C++프로그래밍

일기관리 프로그램

진척 보고서 #3

제출일자: 2024/12/15
제출자명: 윤상권
제출자학번: 213958

1. 프로젝트 목표

1) 배경 및 필요성

개인마다 컴퓨터를 쓰는 경우도 있지만, 가족들끼리 함께 컴퓨터 운영체제 계정을 공유해서 쓰는 경우가 있다. 그럴 때마다 컴퓨터에 써둔 개인적인 일기나 기록들을 다른 가족들이 볼까봐 꼭꼭 숨겨두는 경우가 있다. 그렇지만 파일 암호와 프로그램으로 내가 쓴 일기나 다른 내용들을 암호화 한다면, 다른 사람들이 볼 걱정으로부터 자유로워 질 수 있다. 또, 내가 쓴 일기장을 추가하거나 더 기록하고 삭제하고 검색하는 기능이 있어 일기를 더욱 편하게 관리할 수 있게 된다.

2) 프로젝트 목표

일기장을 관리하는 프로그램을 만든다. 손쉽게 일기를 쓰거나 추가하거나 삭제할 수 있으며 검색 또한 가능하다. 일기들은 간단한 암호화 알고리즘을 사용해서 암호화되거나 복호화 한다.

3) 차별점

공용 컴퓨터에서 쓴 일기를 손쉽게 관리 할 수 있다. 일기의 내용을 암호화 할 수 있고, 검색을 통해 내가 보고 싶었던 요일을 검색할 수 있다. 또, 일기에 추가한 태그를 통해 태그 별로도 일기를 볼 수 있는 것이다.

2. 기능 계획

1) 일기를 쓰는 기능

(1). 사용자가 새로운 일기를 작성하고 저장하게 한다.

(2). 일기를 작성하면 현재 날짜와 시간이 자동으로 기록된다

(3). 일기마다 하나 이상의 태그를 추가하여 분류 및 검색을 용이하게 한다.

(4). 일기 저장 시 암호화 여부를 선택할 수 있으며, 선택한 경우 일기의 내용이 암호화 되어 저장된다.

2) 일기를 삭제하는 기능

(1). 사용자가 삭제하려는 일기를 선택하여 삭제할 수 있다.

(2). 실수로 삭제할 수 있으므로 삭제전에 확인 메시지를 표시한다

3) 일기를 추가하는 기능

(1) 기존 일기의 내용을 수정하거나 태그를 변경할 수 있게 한다.

4) 일기를 검색하는 기능

(1) 날짜로 검색

- 특정 연도, 특정 월, 특정 일 별로 날짜를 검색 할 수 있다.

(2) 태그로 검색

추가된 태그를 기준으로 일기를 검색한다.

5) 암호화 및 복호화 기능

(1) 일기의 내용을 암호화 한다.

6) 로그인 시스템

(1) 최초 프로그램 실행 시 암호 키를 입력 받는다.

3. 진척사항

1) 기능 구현

(1) 일기를 쓰는 기능

- 입출력

1) 새로운 일기를 작성하고 저장하게 함. 이때, 시간과 날짜가 자동으로 기록됨.1)-(1),(2)

입력:

main.cpp

format this_diary : 일기에 작성할 내용들을 정의한 구조체, 사용자가 직접 입력해야 하는 일기 내용과 태그에 대한 정보를 가지고 있다.

this_diary.strContent : 일기의 내용을 가진다.

this_diary.tag : 태그에 대한 정보를 가진다. 공백으로 구분되며, 아무것도 입력하지 않을 때에는 "NONE" 태그가 붙게 된다.

diary.cpp

string dateTime : 일기가 저장될 때의 날짜와 시간의 정보가 저장된다.

string filename : 일기가 저장될 파일의 이름에 대한 정보가 저장된다.

int counter : 만약 같은 날짜에 생성된 일기가 있다면 구분을 위해 순서를 붙여 저장하기 위한 변수이다.

출력: 파일에 내가 새로 쓴 일기가 저장된다.

2) 일기에 태그를 추가해 저장한다. 1)-(3)

입력:

main.cpp

format this_diary.tag : 사용자가 정의한 태그를 저장한다. 공백으로 구분 가능하다.

출력 : 위에 일기가 파일에 저장될 때, 최상단에 같이 저장된다.

3) 일기를 작성 후 암호화 여부를 선택하여 일기 내용을 암호화한다. 1)-(4)

입력 :

diary.cpp

암호화 하는 함수 Caesar()의 인자에 평문과, key값이 전달 된다. key값은 얼마나 shift할지에 대한 정보를 가지고 있다.

string encrypted : 평문의 내용을 복사하여 저장한다.

string specialChars : 특수문자 또한 암호화 하기 위해 암호화 대상인 특수문자들을 모아둔 변수이다.

char base : 대소문자를 구분하여 변경해주기 위한 변수이다. 바꾸려는 character가 대문자인지 소문자인지에 따라 다른 베이스 값을 저장한다. 아스키코드에서 'A'가 먼저 나오고 대문자와 소문자 사이에 간격이 있기 때문에 필요한 변수이다.

auto pos : String에 find메소드를 사용하여 위에서 정의한 현재 character가 위에서 정의한 specialChars에 있는지 검사한다. 만약 없다면 string::npos 로 정의되는 상수를 반환하게 된다.

출력: 평문이 암호화 되어 다시 반환된다.

- 설명

1) 먼저, 사용자가 로그인을 하고 나서, menu에서 일기 작성을 위해 1을 선택 했다고 하자, 이때 "diary.h"에서 정의한 구조체 format을 하나 만든다. format은 일기장에 사용자가 적어야 하는 내용을 저장해 두는 구조체다. string클래스의 strContetn와 tag 두개가 있다. strContent는 작성한 일기를 저장해 두는 변수이고, tag은 이 일기에 달아둘 tag를 저장해두는 변수이다. cin.ignore() 함수를 사용하여 혹시 모를 개행을 지워주고, getline함수를 이용해서 내용을 입력 받는다. 이때, 엔터를 치게 되면 입력이 종료된다. tag또한 마찬가지로 내용을 입력 받는다. 그리고 이후에는 암호화를 할 것인지 묻게 된다. yn_choice()함수는 yes or no의 대답을 받아 yes면 true를 no면 false를 반환하는 함수이다. 대답이 y or n가 아니면 다시 입력을 받아주는 역할도 한다. 이 값이 true가 되면 caesar()함수를 호출한다. 이때 전달해주는 인자는 평문인 strContent와 class로 정의한 admin객체에서 get_key()함수를 호출하여 키를 받아서 전달한다.

caesar함수 내부에서 일어나는 연산은 다음과 같다. 먼저, encrypted 변수에 전달 받은 평문을 저장해 둔다. 이 후에 for_each문을 사용하여 문자열의 character를 하나 전달 받는데 이때, '&'연산자를 사용하여 변수 c의 변경 사항이 encrypted에 저장되게 한다.

