C프로그래밍 및 실습

StudyPlan-30

최종 보고서

제출일자: 2023.12.21
제출자명: 유소연
제출자학번: 234404

1. 프로젝트 목표

1) 배경 및 필요성

학창 시절 열심히 공부해 대학교에 왔지만, 대학 생활의 자유로움으로 인해 고등학교 때만큼 체계적인 공부 습관을 유지하지 못함. 학년이 올라감에 따라 학점도 중요해지는 시점에서 다시 공부 습관을 잡아야 할 필요가 있음. 이 문제를 해결하기 위해 자동으로 학습 플랜을 짜주는 프로그램이 필요함.

2) 프로젝트 목표

학습 계획을 세워 계획대로 공부에만 집중할 수 있도록 하는 것을 목표로 함.

3) 차별점

기존 프로그램들은 직접 계획을 세워야 함. 이는 계획 세우기에만 몰두하거나, 무리한 계획을 세울 문제가 있음. 우리는 계획을 세우고 달성하지 못한 계획과 부족했던 점을 입력해 놓으면 다시 추가적인 계획을 세우는 것에 기존 프로그램과 차별점이 있음.

2. 기능 계획

1) 사용자에게 작업 요청 받기

- 1. 30일용 계획 세우기, 2. 남은 D-day 확인하기, 3. 부족 개념 입력, 4. 퀴즈 풀고 일기 쓰기, 5. 일정 미루기, 6. 종료 의 메뉴를 제작하여 사용자가 기능을 골라 선택하는 기능

2) 30일용 계획 세우기 (① 입력 경우)

- 30일을 기준으로 각 과목의 [n회독, 문제 풀이&오답, 백지 테스트]가 한 세트가 되어 로테이션이 돌면서 계획을 세우는 기능

3) 남은 D-day 확인하기 (② 입력 경우)

- 오늘의 D-day를 입력하면 각 과목의 시험 날짜와 차이를 출력하여 과목별 남은 D-day를 확인할 수 있는 기능

4) 부족 개념 입력하기 (③ 입력 경우)

- 부족했던 개념을 입력하면서 한 번 더 공부할 수 있고, 퀴즈로 확인하기 위해 개념을 입력하는 기능

5) 퀴즈 풀고 일기 쓰기 (④ 입력 경우)

- 부족했던 개념을 입력해 놓으면, 퀴즈 형식으로 출력되는 기능, 입력했던 퀴즈를 전부 수행하면 비밀 번호를 입력하여 일기장을 쓸 수 있는 기능

6) 일정 미루기 (⑤ 입력 경우)

- 사용자에게 오늘 할 일의 수행 여부를 묻고, 수행하지 않았다면 할 일을 미루는 기능

3. 기능 구현

1) 사용자에게 작업 요청 받기

- 사용자가 수행할 번호를 입력하면, 번호에 맞는 기능이 출력
- 번호를 입력하면 해당 기능이 실행되도록 출력
- printMenu 함수와 do – while 문, switch 문을 이용
- 코드 스크린샷

```c
// 메뉴 출력 함수
void printMenu() {
    printf("₩n[메뉴]₩n");
    printf("1. 30일용 계획 세우기₩n");
    printf("2. 남은 D-day 확인하기₩n");
    printf("3. 부족 개념 입력₩n");
    printf("4. 퀴즈 풀고 일기 쓰기₩n");
    printf("5. 오늘의 할 일 수행 여부 입력하기(필수)₩n");
    printf("6. 종료₩n");
    printf("메뉴를 선택하세요: ");
}
```

```c
// 메뉴 선택 및 처리
int menuChoice;
char currentDate[20];
do {

    printMenu();
    scanf_s("%d", &menuChoice);
```

