5주차 1차시 연결자료구조

【학습목표】

1. 연결 자료구조를 설명할 수 있다.
2. 순차 자료구조와 연결 자료구조의 장점과 단점을 설명할 수 있으며, 차이점을 통해 구분할 수 있다.

학습내용1 : 순차자료구조의 문제점

1. 순차자료구조 특징

- 원소들간의 논리적인 순서와 메모리에 저장하는 물리적인 순서가 같은 구조로 되어있음
- 논리적인 순서대로 연속적으로 저장
- 원소의 위치를 찾아 액세스하기 용이
- 삽입 연산이나 삭제 연산 이후에 연속적인 물리적 위치를 유지하기 위하여 원소들을 이동시키는 추가적인 작업과 시간이 소요

2. 순차자료구조의 문제점

- 원소들의 개수가 많고 삽입·삭제 연산이 많이 발생하는 경우에는 원소들의 이동 작업으로 인한 오버헤드가 많이 발생
- 순차자료구조가 배열을 이용하여 구현하기 때문에 배열이 갖고 있는 메모리 사용의 비효율성 문제 발생
    - 예) 희소 다항식이나 희소 행렬 저장시
    ☞ 문제점이 개선된 표현방법으로 연결자료구조(Lnked Data Structure) 또는 비순차 자료구조(Nonsequential Data Structure)

학습내용2 : 연결 리스트의 노드

1. 연결 자료구조

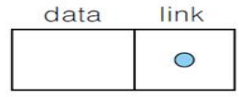
- 원소의 논리적인 순서와 물리적인 순서가 일치할 필요가 없다
- 연속한 물리주소에 의해 원소의 순서를 표현한 것이 아니라 각 원소에 저장되어 있는 다음 원소의 주소에 의하여 순서가 연결되는 방식
- 여러 개의 작은 공간을 연결하여 하나의 전체 자료구조를 표현
    - 크기변경이 유연하고 좀 더 효율적으로 메모리 사용 가능

2. 연결 리스트

- 리스트를 연결 자료구조로 표현한 구조
- 연결방식에 따라 구분
    - 단순 연결 리스트, 원형 연결 리스트, 이중 연결 리스트, 이중 원형 리스트 등

3. 연결 리스트의 노드

- 연결 자료구조에서 하나의 원소를 표현하기 위한 단위 구조
- <원소, 주소> 단위로 저장



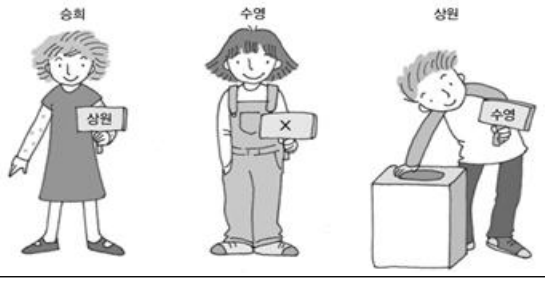
[그림 5-2] 노드에 대한 논리적 구조

- 데이터 필드(Data Field) : 원소의 값을 저장, 원소의 형태에 따라 하나 이상의 필드로 구성
- 링크 필드(Link Field) : 다음 노드의 주소를 저장, 포인터 변수를 사용하여 주소값을 저장, 포인터 또는 링크, 참조

