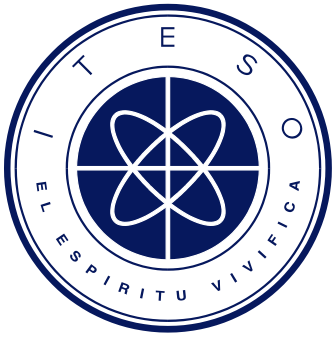
# INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

---

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES




# PROGRAMACIÓN CON MEMORIA DINÁMICA

## TAREA 1. MANEJO DE APUNTADORES

Autor: Soto Pérez Omar

24 de mayo de 2018. Tlaquepaque, Jalisco,

## 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:

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

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

Profesor* averageArray(Profesor arr[40] ,Profesor auxiliar[40], int n);
void readArray(Profesor arr[20], int n);
void mergeArrays(Profesor arr[40], int n, Profesor arr1[20], int n1, Profesor
arr2[20], int n2);
void sortArray(Profesor arr[40], int n);
void printArray(Profesor arr[20], int n);


int main()
{
	setvbuf(stderr, NULL, _IONBF, 0);
	setvbuf(stdout, NULL, _IONBF, 0);
	Profesor arr1[20];  //Primer arreglo
	Profesor arr2[20];  //Segundo arreglo
	Profesor arrF[40];  //Arreglo final, con elementos fusionados y ordenados
	Profesor final[40];
	int n1, n2; //Longitud de los arreglos

	printf("Ingrese el numero de profesores (Ingenieria) [1,20]\n");
	scanf("%d",&n1);
	while(n1<1 || n1>20)
	{
		printf("Ingrese el numero de profesores (Ingenieria) [1,20]\n");
		scanf("%d",&n1);
	}
	readArray(arr1,n1); //leer el primer arreglo

	printf("Ingrese el numero de profesores (Licenciatura) [1,20]\n");
	scanf("%d",&n2);
```

```
	while(n2<1 || n2>20)
	{
		printf("Ingrese el numero de profesores (Licenciatura) [1,20]\n");
		scanf("%d",&n2);
	}
	readArray(arr2,n2); //leer el segundo arreglo

	mergeArrays(arrF,n1+n2,arr1,n1,arr2,n2);  //Fusionar los dos arreglos en un tercer arreglo
	averageArray(arrF,final,n1+n2);
	sortArray(final,n1+n2);  //Ordenar los elementos del tercer arreglo, recuerde que pueden  //existir profesores repetidos
	printArray(final,n1+n2);   //Imprimir el resultado final
	return 0;
}
void readArray(Profesor arr[20], int n)
{
	Profesor * ap=arr;
	for(int i=0;i<n;i++)
	{
		scanf("%s[^\n]",(ap+i)->nombre);
		scanf("%f",&(ap+i)->calificacion);
	}

}
void printArray(Profesor arr[20], int n)
{
	Profesor * ap=arr;
	for(int i=0;i<n;i++)
	{
		if(strcmp((ap+i)->nombre," ")==1)
		{
			printf("%s\t\t",(ap+i)->nombre);
			printf("%.2f\n",(ap+i)->calificacion);
		}
	}
}
void mergeArrays(Profesor arr[40], int n, Profesor arr1[20], int n1, Profesor arr2[20], int n2)
{
	//Profesor * ap=arr;
	//Profesor * ap1=arr1;
	//Profesor * ap2=arr2;
	int i,j;
	for(i=0;i<n1;i++)
		arr[i]=arr1[i];
		//strcpy((arr+i)->nombre,(arr1+i)->nombre);
		//(arr+i)->calificacion=(arr1+i)->calificacion;
	for(j=0;j<n2;j++)
		arr[i++]=arr2[j];
		//i++;
		//strcpy((arr+i)->nombre,(arr2+j)->nombre);
		//(arr+i)->calificacion=(arr2+j)->calificacion;
}
Profesor* averageArray(Profesor arr[40] ,Profesor auxiliar[40], int n)
```

```
{
	int i,j;
	float acum,rep,cf;
	for(i=0;i<n;i++)
	{
		acum=0;
		rep=1;
		for(j=0;j<n;j++)
		{
			if(strcmp(arr[i].nombre,arr[j].nombre)==0 && i!=j &&
strcmp(arr[i].nombre," ")==1)
			{
				acum+=arr[i].calificacion;
				rep++;
				strcpy(arr[j].nombre," ");
				arr[j].calificacion=0;
			}
		}
		cf=(acum+arr[i].calificacion)/rep;
		strcpy(auxiliar[i].nombre,arr[i].nombre);
		auxiliar[i].calificacion=cf;
	}
	return auxiliar;
/*	Profesor * ap=arr;
	Profesor * aux=auxiliar;
	for(i=0;i<n;i++)
	{
		acum=0;
		rep=1;
		for(j=0;i<n;j++)
		{
			if(strcmp((*(ap+i)).nombre,(ap+j)->nombre)==0 && i!=j &&
strcmp((ap+i)->nombre," ")==1)
			{
				acum+=(ap+j)->calificacion;
				rep++;
				strcpy((ap+j)->nombre," ");
				(ap+j)->calificacion=0;
			}
		}
		strcpy((aux+i)->nombre,(ap+i)->nombre);
		(aux+i)->calificacion=(acum+(arr+i)->calificacion)/rep;
	}*/

}

void sortArray(Profesor arr[], int n)
{
	int h, i;
	int huboCambios; //bandera
	Profesor aux;
	for(h=0; h<n-1; h++)
	{
		huboCambios = 0;
		for(i=0; i<n-1-h; i++)
```

```
                        if(arr[i].calificacion > arr[i+1].calificacion)
                        {
                                aux = arr[i];
                                arr[i] = arr[i+1];
                                arr[i+1] = aux;
                                huboCambios = 1;
                        }
                if(huboCambios==0)
                        break;
        }

}
```

## Ejecución:

```
<terminated> (exit value: 0) tarea 2.exe [C/C++ Application] C:\Users\OM
Ingrese el numero de profesores (Ingenieria) [1,20]
3
omar 8.65
fer 8.21
edgar 6.3
Ingrese el numero de profesores (Licenciatura) [1,20]
2
omar 10
juan 8.12
edgar           6.30
juan            8.12
fer             8.21
omar            8.65
```

```
<terminated> (exit value: 0) tarea 2.exe [C/C++ Application] C:\Users\OMA
Ingrese el numero de profesores (Ingenieria) [1,20]
2
omar 10
juan 5.6
Ingrese el numero de profesores (Licenciatura) [1,20]
4
omar 10
juan 9
edgar 8.3
luis 5
luis            5.00
juan            5.60
edgar           8.30
omar            10.00
```

## Conclusiones (obligatorio):

Esta practica me hizo darme cuenta de que necesito practicar mas el uso de apuntadores, pero al mismo tiempo me ayudo a consolidar el tema, que hasta el momento no me quedaba tan claro, en general toda la practica me costo trabajo y finalmente con ayuda del profesor y algunos compañeros para solucionar mis dudas. En general no pude usar todos los punteros que quería pues en ciertas funciones utilizaba arreglos en lugar de punteros, pues al usar punteros con la misma lógica el programa simplemente no funcionaba, tampoco pude sacar el promedio correctamente.

**lo que se encuentra en comentarios en cada función es lo mismo, pero en apuntadores, pero por alguna razón al momento de cambiarlo el programa dejaba de funcionar en plena ejecución**