231630 김도형

프로젝트 목적 : 7주차까지 배운 내용에 대한 실습

목표 : TODO리스트 만들기

요구사항 : 할 일을 삭제 및 출력, 할 일을 수정을 했을 경우, 인덱스와 할 일을 입력받고, 해당 인덱스수정, 할 일이 다 찬 경우에는 할 일이 다 찼다고 출력하고 프로그램 종료.
할 일 추가 기능, 할 일 삭제 기능, 할 일 목록 출력 기능 함수화.

1.설계 및 구현

```
int main() {
    char tasks[MAX_TASKS][CHAR_NUM] = { "" }; // 할 일 목록을 저장하기 위한 2차원
    int taskCount = 0; // 할 일의 수를 저장하기 위한 변수

    printf("TODO 리스트 시작! ₩n");
```

Int main 함수 > 문자형 tasks(2차원 배열) 변수와 taskcount 변수 선언

```
    while (1) {
        int choice = -1; // 사용자 입력 메뉴를 저장하기 위한 변수

        // 사용자에게 메뉴를 출력하고, 메뉴를 입력받기
        printf("-----------------₩n");
        printf("메뉴를 입력해주세요.₩n");;
        printf("1. 할 일 추가₩n2. 할 일 삭제₩n3. 목록 보기₩n4. 종료₩n5.할 일 수정
        printf("현재 할 일 수 = %d₩n", taskCount);
        printf("-----------------₩n");
        scanf_s("%d", &choice);

        int terminate = 0; // 종료를 위한 flag
        int delIndex = -1;  // 할 일 삭제를 위한 index 저장 변수
        int changeIndex = -1; // 할 일 수정을 위한 index 저장 변수
        char ch; // 할 일 수정시 버퍼를 받기 위한 문자 변수
```

While 반복문 사용 : choice 변수 설정 > 사용자에게 메뉴를 입력하기 위함

종료, 삭제, 수정을 위한 각각의 변수 설정

```
// 입력에 따른 기능 수행
switch (choice) {
case 1:
    printf("할 일을 입력하세요 (공백 없이 입력하세요): ");
    scanf_s("%s", tasks[taskCount], (int)sizeof(tasks[taskCount]));
    printf("할 일 ""%s""가 저장되었습니다₩n₩n", tasks[taskCount]);
    taskCount++;
    break;
case 2:
    // 할 일 삭제하는 코드 블록
    printf("삭제할 할 일의 번호를 입력해주세요. (1부터 시작):");
    scanf_s("%d", &delIndex);
    if (delIndex > taskCount || delIndex <= 0) {
        printf("삭제 범위가 벗어났습니다.₩n");
    }
    else {
        printf("%d. %s : 할 일을 삭제합니다.₩n", delIndex, tasks[delInd

        // 배열간 대입 (=배열에 문자 배열인 문자열의 대입) 이 불가능하기
        // 문자열 복사 함수로 삭제
        strcpy_s(tasks[delIndex - 1], sizeof(tasks[delIndex - 1]), "");

        // 특정 인덱스의 할 일 삭제 후 뒤에 있는 할 일 앞으로 옮기기
        for (int i = delIndex; i < taskCount + 1; i++) {
            strcpy_s(tasks[i - 1], sizeof(tasks[i]), tasks[i]);
        }
        taskCount -= 1;
    }
    break;
case 3:
    printf("할 일 목록₩n");
    for (int i = 0; i < taskCount; i++) {
        printf("%d. %s ₩n", i + 1, tasks[i]);
    }
```

Switch 조건문을 이용해 사용자에게 입력받은 choice에 따라 할 일을 입력받는다

Case1 : 1 을 입력받으면 할 일을 입력하라고 하고, 입력받으면 저장되었다는 문구를 출력한다.

할 일을 입력받을 때 마다 taskcount가 하나씩 올라간다

Case2 : 2를 입력받으면, 삭제할 일 의 번호를 입력받고, 만약, 입력 받은 번호가 전체 taskcount보다 크다면

범위가 벗어났다는 문구를 출력. 그렇지 않은 나머지 경우에는 입력받은 일을 삭제한다고 출력.