(좌) 메뉴 출력 함수 , (우) main 함수에서의 메뉴 출력 실행 코드

- 메뉴 실행 코드는 (2)~ 코드에서 설명

2) 30일용 계획 세우기

- 시험 보는 과목의 개수와 이름을 입력하면, Day30 까지의 계획을 출력
- 앞서 설명한 입력 값을 넣으면 [회독, 문제풀이&오답, 백지테스트] 1세트가 과목 하나당 로테이션이 돌아가도록 Day30까지의 계획을 출력함.
- for 문과 switch 문 이용하였고, 구조체를 사용하여 동적으로 할당된 Subject 배열에서 subjectIndex에 해당하는 과목의 이름에 접근함.
- 코드 스크린샷

```c
// 동적으로 할당된 Subject 구조체 배열
struct Subject* subjects = (struct Subject*)malloc(subjectCount * sizeof(struct Subject));
if (subjects == NULL) {
    printf("메모리 할당 오류₩n");
    return -1;
}
```

```c
case 1:
    totalDays = MAX_DAYS;
    printf("₩n[스터디 계획]₩n");

    // 로테이션 계획 출력
    for (int day = 1; day <= totalDays; ++day) {
        int subjectIndex = (day - 1) / 3 % subjectCount;
        int rotationCount = ((day - 1) / (3 * subjectCount)) + 1;

        printf("%d 일차 계획 : %s ", day, subjects[subjectIndex].name);

        switch ((day - 1) % 3) {
        case 0:
            printf("%d회독₩n", rotationCount);
            break;
        case 1:
            printf("문제풀이&오답 체크₩n");
            break;
        case 2:
            printf("백지 테스트₩n");
            break;
        }

        if (day % 3 == 0) {
            printf("₩n");
        }
    }
    break;
```

3) 남은 D-day 확인하기

- 오늘의 날짜를 입력하면 각 과목별 남은 D-day를 출력
- 시험까지 과목마다 얼마나 남았는지 확인하고, 더 가까운 시험을 계획 외에 추가로 공부하거나 미루는 등 사용자가 능동적으로 사용할 수 있음.
- calcDaysLeft 함수를 이용하여 두 날짜 사이의 남은 일수를 계산, 현재 날짜와 시험 날짜를 받아와 mktime 함수를 사용하여 time_t 형식으로 변환한 후, 두 날짜 간의 차이를 일 단위로 계산함. 외에도 배열, 구조체를 이용하여 과목 이름과 시간에 접근함.
- 코드 스크린샷

```
case 2:
    // 남은 D-day 확인하기
    printf("오늘의 날짜를 입력하세요 (월일 형식, ex: 12월11일): ");
    printf("₩n");
    scanf_s("%s", currentDate, 20);

    int daysElapsed = calcDaysLeft(currentDate, examStartDate);

    for (int i = 0; i < subjectCount; ++i) {
        int daysLeft = calcDaysLeft(examStartDate, subjects[i].examDate);

        printf("%s - %d일 남았습니다.₩n", subjects[i].name, daysElapsed + daysLeft);
    }
    printf("₩n");
    break;
```

```
//남은 D-day를 계산하는 기능
int calcDaysLeft(const char* currentDate, const char* examDate) {
    struct tm currentTime, examTime;
    memset(&currentTime, 0, sizeof(struct tm));
    memset(&examTime, 0, sizeof(struct tm));

    // 현재 날짜 설정
    sscanf_s(currentDate, "%d월%d일", &currentTime.tm_mon, &currentTime.tm_mday);
    currentTime.tm_mon -= 1;  // tm_mon은 0부터 시작하므로 1을 빼줍니다.
    currentTime.tm_year = 2023 - 1900;  // 현재 연도 설정 (2023년 기준)

    // 시험 날짜 설정
    sscanf_s(examDate, "%d월%d일", &examTime.tm_mon, &examTime.tm_mday);
    examTime.tm_mon -= 1;  // tm_mon은 0부터 시작하므로 1을 빼줍니다.
    examTime.tm_year = 2023 - 1900;  // 현재 연도 설정 (2023년 기준)

    time_t currentTimestamp = mktime(&currentTime);
    time_t examTimestamp = mktime(&examTime);

    // 두 날짜 간의 차이 계산
    int daysLeft = (examTimestamp - currentTimestamp) / (60 * 60 * 24);
    return daysLeft;
}
```

4) 부족한 개념 입력하기

- 부족한 개념의 단어와 정의를 입력하면 출력 없이 입력 값 저장
- 프로그램 초반에 입력한 과목들을 돌아가며 부족한 개념을 묻고, 입력 받은 개

념과 정의를 입력함.

