자료구조 Data Structure | 조행래

연결 리스트

추가적인 리스트 연산

학습 목표

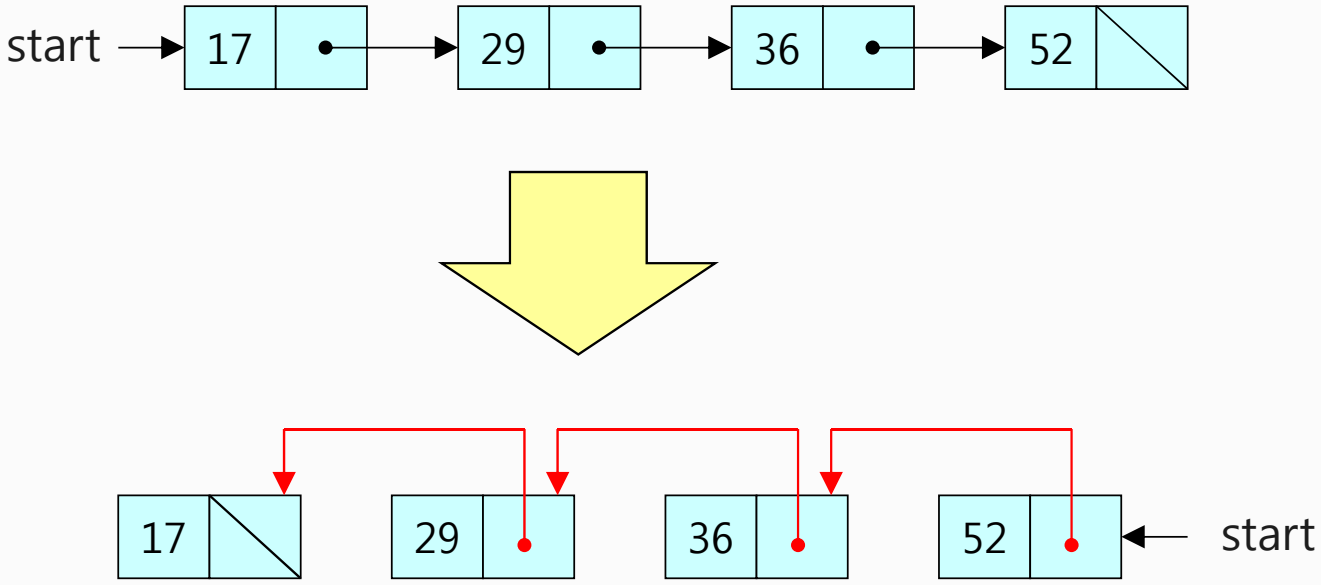
- 연결 리스트의 링크를 반대로 변경할 수 있다.
- 두 개의 연결 리스트를 하나의 리스트로 통합할 수 있다.
- 원형 연결 리스트의 길이를 계산할 수 있다.
- 원형 리스트에 노드를 추가할 수 있다.

1. 추가적인 리스트 연산들

- **단순 연결 리스트(체인) 에 대한 연산**
  - 체인의 방향을 반대로 : invert()
  - 두 개의 체인을 통합 : concatenate()

- **원형 리스트에 대한 연산**
  - 원형 리스트의 길이를 계산하는 연산: length()
  - 원형 리스트의 제일 앞에 새로운 노드를 삽입: insert_front()