첫번째 if문에서는 c가 알파벳인지를 검사한다. 알파벳이라면 다시 한번 비교 연산을

진행하는데 만약, 소문자라면 base에 'a'를 저장하게 되고, 대문자에는 'A'를 저장한다. 이후에 실적인 암호화를 적용시키는데 c에서 base인 'a'의 값인 97만큼 빼서 'a'가 0이 되게 한다음 문자의 shift연산을 적용시킨다. 만약 c가 'z'라고 할 때 오른쪽을 1만큼 shift 시키게 하면 c에는 'a'가 저장 되어야 하는데 아스키 코드에서는 'z'에서 1만큼 증가 시키면 'a'가 아닌 다른 값이 나오게 된다. 그러므로 26이 되면 다시 0이 되게 하게끔 나머지 연산을 통해 선형적인 값들이 원형으로 회전할 수 있게 한다. 이후에 다시 base를 더해 대소문자가 구분되어 변환되게 한다.

두번째 if문에서는 정수인지 구분한다. 아스키코드에서 문자형 정수는 그냥 정수 0과 그 값이 다르기에 위에 첫번째 방법과 비슷하게 암호화를 시킨다.

세번째 else에서는 기존에 선언해둔, 특수문자에 해당하는 경우 암호화를 시켜주는 함수이다. string클래스의 find함수를 통해 c가 specialChars의 문자열 중에 있는지 검사한다. 만약에 pos에 저장해둔 내용이 찾은 내용이 없다는 string::npos가 아니라면 c를 암호화 시키고 if문이 아니라면 암호화 시킬 대상이 아니므로 암호화 시키지 않고 그냥 두게 된다.

변경된 암호문을 반환하게 된다. 만약 yn_choice에서 대답이 'n'이였으면 그냥 암호화를 하지 않고 지나가게 된다. 그리고 writediary함수를 호출한다. 전달해줄 인자는 위에서 선언한 구조체 format의 this_diary이다.

writediary함수의 작동은 다음과 같다. 먼저 dateTime에는 일기내부에 저장할 "년-월-일-시-분"에 대한 문자열을 저장한다. 그리고 filename 변수에는 파일 이름을 형식에 맞춰 만든 문자열을 저장한다. 이때 호출 되는 getCurrentDate()는 위와 달리 "년-월-일"에 대한 정보만 가지고 있다. 그 다음에는 fileExists()함수를 이용해 내가 작성하려는 파일의 이름이 이미 저장되어 있는지 검사한다. 만약에 있다면 중간에 counter변수를 이용해서 번호를 추가하여 저장한다.

이후에는 일기를 형식에 맞춰서 사용자가 입력한 strContent나 tag를 추가하여 파일 스트림에 작성하고 파일을 저장한다.

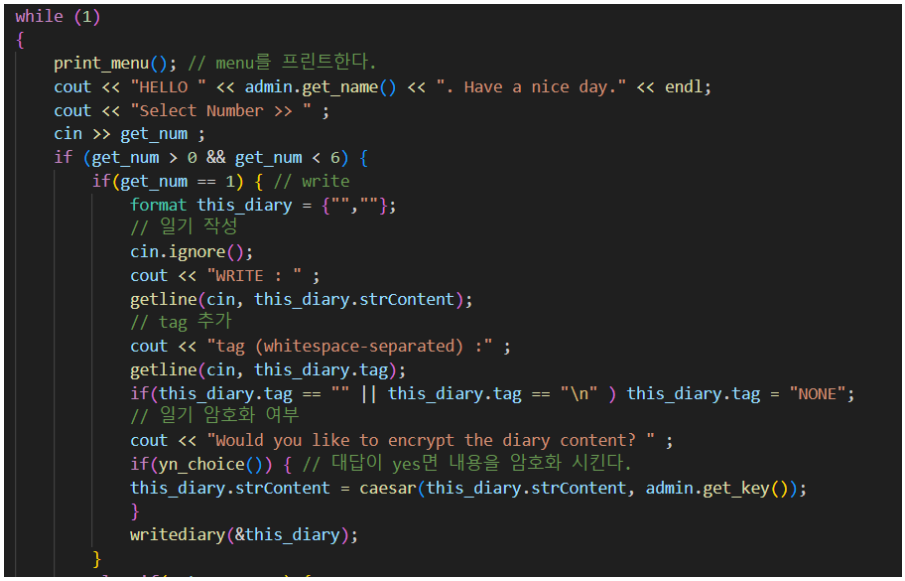
만약 파일이 정상적으로 열리지 않았다면 에러를 출력한다. 이 함수에서 반환되는 값은 없다.

- 적용된 배운 내용

구조체, 반복문, 조건문 등의 개념을 최대한 활용해서 적용해 보았다. 아직 수업에서 파일 입출력을 배우지 않아 조금 어려웠다. 기존 c 파일 입출력의 방식이 달랐기 때문이다. 나중에 파일 입출력을 배운다면 더 개선할 점이 있을 것이라 기대된다.

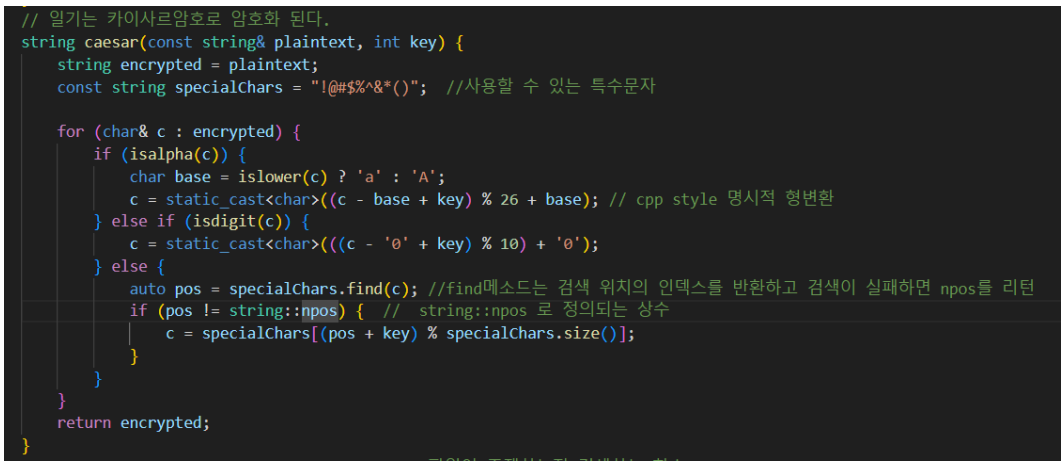
- 코드 스크린샷

main.cpp의 일기를 쓰는 부분의 코드



```
while (1)
{
    print_menu(); // menu를 프린트한다.
    cout << "HELLO " << admin.get_name() << ". Have a nice day." << endl;
    cout << "Select Number >> " ;
    cin >> get_num ;
    if (get_num > 0 && get_num < 6) {
        if(get_num == 1) { // write
            format this_diary = {"",""};
            // 일기 작성
            cin.ignore();
            cout << "WRITE : " ;
            getline(cin, this_diary.strContent);
            // tag 추가
            cout << "tag (whitespace-separated) :" ;
            getline(cin, this_diary.tag);
            if(this_diary.tag == "" || this_diary.tag == "\n" ) this_diary.tag = "NONE";
            // 일기 암호화 여부
            cout << "Would you like to encrypt the diary content? " ;
            if(yn_choice()) { // 대답이 yes면 내용을 암호화 시킨다.
            this_diary.strContent = caesar(this_diary.strContent, admin.get_key());
            }
            writediary(&this_diary);
        }
```

