

데이터 구조 설계 및 실습

Data Structure Lab. Project #1

Complete date: 2016년 10월 07일 (금)

학 과:컴퓨터공학과

담당교수:이기훈 교수님

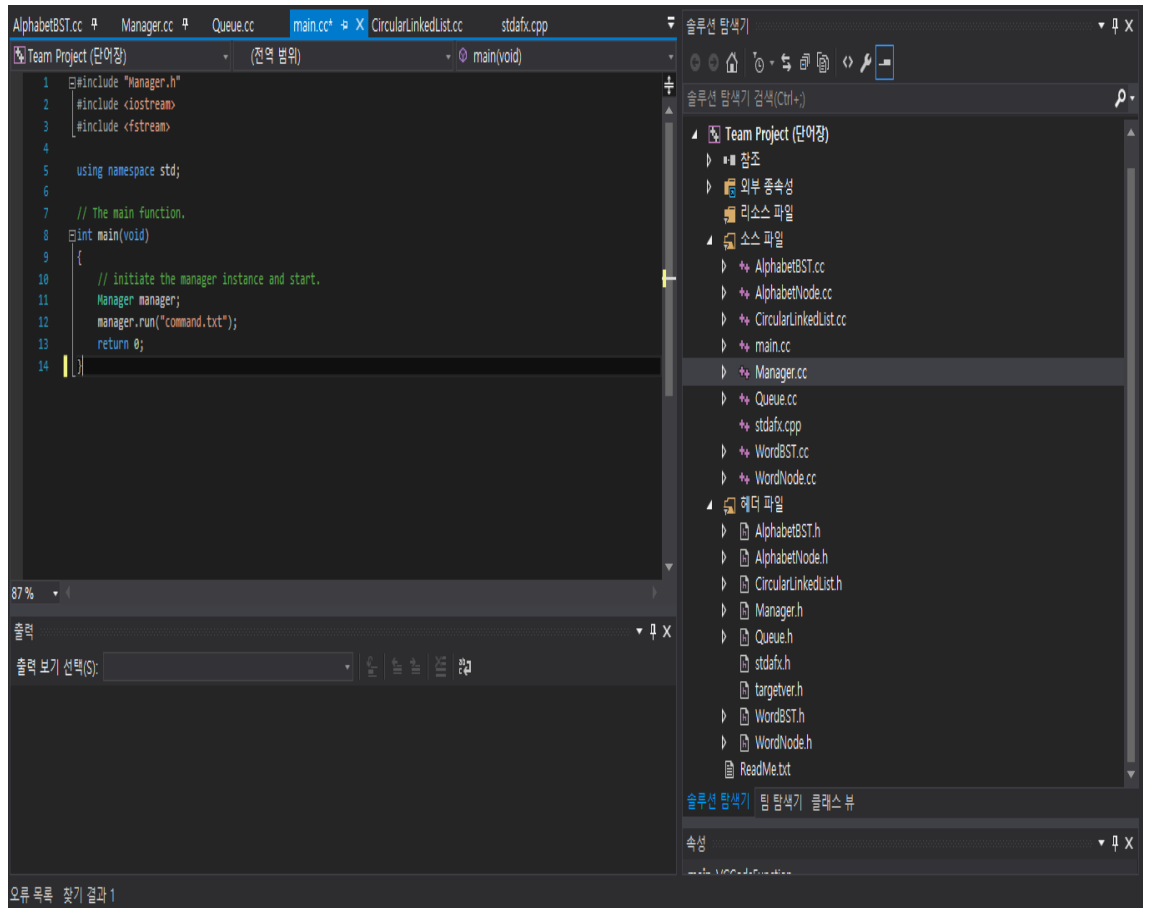
팀 장:손상렬

손상렬(2012722063)

박 현(2012722046)

