

1. 프로젝트 목적 및 배경

7주차 까지 배운 내용에 대한 실습을 위해 진행 하였습니다.

2. 목표

TODO리스트를 만들고 할 일을 수정하는 코드 만들기.

3. 사용자 요구 사항

사용자가 할 일을 입력, 삭제, 출력, 수정 할 수 있는 프로그램

4. 기능 요구사항

① 사용자에게 작업 요청 받기

- 1. 할 일 추가, 2. 할 일 삭제, 3. 목록 보기, 4. 종료, 5. 수정

② 요청 받은 작업에 따라 아래 기능 수행

- ① 할 일 추가를 입력했을 경우, 사용자에게 할 일을 입력 받고 저장
- ② 할 일 삭제를 입력했을 경우, 인덱스를 입력 받고 해당 할 일 삭제
- ③ 목록 보기를 입력했을 경우, 전체 할 일 목록을 보여주기
- ④ 종료를 입력했을 경우, 프로그램 종료
- ⑤ 할 일 수정을 입력했을 경우, 인덱스와 할 일 (문자열)을 입력 받고, 해당 인덱스의 할 일 변경

- 주의: 입력 받는 인덱스에 -1 한 것이 실제 배열의 인덱스가 됨

③ 할 일이 10개로 다 찬 경우는 할 일이 다 찼다고 출력하고 프로그램 종료

5. 요구사항 별 코드

① 사용자에게 작업 요청 받기

1.

```
int choice = -1; // 사용자 입력 초기화

// 사용자에게 메뉴를 출력하고, 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);
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- taskCount = 현재 작업 수
- choice = 사용자가 요청할 작업 번호

3. 반환 값 (함수의 경우 작성)

- 함수가 아니므로 없음

4. 결과 (블록/함수가 종료된 결과)

- 작업을 추가, 삭제, 보기, 수정 종료 하는 메뉴를 출력

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- 사용자에게 다양한 메뉴 옵션을 제공하고 작업 번호를 입력 받는다.
- 사용자의 선택에 따라 다양한 결과를 갖는다.

② 요청 받은 작업에 따라 아래 기능 수행

① 할 일 추가를 입력했을 경우, 사용자에게 할 일을 입력 받고 저장

1.

```
//선택에 따라 실행되는 기능 코드 블록
switch (choice) {
case 1: //choice=1일때
    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++; //할 일 변수에 1추가
    break;
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- choice = 사용자가 메뉴에서 선택한 작업 번호

3. 반환 값 (함수의 경우 작성)

- 함수가 아니므로 없음

4. 결과 (블록/함수가 종료된 결과)

- 할 일 추가 작업을 수행하며, 사용자가 입력한 할 일을 배열에 저장하고, 할 일 수를 1증가시킴

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- case 1:은 choice값이 1일 때 실행
- scanf_s로 사용자의 할 일을 입력 받아서 할 일은 tasks 배열의 taskCount에 저장
- 할 일이 저장되었음을 출력
- taskcount 값을 1증가시키고 break; 를 통해 switch 블록에서 빠져나감

② 할 일 삭제를 입력했을 경우, 인덱스를 입력 받고 해당 할 일 삭제

1.

```
case 2://choice=2일때
// 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
    // 문자열을 정의된 문자 배열에 대입
    strcpy_s(tasks[delIndex - 1], sizeof(tasks[delIndex - 1]), "");

    // 특정 인덱스의 할 일 삭제 후 뒤에 있는 할 일을 앞 번호로 가져오기
    for (int i = delIndex; i < taskCount + 1; i++) {
        strcpy_s(tasks[i - 1], sizeof(tasks[i]), tasks[i]);
    }
    taskCount -= 1;
}
break;
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- taskCount = 현재 작업 수

- tasks = 할 일 목록 저장 2차원 배열
- delIndex=사용자가 삭제할 인덱스 번호

3. 반환값 (함수의 경우 작성)

- 함수가 아니므로 없음

4. 결과 (블록/함수가 종료된 결과)

- 할 일 삭제 작업을 수행하며, 사용자가 선택한 인덱스에 해당하는 할 일을 배열에서 삭제

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- 사용자에게 삭제할 할 일의 인덱스를 입력 받는다. 삭제 범위를 벗어나면 "삭제 범위가 벗어났습니다." 라는 메시지가 출력
- 그렇지 않으면 입력 받은 인덱스 -1 에 있는 할 일을 배열 간에 문자열을 대입할 수 없어서 strcpy_s를 사용하여 삭제 위치에 빈 문자열을 할당하고 할 일 들을 앞 번호로 이동 후 taskcount를 1감소시켜 작업 수를 업데이트함
- break; 를 통해 switch 블록에서 빠져나감

③ 목록 보기를 입력했을 경우, 전체 할 일 목록을 보여주기

1.

