C프로그래밍및실습

대학생활 도우미 프로그램

최종 보고서

제출일자: 2023-12-10
제출자명: 최지원
제출자학번: 224571

1. 프로젝트 목표 (16 pt)

1) 배경 및 필요성 (14 pt)

학교에 막 입학한 학생이나 복학한 학생들은 학교 생활에 적응하기 힘들 수도 있다. 수업 장소는 어딘지 정확한 수업 시간은 언제인지 쉬는 시간은 몇 분인지 정확하게 파악하기 힘든 경우가 많다. 또 학점을 관리하고 해야 할 일이 많은 경우 기억하지 못하는 경우도 있다. 이 모든 걸 관리해주는 프로그램이 있으면 학교 생활에 더 도움이 될 것이라고 생각한다.

2) 프로젝트 목표

수업 장소와 시간, 교수님을 알려주고 목표학점을 입력하면 그에 맞게 학점을 분석해주고 투두리스트를 만들어 해야할 일의 마감기한을 관리할 수 있는 학교생활 도우미 프로그램

3) 차별점

기존에 있던 일반적인 대학생 앱과 같은 서비스와는 개인의 학교생활 관리에 더 중심적인 프로그램임. 더 직관적이게 원하는 항목을 선택 가능하게 하여 사용자가 더 쉽게 프로그램을 이용할 수 있음. 해야할 일 마감기한과 학점 분석과 같은 기능을 추가하여 기존에 앱과는 차별화된 기능이 있음.

2. 기능 계획

1) 시간표 정리

- 현재 시간표를 정리하는 기능

(1) 사용자가 수업 시간, 장소, 교수님을 입력함

(2) 사용자 입력 후 현재 시간표를 정리해서 보여준다

- 입력이 끝난 후 수업 시간, 수업 장소, 교수님 정보를 확인 가능함

(3) 시간표의 수업 수정, 삭제, 추가 기능

2) 성적 분석

- 목표 성적과 현재 각 과목의 성적을 입력하면 목표성적까지의 성적을 분석

(1) 목표 성적 입력

(2) 사용자의 과목별 성적 입력

(3) 사용자의 성적 평균 분석

(4) 목표 성적까지 얼마나 성적을 올려야 하는지 알려줌

3) 투두리스트 관리&디데이 기능

- 해야할 일의 마감 디데이와 해야할 일 목록을 확인하는 기능

(1) 할 일 추가, 수정, 삭제 기능

- 할 일 추가에서는 해야할 일의 이름과 마감기한을 입력 받는다.

(2) 현재 컴퓨터 시간을 기반으로 디데이 계산

- 현재 컴퓨터 시간과 입력한 시간정보를 바탕으로 디데이를 계산함

(3) 현재 해야 일 과제 목록 나열

- 해야 할 일 목록에는 디데이가 같이 출력되며 디데이가 0인경우, 1이상으로 남은 경우, 이미 지난경우 세가지 경우로 나뉘어서 다르게 메시지가 출력되도록 함.

3. 기능 구현

(1)-1. 사용자의 초기 시간표 입력

- 입출력

  - 입력: 수업 과목명, 교수명, 수업 장소와 총 듣는 수업의 개수
  - 출력: 수업 시간표

- 설명: 초기에는 의무적으로 사용자의 수업 시간표를 입력하게 하게 한다. 입력 후에는 timetableCreated라는 변수가 1이 되면서 다시 초기 전체 시간표를 입력 받지 않게 한다.

- 적용된 배운 내용: if문, 포인터, 동적 배열, 함수

- 코드 스크린샷

```
if (num == 1)
{
    clrscr();
    if (!timetableCreated) {
        int grade = 0;
        int semester = 0;

        printf("시간표 관리 메뉴입니다.₩n");
        printf("몇학년 몇학기 수업인가요? (띄어쓰기로 구분해서 학년,학기 작성)₩n");
        scanf_s("%d %d", &grade, &semester);
        printf("총 몇개의 수업을 들으시나요?");
        scanf_s("%d", &hakjum);

        timetable = (struct Course*)malloc(hakjum * sizeof(struct Course));

        if (timetable == NULL) {
            printf("메모리 할당 오류₩n");
            return;
        }

        printf("듣는 수업의 수업명, 교수명, 장소명을 차례대로 입력하세요. (띄어쓰기로 구분)₩n");

        for (int i = 0; i < hakjum; i++) {
            scanf_s("%49s %49s %49s", timetable[i].className, MAX_LENGTH, timetable[i].professorName, MAX_LENGTH, timetable[i].classPlace, MAX_LENGTH)
        }

        timetableCreated = 1;
    }
    manageTimetable(&timetable, &hakjum);
}
```

