Data Structure Project 1

<VOCABULARY PROGRAM>

학과: 컴퓨터공학과
담당교수: 이기훈 교수님
실습분반: 목요일 0, 1, 2
학번: 2015722039, 2015722030
이름: 정현우, 박우혁

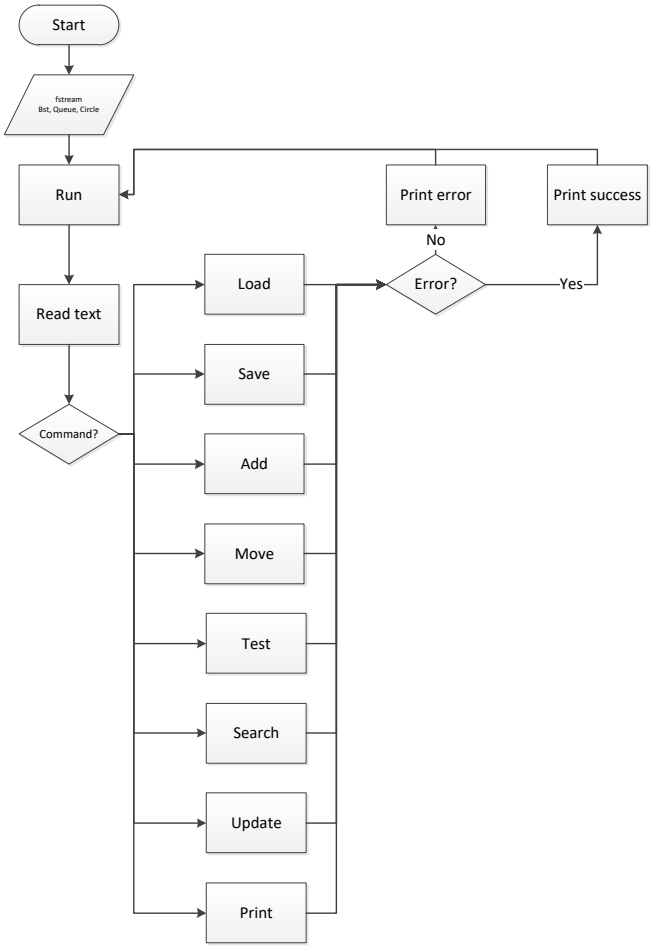
목차

Introduction

이번 프로젝트의 목적은 영어단어를 외우기 전, 외우는 중, 외운 후의 단어장들을 각각 나누어 이들을 공부할 수 있는 영어 단어장 프로그램을 만드는 것이 목적이다.

각각의 단어장들은 다른 형태의 데이터 저장 방식을 사용하는데 이진 탐색 트리 (Binary Search Tree, BST), 환형 연결 리스트 (Circular Linked List), 큐 (Queue)를 이용해 각각의 영어 단어장 프로그램을 구현한다. 이들의 데이터들은 각각의 텍스트 파일에서 불러오며 만약 중복된 단어가 있을 경우 자동으로 삭제를 해 준다. 각각의 데이터 저장 형태로 구성된 각각의 단어장들의 데이터는 서로 독립되어 있다. 또한, 저장된 단어 자료들을 외우기 전, 외우는 중, 외운 후의 한 방향으로만 이동하게 된다. 데이터들은 class 안에 구현된 함수를 통해 전달된다.

Flowchart



<그림 – Flowchart>

Algorithm

외우기 전 (To Memorize)

외우기 전 단어들은 Queue 를 사용한 데이터 저장방식을 사용해 저장을 한다. Queue 는 대표적인 First-in-First-out 방식으로 단어를 Queue 에 저장한 순서대로 단어가 빠져나오게 된다. Queue 는 원래의 헤더 함수가 존재하지만 이번 프로젝트에서는 직접 구현했으며 자료형을 template 로 사용하지 않고 Word Node 형 즉, 단어 저장을 하는 변수형으로 한정을 지었다.

Queue 에 존재하는 멤버 함수는 종류는 여러가지가 있다.