```
case 3://choice=3일때
    printf("할 일 목록\n"); //문구 출력
    for (int i = 0; i < taskCount; i++) { //할 일 변수에 저장된 수보다 작을 때까지
        printf("%d. %s\n", i + 1, tasks[i]); //할 일 목록 순서대로 하나씩 반환
    }
    printf("\n");
    break;
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- taskCount = 현재 작업 수
- tasks = 할 일 목록 저장 2차원 배열

3. 반환값 (함수의 경우 작성)

- 함수가 아니므로 없음

4. 결과 (블록/함수가 종료된 결과)

- 현재 저장된 할 일 목록을 출력

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- "할일 목록" 라는 메시지 출력 후 for문을 통해 모든 할일 목록을 순서대로 출력
- taskCount보다 작을 때 까지 증가시키며 i+1로 할 일 번호를 출력하고 task[i]에서 할 일의 내용을 가져와 출력
- break; 를 통해 switch 블록에서 빠져나감

④ 종료를 입력했을 경우, 프로그램 종료

1.

```
case 4://choice=4일때
    terminate = 1; //종료를 위한 변수에 1대입
    break;
```

```
if (terminate == 1) { //종료를 위한 변수가 1일때
    printf("종료를 선택하셨습니다. 프로그램을 종료합니다.\n"); //문구 출력
    break; //반복문 탈출
}
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- terminate = 종료를 위한 변수

3. 반환값 (함수의 경우 작성)

- 함수가 아니므로 없음

4. 결과 (블록/함수가 종료된 결과)

- 프로그램을 종료시키고 종료 문구 출력

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- terminate를 1로 설정하여 종료를 나타내고 break; 를 통해 switch 블록에서 빠져나감
- if문을 사용하여 종료 문구를 출력하고 다시 break;를 통해 반복문에서 탈출하여 프로그램을 종료함

⑤ 할 일 수정을 입력했을 경우, 인덱스와 할 일 (문자열)을 입력 받고, 해당 인덱스의 할 일 변경

1.

```

case 5://choice=5일때
printf("수정할 할 일의 번호를 입력해주세요. (1부터 시작):"); //문구 출력
scanf_s("%d", &changeIndex); //수정할 인덱스 입력받기
printf("할 일을 입력하세요 (공백 없이 입력하세요): "); //문구 출력
scanf_s("%s", tasks[taskCount], (int)sizeof(tasks[taskCount])); //할 일 입력받기
strcpy_s(tasks[changeIndex - 1], sizeof(tasks[changeIndex - 1]), tasks[taskCount]); //문자열을 정의된 문자 배열에 대입
printf("할 일 \"%s\"가 저장되었습니다.\n\n", tasks[taskCount]); //문구 출력
break;

```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- taskCount = 현재 작업 수
- tasks = 할일 목록 저장 2차원 배열
- changeIndex=사용자가 수정할 인덱스 번호

3. 반환값 (함수의 경우 작성)

- 함수가 아니므로 없음

4. 결과 (블록/함수가 종료된 결과)

- 할 일 삭제 작업을 수행하며, 사용자가 선택한 인덱스에 해당하는 할 일을 배열에서 삭제

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- 사용자에게 수정할 할 일의 인덱스를 입력 받음
- 새로운 할 일을 입력받아 tasks[taskCount]에 저장하고 strcpy_s를 사용해 새로운 할 일을 tasks 배열의 수정할 인덱스 (changeIndex-1)에 복사함
- 사용자가 입력한 할 일이 저장되었다는 메시지 출력
- break; 를 통해 switch 블록에서 빠져나감

③ 할 일이 10개로 다 찬 경우는 할 일이 다 찼다고 출력하고 프로그램 종료

1.