Case3 : 3을 입력받으면, for 반복문을 통해, i가 taskcount보다 작을 때 까지 1씩 늘려가며 task를 출력한다.

결과적으로 할 일 목록을 보여주고, i+1번째의 할 일들을 task[i] 배열에서 가져온다.

```
        break;
    case 4:
        terminate = 1;
        break;
    default:
        printf("잘못된 선택입니다. 다시 선택하세요.₩n");
    }

    if (terminate == 1) {
        printf("종료를 선택하셨습니다. 프로그램을 종료합니다.₩n");
        break;
    }
```

Case4 : 용자에게 4를 입력 받음, 만약 종료 변수 terminate가 1 이면, 종료한다는 문구를 출력

default를 사용하여, 1234 외에 나머지를 입력받으면 잘못된 선택이라는 문구 출력 후

다시 선택하라는 문구 출력.

Todo_manager.c

———

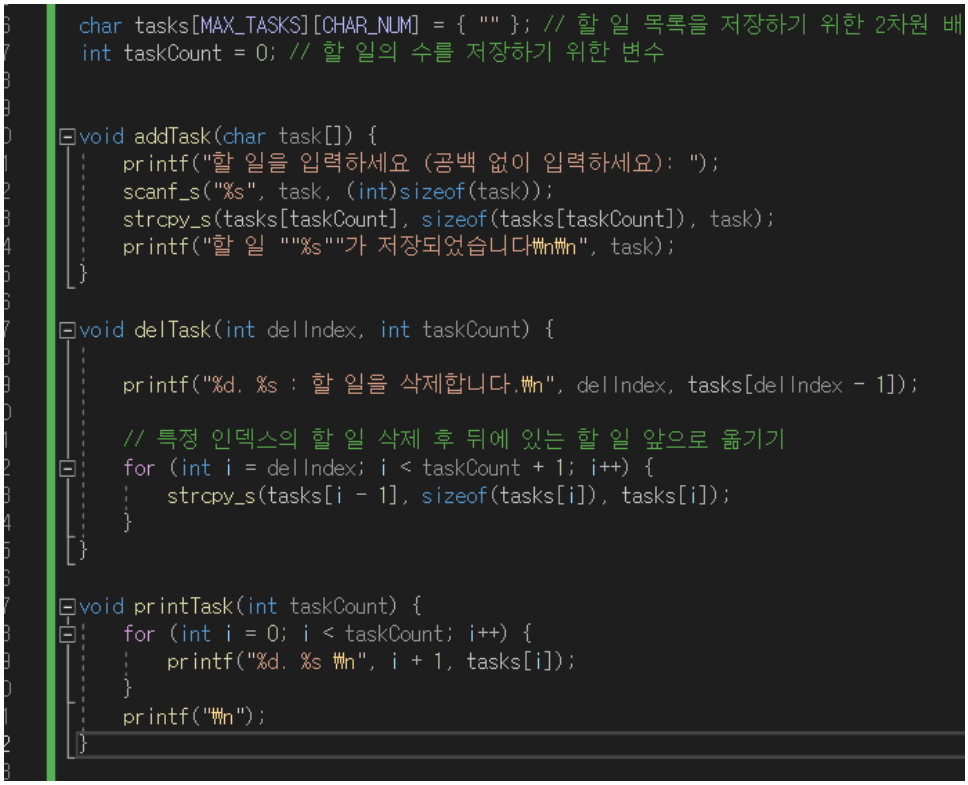
Todo_manager.func.c

```
    case 5:
        printf("수정할 할 일의 번호를 입력해주세요. (1부터 시작): ");
        scanf_s("%d", &modifyIndex);
        ch = getchar();
        printf("새로운 할 일을 입력해주세요");
        scanf_s("%s", tasks[modifyIndex - 1], (int)sizeof(tasks[modifyIndex -
        printf("새로운 할 일이 추가되었습니다: %d. %s₩n", modifyIndex, tasks[
        break;
    default:
        printf("잘못된 선택입니다. 다시 선택하세요.₩n");
    }

    if (terminate == 1) {
        printf("종료를 선택하셨습니다. 프로그램을 종료합니다.₩n");
        break;
    }
    if (taskCount == 10) {
        printf("할 일이 %d개로 다 찼습니다.", taskCount);
        break;
    }
}
return 0;
```

Case5 : 수정할 일의 번호를 입력받고, modifyindex에 저장 후 출력.