암호화 하는 코드



```
// 일기는 카이사르암호로 암호화 된다.
string caesar(const string& plaintext, int key) {
    string encrypted = plaintext;
    const string specialChars = "!@#$%^&*()";  //사용할 수 있는 특수문자

    for (char& c : encrypted) {
        if (isalpha(c)) {
            char base = islower(c) ? 'a' : 'A';
            c = static_cast<char>((c - base + key) % 26 + base); // cpp style 명시적 형변환
        } else if (isdigit(c)) {
            c = static_cast<char>(((c - '0' + key) % 10) + '0');
        } else {
            auto pos = specialChars.find(c); //find메소드는 검색 위치의 인덱스를 반환하고 검색이 실패하면 npos를 리턴
            if (pos != string::npos) {  //  string::npos 로 정의되는 상수
                c = specialChars[(pos + key) % specialChars.size()];
            }
        }
    }
    return encrypted;
}
```

```cpp
// 일기는 카이사르암호로 암호화 된다.
string Caesar(const string &plaintext, int key)
{
    string encrypted = plaintext;

    const string special_chars = "!@#$%^&*()"; // 사용할 수 있는 특수문자

    // 알파벳과 숫자의 키를 양수로 변환
    int alpha_key = (key % 26 + 26) % 26; // 알파벳용
    int digit_key = (key % 10 + 10) % 10; // 숫자용

    for (char &c : encrypted)
    {
        if (isalpha(c))
        {
            char base = islower(c) ? 'a' : 'A';
            c = static_cast<char>((c - base + alpha_key) % 26 + base);
        }
        else if (isdigit(c))
        {
            c = static_cast<char>(((c - '0' + digit_key) % 10) + '0');
        }
        else
        {
            auto pos = special_chars.find(c);
            if (pos != string::npos)
            {
                c = special_chars[(pos + key) % special_chars.size()];
            }
        }
    }
    return encrypted;
}
```

(12월 13일 Casesar()함수의 'xyz'에 대해 복호화 부분에 오류가 발생해 수정되었음.)

기존 함수에서 key값이 음수가 되면 원했던 아스키코드의 구간이 넘어가는 오류가 발생했다. 음수로 들어온 키라도 양수가 되게 변화하는 코드를 추가했다.

파일을 작성하는 코드

```cpp
bool fileExists(const string& fileName) { // 파일이 존재하는지 검색하는 함수
    ifstream file(fileName);
    return file.good(); //정상적으로 파일이 열리면 true를 반환...
}

void writediary(format * diaryFormat) {
    string dateTime = getCurrentDateTime();
    string fileName = "2024_diary/diary_" + getCurrentDate() + ".txt"; //일기 제목의 형식
    int counter = 1;
    while (fileExists(fileName)) {
        fileName = "2024_diary/diary_" + getCurrentDate() + "_" + std::to_string(counter++) + ".txt";
    }
    ofstream diaryFile(fileName, ios::app);

    if (diaryFile.is_open()) {
        diaryFile << "tags: " << diaryFormat -> tag << endl;
        diaryFile << "-------------------------\n";
        diaryFile << "Date & Time: " << dateTime << endl;
        diaryFile << "Content: " << diaryFormat -> strContent << endl;
        diaryFile << "-------------------------\n";
        diaryFile.close();
        cout << "Diary entry saved successfully to " << fileName << "!" << endl;
    } else {
        cerr << "Unable to open the diary file for writing." << endl;
    }
}
```

지금 날짜에 대해 문자열로 만들어서 반환해주는 함수

```
string getCurrentDate() { // 날짜만
    time_t now = time(0);
    tm *ltm = localtime(&now);

    char buffer[20];
    strftime(buffer, sizeof(buffer), "%Y-%m-%d", ltm);

    return string(buffer);
}
string getCurrentDateTime() { // 날짜랑 시간까지
    time_t now = time(0);
    tm *ltm = localtime(&now);

    char buffer[100];
    strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", ltm);

    return string(buffer);
}
```

일기에 쓸 내용 형식에 대해 선언해둔 구조체

```
using namespace std;
typedef struct format {
    string tag ;
    string strContent;

}format ;
```

(2) 로그인 시스템

- 입출력

입력 :

class My_diary admin에 이름에 "admin", 비밀번호에 "1234", key에 4가 저장된다.

string passwd : 입력 받은 passwd를 저장한다.

Int get_num : 메뉴에 선택된 값을 저장한다.

출력 :

로그인 화면이 출력되고, 로그인을 진행시킨다. 로그인이 되면 메뉴를 출력시키고 메뉴 값을 입력 받는다.

- 설명

함수 main()이 시작될 때, class My_diary의 객체 admin을 만들고 초기값을 “admin”, “1234”, 4를 준다. 순서대로 이름, 비밀번호, key값이다. 그리고 조건이 1인 while문 내부로 들어가게 된다. print_main()함수는 프로그램을 맨 처음 실행시켰을 때 콘솔에 첫 화면을 뿌려주는 함수이다. 단순히 cout을 이용해 화면 출력만 하는 함수이다. 이후에 passwd에 비밀번호를 입력 받는다. 만약 비밀번호가 “exit”이면 그냥 프로그램을 종료시킨다. 그 외에는 클래스의 check_login()함수를 이용해서 객체 내부에 있는 비밀번호와 대조시켜준다. 맞은면 true를 반환하여 로그인창에서 나가 메뉴 창으로 갈 수 있게 한다.

메뉴 부분에서는 메뉴를 출력해주는 print_menu()함수가 실행되고, 지금 사용자의 이름을 출력해 주는 동시에 무슨 작업을 할 것인지 get_num변수를 통해 입력받아 작업을 실행 시킨다.

- 적용된 배운 내용

반복문, 클래스, 조건문의 개념들을 사용했다.

- 코드 스크린샷

메인 함수 부분

```cpp
int main()
{
    My_diary admin("admin", "1234", 4);
    string passwd = "";
    while (1)
    { // 로그인 화면
        print_main();
        cout << "If you want to exit, please enter the password 'exit'." << endl;
        cout << "Please enter password >> ";
        cin >> passwd;
        if (passwd == "exit")
        { // 종료 여부를 검사한다.
            cout << "Exit my diary";
            return 0;
        }
        if (admin.check_login(passwd))
            break; // 유저의 페스워드가 맞으면 다음 화면으로 넘어간다.
        else
            cout << "Incorrect password" << endl;
    }

    int get_num = 0;
    while (1)
    {
        print_menu(); // menu를 프린트한다.
        cout << "HELLO " << admin.get_name() << ". Have a nice day." << endl;
        cout << "Select Number >> " ;
        cin >> get_num ;
        if (get_num > 0 && get_num < 6) {
```

print_main()

```cpp
void print_main()
{
    cout << "\n\n";
    cout << "-----------------------------------------------------------------------------------------------------------" << endl;
    cout << "       ###       ###   ##      ##   ########     ########       ####        #######     ##      ##" << endl;
    cout << "       ## #     # ##    ##    ##    ##     ##       ##         ##  ##       ##    ##     ##    ##" << endl;
    cout << "       ##  #   #  ##     ##  ##     ##      ##      ##        ##    ##      #######       ##  ##" << endl;
    cout << "       ##   # #   ##      ##        ##     ##       ##       ## #### ##     ##    ##       ##" << endl;
    cout << "       ##    #    ##      ##        ########     ########   ##        ##    ##     ##      ##" << endl;
    cout << "-----------------------------------------------------------------------------------------------------------" << endl;
}
```

