데이터구조설계

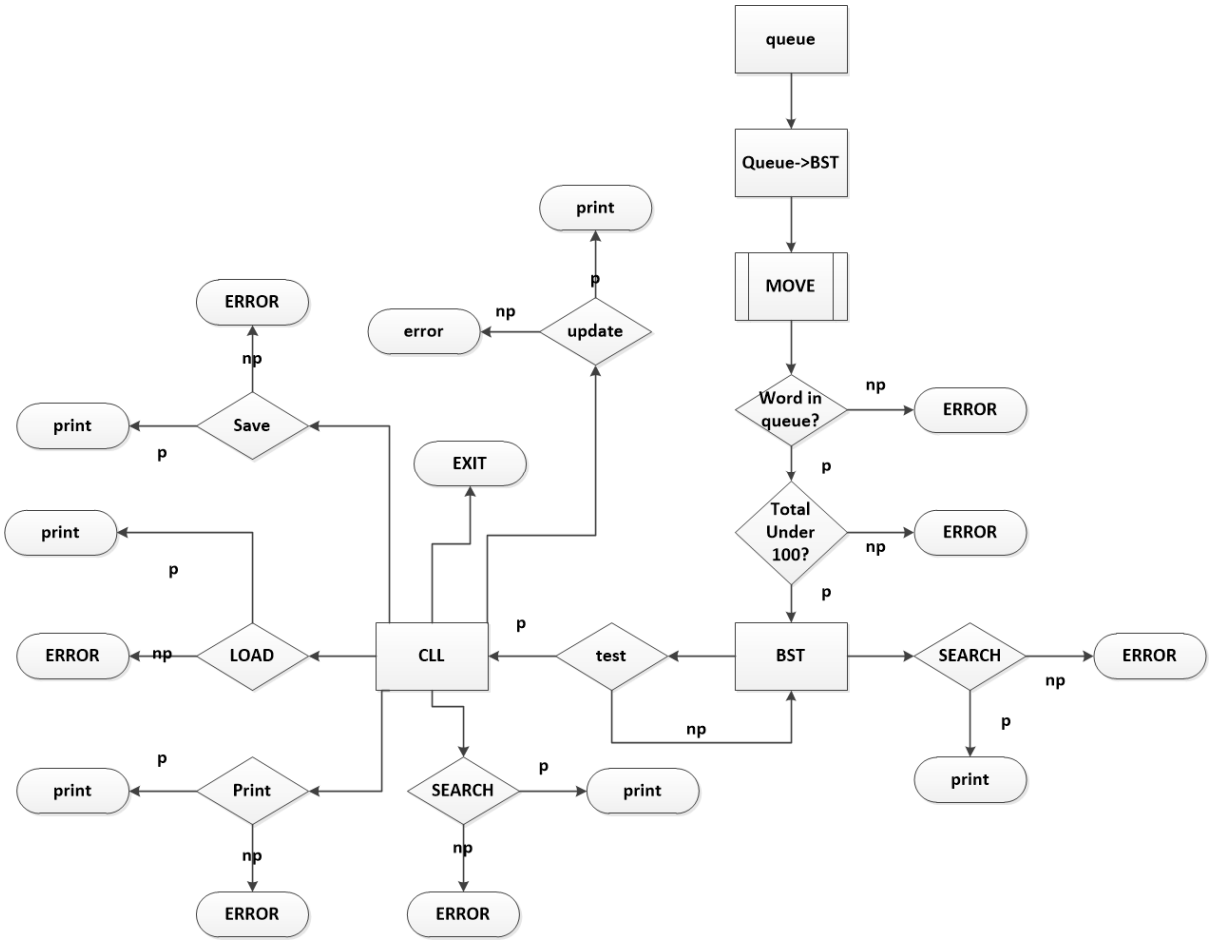
Project 1

교 수	이기훈 교수님
학 과	컴퓨터공학과
조 장	2015722011 조만희
조 원	2015722016 김경호 2016722092 조동호
제 출 일	2016 / 10 / 07

1. Introduction

Queue, BST, Circular Linked-List를 이용하여 영어 단어장을 만든다. 단어들의 목록은 Queue에, 외울 단어들은 최대 100개를 BST에, 다 외운 단어들은 Circular Linked-List구조에 저장하고, 각각 'to_memorize_word.txt', 'memorizing_word.txt', 'memorized_word.txt' 텍스트 파일에 출력하여 저장한다. command창에 LOAD, ADD, MOVE, SAVE, TEST, SEARCH, PRINT, UPDATE, EXIT의 9가지 커맨드들을 입력하여 구동된 코드들이 동작하면 동작된 내용을, 동작하지 않으면 각각에 해당하는 에러 코드를 출력한다.

