**2024 Spring OOP Assignment Report**

과제 번호 : 6

학번 : 20230642

이름 : 이채영

Povis ID : chyng

**명예서약 (Honor Code)**

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

1. **프로그램 개요**
   * 본 프로그램은 Head가 여러 개 있는 Doubly linked list를 template통해 구현한 프로그램이다. 또한 구현한 multiheadlist를 이용해 n단 논법을 구현하였다.
   * Doubly linked list의 노드는 이전 노드를 가리키는 prev(Node\*), 정보를 담고 있는 data(T), 다음 노드를 가리키는 next(Node\*)로 구성된다. 제일 첫 노드를 ehad라 부르며, 제일 마지막의 노드는 tail이라 부른다. Head의 prev와 tail의 next는 nullptr을 가리킨다.
   * Multiheadlist는 head가 여러 개인 doubly linked list이다. Head의 주소는 headList에서 관리된다.
2. **프로그램의 구조 및 알고리즘**

* Class 설명:
  + MultiHeadList 클래스
    - 변수:
      * headList: Head의 주소가 담겨 있는 vector
    - 함수
      * Void push\_back(const T& data, int headIdx = -1) : headIdx번째 list 맨 끝에 data를 추가한다. headIdx가 음수이거나 headList에 저장되어 있는 head의 인덱스를 초과하는 값이 들어올 때는 새로운 head를 생성하여 추가한다.
      * Void push\_front(const T& data, int headIdx=-1): headIdx번째 list 맨 앞에 data를 추가한다. headIdx가 음수이거나 headList에 저장되어 있는 head의 인덱스를 초과하는 값이 들어올 때는 새로운 head를 생성하여 추가한다.
      * Void insert(Iterator pos, const T& data): Iterator pos가 가리키는 곳 이전에 data를 삽입한다.
      * Void pop\_back(int headIdx): headIdx번째 list 맨 끝 데이터를 삭제한다.
      * Void pop\_front(int headIdx): headIdx번째 list 맨 앞 데이터를 삭제한다.
      * Void merge(int frontHeadIdx, int backHeadIdx): frontHeadIdx번째 List 뒤에 backHeadIdx번째 List를 붙인다.
      * Bool erase(const T& data, int targetHeadIdx): targetHeadIdx번째 list에 data가 있다면 삭제한다. 이때 head부터 시작하여 제일 처음 발견된 data만 삭제한다.
      * Int headSize(): headList의 크기를 반환한다.
      * Iterator begin(int headIdx): headIdx번째 list의 begin()
      * Iterator end(): nullptr 반환
      * ReverseIterator rbegin(int headIdx): headIdx번째 list의 rbegin(), headIdx번째 list의 tail부터 역순으로 움직임.
      * ReverseIterator rend(): nullptr 반환
  + Iterator / ReverseIterator: MultiHeadList 내부에서 정의되는 클래스
    - 변수
      * Curr: List에서 현재 노드를 가리키는 포인터
    - 함수
      * Iterator / ReverseIterator : 생성자
      * Operator++:전위/후위 연산자
      * Operator--: 전위/후위 연산자
      * Operator+(int n): n번 ++ 연산
      * Operator-(int n): n번 – 연산
      * Operator!=: != 구현
      * Operator==: == 구현
      * Operator\*(): curr의 data를 출력
      * Operator->: -> 구현
      * Node<T>\* access\_data: Iterator가 가리키는 노드에 접근
    - Syllogism
      * 변수
        + Syl: 논리가 저장되는 MultiHeadList
      * 함수
        + Void put(const std::pair <std::string, std::string& argument>: 논법을 넣고 해당 논법이 잘 연결될 수 있도록 구성.
        + Void qna(const std::string& q): 질문 q를 던졌을 때 n단 논법을 거친 후 최종 결론을 대답한다.(최종 결론이 p일 때 “If q, then p” 출력)
        + Void print(): MultiHeadList의 전체적인 상황을 표현한다.

1. **토론 및 개선**
   * + 본 프로그래밍 과제를 통해 Doubly Linked List 및 Multi head list의 기본 구조를 이해하고, 데이터 삽입, 삭제, 리스트 병합 등 연산을 구현해볼 수 있었다. 또한 클래스 내에 클래스를 구현하는 것을 실습해볼 수 있었고, template을 활용해 재사용성이 높은 프로그램을 구현할 수 있었다.
     + 이전 Assignment에서 Singly linked list를 구현했었는데, doubly linked list과의 차이점을 비교해볼 수 있었다. Doubly linked list는 singly linked list와 비교했을 때 노드 하나하나가 저장하는 정보가 많지만 데이터 삽입, 삭제를 하기에는 더욱 용이하였다.
2. **참고 문헌**
   * + 1. <https://en.cppreference.com/w/cpp/language/templates>
          1. template를 이해하는 데 위의 사이트를 참고하였다.