오픈소스 중간프로젝트 발표

독서 감상문

지능로봇공학과
2021042040
김민재

담당교수 : 황영배 교수님

Part

1 작품 설명

2 코드 설명

3 실행 화면

4 아쉬웠던 점 & GitHub 링크

Part 1
작품 설명

Part 1

작품 설명

감상문 작성 & 수정

감상문 출력

감상문 삭제

도서 추천

책을 읽고 저자와 제목, 내용만 입력하면 같은 저자의 감상문끼리 저자명 폴더에 모이게 되며 나중에 저자의 어떤 책을 읽었는지 가시적으로 확인이 가능하며 저자와 제목을 입력하면 작성했던 감상문을 출력해줍니다.
또한 책을 다 읽고 다음 도서를 추천해주는 기능을 포함하고 있습니다.

Part 2

코드 설명

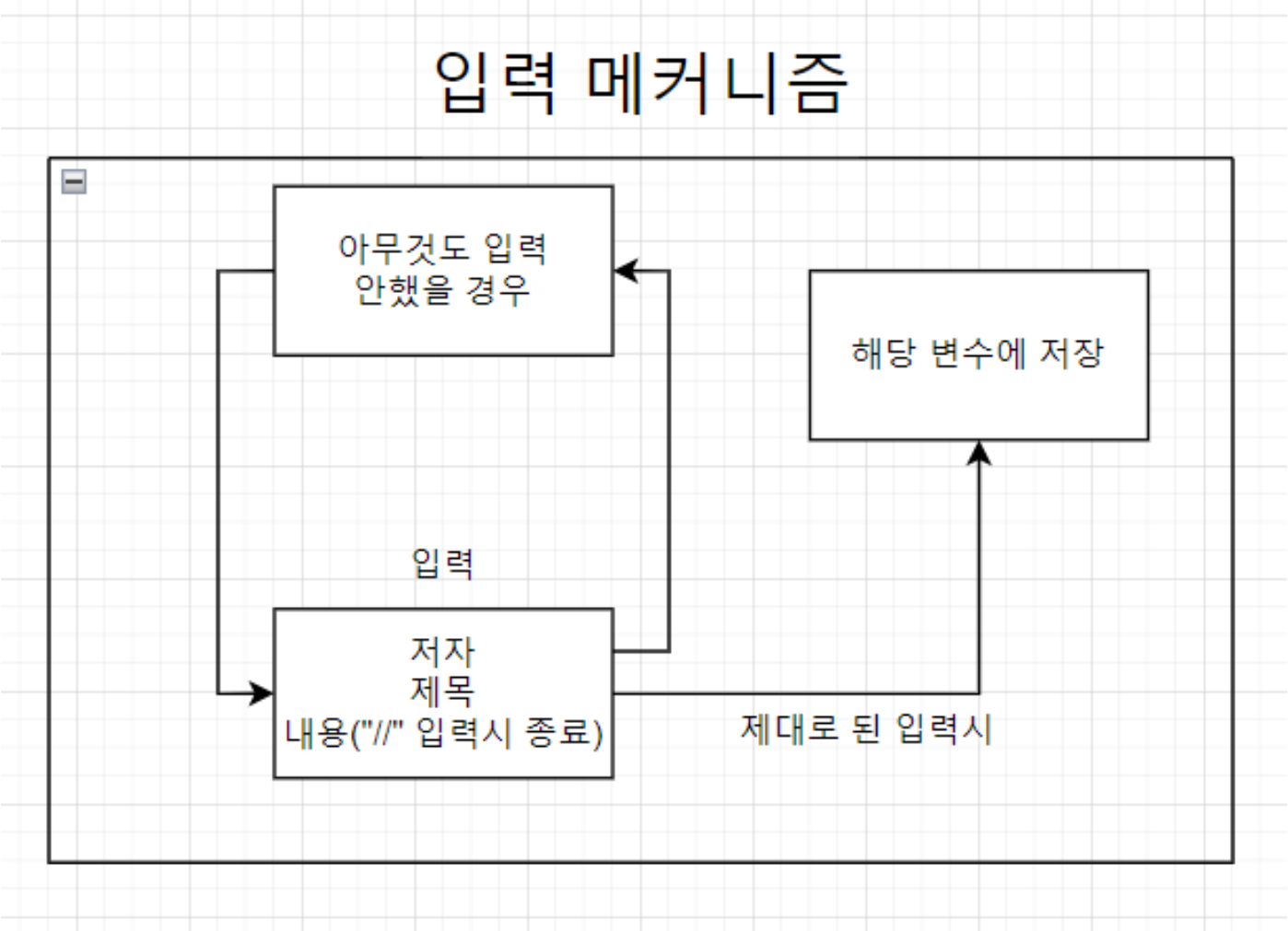
Part 2

코드 설명

1	저자, 제목, 내용 입력 매커니즘
2	make_dir_folder (저자 명) & make_write_text (저자명, 제목, 내용)
3	read_text (저자명, 제목)
4	edit_text (저자명, 제목, 내용)
5	delete_text (저자명, 제목)
6	recommend _book (파일 경로) & add_recommend_book (파일경로)

