c_과제4_노기문_2025402040_컴퓨터정보공학부

/*전 보고서와 다른점 보통 코드 설명을 따로 적어두었지만 모두 <code>의 주석으로 대체*/

과제01. 성적 오름차순

- 5명의 학년, 성적, 이름을 받는 구조체를 만들고, 학년이 같으면 성적 순서, 성적이 같으면 이름 순서로 3가지 요소를 모두 고려하여 오름차순으로 정렬하시오.

입력 : 4 4.5 kwon
1 3.2 jo
3 2.3 yoon
3 1.6 koh
3 2.3 park

출력 : 1 3.2 jo
3 1.6 koh
3 2.3 park
3 2.3 yoon
4 4.5 kwon

<code>

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

typedef struct {//구조체 선언
	int grade;
	float gpa;
	char name[101];
} student;

// 이름 전체 알파벳 + 공백 비교 함수
int alpha_ascending(const char* a, const char* b) { //알파벳 오름 차순 (알파벳 순서대로)
	int i = 0;
	while (a[i] != '₩0' && b[i] != '₩0') {//마지막이 널문자니까 그 널문자가 두개가 다 아닐시 반복문 진행
		char aa = a[i];
		char bb = b[i];

		// 대소문자 구분 없이 비교 => 영어 문자는 정수형으로 대문자 +32 하면 소문자 반대는 -일경우
		if (aa >= 'A' && aa <= 'Z') aa += 32;
		if (bb >= 'A' && bb <= 'Z') bb += 32;
```

```
                if (aa < bb) return -1;
                else if (aa > bb) return 1;
                i++;
        }
        if (a[i] == '₩0' && b[i] != '₩0') return -1;
        if (a[i] != '₩0' && b[i] == '₩0') return 1;
        return 0;
}

// 영어 알파벳만 허용
int valid_name(const char* name) {
        for (int j = 0; name[j] != '₩0'; j++) {
                if (!((name[j] >= 'A' && name[j] <= 'Z') || (name[j] >= 'a' && name[j] <=
'z')))// 아까와 비슷하게 a-z/A-Z사이가 아니면 0으로 반환 즉 다시 출력하도록 설정
                        return 0;
        }
        return 1;
}

int main(void) {
        student list[5];

        for (int i = 0; i < 5; i++) {
                printf("%d번째 학생₩n", i + 1);

                printf("입력 : (학년 성적 이름(띄어쓰기 없이 성만 입력해주세요.))");
                scanf("%d %f %s", &list[i].grade , &list[i].gpa, list[i].name);

                if ((list[i].grade < 1 || list[i].grade > 4) || (list[i].gpa < 0 || list[i].gpa > 4.5)||
(!valid_name(list[i].name))) {
                        printf("유효하지 않습니다. 다시 입력하세요.₩n₩n");
                        i--;
                        continue;
                }
        }

        // 정렬
        for (int i = 0; i < 4; i++) {
```

```
                for (int j = i + 1; j < 5; j++) {
                        if ((list[i].grade > list[j].grade) || (list[i].grade == list[j].grade &&
list[i].gpa > list[j].gpa) || (list[i].grade == list[j].grade && list[i].gpa == list[j].gpa &&
alpha_ascending(list[i].name, list[j].name) > 0)
                                ) {
                                student temp = list[i];
                                list[i] = list[j];
                                list[j] = temp;
                        }
                }
        }

        // 출력
        printf("₩n정렬 결과:₩n");
        for (int i = 0; i < 5; i++) {
                printf("%d / %.2f / %s₩n", list[i].grade, list[i].gpa, list[i].name);
        }

        return 0;
}
```

<result screen>

```
1번째 학생
입력 : (학년 성적 이름(띄어쓰기 없이 성만 입력해주세요.))5 2.3 noh
유효하지 않습니다. 다시 입력하세요.

1번째 학생
입력 : (학년 성적 이름(띄어쓰기 없이 성만 입력해주세요.))-1 2.3 noh
유효하지 않습니다. 다시 입력하세요.

1번째 학생
입력 : (학년 성적 이름(띄어쓰기 없이 성만 입력해주세요.))1 -1 noh
유효하지 않습니다. 다시 입력하세요.

1번째 학생
입력 : (학년 성적 이름(띄어쓰기 없이 성만 입력해주세요.))1 6 noh
유효하지 않습니다. 다시 입력하세요.

1번째 학생
입력 : (학년 성적 이름(띄어쓰기 없이 성만 입력해주세요.))1 4.0 노
유효하지 않습니다. 다시 입력하세요.

