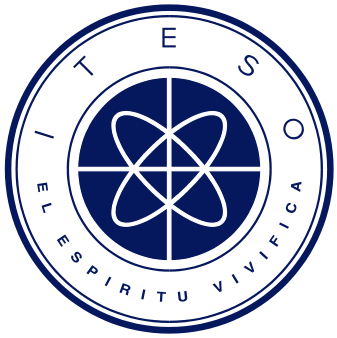
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 1. MANEJO DE APUNTADORES

Autor: Vázquez Martínez, Ignacio Itamar
Expediente: is714533

Funcionalidad: 60 pts
Pruebas: 20 pts
Presentación: 5 pts

24 de mayo de 2018. Tlaquepaque, Jalisco,

No se cumplió con los requerimientos funcionales de la tarea.

Todas las figuras e imagenes deben tener un título y utiilizar una leyenda que incluya número de la imagen ó figura y una descripción de la misma. Adicionalmente, debe de existir una referencia a la imagen en el texto.

La documentación de pruebas implica:
1) Descripción del escenario de cada prueba
2) Ejecución de la prueba
3) Descripción y análisis de resultados.

Instrucciónes para entrega de tarea

Es ***IMPRESCINDIBLE*** apegarse a los formatos de entrada y salida que se proveen en el ejemplo y en las instrucciones.

Esta tarea, como el resto, se entregará de la siguiente manera:

- **Reporte:** vía *moodle* en **un archivo PDF**.
- **Código:** vía su repositorio **Github**.

La evaluación de la tarea comprende:

- 10% para la presentación
- 60% para la funcionalidad
- 30% para las pruebas

Es necesario responder el apartado de conclusiones, pero no se trata de llenarlo con paja. Si no se aprendió nada al hacer la práctica, es preferible escribir eso.

Objetivo de la actividad

El objetivo de la tarea es que el alumno aplique los conocimientos y habilidades adquiridos en el tema de apuntadores para la resolución de problemas utilizando el lenguaje ANSI C.

Descripción del problema

Denisse estudia una ingeniería en una universidad de excelencia, donde constantemente invitan a sus estudiantes a evaluar el desempeño académico de los profesores. Cuando Denisse esta inscribiendo asignaturas para su próximo semestre, descubre que tiene diversas opciones con profesores que no conoce, entonces, decide crear un aplicación que le ayude a ella, y a sus compañeros a seleccionar grupos acorde a los resultados de las evaluaciones de los profesores.

Para iniciar, Denisse solicitó apoyo a traves de Facebook para que sus compañeros de toda la Universidad le apoyaran en la asignación de calificaciones de los profesores. Esto en base a sus experiencias previas en los diversos cursos. La respuesta que obtuvo fue 2 listas de profesores evaluados, la primer lista correspondia a profesores que imparten clases en Ingenierías y la segunda contenia a todos los profesores que imparten clases en el resto de las carreras.

Debido a que Denisse, le gusta programar, decidio crear una pequeña aplicación que le permitiera capturar los datos de los profesores y posteriormente le imprimiera una sola lista con todos los profesores ordenados acorde a su calificación. Lamentablemente, debido a que Denisse salio de viaje, no pudo terminar el programa. Tu tarea es ayudar a Denisse para completar el código.

Código escrito por Denisse

Importante: no modificar el código escrito por Denisse, solamente terminar de escribir el código e implementar las funciones.

```
typedef struct{
        char nombre[15];
        float calificacion;
} Profesor;

float averageArray(Profesor ______  , int _____);
void readArray(Profesor ________, int _____);
void mergeArrays(Profesor _____ , int _____, Profesor ____, int _____, Profesor _____, int ____);
void sortArray(Profesor _____, int ____);
void printArray(Profesor ____ , int _____);
```

```
void main(){
        Profesor arr1[20];  //Primer arreglo
        Profesor arr2[20];  //Segundo arreglo
        Profesor arrF[40];  //Arreglo final, con elementos fusionados y ordenados
        int n1, n2; //Longitud de los arreglos


        readArray(_______________); //leer el primer arreglo


        readArray(_______________); //leer el segundo arreglo


        mergeArrays(_____________________________);  //Fusionar los dos arreglos en un tercer arreglo

        sortArray(_____________);  //Ordenar los elementos del tercer arreglo, recuerde que pueden
                                   //existir profesores repetidos

        printArray(____________);   //Imprimir el resultado final

        return 0;
}
```