(1)-2. 사용자의 초기 시간표 입력 후 시간표 관리 메뉴

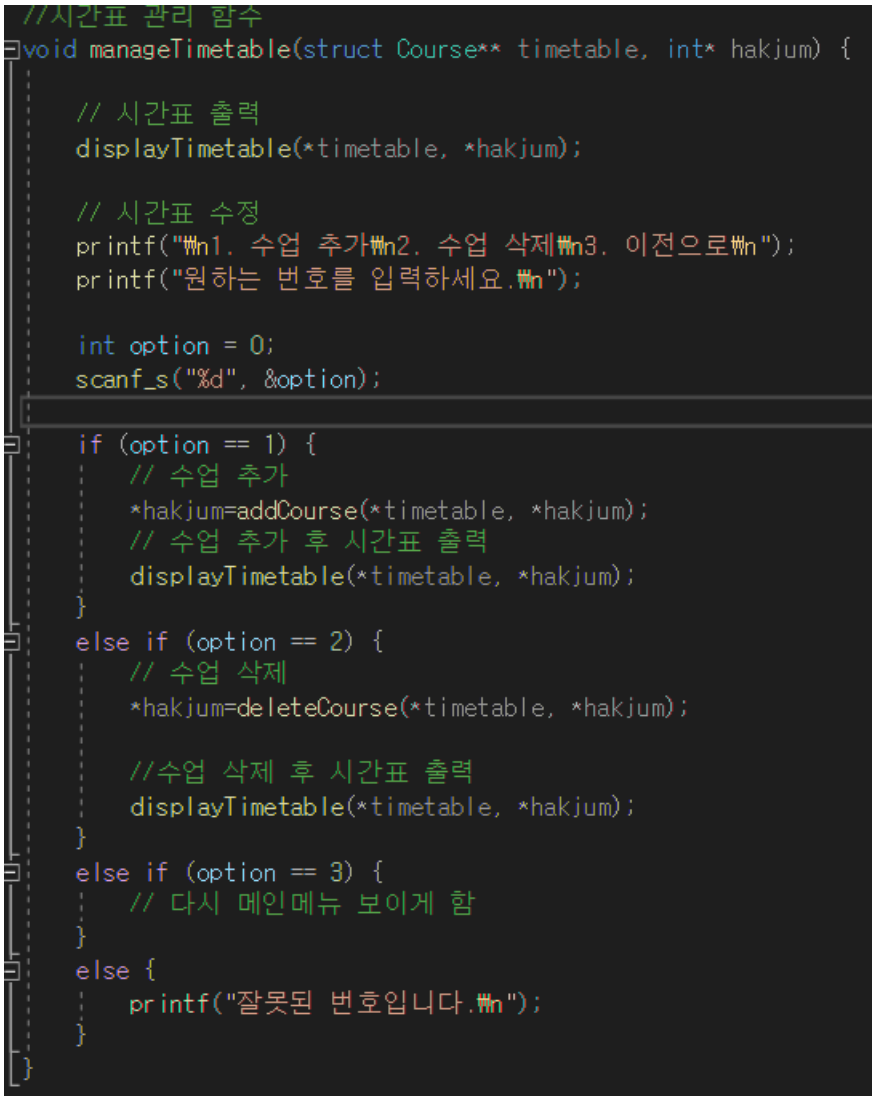
- 입출력

  - 입력: 시간표 관리 메뉴
  - 출력: 사용자가 원하는 메뉴에 관한 함수 호출

- 설명: 초기 시간표 입력 후에 전체 메인 메뉴에서 시간표 관리를 눌렀을 때 뜨게 할 화면. 수업 추가, 삭제, 수정이 가능하다.

- 적용된 배운 내용: 함수, 구조체, if문, 포인터

- 코드 스크린샷



```
//시간표 관리 함수
void manageTimetable(struct Course** timetable, int* hakjum) {

    // 시간표 출력
    displayTimetable(*timetable, *hakjum);

    // 시간표 수정
    printf("₩n1. 수업 추가₩n2. 수업 삭제₩n3. 이전으로₩n");
    printf("원하는 번호를 입력하세요.₩n");

    int option = 0;
    scanf_s("%d", &option);

    if (option == 1) {
        // 수업 추가
        *hakjum=addCourse(*timetable, *hakjum);
        // 수업 추가 후 시간표 출력
        displayTimetable(*timetable, *hakjum);
    }
    else if (option == 2) {
        // 수업 삭제
        *hakjum=deleteCourse(*timetable, *hakjum);

        //수업 삭제 후 시간표 출력
        displayTimetable(*timetable, *hakjum);
    }
    else if (option == 3) {
        // 다시 메인메뉴 보이게 함
    }
    else {
        printf("잘못된 번호입니다.₩n");
    }
}
```

(1)-3. 시간표 입력 후 수업 추가 기능

- 입출력

  - 입력: 수업 추가
  - 출력: 수업이 추가 됐다는 메시지

- 설명: 처음에 시간표를 입력하고 그 이후에 수업을 추가할 수 있는 기능

- 적용된 배운 내용: 배열, if문

- 코드 스크린샷

```
// 수업 추가 함수
int addCourse(struct Course* timetable, int hakjum) {
    printf("추가할 수업의 수업명, 교수명, 장소명을 차례대로 입력하세요. (띄어쓰기로 구분)₩n");
    scanf_s("%s %s %s", timetable[hakjum].className, MAX_LENGTH, timetable[hakjum].professorName, MAX_LENGTH, timetable[hakjum].classPlace, MAX_LENGTH);

    hakjum++;
    printf("수업이 추가되었습니다.₩n");

    return hakjum;
}
```

(1)-4. 시간표 입력 후 수업 삭제 기능

- 입출력
    - 입력: 수업 삭제
    - 출력: 수업이 삭제됐다는 메시지
- 설명: 처음에 시간표를 입력하고 그 이후에 수업을 삭제할 수 있는 기능
- 적용된 배운 내용: 반복문, 배열, if문
- 코드 스크린샷

```
// 수업 삭제 함수
int deleteCourse(struct Course* timetable, int hakjum) {
    if (hakjum > 0) {
        printf("삭제할 수업의 번호를 입력하세요 (1부터 시작)₩n");
        int courseNumber = 0;
        scanf_s("%d", &courseNumber);

        if (courseNumber >= 1 && courseNumber <= hakjum) {
            for (int i = courseNumber - 1; i < hakjum - 1; i++) {
                timetable[i] = timetable[i + 1];
            }
            hakjum--;
            printf("수업이 삭제되었습니다.₩n");
        }
        else {
            printf("잘못된 번호입니다.₩n");
        }
    }
    else {
        printf("현재 수업이 없어 삭제할 수 없습니다.₩n");
    }

    return hakjum;
}
```

(1)-5. 시간표 목록 함수

- 입출력

  - 입력: 없음
  - 출력: 시간표 목록

- 설명: 입력했던 수업들의 목록들을 수업명, 교수명, 장소명 순서대로 보여준다.

- 적용된 배운 내용: 반복문, 구조체

- 코드 스크린샷

```
//시간표 보여주는 함수
void displayTimetable(struct Course* timetable, int hakjum) {
    printf("₩n시간표:₩n----------------------₩n");
    for (int i = 0; i < hakjum; i++) {
        printf("%s | %s | %s₩n", timetable[i].className, timetable[i].professorName, timetable[i].classPlace);
    }
}
```

(2). 성적관리 프로그램

- 입출력

  - 입력: 없음
  - 출력: 성적관리 함수 호출

- 설명: 성적관리 프로그램의 전체적인 틀, 목표 성적을 기존에 입력 했다면 목표 성적이 처음에 뜨도록 함. 사용자는 성적관리 프로그램에서 원하는 항목을 선택할 수 있고 볼 수 있음.

- 적용된 배운 내용: if문, 함수

- 코드 스크린샷

```
else if (num == 2) {
    clrscr();
    if (*goalGrade != 0.0) {
        printf("당신의 목표 성적은 %.2lf 입니다.₩n₩n", *goalGrade);
    }
    else {
        printf("아직 목표 성적이 입력되지 않았습니다.₩n1번을 눌러 목표 성적을 입력하세요.₩n");
    }

    manageGrades(timetable, hakjum, goalGrade, check);
    check = 1;
}
```

(2)-1. 성적관리 프로그램 함수

- 입출력

  - 입력: 성적관리 프로그램 메뉴 번호
  - 출력: 해당 번호에 해당하는 함수 호출

- 설명: 성적관리 프로그램으로 사용자가 목표 성적을 설정할 수 있고, 과목별 성적 추가 및 수정, 성적 분석 서비스를 이용할 수 있음. setGrade 변수를 이용해서 초기 목표 성적을 의무적으로 입력하도록 함.