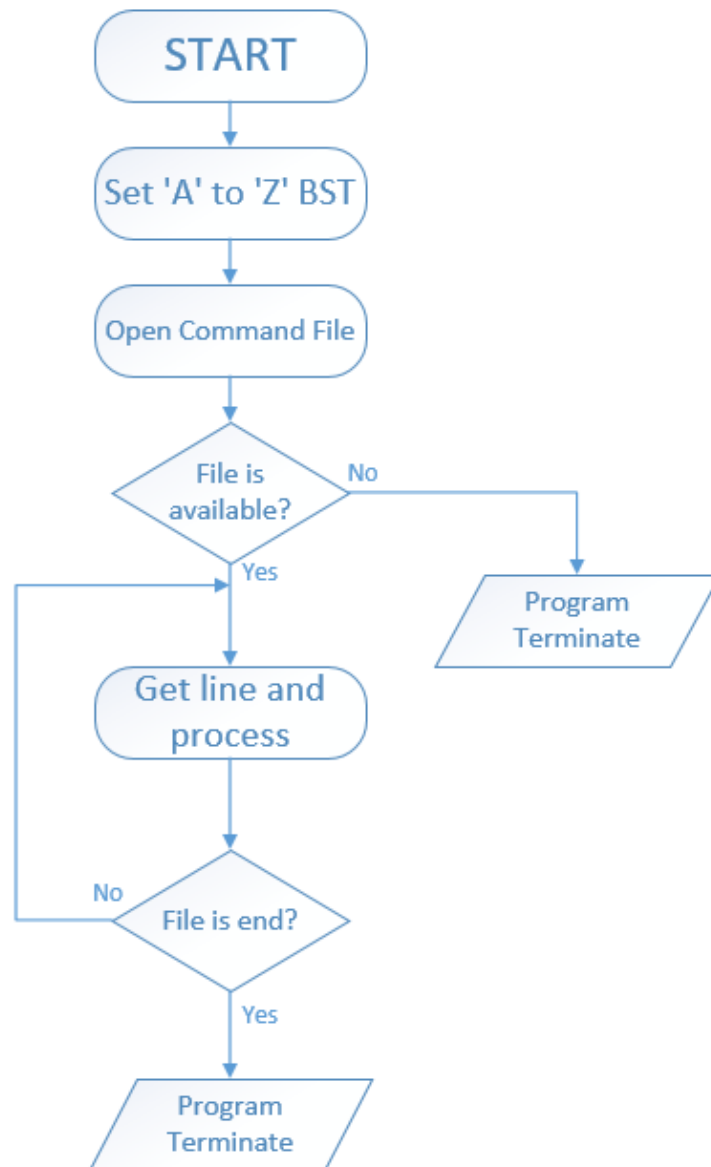
1. Introduction

이 프로젝트는 자료구조 프로젝트로서 자료구조의 핵심적인 알고리즘을 이용하여 활용하는 것에 목적을 두고 있습니다. 무엇보다 이진 탐색 트리, 환형 연결 리스트, 큐를 모두 구현할 수 있어야 완료할 수 있는 프로젝트라는 점에서 자료구조의 응용적인 측면에서 의의가 있다고 생각합니다.



프로젝트 작업은 비주얼 스튜디오 2015에서 작업을 하였으며 구성은 위와 같이 초기에 과제에서 제시하였던 구성을 따르고 있습니다. 프로그램이 실행되면 처음에 메인 함수에서 Manager 클래스의 객체를 선언하여 프로그램의 중심적인 내용을 실행하게 됩니다. Manager 객체는 실행되자마자 알파벳(Alphabet) BST를 기준으로 모든 알파벳을 추가하게 되며 그 이후에 단어가 들어오면 그 알파벳 객체의 변수로 삽입됩니다. 나중에 PRINT, ADD 등의 다양한 작업을 처리한 뒤에 SAVE 및 EXIT의 과정을 거쳐 프로그램이 종료되게 됩니다. 구체적인 알고리즘 및 설계에 관한 설명은 뒤에 제시됩니다.

2. Flowchart



기본적인 프로그램 구성 및 실행 그 전체를 담당하는 Manager 클래스입니다. 이 클래스에서는 위와 같이 프로그램이 실행되자마자 A부터 Z까지의 Alphabet BST를 구성하게 됩니다. 이후에 Command 파일을 찾게 되고 찾으면 그것이 끝날 때까지 각 줄을 읽어 명령을 수행하게 됩니다. 만약에 파일을 찾지 못하거나 전부 읽었을 경우에는 프로그램을 정상 종료합니다.

3. Algorithm

위에서는 전체적인 프로젝트의 흐름을 명시하는 플로우차트를 명시했습니다. 여기에서는 프로젝트에서 각각 사용된 주요 알고리즘들의 동작을 설명하게 될 것입니다. 프로젝트의 핵심 알고리즘이 이진 탐색 트리, 환형 연결 리스트, 큐이기 때문에 이 3가지 알고리즘을 중심으로 하여 설명할 것입니다.