새로운 할 일을 입력하고,

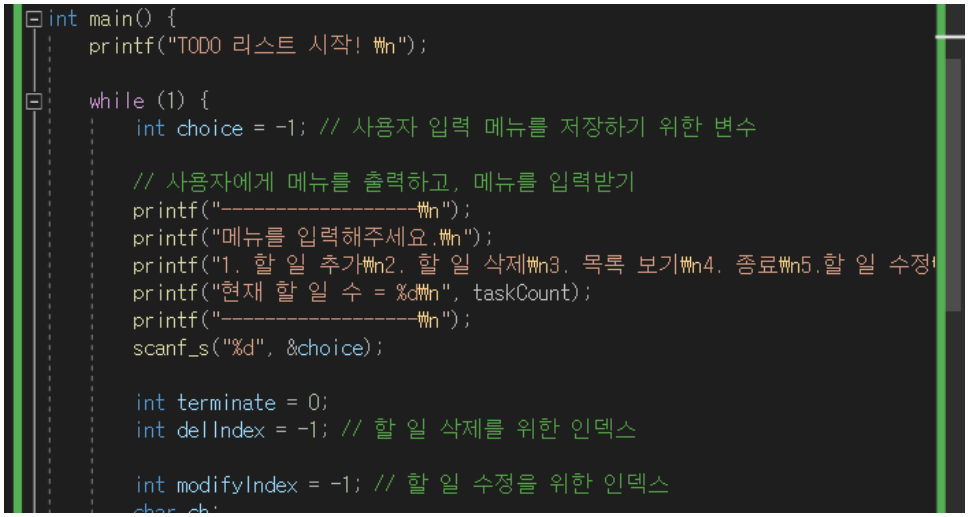
If를 사용하여, terminate가 1이면 프로그램을 종료하고, 할 일 10개가 다 차면(taskcount 가 10이 되면) 할 일이 10개로 다 찼다는 문구를 출력.



```
char tasks[MAX_TASKS][CHAR_NUM] = { "" }; // 할 일 목록을 저장하기 위한 2차원 배
int taskCount = 0; // 할 일의 수를 저장하기 위한 변수


void addTask(char task[]) {
    printf("할 일을 입력하세요 (공백 없이 입력하세요): ");
    scanf_s("%s", task, (int)sizeof(task));
    strcpy_s(tasks[taskCount], sizeof(tasks[taskCount]), task);
    printf("할 일 ""%s""가 저장되었습니다₩n₩n", task);
}

void delTask(int delIndex, int taskCount) {

    printf("%d. %s : 할 일을 삭제합니다.₩n", delIndex, tasks[delIndex - 1]);

    // 특정 인덱스의 할 일 삭제 후 뒤에 있는 할 일 앞으로 옮기기
    for (int i = delIndex; i < taskCount + 1; i++) {
        strcpy_s(tasks[i - 1], sizeof(tasks[i]), tasks[i]);
    }
}

void printTask(int taskCount) {
    for (int i = 0; i < taskCount; i++) {
        printf("%d. %s ₩n", i + 1, tasks[i]);
    }
    printf("₩n");
}
```

함수 선언 : void함수를 사용하여, addtask(할 일 추가), deltask(할 일 삭제), printtask(할 일 목록 출력) 선언.

