C프로그래밍 및 실습

24시간이 충분한 스케줄 플래너

진척 보고서 #1

제출일자:2023.11.26
제출자명:최세인
제출자학번:231344

1. 프로젝트 목표

1) 배경 및 필요성

많은 사람들이 하루를 효율적으로 보내기를 희망하지만 계획 세우는 일에 능하지 않거나 무리한 계획을 세우는 경우가 다반사이다. 또한 아예 계획을 세우지 않고 즉흥적으로 일을 처리하는 사람들도 존재한다. 시간을 효율적으로 쓰지 못하는 현대인들은 24시간이 부족하다고 느끼며 이러한 악습관은 바쁜 일상 속에서 쉽게 피로와 무기력함을 느끼게 한다. 이 문제를 해결하기 위해 자동으로 일정을 짜주는 사용자 맞춤형 스케줄 플래너가 필요하다.

2) 프로젝트 목표

주어진 24시간을 적절하게 자동 분배해 사용자가 입력한 스케줄을 가지고 합리적인 플래너를 작성해주는 프로그램을 제작한다. 이 프로그램은 해야 할 일을 효율적으로 수행 한 후, 남는 시간을 활용할 수 있도록 도와 삶의 균형을 맞추고 여가 시간 확보, 심리적 불안 감소, 삶의 효율 증진을 목표로 한다.

3) 차별점

기존의 스케줄 프로그램은 단순히 사용자가 입력한 스케줄을 저장하여 잊어버리지 않게 끔 환기해 주는 역할에 그쳤으며 시간을 직접 계산해서 짜야하는 번거로움이 있다. 이번 프로젝트에서 시행될 프로그램은 예상 소요시간, 일의 난이도 등을 고려하여 최적의 시간표와 일정을 제공하는 차별화된 서비스가 특징이다.

2. 기능 계획

1) 기능 1 : 모드 설정

- 직장, 공부, 휴일, 여행의 4가지 모드를 제공하여 스케줄의 강도를 결정한다.

(1) 세부 기능 1 : 집중 모드

- 직장, 공부 모드는 집중 모드로 분류하여 정해진 시간 내에 집중하여 일의 효율을 높일 수 있도록 예상 소요시간에 +- 0.5~1을 해준다.

(2) 세부 기능 2 : 여유 모드

휴일, 여행 모드는 여유 모드로 분류하여 시간에 쫓기지 않고 자유로움을 최대한 즐길 수 있도록 예상 소요시간에 +0.5~1을 해준다.

2) 기능 2 : 스케줄 시작 시간 설정 및 할 일 입력 받기

- 하루 중 스케줄을 시작하고자 하는 시간을 입력 받아 기준으로 삼는다.

- 할 일을 하나씩 입력 받는다.

3) 기능 3 : 세부 고려사항

- 일을 처리하는 데 걸리는 예상 소요시간과 일의 난이도를 입력 받는다.

(1) 세부 기능 1 : 난이도

- 1~5점으로 평가한다.

(2) 세부 기능 2 : 시간 분배

- 예상 소요시간과 난이도의 차를 구하고 미리 설정해 둔 값의 범위에 따라 스케줄 플래너에 분배할 최종 시간을 구한다.

3. 진척사항

1) 기능 구현

(1) 모드 설정

- 입력 값: 모드 번호/ 출력 값: 선택한 모드

- 1~4번까지 4가지 모드 중에 하나를 선택한다.