- 이진 탐색 트리

이진 탐색 트리는 기본적으로 삽입, 삭제 구현이 핵심적이라고 할 수 있습니다. 먼저 삽입 기능의 알고리즘은 다음과 같습니다.

1. 삽입하려는 특정 단어가 기준 단어보다 앞에 있는 경우 기준 단어의 왼쪽 노드에 특정 단어를 재귀적으로 삽입
2. 삽입하려는 특정 단어가 기준 단어보다 뒤에 있는 경우 기준 단어의 오른쪽 노드에 특정 단어를 재귀적으로 삽입
3. 현재의 기준 단어가 NULL 값을 가지는 경우 현재의 기준 단어에 특정 단어를 삽입

이후에 Manager 클래스의 객체에서 TEST 기능을 사용할 때 이진 탐색 트리에서 삭제 기능이 이루어져야 합니다. 삭제 기능은 삽입 기능보다 훨씬 복잡하지만 한번 제대로 구현하면 쉽게 사용이 가능합니다.

1. 삭제하려는 특정 단어가 기준 단어보다 앞에 있는 경우 기준 단어의 왼쪽 노드에 특정 단어를 재귀적으로 삭제
2. 삭제하려는 특정 단어가 기준 단어보다 뒤에 있는 경우 기준 단어의 오른쪽 노드에 특정 단어를 재귀적으로 삭제
3. 삭제하려는 특정 단어가 기준 단어와 일치하는 경우 기준 단어의 왼쪽 혹은 오른쪽 자식이 있는지에 대한 여부에 따라서 재귀적으로 특정 단어를 삭제
4. 삭제가 이루어진 이후에는 이진탐색트리가 정상적으로 작동할 수 있도록 메모리에서 제거하여 정렬

- 환형 연결 리스트

환형 연결 리스트는 간단하게 서로 연결이 되어 있는 연결 리스트 구조의 한 종류입니다. 여기서 기본적으로 환형 연결리스트는 리스트의 헤드 부분과 꼬리 부분이 붙어있는 구조가 더해진다고 생각하면 아주 쉬운 자료구조의 한 형태라고 할 수 있습니다. 이 프로젝트에서는 환형 연결 리스트에서 삭제 부분을 구현할 필요가 없기 때문에 검색 기능 및 삽입 기능만 다루어보도록 하겠습니다. 먼저 삽입 기능은 다음과 같은 알고리즘을 따르게 됩니다. 여기서 의미하는 노드란 WordNode로서 단어 노드를 의미합니다.

1. 현재 연결 리스트의 머리 부분에서 제일 꼬리 부분까지 이동하며 꼬리 부분의 노드를 찾음
2. 꼬리 부분의 노드의 다음 노드에 삽입하려는 노드를 연결
3. 삽입된 노드의 다음 부분을 머리 부분의 노드와 연결

반면에 환형 연결 리스트에서의 검색 기능은 다음의 일련의 알고리즘을 따르며 실행됩니다.

1. 기준 노드의 다음 노드에서부터 탐색을 시작
2. 현재의 노드가 시작 노드와 일치하지 않는다면 현재의 노드가 찾으려는 노드와 일치한지 판단
3. 일치하지 않는다면 다음의 노드로 이동한 뒤에 반복적으로 재탐색

이 프로젝트에서 환형 연결 리스트는 위의 구현 부분이면 충분할 정도로 간단하게 사용할 수 있습니다. 삭제 기능 또한 필요가 없기 때문입니다.

- 큐

이 프로젝트에서 중심적인 역할을 하는 자료구조입니다. 큐는 기본적으로 삽입(PUSH)와 삭제(POP)의 두 가지 기능을 적절히 조합하며 사용하는 것으로 구현이 됩니다. 프로젝트의 시작에서 단어를 단어장에 삽입하는 등의 대부분의 과정인 큐를 기반으로 시작되기 때문에 큐를 특별히 신경 써서 구현할 필요가 있었습니다. 먼저 중심적인 역할을 하는 삽입 부분의 알고리즘을 살펴보면 다음과 같습니다.

1. 삽입하려는 단어를 WordNode 객체를 생성하여 삽입
2. 해당 단어 노드를 큐의 꼬리의 다음 노드에 삽입
3. 만약에 현재 큐의 머리 부분이 NULL 값을 가진다면 머리 부분에 삽입하려는 단어를 삽입

이후에 Manager 객체에서 단어를 현재 외우고 있는 단어장에 등록하는 기능인 MOVE 기능을 사용할 때 POP(삭제) 기능이 사용될 것입니다. 이 삭제 기능은 더 쉽게 구현이 가능한데 그 자세한 알고리즘을 다음과 같습니다.