- 적용된 배운 내용: switch문, 함수, while문

- 코드 스크린샷

```
while (1) {
    printf(_Format:"₩n₩n성적관리 프로그램₩n----------₩n");
    printf(_Format:"1. 목표성적 입력 및 수정₩n2. 각 수업 별 성적 입력 및 수정₩n3. 성적분석₩n4. 이전으로₩n");
    scanf_s(_Format:"%d", &option);

    switch (option) {
    //목표 설정
    case 1:
        setGoalGrade(goalGrade);
        setGrade = 1;
        break;
    //과목별 성적 추가 및 수정
    case 2:
        if (setGrade) {
            if (!check) {
                setSubjectGrades(timetable, hakjum);
                check = 1;
            }
            else {
                modifySubjectGrade(timetable, hakjum);
            }
            displayGradeTimetable(timetable, hakjum);
        }
        else {
            printf(_Format:"목표 성적을 먼저 입력하세요.₩n");
        }
        break;
    //성적 분석
    case 3:
        if (setGrade) {
            analyzeGrade(timetable, hakjum, *goalGrade);
        }
        else {
            printf(_Format:"목표 성적을 먼저 입력하세요.₩n");
```

```
        }
        break;
    //성적 분석
    case 3:
        if (setGrade) {
            analyzeGrade(timetable, hakjum, *goalGrade);
        }
        else {
            printf(_Format:"목표 성적을 먼저 입력하세요.₩n");
        }
        break;

    case 4:
        return;

    default:
        printf(_Format:"잘못된 번호입니다.₩n");
        break;
    }
  }
}
```

(2)-2. 목표 성적 입력 및 수정

- 입출력

  - 입력: 사용자의 목표 성적
  - 출력: 사용자가 입력한 목표 성적

- 사용자가 목표 성적을 입력하고 수정할 수 있는 기능

- 적용된 배운 내용: if문, 포인터

- 코드 스크린샷

```
//목표 성적 입력 및 수정
void setGoalGrade(double* goalGrade) {
    if (*goalGrade == 0.0) {
        printf(_Format: "목표 성적을 입력하세요: ");
        scanf_s(_Format: "%lf", goalGrade);
        printf(_Format: "당신의 목표 성적은 %.2lf 입니다.", *goalGrade);
    }
    else {
        printf(_Format: "수정할 목표 성적을 입력하세요.₩n");
        scanf_s(_Format: "%lf", goalGrade);
        printf(_Format: "수정한 당신의 목표 성적은 %.2lf입니다.", *goalGrade);
    }
}
```

(2)-3. 과목 별 성적 입력

- 입출력

  - 입력: 사용자의 목표 성적
  - 출력: 사용자가 입력한 목표 성적

- 사용자가 수강하고 있는 과목 별 성적을 입력할 수 있는 기능.

- 적용된 배운 내용: if문, 포인터, 함수, 반복문, 배열

- 코드 스크린샷

```
//과목별 성적 입력
void setSubjectGrades(struct Course* timetable, int hakjum) {
    if (hakjum > 0) {
        for (int i = 0; i < hakjum; i++) {
            printf(_Format:"%s 과목의 성적을 입력하세요: ₩n", timetable[i].className);
            scanf_s(_Format:"%lf", &timetable[i].grade);
        }
    }
    else {
        printf(_Format:"현재 수업이 없어 성적을 입력할 수 없습니다.₩n");
    }
}
```

(2)-4. 과목 별 성적 수정

- 입출력

  - 입력: 과목 별 성적 수정
  - 출력: 사용자가 입력한 수정된 과목별 성적

- 설명: 사용자가 수강하고 있는 과목 별 성적을 수정할 수 있음.

- 적용된 배운 내용: if문, 포인터, 함수, 반복문, 구조체

- 코드 스크린샷

```
//과목 별 성적 수정
void modifySubjectGrade(struct Course* timetable, int hakjum) {
    if (hakjum > 0) {
        int subNum = 0;
        printf(_Format:"성적을 수정할 과목의 번호를 입력하세요.₩n");
        for (int i = 0; i < hakjum; i++) {
            printf(_Format:"%d. %s₩n", i + 1, timetable[i].className);
        }
        scanf_s(_Format:"%d", &subNum);

        if (subNum >= 1 && subNum <= hakjum) {
            printf(_Format:"%s 과목의 성적을 수정하세요.₩n", timetable[subNum - 1].className);
            scanf_s(_Format:"%lf", &timetable[subNum - 1].grade);
            printf(_Format:"성적이 수정 되었습니다.₩n");
        }
        else {
            printf(_Format:"잘못된 번호입니다.₩n");
        }
    }
    else {
        printf(_Format:"현재 수업이 없어 성적을 수정할 수 없습니다.₩n");
    }
}
```

(2)-5. 목표 성적과 현재 성적 비교

- 입출력

  - 입력: 사용자가 기존에 입력했던 목표 성적과 과목 별 성적
  - 출력: 목표 성적과 사용자의 성적 비교

- 목표 성적과 현재 사용자의 성적을 비교하여 알려줌. 목표 성적과 현재 성적을 비교하여 성적에 따라서 출력문이 다르게 나옴.

- 적용된 배운 내용: if문, 포인터, 배열, 함수

- 코드 스크린샷

```
//성적 분석
void analyzeGrade(struct Course* timetable, int hakjum, double goalGrade) {
    printf(_Format:"목표 성적: %.2lf₩n", goalGrade);

    double currentGrade = 0.0;
    double sum = 0.0;

    for (int i = 0; i < hakjum; i++) {
        sum += timetable[i].grade;
    }

    currentGrade = sum / hakjum;

    if (sum != 0.0) {
        printf(_Format:"현재 당신의 성적: %.2lf₩n", currentGrade);

        if (goalGrade > currentGrade) {
            printf(_Format:"목표 성적까지 %.2lf만큼 남았습니다.₩n", goalGrade - currentGrade);
            printf(_Format:"조금 더 열심히 공부하세요!₩n");
        }
        else if (goalGrade < currentGrade) {
            printf(_Format:"목표 성적보다 %.2lf만큼 높습니다.₩n", currentGrade - goalGrade);
            printf(_Format:"멋집니다!₩n");
        }
        else {
            printf(_Format:"목표성적과 현재 성적이 같습니다.₩n");
        }
    }
    else {
        printf(_Format:"현재 성적을 입력해주세요.₩n");
    }
}
```

(2)-6. 과목 별 성적 목록 함수

- 입출력

  - 입력: 없음
  - 출력: 과목 별 성적

- 사용자가 입력 한 과목 별 성적을 수업명, 성적을 순서대로 목록으로 보여줌.

