2024 Spring OOP Assignment Report

과제 번호 : 6-1, 2
학번 : 20230499
이름 : 김재환
Povis ID : carotinoid

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

1번 문제에서는, 템플릿을 이용해 임의 타입의 자료를 저장할 수 있는 이중 연결 리스트의 헤드를 벡터에 저장하여 MultiHeadList를 구현한다. 다중 헤드 리스트에는 각 이중 연결 리스트의 가장 앞 노드를 저장하는 벡터를 가지며, 자유롭게 크기가 변할 수 있다. 이중 연결 리스트의 처음과 끝 원소의 주소는 각각 Head, Tail 포인터에 저장되며, 양 끝에 더미노드는 넣지 않았다. 이러한 추상 자료구조를 구현하기 위한 함수를 구현하는 것을 목표로 한다.
2번 문제에서는, 이전에 구현한 MultiHeadList를 이용하여 n단 논법을 연결해주는 프로그램을 작성한다. A -> B 의 관계를 (A, B) 와 같이 문자열 쌍으로 저장하며, 각 명제의 관계 쌍을 입력받은 후, 쿼리로서 하나의 명제를 입력받으면 명제로 도출할 수 있는 마지막 결론에 해당하는 명제 문자열을 출력해야한다. 이때 순환이나 중복 문제는 생기지 않는다고 가정한다.

2. 프로그램의 구조 및 알고리즘

- 헤드에 들어갈 각 이중 연결 리스트는 Node 구조체와 포인터를 이용해 MultiHeadList 클래스 내에 구현되었으며, 이중 연결 리스트를 순회하기 위한 이터레이터 클래스 또한 MultiHeadList 클래스 안에 선언되어 있다. 즉, MultiHeadList는 중첩클래스로서 두 종류의 이터레이터를 가지지만, Node는 중첩되어 있지 않다.
- 각 클래스에 대한 설명, 각 멤버함수에 대략적인 설명은 과제 파일에 제시되어 있으므로, 코드 작성시 고려한 부분에 대해서만 작성하였다.
  - Node 구조체 : 템플릿 타입 T형 데이터와 이전, 다음 노드의 포인터를 변수로 갖는다. T형 타입인 데이터를 받아 생성자에서 데이터를 초기화한다.
  - MultiHeadList 클래스 :
    - 1. 일반적으로 리스트를 구현할 때 원소가 없는 경우의 예외처리를 하기 위해서 head와 tail에 더미 노드를 넣기도 하지만, 이번 과제에서는 더미 노드 없이 각 경우

에 예외처리를 해주었다.

- 2. Iterator: 이터레이터는 하나의 이중 연결 리스트 내부 요소를 순회하기 위한 중첩 클래스이다. 이터레이터는 현재 노드를 가르키는 포인터와 연산자 오버로딩, 생성자를 갖는다. 이때 현재 노드를 가르키는 포인터는 private 구역이 아닌 public 구역에 선언하였다. public 구역에 선언함으로써 이터레이터 외부의, multiheadList의 멤버 함수에서 접근하여 포인터를 비교할 수 있다.

- 3. 이중 연결 리스트 앞이나 뒤에서 순차적으로 요소를 삽입, 제거하는 push_front, push_back, pop_front, pop_back 함수를 구현할 때 원소가 없는 경우를 고려해야한다. 삽입시 원소가 없는 경우 새로운 헤드를 만들어 벡터에 삽입해야하며, 제거한 후 리스트에 원소가 없는 경우 벡터에서 해당 원소를 제거해야한다.

- 4. Erase 함수는 특정 원소를 제거하고, 앞과 후의 리스트를 분리하는 함수이다. 한번 오버로딩 되어 두가지 유형으로 작동할 수 있다. erase 함수가 작동할 때, 가능한 경우를 4가지로 나누어 작성하였다.
  1. 해당 이중연결리스트에 원소가 하나인 경우, 즉 head == target == tail인 경우는 원소를 제거한 이후 해당 인덱스를 벡터에서 제거한다.
  2. 지우고자 하는 원소가 리스트의 맨 앞에 있을 경우, 지운 다음 벡터가 가르키는 헤드를 다음 원소로 이동시킨다.
  3. 지우고자 하는 원소가 리스트의 맨 뒤에 있을 경우, 원소를 제거한 후 테일을 이전 원소로 이동시킨다.
  4. 지우고자 하는 원소 target이 head도 아니고 tail도 아닌 경우, 원소를 지운 후 타깃의 이전 원소를 원래 리스트의 마지막으로 만든다. 타깃 뒤에 있는 원소는 하나씩 제거하면서 새로운 리스트에 삽입한다.

3. 토론 및 개선

- 이번 과제에서는 C++ 표준 라이브러리를 사용해 보는 것을 목표로 하며, 벡터와 쌍(pair) 라이브러리르 사용하였다. 벡터는 배열과 매우 유사하지만, 편리한 방법으로 동적으로 배열의 크기를 조절할 수 있는 자료구조이다. 그러나 벡터를 이용한 탐색은 선형이므로 O(N) 시간복잡도를 갖는다는 단점이 있다. pair은 원하는 타입 두개를 묶어서 저장할 수 있는 자료구조로, first와 second를 통해 각 자료에 접근할 수 있다.
- 증가 연산자를 오버로딩하기 위해 두가지 연산자를 오버로딩해야한다.
  1. Iterator& operator++() (전위 증감 연산자)
  2. Iterator operator++(int) (후위 증감 연산자)
  후위 증가 연산자의 매개변수는 사용되지 않으며, 또한 레퍼런스를 반환하지 않는다. 이는

감소 연산자도 마찬가지이다.

- 내부적으로 이중연결리스트를 구현하기 위해서 헤드와 테일에 더미 노드를 추가함으로써 예외 처리를 간단하게 하고 메모리 런타임 에러의 위험을 줄일 수 있다. 또한 포인터 대신 스마트 포인터를 사용함으로써 메모리 누수의 위험을 줄일 수 있다.

4. 참고 문헌

□ Lecture note.