Queue 의 멤버 함수	기능
void Push(void)	Queue 에 자료를 넣어주는 함수
Word Node *Pop(void)	Queue 에 저장 되어 있는 자료를 출력해 주는 함수
Word Node *Search(char*)	Queue 에 저장되어 있는 영단어와 일치하는 노드의 위치를 출력해 주는 함수
bool Print(void)	Queue 에 저장 되어 있는 모든 노드들의 영단어와 뜻을 콘솔창과 log text 에 출력 해 주는 함수
bool Save(void)	Queue 에 저장 되어 있는 모든 노드들을 to_memorized_word.txt 에 저장 해 주는 함수
bool Empty(void)	Queue 에 저장 되어 있는 노드가 존재하는지 판별 해 주는 함수로 없다면 1 의 값을 출력 해 준다.
WodeNode *Front(void)	Queue 의 제일 첫번째 노드 즉, Queue 의 head 를 출력 해 주는 함수

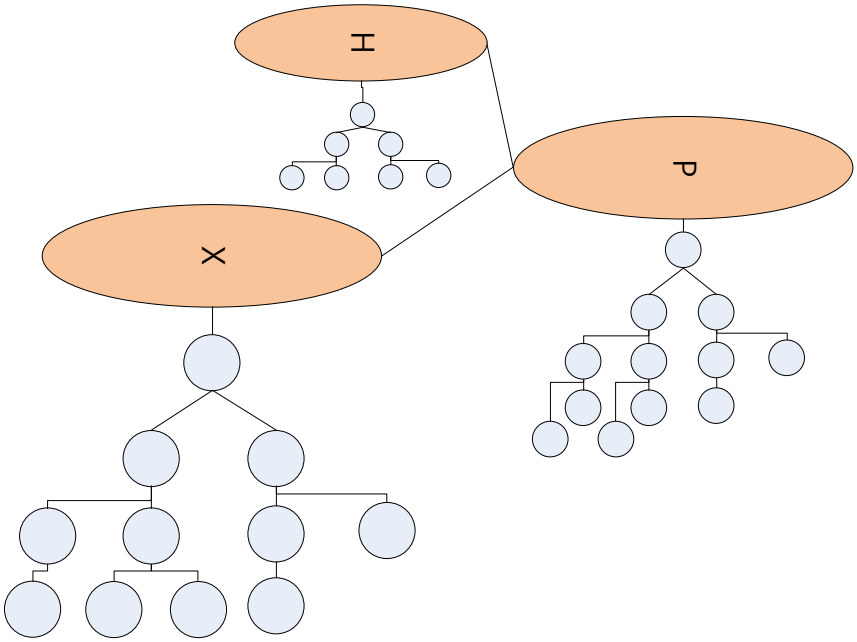
< 표 – Queue's member function >

외우기 전의 단어들은 외우는 중의 단어장으로 이동할 수 있으며 이때 이동시켜주는 함수는 Manage 클래스 안에 있는 Move 가 수행 해 준다.

외우는 중 (Memorizing)

외우는 중의 단어장은 이진 탐색 트리를 사용해 데이터를 저장 해 주었다. 이진 탐색 트리는 한 개의 부모 노드에 2 개의 자식 노드가 있으며 이 노드들이 위에서 아래로 이어져 있는 형태이다. 이러한 형태는 자식이 둘이며 나무가 거꾸로 서있는 형태이기 때문에 Binary Search Tree 구조라 하며 제일 위의 부모는 root 라 한다.

영단어들을 저장 하기 위해 BST 구조를 만들어야 하는데 알파벳의 첫 글자를 사전처럼 찾기 쉽게 하기 위해 첫 글자를 노드로 해 찾을 수 있는 알파벳 BST 를 만들어준다. 이렇게 만들어진 알파벳 BST 는 단어들을 저장할 워드 BST 의 root 가 되었다.



< 그림 – Alphabet Node 와 Word Node>

그림을 보면 알파벳과 단어 노드를 형상화 한 것이다. 위의 타원은 알파벳 노드를 표현한 것으로 타원형의 노드들이 서로 연결 되어 있다. 아래의 원은 워드 노드를 표현 했으며 이 원들이 실제 단어들이 저장 된 형태이다. 간략하게 표현한 그림이지만 실제 프로그램에서는 타원에 저장 되어 있는 알파벳을 찾은 뒤 원하는 단어를 찾아 낼 수 있다.