Part 2 코드 설명

저자, 제목, 내용 입력 매커니즘



Part 2 코드 설명

```c
int valid_input = 0;

while (!valid_input)
{
    // 저자명 입력 받기
    printf("저자명을 입력하세요: ");
    input_folder = (char*)malloc(MAX_TITLE_LENGTH * sizeof(char));

    if (input_folder == NULL)
    {
        printf("메모리 할당 실패₩n");
        return 1;
    }
    fgets(input_folder, MAX_TITLE_LENGTH, stdin);
    input_folder[strcspn(input_folder, "₩n")] = '₩0'; // 개행 문자 제거

    // 저자명이 입력되지 않은 경우
    if (strlen(input_folder) == 0)
    {
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
        printf("저자명이 입력되지 않았습니다. 다시 입력해주세요.₩n");
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글자색 기본값
        free(input_folder);
    }
    else
    {
        valid_input = 1; // 입력이 있으면 반복문 종료
    }
}
```

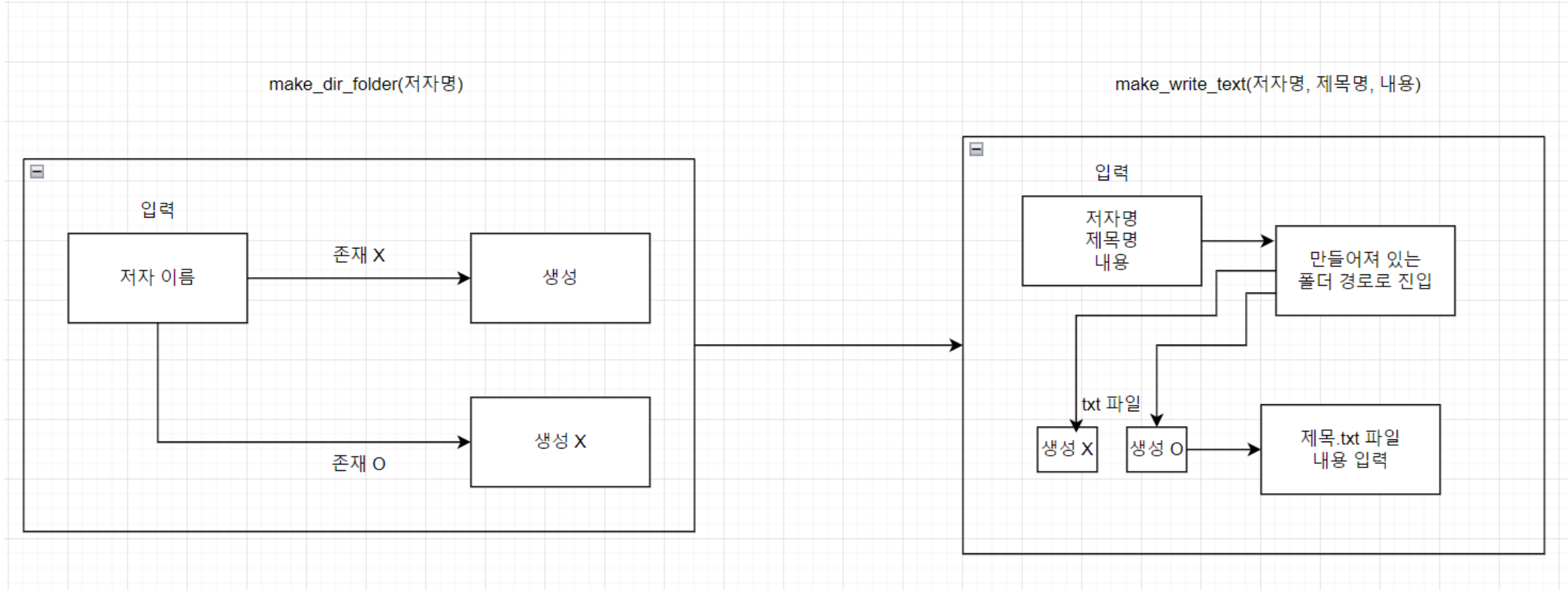
같은 메커니즘으로 저자, 제목을 입력 받음

```c
while (!valid_input)
{

    printf("내용을 입력하세요. 입력을 종료하려면 '// '을 입력하세요.₩n");
    input_content = (char*)malloc(MAX_CONTENT_LENGTH * sizeof(char));

    if (input_content == NULL)
    {
        printf("메모리 할당 실패₩n");
        free(input_folder); // 저자명 메모리 해제
        free(input_title);  // 제목 메모리 해제
        return 1;
    }

    input_content[0] = '₩0'; // 내용 배열 초기화
    // 내용 입력 받기

    while (1)
    {
        fgets(input_content + strlen(input_content), MAX_CONTENT_LENGTH - strlen(input_content), stdin); // 사용자 입력 받기
        // "//" 입력하면 종료
        if (strncmp(input_content + strlen(input_content) - 3, "//", 2) == 0)
        {
            input_content[strlen(input_content) - 3] = '₩0'; // "//" 제거
            break;
        }
    }

    // 내용이 입력되지 않은 경우
    if (strlen(input_content) == 0)
    {
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
        printf("내용이 입력되지 않았습니다. 다시 입력해주세요.₩n");
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글자색 기본값
        free(input_content);
    }
    else
    {
        valid_input = 1; // 입력이 있으면 반복문 종료
    }
}
```

