

C프로그래밍 및 실습

파일 정리 시스템

진척 보고서 #2

제출일자: 2023.12.10

제출자명: 이현승

제출자학번: 236179

1. 프로젝트 목표

1) 배경 및 필요성

회사나 공동으로 업무를 하는 공간에서 시간이 지날수록 개인 PC의 자료들은 많이 쌓일 수 밖에 없음. 이를 보통은 개인이 개인의 분류 기준에 의해서 파일들을 분류 해놓거나 정리를 전혀 하지 않기도 함. 그래서 타인의 PC에서 자료를 찾으려 할때 필요없는 시간이 걸리거나, PC의 주인의 도움이 필요함. 그리고 중요한 파일들을 정리를 해놓지 않아 파일 위치를 잊거나 파일을 실수로 지워버려서 시간이 지나 그 파일을 찾을때 시간이 걸리거나 찾지 못하는 경우가 생김. 그렇기 때문에 같은 기준으로 파일을 정리해야 할 필요성이 있지만, 파일을 정리할 인력, 시간이 필요하거나 그럼에도 불구하고 정리를 전혀 하지 않는 인원이 생길 수 있음. 이러한 이유들로 인해 파일을 자동으로 분류할 수 있는 프로그램이 필요함.

2) 프로젝트 목표

사용자가 정해진 규칙에 의해 파일을 자동으로 정리하고, 그 규칙들을 관리할 수 있도록 함. 또한 파일들이 자동으로 정리된 이후 그 히스토리를 시각화 하여 파일이 어떻게 정리 되었는지 사용자가 확인 할 수이 있게 하는것을 목표로 함.

3) 차별점

기존 프로그램들은 분류 정해진 기준에 의해 분류를 했음. 이 기준들이 변경될 수 있는 점을 고려해서 기준을 추가, 수정, 삭제 할 수있도록 기준관리 기능 추가 함. 그리고 추가로 파일이 이동, 삭제, 수정 등 정리가 된 이후의 히스토리를 파일에 저장하여 파일이 어떻게 정리가 되었는지 히스토리 파일에서 사용자가 한번에 알 수있도록 하기 때문에 기존의 프로그램과 차별점이 있음.

2. 기능 계획

1) 기능 1

- 설명 : 파일 목록을 수집, 정보 분석

(1) 세부 기능 1

- 설명 : 대상 디렉토리를 입력받아서 저장함. 그 디렉토리 내의 파일들을 순회하여 확장자, 생성일, 크기, 내용, 제목 등(변경 될 수 있음)을 추출함

2) 기능 2

- 설명 : 확장자, 생성일, 크기 등의 분류 기준을 설정하며, 조건문을 통해 처리

(1) 세부 기능 1

- 설명 : 분류 기준을 설정한 파일을 만들고, 그 파일에 접근해서 분류 기준을 가져옴.

(2) 세부 기능 2

- 설명 : 기준을 관리할 수 있는 메뉴를 만들어 메뉴에 접근하게 되면, 기준 추가, 삭제, 변경 등을 할 수 있게 함

3) 기능 3

- 설명 : 수집된 정보, 정해진 기준에 따라 파일을 분류하고, 이동, 반복

(1) 세부 기능 1

- 설명 : 설정한 기준에 따라 파일들을 분류하고, 각각의 파일을 적절한 디렉토리로 이동함.

(2) 세부 기능 2

- 설명 : 프로그램이 어떤 주기마다 실행 될 것인지 설정함. 시스템의 스케줄러, 프로그램 내에서 sleep 함수 사용.

4) 기능 4

- 설명 : 파일 정리 히스토리를 파일에 저장

(1) 세부 기능 1

- 설명 : 히스토리파일에 접근하여 정리가 완료된기록을 파일에 저장 ("file1.csv" 가 folder1 → folder2 로 이동 되었습니다. / "trash.txt" 가 "기준"에 의하여 folder1 → trash_folder1 로 이동 되었습니다.)

3. 진척사항

1) 기능 구현

(1) 규칙 추가 함수(add_rule)

- 입출력

입력 : RuleManager의 포인터(rule_manager), 문자열(rule_name: 규칙의 이름), 문자열(condition: 규칙의 조건)

출력 : 없음

- 설명 : RuleManager에 새로운 규칙을 추가하는 기능을 수행함. 규칙의 이름(rule_name)과 조건(condition)을 인자로 받아 RuleManager의 마지막 위치에 저장하고, 규칙의 개수를 하나 증가시킴. 문자열 복사는 보안이 강화된 strcpy_s 함수를 이용하며, 규칙의 개수를 증가시키는 것은 규칙 추가 후에 수행 됨.

- 적용된 배운 내용 : 구조체, 포인터, 문자열

- 코드 스크린샷

```
// 규칙 추가 함수
void add_rule(RuleManager* rule_manager, const char* rule_name, const char* condition) {
    // 복사
    strcpy_s(rule_manager->rules[rule_manager->count].rule_name, sizeof(rule_manager->rules[rule_manager->count].rule_name), rule_name);
    strcpy_s(rule_manager->rules[rule_manager->count].condition, sizeof(rule_manager->rules[rule_manager->count].condition), condition);
    rule_manager->count++;
}
```

(2) 규칙 삭제 함수(delete_rule)

- 입출력

입력 : RuleManager의 포인터(rule_manager), 문자열(rule_name: 삭제할 규칙의 이름)

출력 : 없음

- 설명 : RuleManager에서 특정 규칙을 삭제하는 기능을 수행함. 삭제할 규칙의 이름(rule_name)을 인자로 받아 해당 규칙을 찾고, 그 위치에서부터 마지막까지의 규칙을 한 칸씩 앞으로 이동시킴. 이후 규칙의 개수를 하나 감소. 만약 삭제할 규칙이 없다면 아무런 동작도 수행하지 않음.