print_menu()

```cpp
void print_menu(){
    cout << "\n\n";
    cout << "-----------------------------------------------------------------------------------------------------------" << endl;
    cout << "1. Write a diary" << endl;
    cout << "2. Delete a diary" << endl;
    cout << "3. Edit a diary" << endl ;
    cout << "4. Search a diary" <<endl;
    cout << "5. Exit" << endl ;
    cout << "-----------------------------------------------------------------------------------------------------------" << endl;

}
```

class My_diary

```
class My_diary
{
    private :
    string my_password ;
    string my_name;
    int my_key;

    public :
    My_diary(string name = "None", string passwd = "0000", int key = 0){
        my_name = name;
        my_password = passwd;
        my_key = key ;
    }
    bool check_login(string inputPass) ;
    int get_key() ;
    string get_name();
};
```

```
#include"user.h"

bool My_diary::check_login(string inputPass){
    if (my_password == inputPass) return true;
    else return false;
}

int My_diary::get_key(){
    return my_key;
}

string My_diary::get_name(){
    return my_name;
}
```

(3) 일기를 삭제하는 기능

- 입출력

1) 사용자가 삭제하려는 일기를 선택하여 삭제할 수 있다.

2) 실수로 삭제할 수 있으므로 삭제 전에 확인 메시지를 표시한다.

입력:

RemoveDiary() 함수에 file_name이 입력으로 들어온다. 이때, file_name이 존재하는 파일인지에 대한 검사는 호출자에서 검사를 하고 함수를 사용해야 한다.

출력:

내가 선택한 file_name의 파일이 지워진다.

- 설명:

처음 메뉴 화면에서 2번을 선택하면 파일을 지울 수 있는 화면으로 넘어가게 된다. 이때, 문자열 벡터인 file_list에 FileList()함수가 호출되어 리스트로 저장된다. 이 함수는 저장 폴더로 지정된 곳에서 존재하는 파일의 이름을 가져와 스트링 벡터에 저장하고 반환해주는 함수이다.

폴더에 저장되어 있는 일기들을 보여준 이후에는 rm_file이라는 변수에 삭제할 파일의 이름을 입력 받는다. 이때 ExistFile()함수를 옳바르게 입력을 했는지, 파일이 존재하는 파일인지 검사를 하게 된다. 검사를 진행한 후에 입력이 "No"였다면 동작을 그만하고 다시 메뉴 선택창이 나오게 한다. 또한, "Are you sure you want to delete [" << rm_file << "] file? : " 이라는 문구를 뜨게 하여 정말로 선택한 일기를 지울 것인지 묻게 하고 IsYes()함수를 통해 대답이 yes라면 지우고 아니면 다시 메뉴 창이 뜨게 한다.

- 적용된 배운 내용

벡터 연산자, for_each문

- 코드 스크린샷

FileLsit()

```
vector<string> FileList()
{
    vector<string> file_list;
    fs::directory_iterator itr(fs::current_path() / save_directory);
    while (itr != fs::end(itr))
    {
        const fs::directory_entry &entry = *itr;
        file_list.push_back(entry.path().filename().string());
        itr++;
    }
    return file_list;
}
```

RemoveDiary()

```
// 존재하는 file_name임을 가정
void RemoveDiary(string file_name)
{

    fs::remove(file_name);
}
```

Main()의 menu 2번

```
else if (get_num == 2)
{
    vector<string> file_list = FileList();
    int i = 0;
    string rm_file = "";
    cout << "FILE LIST : " << endl;
    for (string file_name : file_list)
    {
        cout << "[" << ++i << "] : " << file_name << std::endl;
    }
    cout << endl;
    cout << "Which files should be removed? " << endl;
    cout << "If you enter \"No\" nothing will be removed :" ;
    cin >> rm_file;
    rm_file = save_directory + rm_file;
    while(!ExistsFile(rm_file)){
        if(rm_file == save_directory+"No") break;
        cout << "Please enter the correct file name :";
        cin >> rm_file;
        rm_file = save_directory + rm_file;
    }
    if(rm_file == save_directory+ "No") continue;
    cout << "Are you sure you want to delete [" << rm_file << "] file? : " << endl;
    if (IsYes()) RemoveDiary(rm_file);
    else continue;
    cout << "REMOVED!!!!" << endl;

}
```

(4) 일기를 추가하는 기능

- 입출력

1) 기존 일기의 내용을 수정하거나 태그를 변경 할 수 있게 한다.

입력: EditText(), EditTag()함수는 공통적으로 file_name을 입력받는다. 이때 file_name은 폴더에 존재하는 파일이어야 한다. 그리고 둘 다 파일의 일부를 읽어와 편집하고, SaveEdit함수를 호출하는데 파일 이름과, 바꿀 내용, tag인지 context인지 구분하는 int형 정보를 입력한다.

출력: 파일에 변경된 내용이 저장된다.

- 설명

처음에 FIleList()함수가 호출되며 작업 폴더에 어떤 일기가 있는지 보여준다. 그리고 편집할 일기의 이름을 입력 받는다. 편집할 파일이 유효한 파일인지 검사를 한 후에 tag를 편집할 것인지 일기의 내용을 편집할 것인지 입력을 받고 그에 따라 알맞은 함수를 호출한다. EditTag함수에서 첫번째로 GetLine함수가 호출된다. GetLine함수는 태그가 저장된 줄에서 태그만 잘라와 반환한다. 이를 저장해두고 EditLine()함수를 호출한다. EditLine()함수는 문자열을 수정하게 하는 함수이다. Backspace로 문자열을 지울 수도 있고 좌우 방향키를 누르고 입력을 하면 문자열 사이에 내가 입력한 문자가 삽입 된다. Enter를 누르면 작업이 종료되고 바뀐 문자열이 반환된다. 이후에는 SaveEdit함수가 호출되어 바뀐 내

용으로 파일을 다시 쓰게 된다.

EditText함수도 위와 같은 작업으로 진행된다. 다만, 저장된 줄 위치가 tag는 index 0이고 일기 내용은 index 3에 저장되어 있다.