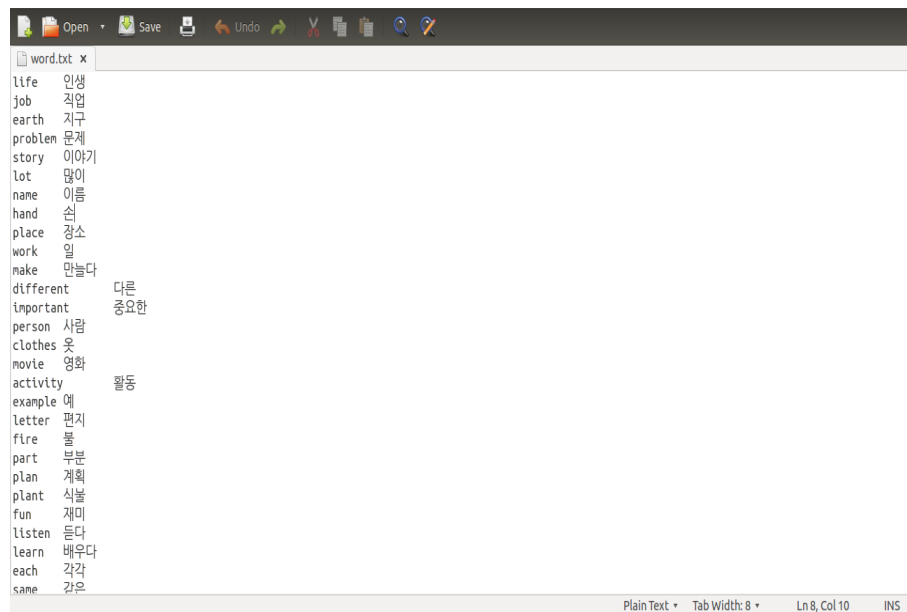
1. 만약에 큐가 비어있다면 NULL 값을 반환
2. 큐의 머리 부분을 머리 부분의 다음 노드의 값으로 정하기
3. 삭제된 부분의 메모리 할당 부분 제거

4. Result Screen

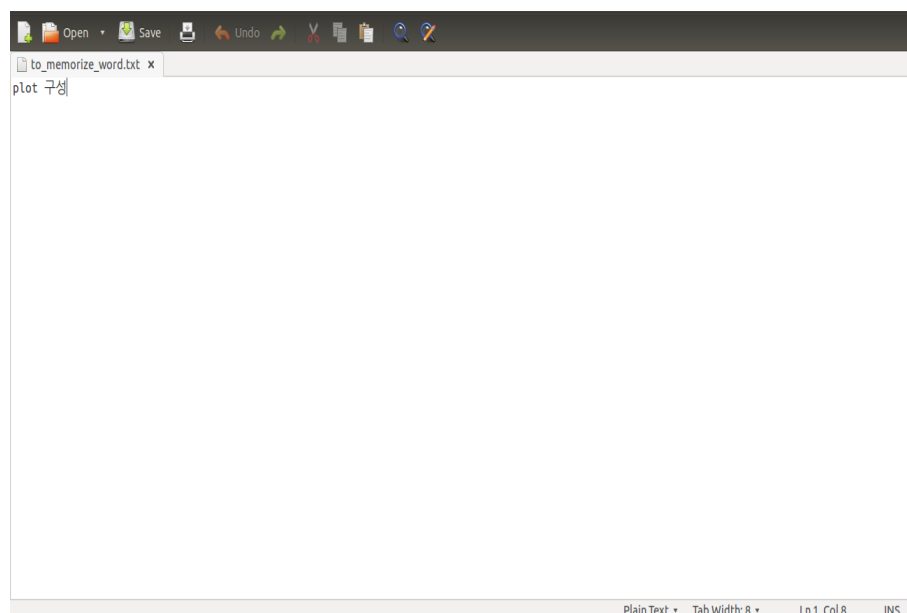
프로젝트의 구현이 완료된 이후에 프로젝트가 정상적으로 작동하는지 확인하기 위하여 실제 예시 데이터를 삽입하여 프로젝트를 점검하는 시간을 가졌습니다. 프로젝트의 구현을 위해서는 파일 입출력 기능의 정상화를 위해서 각각의 메모장 정보를 구체화할 필요가 있었습니다.

