SW3106 : 프로그래밍 입문 Project #2

1-1 print_student_list 함수

미리 선언된 구조체 배열을 학기별로 오름차순으로 정렬해준 뒤 출력해주도록 구현했다. 정렬할 때 실제 배열의 순서를 바꾸도록 설정했다.

```
void print_student_list(STUDENT* s)
{
    printf("\n%8s  %-11s %-5s %10s %14s\n", "번호", "이름", "학번", "학기", "수강과목");
    STUDENT temp;

    // 정렬
    for (int i = array_size; i > 1; i--)
    {
        for (int j = 1; j < i; j++)
        {
            if (s[j-1].semester > s[j].semester)
            {
                temp = s[j - 1];
                s[j - 1] = s[j];
                s[j] = temp;
            }
        }
    }

    // 출력
    for(int i=0; i<array_size; i++) {
        printf("%3d번: ", i+1);
        print_student(&s[i]);
    }
}
```

1-2 find_student 함수

구조체 배열 포인터의 요소를 하나씩 받아올 구조체 포인터 sour을 만들어 위에서 정렬된 배열을 기준으로 하나씩 순회하며 속성값(class_name, ID, name)을 비교하고 만족하는 것이 있으면 출력해주고 없으면 해당 학생 정보를 찾을 수 없다고 출력한다. 이때 속성값들은 다 문자열이므로 strcmp() 함수를 이용하여 비교한다.

```
void find_student(STUDENT *s) {
    char input[BUF_SIZE];
    int is_checked;

    STUDENT *sour;
    while(1) {
        is_checked = 0;
```

```
        printf("\n찾으려는 학생의 이름 또는 수강과목을 입력하세요(종료는 exit): ");
        scanf("%s", input);

        if(strcmp("exit", input) == 0)
            break;

        sour = s;
        for(int i=0; i<array_size; i++) {
            if(strcmp(sour->class_name, input) == 0 || strcmp(sour->ID, input) ==
0 || strcmp(sour->name, input) == 0) {
                printf("%3d번: ", i+1);
                print_student(&s[i]);
                is_checked = 1;
            }
            sour++;
        }

        if(is_checked == 0)
            printf("해당 학생 정보를 찾을 수 없습니다.");
    }
}
```

1-3 print_presentation_list 함수

기본적으로 발표자 수가 MAX_PRESENTATION_LIST와 같아지면 알아서 종료하도록 설정해뒀다. 이상태에서 발표 리스트에 추가할 때 미리 오름차순으로 정렬된 배열을 기준으로 학생을 가져오도록 해야 하는데, 함수 호출 순서상 print_student_list() 함수가 먼저 호출되므로 구조체 배열은 이미 오름차순 정렬된 상태라고 볼 수 있다.

발표 리스트에 추가할 때 이미 존재하는지 여부를 판단하기 위해 compare() 함수를 만들었다. 발표 리스트를 순회하며 넣고자 하는 사람과 비교했을 때 같으면 1을 출력하고 아니면 0을 출력함으로써 판단한다.