2. Flowchart



3. Algorithm

- LOAD : to_memorize_word, memorizing_word, memorized_word 텍스트 파일들을 순서대로 Queue 구조인 to_memorize, BST 구조인 memorizing, Circular Linked-List 구조인 memorized 에 불러와 저장한다. 에러 코드는 3개의 텍스트파일이 없을 경우, 혹은 자료구조에 이미 데이터가 들어가 있을 때 출력한다. 에러 코드 번호는 100. 사용 예시는 LOAD

- ADD : word.txt에 존재하는 단어들을 to_memorize에 저장하는 함수로, 에러 코드는 단어 텍스트파일이 존재하지 않거나 파일에 단어가 존재하지 않을 때 출력한다. 에러 코드 번호는 200. 사용 예시는 ADD

- MOVE : Queue에서 BST로 입력받을 만큼의 1~100의 숫자를 사용자가 입력하여 단어 정보를 이동시킨다. 에러 코드는 이미 존재하는 단어와 새로 입력받는 단어 개수의 합이 100개 보다 많을 때, 사용자가 입력한 수만큼의 단어가 Queue에 존재하지 않을 때 출력한다. 에러 코드 번호는 300. 사용 예시는 MOVE 50

- SAVE : 현재 단어장 정보를 저장하는 명령어로, Queue, BST, Circular Linked-List의 단어 장 내용을 각각 to_memorize_word, memorizing_word, memorized_word 텍스트 파일에 저장한다. 에러 코드는 단어장의 정보가 존재하지 않을 때 출력한다. 에러 코드 번호는 400. 사용 예시는 SAVE

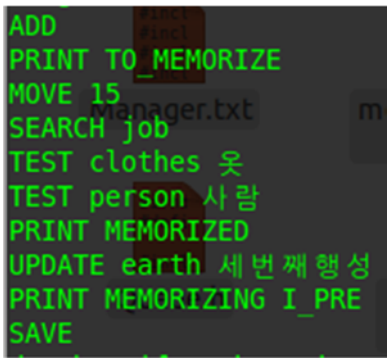
- TEST : 단어를 외웠는지 확인하는 명령어로, BST에 있는 단어인지 확인 후, 의미가 맞는지 확인하여 의미가 맞으면 Circular Linked-List로 단어를 이동시킨다. 에러 코드는 입력한 단어의 의미가 틀리거나 BST에 존재하지 않을 때 출력한다. 에러 코드 번호는 500. 사용 예시는 TEST banana 바나나

- SEARCH : 단어의 뜻을 검색하는 명령어로 Queue, BST, Circular Linked-List에서 입력한 단어와 일치하는 단어가 있는지를 검색하여 존재하면 단어와 뜻을 출력해준다. 에러 코드는 입력한 단어가 존재하지 않으면 출력한다. 에러 코드 번호는 600. 사용 예시는 SEARCH apple('apple 사과' 결과 출력)