외우는 중의 단어장에는 저장을 위하여 4 개의 클래스가 뭉쳐 있다. 단어를 저장할 노드인 WordNode, 노드를 묶어줄 WordBST, 단어들의 첫번째 알파벳을 비교해 저장해 줄 AlphabetNode, 이 알파벳 노드들을 묶어 줄 AlphabetBST 가 있다.

BST 는 기존의 Linked List 와는 다른 방식으로 출력을 해 주어야 한다. BST 안의 노드들은 서로 모두와 연결되어 있는 것이 아니기 때문에 다른 방식으로 출력 해 주어야 하는데 출력

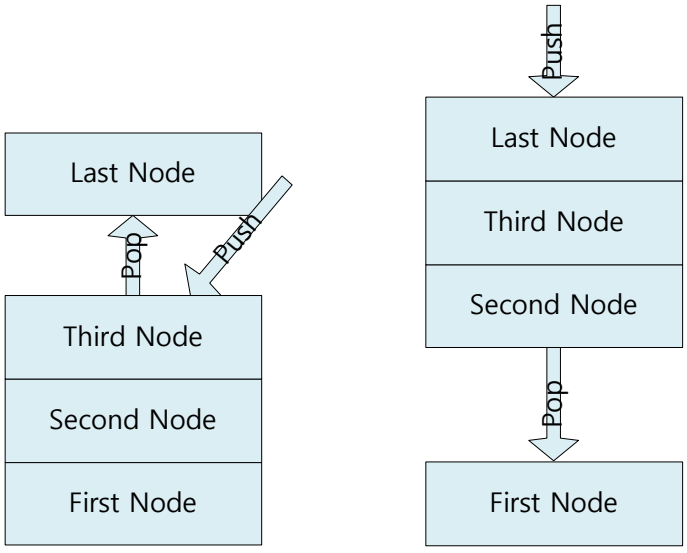
함수의 종류는 4 종류가 있다. Pre-order, In-order, Post-order, Level order 인데 level-order 를 제외한 3 가지의 함수들은 재귀적인 방식으로도 반복하는 방법으로도 출력을 할 수 있다.

출력 방식	기능
Pre-order	자신의 노드를 출력하고 왼쪽의 자식을, 마지막엔 오른쪽 자식의 노드를 출력하는 방식이다.
In-order	왼쪽의 자식 노드를 출력하고 자신의 노드를 출력하고 마지막엔 오른쪽 자식의 노드를 출력하는 방식이다.
Post-order	왼쪽의 자식 노드를 출력하고 오른쪽의 자식 노드를 출력한 뒤 마지막에 자신의 노드를 출력하는 방식이다.
Level-order	위에서부터 차례대로 모든 노드들을 출력해 주는 방식이다

< 표 – BST 의 출력방식 >

재귀적인 방식은 간단하게 구현할 수 있지만 반복적인 방식은 읽어온 노드들을 저장해 주어야한다. 이때 필요한 데이터 저장 방식이 stack 이다.

Stack 은 위에서 설명한 Queue 와 마찬가지로 대표적인 저장 방식으로 First-in Last-out 의 저장 방식이다. Stack 은 접시를 쌓은 것처럼 처음 넣어준 노드는 제일 마지막에 출력 해 줄 수 있다. 반복적인 방식의 출력의 함수들은 특정 방향의 자식들로 내려가면서 반대쪽 자식들을 stack 에 저장을 해주어 이전 노드들로 돌아 갈 수 있다.



< 그림 – Queue 와 Stack 의 모형 (좌 – Queue, 우 - Stack) >

외운 후 (Memorized)

외운 후의 단어장은 외우는 중의 단어장에서 Test 를 마친 단어들과 Text 파일에서 불러온 단어들이 Circular Linked List 의 형태로 저장하게 된다. 이들은 기본 linked list 와 형태는 동일하지만 list 의 제일 마지막 노드가 제일 앞에 있는 list 의 헤드와 연결이 된다.

Circled Linked List 의 맴버함수이다.

