4주차 결과보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181288 이름: 윤성호

* 1. 첫번째 과제는 int형 데이터만을 받을 수 있는 LinkedList 클래스를 템플릿을 이용하여 모든 타입의 데이터를 받을 수 있도록 변형하는 것이었다. template <typename T>를 클래스와 함수 상단에 추가해주고, 내부에서 정의하는 변수들의 데이터타입을 T로 설정해줌으로써 구현할 수 있다. 이때 T는 모든 타입을 대변하는 이름이다. LinkedList 클래스는 여러개의 Node 클래스가 연결되어 있는 형태로 정의한다. Node 클래스는 총 멤버변수를 가지는 클래스로, 데이터를 저장할 변수와, 다음노드의 주소를 저장할 포인터를 갖고있다. 이때 다음 노드 포인터는 T타입으로 정의한다. 현재 구현하는 LinkedList의 형태는 Singly Linked List로, 마지막 노드를 찾아가기 위해서는 첫번째 노드부터 차례대로 다음노드를 따라가야 한다. 이는 멤버함수 중 Delete의 구현에서 중요하게 다뤄지게 된다.

LinkedList는 아래의 2개의 멤버변수와 5개의 멤버함수로 이루어져 있다.

<멤버변수>

* Node<T> \*first
* Int current\_size

<멤버함수>

* 생성자
* Int GetSize()
* Void Insert(T element)
* Virtual bool Delete(T &element)
* Void Print()

Node<T> \*first는 첫번째 노드의 주소를 저장할 포인터이다. 데이터타입은 T타입으로 설정함으로써 모든 타입을 커버할 수 있게 설정하였다. Current\_size는 현재 linkedList에 있는 노드의 개수를 담고있는 변수이다.

생성자는 first와 current\_size를 0으로 초기화해준다. Int GetSize() 함수는 현재 노드의 개수를 반환해주는 함수이다. 현재 linkedList는 queue 자료구조와 같이 작동한다. 따라서 원소 삽입과 삭제 과정은 First In First Out을 생각하여 구현해주면 된다. Void Insert(T element)는 T타입의 element를 삽입해주는 함수이다. 이때 새로운 노드를 T타입으로 생성한 후 그 노드를 LinkedList의 가장 첫번째 노드로 설정해준다. Virtual bool Delete(T &element) 현재 원소가 없다면 False를 반환해주고, 그게 아니라면 마지막 노드를 첫번째노드부터 차례차례 따라가서 찾아서, 삭제해준다. 그리고 current\_size를 1감소시켜준다. 삭제가 성공했으므로 True를 반환해준다. Delete 함수를 가상함수로 만들어준 이유는 2번과제에서 stack을 구현할때는 다른 방식으로 동작해야 하기 때문에 Delete라는 이름으로 다시 정의해주기 위함이다. 같은 이름의 멤버함수를 객체의 형태에 따라 다르게 동작하게 하는 서브타입 다형성을 확인할 수 있다. Void Print()는 현재의 모든 노드를 처음부터 출력해주는 함수이다. Node<T> 타입의 포인터변수 i를 설정하고, 그 변수를 이용해 현재노드에서 다음노드로 가는 반복문을 구성한다. 그렇게 이동하면서 data 멤버변수의 값을 출력한다.

* 1. LinkedList 클래스를 상속받아 Stack 클래스를 작성하는 과제이다. 이때 멤버변수에 접근하기 위해서는 객체 자신을 가리키는 포인터인 this를 이용한다. 모든 멤버변수와 멤버함수를 상속받아 사용하지만 LinkedList와 다르게 Stack은 Last in First out 으로 작동하기 때문에 삭제 함수를 다르게 적용해야 한다. 그러나 같은 이름의 Delete를 사용하여 LinkedList와 Stack을 모두 작동시키고 싶기 때문에 Delete함수를 가상함수로 정의하였다. Stack의 경우 LinkedList때처럼 마지막노드를 찾기위해 첫번째노드부터 쭉 따라가는 반복문이 필요없다. 삭제 함수는 첫번째노드를 삭제하기 때문이다. Stack의 원소가 하나도 없으면 False를 리턴하고, 그러지 않다면 가장 첫번째 노드를 삭제하고, current\_size 멤버면수를 하나 줄이고, 삭제를 성공했으므로 True를 리턴한다. 이렇게 한다면 같은이름의 Delete로 LinkedList와 Stack에서 다르게 동작하는 함수를 만들어 낼 수 있다. 여기서 서브타입 다형성을 확인할 수 있다.