- 포인터를 이용하여 inputConcepts 함수에 입력 받은 단어와 정의를 전달함. 함수에서는 for문을 통해 과목을 돌며, 입력 받은 단어와 정의는 앞서 할당한 2차원 문자열 배열 concepts과 definitions에 저장함. 입력을 종료하는 경우를 위해 if문을 사용함.

- 코드 스크린샷

```
case 3:
    // 부족 개념 입력
    inputConcepts(subjects, subjectCount);
    break;
```

```
void inputConcepts(struct Subject* subjects, int subjectCount) {
    for (int i = 0; i < subjectCount; ++i) {
        printf("%s 과목의 부족했던 개념을 입력해주세요 (입력을 마치려면 q를 입력하세요):₩n", subjects[i].name);

        for (int j = 0; j < MAX_CONCEPTS; ++j) {
            printf("단어: ");
            if (scanf_s("%s", subjects[i].concepts[j], 50) != 1 || subjects[i].concepts[j][0] == 'q') {
                break;
            }

            printf("정의: ");
            scanf_s("%s", subjects[i].definitions[j], 100);
        }
    }
}
```

5) 퀴즈 풀고 일기 쓰기

- 퀴즈를 풀 과목의 번호를 입력하면 그 과목의 단어를 보여주고 정의를 묻는 퀴즈 출력

  - 퀴즈를 출력한 뒤 동일 문제를 다시 풀지 묻고, 이를 종료하면 다시 과목의 번호를 물어 퀴즈 반복이 돌아감.

- 퀴즈 종료를 입력하면 일기장을 출력

  - 앞서 출력된 모든 퀴즈 관련 선택을 종료하면, 일기장을 열지 묻고 비밀번호를 입력받음. 비밀번호가 맞으면 일기장이 열림.

- do-while 문을 이용해 과목을 입력받고, if문을 통해 사용자의 입력 값에 대한 경우를 만들어놓음. 사용자가 과목의 번호를 입력하면 quizConcepts 함수가 호

출되어 do-while 문와 중첩 for문을 이용하여 퀴즈를 출력함. 퀴즈의 출제가 끝나면 allCorrect 값을 반환. 퀴즈 후에는 일기장 여부를 사용자에게 묻고 파일 입출력을 통해 작성. EXIT를 입력하면 일기 종료.

- 코드 스크린샷

```
case 4:
    // 부족한 개념 확인하고 일기 쓰기
    do {
        // 사용자에게 과목을 선택하도록 물어봄
        printf("과목을 선택하세요 (1부터 %d까지, 0: 종료): ", subjectCount);
        int selectedSubjectIndex;
        scanf_s("%d", &selectedSubjectIndex);

        if (selectedSubjectIndex == 0) {
            break;
        }
        if (selectedSubjectIndex >= 1 && selectedSubjectIndex <= subjectCount) {
            allCorrect = quizConcepts(subjects, selectedSubjectIndex - 1);
        }
        else {
            printf("잘못된 선택입니다.₩n");
        }

    } while (1);

    //openDiary 함수 호출
    printf("일기장을 여시겠습니까? (1: 예, 0: 아니오): ");
    int openDiaryChoice;
    scanf_s("%d", &openDiaryChoice);

    if (openDiaryChoice == 1) {
        openDiary();
    }

    break;
```

<main 함수에서의 실행 코드>

```
int quizConcepts(struct Subject* subjects, int subjectIndex) {
    int allCorrect = 1;
    int additionalQuizzes;

    do {
        printf("%s 과목의 부족했던 개념을 퀴즈로 확인합니다:₩n", subjects[subjectIndex].name);

        for (int i = 0; i < MAX_CONCEPTS; ++i) {
            if (subjects[subjectIndex].concepts[i][0] == 'q') {
                break;
            }

            // 사용자에게 단어를 물어보고 정답 체크
            printf("단어: %s₩n", subjects[subjectIndex].concepts[i]);
            printf("정의: ");
            char userDefinition[100];
            scanf_s("%s", userDefinition, sizeof(userDefinition));

            if (strcmp(userDefinition, subjects[subjectIndex].definitions[i]) == 0) {
                printf("정답입니다!₩n");
            }
            else {
                printf("틀렸습니다. 정답은: %s₩n", subjects[subjectIndex].definitions[i]);
                allCorrect = 0;
            }
        }

        // 추가 퀴즈 여부 물어보기
        printf("동일한 문제를 다시 풀어보시겠습니까? (1: 예, 0: 아니오): ");
        scanf_s("%d", &additionalQuizzes);

    } while (additionalQuizzes == 1);

    return allCorrect;
}
```

