인공지능학부 231126 채수아

1. 서론

1) 프로젝트 목적 및 배경: 7주차까지 배운 내용을 이해했는지 확인하기 위해 프로젝트 실습을 진행함

2) 목표: 사용자의 할 일을 관리해주는 프로그램, TODO 리스트 관리 프로그램 만들기

2. 요구사항

1) 사용자 요구사항: 사용자가 할 일을 입력, 삭제, 수정, 출력할 수 있는 프로그램

2) 기능 요구사항:

① 사용자에게 작업 요청 받기

(1. 할 일 추가, 2. 할 일 삭제, 3. 목록 보기, 4. 종료)

② 요청 받은 작업에 따라 아래 기능 수행

- 할 일 추가를 입력했을 경우, 사용자에게 할 일을 입력 받고 저장
- 할 일 삭제를 입력했을 경우, 인덱스를 입력 받고 해당 할 일 삭제
- 목록 보기를 입력했을 경우, 전체 할 일 목록을 보여주기
- 종료를 입력했을 경우, 프로그램 종료
- 할 일 수정을 입력했을 경우, 인덱스와 할 일(문자열)을 입력 받고, 해당 인덱스의 할 일 변경
- 할 일이 10개로 다 찬 경우는 할 일이 다 찼다고 출력하고 프로그램 종료

3. 설계 및 구현

1) 기능 별 구현 사항:

① 사용자에게 작업 요청 받기

(1) 코드블록/함수 스크린샷

```
int choice = -1; // choice를 관련 없는 수로 초기화

// 사용자에게 메뉴를 출력하고 입력받음
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

- choice = 사용자가 선택하는 메뉴 숫자
- taskCount = 현재 할 일 수

(3) 반환값

- 함수가 아니므로 없음

(4) 결과

- 사용자가 입력한 메뉴 번호를 choice에 저장함

(5) 설명

- 사용자가 메뉴를 볼 수 있게 출력함
- 사용자에게 실행할 메뉴 번호를 입력 받음

② 할 일 추가하기

(1) 코드블록/함수 스크린샷

```
case 1:
    // 추가할 할 일을 입력받아 저장함
    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;
```

(2) 입력

- taskCount = 현재 할 일 수
- tasks = 할 일 목록을 저장하는 2차원 배열

(3) 반환값

- 함수가 아니므로 없음

(4) 결과

- 사용자가 입력한 할 일이 추가된 tasks

(5) 설명

- 사용자에게 추가할 할 일을 입력 받음
- 입력 받은 할 일을 task에 추가함

③ 할 일 목록 보여주기

(1) 코드블록/함수 스크린샷

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

(2) 입력

- taskCount = 현재 할 일 수
- tasks = 할 일 목록을 저장하는 2차원 배열

(3) 반환값

- 함수가 아니므로 없음

(4) 결과

- 할 일 목록을 출력함

(5) 설명

- for문으로 task를 돌면서 할 일 목록을 출력함

④ 할 일 수정하기

(1) 코드블록/함수 스크린샷

```
case 5:
    // 수정할 할 일의 번호를 입력받음
    printf("할 일 수정₩n");
    printf("수정할 할 일의 번호를 입력해주세요. (1부터 시작): ");
    scanf_s("%d", &changeIndex);

    // 없는 번호를 입력한 경우 실행
    if (changeIndex > taskCount || changeIndex <= 0) {
        printf("수정 범위가 벗어났습니다.₩n");
    }

    // 번호를 제대로 입력한 경우 실행
    else {
        printf("%d. %s : 할 일을 수정합니다.₩n", changeIndex, tasks[changeIndex - 1]);

        // 기존의 할 일을 새로운 할 일로 초기화함
        printf("새로운 할 일을 입력하세요 (공백 없이 입력하세요): ");
        scanf_s("%s", tasks[changeIndex - 1], (int)sizeof(tasks[changeIndex - 1]));
        printf("할 일 ""%s""가 저장되었습니다₩n₩n", tasks[changeIndex - 1]);
    }
    break;
```

(2) 입력

- changeIndex = 사용자가 수정할 할 일의 번호
- taskCount = 현재 할 일 수
- tasks = 할 일 목록을 저장하는 2차원 배열

(3) 반환값

- 함수가 아니므로 없음

(4) 결과

- 새롭게 수정된 tasks

(5) 설명

- 수정할 할 일의 번호를 입력 받음
- 제대로 된 번호를 입력했는지 확인함
- 원래 있던 할 일을 새로운 할 일로 수정함

⑤ 할 일 삭제하기

(1) 코드블록/함수 스크린샷

```
case 2:
    // 삭제할 할 일의 번호를 입력받음
    printf("삭제할 할 일의 번호를 입력해주세요. (1부터 시작): ");
    scanf_s("%d", &delIndex);

    // 없는 번호를 입력한 경우 실행
    if (delIndex > taskCount || delIndex <= 0) {
        printf("삭제 범위가 벗어났습니다.₩n");
    }