- 적용된 배운 내용: 배열

-코드 스크린샷

```
/*모드 설정 코드 블록*/
int modenum;
char mode[4][10] = { "직장", "공부", "휴일", "여행" };  //4가지 모드 배열에 저장
printf("모드를 설정합니다.₩n");
printf("<집중 모드>₩n");
printf("1. 직장₩n");
printf("2. 공부₩n");
printf("<여유 모드>₩n");
printf("3. 휴일₩n");
printf("4. 여행₩n");
printf("-----------------------------₩n");
printf("원하는 모드의 번호를 입력하세요: ");
scanf_s("%d", &modenum);
printf("%s모드", mode[modenum - 1]);
printf("₩n-----------------------------₩n");
```

(2) 스케줄 시작 시간 설정

- 입출력 값: 스케줄 시작 시간

-코드 스크린샷

```
/*스케줄 시작 시간 설정 코드 블록*/
double start;
printf("스케줄 시작 시간을 입력하세요(00시~24시): ");
scanf_s("%lf", &start);
printf("%.1lf시", start);
printf("₩n-----------------------------₩n");
```

(3) 할 일 입력 받기

- 입력 값: 할 일

- 문자열을 입력 받은 후 할당한 동적 메모리에 값을 복사하여 저장한다. end를 입력하면 입력 받기를 멈춘다.

- 적용된 배운 내용: 버퍼 비우기, 포인터, 반복문, 입력 함수, 문자열 비교 함수, 조건문, 동적 메모리, 문자열 복사함수, 문자열 길이 계산 함수

-코드 스크린샷

```c
/*할 일 입력 받는 코드 블록*/
char tasks[100] = { "" };
char* str[100];
int i;
int count = 0;

getchar(); //버퍼에 남아있는 개행 문자 제거
for (i = 0; i < 30; i++) {

    printf("할 일을 하나씩 입력하세요(end를 입력하면 종료): ");
    fgets(tasks, sizeof(tasks), stdin);
    tasks[strlen(tasks) - 1] = '₩0';      //개행문자 제거
    if (strcmp(tasks, "end") == 0) {      //end를 입력하면 할 일 입력 받기 중지
        break;
    }
    str[i] = (char*)malloc(strlen(tasks) + 1);  //할 일을 저장할 메모리 동적할당
    if (str[i] == NULL) {                    //동적 할당에 실패할 경우
        printf("메모리가 부족합니다.₩n");
        exit(1);                             // 프로그램 종료
    }
    strcpy_s(str[i], strlen(tasks) + 1, tasks);     //할 일 동적 메모리에 복사
    count++;   //할 일 수 세기

}
```

(4) 예상 소요시간과 난이도 입력 받기

- 입력 값: 사용자가 예상하는 소요시간과 난이도

- 집중모드와 여유모드로 나눠 시간 분배에 필요한 예상 소요시간과 난이도를 입력 받는다.

- 적용된 배운 내용: 포인터, 조건문, 반복문

-코드 스크린샷

```c
/*예상 소요시간과 난이도를 입력 받는 코드 블록*/
double req_time;  //예상 소요시간 변수
int level;        //난이도 변수
double gap;       //예상 소요시간 - 난이도
double* end[10];  //할 일 끝나는 시각 변수
double* setting_time[10];    //할 일 시작하는 시각 변수
printf("------------------------------₩n");
//집중모드
if ((modenum == 1) || (modenum == 2)) {
    for (i = 0; i < count; i++) {

        printf("%s의 예상 소요시간을 입력하세요: ", str[i]);
        scanf_s("%lf", &req_time);

        printf("%s의 난이도를 입력하세요(1~5): ", str[i]);
        scanf_s("%d", &level);
```

```c
//여유모드
if ((modenum == 3) || (modenum == 4)) {
    for (i = 0; i < count; i++) {

        printf("%s의 예상 소요시간을 입력하세요: ", str[i]);
        scanf_s("%lf", &req_time);

        printf("%s의 난이도를 입력하세요(1~5): ", str[i]);
        scanf_s("%d", &level);
```

(5) 플래너 출력(예상 소요시간과 난이도의 차 이용)

- 입력 값: 예상 소요시간, 난이도 / 출력 값: 차이 값, 일의 끝나는 시각, 시작하는 시각, 완성된 플래너

- 집중 모드는 차이 값의 범위에 따라 예상 소요시간에 +- 0.5~1을 해주어 플래너에 작성할 시간을 최종 결정한다. 여유 모드는 차이 값의 범위에 따라 최대 1시간을 추가적으로 부여하여 자유를 즐길 수 있도록 시간을 최종 결정한다.

