중간고사 대체 과제 : 검열프로그램 작성 202012468 김현서

1. 문제에 대한 자신의 알고리즘 설명

저는 우선 main 함수에서 Stack을 4개 선언했습니다. 스택의 경우 후입선출/LIFO(Last-In, First-Out) 구조라 나중에 들어간 데이터가 먼저 나오는 구조입니다. 그렇기 때문에 문제에서 주어진 알고리즘을 이용해 검열을 하려면 입력받은 단어(A)와 문장(T) 각각 스택이 두 개씩 필요하다고 생각했습니다.(여기서 알고리즘은 : ①T에 A가 없으면 알고리즘을 종료한다. ②T에서 처음 등장하는 A를 찾은 뒤, 삭제한다. ③T에 A가 없으면 알고리즘을 종료한다. ④T에서 마지막으로 등장하는 A를 찾은 뒤, 삭제한다. ⑤1번으로 돌아간다) 두 스택은 입력받은 단어(A)와 문장(T)을 각각 순서대로 입력받아 역순으로 저장할 stack_reverse_A, stack_revers_T와 역순으로 저장된 각각의 stack_reverse 스택들을 입력받은 순서대로 저장할 stack_A, stack_T으로 구성되어 있습니다.

사용할 스택을 선언한 후, StackInit 함수를 이용해 초기화를 시켜두고, str_T와 str_A을 문제에서 주어진 크기(각각 300000, 25)를 최대로 가질 수 있게 초기화를 시킵니다. 다음으로 10개 이상의 단어를 입력하기 위해 fgets 함수를 사용해 str_T와 str_A에 각각 문장과 단어를 입력합니다.

이후 배열로 입력받은 단어(A)와 문자(T)를 반복문과 SPush 함수를 이용해 각각의 reverse 스택에 집어넣습니다.(→HW4_202012468.c 파일 안에서 Input_Data 함수에 해당)

그 다음, 순서를 맞춰주기 위해 반복문과 SPush 함수를 이용해 각각의 정방향 스택에 집어넣습니다.(→HW4_202012468.c 파일 안에서 Sequencing_Data 함수에 해당)

그리고, 문제에서 주어진 알고리즘을 이용해 검열을 합니다.(→HW4_202012468.c 파일 안에서 Remove_Data 함수에 해당) 이 함수 내에서도 스택을 2개 선언하고 초기화합니다. 문장 안에 찾는 단어가 있을 때 하나는 문장의 스택을 보관할 곳이고, 하나는 단어의 스택을 보관할 곳입니다. 값이 입력된 문장 스택, 빈 문장 스택, 값이 입력된 단어 스택, 빈 단어 스택을 함수의 인수로 입력하고, 값이 입력된 문장 스택의 길이를 int 변수에 저장해 둡니다.

그리고 반복문이 시작됩니다. 값이 입력된 문장과 값이 입력된 단어가 빈 스택이 되지 않았다면, 값이 입력된 문장과 값이 입력된 단어의 맨 위 스택 값을 비교하고, 만약 이게 같다면 맨 위 스택을 SPop을 이용해 문장은 temp_sentence스택, 단어는 temp_word 스택에 잠시 보관해둡니다.(이때 temp_sentence와 temp_word는 원래 입력된 문장과 단어와는 역순이 됨) 만약 원래 단어나 원래 문장이 빈 스택이 되거나, 맨 위 스택이 같지 않을 때 SPop 함수를 이용해 temp 스택에 데이터를 넣는 걸 멈춥니다. 만약 원래 단어 스택이 비어있지 않다면 아직까진 원래 문장에 찾는 단어가 없었다는 이야기가 되므로, temp_sentence에 저장한 스택을 다시 원래 문장에 돌려놓고(원래 순서대로), temp_word에 저장한 스택은 원래 단어에 돌려놓습니다. 하지만 만약 단어 스택이 비어있다면, 원래 문장에 찾는 단어가 있었다는 말이므로 temp_word에 담아두었던 단어들을 원래 단어 스택에 넣었다가 새로운 단어 스택으로 옮깁니다.(그래야 원래 단어와 역순이 됨) temp_sentence의 경우 제거해야하는 단어들이므로 그대로 넣어두고, 원래 문장 스택에 있던 스택들을 새로운 문장 스택으로 옮깁니다.(원래 단어와 역순) 만약 원래 문장이 비어있지 않다면 SPush를 이용해 원래 문장의 맨 위 스택을 새로운 문장 스택으로 옮깁니다. 하지만 만약 원래 문장이 비어있다면, temp_word에 담아두었던 단어들을 다시 원래 단어에 담고, 원래 단어 스택에 담긴 단어들을 새로운 단어 스택에 담습니다.(원래 단어와 역순)

만약 while문이 실행된 후에 새로 생성된 새로운 문장 스택의 개수와 while문 실행 전 int 변수에 저장해 두었던 원래 문장의 스택 개수가 같지 않다면 Remove_Data 함수에서 인수 위치만 바꿔서 다시 진행합니다.(원래 문장 스택과 새로운 문장 스택의 위치 바꿈, 원래 단어 스택과 새로운 단어 스택 위치 바꿈) 이렇게 하면 처음엔 입력된 문장 맨 앞에서부터 입력된 단어를 순서대로 제거할 수 있고, 다음으로는 입력된 문장 맨 뒤에서부터 입력된 단어를 역순으로 제거할 수 있습니다. 그리고 이 과정은 입력된 문장에서 입력된 단어들이 발견되지 않을 때까지 진행될 겁니다.

Remove_Data 함수 진행이 끝난 후, 만약 stack_T(입력받은 문장의 정방향)에 데이터가 저장이 되어있지 않다면, stack_reverse_T(입력받은 문장의 역방향)에 데이터가 저장되어 있다는 말이므로, Sequencing_Data 함수를 이용하여 stack_reverse_T에 저장된 데이터들을 stack_T에 순서를 바꿔 반대방향으로 저장합니다.

저장한 후, 삭제 후의 단어를 출력하기 위해 SPop을 이용해 stack_T의 값을 하나씩 뽑으며 출력해냅니다.

2. 수업시간에 배운 스택의 ADT 변경 시 무엇을 왜 변경했는지 설명

우선 헤더파일에서 연결리스트 기반 스택을 표현한 구조체를 typedef로 정의할 때 Node* head;만 선언하는 게 아니라, int numOfData;도 선언해 숫자를 셀 수 있게 만들었습니다. 그리고 int SCount(Stack* pstack); 함수를 선언하고, c파일에서 정의했습니다. 우선 void StackInit(stack* pstack);은 스택을 초기화시키는 함수이기 때문에 numOfData = 0;을 적어주었습니다. 그리고, void SPush(Stack* pstack, Data data)함수의 경우 리스트의 머리에 새 노드를 하나 추가하는 개념이기 때문에 (numOfData)++;를 적어 1씩 추가했습니다. 반면, void SPop(Stack* pstack)의 경우 노드를 하나 소멸시키는 개면미기 때문에 (numOfData)--;를 적어 1씩 감소시켰습니다. 마지막으로, int SCount(Stack* pstack) 함수를 이용해 numOfData를 반환할 수 있게 했습니다.

스택의 개수는 main 함수에서 Remove_Data 함수를 이용하기 위해 꼭 필요합니다. 반복문 통과 이전 문장의 스택 수와 반복문 통과 이후 문장의 스택 수를 비교해야 문장 안에 단어의 유무를 판단할 수 있기 때문입니다.