내용을 입력할 때 **enter**키가 필요하기 때문에
“//”을 입력시 종료

Part 2 코드 설명

make_dir_folder (저자 명) & make_write_text (저자명, 제목, 내용)



Part 2 코드 설명

make_dir_folder (저자 명)

```
void make_dir_folder(const char* name)
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE); //글자색 변환 코드

    char* folderPath = NULL;
    folderPath = (char*)malloc(MAX_CONTENT_LENGTH + sizeof(char));

    strcpy(folderPath, "C:₩₩Users₩₩kimmi₩₩Desktop₩₩독후감상문₩₩"); // 독후감상문 폴더에 저자명으로 폴더 생성
    strcat(folderPath, name); // 원래있던 기본 주소와 저자명과 합침.

    BOOL result = CreateDirectoryA(folderPath, NULL);

    if (result)
    {
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
        clearScreen();
        printf("감상문 작성 완료.₩n");
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글자색 기본값
    }
    else
    {
        DWORD error = GetLastError();

        if (error == ERROR_ALREADY_EXISTS)
        {
            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
            clearScreen();
            printf("저자명 폴더가 존재합니다. ₩n");
            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글자색 기본값
        }
        else
        {
            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
            clearScreen();
            printf("폴더를 만드는 데 실패했습니다. 오류 코드: %lu₩n", error);
            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글자색 기본값
        }
    }

    free(folderPath);
}
```

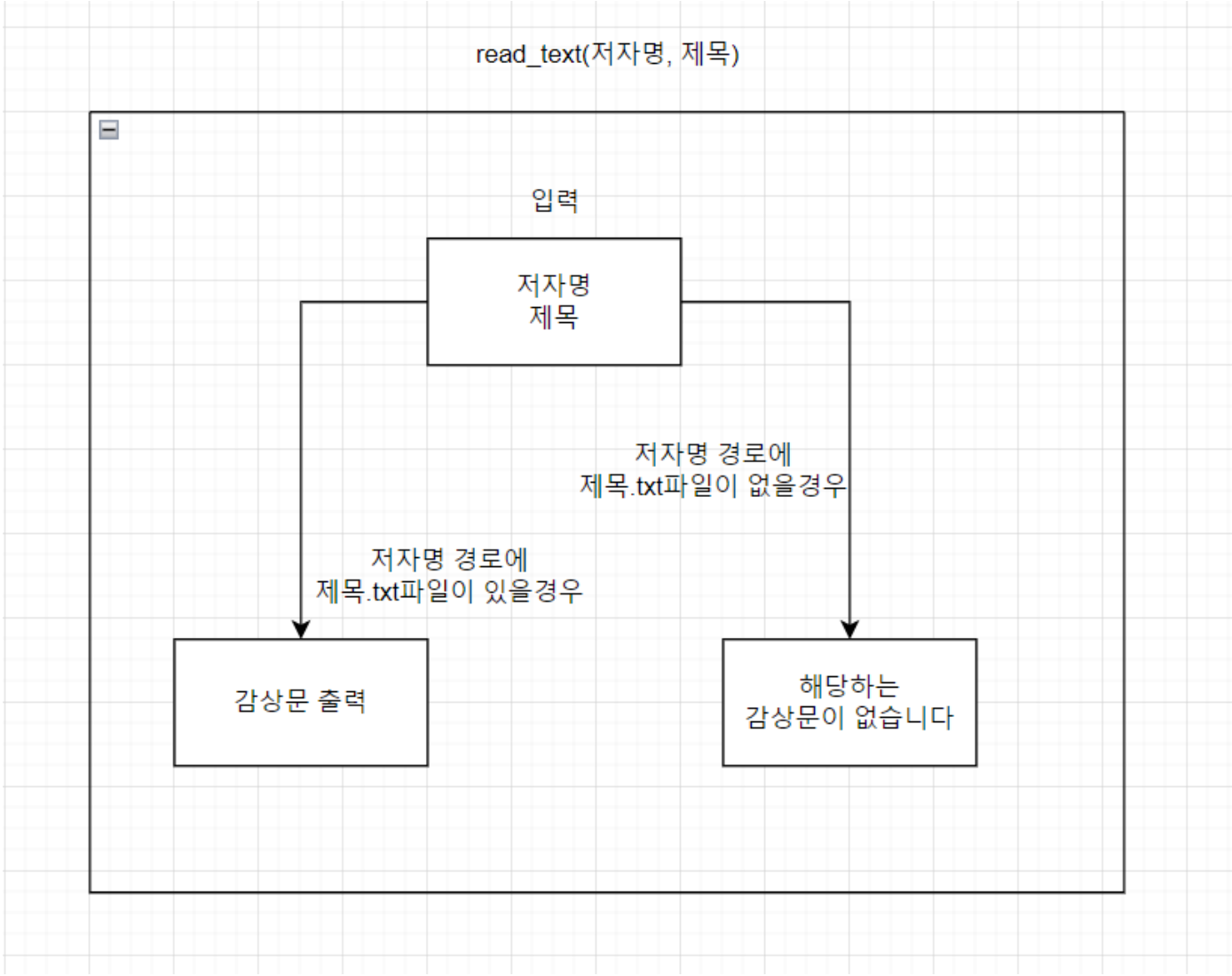
make_write_text (저자명, 제목, 내용)

```
    if (filePointer != NULL)
    {
        // 파일이 이미 존재하는 경우
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
        clearScreen();
        printf("₩n₩n");
        printf("%s에 대한 감상문이 이미 존재합니다.₩n", input_title);
        printf("₩n₩n");
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글자
        fclose(filePointer);
        return;
    }
```

```
// txt 파일에 내용 입력
fprintf(filePointer, "%s", input_content);

fclose(filePointer);
SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨

clearScreen();
printf("₩n₩n");
printf("감상문이 성공적으로 작성되었습니다.₩n");
printf("₩n₩n");
SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글자색

free(folderPath);
free(filePath);
```