Circled Linked List 의 멤버 함수	기능
void insert (WordNode * node)	Linked List 에 자료를 넣어주는 함수
WordNode *Search(char *word)	Linked List 에 저장 되어 있는 자료를 입력한 단어의 노드의 주소를 출력 해주는 함수
bool Print(void)	Linked List 에 저장 되어 있는 자료를 모두 출력해 주는 함수
bool Save(void)	Linked List 에 저장 되어 있는 자료들을 텍스트 파일로 저장해 주는 함수

< 표 – Queue's member function >

Manager Class

Manager 클래스는 전체적인 데이터들을 모두 관리 해 주는 함수이다. 모든 데이터들의 Load, Add, Move, Save 를 관리하며 단어장 간의 데이터 교환인 Move, Test 저장 되어있는 단어들을 관리해 줄 update 출력 해 줄 print 함수들이 구현 되어있다. 또한 모든 함수들이 오류가 생길 경우 에러를 출력 해 줄 error 함수도 포함이 되어있다. 각각의 함수가 작동을 하지 않을 경우, 함수가 작동을 할 수 없을 경우 오류 코드를 출력을 해 준다.

Move 와 Test 함수들은 단어장 사이의 데이터를 교환 시켜주는 역할을 한다. 외워야 할 단어들이 저장 되어있는 단어장에서 외우고 있는 단어장에 단어를 전달해 주고 기존의 단어 노드는 삭제해 준다. Test 는 프로그램의 목적이 단어를 외우기 위한 프로그램이므로 단어를 외웠는지 외우지 못했는지 시험을 쳐 외웠다면 외운 단어장으로, 외우지 못했다면 단어를 그대로 둔다.

Result screen

Commend example 1

```
LOAD ds
ADD
MOVE 30
PRINT MEMORIZING R_IN
UPDATE area 구역
SEARCH area
TEST area 구역
PRINT MEMORIZIG I_IN
PRINT MEMORIZED
SAVE
EXIT
```

< 사진 – commend text example file 1 >

```
======== ERROR ========
         100
======================
======== ADD========
         success
======================
======== MOVE========
         success
======================
======== PRINT ========
activity 활동
bird 새
clothes 옷
different 다른
each 각각
earth 지구
example 예
fire 불
fun 재미
hand 손
important 중요한
job 직업
learn 배우다
letter 편지
```

```
work 일
======================
======== UPDATE ========
area지역->구역
======================
======== UPDATE========
         success
======================
======== SEARCH ========
area 구역
======================
======== ERROR ========
         500
======================
======== ERROR ========
         700
======================
======== ERROR ========
         700
======================
======== ERROR ========
         400
======================
```

< 사진 – result screen of commend example 2>

Load 의 명령어는 다른 인자가 들어가면 안되기 때문에 Error 코드 100 을 출력한다. Add 와 Move 의 함수는 성공적으로 수행되었다 출력을 해 준다. Memorizing 단어장에서 재귀적인 방식이며 in-order 방식으로 단어들을 출력 해 준다. Area 라는 단어를 업데이트한다. 그 다음 area 의 기존의 뜻으로 test 하면 단어의 뜻이 바뀌었기 때문에 에러코드가 나오게 된다. 단어장이 오타가 날 경우 에러를 출력해 준다. Save 할 내용이 없기 때문에 save 오류가 나오고 프로그램이 종류 된다.

Commend example 2

```
LOAD
ADD
PRINT MEMORIZING R_IN
MOVE 15
PRINT MEMORIZING I_POST
UPDATE DataSturucture 대구
SEARCH miracle
TEST plant 식물
PRINT MEMORIZIG I_LEVEL
PRINT MEMORIZED
SAVE
EXIT
```

