5주차 2차시 단순 연결 리스트1

【학습목표】

1. 단순 연결 리스트를 설명할 수 있다.
2. 단순 연결 리스트 연산방법을 설명할 수 있다.

학습내용1 : 단순 연결 리스트의 개요

1. 단순 연결 리스트란?

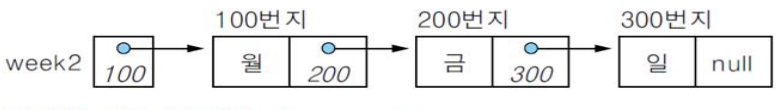
* 노드가 하나의 링크 필드에 의해서 다음 노드와 연결된 구조를 갖는 연결 리스트
* 연결 리스트, 선형 연결 리스트(Linear Linked List), 단순 연결 선형 리스트(Singly Linked Linear List)라고도 함

2. 단순 연결 리스트의 예

* 단순 연계 리스트 예1)
- 이름표를 하나만 가지고 한쪽으로만 연결된 인간 기차

[그림 5-4] 기차놀이 : 뽑은 이름표대로 기차 연결하기

- 단순 연계 리스트 예2)



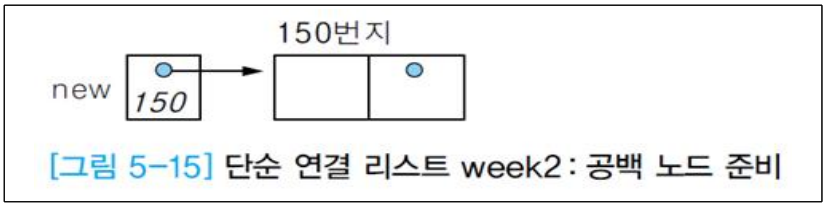
[그림 5-14] 단순 연결 리스트 week2

학습내용2 : 단순 연결 리스트의 삽입과 삭제

1. 삽입 연산

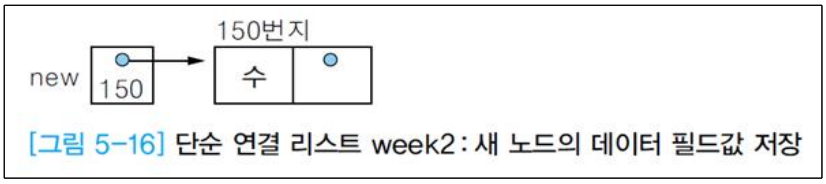
* 리스트 week2=(월, 금, 일)에서 원소 “월”, “금” 사이에 “수”를 삽입

