할 일 관리 프로그램 보고서

학과	물리학과
학번	185132
이름	송재호
제출일	2023, 10. 27

1. 서론

1. 프로젝트 목적 및 배경 - 1주차~ 7주차까지 배운내용(반복 조건 배열등) 활용한다.

- 1주차~7주차까지 배운 내용을 활용하는 코드를 작성하는게 목표이다.

- 반복문, 조건문, 함수, 배열, 문자열, 다차원 배열등을 배웠다.

2. 목표: TODO 리스트 만들기

- 2차원 배열을 기본으로 하는 TODO 리스트 작성을 목표로 한다.

- 배운 내용인 조건문과, 반복문, 함수, 배열, 문자열 등을 사용한다.

2. 요구사항

1. 사용자 요구사항: 사용자가 할 일을 배열에 입력, 삭제, 출력, 수정 할 수 있는 프로그램

2. 기능 요구사항(입력, 삭제, 출력, 수정, 종료, 자동 종료)

    - 사용자의 할 일을 입력받아 배열에 저장하는 기능을 만들어 준다.

    - 입력 받은 할 일을 삭제해야하는 일이 있을 수 있으므로 할 일 삭제 기능을 추가한다.

    - 입력 받은 할 일을 수정해야하는 일이 있을 수 있으므로 할 일 수정 기능을 추가한다.

    - 입력 받은 할 일들을 한번에 확인하기 위한 할 일 목록 출력 기능을 추가한다.

    - 입력하고자 하는 할 일이없거다 다 입력하였을시 프로그램 종료 기능을 추가한다.

    - 할 일은 10개까지만 저장하며 10개를 달성시 자동 종료 기능을 추가한다.

3. 설계 및 구현

1. 기능 별 구현 사항: (요구사항 별 함수 설계- 1. 입력, 2. 삭제, 3. 목록 출력, 4. 종료, 5. 수정)

```c
#include <stdio.h>
#define MAX_TASKS 10
#define CHAR_NUM 100
#include <string.h>

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

1. 코드블록/함수스크린샷(좌)
2. 입력(블록/함수에 입력되는 변수, 값들과 설명)
   - MAX_TASKS: 배열 세로 최대치에 필요한 변수
   - CHAR_MAX: 배열 가로 최대치에 필요한 변수
   - tasks[MAX_TASKS][CHAR_MAX]= {" "}
     할 일 목록을 저장하기 위한 2차원 배열
   -int taskCount = 0 할 일을 세어주는 변수
3. 반환값(함수 경우 작성)
   - 함수가 아니므로 없음
4. 결과(블록/함수가 종료된 결과)
   - task[10][100] 배열 생성
   - taskCount = 0 변수 생성
5. 설명(코드내 작동순서, 내용 추가 설명)

기본적으로 사용하는 전처리기를 입력한다.

사용할 변수를 define 으로 정의해준다.

할 일을 저장하려는 배열을 만들어준다.

할 일의 수를 저장하기 위한 변수를 정의해준다.

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

1. 코드블록/함수스크린샷(좌)
2. 입력(블록/함수에 입력되는 변수, 값들과 설명)
   - scanf_s로 변수 choice 입력 – 메뉴 선택
   - taskCount :저장된 할 일수를 저장한 변수
3. 반환값(함수 경우 작성)
   - 함수가 아니므로 없음
4. 결과(블록/함수가 종료된 결과)
   - 메뉴 출력
   - taskCount(할 일 수) 출력
   - 입력받은 기능 코드 실행
5. 설명(코드내 작동순서, 내용 추가 설명)

사용자에게 시작 메뉴를 출력하여 보여준다.

어떤 기능을 사용할지 입력 받는다.

입력 받은 기능의 코드를 실행한다.