< 실행 전 파일 구성 >

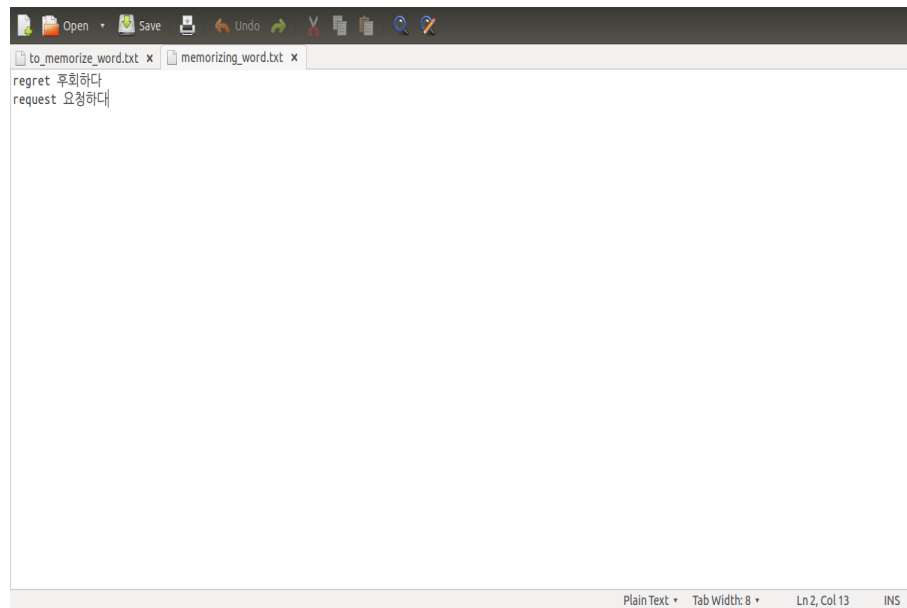
- word.txt



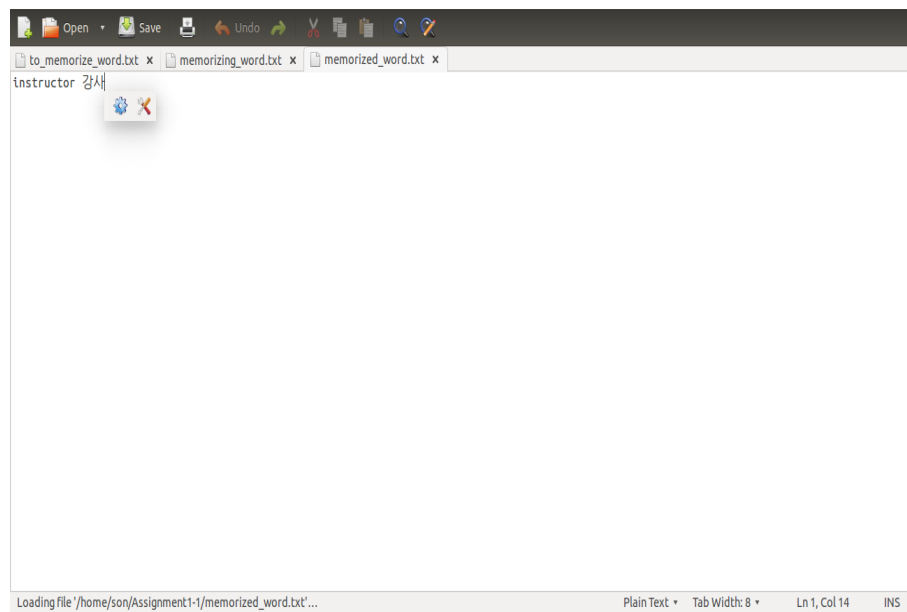
- to_memorize_word.txt



- memorizing_word.txt



- memorized_word.txt



- **command.txt**

```
LOAD word.txt
ADD
MOVE 80
TEST fire 불
PRINT MEMORIZED
PRINT MEMORIZING R_IN
UPDATE area 구역
SEARCH area
PRINT MEMORIZED
SAVE
EXIT
```