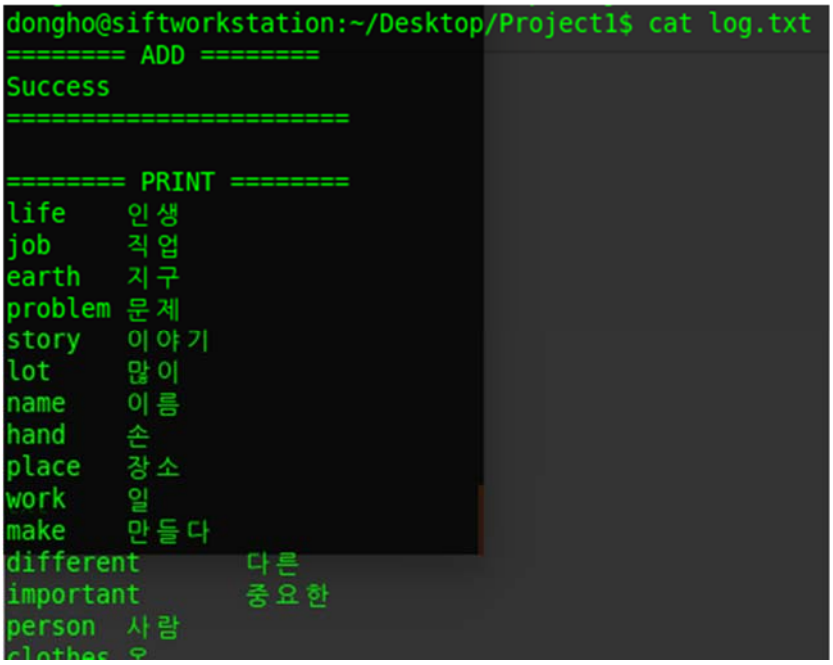
- PRINT : 입력한 단어장의 단어를 출력하는 명령어로, Queue를 PRINT할 때는 Header부터 차례대로 전부 출력하고, BST를 PRINT할 때는 추가로 조건을 입력받아 pre-order, in-order, post-order중 알맞은 순서대로 단어들을 전부 출력한다. Circular Linked-List를 PRINT할 때는 단어들을 외운 순서대로(TEST를 통해 입력받은 순서대로) 전부 출력한다. 에러 코드는 입력한 단어장 정보가 없을 경우 출력한다. 에러 코드 번호는 700. 사용 예시는 PRINT TO_MEMORIZE, PRINT MEMORIZING R_PRE

- UPDATE : 단어의 의미를 변경하는 명령어로, Queue, BST, Circular Linked-List에 존재하는 단어의 의미를 새로 입력한 단어로 변경해준다. 에러 코드는 단어나 단어장이 존재하지 않을 때 출력한다. 에러 코드 번호는 800. 사용 예시는 UPDATE apple 바나나('apple 사과 -> apple 바나나'로 변경)
- EXIT : 프로그램에 할당된 메모리를 모두 해제하고, 프로그램을 종료시키는 명령어이다. 사용 예시는 EXIT

4. Result Screen

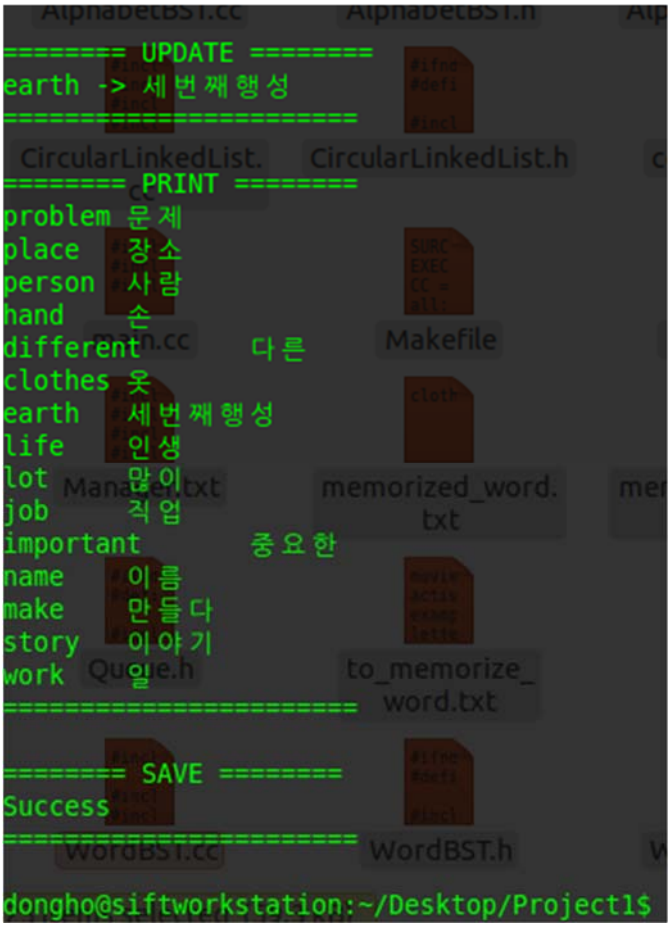


사용한 Command.txt 예제. 구현한 함수들을 전부 실행하여 그 동작을 검증해 주도록 한다.



