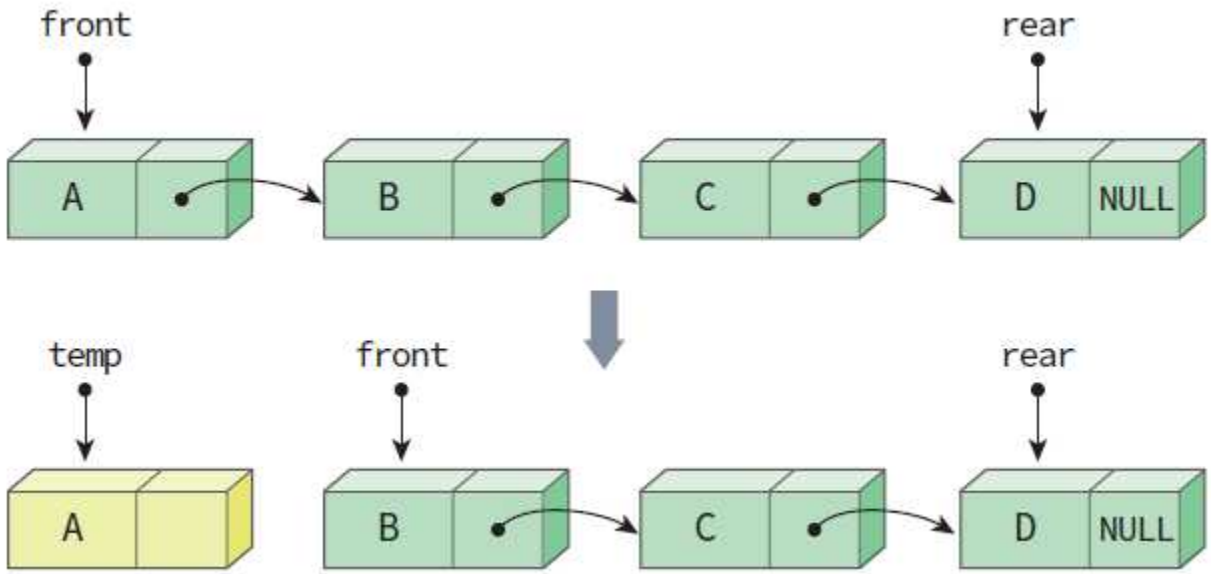
삽입 연산 (enqueue_rear())

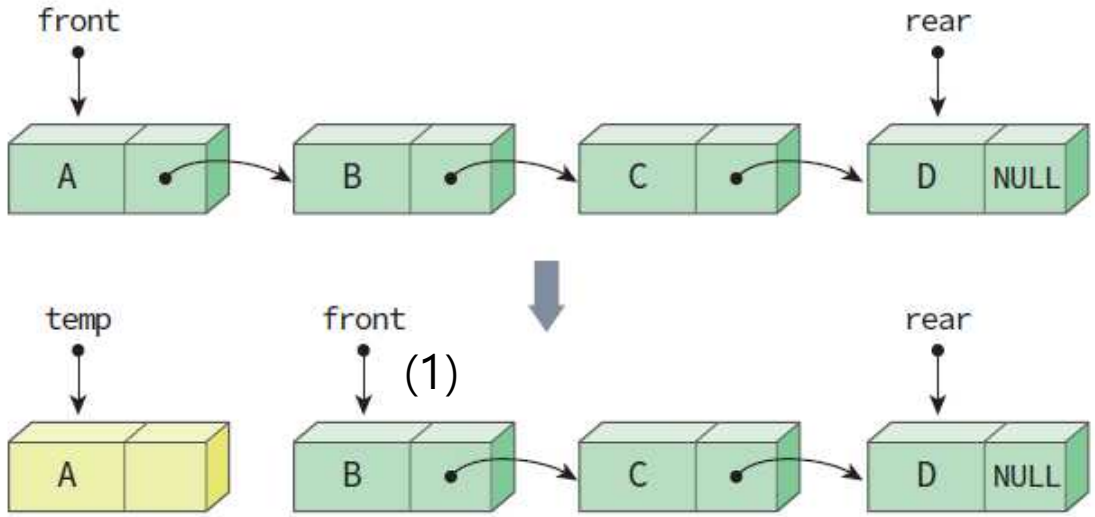





```
// 삽입 함수
void enqueue(LinkedQueueType *q, element data)
{
        QueueNode *temp = (QueueNode *)malloc(sizeof(QueueNode));
        temp->data = data;                          // 데이터 저장
        temp->link = NULL;              // 링크 필드를 NULL
        if (is_empty(q)) {              // 큐가 공백이면
                q->front = temp;
                q->rear = temp;
        }
        else {                  // 큐가 공백이 아니면
                q->rear->link = temp;  // (1)
                q->rear = temp;  // (2)
        }
}
```

삭제 연산 (dequeue_front())