< 실행 결과 >

```
Manager.cc:189:29: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
son@ubuntu:~/Assignment1-1$ ./wordbook
===== LOAD =====
Success
=====
===== ADD =====
Success
=====
===== MOVE =====
Success
=====
===== TEST =====
Success
=====
===== PRINT =====
instructor 강사
fire 불
=====
===== PRINT =====
activity 활동
advertisement 광고
always 언제나
autumn 가을
back 뒤
bird 새
clean 깨끗한
clothes 옷
course 강좌
culture 문화
different 다른
each 각각
earth 지구
easy 쉬운
enjoy 즐기다
example 예
```


earth 지구
easy 쉬운
enjoy 즐기다
example 예
face 얼굴
famous 유명한
fast 빨리
fly 날다
fun 재미
gene 유전자
group 집단
habit 습관
hand 손
history 역사
important 중요한
information 정보
island 섬
job 직업
just 단지
learn 배우다
letter 편지
life 인생
light 빛
listen 듣다
lot 많이
make 만들다
mind 마음
movie 영화
name 이름
nature 자연
newspaper 신문
office 사무실
paper 종이
part 부분

paper 종이
part 부분
person 사람
piece 조각
place 장소
plan 계획
plant 식물
plot 구성
poor 가난한
problem 문제
regret 후회하다
request 요청하다
restaurant 식당
rock 바위
same 같은
science 과학
soldier 군인
sound 소리
space 공간
special 특별한
spring 봄
start 시작
state 상태
store 가게
story 이야기
street 거리
summer 여름
trip 여행
vacation 휴가
village 마을
visit 방문
volunteer 자원봉사자
war 전쟁
watch 시계

```
war 전쟁
watch 시계
win 이기다
winter 겨울
work 일

=====
===== UPDATE =====
area 지역 -> 구역
=====
===== SEARCH =====
area 구역
=====
===== PRINT =====
instructor 강사
fire 불
=====
===== SAVE =====
Success
=====
son@ubuntu:~/Assignment1-1$ ls -l
```

```
===== LOAD =====
Success
===== ADD =====
Success
===== MOVE =====
Success
===== TEST =====
Success
===== PRINT =====
instructor 강사
fire 불
===== PRINT =====
activity 활동
advertisement 광고
always 언제나
autumn 가을
back 뒤
bird 새
clean 깨끗한
clothes 옷
course 강좌
culture 문화
different 다른
```

```
each 각각
earth 지구
easy 쉬운
enjoy 즐기다
example 예
face 얼굴
famous 유명한
fast 빨리
fly 날다
fun 재미
gene 유전자
group 집단
habit 습관
hand 손
history 역사
important 중요한
information 정보
island 섬
job 직업
just 단지
learn 배우다
letter 편지
life 인생
light 빛
listen 듣다
lot 많이
make 만들다
```

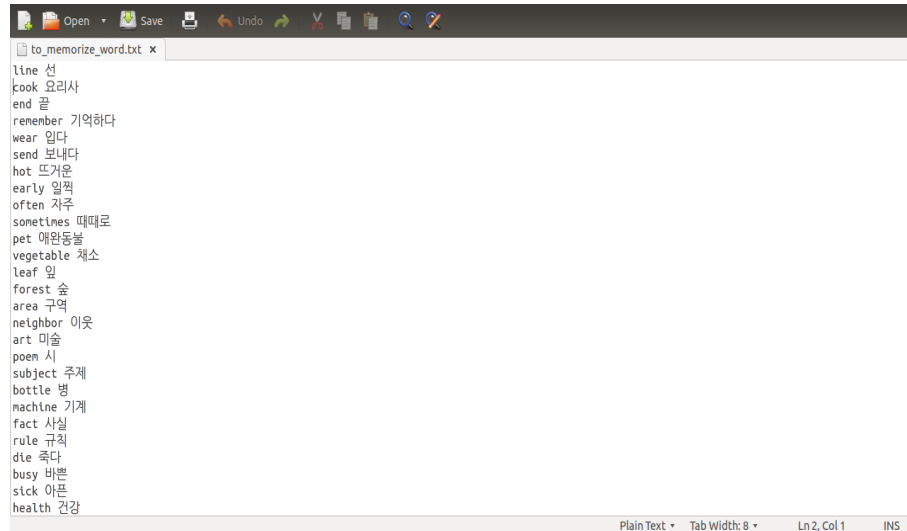
The image contains two screenshots of a text editor window. The top screenshot shows the output of a program execution. The text is as follows:
to_memorize_word.txt x memorizing_word.txt x memorized_word.txt x log.txt x command.txt x
spring 봄
start 시작
state 상태
store 가게
story 이야기
street 거리
summer 여름
trip 여행
vacation 휴가
village 마을
visit 방문
volunteer 자원봉사자
war 전쟁
watch 시계
win 이기다
winter 겨울
work 일
=====
===== UPDATE =====
area 지역 -> 구역
===== SEARCH =====
area 구역
===== PRINT =====
Instructor 강사
fire 불
===== SAVE =====
Success
=====