<퀴즈 출력 함수>

```c
// 비밀 일기장 열기 함수
void openDiary() {
    char password[20];
    char line[MAX];
    char contents[MAX];

    printf("  ~ 일기 쓰기 ~ ₩n");
    printf("비밀번호를 입력하세요 (최대 20자리): ");
    scanf_s("%s", password, sizeof(password));

    printf("₩n₩n=== 비밀번호 확인 중===₩n₩n");

    if (strcmp(password, "project") == 0) {
        printf("=== 비밀번호 확인 완료.===₩n₩n");

        // 파일 열기
        FILE* file;
        if (fopen_s(&file, "diary.txt", "a+") != 0) {
            printf("파일 열기 실패₩n");
            return;
        }

        // 파일 내용 출력
        while (fgets(line, MAX, file) != NULL) {
            printf("%s", line);
        }

        printf("₩n₩n내용을 계속 작성하세요! 종료하려면 EXIT를 입력하세요.₩n₩n");
```

<일기장 작성 함수>

```c
        // 내용 입력 및 파일에 쓰기
        while (1) {
            scanf_s(" %[^₩n]", contents, sizeof(contents));
            getchar();

            if (strcmp(contents, "EXIT") == 0) {
                printf("비밀일기 입력을 종료합니다.₩n₩n");
                break;
            }

            fputs(contents, file);
            fputs("₩n", file);
        }

        // 파일 닫기
        fclose(file);
    }
    else {
        printf("=== 비밀번호가 틀렸어요. ===₩n₩n");
        return;
    }
}
```

6) 일정 미루기

- 오늘 할 일의 수행 여부를 입력하면 할 일이 미뤄졌는지 여부와 내일 해야 할 일 출력

- 30부터 1까지 스택에 넣고, 할 일을 수행했다면 맨 위의 스택을 삭제 후 내일 해야 할 일은 그 다음 값으로 출력하고 수행하지 못했다면 맨 위의 값을 내일

해야 할 일로 출력함.

- handleTodayTask 함수에 포인터로 주소를 전달하여 함수 호출, 포인터를 매개변수로 사용하여 스택과 인덱스에 접근함. If문을 이용하여 사용자에게 할 일 수행 여부를 물음. 각 스택 함수로는 스택을 초기화하고 값을 추가, 삭제하는 함수와 할 일을 미뤘을 때 원래 스택을 그대로 출력하기 위해 Top 함수를 지정함.

- 코드 스크린샷

```c
case 5:
    // 일정 미루기
    handleTodayTask(&stack, &subjectIndex, subjects, subjectCount);
    break;
```

<main 함수에서의 실행 코드>

```c
//할 일 수행 여부 확인 함수
void handleTodayTask(SubjectStack* stack, int* subjectIndex, const struct Subject* subjects, int subjectCount) {
    // 오늘의 할 일 수행 여부 입력 받기
    printf("오늘 할 일을 수행하셨나요? (1: 예, 0: 아니오): ");
    int todayTaskCompleted;
    scanf_s("%d", &todayTaskCompleted);

    if (todayTaskCompleted) {
        // 오늘의 할 일 수행 완료 시 스택에서 맨 위 숫자(pop)를 가져와 출력
        int poppedValue;

        if (Pop(stack, &poppedValue) == 0) {
            printf("오늘 할 일을 수행하셨습니다. 오늘차 계획을 삭제합니다. (내일 해야할 일 : %d일차 계획)₩n", poppedValue + 1, poppedValue + 1);
        }
        else {
            printf("남은 할 일이 없습니다.₩n");
        }
    }

    else {
        // 오늘의 할 일 수행하지 않았을 때 스택은 건드리지 않고 현재 맨 위 숫자(top)만 출력
        int topValue;
        if (Top(stack, &topValue) == 0) {
            printf("오늘 할 일을 수행하지 않으셨습니다. 내일로 미뤄집니다. (내일 해야 할 일 : %d일차 계획)₩n", topValue, topValue);
        }
        else {
            printf("남은 할 일이 없습니다.₩n");
        }
    }
}
```

