8주차 프로젝트형 실습 보고서

231344최세인

1. 서론

   1) 프로젝트 주제: 할 일 관리 프로그램 (TODO리스트) 만들기

   2) 프로젝트의 목적: 7주차까지 학습한 내용을 활용해 코드를 작성해보고 C언어에 대한 이해도를 확인하고자 프로젝트 진행

2. 설계

   1) 요구사항: 할 일 추가, 삭제, 출력, 수정 및 프로그램 종료 가능한 프로그램

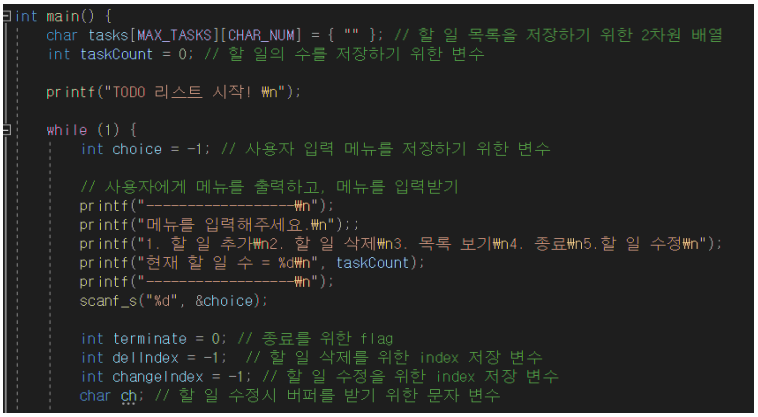
   2) 요구사항의 기능:

      1. 할 일 추가: 사용자에게 할 일 입력 받고 저장
      2. 할 일 삭제: 삭제 할 인덱스를 입력 받고 해당 할 일 삭제
      3. 할 일 출력: 입력한 할 일 목록으로 보여주기
      4. 할 일 수정: 수정 할 인덱스를 입력 받고 새로운 할 일로 변경
      5. 프로그램 종료
      6. 할 일이 10개로 다 찼을 시, 할 일이 다 찼다고 출력하고 종료

   3) 주요 코드: 문자열, 기능 함수화, 반복문, 조건문

3. 구현

   1) 메뉴 출력 코드 블록



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

    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₩n");
        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
```

- Tasks: 할 일 저장하기 위한 10 x 100 2차원 배열 선언
- taskCount: 할 일 수를 저장할 변수 선언
- 사용자에게 원하는 메뉴의 번호를 입력 받는다.

2) 할 일 추가 코드 블록

```
case 1:
    if (taskCount > 9) {
        printf("할 일이 다 찼습니다. 프로그램을 종료합니다.₩n");
        terminate = 2; // 프로그램 종료를 위한 flag
        break;
        // 할 일이 10개로 다 찬 경우 프로그램 종료
    }
    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;
```

- 추가할 할 일을 입력받은 후, 문자열에 저장
- 입력 받은 후 taskCount +1만큼 증가

```
if (terminate == 2) {
    return 0;
}  //할 일이 10개로 다 찬 경우 switch문에서 벗어나 프로그램 종료
```

- 할 일이 10개로 다 찬 경우 terminate==2일 때 실행되는 조건문 사용해 프로그램 종료

3) 할 일 추가 함수화

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

//할 일 추가 함수
//함수 선언 및 정의 전에 배열과 변수를 설정해줬기 때문에 매개변수가 없는 함수
void addTask(void) {
    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]);
}
```

- 함수 밖에서 문자열 tasks와 변수 taskCount를 선언했으므로 매개변수가 없고 반환값 없는 함수 정의 및 선언

4) 할 일 삭제 코드 블록

```
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;
    }
    break;
```

- delIndex: 삭제할 할 일 저장할 변수 선언
- 삭제할 할 일의 번호를 인덱스에 받는다
- 할 일 번호 잘못 입력 시 오류 메시지 출력
- 배열간 대입이 불가하므로 문자열 복사 함수로 배열에 저장된 할 일 삭제
- 삭제 된 할 일 뒤에 있는 할 일들 번호 앞으로 옮기기
- taskCount 하나 감소