- 적용된 배운 내용: if문, 구조체, 함수, 반복문

- 코드 스크린샷

```
//과목 별 성적 보여주기
void displayGradeTimetable(struct Course* timetable, int hakjum) {
    printf(_Format: "₩n수업 별 성적:₩n----------------------₩n");
    for (int i = 0; i < hakjum; i++) {
        printf(_Format: "%s | 성적: %.2lf₩n", timetable[i].className, timetable[i].grade);
    }
}
```

(3) 투두 리스트

- 입출력

  - 입력: 없음
  - 출력: 투두리스트 함수 호출

- 설명: 사용자가 전체 메인 메뉴에서 3번을 클릭 시 투두리스트 함수가 호출됨.

- 적용된 배운 내용: if문, 함수

- 코드 스크린샷

```
//투두리스트 프로그램 작성
else if (num == 3) {
    clrscr();
    manageTodoList(todoList);
}
```

(3)-1 투두 리스트 함수

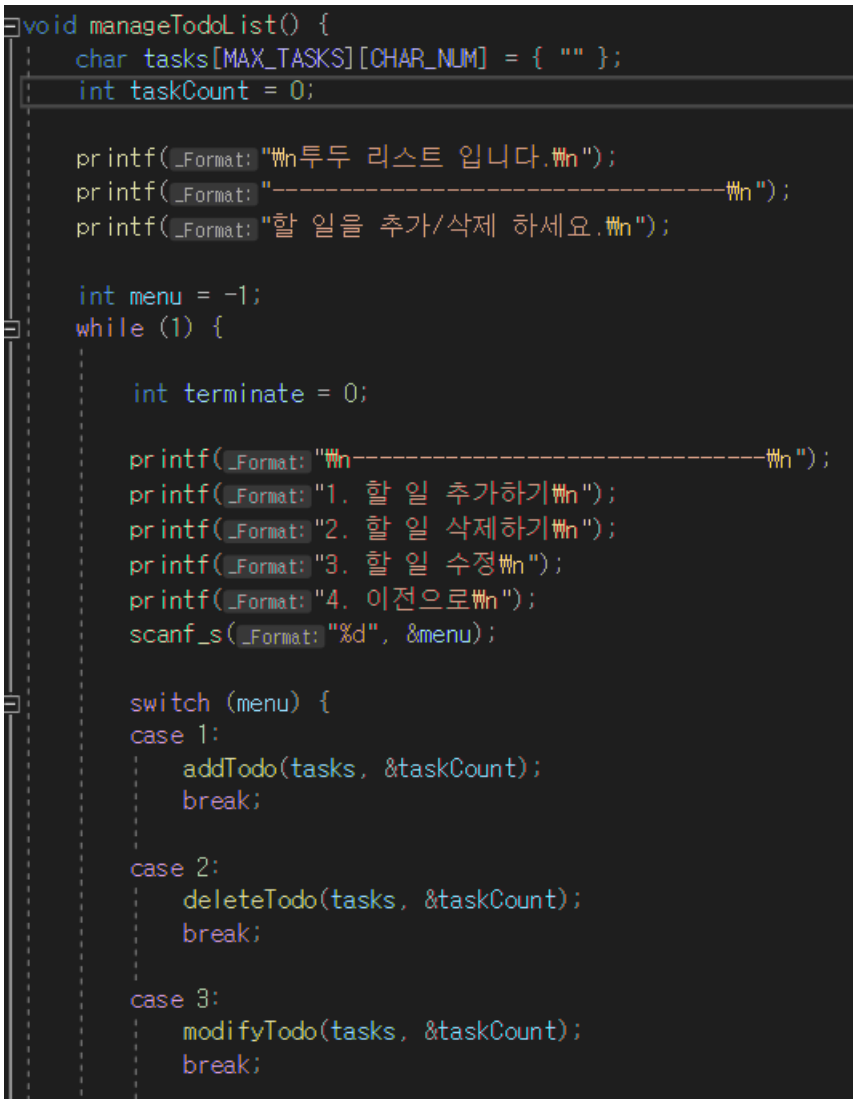
- 입출력

  - 입력: 투두리스트 메뉴 번호
  - 출력: 해당 메뉴 번호에 해당하는 함수 호출

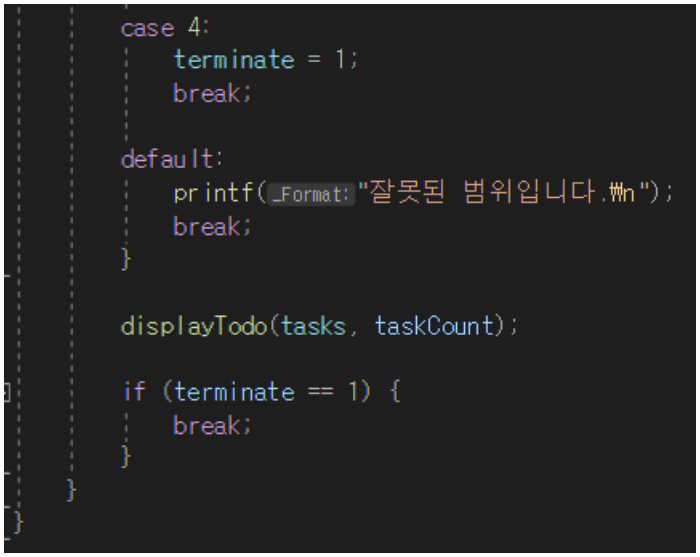
- 설명: 투두리스트 함수. 사용자는 원하는 기능의 번호를 입력하여 해당 번호의 서비스를 이용 할 수 있음. 할 일 추가/할 일 삭제/ 할 일 수정/ 이전으로 선택지를 고를 수 있음