< 사진 – commend text example file 2 >

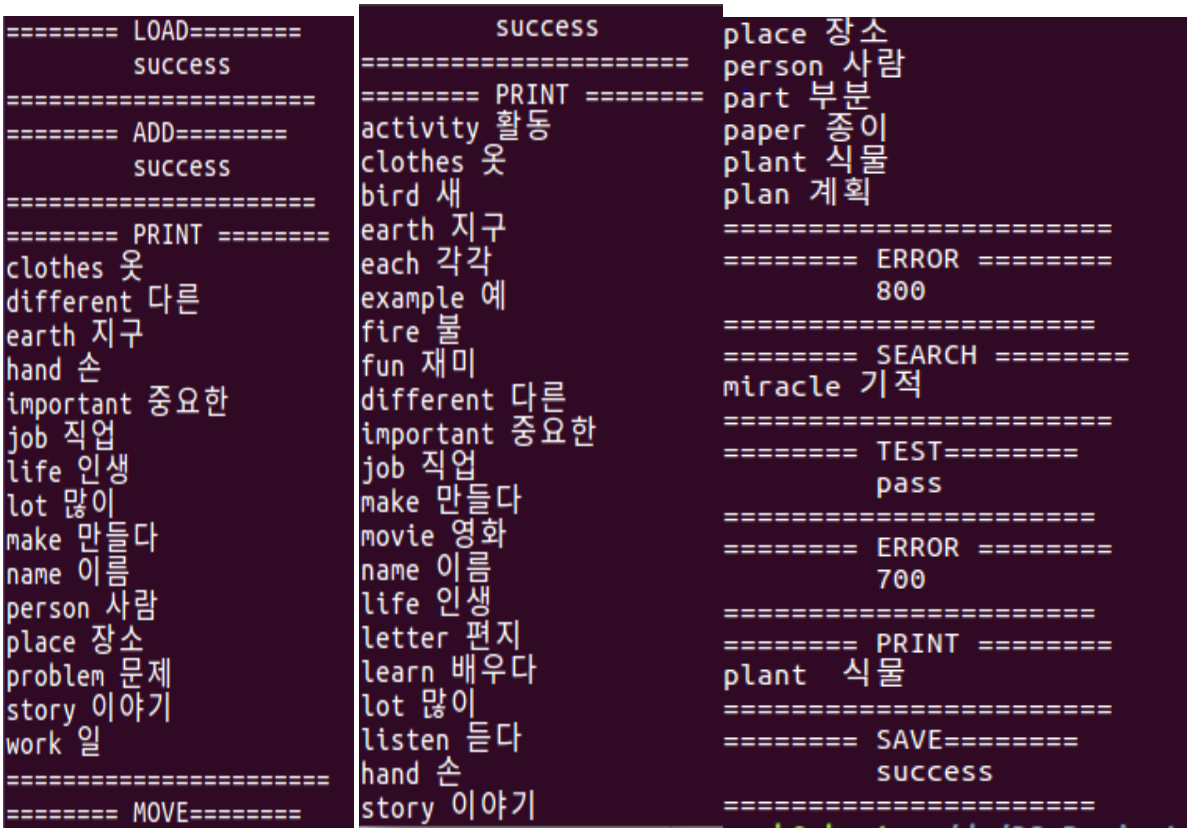


사진 – result screen of commend example 2>

LOAD 명령은 인자의 값이 으므로 success 되었다. 또한 ADD 도 추가 명령이 없으므로 success 되었고 PRINT 도 명령인 I_IN 오더이므로 왼쪽부터 오른쪽으로 모든 word 를 Print 한다. 위 command 를 실행할 때 memorizing_word 만 없는 상태였으므로 ERROR 를 출력하지 않고 LOAD 되었고 또한 100 이 넘지 않으므로 MOVE 도 success 되었다.

그 후 POST order 에 따라서 출력되고 UPDATE 는 없는 데이터 이므로 ERROR 가 출력된다.

SEARCH 는 miracle 을 찾아 출력하고 TEST 에서도 plant 는 MEMORIZING 에 있는 단어이므로 succescc 되어 MEMORIZED 로 이동한다. 그 후 PRINT 는 명령어가 틀렸으므로 ERROR 가 출력되고 PRINT MEMORIZED 에서 MEMORIZED 의 유일한 내용인 plant 이 출력되고 SAVE 된 후 끝난다.

Consideration

정현우

Print Logics, Word and Alphabet node, Word BST, Circled linked list, Report

이번 프로젝트를 진행하면서 처음으로 프로그래밍 팀플을 한 것은 아니지만 복잡한 알고리즘을 생각하고 회의 하며 구현 한 것은 처음 이였다. 기존에 친하게 지내던 동기와 팀을 이루어 과제를 했기 때문에 팀원 간의 불화 때문에 생기는 불화는 없었다. 하지만 서로의 코드 작성 스타일이 달라 수정을 요청할 때면 서로의 코드를 수정할 일이 빈번하게 생기곤 했다. 서로 열심히 노력해 만든 코드이지만 서로가 남의 코드를 봤을 때에는 복잡하며 이해 하기 힘들다는 생각부터 했던 것 같다. 그래서 서로의 코드를 수정 하거나 삭제를 해야할 때 자신이 짠 코드가 아니 였기 때문에 조심스러웠다.