Addtask에서는 할 일을 입력받고, task[]에 저장. Deltask에서는 삭제할 일을 입력받고 delindex에 저장 후, taskcount에서 하나 뺌. printtask에서는 for반복문을 통해서 taskcount전까지의 할 일을 출력함(할 일 목록 = case3에 해당). Void 함수이기 때문에 반환값 없음 = return 0;



```c
int main() {
    printf("TODO 리스트 시작! ₩n");

    while (1) {
        int choice = -1; // 사용자 입력 메뉴를 저장하기 위한 변수

        // 사용자에게 메뉴를 출력하고, 메뉴를 입력받기
        printf("-----------------₩n");
        printf("메뉴를 입력해주세요.₩n");
        printf("1. 할 일 추가₩n2. 할 일 삭제₩n3. 목록 보기₩n4. 종료₩n5.할 일 수정
        printf("현재 할 일 수 = %d₩n", taskCount);
        printf("-----------------₩n");
        scanf_s("%d", &choice);

        int terminate = 0;
        int delIndex = -1; // 할 일 삭제를 위한 인덱스

        int modifyIndex = -1; // 할 일 수정을 위한 인덱스
        char ch;
```

```
// 입력에 따른 기능 수행
switch (choice) {
case 1:
    addTask(tasks[taskCount]);
    taskCount++;
    break;
case 2:
    printf("삭제할 할 일의 번호를 입력해주세요. (1부터 시작):");
    scanf_s("%d", &delIndex);
    if (delIndex > taskCount || delIndex <= 0) {
        printf("삭제 범위가 벗어났습니다.₩n");
    }
    else {
        delTask(delIndex, taskCount);
        taskCount -= 1;
    }
    break;
case 3:
    printf("할 일 목록₩n");
    printTask(taskCount);
    break;
case 4:
    terminate = 1;
    break;
case 5:
    printf("수정할 할 일의 번호를 입력해주세요. (1부터 시작): ");
    scanf_s("%d", &modifyIndex);
    ch = getchar();
    printf("새로운 할 일을 입력해주세요");
    scanf_s("%s", tasks[modifyIndex - 1], (int)sizeof(tasks[modifyIndex -
    printf("새로운 할 일이 추가되었습니다: %d. %s₩n", modifyIndex, tasks[
    break;
default:
    printf("잘못된 선택입니다. 다시 선택하세요.₩n");
```

위에서 선언한 3개의 함수를 int main()함수에서 호출. Switch 조건문을 활용하여 case를 나누고 12345가 아닌 경우에는 잘못된 선택이라고 출력. 문자열은 sizeof를 사용하여 반환.

If조건문을 사용하여 지우려는 delindex가 taskcount보다 크다면, 삭제 범위가 벗어났다는 문구 출력

그렇지 않으면 삭제하고 taskcount에서 1을 뺌.

4.테스트

```
TODO 리스트 시작!
------------------
메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 0
------------------
1
할 일을 입력하세요 (공백 없이 입력하세요): 김도형
할 일 김도형가 저장되었습니다

------------------
메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 1
------------------
1
할 일을 입력하세요 (공백 없이 입력하세요): 231630
할 일 231630가 저장되었습니다
```

메뉴 보여주기, 이름과 학번을 입력했을 때 저장 후, 할 일 수 2로 증가 확인.

```
4. 종료
5.할 일 수정
현재 할 일 수 = 1
------------------
1
할 일을 입력하세요 (공백 없이 입력하세요): 231630
할 일 231630가 저장되었습니다

------------------
메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 2
------------------
2
삭제할 할 일의 번호를 입력해주세요. (1부터 시작):1
1. 김도형 : 할 일을 삭제합니다.
------------------
메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 1
------------------
```