* 노드 연결 방법 - 기차놀이

[그림 5-4] 기차놀이 : 뽑은 이름표대로 기차 연결하기

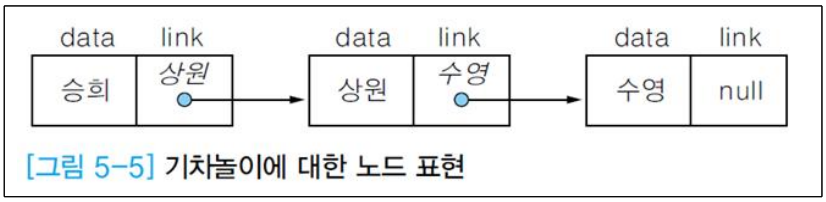
- 이름표 뽑기



[그림 5-3] 기차놀이 : 이름표 뽑

- 자기가 뽑은 이름표의 사람 찾아 연결하기
    - X표를 뽑은 사람은 마지막 기차 칸
    - 이름표를 들고 있는 방향으로 움직임

- 기차놀이와 연결 리스트
    - 기차놀이 하는 아이들 : 연결 리스트의 노드
    - 이름표 : 노드의 링크 필드

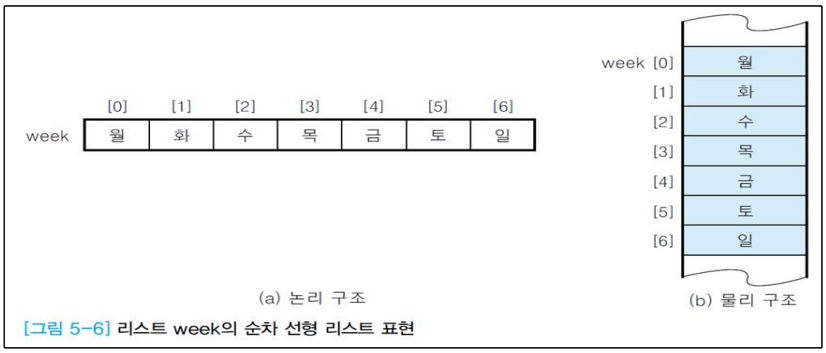


[그림 5-5] 기차놀이에 대한 노드 표현

학습내용3 : 선형 리스트와 연결 리스트 비교

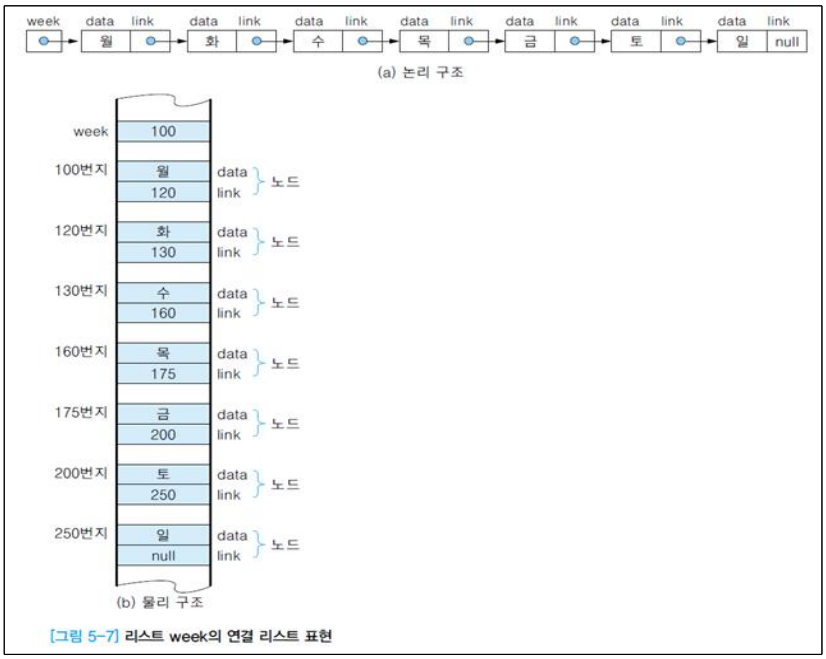
1. 사례 비교

* 리스트 week=(월, 화, 수, 목, 금, 토, 일)
- week에 대한 선형 리스트



[그림 5-6] 리스트 week의 순차 선형 리스트 표현

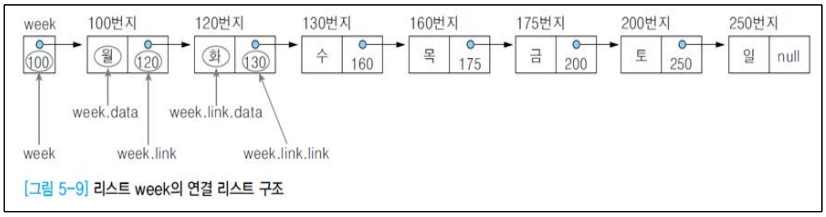
- week에 대한 연결 리스트



[그림 5-7] 리스트 week의 연결 리스트 표현

- 리스트 이름 week : 연결 리스트의 시작을 가리키는 포인터 변수

* 포인터 변수 week는 연결 리스트의 첫 번째 노드를 가리키는 동시에 리스트 전체를 의미
- 연결 리스트의 마지막 노드의 링크필드 - 노드의 끝을 표시하기 위해 null(널) 저장
- 공백 연결 리스트 - 포인터변수 week에 null을 저장(널 포인터)
- 각 노드의 필드에 저장한 값은 포인터의 점 연산자를 사용하여 액세스
  - ▶ week.data : 포인터 week가 가리키는 노드의 데이터 필드 값 “월”
  - ▶ week.link : 포인터 link가 가리키는 노드의 링크 필드에 저장된 주소값 “120”



[그림 5-9] 리스트 week의 연결 리스트 구조

- 리스트 week의 노드에 대한 C 프로그램 구조체 정의

```
typedef struct Node{
    char data[4];
    struct Node*link;
};
```

【학습정리】

1. 연결 자료구조는 다음 원소의 주소에 의해 순서가 연결되는 방식이므로 순차 자료구조와는 달리 물리적인 순서를 맞추기 위한 오버헤드가 발생하지 않는다.

2. 연결 자료구조에서 원소는 연결될 다음 원소에 대한 주소를 저장해야 하기 때문에 〈원소, 주소〉 단위로 저장하는데 이러한 단위 구조를 노드라 한다.

3. 노드는 원소의 값을 저장하는 데이터 필드와 다음 노드의 주소를 저장하는 링크 필드로 구성한다.