기존에 배웠던 자료 저장 방식이 아닌 BST 라는 배우기 전에는 어려웠던 자료형 구조를 배워 우리가 직접 구현해 보는 것이 이번 프로젝트의 기본 목적이다. 이번 프로젝트를 진행하면서 목적은 달성을 한 것 같다. 하지만 수업 때 배우지 않았던 반복적인 방식으로 트리를 출력하는 함수들을 구현하는데 머리가 복잡했었다. 하지만 linked list 를 처음 배웠을 때와 마찬가지로 익숙해 지면 편해질 수 있을 것 같다.

리눅스를 사용하는 것은 이번이 처음이다. 그래서 익숙하지 않았다. 이번 학기에 새로 배우는 언어들이 많아서 많이 느꼈던 것이지만 앞으로 더 사용하면 쉬워질 것이라 생각된다. 처음에는 리눅스도 비주얼처럼 강력한 툴이 있을 것이라 생각을 했었지만 이번 과제로 오류를 잡아 주어야 될 것이 많다는 것을 깨닫게 되었다.

자신 평가: 10/10

박우혁

Linux, Alphabet BST/node, Manager, Queue, Debug

이 프로젝트를 진행하면서 BST 의 구조에 대해 이해하게 되었다 BST 는 binary search tree 의 약자이며 하나의 node 가 최대 2 개의 child 를 가지고 어떠한 기준에 의해서 오른쪽 child 인지 왼쪽 child 인지 정해진다. insert 는 그 기준대로 적절한 node 를 search 하여 그 아래에 insert 하고 search 는 root 부터 시작해서 node 와 찾는 node 를 비교해가며 조건대로 움직여서 찾는 방식이다. Print 는 추가적으로 각 order 를 받아서 그 방식대로 출력하는 방식이다. project 에서 사용한 order 는 각각 pre, post, in, level 인데 이는 각각 먼저 간 node 대로, 하위 node 부터, 왼쪽부터, 위부터 순으로 출력하는 방법이다.

queue 나 stack, circular linked list 등은 2 학년 1 학기 때 공부하여 이미 알고 있어서 문제가 없었지만 주어진 BST 는 생소해서 고생을 했던 것 같다. 또한

Linux 를 하면서 Linux 를 사용하는 것이 처음이어서 생소해서 알아보는 것에서 고생하였다. visual studio 가 표준 대로 컴파일하기 보다는 여러가지 상황을 알아서 교정해주는 tool 인데 비해 Linux 에서 사용하는 g++은 표준 대로 컴파일 하기 때문에 visual studio 에서 사용되는 함수들이 Linux 에서는 사용되지 않는 경우가 있었다. 이런 경우에는 검색엔진 Google 을 사용하여 각 함수들이 Linux 에서도 작용하는지 알아봐야 했고 또한 Linux 는 visual studio 같이 직관적으로 문제를 보여주지 않기 때문에 GDB 를 사용하는 방법을 알아내야 했다.

GDB 는 Linux 에서 사용하는 디버거인데 run: 동작, bt: 오류 가 생긴 함수 출력, b (name) (line) : name 함수의 line 에 break point, c: break point 까지 이동 의 기능을 한다. 또한 Linux 에서는 Segmentation Error 가 뜨곤 하는데 이는 memory 사용 방법에서 문제가 생기면 나타나는 에러이다. 실제로 visual studio 의 code 를 Linux 로 옮길 때 Segmentation error 가 나타났고 GDB 에서 strcpy 함수가 문제가 있다고 출력해서 이유를 찾기 위해 외국의 forum 을 찾아서 알아냈는데 그 이유는 str 함수 내부에 직접적으로든 간접적으로든 NULL 이 들어갔기 때문이었다.

자신 평가: 10/10