Part 2 코드 설명

Read_text(저자명, 제목)



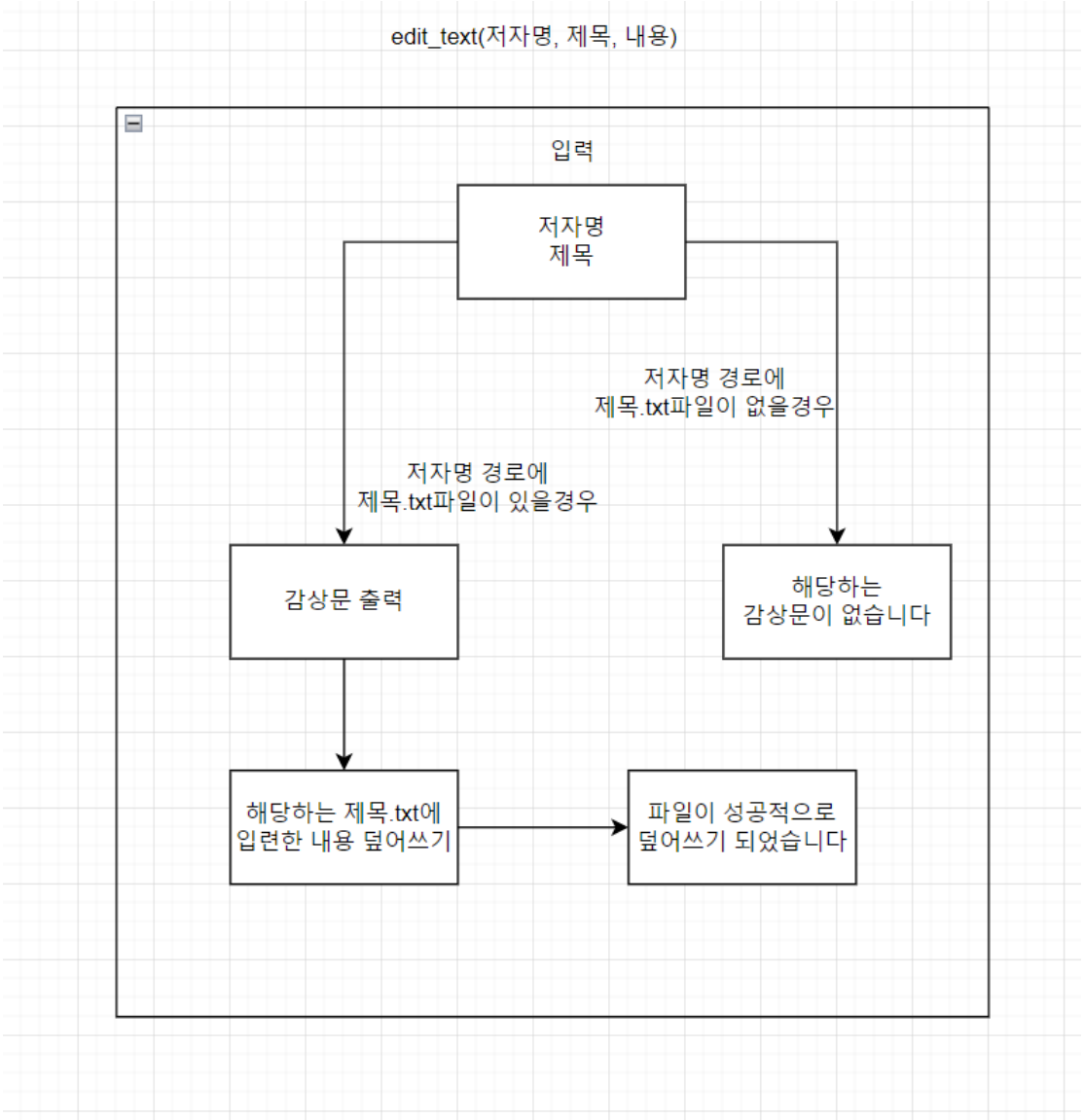
Part 2 코드 설명

Read_text(저자명, 제목)

```
void read_text(const char* folder_name, const char* input_title)
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE); //글자색 변환 코드

    char* filePath = NULL;
    filePath = (char*)malloc(MAX_CONTENT_LENGTH * sizeof(char));

    sprintf(filePath, "C:₩₩Users₩₩kimmi₩₩Desktop₩₩독후감상문₩₩%s₩₩%s.txt", folder_name, input_title); // 파일 경로 생성

    FILE* filePointer = fopen(filePath, "r"); // "r" 모드로 파일 열기

    if (filePointer == NULL)
    {
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
        clearScreen();
        printf("₩n₩n해당하는 감상문이 없습니다.₩n₩n₩n");
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글자색 기본값
        return;
    }

    printf("==================감상문 내용====================₩n");

    // 파일에서 한 줄씩 읽어와 출력하기
    char*buffer = NULL;
    buffer = (char*)malloc(MAX_CONTENT_LENGTH * sizeof(char));// 읽은 내용을 저장할 버퍼

    while (fgets(buffer, sizeof(buffer), filePointer) != NULL)
    {
        printf("%s", buffer);
    }

    printf("₩n================================================₩n");

    fclose(filePointer); // 파일 닫기
    free(filePath);
    free(buffer);
}
```

Part 2 코드 설명

edit_text(저자명, 제목)