- 적용된 배운 내용

String클래스의 메소드, 반복문, 키 입력에 따른 반복문 제어

- 코드 스크린샷

Main()의 menu 3번

```
else if (get_num == 3)
{
    string file_name ="";
    vector<string> file_list = FileList();
    int i = 0;
    string edit_file = "";
    cout << "FILE LIST : " << endl;
    for (string file_name : file_list)
    {
        cout << "[" << ++i << "] : " << file_name << std::endl;
    }
    cout << endl;
    cout << "Which files should be changed? " << endl;
    cout << "If you enter \"No\" nothing will be edited :" ;
    cin >> edit_file;
    edit_file = save_directory + edit_file;
    while(!ExistsFile(edit_file)){
        if(edit_file == save_directory+"No") break;
        cout << "Please enter the correct file name :";
        cin >> edit_file;
        edit_file = save_directory + edit_file;
    }
    if(edit_file == save_directory+ "No") continue;
    int input = 0;
    cout << "Which should I modify, tag (1) or diary (2):" ;
    cin >> input ;
    while(true){
        if (input == 1){
            EditTag(edit_file);
            break;
        }
        else if (input == 2){
            EditText(edit_file);
            break;
        }
        else
        {
            cout << "1과 2중 하나로만 입력해주세요" ;
            cin >> input ;
        }
    }
```

EditTag

```
// 태그를 수정하는 함수
void EditTag(string file_name)
{
    string line = "";
    // 일기 내용을 읽어와 저장
    line = GetLine(file_name, 0); // 태그
    cout << "태그를 수정하세요! (Enetr을 누르면 종료!)" << endl;
    line = EditLine(line);
    SaveEdit(file_name, line, 0);
    cout << "completed" << endl;
}
```

EditText

```
// file_name의 유효성은 caller인 main함수에서 진행
// 일기 내용을 수정하는 함수
void EditText(string file_name)
{
    string line = "";
    // 일기 내용을 읽어와 저장
    line = GetLine(file_name, 3); // 일기 내용
    cout << "일기를 수정하세요! (Enetr을 누르면 종료!)" << endl;
    line = EditLine(line);
    SaveEdit(file_name, line, 3) ;
    cout << "completed" << endl;
}
```

EditLine

```
// string 한줄을 편집하여 반환하는 함수
string EditLine(string line)
{
    unsigned char ch;
    int cursor_point = line.size(); // 커서 끝을 문장 끝으로 설정.
    cout << line;
    cout.flush();
    while (true)
    {
        ch = _getch();
        if (ch == '\r')
            break;
        else if (ch == 8)
        { // 8은 backspace를 뜻한다.

            // erase(size_type index = 0, size_type count = npos);
            line.erase(cursor_point - 1, 1);
            cursor_point--;
        }
        else if (ch == 224)
        {                   // 화살표 키를 입력 받는다. 이때 필요인수는 2개다
            ch = _getch(); // 추가 입력 처리
            if (ch == 75 && cursor_point > 0)
                cursor_point--;
            else if (ch == 77 && cursor_point < line.size())
                cursor_point++;
        }
        else
        {
            line.insert(line.begin() + cursor_point, ch); // iterator insert(iterator pos, char ch);
            cursor_point++;
        }
        cout << string(line.size(), ' '); // 현재 줄 지우기
        cout << "\r";                     // 다시 줄 처음으로 이동
        cout << line;                     // 현재 상태 출력
        cout.flush();
    }
    return line;
}
```

GetLine

```
// 일기에서 한줄 읽어와 ': '를 기준으로 자르고 내용물만 반환해주는 함수
// 0 -> 태그 , 2-> 날짜 , 3-> 내용
string GetLine(string file_name, int idx)
{
    ifstream file(file_name);
    vector<string> lines;
    string line = "";
    // 일기 내용을 읽어와 저장
    while (std::getline(file, line))
    {
        lines.push_back(line);
    }
    file.close();
    line = lines[idx].substr(lines[idx].find(':') + 1); // ':'옆 공백 포함해서 자른다.
    return line;
}
```

SaveEdit

```
//  수정된 파일을 저장하는 함수
// 0 -> 태그 , 2-> 날짜 , 3-> 내용
void SaveEdit(string file_name, string edit_line, int idx) {
    // 파일 읽기
    ifstream file(file_name);
    vector<string> lines;
    string line = "";

    // 파일 내용을 벡터에 저장
    while (getline(file, line)) {
        lines.push_back(line);
    }
    file.close();

    // 기존 줄을 ':' 이후에 새로운 내용으로 편집
    line = lines[idx].substr(0, lines[idx].find(':') + 1); // ':'까지 자름
    lines[idx] = line + edit_line;

    // 파일 쓰기
    ofstream out_file(file_name, ios::out);
    for (const string& line : lines) {
        out_file << line << endl;
    }
    out_file.close();
}
```

(4) 일기를 검색하는 기능

- 입출력

1) 특정 연도, 특정 월, 특정 일 별로 날짜를 검색 할 수 있다.

입력: DateInputHandler 함수를 통해 검색할 날짜를 입력 받는다. 연도 입력은 필수로 입력을 받는다. 그리고 경우에 따라서 연도를 검색하거나, 연도와 월까지

검색을 하거나 년, 월, 일의 데이터 모두를 사용하여 검색을 지원한다.

출력: 입력에 해당하는 일기들이 Search()함수로 찾아지고 verctor<string> 형태로 반환된다. 그리고 PrintResult()함수로 반환된 일기 목록이 출력된다.

- 설명

메뉴 선택 부분에서 4를 입력하면 일기 검색하는 분기로 넘어 올 수 있다. 먼저 이진 검색 트리를 만들어 폴더 내의 일기들을 날짜 데이터를 기준으로 이진 검색 트리가 만들어 진다. 이때, 만들어진 트리의 root가 NULL이라면 예외를 던진다. 이후에 날짜로 검색할 것인지, 태그로 검색을 볼 것인지를 선택하게 된다. 만약 선택이 날짜라면 DateInputHandler()함수가 호출 된다. 이 함수는 오버로딩된 함수 Search를 적절하게 입력값을 주어 검색을 하기 위해 만들어 졌다. 연도와 월, 일을 입력 받은 후에 월에 해당하는 값이 -1이라면 연도로만 검색을 하고, 또 연도와 월만을 검색하거나 연도와 월, 일까지 검색을 하도록 한다. 검색한 후에, 반환된 vector<string>을 입력으로 받는 PrintResult함수를 호출해 결과를 출력하도록 한다.

- 적용된 배운 내용

연산자 오퍼레이터, 함수 오버로딩, exception

- 코드 스크린샷

Main.cpp -> 메뉴 검색 부분

```
else if (get_num == 4)
{
    cout << "Searching" << endl;

    bsTree *root = GetSearchTree();
    // 폴더에 일기가 없으면 예외를 던진다.
    if (root == nullptr)
        throw NullPointerException();
    cout << "complete" << endl;

    string choice;
    cout << "date or tag : ";
    cin >> choice;
    if (choice == "date")
    {
        DateInputHandler(root);
    }
    else if (choice == "tag")
    {
        PrintAllTags(BuildTagFileMap());
    }
    else
        cout << "Invalid input." << endl;
    if (root != nullptr)
    {
        DeleteTree(root);
        root = nullptr;
    }
}
```

diary.cpp GetSearhTree()

```
bsTree *GetSearchTree()
{
    bsTree *root = nullptr;
    vector<string> lists = FileList();
    vector<fdata> list_data;
    for (string list : lists)
    {
        fdata fdata;
        fdata.file_name = list;
        sscanf(list.c_str(), "diary_%d-%d-%d", &fdata.year, &fdata.month, &fdata.day);
        list_data.push_back(fdata);
    }

    for (fdata fdata : list_data)
    {
        root = Insert(fdata, root);
    }
    return root;
}
```

Main.cpp 예외 정의 및 catch

```
class NullPointerException : public exception
{
public:
    const char *what() const noexcept override
    {
        return "File list is NULL";
    }
};
```

```
catch (const NullPointerException &e)
{
    cout << "empty file list : " << e.what() << endl;
    continue;
}
```

Main.cpp DateInputHandler()

```cpp
void DateInputHandler(bsTree *root)
{
    int year = -1, month = -1, day = -1;

    cout << "Enter year: ";
    cin >> year;

    cout << "Enter month (or -1 to skip): ";
    cin >> month;

    cout << "Enter day (or -1 to skip): ";
    cin >> day;
    vector<string> result;
    // 입력값에 따라 함수 호출
    if (month == -1)
    {
        result = Search(year, root); // 년도만 입력
        PrintResult(result);
    }
    else if (day == -1)
    {
        result = Search(year, month, root); // 년도와 월 입력
        PrintResult(result);
    }
    else
    {
        result = Search(year, month, day, root); // 년도, 월, 일 모두 입력
        PrintResult(result);
    }
}
```

Main.cpp PrintResult()

```cpp
// 결과를 출력하는 함수(스트링 벡터)
void PrintResult(const vector<string> &results)
{
    if (results.empty())
    {
        cout << "No results found." << endl;
    }
    else
    {
        cout << "Results:" << endl;
        for (const string &result : results)
        {
            cout << result << endl;
        }
    }
}
```

2) 태그를 기준으로 일기를 검색한다

입력: 요구되는 입력이 없다.