Descripción de la entrada del programa

El usuario ingresara dos listas con máximo 20 elementos (profesores: nombre y calificación). Antes de indicar, uno por uno los datos de los profesores, el usuario debe indicar la cantidad de elementos de la respectiva lista. Así lo primero que introducirá será la cantidad (n1) de elementos de la primer lista (arr1), y en seguida los datos de los profesores de la lista; posteriormente, la cantidad (n2) de elementos de la segunda lista (arr2), seguida por los profesores de los profesores correspondientes.

Ejemplo de entrada:

2
Roberto 7.8
Carlos 8.3

4
Oscar 8.3
Miguel 9.4
Diana 9.5
Oscar 8.5

Descripción de la salida

La salida del programa deberá ser sencillamente la impresión de una lista de profesores y su respectiva calificación (ordenados en orden descendiente, separados por un salto de línea). ¿Qué sucede si tenemos dos o más veces el registro de un profesor? La lista final, deberá mostrar sólo una vez a ese profesor y el promedio de sus calificaciones.

Ejemplo de la salida:

Diana	9.5
Miguel	9.4
Oscar	8.4
Carlos	8.3
Roberto	7.8

SOLUCIÓN DEL ALUMNO, PRUEBAS Y CONCLUSIONES

Código fuente:

<<Copie y pegue su código fuente aquí.>>

```
//
//  main.c
//  Tarea1
//
//  Created by Itamar Vazquez on 28.05.18.
//  Copyright © 2018 Ignacio Itamar Vazquez Martinez. All rights reserved.
//

#include <stdio.h>
#include <string.h>
#define enter printf("\n");
typedef struct{
    char nombre[15];
    float calificacion;
    int times_repeated;
} Profesor;

void readArray(Profesor array[20], int x);
void print(Profesor array[], int x);
void print_Final(Profesor array[40], int x);
void merge_Arrays(Profesor arrayFinal[40], Profesor array1[20], Profesor array2[20]);
void sort_by_name(Profesor arrayFinal[40]);
void merge_Duplicates(Profesor arrayFinal[40]);
void sort_by_Score(Profesor arrayFinal[40]);
void make_Ordered_Array(Profesor arrayFinal[40]);
//void add(Profesor arrayFinal[40], Profesor arrayFF [40], int k);
Profesor arr1[20];  //Primer arreglo
Profesor arr2[20];  //Segundo arreglo
Profesor arrF[40];  //Arreglo, con elementos fusionados
Profesor arrFF[40]; //Arreglo final, con elementos ordenados
int main(){

    int n1;
    int n2;
    int n3 = 40;
    int i; //Longitud de los arreglos
    //This code is here to tell that all cells in lista are empy at the beginning
    for(i=0;i<20;i++){
        (*(arr1+i)).calificacion = 0;
        strcpy((*(arr1+i)).nombre, "empty");
    }
    for(i=0;i<20;i++){
        (*(arr2+i)).calificacion = 0;
        strcpy((*(arr2+i)).nombre, "empty");
    }
    for(i=0;i<40;i++){
```

```
        (*(arrF+i)).calificacion = 0;
        strcpy((*(arrF+i)).nombre, "empty");
        (*(arrF+i)).times_repeated = 1;
    }
    scanf("%d", &n1);
    readArray(arr1,n1);//leer el primer arreglo
    scanf("%d", &n2);
    readArray(arr2, n2);//leer el segundo arreglo

    merge_Arrays(arrF, arr1, arr2);// Combinar los dos arreglos en el tercero
    sort_by_name(arrF); // Acomodar los nombres por orden alfabetico
    merge_Duplicates(arrF); // Borrar los duplicados, y hace el promedio del profesor
    sort_by_Score(arrF); // Ordenar el arreglo por puntajes de maestros
    print_Final(arrF, n3); // Imprimir la tabla ordenada al final
    return 0;
}



//Functions

void readArray(Profesor array[20], int x){
    int i;
    Profesor *pointer = array;
    for(i=0;i<x;i++){
        scanf("%s   %f", (*(pointer+i)).nombre, &(*(pointer+i)).calificacion);
    }
}


void print(Profesor array[], int x){
    int i;
    Profesor *pointer;
    pointer = array;
    for(i=0;i<x;i++){
        enter
        printf("%d Name: ", i);printf("%s\t",(*(pointer+i)).nombre);
        printf("Score: ");printf("%f", (*(pointer+i)).calificacion);enter
    }
    enter
}
void print_Final(Profesor array[40], int x){
    int i;
    Profesor *pointer;
    pointer = array;
    enter
    enter enter
    for(i=0;i<x;i++){
        if((strcmp((*(pointer+i)).nombre,"empty")!=0) ){
            printf("%s\t",(*(pointer+i)).nombre);
            printf("%.1f", (*(pointer+i)).calificacion);enter
        }
    }
}

void merge_Arrays(Profesor arrayFinal[40], Profesor array1[20], Profesor array2[20]){
    int i;
    for(i=0; i<20; i++){