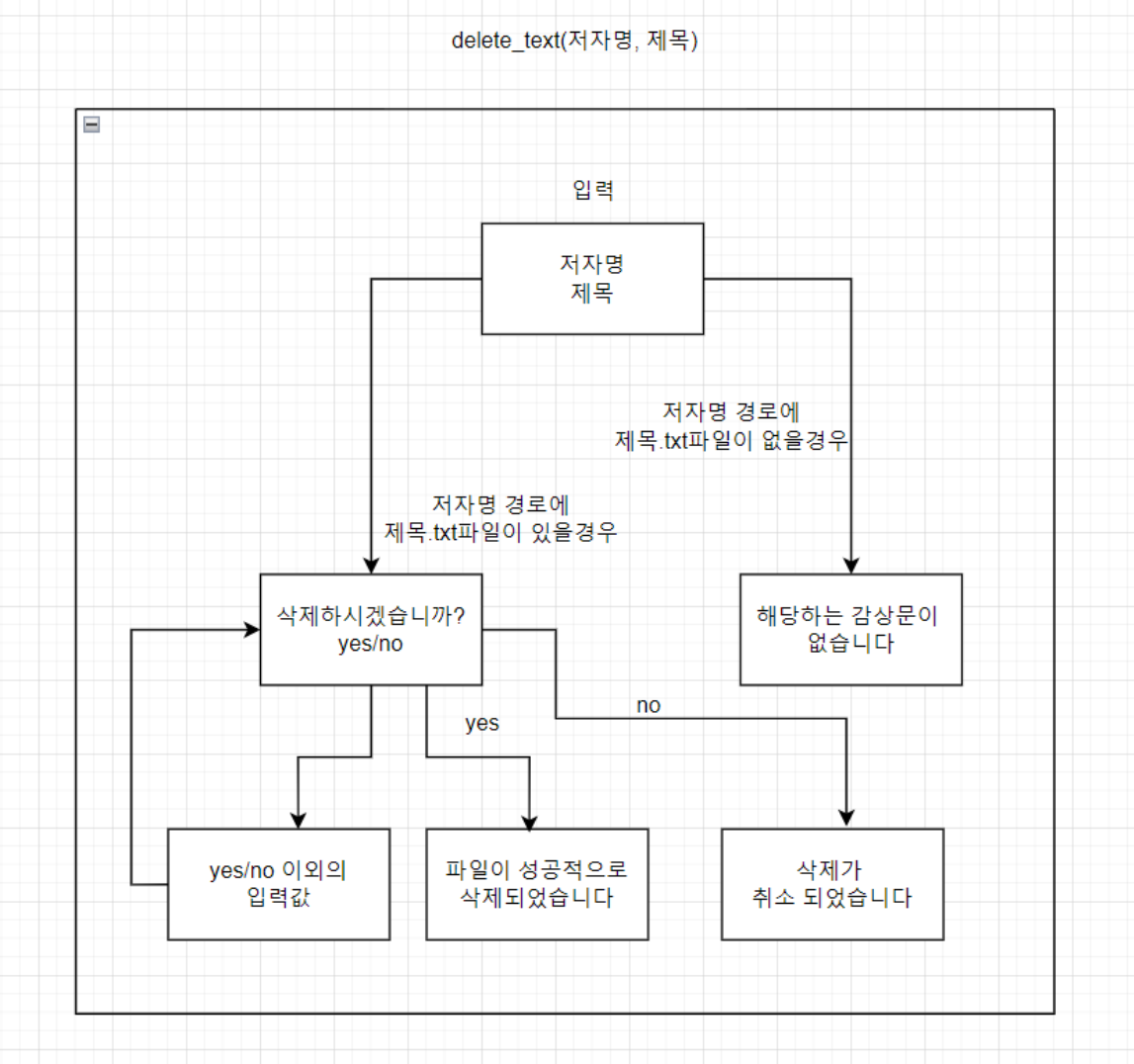
Part 2 코드 설명

edit_text(저자명, 제목)

```
void edit_text(const char* folder_name, const char* input_title, const char* input_content)
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE); //글자색 변환 코드

    char* filePath = NULL;
    filePath = (char*)malloc(MAX_CONTENT_LENGTH * sizeof(char));

    sprintf(filePath, "C:₩₩Users₩₩kimmi₩₩Desktop₩₩독후감상문₩₩%s₩₩%s.txt", folder_name, input_title); // 파일 경로 생

    FILE* filePointer = fopen(filePath, "w"); // "w" 쓰기 모드로 파일 열기

    if (filePointer == NULL)
    {
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
        clearScreen();

        printf("₩n₩n해당하는 감상문이 없습니다.₩n₩n₩n");
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글자색 기본값
        return;
    }

    // 내용 입력
    fprintf(filePointer, "%s", input_content);

    fclose(filePointer);
    clearScreen();
    printf("================================================₩n");
    printf("수정 후 정보₩n");
    printf("저자명: %s₩n", folder_name);
    printf("제목: %s₩n", input_title);
    printf("내용: ₩n%s₩n", input_content);
    printf("================================================₩n");

    SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
    printf("₩n₩n파일이 성공적으로 덮어쓰기 되었습니다.₩n₩n₩n");
    SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글자색 기본값
    free(filePath);
}
```

Part 2 코드 설명