발표 리스트에서 제거할 때 사용하기 위해 delete() 함수를 만들었다. 발표 리스트에서 제거하고자 하는 사람의 위치를 기준으로 한칸씩 당겨 덮어씌우는 방식으로 구현하였다. 이 방식의 단점이라고 하자면 바로 쓰레기값인데, delete() 함수 또한 당기기만 할뿐 맨 뒷부분은 쓰레기값이 들어오게 된다. 하지만 이는 present_num을 통해 출력범위를 제한하는 것으로 커버가 가능하다.

```
// 학생 구조체 비교 함수
int comapre(STUDENT *s1, STUDENT *s2) {
    if(strcmp(s1->name, s2->name) == 0 &&
    strcmp(s1->ID, s2->ID) == 0 &&
    strcmp(s1->class_name, s2->class_name) == 0 &&
    (s1->semester == s2->semester)) {
        return 1;
    }

    return 0;
}

// 발표 리스트에서 제거 함수
void delete(STUDENT *s, int idx) {
    for(int i=idx; i<array_size; i++) {
```

```
            s[i] = s[i+1];
        }
    }

    void print_presentation_list(STUDENT* s) {
        STUDENT list[array_size];
        int size;
        int input;
        int num;
        int present_num = 0;
        int is_exist = 0;

        while(1) {
            if(present_num == MAX_PRESENTATION_LIST)
                break;

            is_exist = 0;

            printf("\n발표리스트에 추가하려면 1번, 삭제하려면 2번, 종료하려면 3번을 누르세
    요: ");
            scanf("%d", &input);

            if(input==3)
                break;
            if(input==1) {
                printf("발표리스트에 추가할 학생 번호를 입력하세요: ");
                scanf("%d", &num);

                if(num > 8 || num < 1) {
                    printf("잘못된 번호입니다.\n");
                    continue;
                }

                for(int i=0; i<present_num; i++) {
                    if(comapre(&list[i], &s[num-1]) == 1) {
                        is_exist = 1;
                        printf("이미 발표리스트에 있는 학생입니다.\n");
                        break;
                    }
                }

                if(is_exist == 1) {
                    continue;
                }

                list[present_num] = s[num-1];
                present_num += 1;
            }
            if(input==2) {
                printf("발표리스트에서 삭제할 학생 번호를 입력하세요: ");
                scanf("%d", &num);

                if(num > 8 || num < 1) {
                    printf("잘못된 번호입니다.\n");
```

```
                continue;
            }

            if(present_num >= num)
                is_exist = 1;

            if(is_exist == 1) {
                delete(list, num-1);
                present_num -= 1;
            } else {
                printf("삭제할 학생이 없습니다.\n");
                continue;
            }
        }
        is_exist = 0;
        // 출력
        printf("<< 발표리스트 >>\n");
        for(int i=0; i<present_num; i++) {
            printf("%3d번: ", i+1);
            print_student(&list[i]);
        }
        printf("총 발표명 수: %d명\n", present_num);
    }
}
```