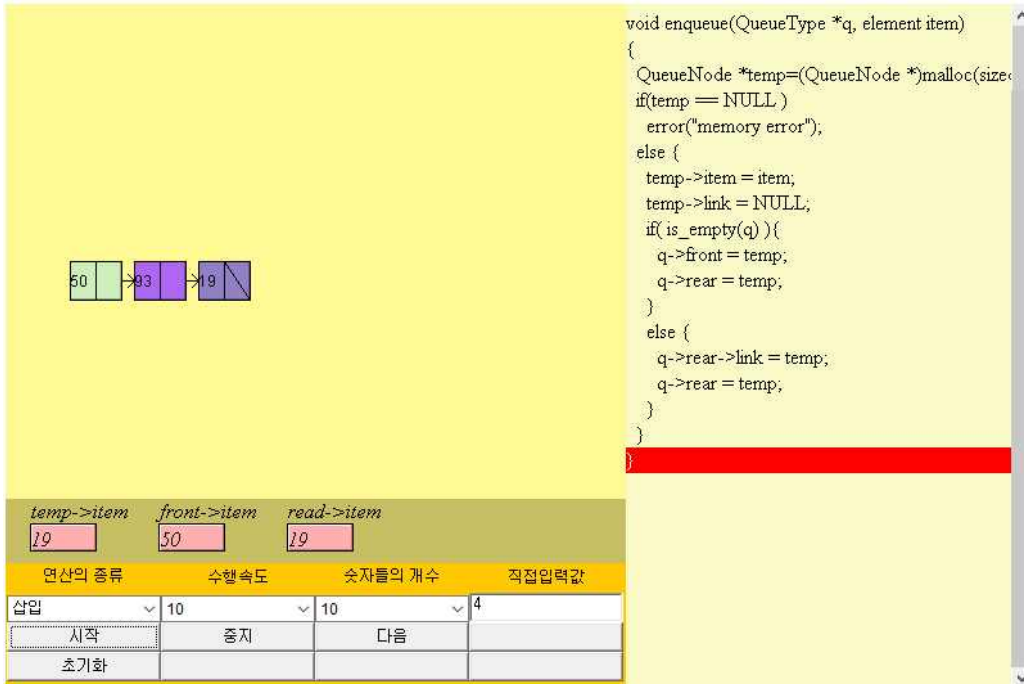





```
// 삭제 함수
element dequeue(LinkedQ
{
        QueueNode *temp =q-> front;
        element data;
        if (is_empty(q)) {              // 공백상태
                fprintf(stderr, "스택이 비어있음\n");
                exit(1);
        }
        else {
                data = temp->data;              // 데이터를 꺼낸다.
                q->front = q->front->link; // (1) front로 다음노드
                if (q->front == NULL)           // 공백 상태
                        q->rear = NULL;
                free(temp);                     // 동적메모리 해제
                return data;                    // 데이터 반환
        }
}
```

연결된 큐 자바 애플릿

- 그림 방식의 자바 애플릿 vs. 텍스트 방식의 VS 디버깅 줄단위로 자료구조와 알고리즘 동시에 봄



<연산>

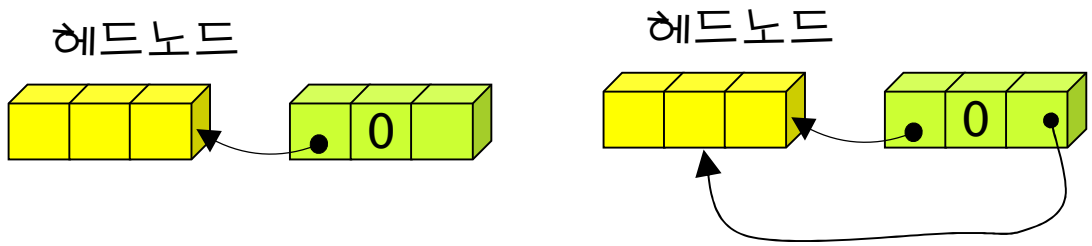
- void enqueue(LinkedQueueType *q, element data)
- element dequeue (LinkedQueueType *q)

<자료구조>

```
typedef int element;
typedef struct QueueNode {
  element data;
  struct QueueNode *link;
} QueueNode;
```

이제 나는 할 수 있다

1. 나는 리스트 자료구조를 원형연결리스트로 구현할 수 있다
2. 나는 리스트 자료구조를 이중연결리스트로 구현할 수 있다.
3. 나는 원형연결리스트와 이중연결리스트 내용변화를 C코드 한줄씩 그림으로 보여줄 수 있다



4. 나는 스택 자료구조를 연결 리스트로 구현할 수 있다
5. 나는 큐 자료구조를 연결 리스트로 구현할 수 있다

이제 나는 원형, 이중연결리스트를 할 수 있기에

1. 자료구조 구현 방법은 배열 또는 연결리스트로 구현
2. 자료구조 구현 방법을 이제 모두 배움
3. 이제까지 배운 자료구조는 4장 스택, 5장 큐, 6장 리스트
   앞으로 8장 트리배우고 2학기에 9장~14장 자료구조 배움
4. 매주 정답이 있는 연습문제 홀수문제 꼭 해결해보세요
5. 코딩시험 성적은 코딩한 만큼 나옴

대학교육의 목표는 학생들의 <문제해결능력생성>

- <문제해결능력생성>의 판단 방법은 연습문제 해결할 수 있나?

< 회사에서 개발자에게 요구사항>

1. 문제해결능력
2. 회사에서는 매일 문제가 터지고 있어서 실전문제가 널려 있음

이제 <문제해결능력생성> 되었기에 회사가서 실전문제 풀면 됩니다