delete_text(저자명, 제목)



Part 2 코드 설명

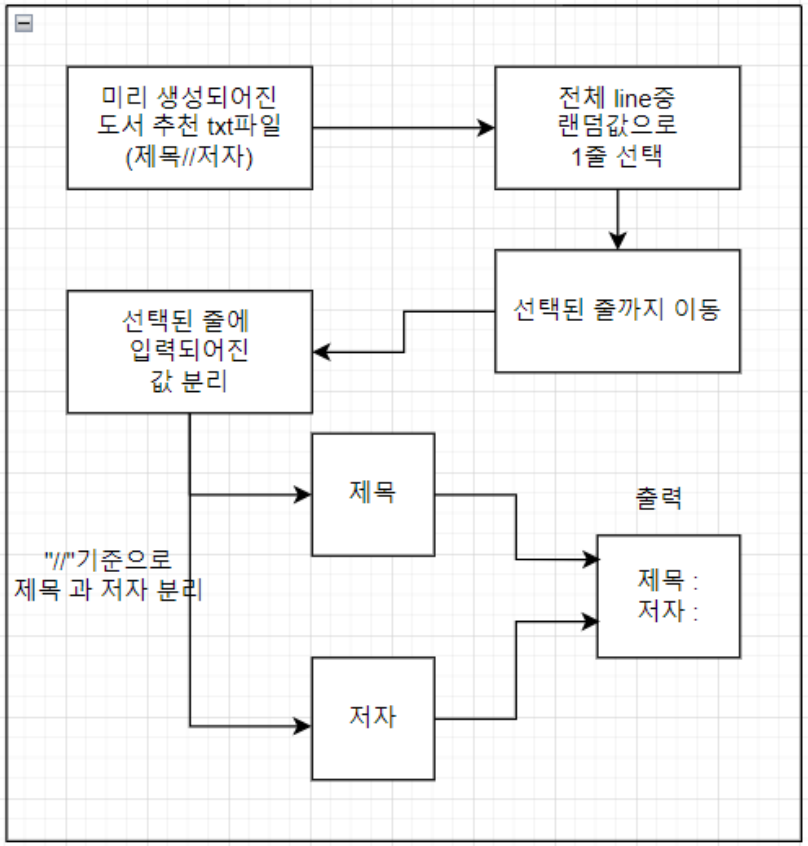
```
SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
printf("₩n=======================================₩n");
printf("삭제하시겠습니까? (복구할 수 없습니다) ₩n");
SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); //
printf("yes/no ==>> ");
scanf("%3s", check);
clearScreen();

if (strcmp(check, "yes") == 0 || strcmp(check, "no") == 0)
{
    valid_input = 1; // 입력이 있는 경우 반복문 종료
}
else
{
    SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
    printf("₩n₩n잘못된 입력입니다. 다시 입력해주세요.₩n₩n");
    SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE);

    while (getchar() != '₩n'); // 입력 버퍼를 비워줌
}
```

```
if (strcmp(check, "yes") == 0)
{
    // 파일 삭제
    if (remove(filePath) == 0)
    {
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
        clearScreen();
        printf("₩n₩n파일이 성공적으로 삭제되었습니다.₩n₩n₩n");
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글
    }
    else
    {
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
        clearScreen();
        printf("₩n해당하는 감상문이 없습니다.₩n₩n₩n");
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글
    }
}
else
{
    SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
    clearScreen();
    printf("₩n₩n삭제가 취소 되었습니다.₩n₩n₩n");
    SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); // 글자색
}
free(filePath);
```