1번째 학생
입력 : (학년 성적 이름(띄어쓰기 없이 성만 입력해주세요.))1 4.0 noh
2번째 학생
입력 : (학년 성적 이름(띄어쓰기 없이 성만 입력해주세요.))1 4.0 alice
3번째 학생
입력 : (학년 성적 이름(띄어쓰기 없이 성만 입력해주세요.))2 4.3 bob
4번째 학생
입력 : (학년 성적 이름(띄어쓰기 없이 성만 입력해주세요.))3 2.1 paul
5번째 학생
입력 : (학년 성적 이름(띄어쓰기 없이 성만 입력해주세요.))4 4.5 byeon

정렬 결과:
1 / 4.00 / alice
1 / 4.00 / noh
2 / 4.30 / bob
3 / 2.10 / paul
4 / 4.50 / byeon

C:\ROBIT\x64\Debug\ROBIT.exe(프로세스 20680)이(가) 0 코드(0x0)와 함께 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요...
```

<알고리즘 설명>

1. 알고리즘 설명:

이 문제는 학년, 성적, 이름을 입력받고 학년 순서대로 오름차순 학년 같으면 성적 오름 차순 성적 까지 같으면 알파벳 순으로 오름차순으로(따로 함수로 정리alpha_ascending) 정리하는 문제 이다.

먼저 학년 성적 이름을 입력받는다

=> 예외처리 학년이 음수, 5이상이면 다시 입력/성적 광운대학교성적 산출 기준으로 0미만, 4.5초과이면 다시 입력 / 이름또한 영문자로만(함수로 구분 valid_name) 입력을 받는다 이외의 경우는 다시입력

다 입력 받으면 정렬함

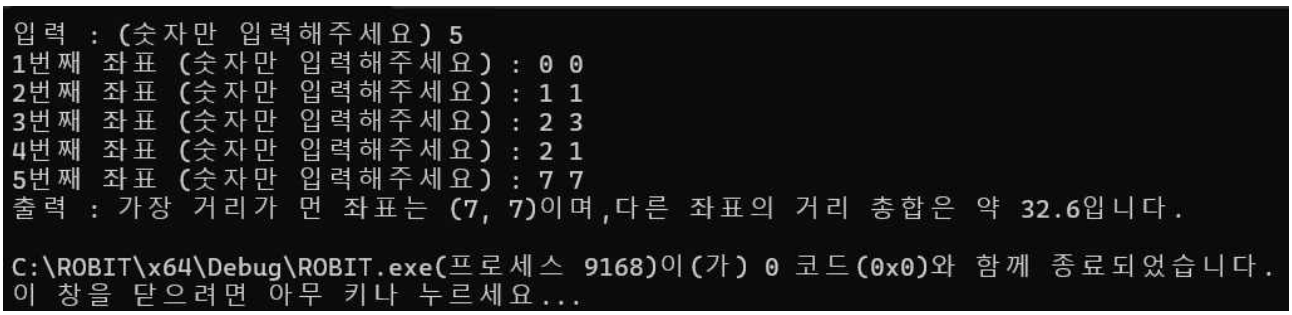
과제02. 좌표

<code>

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int x;
    int y;
} coordinate;

double root(double num) {// 간단하게 제곱근의 값을 나타내는 함수
    double origin = 0.0;
    double step = 0.0001;
    while (origin * origin < num)
        origin += step;
    return origin;
}
/*이 함수는origin * origin이 num 이상이 될 때까지 origin 을 작은 간격으로 증가시키며 반복 => 제곱근의 원리*/

int square(coordinate a, coordinate b) { // 피타고라스 법칙
    int dx = a.x - b.x;
    int dy = a.y - b.y;
    return dx * dx + dy * dy;
}

int main(void) {
    int n;
    printf("입력 : (숫자만 입력해주세요) ");
    scanf("%d", &n);

    coordinate* list = (coordinate*)malloc(n * sizeof(coordinate));
    if (list == NULL) {
        printf("메모리 할당 실패₩n");
        return 1;
    }
```

```
    for (int i = 0; i < n; i++) {
        printf("%d번째 좌표 (숫자만 입력해주세요) : ", i + 1);
        scanf("%d %d", &list[i].x, &list[i].y);
    }

    int far_index = 0; // 먼저 가장 먼좌표는 0,0으로 설정
    int max_sum_dist2 = -1; // 먼 길이를 할때 0이 먼 길이 일수도 있으니 음수 값으로 초기 설정

    for (int i = 0; i < n; i++) {
        int sum = 0;
        for (int j = 0; j < n; j++) {
            if (i != j)
                sum += square(list[i], list[j]);
        }
        if (sum > max_sum_dist2) {//만약 구한 길이가 전에 저장해뒀던 최대 길이보다 크면 최대 길
이에 저장해두고 먼 좌표를 저장
            max_sum_dist2 = sum;
            far_index = i;
        }
    }

    double total_dist = 0.0; // 나머지 거리 구하기
    for (int j = 0; j < n; j++) {
        if (j != far_index) {
            int dist2 = square(list[far_index], list[j]);
            total_dist += root((double)dist2);
        }
    }

    printf("출력 : 가장 거리가 먼 좌표는 (%d, %d)이며,다른 좌표의 거리 총합은 약 %.1f입니다.₩n ",
list[far_index].x, list[far_index].y, total_dist);
    free(list);
    return 0;
}
```