```
case 1:
    printf("할 일을 입력하세요 (공백 없이 입력하세요): ");
    scanf_s("%s", tasks[taskCount], (int)sizeof(tasks[taskCount]));
    // 할 일을 입력받음
    printf("할 일 ""%s""가 저장되었습니다₩n₩n", tasks[taskCount]);
    taskCount++;
    //할 일 추가이므로 할 일 개수 1 증가
    break;
```

1. 코드블록/함수스크린샷(좌)
2. 입력(블록/함수에 입력되는 변수, 값들과 설명)
   - task[taskCount] :배열의 가장 위쪽의 공백
   - taskCount :입력된 할 일수를 저장한 변수
3. 반환값(함수 경우 작성)
   - 함수가 아니므로 없음
4. 결과(블록/함수가 종료된 결과)
   - 입력받은 할 일을 저장
   - 할 일을 추가 하였으므로 taskCount가 1증가
5. 설명(코드내 작동순서, 내용 추가 설명)

사용자에게 할 일을 입력 받아 배열의 가장 앞쪽에 입력해준다.

할 일을 입력 받으면, 할 일의 개수를 세어준다. (taskCount에 1을 더해준다.)

```
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[delIndex - 1]);

        // 배열간 대입 (=배열에 문자 배열인 문자열의 대입) 이 불가능하기 때문에
        // 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;
        // 할 일을 삭제 하였으니까 할 일 개수 삭제
```

1. 코드블록/함수스크린샷(좌)
2. 입력(블록/함수에 입력되는 변수, 값들과 설명)
    - delIndex :삭제하려는 배열의 행 +1 번호
    - task[delIndex - 1] : 삭제하려는 배열의 행
    - strcpy_s : 복사 함수
3. 반환값(함수 경우 작성)
    - 함수가 아니므로 없음
4. 결과(블록/함수가 종료된 결과)
    - 입력 받은 할 일 삭제
    - taskCount 에서 1 감소
5. 설명(코드내 작동순서, 내용 추가 설명)

- 삭제하고자 하는 배열의 인덱스+1값을 입력 받아준다.

- 잘못된 값을 입력 받으면 "삭제 범위가 벗어났습니다." 라는 안내문구를 출력해준다.

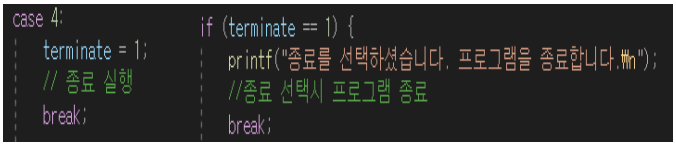
- 복사 함수(strcpy)를 통해 할 일을 삭제 해준다.

- 반복문을 사용하여 뒤에 있는 할 일을 앞으로 이동시켜준다.

```c
case 3:
    printf("할 일 목록₩n");
    for (int i = 0; i < taskCount; i++) {
        printf("%d. %s ₩n", i + 1, tasks[i]);
    }
    printf("₩n");
    break;
```

1. 코드블록/함수스크린샷(좌)
2. 입력(블록/함수에 입력되는 변수, 값들과 설명)
   - taskCount : 입력된 할 일수를 저장한 변수
   - i 반복에서 사용하는 변수
3. 반환값(함수 경우 작성)
   - 함수가 아니므로 없음
4. 결과(블록/함수가 종료된 결과)
   - 할 일 을 전부 출력
5. 설명(코드내 작동순서, 내용 추가 설명)

- 할 일의 목록을 배열과 반복문을 통해 출력해준다.

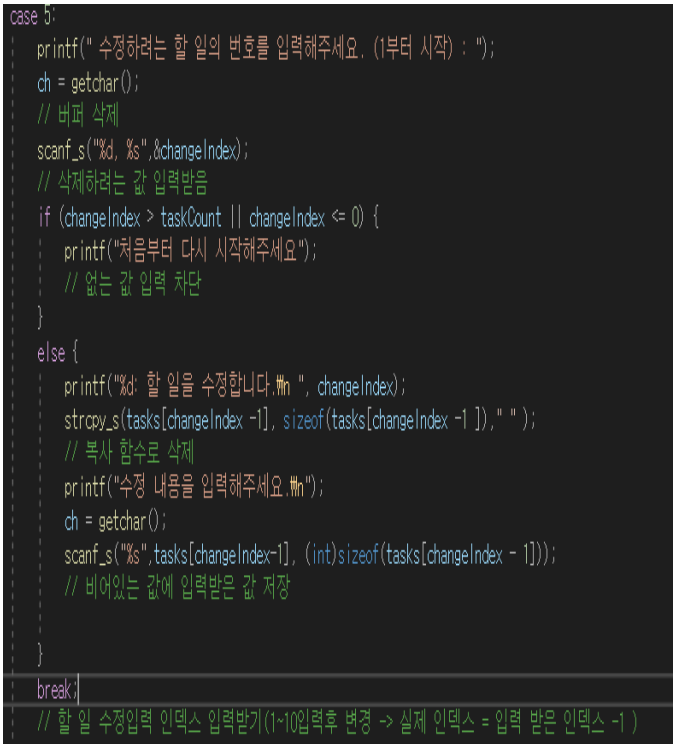