<수행 함수>

```c
int Initialize(SubjectStack* s, int max) {
    s->ptr = 0;
    if ((s->stk = calloc(max, sizeof(int))) == NULL) {
        s->max = 0;
        return -1;
    }
    s->max = max;
    return 0;
}

void Push(SubjectStack* s, int x) {
    if (s->ptr >= s->max) {
        // 스택이 가득 찼을 때의 처리
        printf("스택이 전부 찼습니다.₩n");
        return;
    }
    s->stk[s->ptr++] = x;
}
```

```c
int Pop(SubjectStack* s, int* x) {
    if (s->ptr <= 0) {
        // 스택이 비었을 때의 처리
        printf("스택이 비었습니다.₩n");
        return -1;
    }
    *x = s->stk[--s->ptr];
    return 0;
}

int Top(const SubjectStack* s, int* x) {
    if (s->ptr <= 0) {
        // 스택이 비었을 때의 처리
        printf("스택이 비었습니다.₩n");
        return -1;
    }
    *x = s->stk[s->ptr - 1];
    return 0;
}
```

<스택 관련 함수>

4. 테스트 결과

(1) 사용자에게 작업 요청 받기

- 함수와 do-while 문을 통해 메뉴 함수를 지정하고 해당 메뉴를 출력하는 기능 작성

- 테스트 결과 스크린샷(기능 스크린샷은 (2)~에서 설명)

```
[메뉴]
1. 30일용 계획 세우기
2. 오늘의 D-day 계획 확인하기
3. 부족 개념 입력
4. 퀴즈 풀고 일기 쓰기
5. 오늘 일정을 미루시겠습니까? (Y/N)
6. 종료
메뉴를 선택하세요: 1
```

(2) 30일용 계획 세우기

-반복문을 통해 Day30 까지의 계획 출력 기능 작성

- 테스트 결과 스크린샷

```
시험을 보는 과목의 개수를 입력하세요: 2
시험 기간의 시작 날짜를 입력하세요 (월일 형식, ex: 12월11일): 12월28일
과목 1의 이름을 입력하세요: 이산수학
과목 1의 시험 날짜를 입력하세요 (월일 형식, ex: 12월18일): 12월30일
과목 2의 이름을 입력하세요: 논리회로
과목 2의 시험 날짜를 입력하세요 (월일 형식, ex: 12월18일): 12월29일
```

```
[스터디 계획]
1 일차 계획 : 이산수학 1회독
2 일차 계획 : 이산수학 문제풀이&오답 체크
3 일차 계획 : 이산수학 백지 테스트

4 일차 계획 : 논리회로 1회독
5 일차 계획 : 논리회로 문제풀이&오답 체크
6 일차 계획 : 논리회로 백지 테스트

7 일차 계획 : 이산수학 2회독
8 일차 계획 : 이산수학 문제풀이&오답 체크
9 일차 계획 : 이산수학 백지 테스트

10 일차 계획 : 논리회로 2회독
11 일차 계획 : 논리회로 문제풀이&오답 체크
12 일차 계획 : 논리회로 백지 테스트

13 일차 계획 : 이산수학 3회독
14 일차 계획 : 이산수학 문제풀이&오답 체크
15 일차 계획 : 이산수학 백지 테스트

16 일차 계획 : 논리회로 3회독
17 일차 계획 : 논리회로 문제풀이&오답 체크
18 일차 계획 : 논리회로 백지 테스트

19 일차 계획 : 이산수학 4회독
20 일차 계획 : 이산수학 문제풀이&오답 체크
21 일차 계획 : 이산수학 백지 테스트

22 일차 계획 : 논리회로 4회독
23 일차 계획 : 논리회로 문제풀이&오답 체크
24 일차 계획 : 논리회로 백지 테스트

25 일차 계획 : 이산수학 5회독
26 일차 계획 : 이산수학 문제풀이&오답 체크
27 일차 계획 : 이산수학 백지 테스트

28 일차 계획 : 논리회로 5회독
29 일차 계획 : 논리회로 문제풀이&오답 체크
30 일차 계획 : 논리회로 백지 테스트
```

