**4주차 결과보고서**

전공: 경영학과 학년: 3학년 학번: 20190963 이름: 한다현

**1.**

이번 실험에서는 LinkedList 클래스를 템플릿을 사용해 구현하고, Stack 클래스를 LinkedList 클래스로부터 상속받아 Delete 함수를 재정의하였다. 가장 먼저 기존 LinkedList 클래스는 데이터 타입이 정수형인 경우에만 사용 가능하였지만, 템플릿을 사용하여 정수가 아닌 다른 자료형이 들어와도 처리가 가능하게 하였다 이를 위해 linked list에 사용될 노드를 구현하는 Node 클래스 위에 template<class T>를 추가한다. 그리고 Node에서 데이터를 저장하는 변수 data를 T data, 다음 노드의 주소를 저장할 포인터를 Node \*link라고 정의한다. Data의 자료형을 T라고 함으로써 data에는 정수, 실수 등의 자료형이 들어올 수 있다. 그리고 Node 클래스의 생성자인 Node(int element) 역시 Node(T element)로 바꿈으로써 정수가 아닌 다른 자료형의 element도 들어올 수 있도록 한다. 이와 같은 방식으로 LinkedList의 선언부에도 template<class T>를 추가한다. 그리고 첫번째 노드의 주소를 저장할 포인터를 Node<T> \*first로, 현재 linked list의 크기를 나타내는 변수를 int current\_size로 정의함으로써 첫번째 노드에 다양한 형태의 자료형의 데이터가 입력될 수 있으며, 현재 linked list의 크기는 정수로 나타나기 때문에 int형으로 정의한다. LinkeList의 생성자 LinkedList()는 first=0, current\_size=0으로 초기화 하고, Getsize() 함수는 current\_size의 값을 리턴한다. 생성자와 Getsize() 함수는 클래스 내에서 정의하며, Insert(T element) 함수, virtual bool Delete(T &element) 함수, Print() 함수는 클래스 외부에서 정의한다. 다음은 Insert 함수의 정의부이다.

template <typename T>

void LinkedList<T>::Insert(T *element*)

{

    Node<T> \*newnode = new Node<T>(*element*);

    newnode->link = first;

    first = newnode;

    current\_size++;

}

먼저 template<typename T>를 추가하고, LinkedList<T>와 같은 형태로 함수를 작성한다. 새로운 노드 Node<T> \*newnode를 Node<T>의 생성자를 통해 생성한다. 이후 newnode의 link 부분이 first에 저장되어 있는 노드를 가리키게 하고, first의 위치를 newnode로 옮기며 current\_size의 값에 1을 더한다. Delete 함수도 마찬가지로

template <typename T>

bool LinkedList<T>::Delete(T &*element*)

와 같이 선언하고, 마지막 노드를 찾아서 마지막 노드를 삭제하는 기능을 한다. 이는 queue와 동일하며 만약 first가 0, 즉 LinkedList에 아무 것도 들어있지 않다면 false를 리턴한다. Print 함수도

template <typename T>

void LinkedList<T>::Print()

와 같이 선언하고, LinkedList에 있는 모든 데이터를 출력한다. 이 때 index가 current\_size가 아니면 [데이터]->와 같이 출력하고, index가 current\_size와 같으면 [데이터]와 같이 출력한다. 이제 Stack.h에서 Stack 클래스를 구현한다. 먼저 클래스 선언부에 template <class T>를 추가한다. 그리고 다음과 같이 LinkedList 클래스를 상속한다.

class Stack : public LinkedList<T>

그리고 Delete(T &element) 함수만 재정의하여 사용한다. 기존 LinkedList 클래스는 queue와 같이 가장 먼저 들어온 노드를 가장 먼저 삭제했지만, Stack 클래스에서는 가장 나중에 들어온 노드를 먼저 삭제해야 하기 때문이다. Delete 함수는 다음과 같다.

bool Delete(T &*element*)

    {

*if* (this->first == 0)

*return* false;

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

*element* = current->data;

        this->first = current->link;

        delete current;

        this->current\_size--;

*return* true;

    }

먼저 this->first가 0과 같으면 Stack에 아무것도 들어있지 않을 것이므로 false를 리턴한다. 이 때 this를 사용한 이유는 LinkedList의 멤버 변수이기 때문에 this 포인터를 사용해 가져와야 하기 때문이다. 그 후에 current라는 새로운 노드를 생성해 first를 저장하고, first는 current의 link가 가리키는 노드를 가리키도록 한다. 그리고 current 노드를 삭제하고, current\_size의 값을 1만큼 감소시킨 후 true를 반환한다.