The bottom screenshot shows a list of words in English and Korean. The text is as follows:
to_memorize_word.txt x memorizing_word.txt x memorized_word.txt x log.txt x command.txt x
make 만들기
mind 마음
movie 영화
name 이름
nature 자연
newspaper 신문
office 사무실
paper 종이
part 부분
person 사람
piece 조각
place 장소
plan 계획
plant 식물
plot 구성
poor 가난한
problem 문제
regret 후회하다
request 요청하다
restaurant 식당
rock 바위
same 같은
science 과학
soldier 군인
sound 소리
space 공간
special 특별해

위와 같이 프로그램이 실행된 결과 화면과 log.txt 파일의 결과가 일치한다는 것을 확인할 수 있었습니다. 모든 기능이 정상적으로 작동하였습니다.

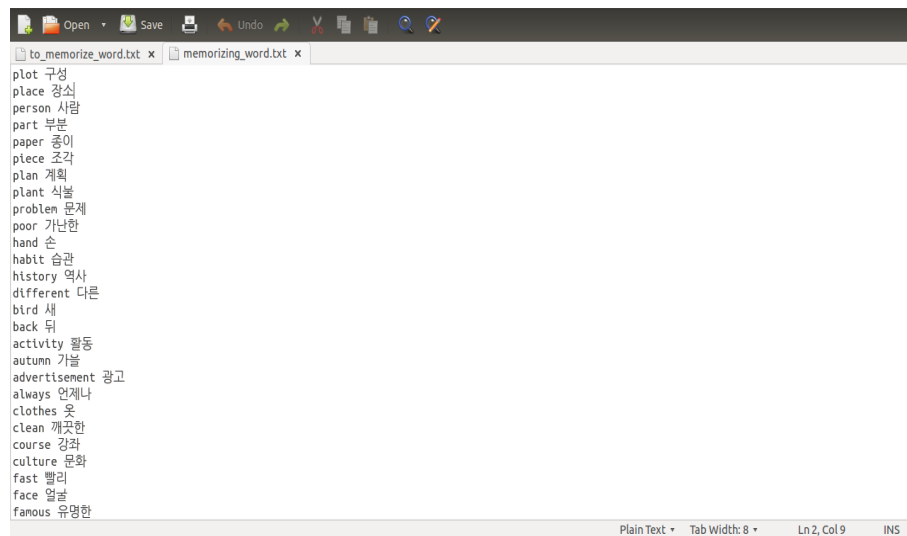
< 실행 후 파일 구성 >

- to_memorize_word.txt



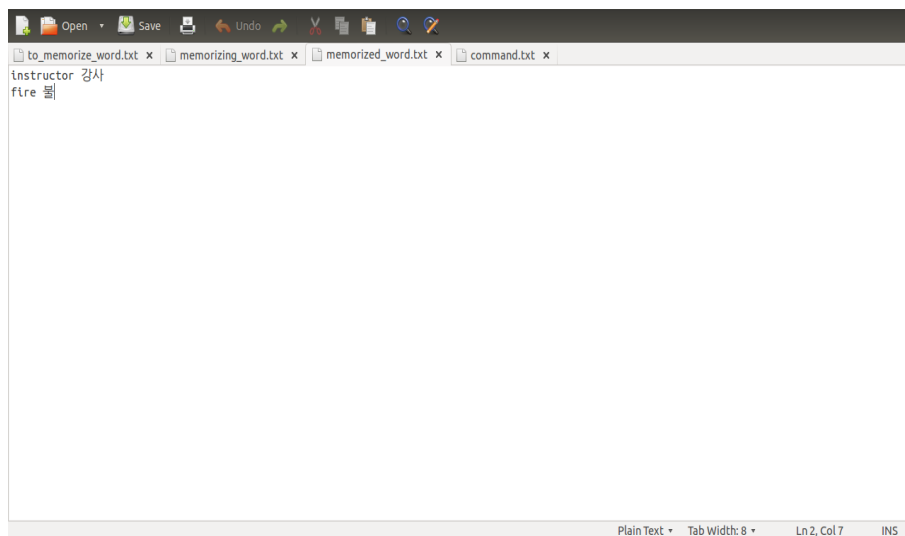
```
line 선
kook 요리사
end 끝
remember 기억하다
wear 입다
send 보내다
hot 뜨거운
early 일찍
often 자주
sometimes 때때로
pet 애완동물
vegetable 채소
leaf 잎
forest 숲
area 구역
neighbor 이웃
art 미술
poem 시
subject 주제
bottle 병
machine 기계
fact 사실
rule 규칙
die 죽다
busy 바쁜
sick 아픈
health 건강
```