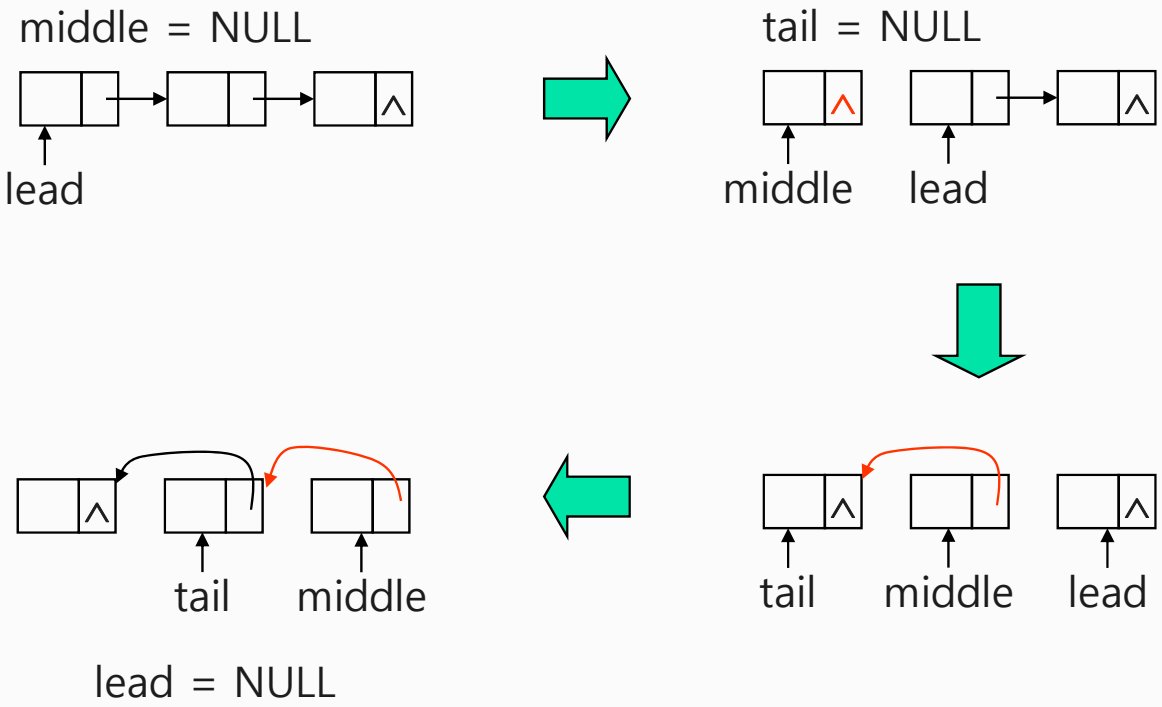
2. invert() 함수



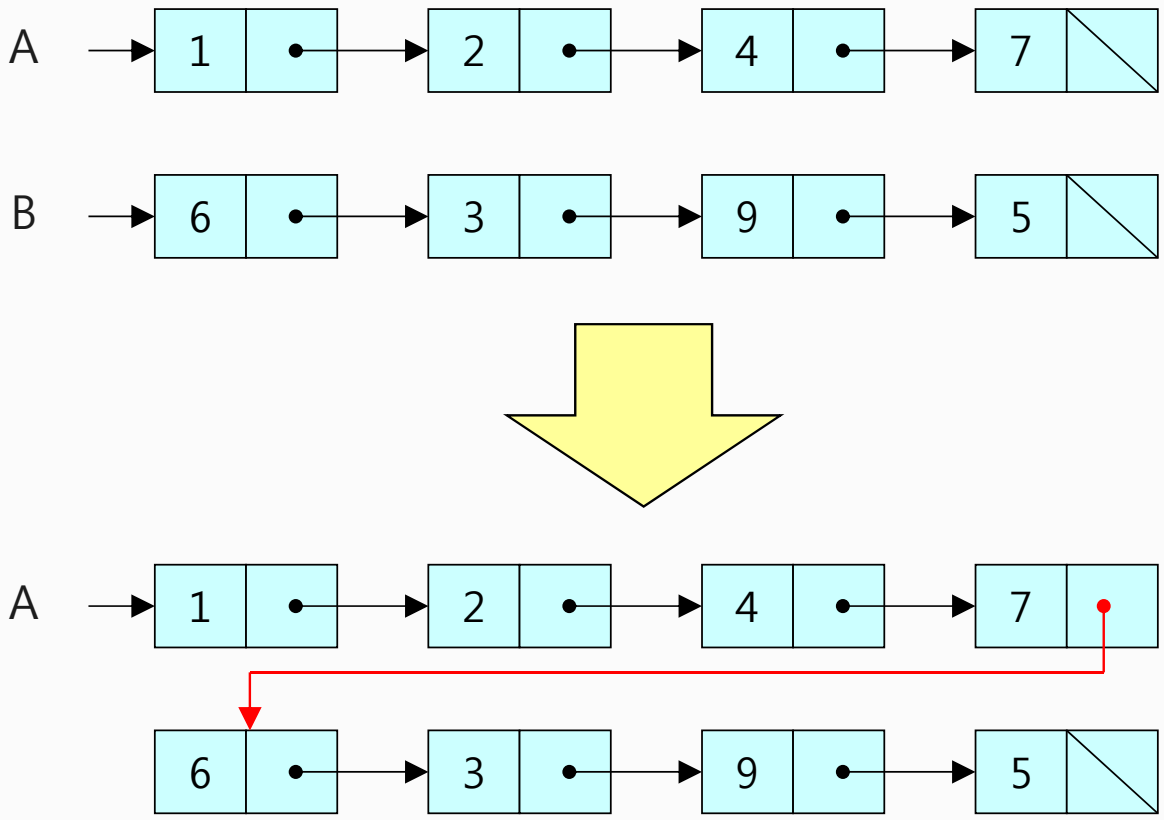
invert() 함수 - 알고리즘

```
list_pointer invert(struct node *lead)
{
//  lead가 가리키는 리스트의 방향을 반대로 변경/
    struct node *middle, *tail;
     middle = NULL;
    while (lead)   {
          tail = middle;
          middle = lead;
          lead = lead→link;
          middle→link = tail;
    }
    return middle;
}
```