할 일이 끝나는 시간을 다른 할 일의 시작 시간으로 지정한다.

- 적용된 배운 내용: 포인터, 조건문, 반복문, 동적 메모리

-코드 스크린샷

```c
        gap = req_time - (double)level;    //예상 소요시간과 난이도의 차 구하기

        /*예상 소요시간에 비해 난이도가 어려운 경우*/

        if (gap < -2) {
            req_time += 1;                 //예상 소요시간 +1
        }
        else if (gap < -1) {
            req_time += 0.5;               //예상 소요시간 +0.5
        }
        else if (-1 <= gap && gap <= 1) {
            req_time += 0;                 // 예상 소요시간 그대로 유지
        }
        /*예상 소요시간에 비해 난이도가 쉬운 경우*/
        else if (1 < gap && gap <= 2) {
            req_time -= 0.5;               //예상 소요시간 -0.5
        }
        else if (gap > 2) {
            req_time -= 1;                 //예상 소요시간 -1
        }
        setting_time[i] = (double*)malloc(sizeof(double));  //할 일 시작하는 시간 저장하는 동적 메모리 할당
        if (setting_time[i] == NULL) {
            exit(1);
        }
        end[i] = (double*)malloc(sizeof(double));          //할 일 끝나는 시간 저장하는 동적 메모리 할당
        if (end[i] == NULL) {
            exit(1);
        }
        if (i == 0) {
            *setting_time[i] = start;
        }
        else {
            *setting_time[i] = *end[i - 1];
        }
        *end[i] = *setting_time[i] + req_time;       //할 일이 끝나는 시간 = 다음 할 일 시작 시간
    }
    /*스케줄 플래너 출력*/
    printf("-----------------------------Wn");
    printf("플래너 작성 완료!Wn");
    for (i = 0; i < count; i++) {
        printf("%.1f시-%.1f시: %sWn", *setting_time[i], *end[i], str[i]);
    }
}
```

```
            gap = req_time - (double)level;
            /*예상 소요시간에 비해 난이도가 어려운 경우*/
            if (gap <= -1) {
                req_time += 1;
            }
            else if (-1 < gap < 0) {
                req_time += 0;
            }
            /*예상 소요시간에 비해 난이도가 쉬운 경우*/
            else if (gap >= 1) {
                req_time += 0.5;      // 여유모드이므로 추가시간 있음
            }
            setting_time[i] = (double*)malloc(sizeof(double));
            if (setting_time[i] == NULL) {
                exit(1);
            }
            end[i] = (double*)malloc(sizeof(double));
            if (end[i] == NULL) {
                exit(1);
            }
            if (i == 0) {
                *setting_time[i] = start;
            }
            else {
                *setting_time[i] = *end[i - 1];
            }
            *end[i] = *setting_time[i] + req_time;
        }
        /*스케줄 플래너 출력*/
        printf("------------------------------₩n");
        printf("플래너 작성 완료!₩n");
        for (i = 0; i < count; i++) {
            printf("%.1f시-%.1f시: %s₩n", *setting_time[i], *end[i], str[i]);
        }
    }
```

(6) 추가 메뉴

- 입력 값: 메뉴 번호 / 출력 값: 코드 반복 또는 프로그램종료

- 새로운 플래너를 만드는 추가하기 항목과 프로그램을 종료하는 항목 중에 선택한다.