출력: 각 태그마다 해당되는 일기파일의 이름이 출력된다.

- 설명

제일 먼저 BuildTagFileMap() 함수가 호출된다. 이 함수에서는 tag를 key값으로 하는 map에 일기 제목들을 vector<string>으로 저장한다. 기존에 정의한 FileLsit() 함수로 저장된 일기 제목들을 불러 온다음에, 불러온 파일이 정상적으로 있는 파일이라면, 마찬가지로 기존에 정의해둔 GetLine()함수로 태그가 저장된 파일의 첫 번째 줄을 가져와, SplitBySpace()함수를 통해 공백을 기준으로 tag를 분리한 뒤 map에 tag를 key로 등록하고 해당 파일의 이름을 저장한다. 그리고 반환된 map을 적절하게 출력해주는 함수인 PrintAllTags() 함수를 통해 태그별로 해당되는 파일의 이름들이 출력된다.

- 적용된 배운 내용

Map

- 코드 스크린샷

Main.cpp 메뉴 검색 부분의 tag선택 시 분기.

```
else if (choice == "tag")
{
    PrintAllTags(BuildTagFileMap());
}
```

diary.cpp BuildTagFileMap()

```
map<string, vector<string>> BuildTagFileMap()
{
    map<string, vector<string>> tag_file_map;

    for (auto file : FileList())
    {
        file = save_directory + file;
        cout << file << endl;
        if (ExistsFile(file))
        {
            vector<string> tags = SplitBySpace(GetLine(file, 0));
            for (string tag : tags)
            {
                tag_file_map[tag].push_back(file);
            }
        }
    }
    return tag_file_map;
}
```

diary.cpp SplitBySpace()

```
vector<string> SplitBySpace(const string &list_strings)
{
    vector<string> results;
    istringstream stream(list_strings);
    string word;
    while (stream >> word)
    {
        results.push_back(word);
    }
    return results;
}
```

diary.cpp PrintAllTags()

```
void PrintAllTags(const map<string, vector<string>> &tag_file_map)
{
    cout << "All tags and associated files:" << endl;
    for (const auto &[tag, files] : tag_file_map)
    {
        cout << "Tag: " << tag << endl;
        for (const string &file : files)
        {
            cout << " - " << file << endl;
        }
    }
}
```

(5) 일기를 암호화, 복호화하는 기능

- 입출력

입력: 일기 이름이 함수의 입력값으로 들어간다.

출력: 선택에 따라 일기 파일의 일기 내용이 암호화 되거나 복호화 된다.

- 설명

메뉴에서 암호화 복호화를 선택하여 들어가면, 먼저 암호화를 할 것인지 복호화를 할 것인지 선택을 한다. 그 다음, 수정할 파일을 입력받고 존재하는지 ExistsFile()함수로 검사를한다. 만약 존재 하지 않는다면 실패 메시지를 띄우고 메뉴 창으로 돌아간다. 반대로 파일이 존재 한다면, GetLine()함수를 통해 기존 일기 내용을 불러와 저장한 뒤 Casesar()암호를 통해 일기 내용을 암호화 하거나 복호화 한다. 암호화의 key값은 user클래스의 get_key() 메소드를 통해 얻을 수 있고 암호화는 그대로 값을 쓰면 되고 복호화는 '-'기호를 붙이면 된다. 바뀐 내용을 SaveEdit()함수를 통해 다시 저장한다.

(12월 13일 Casesar()함수의 'xyz'에 대해 복호화 부분에 오류가 발생해 수정되었음.)

- 코드 스크린샷

```
else if (get_num == 5)
{
    string select;
    string file_name;
    cout << "Encryption OR Decryption FILE!!!" << endl;
    cout << "Please input \"enc\" or \"dec\" >> ";
    cin >> select;
    if (select == "enc")
    {
        cout << "Input file name : ";
        cin >> file_name;
        file_name = save_directory + file_name;
        if (!ExistsFile(file_name))
        {
            cout << file_name << " does not exist" << endl;
            continue;
        }
        string context = GetLine(file_name, 3);
        string enc_text = Caesar(context, admin.get_key());
        SaveEdit(file_name, enc_text, 3);
        cout << "it's completed" << endl;
    }
    else if (select == "dec")
    {
        cout << "Input file name : ";
        cin >> file_name;
        file_name = save_directory + file_name;
        if (!ExistsFile(file_name))
        {
            cout << file_name << " does not exist" << endl;
            continue;
        }
        string enc_text = GetLine(file_name, 3);
        string dec_text = Caesar(enc_text, -(admin.get_key()));
        SaveEdit(file_name, dec_text, 3);
        cout << "it's completed" << endl;
    }
```

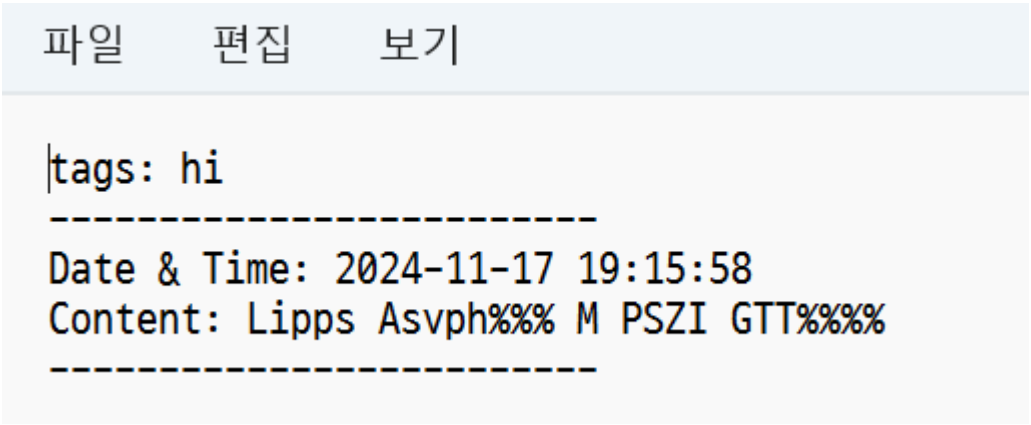
2) 테스트 결과

(1) 일기를 쓰는 기능

- 사용자가 새로운 일기를 작성하고 저장하게 한다.

```
----------------------------------------------------------------------------------------------------
1. Write a diary
2. Delete a diary
3. Edit a diary
4. Search a diary
5. Exit
----------------------------------------------------------------------------------------------------
HELLO admin. Have a nice day.
Select Number >> 1
WRITE : Hello World!!! I LOVE CPP!!!!
tag (whitespace-separated) :hi
Would you like to encrypt the diary content? Please answer y(yes) or n(no) : y
Diary entry saved successfully to 2024_diary/diary_2024-11-17.txt!
```

- 일기를 작성하면 현재 날짜와 시간이 자동으로 기록된다



- 일기마다 하나 이상의 태그를 추가하여 분류 및 검색을 용이하게 한다.

