

Work_Log_2024_07_11

C언어 9일차 과제

작성자 : 박기홍

Remind

Remind

1. 로봇에 입단한 후 저는 군사용 로봇과 구조용 로봇을 개발하여 사람들의 안전과 생명을 지키는 데 기여하는 것을 목표로 임하겠습니다.
2. 지도교수님과 선배님들을 공경하며 동기들을 존중하는 마음가짐으로 임하겠습니다.
3. 로봇에서 배운 공학적 기술과 경험은 사람들을 살리는 데 사용될 수 있으며, 악한 마음을 품게 된다면 사람들에게 피해를 줄 수 있다는 것을 명심하면서 올바른 로봇 엔지니어로서 성장할 수 있도록 윤리를 중요시하겠습니다.
4. 글로벌 로봇 산업의 성장기와 성숙기를 이끌어 나아가는 엘리트로서 성장할 수 있는 능력을 만들겠습니다.

TODO List

Info

어떤 것을 할지 미리 생각해 놓는 시간입니다. 간략하게 2~5개로 적습니다.

 과제하기

Activity

Info

오늘 작업한 내용을 작성합니다.

- 개발을 하면서 고민한 부분
- 개발을 하면서 참고한 문서
- 새로 알게 된 사실 등의 내용

1. 과제하기

Start Time : 20:30

End Time : 2024.07.12. 19:50

Today's Completion

 Info

오늘 완료한 작업을 정리합니다.

Today's Concept

1. 파일 입출력 포맷 저장 불러오기


2. 전처리문

Solving a HomeWork

 Info

과제 풀이를 작성합니다.

HW_001

 과제 설명

출석부 프로그램 만들기.

- Student 구조체 : 번호, 이름, 주소(format : 나라,도,시,구), 성적
- 구조체 배열, 구조체 포인터를 이용한 함수 이용
- 모든 항목 예외처리(ex 숫자, 문자)
 - 필수 기능)
 1. 학생 정렬 기능 : 번호순, 이름순, 주소순(나라,도,시,구에 따라 따로 진행), 성적순 출력 모두 가능
 2. 학생 찾기 기능 : 번호or주소(나라,도,시,구에 따라 따로 진행)or성적을 입력하면 해당하는 모든 학생 이름 출력
 3. 학생 추가, 삭제 기능 : 번호, 이름, 주소, 성적 입력하면 출석부에서 해당 학생 추가/삭제. 중복된 경우 선택하여 삭제

4. 출석부 저장 및 불러오기 : 파일 입출력을 이용해 출석부를 저장하고 불러 오

[소스 코드]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

// 학생 구조체
typedef struct _Student {
    int number; // 번호
    char* name; // 이름
    char* adrCountry; // 국가
    char* adrDo; // 도
    char* adrSi; // 시
    char* adrGu; // 구
    int grade; // 성적
    struct _Student* next; // 다음 학생의 주소
}Student;

// 학생 리스트
typedef struct _StudentList {
    Student* head;
    Student* cur;
    Student* tail;
    Student* students[100]; // 학생 저장용.
    Student* duplicatedList[100]; // 중복된 학생의 index 저장용.
    int size;
}StudentList;

void Initializing_StudentList(StudentList* stdList); // StudentList 초기화 함수.
void Adding_Student(StudentList* stdList); // 학생 추가 함수.
void Deleting_Student(StudentList* stdList); // 학생 삭제 함수.
void Searching_Student(StudentList* stdList); // 학생 검색 함수.
void Sorting_Student(StudentList* stdList); // 학생 정렬 함수.
void Saving_File(StudentList* stdList); // 파일 저장하는 함수.
void Loading_File(StudentList* stdList); // 파일 불러오는 함수.

void Swapping_Value(Student* std1, Student* std2); // 학생 정렬 함수에서 사용하는
학생의 값들을 서로 변경하는 함수.

void Printing_StudentList(StudentList* stdList); // 전체 학생 출력 함수.
int Getting_IsitNumber(char* text, int cnt); // 입력된 값이 숫자인지 아닌지 판별하
는 함수.
int Changing_StringToInt(char* text); // 입력된 문자열 값을 int Type으로 return
```

하는 함수.

```
int main() {
    // StudentList 동적할당 하기.
    StudentList* studentList = (StudentList*)malloc(sizeof(StudentList));
    // 사용자의 명령어(입력값) 동적할당 하기.
    char* inputCommand = (char*)malloc(sizeof(char) * 20);

    Initializing_StudentList(studentList);
    if (studentList == NULL) {
        printf("[SYSTEM]출석부가 초기화 되지 않았습니다.");
    }else {
        printf("[SYSTEM]출석부가 초기화 되었습니다.");
    }

    while (1){
        printf("\n\n|-----|
\n");

        printf("|      [로봇 19기 수습단원 출석부 시스템]      |\n");
        printf("|-----|\n");
        printf("|              사용 가능한 명령어 모음              |\n");
        printf("|-----|\n");
        printf("|  1.add          2.delete          3.search  |\n");
        printf("|  4.sort          5.save           6.load   |\n");
        printf("|  7.exit                                     |\n");
        printf("|-----|
\n\n\n");

        printf("[SYSTEM]명령어를 입력하세요 : ");
        scanf("%s", inputCommand);

        if(strcmp(inputCommand, "add") == 0){
            Adding_Student(studentList);
        }else if (strcmp(inputCommand, "delete") == 0) {
            Deleting_Student(studentList);
        }else if (strcmp(inputCommand, "search") == 0) {
            Searching_Student(studentList);
        }else if (strcmp(inputCommand, "sort") == 0) {
            Sorting_Student(studentList);
        }else if (strcmp(inputCommand, "save") == 0) {
            Saving_File(studentList);
        }else if (strcmp(inputCommand, "load") == 0) {
            Loading_File(studentList);
        }else if (strcmp(inputCommand, "exit") == 0) {
```

