4주차 결과보고서

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

1. 실험 시간에 작성한 프로그램의 알고리즘과 자료구조를 요약하여 기술하시오.

**Parameteric Polymorphisom(파라미터적 다형성)**을 구현하여 코드의 재사용성을 높이기 위해 Template 문법을 사용하였다. int형 data만 저장할 수 있던 원래의 Node가 자료형에 의존하지 않도록 Template <typename T>를 이용해 수정하였다. Node 생성자의 매개변수에도 T를 사용하였다.

template <typename T>

class Node{

    public:

*//데이터를 저장할 변수*

        T data;

*//노드구조체 이용; 다음노드의 주소를 저장할 포인터*

        Node \*link;

Node(T element){

            data = element;

            link = 0;

        }

};

class LinkedList 또한 자료형에 의존되지 않도록 Template를 이용해 처리하였다. Node<T>를 통해 class 호출 시 명시된 T의 자료형에 따라 first 포인터를 세팅한다.

template <typename T>

class LinkedList{

    protected:

*//첫번째 노드의 주소를 저장할 포인터*

        Node<T> \*first;

        int current\_size;

~ 생략 ~

};

**Dynamic Polymorphism(동적 다향성)**을 고려하기 위해 부모 클래스에서 virtual function으로 선언된 Delete를 재정의하였다.

**bool Delete(T &element)** 함수는 아래와 같은 단계로 작동한다.

1. first가 0이면 false를 반환한다.
2. element에 first가 가리키는 data를 저장한다.
3. del 포인터에 삭제할 Node인 first를 가리키도록 한다.

del 포인터가 가리키는 메모리 공간을 해제하기 위한 작업이다.

1. first가 삭제할 노드(del)의 link를 가리키도록 한다.
2. del에 할당된 메모리를 해제한다.

삭제하고자 하는 Node이기 때문이다.

1. current\_size를 -1한다.

Node를 삭제하여 전체 Stack의 크기가 1 감소하였기 때문이다.

이 과정에서 **this 포인터**가 사용된다. this 포인터는 객체 내부에서 스스로의 메모리 주소를 가리킨다. 객체가 자기 자신을 참조하거나 자신의 멤버에 접근할 때 사용된다. 매개변수와 멤버변수의 이름이 동일 한 경우, this포인터를 사용하여 둘 중 어느 것을 의미하는지 명확하게 전달할 수 있다.

*//LinkedList class를 상속받음*

template <typename T> *//파라미터적 다형성*

class Stack : public LinkedList<T>{ *// 서브타입 다형성*

    public:

        bool Delete (T &element){ *// virtual 함수 재정의*

*//first가 0이면 false반환*

            if(!this->first) return false;

*// LinkedList와 달리 Stack은 current가 가리키는 곳을 삭제*

            element = this->first->data;

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

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

*// 삭제할 Node의 할당된 메모리 해제*

            delete del;

            this->current\_size--;

            return true;

            }

};