```
case 4:
    terminate = 1;
    // 종료 실행
    break;
```

```
if (terminate == 1) {
    printf("종료를 선택하셨습니다. 프로그램을 종료합니다.₩n");
    //종료 선택시 프로그램 종료
    break;
```

1. 코드블록/함수스크린샷(좌)
2. 입력(블록/함수에 입력되는 변수, 값들과 설명)
   - terminate : 종료 기능을 실행하기 위한 변수
3. 반환값(함수 경우 작성)
   - 함수가 아니므로 없음
4. 결과(블록/함수가 종료된 결과)
   - 프로그램 종료
5. 설명(코드내 작동순서, 내용 추가 설명)

-종료 기능을 선택하면 조건문을 사용하여 프로그램을 종료한다.



```
case 5:
    printf(" 수정하려는 할 일의 번호를 입력해주세요. (1부터 시작) : ");
    ch = getchar();
    // 버퍼 삭제
    scanf_s("%d, %s",&changeIndex);
    // 삭제하려는 값 입력받음
    if (changeIndex > taskCount || changeIndex <= 0) {
        printf("처음부터 다시 시작해주세요");
        // 없는 값 입력 차단
    }
    else {
        printf("%d: 할 일을 수정합니다.₩n ", changeIndex);
        strcpy_s(tasks[changeIndex -1], sizeof(tasks[changeIndex -1 ])," " );
        // 복사 함수로 삭제
        printf("수정 내용을 입력해주세요.₩n");
        ch = getchar();
        scanf_s("%s",tasks[changeIndex-1], (int)sizeof(tasks[changeIndex - 1]));
        // 비어있는 값에 입력받은 값 저장


    }
    break;
    // 할 일 수정입력 인덱스 입력받기(1~10입력후 변경 -> 실제 인덱스 = 입력 받은 인덱스 -1 )
```

1. 코드블록/함수스크린샷(좌)
2. 입력(블록/함수에 입력되는 변수, 값들과 설명)
    - changeIndex : 수정하려는 배열의 행번호 -1
3. 반환값(함수 경우 작성)
    - 함수가 아니므로 없음
4. 결과(블록/함수가 종료된 결과)
    - 입력 받은 번호의 할 일을 입력 받은 할 일로 수정한다.
5. 설명(코드내 작동순서, 내용 추가 설명)

할 일 수정을 선택하면 2번의 삭제기능을 응용하여 삭제후 재입력 방식으로 수정한다.

수정하려는 배열의 인덱스 행을 복사 함수로 지워준다. (비어 있는 배열은 공백을 유지하기때문에 삭제기능에서는 한칸씩 땡겨 왔지만 수정기능에서는 비어있는 공간에 입력하기 위해 그대로 놔줌)

비어있는 공백부분에 재 입력받은 할 일을 저장한다.