```

        printf("[SYSTEM]프로그램을 종료합니다.");
        break;
    }else if (strcmp(inputCommand, "print") == 0) { // Debugging:
        Printing_StudentList(studentList);
    }

}

// 동적할당 해제.
/*free(studentList);*/

return 0;
}

void Initializing_StudentList(StudentList* stdList){
    stdList->size = 0;
    stdList->head = NULL;
    stdList->cur = NULL;
    stdList->tail = NULL;
}

int Getting_IsitNumber(char* text, int cnt) {
    for (int i = 0; i < cnt; i++) {
        if (text[i] < '0' || text[i] > '9') {
            return 0;
        }
    }
    return 1;
}

int Changing_StringToInt(char* text) {
    /* Student의 number와 grade를 할당하는 데 사용함.
    * 등급은 정수 형태의 등급. 1등급부터 ~ n등급까지 존재함. (1등급이 제일 낮은 것
    으로 하겠음)
    */
    int result = 0;
    for (int i = 0; text[i] != '\0'; i++) {
        result = result * 10 + (text[i] - '0');
        /*
        * ASCII CODE에서 0은 48임.
        * 입력된 문자 Type의 숫자에서 숫자의 기본인 '0'을 빼주면 해당 문자의
        숫자가 return 됨.
        * '1' - '0' = 49 - 48 = 1.
        */
    }
    return result;
}

```

```
}
```

```
void Adding_Student(StudentList* stdList) {
```

```
    char name[100] = { '\0', }, adrCountry[100] = { '\0', }, adrDo[100] =  
{ '\0', }, adrSi[100] = { '\0', }, adrGu[100] = { '\0', };
```

```
    Student* student = (Student*)malloc(sizeof(Student));
```

```
    char text[100];
```

```
    int cntNum = 0, cntGrade = 0;
```

```
    printf("[SYSTEM]추가할 학생의 번호를 입력하세요. : ");
```

```
    scanf("%s", text);
```

```
    for (int i = 0; text[cntNum] != '\0'; i++) {  
        cntNum++;
```

```
    }
```

```
    while (!Getting_IsitNumber(text, cntNum)) {
```

```
        printf("숫자만 입력하세요 : ");
```

```
        scanf("%s", text);
```

```
        cntNum = 0;
```

```
        for (int i = 0; text[cntNum] != '\0'; i++) {  
            cntNum++;
```

```
        }
```

```
    }
```

```
    student->number = Changing_StringToInt(text);
```

```
    printf("[SYSTEM]추가할 학생의 이름을 입력하세요. : ");
```

```
    scanf(" %[^\\n]s", &name);
```

```
    int cntName = 0;
```

```
    for (int i = 0; name[cntName] != '\0'; i++) {  
        cntName++;
```

```
    }
```

```
    student->name = (char*)malloc(cntName + 1); // 이름에 동적할당하기.
```

```
    for (int i = 0; i <= cntName; i++) { // 동적할당한 student의 name에 이곳  
name의 각각의 문자를 할당함.
```

```
        student->name[i] = name[i];
```

```
    }
```

```
    printf("[SYSTEM]추가할 학생의 거주하고 있는 국가를 입력하세요. : ");
```

```
    scanf(" %[^\\n]s", &adrCountry);
```

```
    int cntCountry = 0;
```

```
    for (int i = 0; adrCountry[cntCountry] != '\0'; i++) {  
        cntCountry++;
```

```
    }
```

```
    student->adrCountry = (char*)malloc(cntCountry + 1); // 국가에 동적할당하
```

기.

```
for (int i = 0; i <= cntCountry; i++) { // 동적할당한 student의
adrCountry에 이곳 adrCountry의 각각의 문자를 할당함.
    student->adrCountry[i] = adrCountry[i];
}

printf("[SYSTEM]추가할 학생의 거주하고 있는 도를 입력하세요. : ");
scanf(" %[^\\n]s", &adrDo);
int cntDo = 0;
for (int i = 0; adrDo[cntDo] != '\\0'; i++) {
    cntDo++;
}
student->adrDo = (char*)malloc(cntDo + 1); // 주소-도에 동적할당하기.
for (int i = 0; i <= cntDo; i++) { // 동적할당한 student의 adrDo에 이곳
adrDo의 각각의 문자를 할당함.
    student->adrDo[i] = adrDo[i];
}

printf("[SYSTEM]추가할 학생의 거주하고 있는 시를 입력하세요. : ");
scanf(" %[^\\n]s", &adrSi);
int cntSi = 0;
for (int i = 0; adrSi[cntSi] != '\\0'; i++) {
    cntSi++;
}
student->adrSi = (char*)malloc(cntSi + 1); // 주소-시에 동적할당하기.
for (int i = 0; i <= cntSi; i++) { // 동적할당한 student의 adrSi에 이곳
adrSi의 각각의 문자를 할당함.
    student->adrSi[i] = adrSi[i];
}

printf("[SYSTEM]추가할 학생의 거주하고 있는 구를 입력하세요. : ");
scanf(" %[^\\n]s", &adrGu);
int cntGu = 0;
for (int i = 0; adrGu[cntGu] != '\\0'; i++) {
    cntGu++;
}
student->adrGu = (char*)malloc(cntGu + 1); // 주소-구에 동적할당하기.
for (int i = 0; i <= cntGu; i++) { // 동적할당한 student의 adrGu에 이곳
adrGu의 각각의 문자를 할당함.
    student->adrGu[i] = adrGu[i];
}

printf("[SYSTEM]추가할 학생의 등급을 입력하세요. : ");
scanf("%s", text);
for (int i = 0; text[cntGrade] != '\\0'; i++) {
    cntGrade++;
}
```