- 적용된 배운 내용: 배열, while문, 함수, switch문

- 코드 스크린샷



```
void manageTodoList() {
    char tasks[MAX_TASKS][CHAR_NUM] = { "" };
    int taskCount = 0;

    printf(_Format: "₩n투두 리스트 입니다.₩n");
    printf(_Format: "--------------------------------₩n");
    printf(_Format: "할 일을 추가/삭제 하세요.₩n");

    int menu = -1;
    while (1) {

        int terminate = 0;

        printf(_Format: "₩n------------------------------₩n");
        printf(_Format: "1. 할 일 추가하기₩n");
        printf(_Format: "2. 할 일 삭제하기₩n");
        printf(_Format: "3. 할 일 수정₩n");
        printf(_Format: "4. 이전으로₩n");
        scanf_s(_Format: "%d", &menu);

        switch (menu) {
        case 1:
            addTodo(tasks, &taskCount);
            break;

        case 2:
            deleteTodo(tasks, &taskCount);
            break;

        case 3:
            modifyTodo(tasks, &taskCount);
            break;
```



(3) 할 일 추가 함수

- 입출력

- 입력: 할 일 명, 할 일의 마감 날짜
- 출력: 할일이 저장되었다는 메시지

- 설명: 사용자는 할 일 명과 할 일의 마감 날짜를 입력 할 수 있음. 마감 날짜의 형식이 맞지 않은 경우(yyyy-mm-dd의 형식으로 입력되지 않은 경우)에는 경고메시지가 출력되고 이전 화면으로 돌아감. 할일이 입력된 후에는 taskCount가 증가하여 계속 할 일을 추가할 수 있도록 함.

- 적용된 배운 내용: if문, 포인터, 함수, 구조체

- 코드 스크린샷

```c
void addTodo(struct Todo tasks[MAX_TASKS], int* taskCount) {
    fflush(_Stream: stdin);


    printf(_Format: "할 일을 입력하세요: ");
    scanf_s(_Format: "%s", tasks[*taskCount].task, (int)sizeof(tasks[*taskCount].task));

    printf(_Format: "마감 날짜를 입력하세요(yyyy-mm-dd): ");
    scanf_s(_Format: "%s", tasks[*taskCount].deadline, (int)sizeof(tasks[*taskCount].deadline));

    int year, month, day;
    if (sscanf_s(_Buffer: tasks[*taskCount].deadline, _Format: "%d-%d-%d", &year, &month, &day) != 3) {
        printf(_Format: "날짜 형식이 잘못되었습니다. 다시 입력해주세요.₩n₩n");
        return;
    }


    // 디데이 계산
    int dday_days = calculateDday(tasks[*taskCount].deadline);

    if (dday_days < 0) {
        printf(_Format: "지난 날짜는 입력할 수 없습니다. 다시 입력해주세요.₩n₩n");
        return;
    }
    printf(_Format: "%d", dday_days);
    printf(_Format: "할 일 ₩"%s₩" 가 저장되었습니다.₩n₩n", tasks[*taskCount].task);

    (*taskCount)++;
}
```

(3) 할 일 삭제 함수

- 입출력

  - 입력: 삭제할 할 일의 번호
  - 출력: 삭제 되었다는 메시지

- 설명: 할 일을 삭제 할 수 있음. 사용자가 입력한 번호가 범위에 맞지 않은 경우 경고 메시지가 표시 됨. 범위가 맞는 경우에만 삭제가 진행됨. 삭제 시 taskCount 변수가 감소하여 할 일을 계속 추가할 수 있도록 함.

- 적용된 배운 내용: if문, 함수, for문, 구조체

- 코드 스크린샷

```
void deleteTodo(struct Todo tasks[MAX_TASKS], int* taskCount) {
    int delIndex = -1;

    printf(_Format: "삭제할 할 일의 번호를 입력하세요(1부터 시작): ");
    scanf_s(_Format: "%d", &delIndex);

    if (delIndex > *taskCount || delIndex <= 0) {
        printf(_Format: "삭제 범위가 벗어났습니다.₩n");
    }
    else {
        printf(_Format: "%d. %s : 할 일을 삭제합니다.₩n", delIndex, tasks[delIndex - 1].task);

        for (int i = delIndex - 1; i < *taskCount - 1; i++) {
            tasks[i] = tasks[i + 1];
        }

        (*taskCount)--;
    }
}
```

(3) 할 일 수정 함수

- 입출력

  - 입력: 수정할 할 일의 번호, 수정할 내용
  - 출력: 수정되었다는 메시지

- 설명: 기존에 입력 했던 할 일의 명을 수정할 수 있음. 수정범위가 벗어난 경우 경고 메시지를 출력 함.

- 적용된 배운 내용: if문, 포인터, 구조체

- 코드 스크린샷

```
void modifyTodo(struct Todo tasks[MAX_TASKS], int* taskCount) {
    int changeIndex = -1;

    printf(_Format: "수정할 할 일의 번호를 입력하세요. (1부터 시작): ");
    scanf_s(_Format: "%d", &changeIndex);

    if (changeIndex > *taskCount || changeIndex <= 0) {
        printf(_Format: "수정 범위가 벗어났습니다.₩n");
    }
    else {
        printf(_Format: "%d. %s : 할 일을 수정합니다.₩n", changeIndex, tasks[changeIndex - 1].task);
        printf(_Format: "수정할 내용을 입력하세요₩n");
        scanf_s(_Format: "%s", tasks[changeIndex - 1].task, (int)sizeof(tasks[changeIndex - 1].task));
        printf(_Format: "%d번 할 일이 %s으로 수정되었습니다.₩n", changeIndex, tasks[changeIndex - 1].task);
    }
}
```

(3) 투두 리스트 목록 함수

- 입출력

  - 입력: 없음
  - 출력: 투두리스트 목록

- 설명: 사용자는 할 일의 목록을 할 일 명, 디데이 순으로 확인 할 수 있음. 디데이에 따라서 출력메시지가 다르게 나옴.

- 적용된 배운 내용: if문, 함수, for문

- 코드 스크린샷

```
void displayTodo(struct Todo tasks[MAX_TASKS], int taskCount) {
    printf(_Format: "₩n투두 리스트 입니다.₩n");
    printf(_Format: "--------------------------------₩n");

    for (int i = 0; i < taskCount; i++) {
        int dday = calculateDday(tasks[i].deadline);

        if (dday > 0) {
            printf(_Format: "%d. %s (디데이: %d일)₩n", i + 1, tasks[i].task, dday);
        }
        else if (dday == 0) {
            printf(_Format: "%d. %s (오늘까지입니다!)₩n", i + 1, tasks[i].task);
        }
        else {
            printf(_Format: "%d. %s (마감일이 지났습니다.)₩n", i + 1, tasks[i].task);
        }
    }
    printf(_Format: "₩n");
}
```

(3) 디데이 계산 함수

- 입출력

  - 입력: addTodo함수에서 입력 했던 마감기한
  - 출력: 디데이

- 설명: 현재 날짜와 addTodo 함수에서 사용자가 입력 했던 마감 기한의 남은 날짜를 계산해서 반환하는 함수. 시간 정보를 초기화 해서 날짜 정보만을 비교하도록 함. C 라이브러리인 "time.h"를 이용하였음.