    // 번호를 제대로 입력한 경우 실행
    else {
        printf("%d. %s : 할 일을 삭제합니다.₩n", delIndex, tasks[delIndex - 1]);

        // 배열에 문자 배열인 문자열의 대입이 불가능하므로
        // 문자열 복사 함수로 공백을 복사해 삭제
        strcpy_s(tasks[delIndex - 1], sizeof(tasks[delIndex - 1]), "");

        // 사용자가 선택한 할 일 삭제 후 순서 앞당기기
        for (int i = delIndex; i < taskCount + 1; i++) {
            strcpy_s(tasks[i - 1], sizeof(tasks[i]), tasks[i]);
        }
        taskCount -= 1;
    }
    break;
```

(2) 입력

- delIndex = 사용자가 삭제할 할 일의 번호
- taskCount = 현재 할 일 수
- tasks = 할 일 목록을 저장하는 2차원 배열

(3) 반환값

- 함수가 아니므로 없음

(4) 결과

- 사용자가 고른 할 일이 삭제된 tasks

(5) 설명

- 삭제할 할 일의 번호를 입력 받음
- 제대로 된 번호를 입력했는지 확인함
- 원래 있던 할 일을 삭제하고 순서를 앞당김

⑥ 프로그램 종료

(1) 코드블록/함수 스크린샷

```
// terminate 값이 1이 되면 종료함
if (terminate == 1) {
    printf("종료를 선택하셨습니다. 프로그램을 종료합니다.Wn");
    break;
}
```

(2) 입력

- terminate = 프로그램 종료를 위한 변수

(3) 반환값

- 함수가 아니므로 없음

(4) 결과

- 프로그램을 종료함

(5) 설명

- terminate 값이 1인 경우 프로그램을 종료함

⑦ 할 일이 다 찬 경우 프로그램 종료

(1) 코드블록/함수 스크린샷

```
// 할 일 10개가 다 찬 경우 실행하고 프로그램을 종료
if (taskCount == 10) {
    printf("할 일이 다 찼습니다. 프로그램을 종료합니다.Wn");
    break;
}
```

(2) 입력

- taskCount = 현재 할 일 수

(3) 반환값

- 함수가 아니므로 없음

(4) 결과

- 프로그램을 종료함

(5) 설명

- taskCount 값이 10이 될 경우 프로그램을 종료함

4. 테스트

1) 기능 별 테스트 결과:

① 할 일 추가하기

```
1
할 일을 입력하세요 (공백 없이 입력하세요): 과제하기
할 일 과제하기가 저장되었습니다
```

② 할 일 목록 보여주기

```
3
할 일 목록
1. 과제하기
```

③ 할 일 수정하기

```
5
할 일 수정
수정할 할 일의 번호를 입력해주세요. (1부터 시작): 1
1. 과제하기 : 할 일을 수정합니다.
새로운 할 일을 입력하세요 (공백 없이 입력하세요): 과제제출하기
할 일 과제제출하기가 저장되었습니다
```

④ 할 일 삭제하기

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

⑤ 프로그램 종료

```
4
종료를 선택하셨습니다. 프로그램을 종료합니다.
```

⑥ 할 일이 다 찬 경우 종료

```
할 일이 다 찼습니다. 프로그램을 종료합니다.
```

2) 최종 테스트 스크린샷:

```
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): 과제하기
할 일 과제하기가 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
------------------
3
할 일 목록
1. 과제하기
------------------
메뉴를 입력해주세요.
1. 할 일 추가
2. 할 일 삭제
3. 목록 보기
4. 종료
5. 할 일 수정
현재 할 일 수 = 1
------------------
5
할 일 수정
수정할 할 일의 번호를 입력해주세요. (1부터 시작): 1
1. 과제하기 : 할 일을 수정합니다.
새로운 할 일을 입력하세요 (공백 없이 입력하세요): 과제제출하기
할 일 과제제출하기가 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
------------------
2
삭제할 할 일의 번호를 입력해주세요. (1부터 시작): 1
1. 과제제출하기 : 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
------------------
4
종료를 선택하셨습니다. 프로그램을 종료합니다.
```

5. 결과 및 결론

1) 프로젝트 결과: 사용자의 할 일을 관리해주는 프로그램, TODO 리스트 관리 프로그램을 만들었다.

2) 느낀 점: 할 일 수정하기 기능과 할 일이 10개로 다 찬 경우 프로그램을 종료하는 기능을 추가하는 것은 어렵지 않았다. 그러나 할 일 추가 기능, 할 일 삭제 기능, 할 일 목록 출력 기능을 함수화하는 것은 조금 어려웠다. 함수를 만들면서 관련 코드를 붙여넣기 했는데, 그대로 실행이 안 돼서 무엇을 추가하고 수정해야 하는지 전혀 감이 안 와서 정답이랑 비교도 해보고 인터넷 검색도 하면서 풀었다. 프로젝트 실습 연습도 이렇게 힘든데 실제 기말 대체 프로젝트 실습은 어떻게 해야 할지 걱정이 됐다.