자료구조 Data Structure | 조행래

연결 리스트

연결 리스트의 개념

학습 목표

- 연결 리스트의 개념을 이해한다.
- 연결 리스트의 구성 방법을 이해한다.
- 배열과 연결 리스트의 장, 단점을 정성적으로 비교할 수 있다.

1. 자기 참조 구조체

- **정의**
  - 동일한 타입의 다른 구조체에 대한 포인터를 멤버로 포함하는 구조체

- **예**

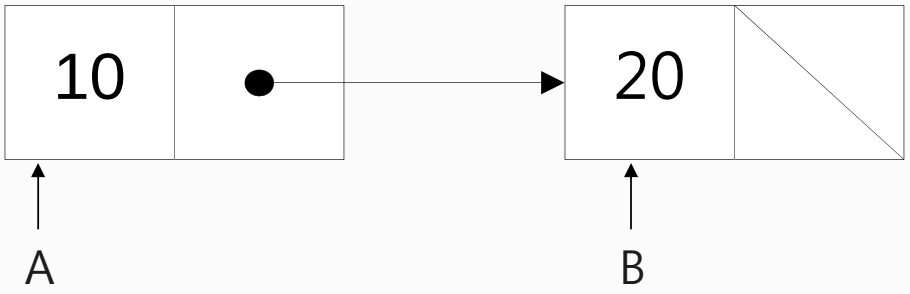
```
struct node {
    int     data;              // 다양한 타입의 data 가능
    struct node *link;
};
```

  - link
    - node 구조체에 대한 포인터
    - 여러 개의 node 구조체를 연결하는 역할

예: 두 개의 노드를 연결

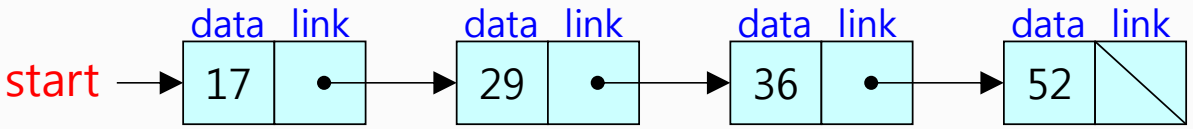
```
struct node {
    int data;
    struct node *link;
};

struct node *A, *B;

A = (struct node *) malloc(sizeof(struct node));
A->data = 10;
B = (struct node *) malloc(sizeof(struct node));
B->data = 20;

A->link = B;
B->link = NULL;
```

```
At <stdio.h>

#define NULL    ((void *) 0 )
```



2. 연결 리스트

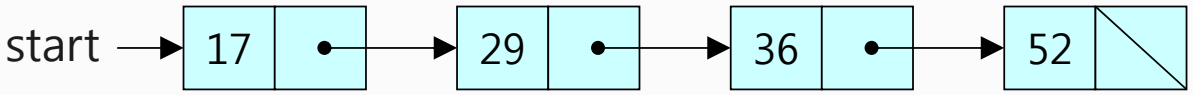
- **단순 연결 리스트** (Singly Linked List)
  - 자기 참조 구조체(노드)들의 연결
  - 첫 번째 노드에 대한 포인터만 유지
    - 리스트의 이름 = 첫 번째 노드의 주소
  - 이후 노드들은 구조체의 link 포인터를 통하여 참조
  - 마지막 노드의 link 포인터는 NULL로 설정



- **다른 연결 리스트**
  - 이중 연결 리스트 (Doubly Linked List)
  - 원형 연결 리스트 (Circular Linked List)

연결 리스트 연산 – 순회

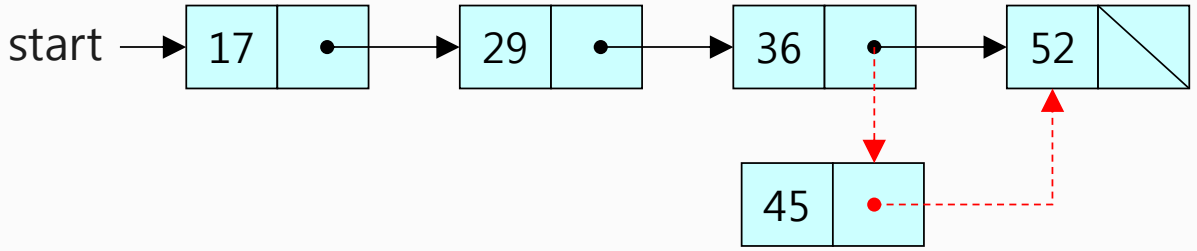
- **연결 리스트의 길이 계산**




```
struct node *ptr;
int  length = 0;

for (ptr = start; ptr != NULL; ptr = ptr->link)
    length++;

printf("노드 수 = %d₩n", length);
```