```c
if (taskCount == 10) {
    printf("할일이 다 찼습니다. 프로그램을 종료합니다.₩n");
    //할 일의 개수 10달성시 프로그램 종료
    break;
```

1. 코드블록/함수스크린샷(좌)
2. 입력(블록/함수에 입력되는 변수, 값들과 설명)
    - 새롭게 입력된 변수 없음
3. 반환값(함수 경우 작성)
    - 함수가 아니므로 없음
4. 결과(블록/함수가 종료된 결과)
    - 프로그램 종료
5. 설명(코드내 작동순서, 내용 추가 설명)

- 할 일의 개수가 10개를 달성하면 자동으로 프로그램을 종료한다.

4. 테스트

1. 기능별(함수별) 정상 작동 확인 여부

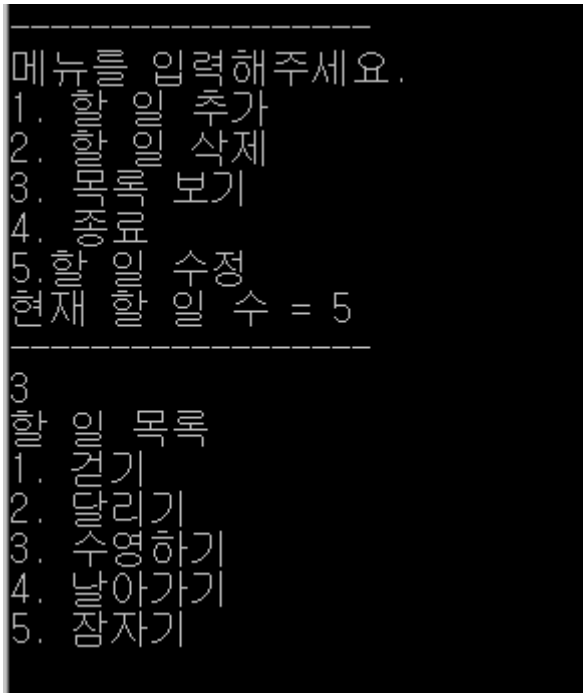
```
TODO 리스트 시작!
------------------
메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 0
------------------
1
할 일을 입력하세요 (공백 없이 입력하세요): 걷기
할 일 걷기가 저장되었습니다

------------------
메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 1
------------------
```

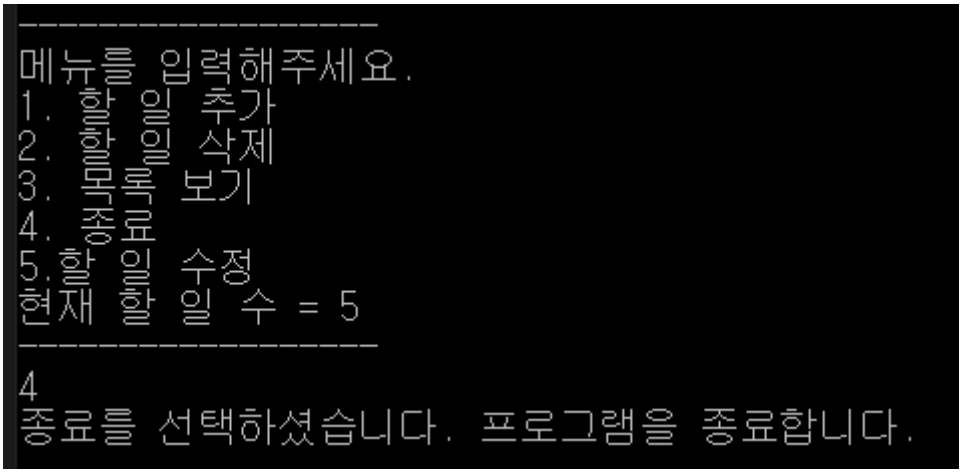
1. 입력