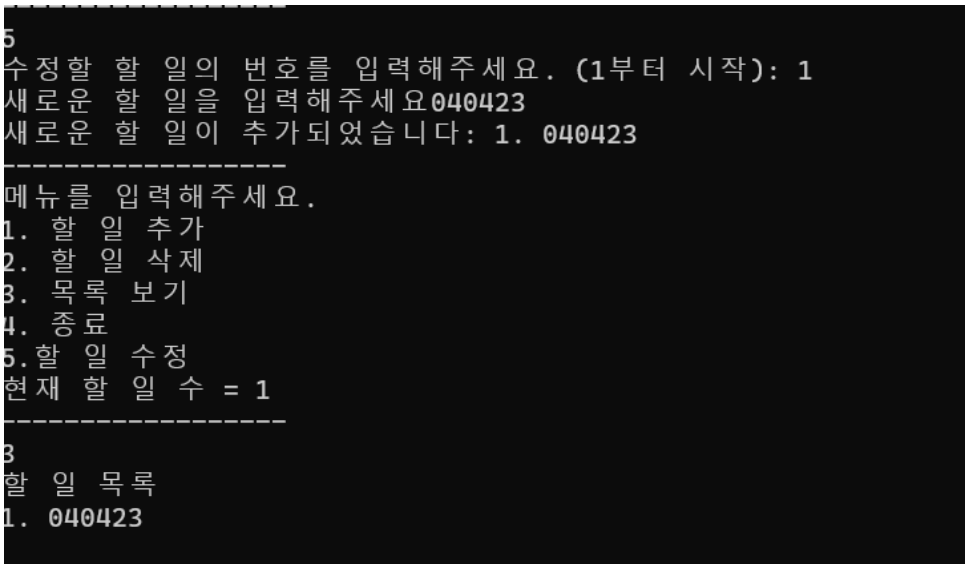
2를 입력 하고 1을 입력했을 때 첫 번째 입력한 '김도형'이 삭제 되어서 할 일이 1개가 됨

```
3
할 일 목록
1. 231630
```

첫 번째에 입력 한 것이 삭제되고, 3을 입력했을 때 출력되는 값인 목록에는학번만 남음.

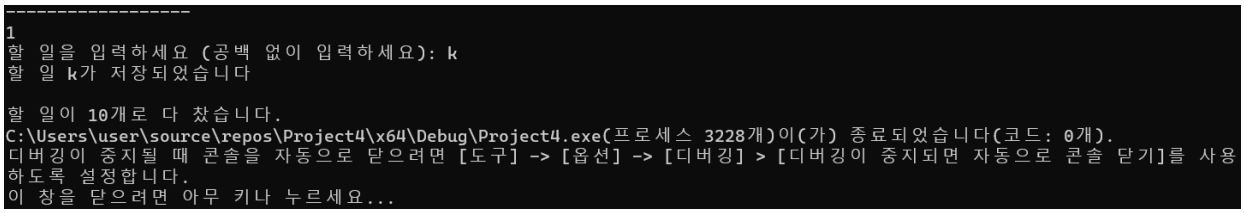
```
4
종료를 선택하셨습니다. 프로그램을 종료합니다.

C:\Users\user\source\repos\Project4\x64\Debug\Project4.exe(프로세스 16960개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

4를 입력하면 프로그램을 종료한다는 메시지 출력



5를 입력하여 할 일 수정. 231630을 040423으로 수정.

3을 입력하여 목록을 확인하면 수정되어있음.



할 일 이 10개가 다 차면, 다 찼다는 문구 출력 후 종료.

5.결과 및 결론

프로젝트 결과 : todo프로그램을 코딩하고, 할 일 수정 후 추가/삭제하기, 할 일 다 차면 종료하기,

코드 함수화하기 등을 수행하였다.

느낀점 : 처음 프로젝트를 받았을 때 너무 막막하고 힘들었지만, 이 프로젝트를 통해 내가 c언어를 배우면서 부족했던 점과 노력해야 할 점을 알게되었고, 함수 부분이 프로젝트를 하기전에는 그저

막막했는데 프로젝트에 주석도 달아보고, 보고서를 쓰는 과정을 통해 함수에 대해 조금 더 깨닫게 되는 계기가 된 거 같다. 틀이 있어서 비교적 쉽게 도전해볼 수 있었지만, 항상 틀이 있을 수 없기에

틀이 없어도 코딩을 할 수 있을만큼 노력해야겠다고 생각했다.