4주차 예비보고서

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

1. 이번 문제는 LinkedList와 Stack의 구현을 통해 다형성 중 파라미터적 다형성과, 서브타입 다형성을 실습해본다.

파라미터적 다형성의 구현을 위해서는 템플릿을 이용한다. 다양한 타입의 입력에 대한 코드의 실행과정이 일치할 경우 사용하는 방법이다. 템플릿을 이용하지 않는다면 이런 상황에서 파라미터의 자료형만 다르게 명시해주고 같은 코드를 여러번 복사, 붙여넣기를 해야한다. 이는 굉장히 비효율적인 과정이며, 버그가 생길경우 모든 부분을 일일이 고쳐주어야 한다는 불편함이 존재한다. 그러나 템플릿을 이용한다면 중요 기능이 구현되는 부분은 한번만 작성하면 되므로 이러한 불편함은 사라진다. 이를 코드의 유지보수가 편해지고, 재사용성이 증가한다고 표현한다. 과제의 LinkedList는 모든 타입의 입력에 대해서 커버를 해야 하기 때문에 템플릿을 이용한다. Template <class T> 라고 함수나 클래스의 가장 윗부분에 적어두고, 기존의 코드의 int 자료형이라고 적어놓았던 부분을 T 자료형이라고 바꿔주면 LinkedList에 넣고 뺄 수 있는 item들의 자료형은 int뿐 아니라 모든 자료형이 된다. String 자료형도 사용해야 하므로 #include <string>도 해준다.

서브타입 다형성은 같은 명령에 대해 각각의 객체들이 다르게 행동한다는 것을 의미한다.

객체들이 다르게 행동하기 위해 모든 함수를 다른 이름으로 설정해놓는다면 매번 함수를 명명하는데에 시간을 쏟아야하므로, 프로그램의 본질에 집중하지 못하게 된다. 그러나 이러한 특성을 이용한다면 코드가 아주 간결해지게 된다. 이는 상속과 뗄 수 없는 관계를 가지고 있는데, 기반 클래스의 함수를 파생클래스에서 재정의하여 사용하는 과정이라고 생각하면 된다. 과제에서는 LinkedList 클래스를 기반클래스로 하는 새로운 파생 클래스 Stack을 정의해서 사용하고 있다. 이떄 LinkedList 클래스의 모든 멤버함수의 기능을 Stack에서도 그대로 사용하지만, 삭제 기능의 경우에는 두 클래스가 다르게 동작해야 한다. LinkedList에서 노드를 삭제하면 맨 뒤의 노드를 삭제하지만, Stack에서 노드를 삭제하면 맨 앞의 노드를 삭제한다. 이처럼 같은 삭제(delete) 명령에 두 객체가 다르게 행동해야 하기 때문에 서브타입 다형성의 성질을 이용해야 한다. 이를 위해 virtual 키워드를 활용하여 delete를 가상함수로 설정하고, Stack에서도 virtual delete로 정의하여 일종의 오버라이딩 과정을 진행해준다. LinkedList의 delete 매서드는 first 노드부터 차례로 뒤로 가서 맨 뒷 데이터를 구한다. 맨 뒷 데이터를 구하기 위해 맨 앞 노드부터 뒤로가는 과정은 비효율적이라고 생각할 수 있는데, 과제에서 현재 노드는 자신의 뒷 노드를 가리키기 때문에 이러한 과정이 불가피하다. 그리고 그 구한 값을 element에 저장해주기만 하면 과정이 끝난다. 그러나 Stack의 delete 매서드는 first라고 명명된 노드의 데이터를 element변수에 대입시키고, 두번째 노드를 first로 만들어주면 과정이 끝난다.

Print() 함수는 반복문을 활용하여 객체 내부에 있는 데이터들에 모두 접근하고, count변수를 두어, 몇번째 데이터인지 또한 체크한다. 이렇게 해야 데이터 등장 순서와 데이터 내용을 모두 출력해줄 수 있기 때문이다.