```

    }
    while (!Getting_IsitNumber(text, cntGrade)) {
        printf("숫자만 입력하세요. : ");
        scanf("%s", text);
        cntGrade = 0;
        for (int i = 0; text[cntGrade] != '\0'; i++) {
            cntGrade++;
        }
    }
    student->grade = Changing_StringToInt(text);

    student->next = NULL;

    if (stdList->head == NULL) { // StudnetList에 Student가 없을 때만 실행 =>
첫 번째 Student 추가시만 실행.
        stdList->head = student;
        stdList->tail = student;
    }else{
        stdList->tail->next = student;
        stdList->tail = student;
    }

    stdList->students[stdList->size] = student;
    stdList->size++;
}

void Deleting_Student(StudentList* stdList) {

    if (stdList->head == NULL || stdList->size == 0) {
        printf("[SYSTEM]학생이 존재하지 않아 삭제할 수 없습니다.");
        return 0;
    }

    Printing_StudentList(stdList);

    int num, grade;
    char name[100] = { '\0', }, adrCountry[100] = { '\0', }, adrDo[100] =
{ '\0', }, adrSi[100] = { '\0', }, adrGu[100] = { '\0', };

    char text[100];
    int cntNum = 0, cntGrade = 0;
    printf("[SYSTEM]삭제를 희망하는 학생의 번호를 입력하세요. : ");
    scanf("%s", text);

    for (int i = 0; text[cntNum] != '\0'; i++) {

```



```

        cntNum++;
    }
    while (!Getting_IsitNumber(text, cntNum)) {
        printf("숫자만 입력하세요 : ");
        scanf("%s", text);
        cntNum = 0;
        for (int i = 0; text[cntNum] != '\0'; i++) {
            cntNum++;
        }
    }
    num = Changing_StringToInt(text);

    printf("[SYSTEM]삭제를 희망하는 학생의 이름을 입력하세요. : ");
    scanf(" %[^\\n]s", &name);

    printf("[SYSTEM]삭제를 희망하는 학생의 거주하고 있는 국가를 입력하세요. : ");
    scanf(" %[^\\n]s", &adrCountry);

    printf("[SYSTEM]삭제를 희망하는 학생의 거주하고 있는 도를 입력하세요. : ");
    scanf(" %[^\\n]s", &adrDo);

    printf("[SYSTEM]삭제를 희망하는 학생의 거주하고 있는 시를 입력하세요. : ");
    scanf(" %[^\\n]s", &adrSi);

    printf("[SYSTEM]삭제를 희망하는 학생의 거주하고 있는 구를 입력하세요. : ");
    scanf(" %[^\\n]s", &adrGu);

    printf("[SYSTEM]삭제를 희망하는 학생의 성적을 입력하세요 : ");
    scanf("%s", text);

    for (int i = 0; text[cntGrade] != '\0'; i++) {
        cntGrade++;
    }
    while (!Getting_IsitNumber(text, cntGrade)) {
        printf("숫자만 입력하세요 : ");
        scanf("%s", text);
        cntGrade = 0;
        for (int i = 0; text[cntGrade] != '\0'; i++) {
            cntGrade++;
        }
    }
    grade = Changing_StringToInt(text);

```

Student* current = stdList->head; // 전체 student를 돌아가면서 일치하는 삭제할 값이 있나 확인하는 용도.

Student* cursor = NULL; // 전체 student 리스트에서 현재 Student를 확인하는

용도.

`int duplicatedCnt = 0;` // StudentList에서 Student가 얼마나 중복되었는지 파악하기 위한 변수.

`int idxDList = 0;` // duplicatedList에 중복된 값을 넣기 위한 index. (i가 아닌 idxDList를 index로 사용하여 중복된 배열들의 0부터 ~ 마지막까지만 참조하게 구현함.)

```
for (int i = 0; i < stdList->size; i++) {
    // 삭제를 희망하는 학생이 학생 리스트에 있는지 확인하기.
    if (current->number == num && strcmp(current->name, name) == 0
    && strcmp(current->adrCountry, adrCountry) == 0 && strcmp(current->adrDo,
    adrDo) == 0 && strcmp(current->adrSi, adrSi) == 0 && strcmp(current->adrGu,
    adrGu) == 0 && current->grade == grade) {
        duplicatedCnt++;
        stdList->duplicatedList[idxDList] = current; // 조건을
만족한 student는 중복된 duplicatedList에 저장함.
        idxDList++;
    }
    current = current->next;
}
```

// 중복된 Student가 있는 경우 선택하여 삭제하기.

```
if (duplicatedCnt > 1) { // StudentList 내부에 중복된 Student가 있을 경우
    printf("중복된 학생의 index : ");
    for (int i = 0; i < duplicatedCnt; i++) {
        printf("%d ", i);
    }
    printf("\n삭제할 학생의 index를 입력하세요. : ");
    int deleteIdx;
    scanf("%d", &deleteIdx);
    current = stdList->duplicatedList[deleteIdx]; // 제거할 Student
의 위치 가리키기.
```

}else if(duplicatedCnt == 1){ // StudentList에서 Student가 중복되지 않은 상태일 때.

```
    current = stdList->duplicatedList[0];
}
else{
    printf("[SYSTEM]해당 학생을 찾을 수 없습니다.");
    return;
}
```

// Student 삭제하기.