- 적용된 배운 내용: 조건문

-코드 스크린샷

```
    }
    /*스케줄 플래너 작성 후 메뉴 선택*/
    int menu_num;
    printf("₩n------------------------------₩n");
    printf("1. 플래너 추가하기 ");        //코드 반복
    printf("2. 종료하기 ");
    scanf_s("%d", &menu_num);
    if (menu_num == 2) {
        break;      //반복문 벗어나기
    }
    printf("₩n");
```

(7) 동적 메모리 반환

- 동적으로 할당한 저장 공간을 free함수를 사용하여 반환

- 적용된 배운 내용: 반복문, 동적 메모리

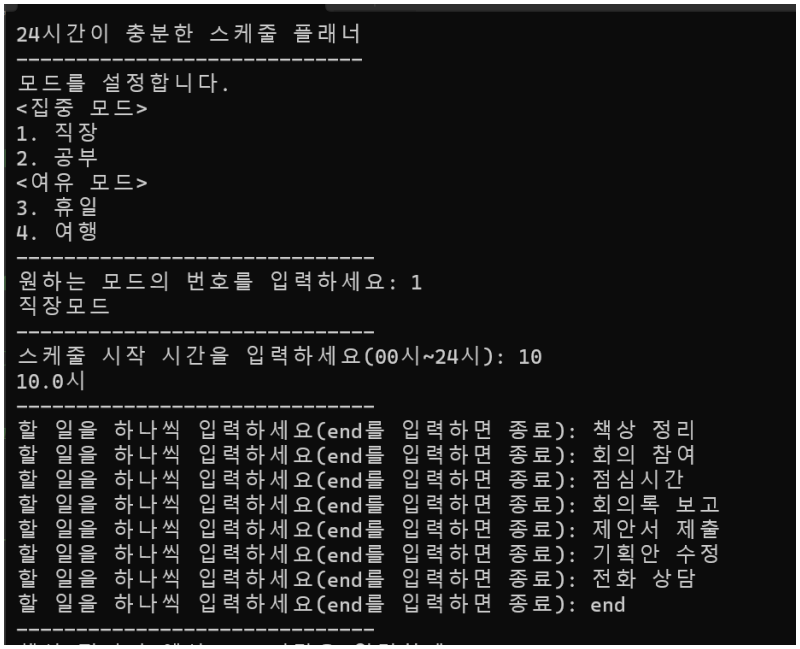
-코드 스크린샷

```
    /*동적 메모리 반환*/
    for (i = 0; i < count; i++) {
        free(str[i]);
    }
    for (i = 0; i < count; i++) {
        free(setting_time[i]);
    }
    for (i = 0; i < count; i++) {
        free(end[i]);
    }
}
```

2) 테스트 결과

(1) 테스트한 기능 이름: 모드 설정- 스케줄 시작 시간 설정- 할 일 입력받기- 예상 소요시간과 난이도 입력 받기- 플래너 출력- 추가 메뉴

- 테스트 결과 스크린샷



4. 계획 대비 변경 사항

1) 24시간 중 제외할 시간 설정하기

- 이전: 기상 시간, 취침 시간, 약속 시간 등 스케줄 조정에서 제외하고자 하는 시간을 따로 설정하려 함.
- 이후: 스케줄 시작 시간을 설정하는 것으로 변경
- 사유: 스케줄 전인 기상 시간과 스케줄 후인 취침 시간은 스케줄 조정 대상이 아니라고 판단했으며 약속 시간은 스케줄에 포함해야 한다고 생각해 제외대상으로 분류하지 않기로 판단함.

2) 이동거리 계산

- 이전: 스케줄 사이에 이동이 필요하다면 장소 간 이동거리를 고려해 스케줄을 작성하려 함.
- 이후: 기능 삭제-> 이동에 걸리는 시간을 포함한 예상 소요시간 입력 받기로 변경
- 사유: 이동 거리는 할 일처럼 난이도와 함께 고려해 계산하기에는 주관적이라고 판단함.

5. 프로젝트 일정

업무	~11/3	~11/21	~11/23	~11/24	~12/10	~12/23
제안서 제출	완료					
기능 1 구현		완료				
기능 2 구현			완료			
기능 3 구현				완료		
기능 디테일 추가				----------→		
최종 프로그램/ 테스트 코드/ 보고서 점검 및 제출						-----→