**2024 Spring OOP Assignment Report**

과제 번호 : 3-1

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 element들이 오름차순으로 정렬되어 있는 OrderedList를 구현한 프로그램이다. OrderedList에는 integer value가 저장되어 있다.
   * OrderedList의 주요 기능으로는 element를 추가하고, 삭제하고, element에 index를 통해 접근하고, value가 list에 존재하는지 확인하고, list의 크기를 확인할 수 있다.
2. **프로그램의 구조 및 알고리즘**

* Struct 설명:
  + Node
    - value: 노드에 저장된 정수 값
    - next: 다음 노드를 가리키는 포인터. List의 마지막 노드에서는 NULL을 가짐.
  + OrderedList
    - m\_size: 리스트의 크기를 나타내는 정수. add, remove 함수가 사용될 때마다 업데이트됨.
    - head: 리스트의 첫 번째 노드를 가리키는 포인터. 리스트가 비어있으면 NULL을 가짐.
* 함수 설명:
  + void add(OrderedList\* ordered, int v) : 오름차순을 유지하며 new element v를 list에 추가한다.
  + void add(OrderedList\* ordered, const int\* arr, int size): integer values를 가지는 array를 받아 오름차순을 유지하며 list에 값들을 추가한다. 각 요소는 add(OrderedList\* ordered, int v) 함수를 통해 개별적으로 추가된다.
  + void remove(OrderedList\* ordered, int index) : 주어진 index에 해당하는 element를 삭제한다. Index는 0부터 시작하며, head 노드는 index 0에 해당한다.
  + int size(OrderedList\* ordered) : OrderedList의 element 수, 즉 list의 크기(m\_size)를 반환한다.
  + bool contains(OrderedList\* ordered, int v): 주어진 element v가 list에 존재하는지 확인한다. 있으면 true, 없으면 false를 반환한다.
  + int getValue(OrderedList\* ordered, int idx): 주어진 index에 해당하는 element value를 반환한다. 인덱스가 리스트의 범위를 벗어나면, std::numeric\_limits<int>::min()을 반환하여 오류 상황을 나타냅니다.
* 변수 설명
  + new\_node: 새로 추가될 노드
  + current: 현재 순회 중인 노드를 가리키는 포인터
  + previous: current 노드의 바로 이전 노드를 가리키는 포인터
  + cnt: 순회 중인 노드의 index를 나타냄

1. **토론 및 개선**
   * + 본 프로그래밍 과제를 통해 Linked List의 기본 구조를 이해하고, 데이터 삽입, 삭제, 검색 등 기본적인 연산을 구현해볼 수 있었다. 또한 c++에서 ‘new’와 ‘delete’를 사용하여 동적으로 메모리를 할당하고 해제하는 방법을 배울 수 있었으며, 메모리 누수를 방지하기 위해 할당된 메모리를 모두 적절히 해제하는 과정의 중요성을 이해할 수 있었다.
     + 본 프로그램에 추가할 수 있는 기능으로는 list의 내용을 역순으로 출력하는 기능, 정렬 알고리즘을 선택할 수 있는 기능 등이 있다.
     + 본 프로그램의 실행 속도를 높이기 위한 방법으로, doubly linked list를 사용하는 방법이 있다. 본 프로그램에서는 linked list를 구현할 때 각 노드에 value, 다음 노드의 포인터만 저장하는 singly linked list를 사용했는데, 각 노드가 이전 노드와 다음 노드를 모두 가리키는 doubly linked list를 사용하면 remove, add의 기능을 더욱 효율적으로 수행할 수 있을 것이다.
2. **참고 문헌**
   * + <https://www.geeksforgeeks.org/singly-linked-list-definition-meaning-dsa/?ref=outind>
       1. Singly linked list를 이해하는 데 위의 사이트를 참고하였다.