```
if (current == stdList->head) { // 첫번째 학생인 경우.
    stdList->head = current->next;
    if (stdList->head == NULL) { // Student List가 비어있는 경우.
        stdList->tail = NULL;
    }
}
```

```

}else{ // 첫 번째 학생이 아닌 경우. (두 번째, 세 번째 ... n번째)
    cursor = stdList->head;
    while (cursor->next != current) {
        cursor = cursor->next;
    }
    cursor->next = current->next; // 가리키는 Student를 넘기기.

    if (current == stdList->tail) { // 삭제할 Student가 마지막
Student인 경우.
        stdList->tail = cursor;
    }
}

stdList->size--;
printf("[SYSTEM] 해당 학생이 삭제되었습니다.");

}

void Searching_Student(StudentList* stdList) {

    Student* student = (Student*)malloc(sizeof(Student));
    Student* current = stdList->head;

    if (stdList->head == NULL || stdList->size == 0) {
        printf("[SYSTEM] 현재 학생이 없으므로, 검색을 할 수 없습니다.");
        return 0;
    }

    printf("\n\n|-----|\n");
    printf("|      [로봇 19기 수습단원 출석부 시스템]      |\n");
    printf("|-----|\n");
    printf("|              검색 카테고리              |\n");
    printf("|-----|\n");
    printf("|  1.number          2.name          3.country  |\n");
    printf("|  4.do              5.si            6.gu       |\n");
    printf("|  7.grade                               |\n");
    printf("|-----|\n");

    char text[100];
    printf("[SYSTEM] 검색할 카테고리를 입력하세요. : ");
    scanf("%s", &text);

    if (strcmp(text, "number") == 0) {
        int num, cnt = 0;
        printf("[SYSTEM] 찾고 싶은 학생의 번호를 입력하세요. : ");

```

```

scanf("%d", &num);

// StudentList에 있는 Student 수 만큼 for문 돌리기.
for (int i = 0; i < stdList->size; i++) {
    if (current->number == num) { // StudentList 내부에 있는
Student의 배열에서 Student의 number가 num과 같다면 출력하기.
        printf("[출석부-%d번째] <번호 : %d>, 이름 : %s,
국가 : %s, 도 : %s, 시 : %s, 구 : %s, 등급 : %d.\n", i, current->number,
current->name, current->adrCountry, current->adrDo, current->adrSi, current-
>adrGu, current->grade);

        cnt++;
    }
    current = current->next; // 다음 Student를 참조하기.
}
if (cnt == 0) { // Exception handling: 해당 번호를 가진 Student가
없을 때.
    printf("[SYSTEM]해당 번호를 가진 학생은 없습니다.");
}
}
else if (strcmp(text, "name") == 0) {
    int cnt = 0;
    char targetText[100];
    printf("[SYSTEM]찾고 싶은 학생의 이름을 입력하세요. : ");
    scanf("%s", &targetText);

    // StudentList에 있는 Student 수 만큼 for문 돌리기.
    for (int i = 0; i < stdList->size; i++) {
        if (strcmp(targetText, current->name) == 0) {
            printf("[출석부-%d번째] 번호 : %d, <이름 : %s>,
국가 : %s, 도 : %s, 시 : %s, 구 : %s, 등급 : %d.\n", i, current->number,
current->name, current->adrCountry, current->adrDo, current->adrSi, current-
>adrGu, current->grade);

            cnt++;
        }
        current = current->next;
    }
    if (cnt == 0) {
        printf("[SYSTEM]해당 이름을 가진 학생은 없습니다.");
    }
}
else if (strcmp(text, "country") == 0) {
    int cnt = 0;
    char targetText[100];
    printf("[SYSTEM]찾고 싶은 학생의 출신 국가를 입력하세요. : ");
    scanf("%s", &targetText);

    // StudentList에 있는 Student 수 만큼 for문 돌리기.
    for (int i = 0; i < stdList->size; i++) {

```

```

        if (strcmp(targetText, current->adrCountry) == 0) {
            printf("[출석부-%d번째] 번호 : %d, 이름 : %s, <
국가 : %s>, 도 : %s, 시 : %s, 구 : %s, 등급 : %d.\n", i, current->number,
current->name, current->adrCountry, current->adrDo, current->adrSi, current-
>adrGu, current->grade);

            cnt++;
        }
        current = current->next;
    }
    if (cnt == 0) {
        printf("[SYSTEM]해당 국가의 출신인 학생은 없습니다.");
    }
}
else if (strcmp(text, "do") == 0) {
    int cnt = 0;
    char targetText[100];
    printf("[SYSTEM]찾고 싶은 학생이 거주하고 있는 도를 입력하세요. :
");

    scanf("%s", &targetText);

    // StudentList에 있는 Student 수 만큼 for문 돌리기.
    for (int i = 0; i < stdList->size; i++) {
        if (strcmp(targetText, current->adrDo) == 0) {
            printf("[출석부-%d번째] 번호 : %d, 이름 : %s, 국
가 : %s, <도 : %s>, 시 : %s, 구 : %s, 등급 : %d.\n", i, current->number,
current->name, current->adrCountry, current->adrDo, current->adrSi, current-
>adrGu, current->grade);

            cnt++;
        }
        current = current->next;
    }
    if (cnt == 0) {
        printf("[SYSTEM]해당 지역에 거주하고 있는 학생은 없습니
다.");
    }
}
else if (strcmp(text, "si") == 0) {
    int cnt = 0;
    char targetText[100];
    printf("[SYSTEM]찾고 싶은 학생이 거주하고 있는 시를 입력하세요. :
");

    scanf("%s", &targetText);