① 삽입할 새 노드를 만들 공백 new(150번지의 노드)를 메모리에서 할당받아서 포인터 변수 new가 가리키게 한다



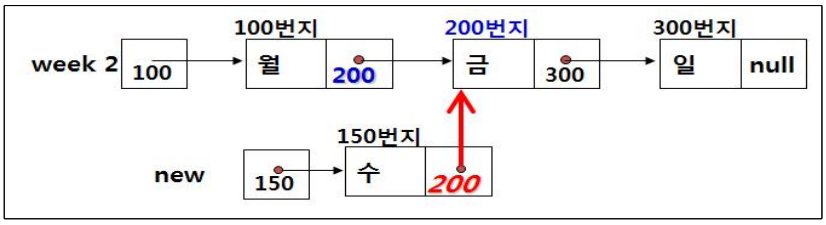
[그림 5-15] 단순 연결 리스트 week2 : 공백 노드 준비

② new 노드의 데이터 필드에 “수”를 저장한다

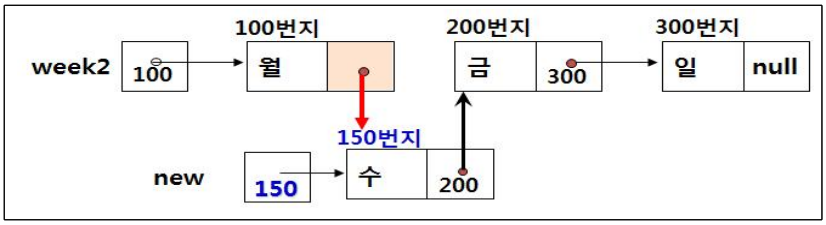


[그림 5-16] 단순 연결 리스트 week2 : 새 노드의 데이터 필드값 저장

③ new 노드의 앞 노드 즉 “월” 노드의 링크 필드값(200)을 new 노드 링크 필드에 저장



④ new 노드의 값(new 노드가 가리키고 있는 새 노드의 주소 150)을 “월” 노드의 링크 필드에 저장

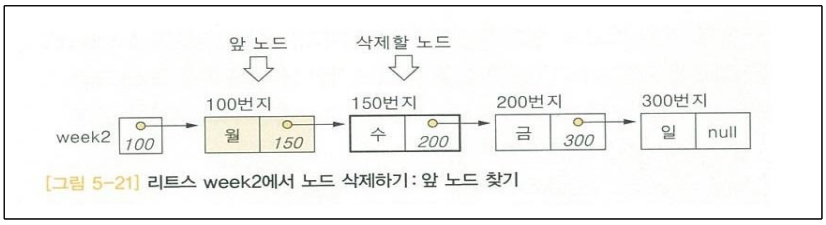


☞ 단순 연결 리스트의 삽입 연산에서는 물리적 순서를 유지하기 위해서 원소들을 이동시키지 않고 링크 필드의 포인터값에 대한 연산만으로 삽입 연산을 수행 하는 장점

2. 삭제 연산

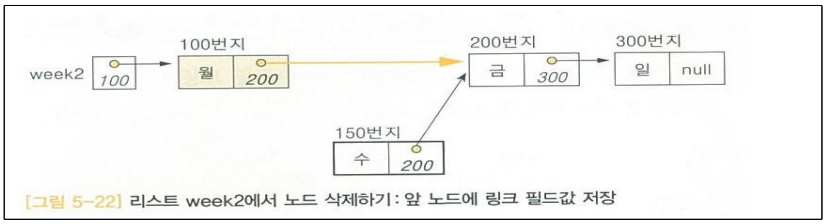
* 리스트 week2=(월, 수, 금, 일)에서 원소 “수”를 삭제

① 삭제할 노드의 앞 노드(선행자) 찾기



[그림 5-21] 리트스 week2에서 노드 삭제하기 : 앞 노드 찾기

② 삭제할 원소의 앞 노드 “월” 노드의 링크 필드에 삭제 노드의 링크 필드값을 저장



[그림 5-22] 리스트 week2에서 노드 삭제하기 : 앞 노드에 링크 필드값 저장

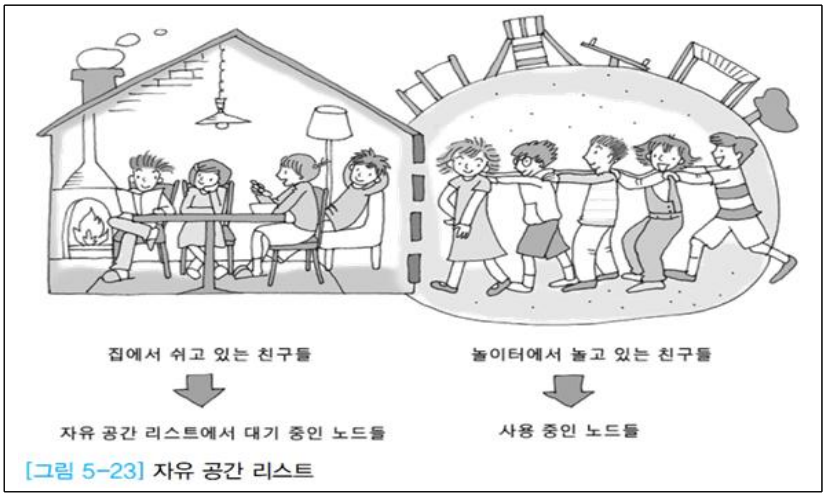
☞ 단순 연결 리스트의 삭제 연산에서도 원소들을 이동시키지 않고 링크 필드의 포인터값에 대한 연산만으로 삭제 연산을 수행

학습내용3 : 자유 공간 리스트(Free Space List)

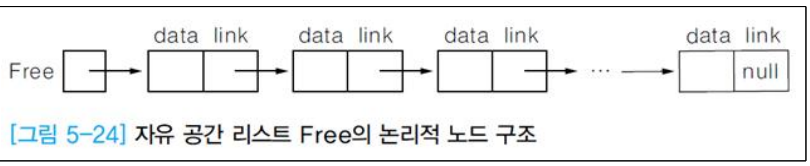
1. 정의

* 메모리를 사용하기 전에 미리 노드로 나누어서 연결해 놓은 리스트
- 새 노드가 필요할 때 리스트에서 할당받아 사용하고, 사용하지 않는 노드는 다시 리스트로 반환