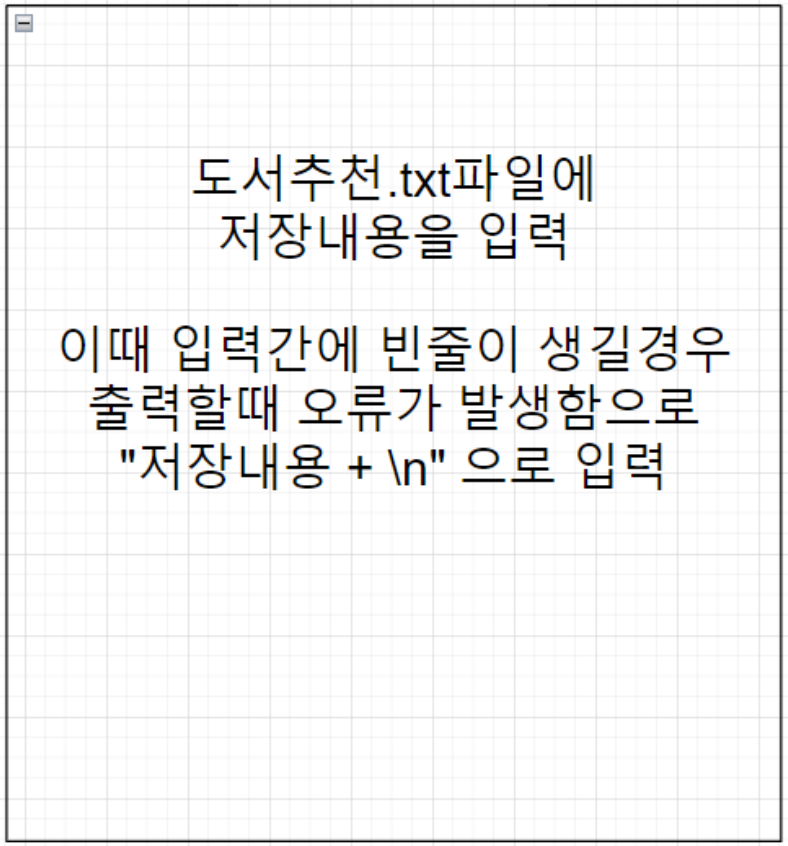
Part 2 **코드 설명**

recommend_book (파일 경로) & add_recommend_book(파일 경로, 저장내용)

recommend_book(도서추천.txt 파일 경로)



add_recommend_book(도서추천.txt 파일 경로, 저장내용)



Part 2 코드 설명

recommend_book(파일 경로)

```
// 랜덤 시드 설정
srand((unsigned int)time(NULL));

// 랜덤 줄 선택
int random_line = rand( ) % lines;


int current_line = 0;
char buffer[1024];
char* title = NULL;
title = (char*)malloc(MAX_CONTENT_LENGTH * sizeof(char)); // 제목 문자 배열
char* author = NULL;
author = (char*)malloc(MAX_CONTENT_LENGTH * sizeof(char)); // 저자 문자 배열

// 선택된 줄까지 이동 후 // 기준으로 제목과 저자를 나누어 출력
while (fgets(buffer, sizeof(buffer), file) != NULL)
{
    if (current_line == random_line)
    {
        // 줄에서 "//"를 기준으로 제목과 저자 분리
        char* slash_pos = strstr(buffer, "//");
        if (slash_pos != NULL)
        {
            *slash_pos = '₩0'; // "//" 이전 내용은 제목
            strcpy(title, buffer);
            strcpy(author, slash_pos + 2); // "//" 이후 내용은 저자
        }
        break;
    }
    current_line++;
}

clearScreen( );

// 선택된 내용 출력
printf("₩n₩n제목 : %s₩n", title);
printf("저자 : %s₩n₩n", author);

free(title);
free(author);

fclose(file);
```

add_recommend_book(파일경로, 제목//저자)

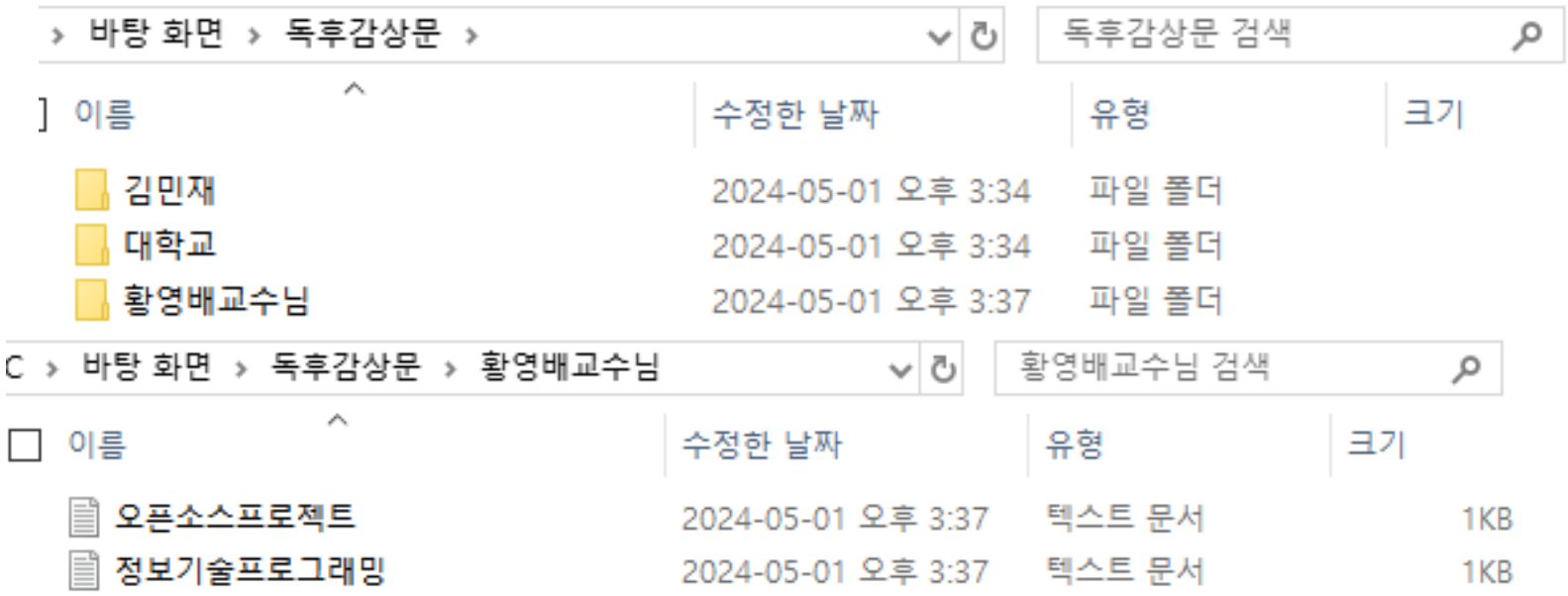
```
void add_recommend_book(const char* filename, const char* new_data)
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE); //글자색 변환 코드
    FILE* file = fopen(filename, "a");
    if (file == NULL)
    {
        clearScreen( );
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY); // 붉은색 글씨
        perror("파일 열기 실패");
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE); //
        return;
    }

    // 데이터를 파일에 추가
    fprintf(file, "%s₩n", new_data);
    fclose(file);
}
```