Problem1

```
#include <stdio.h>
#include <string.h>

#define BUF_SIZE 64
#define MAX_PRESENTATION_LIST 3
#define MAX_SEMESTER 8

int array_size;

typedef struct STUDENT {
    char name[BUF_SIZE];
    char ID[BUF_SIZE];
    int semester;
    char class_name[BUF_SIZE];
} STUDENT;

// 학생 구조체 비교 함수
int comapre(STUDENT *s1, STUDENT *s2) {
    if(strcmp(s1->name, s2->name) == 0 &&
    strcmp(s1->ID, s2->ID) == 0 &&
    strcmp(s1->class_name, s2->class_name) == 0 &&
    (s1->semester == s2->semester)) {
        return 1;
    }
```

```
    return 0;
}

void delete(STUDENT *s, int idx) {
    for(int i=idx; i<array_size; i++) {
        s[i] = s[i+1];
    }
}

/* 출력 부분: 수정하지 마세요 */
void print_student(const STUDENT *s){
    printf("%-9s %-5s %5d %10s\n", s->name, s->ID, s->semester, s->class_name);
}
/* ------------------------- */

/* print_student_list 함수 구현 */
// 반드시 다음 코드 사용해서 출력하세요.
// 가장 윗줄: printf("\n%8s  %-11s %-5s %10s %14s\n", "번호", "이름", "학번", "학
기", "수강과목"); 이용하여 출력
// 학생 번호: printf("%3d번: ",변수); 이용하여 출력
// 학생 정보: print_student 함수 이용하여 출력
/* ------------------------- */
void print_student_list(STUDENT* s)
{
    printf("\n%8s  %-11s %-5s %10s %14s\n", "번호", "이름", "학번", "학기", "수강과
목");
    STUDENT temp;

    // 정렬
    for (int i = array_size; i > 1; i--)
    {
        for (int j = 1; j < i; j++)
        {
            if (s[j-1].semester > s[j].semester)
            {
                temp = s[j - 1];
                s[j - 1] = s[j];
                s[j] = temp;
            }
        }
    }

    // 출력
    for(int i=0; i<array_size; i++) {
        printf("%3d번: ", i+1);
        print_student(&s[i]);
    }
}

/* find_student 함수 구현 */
// 반드시 다음 코드 사용해서 출력하세요.
// 학생 번호: printf("%3d번: ",변수); 이용하여 출력
// 학생 정보: print_student 함수 이용하여 출력
```

```
/* ------------------------- */
void find_student(STUDENT *s) {
    char input[BUF_SIZE];
    int is_checked;

    STUDENT *sour;
    while(1) {
        is_checked = 0;
        printf("\n찾으려는 학생의 이름 또는 수강과목을 입력하세요(종료는 exit): ");
        scanf("%s", input);

        if(strcmp("exit", input) == 0)
            break;

        sour = s;
        for(int i=0; i<array_size; i++) {
            if(strcmp(sour->class_name, input) == 0 || strcmp(sour->ID, input) ==
0 || strcmp(sour->name, input) == 0) {
                printf("%3d번: ", i+1);
                print_student(&s[i]);
                is_checked = 1;
            }
            sour++;
        }

        if(is_checked == 0)
            printf("해당 학생 정보를 찾을 수 없습니다.");
    }
}

/* print_presentation_list 함수 구현 */
// 반드시 다음 코드 사용해서 출력하세요.
// 가장 윗줄: printf("<< 발표리스트 >>\n"); 이용하여 출력
// 학생 번호: printf("%3d번: ",변수); 이용하여 출력
// 학생 정보: print_student 함수 이용하여 출력
/* ------------------------- */
void print_presentation_list(STUDENT* s) {
    STUDENT list[array_size];
    int size;
    int input;
    int num;
    int present_num = 0;
    int is_exist = 0;

    while(1) {
        if(present_num == MAX_PRESENTATION_LIST)
            break;

        is_exist = 0;

        printf("\n발표리스트에 추가하려면 1번, 삭제하려면 2번, 종료하려면 3번을 누르세
요: ");
        scanf("%d", &input);
```

```
        if(input==3)
            break;
        if(input==1) {
            printf("발표리스트에 추가할 학생 번호를 입력하세요: ");
            scanf("%d", &num);

            if(num > 8 || num < 1) {
                printf("잘못된 번호입니다.\n");
                continue;
            }

            for(int i=0; i<present_num; i++) {
                if(comapre(&list[i], &s[num-1]) == 1) {
                    is_exist = 1;
                    printf("이미 발표리스트에 있는 학생입니다.\n");
                    break;
                }
            }

            if(is_exist == 1) {
                continue;
            }

            list[present_num] = s[num-1];
            present_num += 1;
        }
        if(input==2) {
            printf("발표리스트에서 삭제할 학생 번호를 입력하세요: ");
            scanf("%d", &num);

            if(num > 8 || num < 1) {
                printf("잘못된 번호입니다.\n");
                continue;
            }

            if(present_num >= num)
                is_exist = 1;

            if(is_exist == 1) {
                delete(list, num-1);
                present_num -= 1;
            } else {
                printf("삭제할 학생이 없습니다.\n");
                continue;
            }
        }
        is_exist = 0;
        // 출력
        printf("<< 발표리스트 >>\n");
        for(int i=0; i<present_num; i++) {
            printf("%3d번: ", i+1);
            print_student(&list[i]);
        }
        printf("총 발표명 수: %d명\n", present_num);
```

```
    }
}

int main(void)
{
    STUDENT array[] = {
        {"Jihyeon", "2018001", 8, "class1"},
        {"Sujung", "2022015", 2, "class2"},
        {"Minjung", "2021016", 3, "class2"},
        {"Minji", "2021013", 4, "class4"},
        {"Sujung", "2020033", 5, "class3"},
        {"Heejoon", "2020010", 6, "class4"},
        {"Ayoon", "2019022", 5, "class1"},
        {"Jihyeon", "2019001", 7, "class5"},
    };
    array_size = sizeof(array) / sizeof(STUDENT);
    print_student_list(array);
    // #1-2 구현 시 반드시 다음 코드 사용해서 출력하세요.
    find_student(array);

    // #1-3 구현 시 반드시 다음 코드 사용해서 출력하세요.
    // printf("\n발표리스트에 추가하려면 1번, 삭제하려면 2번, 종료하려면 3번을 누르세
요: ");
    // printf("발표리스트에 추가할 학생 번호를 입력하세요: ");
    // printf("이미 발표리스트에 있는 학생입니다.\n");
    // printf("잘못된 번호입니다.\n");
    // printf("발표리스트에서 삭제할 학생 번호를 입력하세요: ");
    // printf("삭제할 학생이 없습니다.\n");

    /* Put your answer */
    print_presentation_list(array);

    return 0;
}
```

