4주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231515 이름: 김다은

이 문제를 효율적으로 해결하기 위해서는 **Parameteric Polymorphisom(파라미터적 다형성)**과 **Dynamic Polymorphism(동적 다향성)**를 고려하여 코드를 작성하는 것이 핵심이다.

**1. Parameteric Polymorphisom(파라미터적 다형성)을 고려하여 기존 코드 수정**

Parameteric Polymorphisom(파라미터적 다형성)이란 동일한 함수나 코드를 여러 datatype에 대해 사용할 수 있도록 하는 기법을 말한다. 코드가 특정 datatype에 종속되지 않기 때문에 코드의 재사용성을 높일 수 있다.

먼저 Parameteric Polymorphisom을 구현하기 위해 각 class에 템플릿 변수를 사용한다. **template<typename T>**를 class Node와 class LinkedList, class Stack에 적용함으로써, 다양한 datatype를 class가 처리하도록 한다.

**2. Dynamic Polymorphism(동적 다향성)을 고려하여 코드 작성**

Dynamic Polymorphism(동적 다향성)은 객체지향 프로그래밍(OOP)에서 중요한 기법이다. 런타임에 객체의 속성에 따라 함수가 동적으로 결정되는 것을 말한다. 상속과 메서드 오버라이딩을 통해 구현할 수 있다.

본 실습에서는 아래와 같이 class Stack이 LinkedList를 상속하도록 한다.

template <typename T>

class Stack : public LinkedList<T> { ~ }

또한 class LinkedList를 보면 ` virtual bool Delete(T &element);`처럼 virtual이 사용된 것을 볼 수 있다. 가상함수란 자식 클래스에서 재정의할 것으로 기대하고 부모 클래스에서 정의해 놓은 함수이다. 생성된 가상함수는 자식 클래스에서 재정의되면 이전에 정의되었던 내용들이 모두 새롭게 정의된 내용으로 바뀐다. virtual 키워드를 사용하여 부모 클래스의 가상 함수를 재정의하는 것이 가능하다. 이렇게 되면 런타임 때 객체의 실제 타입에 따라 함수가 호출되게 된다.

template <typename T>

class LinkedList{

    ~생략~

*//맨 뒤의 원소를 삭제, 제일 나중에 들어온 원소 삭제  - LinkedList*

        virtual bool Delete(T &element);

};

**3. Stack 함수의 Delete 메서드 구현**

first, current\_size는 class의 멤버 변수이므로 this 포인터를 이용해 가져와야 한다는 점에 주의한다.

1. first가 0이면 false를 반환한다.

if(!this->first) return false;

1. element에 current의 data 값을 저장한다.

element = this->first->data;

1. 삭제할 노드를 Node<T> \*del에 저장한다.

Node<T> \*del = this->first;

this->first = this->first->link;

1. del에 할당된 메모리를 해제한다.

delete del;

1. current\_size를 1 감소시킨다.

this->current\_size--;

1. true를 반환한다.