- 적용된 배운 내용: if문, 함수, 구조체

- 코드 스크린샷

```c
int calculateDday(const char* deadline) {
    struct tm current_tm = { 0 };
    time_t current_time = time(_Time: NULL);
    localtime_s(&current_tm, &current_time);

    // 현재 날짜의 시간 정보 초기화
    current_tm.tm_hour = 0;
    current_tm.tm_min = 0;
    current_tm.tm_sec = 0;

    // 마감 날짜 설정
    struct tm deadline_tm = { 0 };
    int year, month, day;
    if (sscanf_s(_Buffer: deadline, _Format: "%d-%d-%d", &year, &month, &day) == 3) {
        deadline_tm.tm_year = year - 1900; // tm_year는 1900년을 기준으로 하므로 1900을 빼줍니다.
        deadline_tm.tm_mon = month - 1;    // tm_mon은 0부터 시작하므로 1을 빼줍니다.
        deadline_tm.tm_mday = day;
    }

    // 마감 날짜의 시간 정보 초기화
    deadline_tm.tm_hour = 0;
    deadline_tm.tm_min = 0;
    deadline_tm.tm_sec = 0;

    time_t current_day_start = mktime(&current_tm);
    time_t deadline_day_start = mktime(&deadline_tm);

    // 디데이 계산
    double dday_seconds = difftime(_Time1: deadline_day_start, _Time2: current_day_start);
    int dday_days = (int)(dday_seconds / (60 * 60 * 24));

    return dday_days;
}
```

4. 테스트 결과

(1) 초기 시간표 입력

- 설명: 초기 시간표가 입력되어 있지 않은 경우 시간표 입력화면이 표시됨

- 테스트 결과 스크린샷

```
시간표 관리 메뉴입니다.
몇학년 몇학기 수업인가요? (띄어쓰기로 구분해서 학년,학기 작성)
2 2
총 몇개의 수업을 들으시나요?3
듣는 수업의 수업명, 교수명, 장소명을 차례대로 입력하세요. (띄어쓰기로 구분)
c언어 김미수 ai융합대학
알고리즘
임형석 공7
역사문화자원의이해 조상현 ai융합대학

시간표:
----------------------
c언어 | 김미수 | ai융합대학
알고리즘 | 임형석 | 공7
역사문화자원의이해 | 조상현 | ai융합대학

1. 수업 추가
2. 수업 삭제
3. 이전으로
원하는 번호를 입력하세요.
```

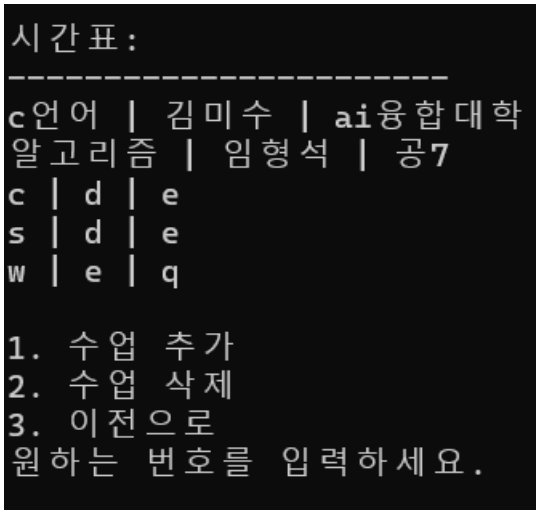
(2) 수업 추가

- 설명: 기존 시간표에서 수업을 추가할 수 있음.

- 아쉬운점, 보완할점: 기존에는 수업 추가 하면 수업이 추가 됐다는 메시지가 뜨고 수업이 추가된 시간표를 바로 보여줬었는데 오늘 테스트 하니까 이전으로 넘어가고 다시 시간표 메뉴로 들어와야지 추가된걸 확인할 수 있다는걸 깨달음. 이 점에 대해서는 사전에 검토를 한번 더 하지 않았던 것에 대해서 굉장히 큰 반성을 함..

- 테스트 결과 스크린샷

```
시간표:
----------------------
c언어 | 김미수 | ai융합대학
알고리즘 | 임형석 | 공7
c | d | e
s | d | e

1. 수업 추가
2. 수업 삭제
3. 이전으로
원하는 번호를 입력하세요.
1
추가할 수업의 수업명, 교수명, 장소명을 차례대로 입력하세요. (띄어쓰기로 구분
)
w e q
```