Problem2

기본적으로 세트가 만들어질 때, 만들어지는 세트의 수량에 따라 다른 메뉴의 단독 수량이 결정된다. 반복문으로 순회하며 검사하고 그때그때 수량의 변동을 주면 좋을 듯 하여 포인터 배열을 만들었다.

min 함수는 음료와 케익의 수량 중 적은 것을 기준으로 세트를 구성할 때 사용하기 위해 만들었다.

위의 기초 설계를 바타으로 포인터 배열을 순회하며 total 값들은 누적하고, 단독 수량들은 포인터를 통해 수량 감소를 적용시킨다. 위의 과정을 구현할 때 각 제품에 대한 값을 계산할 때 편하게 하기 위해 bev_price_arr, cake_price_arr를 포인터 배열을 만들 때의 규칙에 따라 만들었다.

```
#include <stdio.h>
#include <string.h>

/* 변수 선언 부분: 수정하지 마세요 */
#define price_americano 3500
```

```c
#define price_icetea 3000
#define price_smoothie 5000
#define price_carrot 7000
#define price_cheese 6000
#define price_choco 5500
/* ------------------------------ */

int min(int a, int b) {
    if(a == 0 || b == 0)
         return 0;
    if (a>=b)
        return b;
    else
        return a;
}

int main(){
    /* 변수 선언 부분: 수정하지 마세요 */
    int num_americano;
    int num_icetea;
    int num_smoothie;
    int num_carrot;
    int num_cheese;
    int num_choco;
    int num_set1;
    int num_set2;
    int total_price_set1;
    int total_price_set2;
    int total_price;
    /* ------------------------------ */

    // 반드시 다음 코드 사용해서 출력하세요.
    // 주문 받을 때: printf("주문하고자 하는 음료(커피, 아이스티, 스무디)와 케익(당근, 치즈, 초코)의 갯수를 각각 입력하세요:\n"); 이용하여 출력
    // 음료 갯수가 10개 넘었을 때: printf("음료 갯수가 10개를 넘었습니다. 다시 주문해주세요.\n\n"); 이용하여 출력
    // 케익 갯수가 10개 넘었을 때: printf("케익 갯수가 10개를 넘었습니다. 다시 주문해주세요.\n\n"); 이용하여 출력

    /*put your answer*/
    total_price_set1 = 0;
    total_price_set2 = 0;
    total_price = 0;

    int total_bev;
    int total_cake;
    while(1) {
        printf("주문하고자 하는 음료(커피, 아이스티, 스무디)와 케익(당근, 치즈, 초코)의 갯수를 각각 입력하세요:\n");
        scanf("%d %d %d %d %d %d", &num_americano, &num_icetea, &num_smoothie, &num_carrot, &num_cheese, &num_choco);

        total_bev = num_americano + num_icetea + num_smoothie;
        total_cake = num_carrot + num_cheese + num_choco;
```

```
        if(total_bev > 10) {
            printf("음료 갯수가 10개를 넘었습니다. 다시 주문해주세요.\n\n");
            continue;
        }
        if(total_cake > 10) {
            printf("케잌 갯수가 10개를 넘었습니다. 다시 주문해주세요.\n\n");
            continue;
        }

        break;
    }
    // 가격이 비싼 순으로 먼저 넣음
    int* bev_arr[3] = {&num_smoothie, &num_americano, &num_icetea};
    int bev_price_arr[3] = {price_smoothie, price_americano, price_icetea};

    int* cake_arr[3] = {&num_carrot, &num_cheese, &num_choco};
    int cake_price_arr[3] = {price_carrot, price_cheese, price_choco};

    int bev_cnt = total_bev;
    int cake_cnt = total_cake;

    // 세트1이 만들어지는 경우
    num_set1 = min(bev_cnt/2, cake_cnt/2);
    //printf("num_set1 : %d\n", num_set1);
    if(num_set1 >= 1) {
        bev_cnt -= num_set1 * 2;
        cake_cnt -= num_set1 * 2;

        int bev = num_set1 * 2;
        for(int i=0; i<3; i++) {
            bev -= *bev_arr[i];

            if(bev <= 0) {
                total_price_set1 += ((*bev_arr[i] + bev) * bev_price_arr[i]);
                *bev_arr[i] = -1 * bev;