- memorizing_word.txt



```
plot 구성
place 장소
person 사람
part 부분
paper 종이
piece 조각
plan 계획
plant 식물
problem 문제
poor 가난한
hand 손
habit 습관
history 역사
different 다른
bird 새
back 뒤
activity 활동
autumn 가을
advertisement 광고
always 언제나
clothes 옷
clean 깨끗한
course 강좌
culture 문화
fast 빨리
face 얼굴
famous 유명한
```

- memorized_word.txt



이로써 모든 기능이 정상적으로 작동한다는 것을 알 수 있었습니다. 무엇보다 프로그램이 실행되어 결과를 처리한 이후에 파일 입출력이 정상적으로 이루어지는지에 대한 결과에 초점을 맞추어 순차적인 과정으로 프로젝트를 점검하였습니다.

5. Consideration

이름 ; 손상렬

프로젝트에서 맡은 역할 ; BST 일부, Node 일부, Stack, Manager 구현

본인 스스로 생각하는 자신의 점수 ; 8

고찰 ; 대학생할 중 2번째로 약 한 달 동안 팀 프로젝트를 수행해보았다. 절차지향인 C언어와 다르게, 객체지향인 C++이 팀 프로젝트를 수행하는데 더 수월하다고 얼핏 들었지만, 아직 C++에 대한 경험이 부족해서 그런지 많은 어려움을 느꼈다. 특히 c++의 문법적인 면과 리눅스의 사용법 등 실력의 부족함을 많이 느끼게 해준 프로젝트였다. 또한 여태까지 구현하였던 과제의 소스코드와 다르게 프로젝트라 그런지 소스코드의 크기가 비교할 수 없을 정도로 커지니까 디버깅을 할 때 중단점을 사용하여도 프로그램의 흐름을 파악하기가 생각보다 힘들었다.

아직까지는 팀 프로젝트가 익숙하지가 않아서 그런지 '편하다' 보다는 '불편하다'라는 것을 많이 느낀 것 같다. 팀 프로젝트의 장점인 작업을 분담해서 할 수 있다는 것도 있지만, 하나가 잘못되면 전부 다 수정해야 된다는 불상사도 있는 단점도 있기 때문에 이번을 계기로 팀 프로젝트에 대한 좋은 경험을 한 것 같다.

이름 ; 박현

역할 ; BST 일부, Node 일부, Queue, Circular LinkedList 구현

본인 스스로 생각하는 자신의 점수 ; 5

고찰 ; 이번 과제의 핵심은 주어진 조건에 대하여 각자 다른 동작들(이진 탐색 트리, 환형 연결리스트, 큐 등)에 대하여 정확한 동작과 함께 각 동작들을 유기적으로 연결시키는 데에 의의가 있었다고 생각합니다. 각 동작들은 비교적 용도와 목적이 명확하고, 그에 따라 개별적으로 동작하기 때문에 각 파트별로 동작하는 원리에 대해 정확한 이해가 필요했습니다. 예를 들어, 이진 탐색 트리의 경우 이진 트리의 원리와 이진 트리를 사용함으로써 동작의 스테이트가 얼마나 효율적으로 사용되는지에 대해 이해하는 것을 전제하고 있으며, 이를 얼마나 간결하고 명확하게 구현하는지에 대해 생각하게 되었습니다.

리눅스 환경에서 구현하는 것은 낯설고 사소한 오류가 발생해도 대처하는데 애를 먹게 됐지만 이는 설계 과정보다는 코딩 과정에서 사소한 부분에 대한 오류로, 리눅스 환경에서의 코딩에 익숙해짐으로써 극복 가능하다고 생각합니다.