    // StudentList에 있는 Student 수 만큼 for문 돌리기.
    for (int i = 0; i < stdList->size; i++) {
        if (strcmp(targetText, current->adrSi) == 0) {
            printf("[출석부-%d번째] 번호 : %d, 이름 : %s, 국
가 : %s, 도 : %s, <시 : %s>, 구 : %s, 등급 : %d.\n", i, current->number,

```

```

current->name, current->adrCountry, current->adrDo, current->adrSi, current->
>adrGu, current->grade);

        cnt++;
    }
    current = current->next;
}
if (cnt == 0) {
    printf("[SYSTEM] 해당 지역에 거주하고 있는 학생은 없습니
다.");
}
}
else if (strcmp(text, "gu") == 0) {
    int cnt = 0;
    char targetText[100];
    printf("[SYSTEM] 찾고 싶은 학생이 거주하고 있는 구를 입력하세요. :
");

    scanf("%s", &targetText);

    // StudentList에 있는 Student 수 만큼 for문 돌리기.
    for (int i = 0; i < stdList->size; i++) {
        if (strcmp(targetText, current->adrGu) == 0) {
            printf("[출석부-%d번째] 번호 : %d, 이름 : %s, 국
가 : %s, 도 : %s, 시 : %s, <구 : %s>, 등급 : %d.\n", i, current->number,
current->name, current->adrCountry, current->adrDo, current->adrSi, current->
>adrGu, current->grade);

            cnt++;
        }
        current = current->next;
    }
    if (cnt == 0) {
        printf("[SYSTEM] 해당 지역에 거주하고 있는 학생은 없습니
다.");
    }
}
else if (strcmp(text, "grade") == 0) {
    int targetGrade, cnt = 0;
    printf("[SYSTEM] 찾고 싶은 학생의 등급을 입력하세요. : ");
    scanf("%d", &targetGrade);

    // StudentList에 있는 Student 수 만큼 for문 돌리기.
    for (int i = 0; i < stdList->size; i++) {
        if (targetGrade == current->grade) {
            printf("[출석부-%d번째] 번호 : %d, 이름 : %s, 국
가 : %s, 도 : %s, 시 : %s, 구 : %s, <등급 : %d>.\n", i, current->number,
current->name, current->adrCountry, current->adrDo, current->adrSi, current->
>adrGu, current->grade);

            cnt++;
        }
    }
}

```

```

        current = current->next;
    }
    if (cnt == 0) {
        printf("[SYSTEM] 해당 등급의 학생은 없습니다.");
    }
}

}

void Sorting_Student(StudentList* stdList) {

    Student* student = (Student*)malloc(sizeof(Student));
    Student* current = stdList->head;

    if (stdList->head == NULL || stdList->size == 0) {
        printf("[SYSTEM] 현재 학생이 없으므로, 정렬을 할 수 없습니다.");
        return 0;
    }

    printf("\n\n|-----|\n");
    printf("|      [로봇 19기 수습단원 출석부 시스템]      |\n");
    printf("|-----|\n");
    printf("|                        정렬 카테고리                        |\n");
    printf("|-----|\n");
    printf("|  1.number          2.name          3.country  |\n");
    printf("|  4.do              5.si            6.gu       |\n");
    printf("|  7.grade                               |\n");
    printf("|-----|\n");

    char text[100];
    printf("[SYSTEM] 정렬할 카테고리를 입력하세요. : ");
    scanf("%s", &text);

    // 번호 기준 오름차순으로 정렬하기.
    if (strcmp(text, "number") == 0) {
        // StudentList에 있는 Student 수 만큼 for문 돌리기.
        for (int i = 0; i < stdList->size - 1; i++) {
            for (int j = 0; j < stdList->size - 1 - i; j++) {
                if (stdList->students[j]->number > stdList->students[j + 1]->number) {
                    Swapping_Value(stdList->students[j],
stdList->students[j + 1]);
                }
            }
        }
    }
}

```

```

    }else if (strcmp(text, "name") == 0) { // 이름 기준 오름차순으로 정렬하기.
        // StudentList에 있는 Student 수 만큼 for문 돌리기.
        for (int i = 0; i < stdList->size - 1; i++) {
            for (int j = 0; j < stdList->size - 1 - i; j++) {
                /*
                 * strcmp return value.
                 * value < 0 : string1이 string2보다 작음. -> 내
                 * value == 0 : string1과 string2가 같음.
                 * value > 0 : string1이 string2보다 큼. -> 오름
                 * e.x)
                 * 'c', 'a', 'b' 입력.
                 * 1. 'c'와 'a' 비교 -> value > 0
                 * 'a', 'c', 'b'. (변경 전의 케이스는 주석에서 생략
                 * 2. 'c'와 'b' 비교 -> value > 0
                 * 'a', 'b', 'c'. -> 오름차순 정렬 완료.
                 */
                if (strcmp(stdList->students[j]->name,
stdList->students[j + 1]->name) > 0) {
                    Swapping_Value(stdList->students[j],
stdList->students[j + 1]);
                }
            }
        }

    }else if (strcmp(text, "country") == 0) {
        // StudentList에 있는 Student 수 만큼 for문 돌리기.
        for (int i = 0; i < stdList->size - 1; i++) {
            for (int j = 0; j < stdList->size - 1 - i; j++) {
                if (strcmp(stdList->students[j]->adrCountry,
stdList->students[j + 1]->adrCountry) > 0) {
                    Swapping_Value(stdList->students[j],
stdList->students[j + 1]);
                }
            }
        }