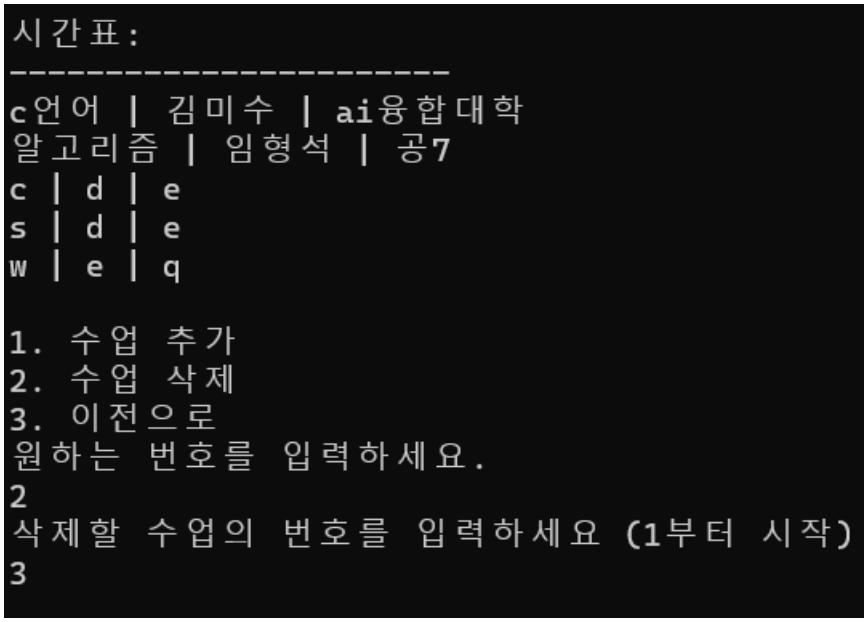


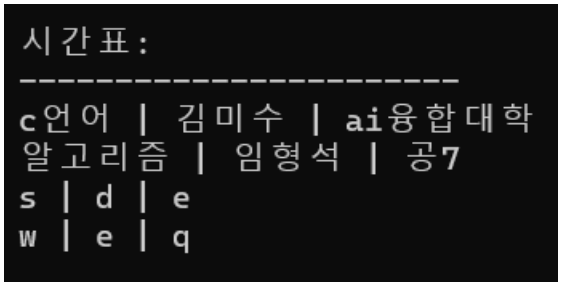
(3) 시간표 수업 삭제

- 설명: 수업 삭제를 선택해서 원하는 수업을 삭제함

- 아쉬운점, 보완할점: 수업 추가와 마찬가지로 이 부분도 삭제 후 바로 삭제되었다는 메시지가 출력되어야 하는데 그러지를 않고 이전화면으로 넘어가고 다시 시간표 메뉴로 와야지 삭제된걸 확인할 수 있다.. 이부분에 대해서도 반성하고 있다..

- 테스트 결과 스크린샷

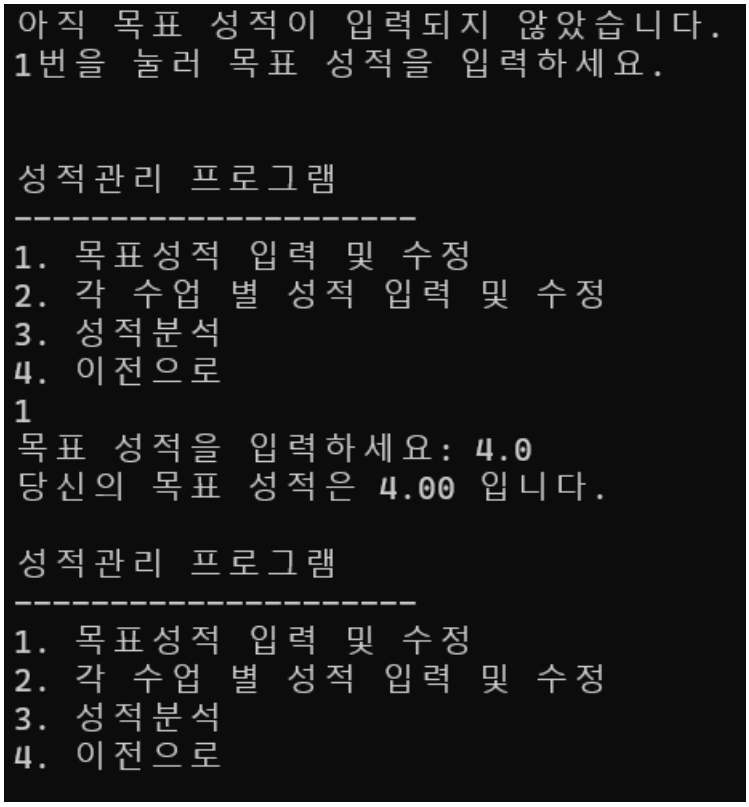




(4) 목표 성적 입력

- 설명: 초기 목표설정이 입력되어 있지 않은 경우 이전화면으로 넘어감

- 테스트 결과 스크린샷



(5) 각 수업 별 성적 입력

- 설명: 시간표가 입력되어 있지 않은 경우 경고 메시지 출력 후 이전 메뉴로 이동. 입력되어 있는 경우 과목 별 성적을 입력

- 테스트 결과 스크린샷

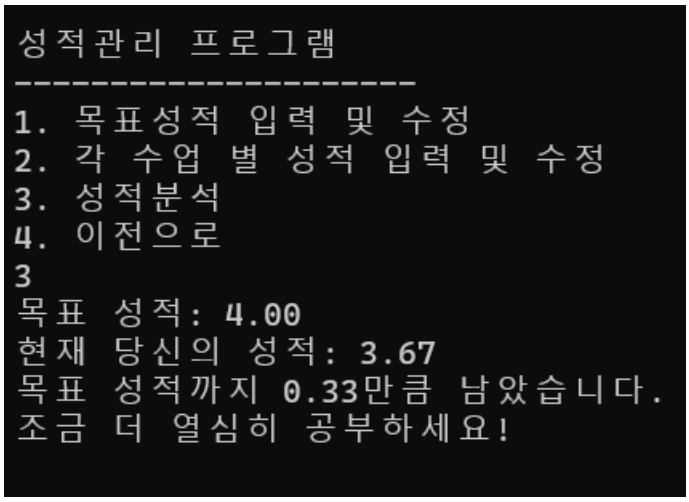
```
성적관리 프로그램
--------------------
1. 목표성적 입력 및 수정
2. 각 수업 별 성적 입력 및 수정
3. 성적분석
4. 이전으로
2
현재 수업이 없어 성적을 입력할 수 없습니다.

수업 별 성적:
----------------------


성적관리 프로그램
--------------------
1. 목표성적 입력 및 수정
2. 각 수업 별 성적 입력 및 수정
3. 성적분석
4. 이전으로
```

```
성적관리 프로그램
--------------------
1. 목표성적 입력 및 수정
2. 각 수업 별 성적 입력 및 수정
3. 성적분석
4. 이전으로
2
c언어 과목의 성적을 입력하세요:
4.0
알고리즘 과목의 성적을 입력하세요:
3.0
역사문화자원의이해 과목의 성적을 입력하세요:
4.0

수업 별 성적:
----------------------
c언어 | 성적: 4.00
알고리즘 | 성적: 3.00
역사문화자원의이해 | 성적: 4.00
```

(6) 성적 분석

- 설명: 목표 성적과 현재 성적을 비교한 결과를 보여줌

- 테스트 결과 스크린샷



(7) 할 일 추가

- 설명: 할 일을 입력하고 마감날짜를 입력한 결과를 보여줌

- 테스트 결과 스크린샷

```
투두 리스트 입니다.
----------------------------------


------------------------------
1. 할 일 추가하기
2. 할 일 삭제하기
3. 할 일 수정
4. 이전으로
1
할 일을 입력하세요: c언어과제
마감 날짜를 입력하세요(yyyy-mm-dd): 2023-12-24
할 일 "c언어과제" 가 저장되었습니다.


투두 리스트 입니다.
----------------------------------
1. c언어과제 (오늘까지입니다!)


------------------------------
1. 할 일 추가하기
2. 할 일 삭제하기
3. 할 일 수정
4. 이전으로
```