- 적용된 배운 내용 : 구조체, 포인터, 문자열, 반복문, 조건문

- 코드 스크린샷

```
// 규칙 삭제 함수
void delete_rule(RuleManager* rule_manager, const char* rule_name) {
    int i;

    // 규칙 관리자의 모든 규칙 검사
    for(i=0; i<rule_manager->count; i++) {
        if(strcmp(rule_manager->rules[i].rule_name, rule_name) == 0) {
            for(int j=i; j<rule_manager->count-1; j++) {
                rule_manager->rules[j] = rule_manager->rules[j+1];
            }
            rule_manager->count--;
            break;
        }
    }
}
```

(3) 규칙 업데이트 함수(update_rule)

- 입출력

입력 : RuleManager의 포인터(rule_manager), 문자열(rule_name: 삭제할 규칙의 이름),

문자열(new_condition: 새로운 조건)

출력 : 없음

- 설명 : RuleManager에서 특정 규칙의 조건을 업데이트하는 기능을 수행함. 업데이트할 규칙의 이름(rule_name)과 새로운 조건(new_condition)을 인자로 받아 해당 규칙을 찾고, 그 규칙의 조건을 새로운 조건으로 변경함. 만약 업데이트할 규칙이 없다면 그대로 종료.

- 적용된 배운 내용 : 구조체, 포인터, 문자열, 반복문, 조건문

- 코드 스크린샷

```
// 업데이트 함수
void update_rule(RuleManager* rule_manager, const char* rule_name, const char* new_condition) {
    for(int i=0; i<rule_manager->count; i++) {
        // 업데이트 할 규칙과 새 규칙 이름 비교
        if(strcmp(rule_manager->rules[i].rule_name, rule_name) == 0) {
            strcpy_s(rule_manager->rules[i].condition, sizeof(rule_manager->rules[i].condition), new_condition);
            break;
        }
    }
}
```

(4) 파일로부터 규칙 읽는 함수(read_rules_from_file)

- 입출력

입력 : RuleManager의 포인터(rule_manager), 문자열(rule_name: 삭제할 규칙의 이름)

출력 : 없음

- 설명 : 파일에서 규칙을 읽어와 RuleManager에 저장하는 기능을 수행함. 파일의 이름(filename)을 인자로 받아 해당 파일을 열고, 규칙의 이름과 조건을 읽어 RuleManager에 추가함. 파일의 각 줄은 하나의 규칙을 나타내며, 규칙의 이름과 조건은 공백으로 구분됨. 만약 파일을 열 수 없다면 에러 메시지를 출력하고 함수를 종료.

- 적용된 배운 내용 : 구조체, 포인터, 문자열, 파일 입출력

- 코드 스크린샷

```
void read_rules_from_file(RuleManager* rule_manager, const char* filename) {
    FILE* file;
    char rule_name[MAX_RULE_NAME];
    char condition[MAX_CONDITION];

    if (fopen_s(&file, filename, "r") != 0) {
        fprintf(stderr, "파일을 열 수 없습니다.\n");
        return;
    }

    while (fscanf_s(file, "%s %s", rule_name, sizeof(rule_name), condition, sizeof(condition)) != EOF) {
        add_rule(rule_manager, rule_name, condition);
    }

    fclose(file);
}
```

2) 테스트 결과

(1) 테스트한 기능 이름

- 설명 : 디렉토리를 찾고있지 못해서, 디렉토리를 찾을 수 없습니다. 라는 경고가 나와야하는데 글자가 깨져서 다른 문구가 나옴.

디렉토리 찾는데 성공했을때 : 아직 구현이 제대로 되지 않아 같은 에러가 발생됨

- 테스트 결과 스크린샷

?뜻밖 應由??뵈뵈뵈뵈 ?뵈뵈뵈 ?뵈뵈뵈???뵈뵈뵈?뵈-뵈???뵈뵈뵈뵈뵈뵈뵈.
 C:\Users\user\Desktop\현승\23_1학기_학교\C 및 PY\C_프로젝트\Project_ws_231126_원도우용으로 변경\x64\Debug\Project_ws.exe
 (프로세스 24592개)가(C) 종료되었습니다(코드: 1개).
 디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
 하도록 설정합니다.
 이 창을 닫으려면 아무 키나 누르세요...

4. 프로젝트 일정

(진행한 작업과 진행 중인 작업 등을 표기)

업무		11/3	11/6~12	11/20~26	11/27~12/3
제안서 작성		완료			
기능1	세부기능1		완료		
기능2	세부기능1			진행중	
	세부기능2				진행중

업무		12/4~10	12/11~14	12/15~22
기능3	세부기능1	진행중			
	세부기능2		----->		
기능4	세부기능1			----->	