            } else {
                total_price_set1 += (*bev_arr[i] * bev_price_arr[i]);
                *bev_arr[i] = 0;
            }
        }

        int cake = num_set1 * 2;
        for(int i=0; i<3; i++) {
            cake -= *cake_arr[i];

            if(cake <= 0) {
                total_price_set1 += ((*cake_arr[i] + cake) * cake_price_arr[i]);
                *cake_arr[i] = -1 * cake;
            } else {
                total_price_set1 += (*cake_arr[i] * cake_price_arr[i]);
                *cake_arr[i] = 0;
            }
        }
```

```
        total_price_set1 = total_price_set1 * 0.8;
    }

    // 세트2가 만들어지는 경우
    num_set2 = min(bev_cnt/2, cake_cnt);
    if(num_set2 >= 1) {
        bev_cnt -= num_set2 * 2;
        cake_cnt -= num_set2 * 2;

        int bev = num_set2 * 2;
        for(int i=0; i<3; i++) {
            bev -= *bev_arr[i];

            if(bev <= 0) {
                total_price_set2 += ((*bev_arr[i] + bev) * bev_price_arr[i]);
                *bev_arr[i] = -1 * bev;
            } else {
                total_price_set2 += (*bev_arr[i] * bev_price_arr[i]);
                *bev_arr[i] = 0;
            }
        }

        int cake = num_set2;
        for(int i=0; i<3; i++) {
            cake -= *cake_arr[i];

            if(cake <= 0) {
                total_price_set2 += ((*cake_arr[i] + cake) * cake_price_arr[i]);
                *cake_arr[i] = -1 * cake;
            } else {
                total_price_set2 += (*cake_arr[i] * cake_price_arr[i]);
                *cake_arr[i] = 0;
            }
        }

        total_price_set2 = total_price_set2 * 0.9;
    }

    total_price = total_price_set1 + total_price_set2 + price_americano *
num_americano + price_icetea * num_icetea +
    price_smoothie * num_smoothie + price_carrot * num_carrot + price_cheese *
num_cheese + price_choco * num_choco;

    /* 출력 부분: 수정하지 마세요 */
    printf("\n품목      갯수   금액\n");
    if (num_set1 > 0)
        printf("세트1    %4d %7d\n",num_set1, total_price_set1);
    if (num_set2 > 0)
        printf("세트2    %4d %7d\n", num_set2, total_price_set2);
    if (num_americano > 0)
        printf("커피     %4d %7d\n", num_americano, price_americano *
num_americano);
    if (num_icetea > 0)
```

```
            printf("아이스티 %4d %7d\n", num_icetea, price_icetea * num_icetea);
        if (num_smoothie > 0)
            printf("스무디   %4d %7d\n", num_smoothie, price_smoothie * num_smoothie);
        if (num_carrot > 0)
            printf("당근케익 %4d %7d\n", num_carrot, price_carrot * num_carrot);
        if (num_cheese > 0)
            printf("치즈케익 %4d %7d\n", num_cheese, price_cheese * num_cheese);
        if (num_choco > 0)
            printf("초코케익 %4d %7d\n", num_choco, price_choco * num_choco);
        printf("---------------------------\n");
        printf("총 지불 금액 %8d\n", total_price);
        /* ------------------------ */

        return 0;

    }
```