* 연산과정과 메모리 관리가 효율적으로 이루어짐



[그림 5-23] 자유 공간 리스트



[그림 5-24] 자유 공간 리스트 Free의 논리적 노드 구조

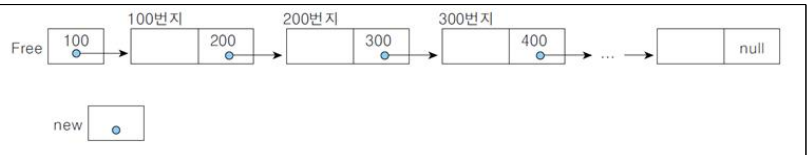
2. 자유 공간 리스트에서 노드 할당

* 노드 할당 알고리즘

알고리즘 5-1 자유 공간 리스트에서의 노드 할당 알고리즘

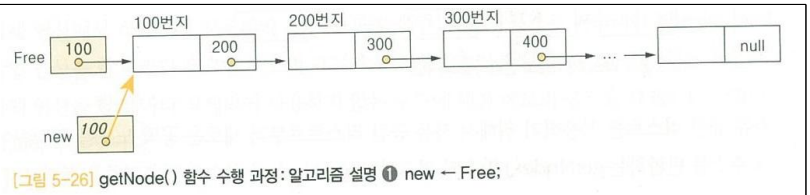
```
getNode()
        if (Free = null) then
            underflow();            // 언더플로우 처리 루틴
        new ← Free;                 // ❶
        Free ← Free.link;           // ❷
        return  new;
end getNode()
```

* 자유 공간 리스트에서의 노드 할당 과정
- 자유 공간 리스트 free에서 노드를 할당할 때에는 항상 첫 번째 노드를 할당
- 초기 상태



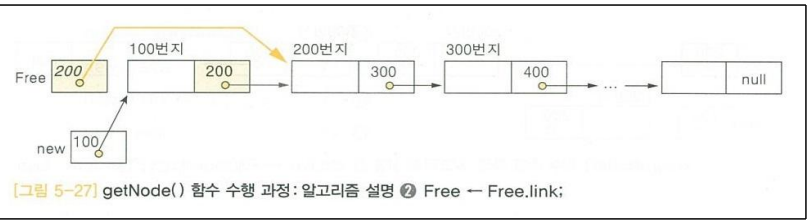
[그림 5-25] getNode( ) 함수 수행 과정: 초기 상태

❶ new ← Free;
- 리스트 Free의 첫 번째 노드의 주소(100)를 포인터 new에 저장하여 포인터 new가 할당할 노드를 가리키게 함



[그림 5-26] getNode( ) 함수 수행 과정: 알고리즘 설명 ❶ new ← Free;

❷ Free ← Free.link;
- 포인터 Free에 리스트의 두 번째 노드의 주소(Free.link)를 저장



[그림 5-27] getNode( ) 함수 수행 과정: 알고리즘 설명 ❷ Free ← Free.link;

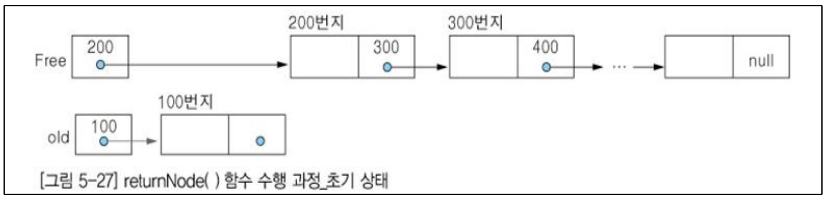
3. 자유 공간 리스트로의 노드 반환

* 노드 반환 알고리즘

알고리즘 5-2 자유 공간 리스트로의 노드 반환 알고리즘

```
returnNode(old)
        old.link ← Free;     // ❶
        Free ← old;          // ❷
end returnNode()
```

* 자유 공간 리스트로의 노드 반환 과정
- 사용이 끝난 노드를 자유 공간 리스트 Free의 첫 번째 노드로 다시 삽입
- 초기상태



[그림 5-27] returnNode( ) 함수 수행 과정_초기 상태

❶ old.link ← Free;
- 자유 공간 리스트 Free의 첫 번째 노드 주소(200)를 반환할 노드 포인터 old.link에 저장

【학습정리】

1. 연결 리스트는 리스트를 연결 자료구조로 표현한 구조이다.

2. 단순 연결 리스트의 삽입과 삭제 과정에 대하여 알아보았다.

3. 자유 공간 리스트란 사용하기 전의 메모리나 사용이 끝난 메모리를 관리를 용이하게 하기 위해 노드로 구성한 연결 리스트이다.