4주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20191619 이름: 이동석

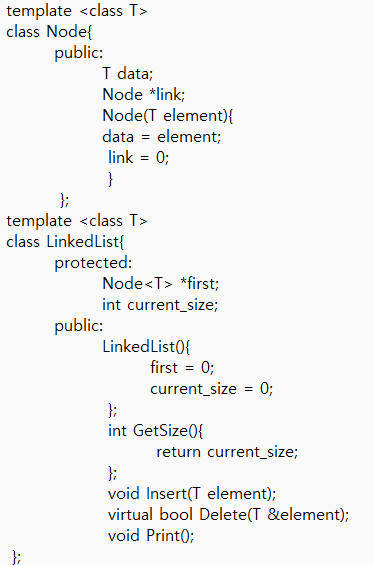
**1. 코드**

**1-1. LinKedLish.h**

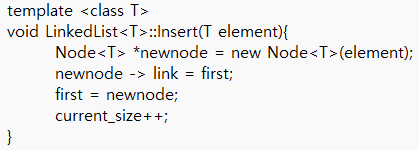
기본 구조는 두 개의 클래스 class Node , class LinkedList 로 이루어진다. 이를 템플릿 자료형 template <class T>를 이용해 선언하여 어떤 자료형에 상관없이 연결이 가능한 LinkedList 코드로 일반화 시켜준다.

클래스 Node에는 데이터를 저장할 변수 data와 LinkedList의 핵심인 다음 노드의 주소를 저장 할 포인터 변수 link가 있다.

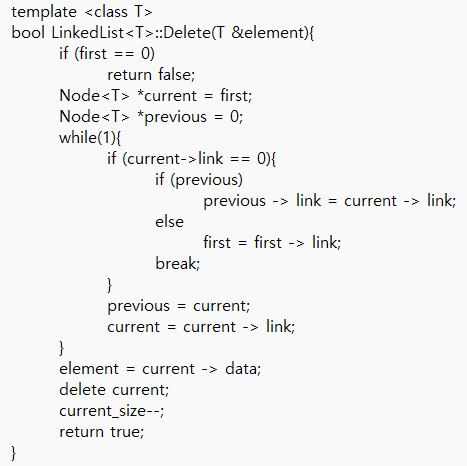
클래스 LinkedList에는 protected 변수로 첫 번째 node의 주소를 담고 있는 포인터 변수 first와, 전체 node의 사이즈를 담고 있는 정수형 변수 current\_size가 있다. public 변수로는 원소의 값을 삽입하기 위한 함수 Insert, 원소를 삭제하기 위한 함수 Delete, 마지막으로 현재 LinkedList를 출력하기 위한 함수 Print로 구성되어 있다.



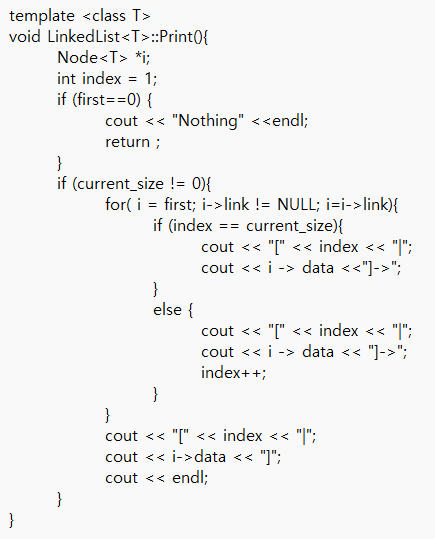
이제, LinkedList 클래스 내부의 함수들을 구현해야 한다. Insert 함수는 앞서 말했듯 맨 앞에 원소를 삽입하는 역할을 한다. 이를 구현하기 위해서는 우선 새로운 데이터를 저장 할 Node를 생성해야한다. 기존의 첫 번째 노드의 주소를 저장하고 있던 포인터 변수 first는 새로 생성된 노드의 다음 노드의 주소가 된다. 또한, 이제 첫 번째 노드는 현재 삽입된 노드이므로, first에 새로운 노드의 주소를 할당한다.



Delete 함수는 Stack과 LinkedList에서 삭제하는 정의가 다르므로, 가상함수를 뜻하는 virtual 로 선언되었다. LinkedList에서 원소의 삭제는 가장 최근에 삽입 된 원소를 삭제하는 FIFO(First In First Out)의 방식이다. 따라서, 우선적으로 삭제할 원소가 있는지 확인해준다. 이후, 첫 번째 노드에서 link를 따라가며 마지막 node 까지 찾고, 마지막 이전의 노드가 마지막 노드가 가르키던 주소를 할당 받는다.

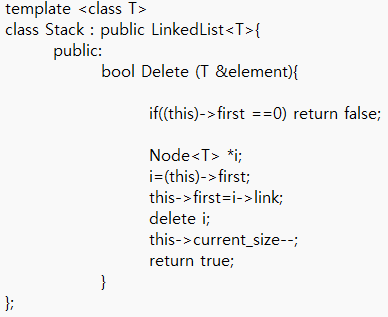


Print 함수는 단순하게 현재 연결되어있는 노드들의 값을 형식에 맞게 출력한다. first에서부터 link를 타고 마지막 노드의 link가 null이 될 때 까지 출력을 해주면 된다.



**1-2. Stack.h**

스택의 경우, 앞서 말한 Delete 함수를 제외하고 나머지 기능은 LinkedList와 같다. 따러서 LinkedList를 상속받은 후, Delete 함수만 스택의 기능에 맞게 재정의를 해주면 된다. Stack의 경우, LIFO(Last In First Out)을 따른다. 따라서, first는 기존의 두 번째 노드의 주소를 할당 받게 되고, 기존의 첫 번째 노드는 delete 해주면 된다. 이때 주의할 점은 class의 멤버 변수에 접근 할 때 this 포인터를 사용해야 한다.



**2. 시간 복잡도**

연결리스트의 시간 복잡도는 어떠한 것을 할지에 따라 다르다. 가장 먼저, 원소의 삽입의 경우 맨 앞에 삽입을 하므로 시간복잡도는 O(1)을 가진다. 원소의 삭제의 경우, 가장 마지막 node 까지 값을 찾으러 가야하므로, 시간복잡도 O(n)을 가지게 된다. 반면, 스택의 경우는 맨 앞의 원소를 삭제하므로, O(1)의 시간복잡도를 가진다.

일반적인 배열의 경우와 비교해서 연결리스트의 시간복잡도가 더 느리다는 것을 알 수 있다. 또한, 특정 값을 찾기 위해서 일일히 node를 탐색해야 한다. 하지만, 연결리스트는 배열과 다르게 원소의 삽입과 삭제가 자유롭다는 점에서 배열과는 다른 장점을 가지고 있다.