invert() 함수 - 동작 과정



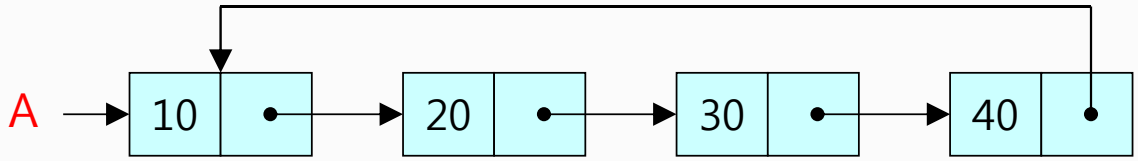
3. Concatenate() 함수



Concatenate() 함수 - 알고리즘

```
struct node *concatenate(struct node *ptr1, struct node *ptr2)
{
/* ptr1다음에 ptr2를 연결한 새로운 리스트를 반환.
   ptr1이 가리키는 리스트는 새로운 리스트로 변경됨.  */
   struct node *t;
   if (ptr1 == NULL)   return ptr2;
   else   {
        if (ptr2 != NULL)   {
                    for (t = ptr1; t→link != NULL; t = t→link)
                    ;
                    t→link = ptr2;
        }
        return  ptr1;
   }
}
```

4. 원형 연결 리스트의 순회




```
struct node *ptr;

for (ptr = A; ptr != null; ptr = ptr->link)
    sum += ptr->data;
```

Chain의 순회

```
struct node *ptr = A;
sum += ptr->data;       // 처음 데이터는 별도로 처리
for (ptr = A->link; ptr != A; ptr = ptr->link)
    sum += ptr->data;
```

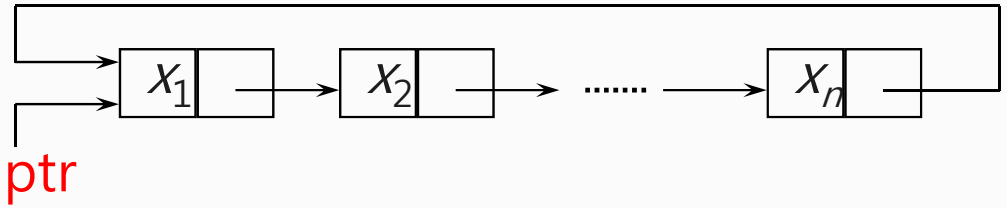
원형 연결 리스트의 순회

원형 연결 리스트의 길이 계산

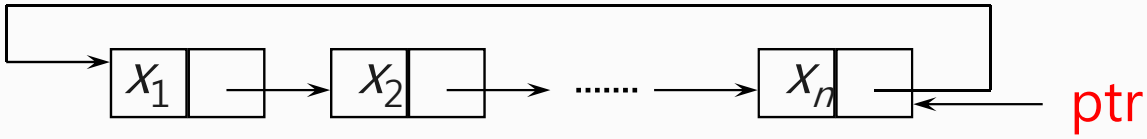
```
int length (struct node *ptr)
{
// ptr이 가리키는 원형 리스트의 노드 수를 return
        struct node *t;
        int count = 0;
        if (ptr != NULL)   {
                t = ptr;                    // 처음 노드를 먼저 처리
                do  {
                                count++;
                                t = t→link;
                }   while (t != ptr);
        }
        return count;
}
```