5) 할 일 삭제 함수화

```
//할 일 삭제 함수
void delTask(int delIndex, int taskCount) {
    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]);
    }
}
```

- delIndex와 taskCount를 매개변수로 반환하지 않는 함수 선언 및 정의

6) 목록 보기 코드 블록

```
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;
```

- For문 사용해서 할 일 목록 출력

7) 목록 보기 함수화

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

- taskCount를 매개변수로 반환하지 않는 함수 선언 및 정의

8) 프로그램 종료 코드 블록

```
case 4:
    terminate = 1;
    break;
```

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

- 종료 flag를 이용해서 switch문 밖에서 프로그램 종료

9) 할 일 수정 코드 블록

```
case 5:
    //할 일 수정하는 코드블록
    printf("수정할 할 일의 번호를 입력해주세요. (1부터 시작):");
    scanf_s("%d", &changeIndex);
    ch = getchar();
    if (changeIndex > taskCount || changeIndex <= 0) {
        printf("수정 범위가 벗어났습니다.₩n");
    }
    else {
        printf("%d. %s : 할 일을 수정합니다.₩n", changeIndex, tasks[changeIndex - 1]);
        strcpy_s(tasks[changeIndex - 1], sizeof(tasks[changeIndex - 1]), "");
        // 문자열 복사 함수로 삭제 후 수정할 할 일 입력받기
        printf("(공백없이 입력하세요.)");
        scanf_s("%s", tasks[changeIndex - 1], (int)sizeof(tasks[changeIndex - 1]));
        // 입력받은 할 일 처음 문자열 위치에 다시 저장


    }
    break;
```

- changeIndex: 수정할 할 일 저장하는 변수
- ch: 버퍼 받기 위한 문자 변수
- 번호 잘못 입력 시 오류 메시지 출력
- 문자열 복사 함수로 배열에 저장된 할 일 삭제 후, 다시 입력 받아서 배열에 저장

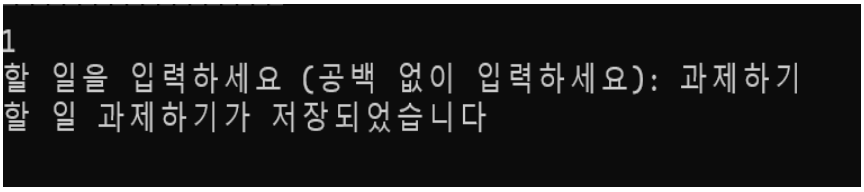
10) 메뉴에 없는 번호 입력 시 실행되는 코드 블록

```
default:
    printf("잘못된 선택입니다. 다시 선택하세요.₩n");
}
```