```

//코드를 종료하는 코드 블록
if (taskCount == 10) { //할일이 10일때
printf("할 일이 다 찼습니다. 프로그램을 종료합니다.\n"); //문구 출력
break; //반복문 탈출
}

```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- taskCount = 현재 작업 수

3. 반환값 (함수의 경우 작성)

- 함수가 아니므로 없음

4. 결과 (블록/함수가 종료된 결과)

- 프로그램을 종료시키고 할 일이 다 차서 종료되었다는 문구 출력

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- if문을 사용해 taskCount가 10일 때 실행됨
- 만약 조건이 참이라면 즉, 할 일이 10개라면 "할 일이 다 찼습니다. 프로그램을 종료합니다" 라는 문구를 출력하고 break;를 통해 반복문에서 탈출하여 프로그램을 종료함

6. 테스트

1. 기능 별 테스트 결과

① 사용자에게 작업 요청 받기

```
-----  
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|
```

② 요청 받은 작업에 따라 아래 기능 수행

① 할 일 추가를 입력했을 경우, 사용자에게 할 일을 입력 받고 저장

```
1  
할 일을 입력하세요 (공백 없이 입력하세요): 공부하기  
할 일 공부하기가 저장되었습니다
```

② 할 일 삭제를 입력했을 경우, 인덱스를 입력 받고 해당 할 일 삭제

```
2  
삭제할 할 일의 번호를 입력해주세요. (1부터 시작):2  
2. 밥먹기 : 할 일을 삭제합니다.
```

③ 목록 보기를 입력했을 경우, 전체 할 일 목록을 보여주기

```
3  
할 일 목록  
1. 공부하기
```

④ 종료를 입력했을 경우, 프로그램 종료

```
4  
종료를 선택하셨습니다. 프로그램을 종료합니다.  
C:\Users\82106\source\repos\Project silsep\x64\Debug\Project silsep.exe(프로세스 27580개)이(가) 종료되었습니다(코드: 0개)  
).  
이 창을 닫으려면 아무 키나 누르세요...
```

⑤ 할 일 수정을 입력했을 경우, 인덱스와 할 일 (문자열)을 입력 받고, 해당 인덱스의 할 일 변경

5

수정할 할 일의 번호를 입력해주세요. (1부터 시작):1
할 일을 입력하세요 (공백 없이 입력하세요): 공부하기
할 일 공부하기가 저장되었습니다

③ 할 일이 10개로 다 찬 경우는 할 일이 다 찼다고 출력하고 프로그램 종료

```
현재 할 일 수 = 9
-----
1
할 일을 입력하세요 (공백 없이 입력하세요): 강의듣기
할 일 강의듣기가 저장되었습니다

할 일이 다 찼습니다. 프로그램을 종료합니다.

C:\Users\82106\source\repos\Project silsep\x64\Debug\Project silsep.exe(프로세스 22312개)이(가) 종료되었습니다(코드: 0개
).
이 창을 닫으려면 아무 키나 누르세요...
```

2. 최종 테스트 스크린샷

<pre>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): 밥먹기 할 일 밥먹기가 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): 공부 할 일 공부가 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 ----- 3 할 일 목록 1. 밥먹기 2. 공부</pre>	<pre>----- 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):2 2. 공부 : 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 ----- 3 할 일 목록 1. 밥먹기 ----- 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): 공부 할 일 공부가 저장되었습니다</pre>
---	---

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

5
수정할 할 일의 번호를 입력해주세요. (1부터 시작):2
할 일을 입력하세요 (공백 없이 입력하세요): 씨언어공부하기
할 일 씨언어공부하기가 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

3
할 일 목록
1. 밥먹기
2. 씨언어공부하기

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

4
종료를 선택하셨습니다. 프로그램을 종료합니다.
C:\Users\82106\source\repos\Project silsep\x64\Debug\Project silsep.exe(프로세스 30224개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

7. 결과 및 결론

① 프로젝트 결과:

할 일을 추가하고 삭제하고 목록을 보여주고 수정이 가능한 할 일 관리 프로그램을 만들었다.

프로그램은 종료인 4번을 입력하면 종료되고 또는 할 일이 10개가 되면 종료가 된다.

② 느낀 점:

할 일 관리 시스템을 통해 코드의 가독성과 유지보수성을 고려한 구조를 만드는 방법을 배웠고 사용자의 요구에 맞춰 기능으로 구분하고 우선순위를 정하여 기술을 단계별로 구성하다 보니 프로젝트를 통해 능력을 향상 시킬 수 있었고 다양한 프로그램 개념을 실제로 적용하는 기회였던 것 같습니다. 사용자 입력, 함수, 데이터 구조 등을 다루며 기술적 역량이 향상되었습니다. 또한 디버깅 하는 과정을 통해 오류를 발견하고 문제를 해결하는 능력 또한 향상시켰습니다. 할 일 관리 프로그램을 통해 프로그래밍 능력과 프로젝트 관리 및 문제 해결 능력을 강화할 수 있는 좋은 경험이었습니다. 이러한 경험은 미래의 프로젝트나 직무에서 유용하게 활용될 것입니다.