    }else if (strcmp(text, "do") == 0) {
        // StudentList에 있는 Student 수 만큼 for문 돌리기.
        for (int i = 0; i < stdList->size - 1; i++) {
            for (int j = 0; j < stdList->size - 1 - i; j++) {
                if (strcmp(stdList->students[j]->adrDo,
stdList->students[j + 1]->adrDo) > 0) {

```



```

                                Swapping_Value(stdList->students[j],
stdList->students[j + 1]);
                                }
                                }
                                }

    }else if (strcmp(text, "si") == 0) {
        // StudentList에 있는 Student 수 만큼 for문 돌리기.
        for (int i = 0; i < stdList->size - 1; i++) {
            for (int j = 0; j < stdList->size - 1 - i; j++) {
                if (strcmp(stdList->students[j]->adrSi,
stdList->students[j + 1]->adrSi) > 0) {
                                Swapping_Value(stdList->students[j],
stdList->students[j + 1]);
                                }
                            }
                        }
                    }

    }else if (strcmp(text, "gu") == 0) {
        // StudentList에 있는 Student 수 만큼 for문 돌리기.
        for (int i = 0; i < stdList->size - 1; i++) {
            for (int j = 0; j < stdList->size - 1 - i; j++) {
                if (strcmp(stdList->students[j]->adrGu,
stdList->students[j + 1]->adrGu) > 0) {
                                Swapping_Value(stdList->students[j],
stdList->students[j + 1]);
                                }
                            }
                        }
                    }

    }else if (strcmp(text, "grade") == 0) {
        // StudentList에 있는 Student 수 만큼 for문 돌리기.
        for (int i = 0; i < stdList->size - 1; i++) {
            for (int j = 0; j < stdList->size - 1 - i; j++) {
                if (stdList->students[j]->grade > stdList-
>students[j + 1]->grade) {
                                Swapping_Value(stdList->students[j],
stdList->students[j + 1]);
                                }
                            }
                        }
                    }

    }

    printf("[SYSTEM]정렬이 완료되었습니다.");
}

```

```

void Swapping_Value(Student* std1, Student* std2) {

    // number 변경하기.
    int tempNum = std1->number;
    std1->number = std2->number;
    std2->number = tempNum;

    // name 변경하기.
    char* tempName = std1->name;
    std1->name = std2->name;
    std2->name = tempName;

    // counrtry 변경하기.
    char* tempCountry = std1->adrCountry;
    std1->adrCountry = std2->adrCountry;
    std2->adrCountry = tempCountry;

    // do 변경하기.
    char* tempDo = std1->adrDo;
    std1->adrDo = std2->adrDo;
    std2->adrDo = tempDo;

    // si 변경하기.
    char* tempSi = std1->adrSi;
    std1->adrSi = std2->adrSi;
    std2->adrSi = tempSi;

    // gu 변경하기.
    char* tempGu = std1->adrGu;
    std1->adrGu = std2->adrGu;
    std2->adrGu = tempGu;

    // grade 변경하기.
    char* tempGrade = std1->grade;
    std1->grade = std2->grade;
    std2->grade = tempGrade;
}

```

```

void Saving_File(StudentList* stdList) {

    FILE* studentFile;
    studentFile = fopen("StudentList.txt", "w");
    printf("%d", stdList->size);
}

```

```

    if (stdList->size == 0) {
        printf("[SYSTEM]학생이 추가되지 않아, 빈 파일이 저장되었습니다.\n");
        // StudentList 사이즈 저장하기.
        fprintf(studentFile, "StudentList Size : %d\n", stdList->size); // Load할 때 사용할 것.
        fclose(studentFile);
        return;
    }

    // StudentList 사이즈 저장하기.
    fprintf(studentFile, "StudentList Size : %d\n", stdList->size);

    // StudentList에 있는 Student의 개수 만큼 저장하기.
    for (int i = 0; i < stdList->size; i++) {
        Student* student = stdList->students[i];
        fprintf(studentFile, "출석부-%d\n번째\n번호 : %d\n이름 : %s\n국가 : %s\n도 : %s\n시 : %s\n구 : %s\n등급 : %d\n", i, student->number, student->name, student->adrCountry, student->adrDo, student->adrSi, student->adrGu, student->grade);
    }

    fclose(studentFile);
    printf("[SYSTEM]파일이 정상적으로 저장되었습니다!");
}

void Loading_File(StudentList* stdList) {

    FILE* studentFile;
    studentFile = fopen("StudentList.txt", "r");

    if (studentFile == NULL) { // 파일이 없을 경우.
        printf("[SYSTEM]파일이 존재하지 않습니다.\n");
        return;
    }

    Initializing_StudentList(stdList);

    fscanf(studentFile, "StudentList Size : %d\n", &stdList->size);

    if (stdList->size == 0) {
        printf("[SYSTEM]읽어들일 수 있는 파일이 없습니다.\n");
        fclose(studentFile);
        return;
    }
}

```