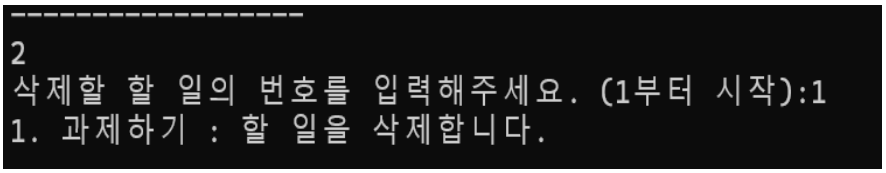
- default 사용해 번호 잘못 입력 시 오류 메시지 출력

4. 테스트

1) 할 일 추가

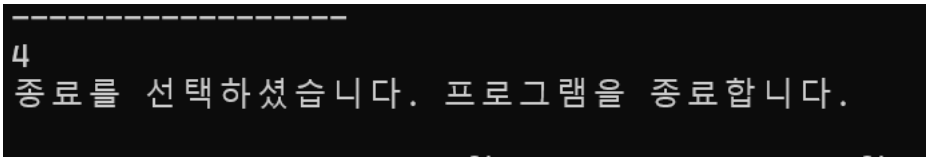


2) 할 일 삭제

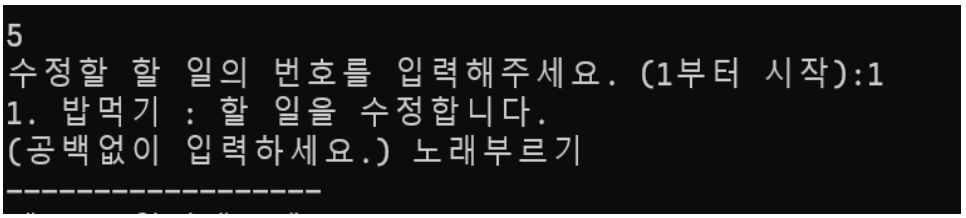


3) 목록 보기

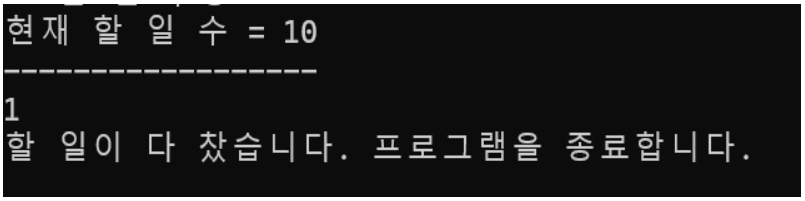
4) 프로그램 종료



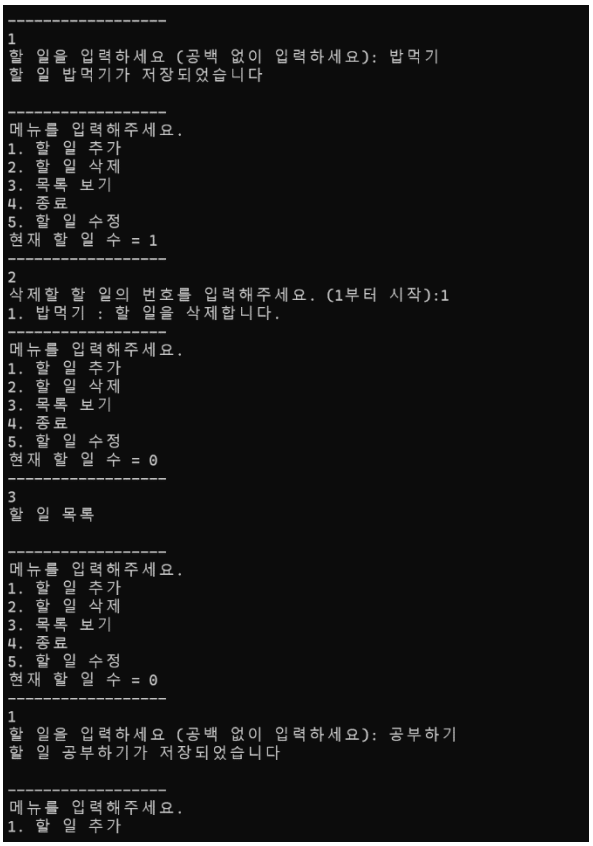
5) 할 일 수정



6) 할 일 10개 다 찬 경우



7) 최종



5. 결과 및 결론

1) 프로젝트 결과: 지금까지 배운 C언어를 가지고 5가지 메뉴의 TO DO리스트를 구현해 보았다.

2) 느낀점: 확실히 이론으로만 C언어를 접했을 때보다 난이도가 높았고 어렵게 느껴졌다. 변수와 배열 설정부터 메인 함수, 각 기능별 함수까지 생각할 부분이 많았고 오류도 많았다. 코드를 작성하고 빌드하는 과정이 가장 기억에 남는다. 빌드를 할 때마다 오류 없이 실행되길 간절히 바랬기 때문이다. 오류가 났을 때는 당황스럽고 수정하는 과정이 어려웠지만 되돌아 보니 놓친 부분을 확인 할 수 있어서 코딩하는 데 많은 도움이 된 것 같다. 간단한 TO DO리스트를 만드는 데도 꽤 오랜 시간이 걸렸고 여러 방법을 생각해 내는 게 쉽지 않았다. C언어에 능숙한 프로그래머들의 대단함도 느꼈고 앞으로 더 많은 공부가 필요함도 느꼈다. 처음에는 막막하기만 했던 코드들이 실행되어 제대로 기능을 했을 때 느낀 희열을 토대로 계속해서 다양한 코드를 접하고 직접 구현해 보면서 실력을 키워보고 싶다.