(8) 할 일 수정

- 설명: 할 일의 이름을 수정

- 테스트 결과 스크린샷

```
1. 할 일 추가하기
2. 할 일 삭제하기
3. 할 일 수정
4. 이전으로
3
수정할 할 일의 번호를 입력하세요. (1부터 시작): 1
1. c언어과제 : 할 일을 수정합니다.
수정할 내용을 입력하세요
c++과제
1번 할 일이 c++과제으로 수정되었습니다.

투두 리스트 입니다.
----------------------------------
1. c++과제 (오늘까지입니다!)


--------------------------------
```

(9) 할 일 삭제

- 설명: 기존 입력되어 있던 할 일을 삭제

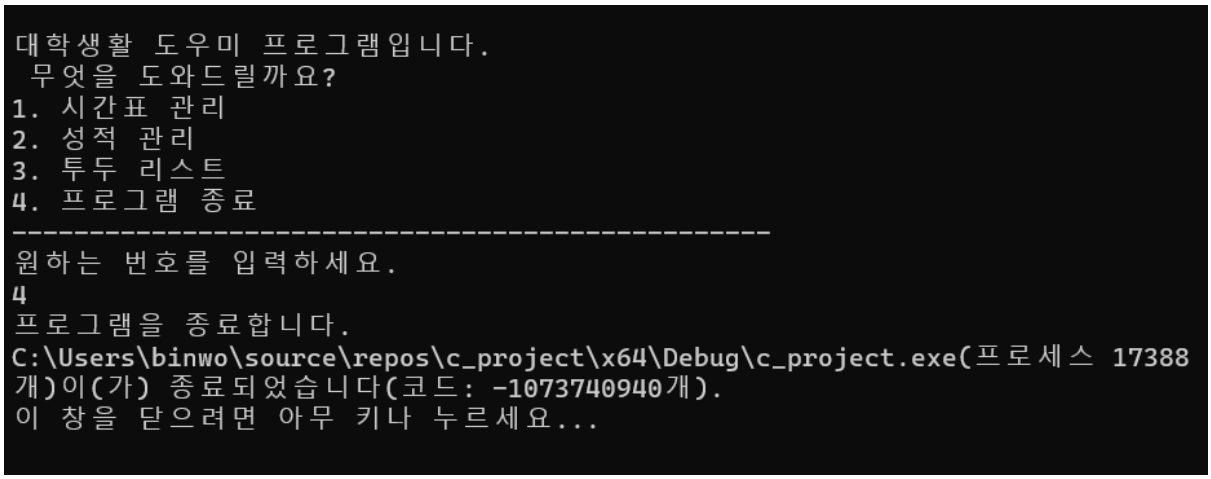
- 테스트 결과 스크린샷

```
----------------------------------
1. c++과제 (오늘까지입니다!)


--------------------------------
1. 할 일 추가하기
2. 할 일 삭제하기
3. 할 일 수정
4. 이전으로
2
삭제할 할 일의 번호를 입력하세요(1부터 시작): 1
1. c++과제 : 할 일을 삭제합니다.

투두 리스트 입니다.
----------------------------------


--------------------------------
1. 할 일 추가하기
2. 할 일 삭제하기
3. 할 일 수정
4. 이전으로
```

(10) 프로그램 종료

- 설명: 4번을 입력 시 프로그램이 종료 된다

- 테스트 결과 스크린샷



5. 계획 대비 변경 사항

1) 과제 관리 -> 투두리스트

- 이전: 기존에는 수업 과목 별 과제를 입력할 수 있게 하여 조금 더 사용자의 편의를 주었음. 수업이 삭제되면 그 수업안에 있는 과제들도 다 삭제가 될 수 있도록 구조체로 연결 할 의도였음.

- 이후: 과제관리에서 투두리스트로 기능 변경.

- 사유: 과제 관리 시 과제 구조체 배열을 동적 할당하는데 이 경우 구조체의 크기가 처음부터 정해지는게 아니고 0부터 시작해서 과제가 계속 늘어나면 하나씩 늘어나는 구조인데 코드를 짜도 계속 과제명을 입력하는 부분만 가면 프로그램이 몇번 깜박이다가 종료되는 현상이 발생함. 또한 구조체 멤버변수인 과제의 수를 처음 0으로 초기화하는데 어려움을 겪음. 어쩔 수 없이 수업 별 과제 입력이 아닌 그냥 할 일을 입력할 수 있는 기능으로 변경함. 디데이 기능은 여전히 유지.

2 디데이 기간이 지나면 자동 삭제 -> 마감기한이 지났다는 메시지

- 이전: 기존에는 디데이 기간이 지나면 할 일 목록에서 자동으로 삭제되게 하려고 했음

- 이후: 할 일 목록에서 마감기한이 지났다고 메시지가 출력되도록 하고 사용자가 직접 삭제하도록 함

- 사유: 사용자 편의상 할일의 마감 기한이 지났다고 자동으로 삭제가 되면 일정 관리를 하는데 불편함이 있을 거라고 판단함. 그래서 마감 기한이 지났다는 메시지를 보여주어 사용자가 일정을 더 쉽게 파악할 수 있게 하고 자신이 할 일을 삭제함으로써 좀더 할 일에 대한 인지를 주게 함.

6. 느낀점

배운점: 이 과제를 통해 수업 시간에 배웠던 내용을 활용할 수 있어서 좋았다. 기능 구현 시 생각이 안나는 부분에 대해서는 다시 수업 자료를 보게 되면서 꾸준히 c언어 문법에 대해서 복습 할 수 있었다. 또 내가 원하는 기능을 배웠던 걸 기반으로 구현하면서 뿌듯함을 느꼈다. 또 평소에 c언어에 대해서는 자세히 몰랐었던 것 같아 다시 알게되는 좋은 기회였던 것 같다. 예를 들어 헤더파일이나 c파일을 여러 개로 분리하여 조금 더 코드를 편리하게 관리할 수 있었던 건 이 수업을 통해서 처음 알게 된 사실이다.

아쉬운점, 보완활점: 코드에 예외처리를 안 한 부분이 많다. 예외처리를 안해서 문제가 되는 부분도 있어서 이 점에 대해선 더 보완해야 할 필요가 있다고 느꼈다. 또 초기 테스트 결과만 믿고 후에 완벽하게 모든 기능을 테스트 해보지 못한 부분이 정말 보완할 점이다. 초기에는 됐었는데 오늘 모든 기능을 하나하나 구현해 보니 제대로 되지 않는 부분들이 있어서 굉장히 아쉬웠다.

교수님 한학기동안 수업하시느라 정말 수고 많으셨습니다!