Part 3
실행 화면

Part 3 실행화면


==========
1. 감상문 작성
2. 감상문 출력
3. 감상문 수정(재입력)
4. 감상문 삭제
5. 책 추천
6. 책 추천 목록 추가
0. 종료
==========
원하는 메뉴를 입력하세요 :

Part 3 실행화면



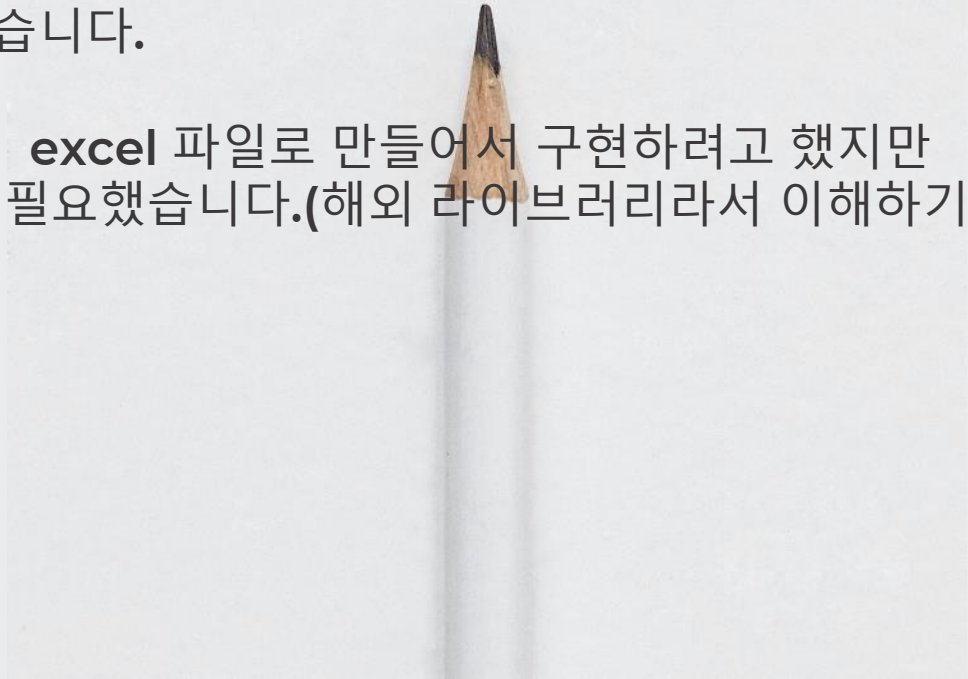
감상문을 작성하게 되면 저자명 폴더 안에 같은 저자의 독후감상문이 모이게 됩니다.

아쉬웠던점

edit_text 함수에서 감상문 내용을 수정할 때 수정한 내용을 덮어 쓰는게 아닌 기존의 감상문을 출력하여 커서로 이동시켜 수정하고 싶은 부분만 수정하고 싶었지만 콘솔창에서 커서이동은
외부 라이브러리를 사용해야 하는데 외부라이브러리를 활용하기가 쉽지 않았습니다.

GUI를 구현하고 싶었지만 **C**언어에서 **GUI**를 구현하기가 힘들고 외부라이브러리가 필요하고 또한 찾기도 활용하기도 어려웠습니다.

도서 추천해줄 때 초기에는 **excel** 파일로 만들어서 구현하려고 했지만
이 또한 외부라이브러리가 필요했습니다.(해외 라이브러리라서 이해하기도 힘들었습니다)



https://github.com/Minjea31/opensource_midproject

GitHub

감사합니다

2021042040
지능로봇공학과
김민재

담당교수 : 황영배 교수님