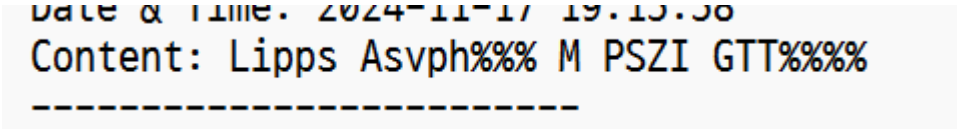
```
tag (whitespace-separated) :hi
```

```
tag (whitespace-separated) :test
```

- 일기 저장 시 암호화 여부를 선택할 수 있으며, 선택한 경우 일기의 내용이 암호화 되어 저장된다.

(암호화 선택)

```
Would you like to encrypt the diary content? Please answer y(yes) or n(no) : y
Diary entry saved successfully to 2024_diary/diary_2024-11-17.txt!
```



(암호화 선택안함)

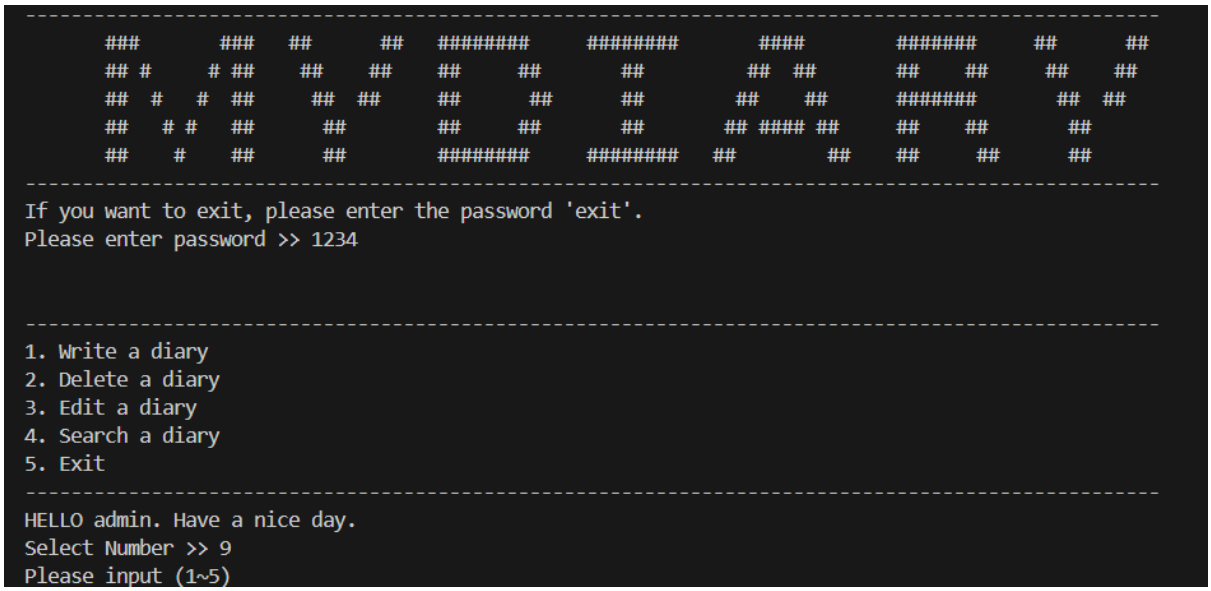
```
Would you like to encrypt the diary content? Please answer y(yes) or n(no) : n
Diary entry saved successfully to 2024_diary/diary_2024-11-17_1.txt!
```

```
Content: hihihihihihi!!!!@@@11112222aaaaaatest
--------------------------
```

(2) 로그인 기능

(로그인 시 비밀번호를 틀렸을 때)

(올바른 비밀번호를 입력했을 때)



(3) 일기를 삭제하는 기능

```
----------------------------------------------------------------------------------------------------
1. Write a diary
2. Delete a diary
3. Edit a diary
4. Search a diary
5. Exit
----------------------------------------------------------------------------------------------------
HELLO admin. Have a nice day.
Select Number >> 2
FILE LIST :
[1] : diary_2024-12-01.txt
[2] : diary_2024-12-01_1.txt
[3] : diary_2024-12-01_2.txt

Which files should be removed?
If you enter "No" nothing will be removed :diary_2024-12-01_2.txt
Are you sure you want to delete [2024_diary\diary_2024-12-01_2.txt] file? :
Please answer y(yes) or n(no) : n
```

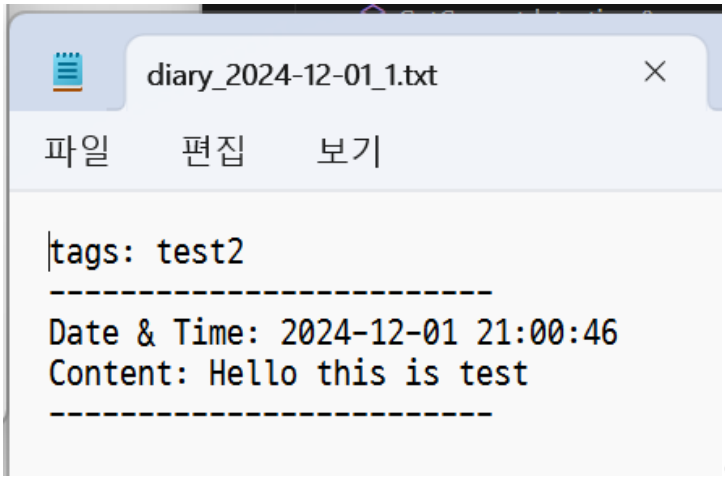
```
Select Number >> 2
FILE LIST :
[1] : diary_2024-12-01.txt
[2] : diary_2024-12-01_1.txt
[3] : diary_2024-12-01_2.txt

Which files should be removed?
If you enter "No" nothing will be removed :diary_2024-12-01_2.txt
Are you sure you want to delete [2024_diary\diary_2024-12-01_2.txt] file? :
Please answer y(yes) or n(no) : y
REMOVED!!!!
```

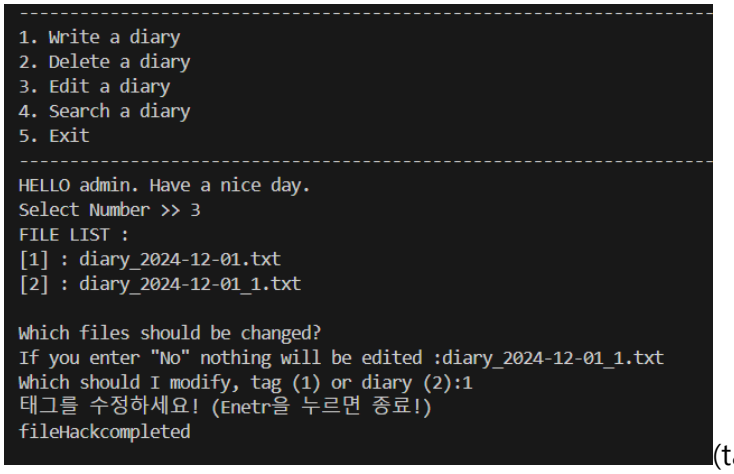
```
Select Number >> 2
FILE LIST :
[1] : diary_2024-12-01.txt
[2] : diary_2024-12-01_1.txt
```

지우겠냐는 확인 메시지의 답이 출력되고 답에 따라 지워지는 작동을 한다.