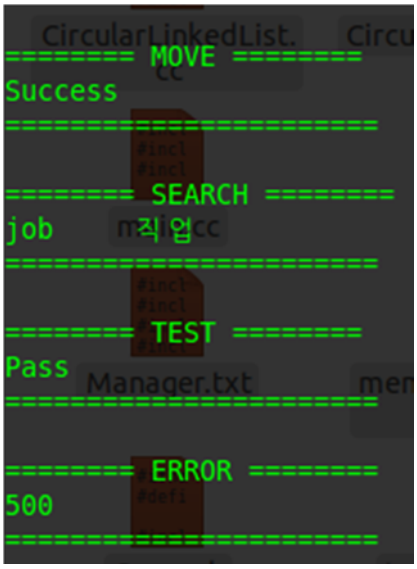
ADD와 PRINT 명령어 검증.

ADD와 MOVE는 성공적으로 이루어져 에러 출력 없이 Success 출력, Print 명령어로 저장된 단어장의 리스트 출력.



UPDATE, SAVE 명령어 검증.

UPDATE 명령어를 이용하여 단어 earth의 뜻을 '세번째행성'으로 update해주고, PRINT를 통해 단어의 뜻이 정상적으로 변경됨을 확인한다. SAVE 에서는 단어장이 성공적으로 저장되었으므로 에러 없이 Success로 출력된다.



MOVE, SEARCH, TEST 명령어 검증.

MOVE 명령어를 통해 단어를 To_Memorize에서 Memorizing으로 옮긴다. 성공하여 Success가 출력된다. SEARCH 명령어로 단어를 검색하는 기능이 구현된다. TEST는 command에서 입력된 단어&뜻이 리스트에 존재할 경우 Pass를, 그렇지 않을 경우 왼쪽 그림과 같이 ERROR 500을 출력한다.

5. Consideration

<table>
<tr><th>이름</th><th>프로젝트에서 맡은 역할</th><th>본인 스스로 생각하는 자신의 점수 (10)</th></tr>
<tr><td rowspan="2">김경호</td><td>플로우차트 작성, 주석처리, 보고서 작성 전반</td><td>7</td></tr>
<tr><td colspan="2">부족한 점을 팀원들이 보충해 주는 것을 보고 교수님이나 선배님뿐만 아니라 함께 작업하는 동료에게서도 배울 수 있는 점이 많다는 것을 배울 수 있었다. 가상머신이 켜지지 않아서 당황했지만, 다행히 다른 팀원의 가상머신으로 리눅스 컴파일을 할 수 있었다.</td></tr>
<tr><td rowspan="2">정동호</td><td>WordBST, WordNode, AlphabetBST, AlphabetNode, Manager 구현 및 리눅스 포팅</td><td>7</td></tr>
<tr><td colspan="2">이진 트리를 머리로만 알고 직접 구현해본적은 없었는데 이번 기회에 직접 구현해볼 수 있어서 정말 재밌었습니다.
그리고 Manager를 구현하며 큐와 링크드리스트 메커니즘도 알아볼 수 있었던 좋은 기회였습니다.
다만 어려웠던 점은 C++을 제대로 다뤄본적이 없어서 FILE* 대신 fstream을 처음 써봤는데 한글 인코딩 관련 부분에서 많이 삽질했습니다. 그리고 두번째로 어려웠던건 리눅스로 포팅하는 과정이었습니다.
이는 옵션에서 -std=c++11을 주어 해결하였습니다. 마지막으로 가장 어려웠던 부분은 log.txt를 출력하는 부분에서 Manager-> AlphabetBST ->WordBST의 마지막 부분에서 logfile에 접근하려 했으나 안되서 삽질하다 결국 시간이 부족하여 레퍼런스로 하나하나 전달해주어 해결하였습니다.</td></tr>
<tr><td rowspan="2">조만희</td><td>Queue, Circular Linked List 코드 작성, 조장을 맡아 일정 관리, 파일 제출</td><td>6</td></tr>
<tr><td colspan="2">처음 프로젝트를 보았을 때에는 비교적 간단하게 느껴졌으나, 새로 배우게 된 BST를 사용해야 한다는 점과, 리눅스를 이용하여 프로젝트를 검증해야 한다는 점에서 어려움이 있었다고 생각한다. 비교적 늦게 프로젝트를 시작했으나 정동호 학우가 많이 노력해 주어 짧은 시간 내에 코드를 완성할 수 있었다. 또한, 같은 동아리에 속하는 세명이서 조를 짜서 활동하여 팀원간의 단합도 잘 이루어 졌고, 연락도 정기적으로 잘 이루어져 효율적으로 과제 진행이 가능했다.</td></tr>
</table>