```

// StudentList에 있는 Student의 개수 만큼 불러오기.
for (int i = 0; i < stdList->size; i++) {
    // 파일에서 불러온 값들 동적할당하기.
    Student* student = (Student*)malloc(sizeof(Student));

    student->name = (char*)malloc(100 * sizeof(char));
    student->adrCountry = (char*)malloc(100 * sizeof(char));
    student->adrDo = (char*)malloc(100 * sizeof(char));
    student->adrSi = (char*)malloc(100 * sizeof(char));
    student->adrGu = (char*)malloc(100 * sizeof(char));
    // 저장 형식을 준수하여 값을 불러옴.
    fscanf(studentFile, "출석부-%d\n번째\n번호 : %d\n이름 : %s\n국가
: %s\n도 : %s\n시 : %s\n구 : %s\n등급 : %d\n", &i, &student->number, student-
>name, student->adrCountry, student->adrDo, student->adrSi, student->adrGu,
&student->grade);
    printf("출석부-%d번째, 번호 : %d, 이름 : %s, 국가 : %s, 도 : %s,
시 : %s, 구 : %s, 등급 : %d\n\n\n", i, student->number, student->name, student-
>adrCountry, student->adrDo, student->adrSi, student->adrGu, student->grade);

    student->next = NULL;

    if (stdList->head == NULL) { // StudnetList에 Student가 없을 때
만 실행 => 첫 번째 Student 추가시만 실행.
        stdList->head = student;
        stdList->tail = student;
    }else{
        stdList->tail->next = student;
        stdList->tail = student;
    }

    stdList->students[i] = student;
}

fclose(studentFile);
printf("[SYSTEM]파일을 정상적으로 불러왔습니다!");
}

void Printing_StudentList(StudentList* stdList) {

    Student* current = stdList->head;

```

```

        if (stdList->head == NULL || stdList->size == 0) {
            printf("[SYSTEM]현재 학생이 없습니다.");
            return;
        }

        printf("\n\n|-----|\n");
        printf("|          [로봇 19기 수습단원 출석부]          |\n");
        printf("|-----|\n");
        printf("|          현재 인원 명단          |\n");
        printf("|-----|\n");
        for (int i = 0; i < stdList->size; i++) {
            printf("[출석부-%d번째] 번호 : %d, 이름 : %s, 국가 : %s, 도 : %s,
시 : %s, 구 : %s, 등급 : %d.\n", i, current->number, current->name, current-
>adrCountry, current->adrDo, current->adrSi, current->adrGu, current->grade);
            current = current->next;
        }
        printf("|-----|\n\n\n");
    }
}

```

[실행 결과]

Test Case#1

```

C:\Users\wordk\git\ub\ROBIT_intem_KiHongPark_HW_repo\Projects\WC_Language\HomeWork\HW_20240711\Wx64\Debug\HW_20240711.exe
[SYSTEM]출석부가 초기화 되었습니다.

[로봇 19기 수습단원 출석부 시스템]
사용 가능한 명령어 모음
1.add      2.delete    3.search
4.sort     5.save     6.load
7.exit

[SYSTEM]명령어를 입력하세요 : add
[SYSTEM]추가할 학생의 번호 : 1
[SYSTEM]추가할 학생의 이름 : hong2
[SYSTEM]추가할 학생의 국가 : korea
[SYSTEM]추가할 학생의 도 : aa
[SYSTEM]추가할 학생의 시 : aa
[SYSTEM]추가할 학생의 구 : bb
[SYSTEM]추가할 학생의 등급 : 1
[SYSTEM]명령어를 입력하세요 : print

[로봇 19기 수습단원 출석부]
현재 인원 명단
출석부-(번호) 번호 : 1, 이름 : hong2, 국가 : korea, 도 : aa, 시 : aa, 구 : bb, 등급 : 1.

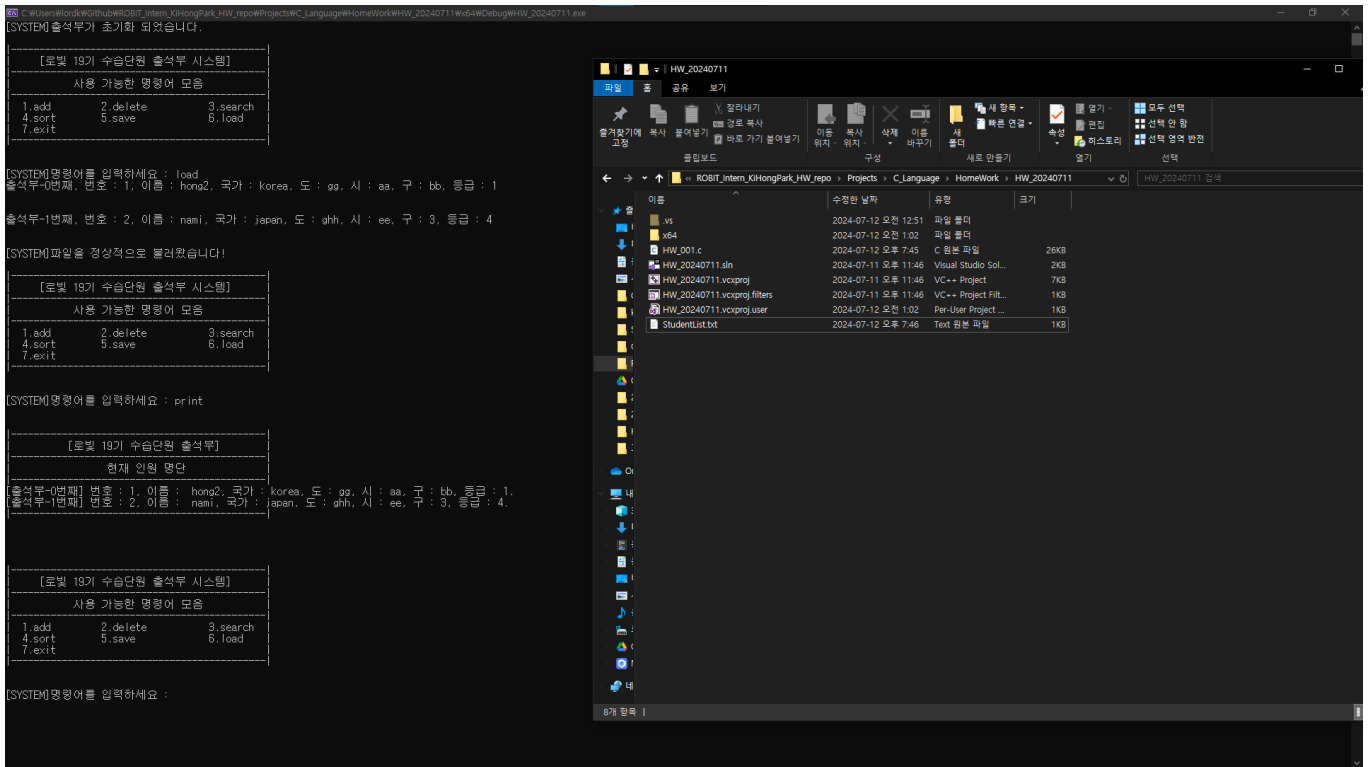
[로봇 19기 수습단원 출석부 시스템]
사용 가능한 명령어 모음
1.add      2.delete    3.search
4.sort     5.save     6.load
7.exit