        strcpy((*(arrayFinal+i)).nombre,(*(array1+i)).nombre);
        (*(arrayFinal+i)).calificacion = (*(array1+i)).calificacion;
    }
    for(i=20; i<40; i++){
        strcpy((*(arrayFinal+i)).nombre,(*(array2+(i-20))).nombre);
        (*(arrayFinal+i)).calificacion = (*(array2+(i-20))).calificacion;
    }
}
void sort_by_name(Profesor arrayFinal[40]){
    int i,j;
    Profesor temp;
    for(i=0;i<40;i++){
        for(j=0;j<39;j++){
            if((*(arrayFinal+j)).nombre[0]>(*(arrayFinal+(j+1))).nombre[0]){
```

```
                //Copiar a temp
                strcpy(temp.nombre, (*(arrayFinal+(j+1))).nombre);
                temp.calificacion =(*(arrayFinal+(j+1))).calificacion;
                //anterior copiar a siguiente
                strcpy((*(arrayFinal+(j+1))).nombre, (*(arrayFinal+j)).nombre);
                (*(arrayFinal+(j+1))).calificacion = (*(arrayFinal+j)).calificacion;
                // temp copiar a anterior
                strcpy((*(arrayFinal+j)).nombre, temp.nombre);
                (*(arrayFinal+j)).calificacion = temp.calificacion;
            }
            if((*(arrayFinal+j)).nombre[0]==(*(arrayFinal+(j+1))).nombre[0]){
                if((*(arrayFinal+j)).nombre[1]>(*(arrayFinal+(j+1))).nombre[1]){
                    //Copiar a temp
                    strcpy(temp.nombre, (*(arrayFinal+(j+1))).nombre);
                    temp.calificacion =(*(arrayFinal+(j+1))).calificacion;
                    //anterior copiar a siguiente
                    strcpy((*(arrayFinal+(j+1))).nombre, (*(arrayFinal+j)).nombre);
                    (*(arrayFinal+(j+1))).calificacion = (*(arrayFinal+j)).calificacion;
                    // temp copiar a anterior
                    strcpy((*(arrayFinal+j)).nombre, temp.nombre);
                    (*(arrayFinal+j)).calificacion = temp.calificacion;

                }
                if((*(arrayFinal+j)).nombre[1]==(*(arrayFinal+(j+1))).nombre[1]){
                    if((*(arrayFinal+j)).nombre[2]>(*(arrayFinal+(j+1))).nombre[2]){
                        //Copiar a temp
                        strcpy(temp.nombre, (*(arrayFinal+(j+1))).nombre);
                        temp.calificacion =(*(arrayFinal+(j+1))).calificacion;
                        //anterior copiar a siguiente
                        strcpy((*(arrayFinal+(j+1))).nombre, (*(arrayFinal+j)).nombre);
                        (*(arrayFinal+(j+1))).calificacion = (*(arrayFinal+j)).calificacion;
                        // temp copiar a anterior
                        strcpy((*(arrayFinal+j)).nombre, temp.nombre);
                        (*(arrayFinal+j)).calificacion = temp.calificacion;
                    }
                }
            }
        }
    }
}

void merge_Duplicates(Profesor arrayFinal[40]){
    int i;
    for(i=0;i<39;i++){
        if(strcmp((*(arrayFinal+i)).nombre, (*(arrayFinal+(i+1))).nombre)==0){
            (*(arrayFinal+(i+1))).calificacion += (*(arrayFinal+(i))).calificacion;
            (*(arrayFinal+(i+1))).times_repeated += (*(arrayFinal+(i))).times_repeated;
            (*(arrayFinal+i)).calificacion = (float)0;
            (*(arrayFinal+i)).times_repeated = (int)1;
            strcpy((*(arrayFinal+i)).nombre, "empty");
        }
    }
    for(i=0;i<39;i++){
        float r = arrayFinal[i].calificacion;
        float b = arrayFinal[i].times_repeated;
        (*(arrayFinal+i)).calificacion = r/b;
        (*(arrayFinal+i)).times_repeated = 1;
        }
}

void sort_by_Score(Profesor arrayFinal[40]){
    int i,j;
    Profesor temp;
    for(i=0;i<40;i++){
        for(j=0;j<39;j++){
            if(arrayFinal[j].calificacion < arrayFinal[j+1].calificacion){
                strcpy(temp.nombre, (*(arrayFinal+(j+1))).nombre);
                temp.calificacion =(*(arrayFinal+(j+1))).calificacion;
                //siguiente copiar a anterior
                strcpy((*(arrayFinal+j+1)).nombre, (*(arrayFinal+(j))).nombre);
                (*(arrayFinal+j+1)).calificacion = (*(arrayFinal+(j))).calificacion;
                // temp copiar al suguiente
                strcpy((*(arrayFinal+j)).nombre, temp.nombre);
                (*(arrayFinal+j)).calificacion = temp.calificacion;
```

```
            }

        }
    }
}
```