5. 원형 리스트의 이름

(a) 첫 번째 노드를 pointing

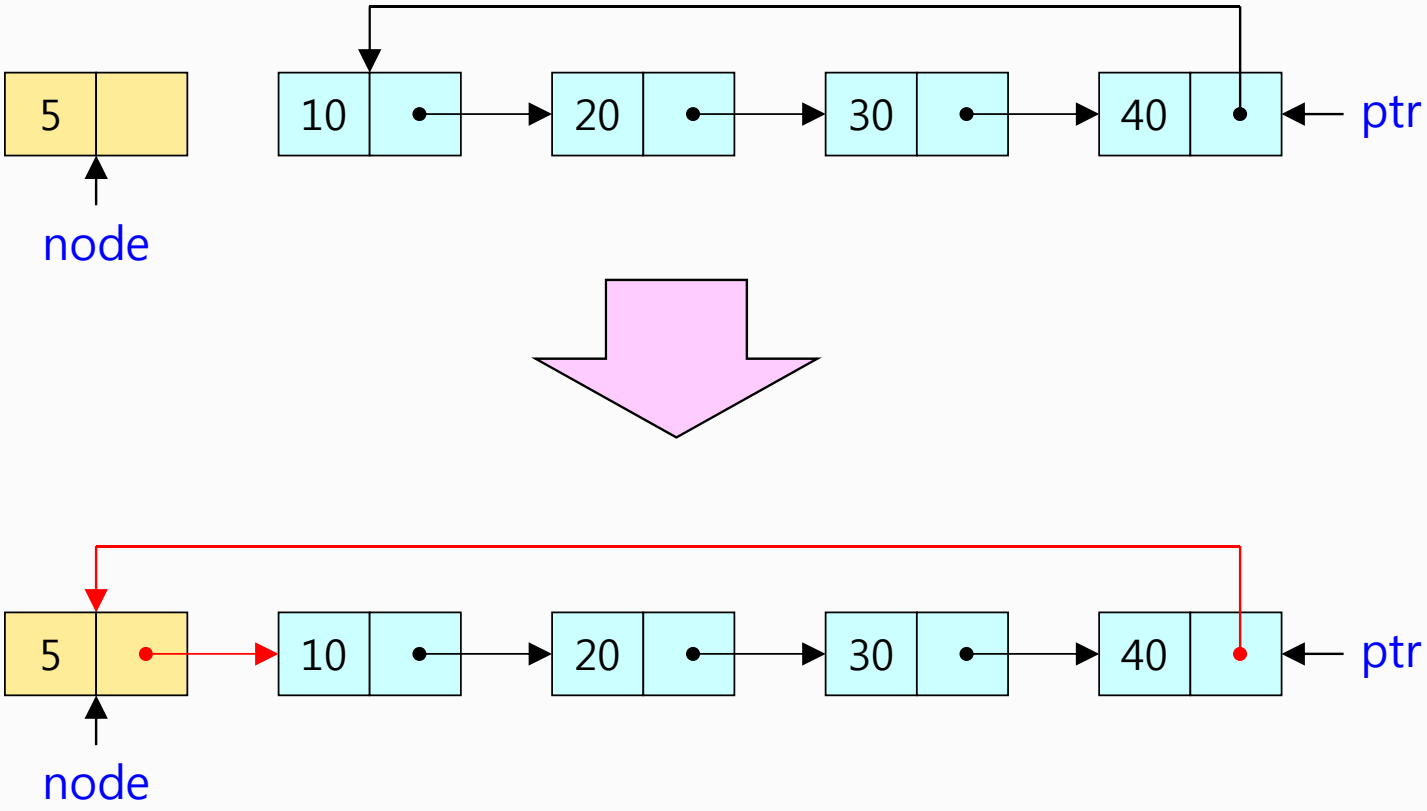


(b) 마지막 노드를 pointing



- **마지막 노드를 pointing할 경우, 다음 연산의 복잡성은 O(1)**
  - 리스트의 첫번째 위치에 새로운 노드 추가/삭제
  - 리스트의 마지막 위치에 새로운 노드 추가

insert_front() 함수 - 개념



insert_front() 함수 - 알고리즘

```
void   insert_front (struct node **ptr, struct node *node )
 /* ptr이 가리키는 원형 리스트의 선두에 node 추가.
    ptr은 원형 리스트의 마지막 노드를 가리키고 있음. */
{
    if (*ptr == NULL )   {
    // 리스트가 비었음: ptr이 새로운 노드를 가리키도록 변경
          *ptr = node;
          node→link = node;
    }
    else   {
    // 리스트에 노드가 존재: 선두에 노드 추가
          node→link = (*ptr)→link;
          (*ptr)→link = node;
    }
}
```

요약 정리

- 체인과 원형 연결 리스트에 대한 다양한 연산들을 설명
  - 연결 리스트의 링크를 반대로 변경
  - 두 개의 연결 리스트를 하나의 리스트로 통합
  - 원형 연결 리스트의 길이를 계산
  - 원형 리스트에 노드를 추가