<result screen>



<알고리즘 설명 >

1. 알고리즘 설명

먼저 입력을 받은 뒤 각각 좌표의 제곱거리를 (A^2 +b^2 =c^2를 구하고 그것이 큰 애가 멀리 있는 좌표 이기에 그것을 저장한뒤 그 좌표를 기준으로 나머지 점들사이의 거리는 sqrt함수를 쓸수 없으니 직접적으로 루트를 구하는 함수를 만들어 0부터 시작해 숫자를 적게 증가 시켜 최대한 c^2과 비슷하면 그 값을 리턴하여 거리의 근사 값을 구한뒤 출력

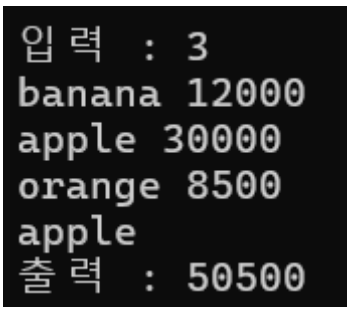
과제03. 금액 총합

<code>

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

typedef struct {// 구조체 선언
    char name[101];
    int money;
} Item;

int my_strcmp(const char* str1, const char* str2) {
    while (*str1 && *str2) {
        if (*str1 != *str2)
            return *str1 - *str2;
        str1++;
        str2++;
    }
    return *str1 - *str2;
}
/*두 개의 문자열 포인터 str1, str2를 받아서 문자열을 사전 순(아스키 값 기준) 으로 비교
두 문자열의 현재 문자가 NULL 문자('₩0')가 아닐 때 반복합니다.즉, 둘 다 끝나지 않았을 때까지 비교
두 문자가 다르면, 바로 그 차이를 리턴 ex : 'b' - 'a' = 1 ==> str1이 str2보다 사전 순으로 뒤에 있
음.
반복 종료 후 두 문자열이 지금까지 모두 같았다면, 이제 한쪽이 끝났는지 확인*/

int main(void) {
    int n, sum = 0;
    int found = 0;//flase
    while (1) {
        printf("입력 : ");
        scanf("%d", &n);
        if (n <= 0)
        {
            printf("다시입력하세요₩n");
            continue;
        }
        break;
```

```
    }

    Item* list = malloc(n * sizeof(Item)); // 구조체의 동적할당


    for (int i = 0; i < n; i++) {
        scanf("%s %d", list[i].name, &list[i].money);
    }

    char target[101];
    scanf("%s", target);// 마지막 겹치는or안겹치는 단어

    for (int i = 0; i < n; i++) {
        if (my_strcmp(list[i].name, target) == 0) {
            found = 1;
            break;
        }
    }

    if (found) {
        for (int i = 0; i < n; i++) {
            sum += list[i].money;
        }
        printf("출력 : %d₩n", sum);
    }
    else {
        printf("출력 : 0₩n");
    }