(3) 남은 Dday 확인하기

- 오늘의 날짜를 입력 받아 과목별 시험 날짜와의 차이를 구해 for문을 이용하여 남은 Dday 출력하는 기능 작성

- 테스트 결과 스크린샷

```
시험을 보는 과목의 개수를 입력하세요: 2
시험 기간의 시작 날짜를 입력하세요 (월일 형식, ex: 12월11일): 12월28일
과목 1의 이름을 입력하세요: 이산수학
과목 1의 시험 날짜를 입력하세요 (월일 형식, ex: 12월18일): 12월30일
과목 2의 이름을 입력하세요: 논리회로
과목 2의 시험 날짜를 입력하세요 (월일 형식, ex: 12월18일): 12월29일
```

<코드 초반 입력한 날짜>

```
메뉴를 선택하세요: 2
오늘의 날짜를 입력하세요 (월일 형식, ex: 12월11일):
12월21일
이산수학 - 9일 남았습니다.
논리회로 - 8일 남았습니다.
```

(4) 부족한 개념 입력

- for문을 통해 프로그램 초반에 입력한 과목들을 돌아가며 부족한 개념을 묻고, 입력 받은 개념들을 저장하는 기능 작성

- 테스트 결과 스크린샷

```
메뉴를 선택하세요: 3
이산수학 과목의 부족했던 개념을 입력해주세요 (입력을 마치려면 q를 입력하세요):
단어: 이산수학은
정의: 어렵다
단어: q
논리회로 과목의 부족했던 개념을 입력해주세요 (입력을 마치려면 q를 입력하세요):
단어: 논리회로는
정의: 더어렵다
단어: q
```

(5) 퀴즈 풀고 일기 쓰기

- 사용자가 선택한 과목의 입력해둔 단어-정의 형식을 단어를 보여주며 정의를 묻는 방식으로 출력하는 기능 작성

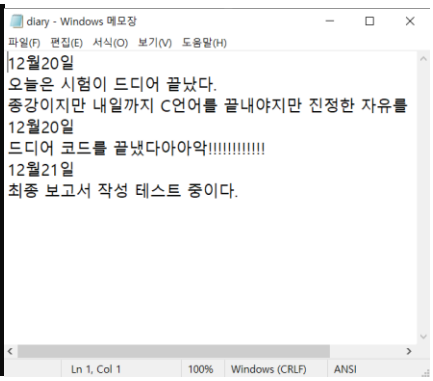
- 테스트 결과 스크린샷

```
메뉴를 선택하세요: 4
과목을 선택하세요 (1부터 2까지, 0: 종료): 1
이산수학 과목의 부족했던 개념을 퀴즈로 확인합니다:
단어: 이산수학
정의: 수학
정답입니다!
동일한 문제를 다시 풀어보시겠습니까? (1: 예, 0: 아니오): 0
과목을 선택하세요 (1부터 2까지, 0: 종료): 2
논리회로 과목의 부족했던 개념을 퀴즈로 확인합니다:
단어: 논리회로
정의: 논리
틀렸습니다. 정답은: q
동일한 문제를 다시 풀어보시겠습니까? (1: 예, 0: 아니오): 1
논리회로 과목의 부족했던 개념을 퀴즈로 확인합니다:
단어: 논리회로
정의: q
정답입니다!
동일한 문제를 다시 풀어보시겠습니까? (1: 예, 0: 아니오): 0
과목을 선택하세요 (1부터 2까지, 0: 종료): 0
```