```
------------------
메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 1
------------------
2
삭제할 할 일의 번호를 입력해주세요. (1부터 시작):1
1. 걷기 : 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 = 0
------------------
```

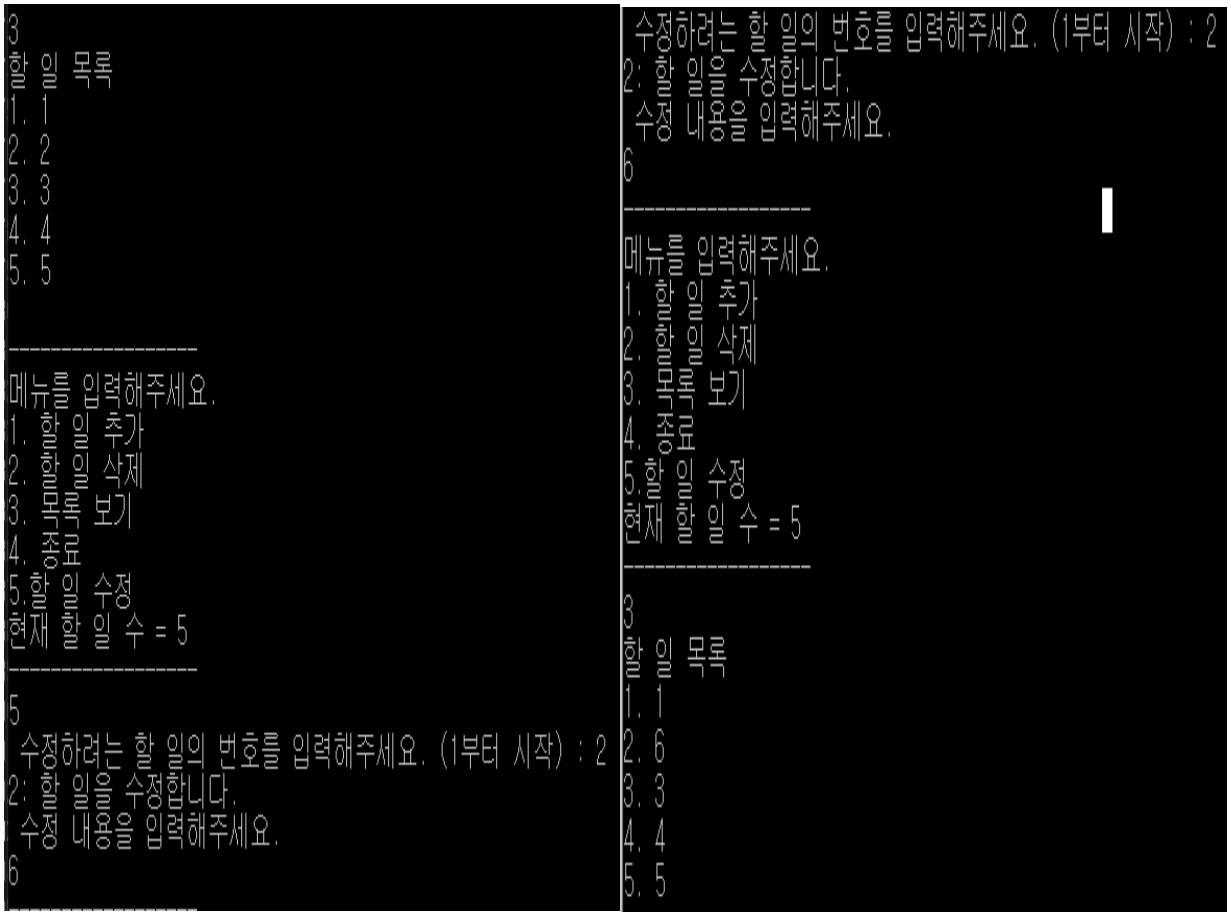
2. 삭제


메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 5
3
할 일 목록
1. 걷기
2. 달리기
3. 수영하기
4. 날아가기
5. 잠자기

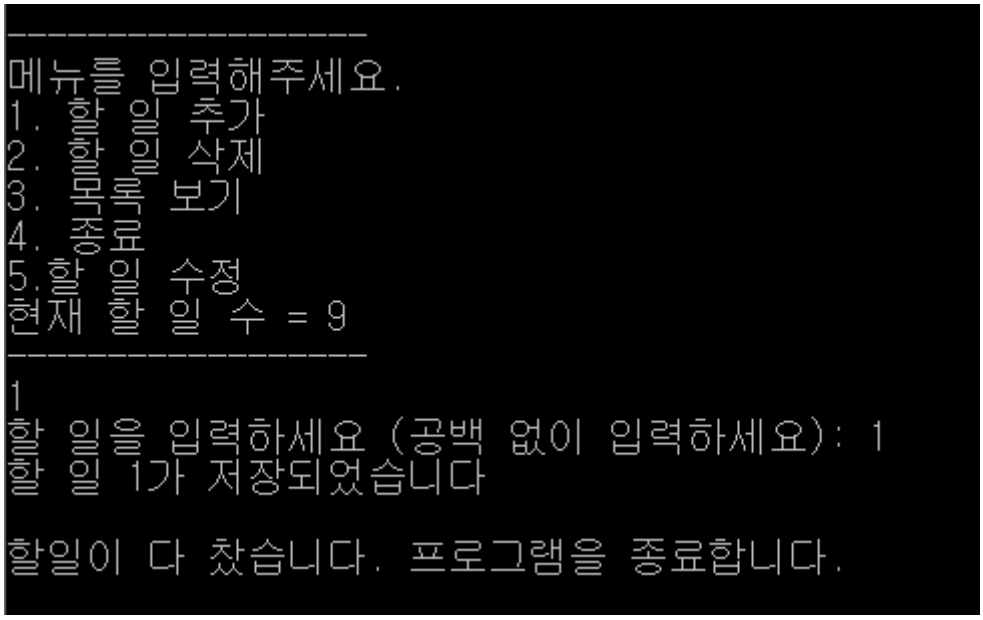
3. 목록 출력


메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 5
