

# 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 APUNTADES

Autor: Santana Hernández, Andrea Anhi

24 de mayo de 2018. Tlaquepaque, Jalisco,

Funcionalidad: 60 pts  
Pruebas: 25 pts  
Presentación: 5 pts

Todas las figuras e imágenes deben tener un título y utilizar 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.

## Instrucciones para entrega de tarea

Es **IMPRESINDIBLE** 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:

```
/*
 * Tarea 1.c
 *
 * Created on: May 28, 2018
 * Author: Anahí
 */

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

int i=0;

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

float averageArray(float y, float x);
void readArray(Profesor *pa, int n);
void mergeArrays(Profesor *pa1, int n1, Profesor *pa2, int n2, Profesor *paF, int nF);
void sortArray(Profesor *paF, int nF);
void printArray(Profesor *pa, int n);

int main(){
    setbuf(stdout, NULL);
    Profesor arr1[20]; //Primer arreglo
    Profesor arr2[20]; //Segundo arreglo
    Profesor arrF[40]; //Arreglo final, con elementos fusionados y ordenados
    Profesor *pa1=arr1;
    Profesor *pa2=arr2;
    Profesor *paF=arrF;
    int n1, n2, nF; //Longitud de los arreglos

    scanf("%d",&n1);
    readArray(pa1,n1); //leer el primer arreglo
    scanf("%d",&n2);
    readArray(pa2,n2); //leer el segundo arreglo
    nF=n1+n2;
    mergeArrays(pa1,n1,pa2,n2,paF,nF); //Fusionar los dos arreglos en un tercer
    arreglo
}
```

```

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

        //printArray(paF,nF); //Imprimir el resultado final

        return 0;
    }
    void readArray(Profesor *pa, int n)
    {
        float cal;
        for(i=0;i<n;i++){
            fflush(stdin);
            scanf("%s", (pa+i)->nombre);
            scanf("%f",&cal);
            (pa+i)->calificacion =cal;
        }
    }
    void printArray(Profesor *pa , int n){
        for(i=0;i<n;i++){
            printf("%s %0.1f \n", (pa+i)->nombre, (pa+i)->calificacion);
        }
    }

    void mergeArrays(Profesor *pa1 , int n1, Profesor *pa2, int n2, Profesor *paF, int nF){
        for (i=0;i<nF;i++){
            if (i<n1){
                strcpy((paF+i)->nombre, (pa1+i)->nombre);
                (paF+i)->calificacion=(pa1+i)->calificacion;
            }
            else{
                strcpy((paF+i)->nombre, (pa2+i-n1)->nombre);
                (paF+i)->calificacion=(pa2+i-n1)->calificacion;
            }
        }
    }

    void sortArray(Profesor *paF, int nF){
        Profesor arrC[40];
        Profesor ordenados[40];
        char new[15]="\0";
        char h[15],h1[15];
        int j,o=0, k;
        float x=1,y=0;

        for (i=0;i<nF;i++){
            x=1;
            if (strcmp((paF+i)->nombre, new)!=0){
                strcpy(arrC[o].nombre, (paF+i)->nombre);
                y=(paF+i)->calificacion;
                // printf("i= %d \n",i);
                // printf("y= %f \n",y);
                //arrC[o].calificacion= averageArray(y,x);
                o++;

                for(j=1;j<(nF-i);j++){

```

```

        k=j+i;
        strcpy(h,((paF+i)->nombre));
        strcpy(h1,((paF+k)->nombre));
        // printf("(paF+%d)-
>nombre =%s\n",i,h);
        // printf("(paF+%d)-
>nombre =%s\n",k,h1);
        // printf("***\n");

        if (strcmp(h, h1)==0){
            //printf("iguales\n");
            x++;
            y=y+((paF+k)->calificacion);
            //printf("y= %f \n",y);
            strcpy(((paF+k)->nombre),new);
        }
        //printf("y= %f \n",y);
        //printf("x= %f \n",x);
    }

    arrC[o-1].calificacion= averageArray(y,x);
}
//printArray(arrC,o);
//printArray(paF,nF);
//Profesor *paC=arrC;

for (i=0; i<o; i++ )
{
    ordenados[i].calificacion=arrC[i].calificacion;
    strcpy(ordenados[i].nombre,arrC[i].nombre);
}

for (i=0; i<o; i++ )
{
    for (j=0; j<o; j++ )
    {
        if (arrC[i].calificacion>ordenados[j].calificacion)
        {
            ordenados[i].calificacion=arrC[j].calificacion;
            strcpy(ordenados[i].nombre,arrC[j].nombre);

            ordenados[j].calificacion=arrC[i].calificacion;
            strcpy(ordenados[j].nombre,arrC[i].nombre);

            arrC[i].calificacion=ordenados[i].calificacion;
            strcpy(arrC[i].nombre,ordenados[i].nombre);

            arrC[j].calificacion=ordenados[j].calificacion;
            strcpy(arrC[j].nombre,ordenados[j].nombre);
        }
    }
}
printArray(ordenados,o);
}

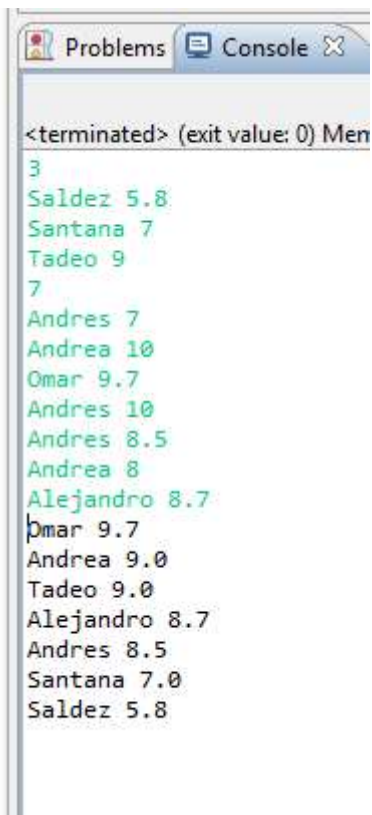
```

## Ejecución:

Ejecución donde la primera lista tenga la suma mayor

```
6
Enrique 10
Juan 8
Enrique 9.5
Omar 8.7
Juan 9.2
Enrique 8.9
3
Salma 6.7
Isaac 7.6
Andrea 8.3
Enrique 9.5
Omar 8.7
Juan 8.6
Andrea 8.3
Isaac 7.6
Salma 6.7
```

La segunda lista tenga la suma mayor



```
<terminated> (exit value: 0) Men
3
Saldez 5.8
Santana 7
Tadeo 9
7
Andres 7
Andrea 10
Omar 9.7
Andres 10
Andres 8.5
Andrea 8
Alejandro 8.7
Omar 9.7
Andrea 9.0
Tadeo 9.0
Alejandro 8.7
Andres 8.5
Santana 7.0
Saldez 5.8
```



### Conclusiones (obligatorio):

En esta práctica reforcé mis conocimientos en estructuras ya que en programación estructurada no las había utilizado mucho y no sabía bien como. Usar los apuntadores con estructuras fue algo nuevo, pero no me costó trabajo por que su sintaxis es sencilla y fácil de utilizar. Me cuesta más trabajo usar los apuntadores para datos que no están dentro de arreglos o estructuras.

Lo que más me costó trabajo fue quitar los nombre repetidos y sacar su promedió, lo solucioné primero quitando los repetidos y sacando el promedió y después acomodando eso de mayor a menor. Logré solucionar todos los problemas buscando en las presentaciones de las clases la correcta sintaxis y pensando en algoritmos que pudieran solucionarlo.