Ejecución:

<<Inserte capturas de pantalla de una ejecución donde la primera lista tenga la suma mayor y de otra donde la segunda lista tenga la suma mayor.>>

```
2
Roberto 7.8
Carlos  8.3
4
Oscar   8.3
Miguel  9.4
Diana   9.5
Oscar   8.5



Diana   9.5
Miguel  9.4
Oscar   8.4
Carlos  8.3
Roberto 7.8
Program ended with exit code: 0
```

```
3
Roberto 1
Juan 1.3
Pedro 1.1
4
Oscar 10
Miguel  9.7
Diana   10
Oscar   10



Diana   10.0
Oscar   10.0
Miguel  9.7
Juan    1.3
Pedro   1.1
Roberto 1.0
Program ended with exit code: 0
```

```
5
Pedro   10
Juan    8.5
Pancho  9.7
Federico    3.9
Igancio 7.8
3
Juan    3
Pablo   1
Maria   1



Pedro   10.0
Pancho  9.7
Igancio 7.8
Juan    5.8
Federico    3.9
Maria   1.0
Pablo   1.0
Program ended with exit code: 0
```

Conclusiones (obligatorio):

- ✓ Lo que aprendí con esta práctica. Lo que ya sabía.
- ✓ Lo que me costó trabajo y cómo lo solucioné.
- ✓ Lo que no pude solucionar.

Con esta practica aprendí que es muy importante probar el codigo en partes para asegurarte que funcione bien, a no pasar por alto pequeños errores en la salidad de datos. Por ejemplo en mi tarea me paso que decia

nan y lo ignore y resulto que estaba dividiendo entre cero a un numero. Y gaste mucho tiempo en ese error por no checar todas las partes de mi codigo. Me costo trabajo quitar los profesores repetidos de mi arreglo. Lo que hice fue ordenarlos por orden alfabetico y sumarlos a la derecha mientras fueran iguales. Los que iban quedando a la izquierda los borraba. Finalmente pude completar el ejercicio.