연결 리스트 연산 – 노드 추가

- **36 다음에 45를 추가**

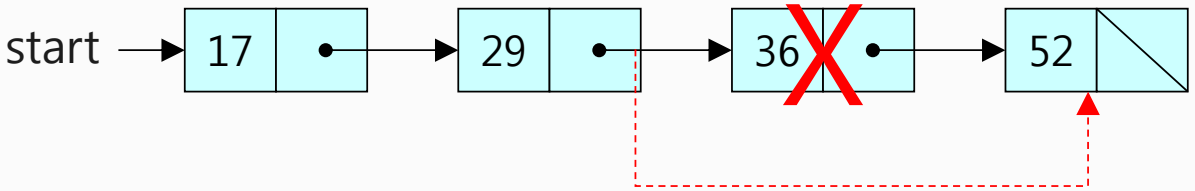



```
struct node *ptr, *tmp;

tmp = (struct node *) malloc(sizeof(struct node));
tmp->data = 45;
for (ptr = start; ptr->data != 36; ptr = ptr->link) ;

tmp->link = ptr->link;
ptr->link = tmp;
```

노드를 추가할 때, 다른 노드의 위치 변경이 없다!

연결 리스트 연산 – 노드 삭제

- **데이터 36을 가진 노드를 삭제**




```
struct node *ptr, *target;

for (ptr = start; ptr->link->data != 36; ptr = ptr->link) ;

target = ptr->link;
ptr->link = target->link;
free(target);
```

노드를 삭제할 때, 다른 노드의 위치 변경이 없다!

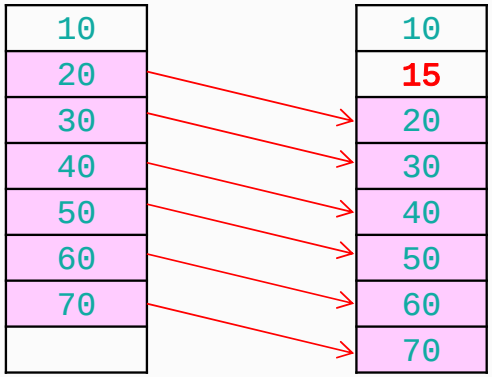
3. 배열과 연결 리스트의 비교

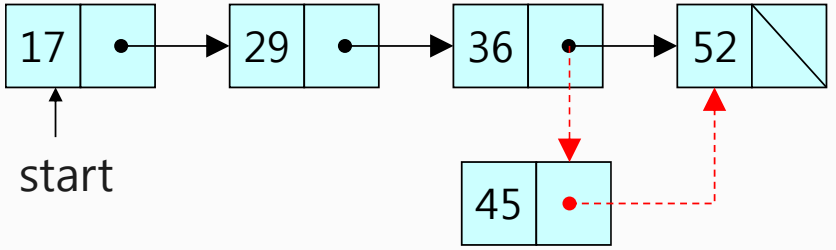
- **저장 방식의 차이**
  - 배열: int A[4]; ← 메모리의 인접한 곳에 저장
    - 다음 데이터에 대한 주소를 알 필요 없음
  - 연결 리스트: struct node에 대한 네 번의 malloc
    - 각 노드들은 메모리의 여러 곳에 나누어 저장
    - link를 이용하여 다음 노드의 주소 유지

- **메모리 사용 측면**
  - 저장될 데이터의 수를 안다면 배열이 효과적
  - 데이터의 수를 모를 경우, 연결 리스트가 유리
    - 새로 데이터가 입력될 때마다 malloc 실행 후 연결

배열과 연결 리스트의 비교 (계속)

- **정렬된 데이터의 순서 유지**
  - 배열:
    - 데이터가 추가될 때 기존 데이터의 위치 변경 가능
    - 이진 검색 가능
  - 연결 리스트
    - 기존 데이터의 위치 변경은 발생하지 않음
    - 이진 검색은 불가능





요약 정리

- 연결 리스트의 개념을 설명
- 연결 리스트의 순회, 노드 추가 및 삭제 방법을 설명
- 배열과 연결 리스트의 장, 단점을 설명