4
종료를 선택하셨습니다. 프로그램을 종료합니다.

4. 종료

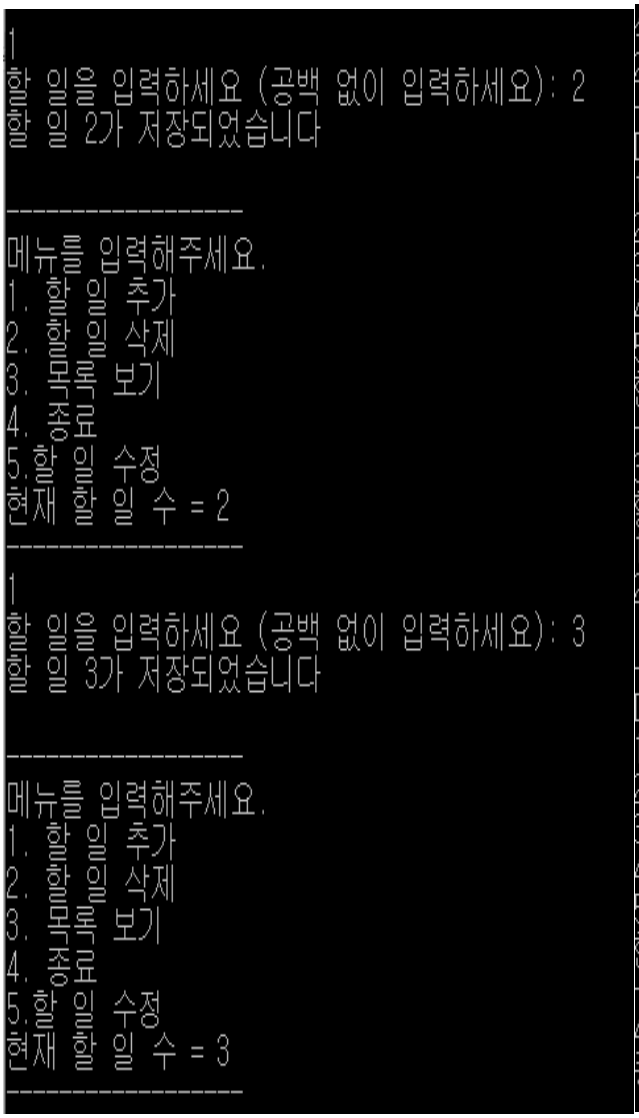

3
할 일 목록
1. 1
2. 2
3. 3
4. 4
5. 5
메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 5
5
수정하려는 할 일의 번호를 입력해주세요. (1부터 시작) : 2
2: 할 일을 수정합니다.
수정 내용을 입력해주세요.
6
메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 5
3
할 일 목록
1. 1
2. 6
3. 3
4. 4
5. 5

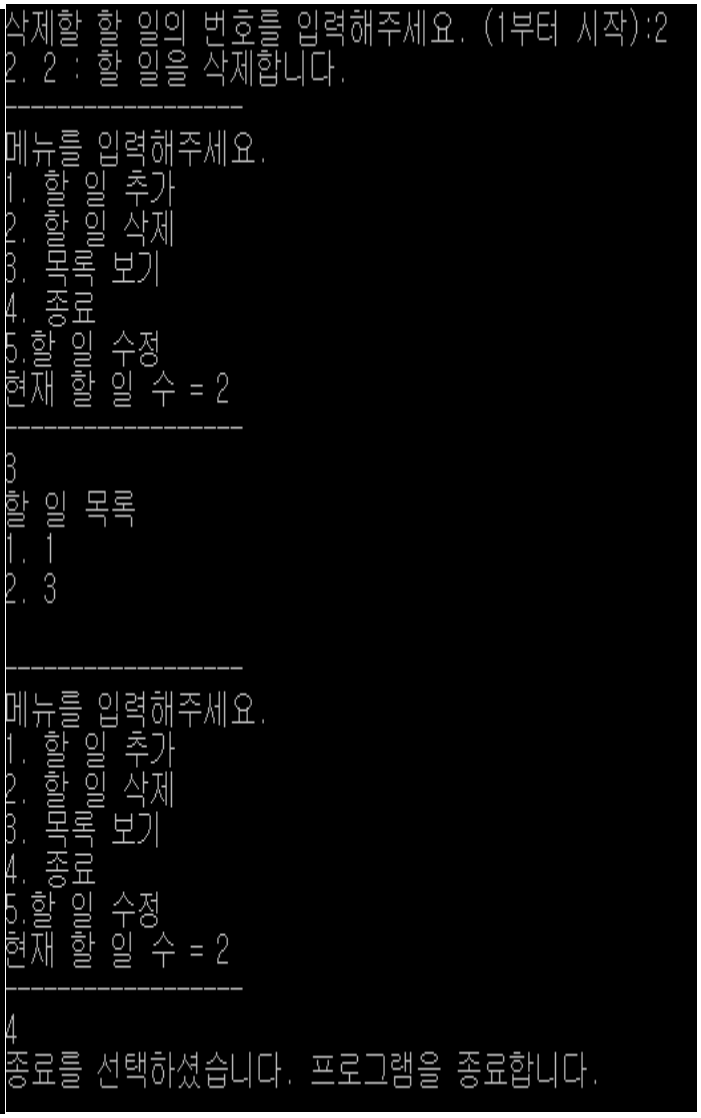
5. 수정


메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5.할 일 수정
현재 할 일 수 = 9
1
할 일을 입력하세요 (공백 없이 입력하세요): 1
할 일 1가 저장되었습니다
할일이 다 찼습니다. 프로그램을 종료합니다.

6. 자동 종료

2. 최종 테스트 - 모든 함수 연속으로 실행





5. 결과 및 결론

1. 프로젝트 결과: Todo 관리 시스템 설계

- 할 일의 입력과 삭제, 수정, 목록 출력하는 시스템을 2차원 배열을 기반으로 설계하였습니다.

- 1주차부터 학습하였던 c언어의 여러가지 함수 등을 사용하여 각 기능을 만들었습니다.

2. 개인적인 의견

- 10가지의 할 일을 입력 받고 자동종료하는 기능은 불필요한 것 같습니다. 모든 자동종료 기능을 삭제하고 월, 일 단위로 할 일을 구분하는 기능이 있었으면 더 활용하기 좋을 것 같습니다. 할 일의 목록중 완료한 일 들을 (~~1.할 일~~) 등으로 표시 할 수 있는 기능이 있으면 좋을 것 같습니다. 목록 보기 기능을 현재 할 일 수 = 자리에 목록을 자동 출력하는 것도 좋을 것 같습니다.