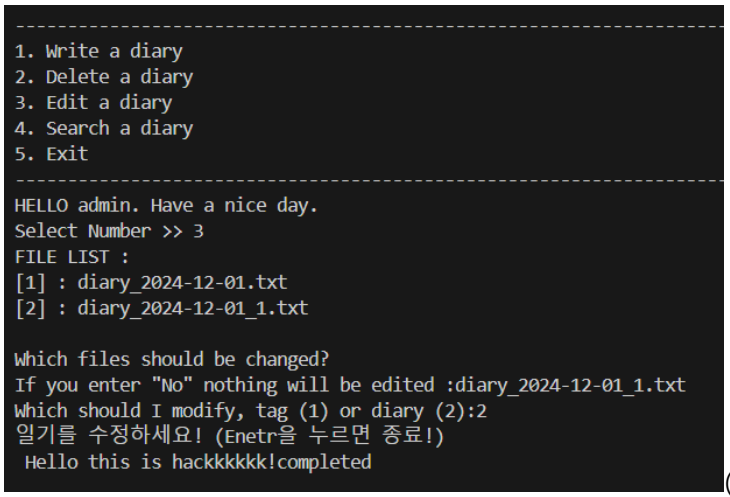
(3) 일기의 태그와 내용을 수정하는 기능



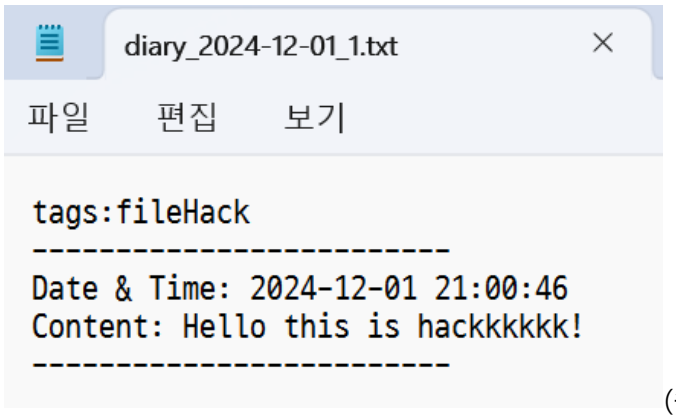
(초기 일기 내용)



(tag수정)



(file context 수정)



(수정된 파일 내용)

(4) 일기를 검색하는 기능

-날짜로 일기를 검색한다.

```
---------------------------------
1. Write a diary
2. Delete a diary
3. Edit a diary
4. Search a diary
5. Encryption OR Decryption
6. Exit
---------------------------------
HELLO admin. Have a nice day.
Select Number >> 4
Searching
complete
date or tag : date
Enter year:
```

-연도 만으로 검색

```
HELLO admin. Have a nice day.
Select Number >> 4
Searching
complete
date or tag : date
Enter year: 2024
Enter month (or -1 to skip): -1
Enter day (or -1 to skip): -1
Results:
diary_2024-10-16.txt
diary_2024-11-15.txt
diary_2024-12-06.txt
diary_2024-12-11.txt
diary_2024-12-11_1.txt
diary_2024-12-11_2.txt
diary_2024-12-13.txt
```

-연도, 월까지 검색

```
HELLO admin. Have a nice day.
Select Number >> 4
Searching
complete
date or tag : date
Enter year: 2024
Enter month (or -1 to skip): 12
Enter day (or -1 to skip): -1
Results:
diary_2024-12-06.txt
diary_2024-12-11.txt
diary_2024-12-11_1.txt
diary_2024-12-11_2.txt
diary_2024-12-13.txt
```

-연도, 월, 일 모두 검색

```
HELLO admin. Have a nice day.
Select Number >> 4
Searching
complete
date or tag : date
Enter year: 2024
Enter month (or -1 to skip): 12
Enter day (or -1 to skip): 11
Results:
diary_2024-12-11.txt
diary_2024-12-11_1.txt
diary_2024-12-11_2.txt
```

- 태그로 검색

```
HELLO admin. Have a nice day.
Select Number >> 4
Searching
complete
date or tag : tag
All tags and associated files:
Tag: bbb
 - 2024_diary\diary_2024-12-06.txt
Tag: test
 - 2024_diary\diary_2024-12-11.txt
 - 2024_diary\diary_2024-12-11_1.txt
 - 2024_diary\diary_2024-12-11_2.txt
Tag: test222
 - 2024_diary\diary_2024-10-16.txt
 - 2024_diary\diary_2024-11-15.txt
 - 2024_diary\diary_2024-12-13.txt
```

(5) 일기를 검색하는 기능

-암호화

```
tags: test222
-------------------------
Date & Time: 2024-12-13 16:38:59
Content: abcdefghijklmnopqrstuvwxyz0123456789!@#$
-------------------------
```

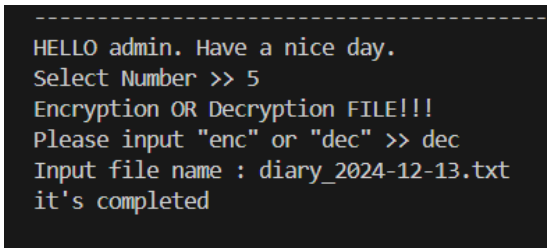
(원본 파일)

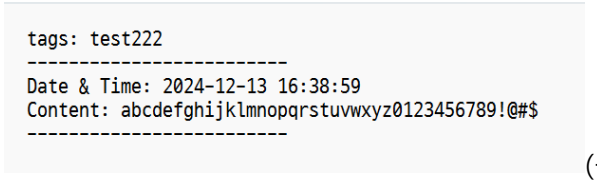
```
HELLO admin. Have a nice day.
Select Number >> 5
Encryption OR Decryption FILE!!!
Please input "enc" or "dec" >> enc
Input file name : diary_2024-12-13.txt
it's completed
```

```
tags: test222
-------------------------
Date & Time: 2024-12-13 16:38:59
Content: efghijklmnopqrstuvwxyzabcd4567890123%^&*
-------------------------
```

(암호화 적용후)

-복호화 (같은 파일을 이어서 적용)





(복호화 적용 후)

4. 계획 대비 변경 사항

1) 기능

5. 프로젝트 일정

업무	11/3	11/17	12/1	12/15	12/22
제안서 작성	완료				
기능1, 기능 6		완료			
기능 2, 기능 3			완료		
기능 4, 기능 5		완료			
최종보고서 작성					--------→