[SYSTEM]명령어를 입력하세요 : add
[SYSTEM]추가할 학생의 번호 : 2
[SYSTEM]추가할 학생의 이름 : japan
[SYSTEM]추가할 학생의 국가 : japan
[SYSTEM]추가할 학생의 도 : shh
[SYSTEM]추가할 학생의 시 : shh
[SYSTEM]추가할 학생의 구 : shh
[SYSTEM]추가할 학생의 등급 : 1
[SYSTEM]명령어를 입력하세요 :

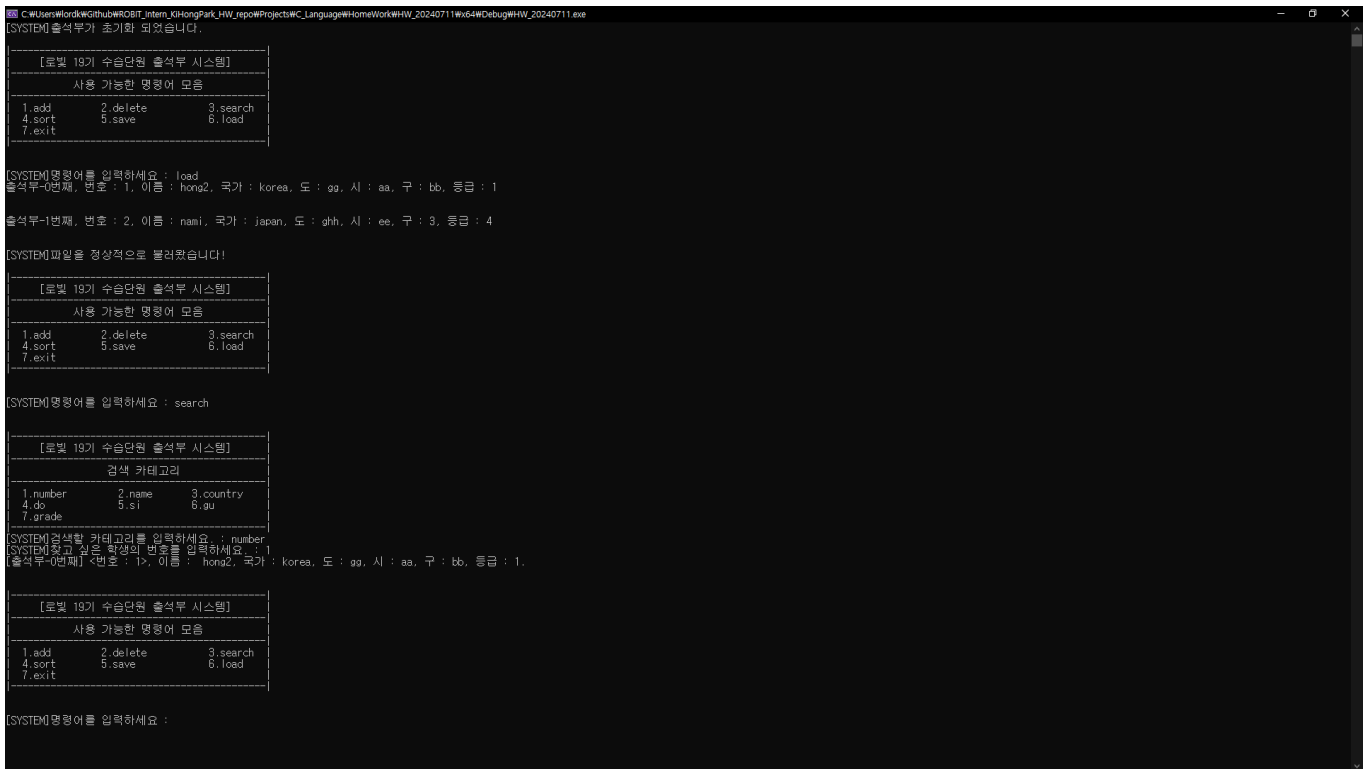
[로봇 19기 수습단원 출석부 시스템]
사용 가능한 명령어 모음
1.add      2.delete    3.search
4.sort     5.save     6.load
7.exit

```

Test Case#2



Test Case#3



Ideas & Important Information

Info

아이디어 및 중요 정보를 작성합니다.

Memo

Info

작업 중 기타 내용을 메모합니다.

Review

Info

하루 작업에 대한 피로도, 기분 등을 평가합니다.

Feelings : (😞)

Fatigue : (5)

Summary : ~~~