4주차 결과보고서

전공: 아트&테크놀로지학과 학년: 4학년 학번: 20191048 이름: 김도솔

**1.**

**template <typename T> class Node**

*(public)*

* T data : 노드의 값을 저장하는 변수.
* Node \*link : 다음 노드의 주소를 저장하는 포인터.

**template <typename T> class LinkedList**

*(protected)*

* Node<T> \*first : 첫 번째 노드의 주소를 저장하는 포인터.
* int current\_size : 리스트의 노드 개수를 저장하는 변수.

*(public)*

* LinkedList() : 생성자, first와 current\_size 변수 초기화.
* T GetSize() : 노드 개수를 리턴.
* void Insert(T element) : 리스트의 맨 앞에 노드를 삽입하는 함수.
* virtual bool Delete(T &element) : 리스트의 맨 뒤에 있는 노드를 삭제하는 함수.
* void Print() : 현재 리스트의 모습 출력.

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

Node<T> \*newnode = new Node<T>(element) : 새로운 노드 생성.

newnode -> link = first : 새로운 노드의 link가 첫 번째 노드를 가리키게 함.

first = newnode : 새로운 노드를 첫 번째 노드로 설정.

current\_size++ : 리스트의 노드 개수 +1.

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

if (first == 0) return false : first가 0일 경우(지울 노드가 없는 경우) false를 반환.

Node<T> \*current = first : 첫 번째 노드를 가리키는 포인터 current 생성.

Node<T> \*previous = 0 : NULL 포인터 previous 생성.

while 반복문 : 리스트의 끝에 도달할 때까지 previous가 current를 가리키고 current는 current의 link를 가리키며 한 칸씩 옆으로 이동. 만약 마지막 노드를 찾으면 previous가 0이 아닐 경우(리스트의 노드 수가 여러 개일 경우) previous의 link를 current의 link로 설정하고 루프 탈출, 그렇지 않으면(리스트의 노드 수가 한 개일 경우) first를 first의 link로 설정하고 루프 탈출.

element = current -> data : 삭제할 노드의 데이터 element에 저장.

delete current : 삭제할 노드 메모리 할당 해제.

current\_size-- : 리스트의 노드 개수 -1.

return true : true 반환.

**template <typename T> void LinkedList<T>::Print()**

Node<T> \*i : 노드를 가리킬 포인터 선언.

int index = 1 : 인덱스 1로 설정.

리스트의 사이즈가 0이 아닐 경우, for 반복문을 이용해 마지막 노드에 도착할 때까지 모든 노드를 탐색하며 하나씩 출력. 마지막 노드에 도착하면 노드와 줄바꿈 문자 출력 후 종료.

**template <typename U> class Stack : public LinkedList<U>**

*(public)*

**virtual bool Delete (U &element)** : Delete 함수 재정의, LinkedList와 달리 Stack은 current가 가리키는 곳을 삭제.

if(this->first == 0) return false : first가 0일 경우(지울 노드가 없는 경우) false를 반환.

Node<U> \*current = this->first : 첫 번째 노드를 가리키는 포인터 current 생성.

element = current->data : 삭제할 노드의 데이터 element에 저장.

this->first = this->first->link : 두 번째 노드를 첫 번째 노드로 설정.

delete current : 첫 번째 노드 삭제.

this->current\_size-- : 리스트의 노드 개수 -1.

return true : true 반환.