- 사용자가 퀴즈를 전부 풀고 일기장을 작성하는 기능 작성

- 테스트 결과 스크린샷

```
일기장을 여시겠습니까? (1: 예, 0: 아니오): 1
- 일기 쓰기 ~
비밀번호를 입력하세요 (최대 20자리): project

=== 비밀번호 확인 중===

=== 비밀번호 확인 완료.===

12월20일
오늘은 시험이 드디어 끝났다.
중강이지만 내일까지 C언어를 끝내야지만 진정한 자유를 찾을 수 있다.
12월20일
드디어 코드를 끝냈다아아악!!!!!!!!!!!!

내용을 계속 작성하세요! 종료하려면 EXIT를 입력하세요.

12월21일
최종 보고서 작성 테스트 중이다.
EXIT
비밀일기 입력을 종료합니다.
```



(좌) 코드 실행 테스트, (우) 작성된 txt 파일

(5) 일정 미루기

- 스택을 이용해 공부가 끝난 후, 오늘 계획을 수행했는지 묻고 답에 따라 할 일을 미루는 기능 작성

- 테스트 결과 스크린샷

<계획 1일차로 가정, 1일차 계획 수행 시 코드>

```
메뉴를 선택하세요: 5
오늘 할 일을 수행하셨나요? (1: 예, 0: 아니오): 1
오늘 할 일을 수행하셨습니다. 오늘차 계획을 삭제합니다. (내일 해야할 일 : 2일차 계획)
```

<계획 2일차, 2일차 계획을 수행하지 못했을 때 코드(내일 밀린 2일차 계획 수행)>

```
메뉴를 선택하세요: 5
오늘 할 일을 수행하셨나요? (1: 예, 0: 아니오): 0
오늘 할 일을 수행하지 않으셨습니다. 내일로 미뤄집니다. (내일 해야 할 일 : 2일차 계획)
```

5. 계획 대비 변경 사항

1) 오늘의 계획 출력 기능

- 전: 사용자에게 시험 기간 시작 날짜를 입력 받아 오늘의 날짜와의 차이를 계산하여 날짜에 맞는 오늘의 계획 출력

- 후: 과목별 남은 일자 출력

사유: 30일의 계획에서 수행하지 못할 것을 감안하여 남은 일자를 보여주도록 수정하고, 수행했을 때 다음 날 해야 할 일은 메뉴 5번의 일정 미루기를 통해 보여주도록 함.

2) 30일치의 계획 저장 구조 변경

- 전: 배열을 이용해 1부터 30까지의 계획을 저장

- 후: 스택을 이용하여 1부터 30까지의 값 저장

사유: 배열로 30개의 값을 하나씩 미루는 과정보다 스택의 후입선출 방식을 이용하여 일정을 미루는 경우가 편리하다고 판단함.

3) 일기장 기능 추가

- 전: 일기 기능 없이 퀴즈 출력

- 후: 입력한 개념을 이용한 퀴즈를 전부 수행했을 때 일기장 작성 기능 도입

사유: 하루의 공부를 끝마치고 최종적으로 퀴즈를 풀며 마무리 정리를 다 하고 나서 일기를 작성하면 그 날의 뿌듯함과 하루를 돌아볼 수 있는 장점을 고려하여 추가함.

6. 느낀점

이 학부에 들어와 처음으로 체계적인 하나의 프로젝트를 진행해보았는데, 진행하는 과정 내내 나름 개발자가 된 기분이었습니다. 종일 구현할 기능에 대해 구상

하고 기능의 장단점을 분석하며 내 프로젝트와 맞는 기능을 생각해내는 시간이 소중한 경험이 되었습니다. 과에 대한 아는 바 없이 점수만 맞춰 들어온 과였는데, 과와의 적성에 대해서도 어느정도 느껴볼 수 있는 시간이었습니다. 무작정 컴퓨터 언어에 대해 달달 외우고 문제를 수행하는 테스트도 좋지만, 하나부터 열까지 배운 점을 적용하여 스스로 프로젝트를 수행한다는 것 또한 언어를 배우는 데에 있어 큰 도움이 될 것 같습니다. 교수님 한 학기 정말 수고 많으셨습니다!