    free(list);
    return 0;
}
```

<result screen>



<알고리즘 설명 및 코드 설명>

1. 알고리즘 설명:

이번 문제의 핵심은 3개를 받고 추가적으로 하나를 더 받아 하나와 나머지를 비교해서 비교한 것이 0(False)면 0출력 1(True)면 금액의 총합을 내는 것이다

구조체에서 이름, 금액을 요소로 가지고 구조체를 동적할당하고 배열처럼 입력을 받는다는 것이 핵심이다

또한 아직 strcmp를 쓸수 없기에 따로 strcmp를 구사할수 있는 함수를 만들었다 자세한건 주석 참고 바란다.

과제04. 시간차 계산

<code>

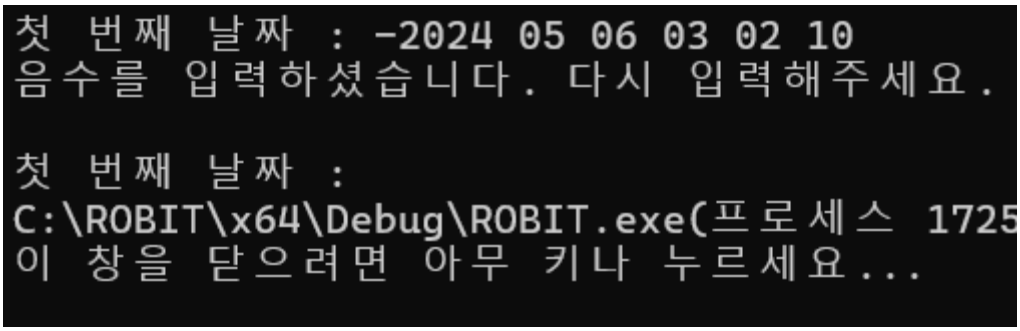
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

typedef struct {//자식 구조체
    int year;
    int month;
    int date;
} Day;

typedef struct {//자식 구조체
    int hour;
    int min;
    int sec;
} Time;

typedef struct { //엄마 구조체
    Day day;
    Time time;
} DateTime;

int yoon(int year) { //윤년 판단
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}

int month_change(int year, int month) {//1~12월 마지막 날짜 할당 + 윤년 고려
    int days[] = { 31,28,31,30,31,30,31,31,30,31,30,31 };
    if (month == 2 && yoon(year)) return 29;
    return days[month - 1];
}

// 날짜를 총 일수로 환산
int day_change(Day d) {
    int days = 0;

    for (int y = 1; y < d.year; y++) { //운년이면 하루 추가 => 86400초 추가
```

```
        if (yoon(y)) {
            days += 366;
        }
        else {
            days += 365;
        }
    }

    for (int m = 1; m < d.month; m++) {
        days += month_change(d.year, m);
    }

    days += d.date;
    return days;
}


// 날짜 + 시간 전체를 초로 환산
int second_chage(DateTime dt) {
    int days = day_change(dt.day);
    int seconds = days * 86400 + dt.time.hour * 3600 + dt.time.min * 60 + dt.time.sec;//하루는
86400초,1시간은3600초
    return seconds;
}

int main(void) {
    DateTime dt1, dt2;


    while (1) {
        printf("첫 번째 날짜 (YYYY MM DD): ");
        scanf("%d %d %d", &dt1.day.year, &dt1.day.month, &dt1.day.date);
        printf("첫 번째 시각 (HH MM SS): ");
        scanf("%d %d %d", &dt1.time.hour, &dt1.time.min, &dt1.time.sec);

        if (dt1.day.year < 0 || dt1.day.month < 0 || dt1.day.date < 0 || dt1.time.hour < 0 ||
dt1.time.min < 0 || dt1.time.sec < 0)
        {
```

```
            printf("음수를 입력하셨습니다. 다시 입력해주세요.₩n₩n");
            continue;
        }
        break;
    }

    while (1) { // 음수 예외처리
        printf("두 번째 날짜 (YYYY MM DD): ");
        scanf("%d %d %d", &dt2.day.year, &dt2.day.month, &dt2.day.date);
        printf("두 번째 시각 (HH MM SS): ");
        scanf("%d %d %d", &dt2.time.hour, &dt2.time.min, &dt2.time.sec);

        if (dt2.day.year < 0 || dt2.day.month < 0 || dt2.day.date < 0 || dt2.time.hour < 0 ||
dt2.time.min < 0 || dt2.time.sec < 0)
        {
            printf("음수를 입력하셨습니다. 다시 입력해주세요.₩n₩n");
            continue;
        }
        break;
    }

    int sec1 = second_chage(dt1);
    int sec2 = second_chage(dt2);
    int diff = abs(sec2 - sec1);//abs=> 절대값 / 만약 차이가 음수라도 그 절댓 값만 구하면 시간
차이를 알수 있기에


    //거스롬돈 계산과 같음.
    int h = diff / 3600;
    int m = (diff % 3600) / 60;
    int s = diff % 60;

    printf("시간 차이: %d시간 %d분 %d초₩n", h, m, s);

    return 0;
}
```

<result screen>

```
첫 번째 날짜 : 2024 01 01 10 20 30
두 번째 날짜 : 2024 06 18 12 05 10
시간 차이: 4057시간 44분 40초

C:\ROBIT\x64\Debug\ROBIT.exe(프로세스 17648)이(가) 0 코드(0x0)와 함께 종료되
었습니다.
이 창을 닫으려면 아무 키나 누르세요...
```



<알고리즘 설명 및 코드 설명>

먼저 날짜 차이를 계산 하려면 첫 번째 1초는 1초 / 1분은 60초 / 1시간은 3600초 / 하루는 84600초 / 월은 월별로 따로 계산 / 1년은 30,879,000초 윤년일때는30,963,600초 => 이것을 입력 받을 날짜를 끼리끼리 뺀후에 누가 날짜가 이른지 느린지 모르니까 절댓값으로 양수로 만들어 주고 난뒤 그 차이를 바탕으로 각각 곱해준뒤 거스름돈 계산 하는 것처럼 처음 시간은 3600을 나눈 몫, 분은 시간을 나눈 나머지의 60 나누기 , 초는 60의 나머지를 구하면 끝.

-끝-