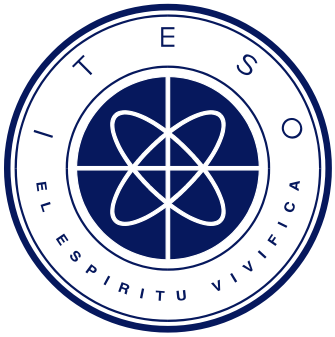
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA ELECTRÓNICA



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 1. MANEJO DE APUNTADORES

Autor: Galeana Torres, Fernanda

30 de mayo de 2018. Tlaquepaque, Jalisco,

Presentación: 5 pts

Funcionalidad: 60 pts

Pruebas: 25 pts

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.

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-Programación con Memoria Dinámica
 * 30/05/2018
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

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

float averageArray(Profesor *a,int *n);
void readArray(Profesor *a,int *n);
void mergeArrays(Profesor *a,int *an,Profesor *b,int *bn,Profesor *c,int *cn);
void sortArray(Profesor *a,int *n);
void printArray(Profesor *a,int *n);

int main(){
	setvbuf(stderr, NULL, _IONBF, 0);
	setvbuf(stdout, NULL, _IONBF, 0);
	Profesor arr1[20];
	Profesor arr2[20];
	Profesor arr3[40];
	Profesor *a1,*a2,*a3;
	a1=arr1;
	a2=arr2;
	a3=arr3;

	int n1,n2,n3;
	printf("Profesores de Ingenierías\n¿Cuántos elementos desea ingresar?");
	scanf("%d",&n1);
	readArray(a1,&n1);
	printf("Profesores de otras áreas\n¿Cuántos elementos desea ingresar?");
	scanf("%d",&n2);
	readArray(a2,&n2);
	n3=n1+n2;
	mergeArrays(a1,&n1,a2,&n1,a3,&n3);
```

```
	sortArray(a3,&n3);
	printArray(a3,&n3);

	return 0;

}
float averageArray(Profesor *a,int *n){
	int N=*n;
	float promedio=0;
	for(int i=0;i<N;i++)
		promedio+=(a+i)->calificacion;
	return promedio/N;
}
void readArray(Profesor *a,int *n){
	int N=*n;
	for(int i=0;i<N;i++)
	{
		printf("Nombre: ");
		fflush(stdin);
		gets((a+i)->nombre);
		printf("Calificación: ");
		scanf("%f",&(a+i)->calificacion);
	}

}
void mergeArrays(Profesor *a,int *an,Profesor *b,int *bn,Profesor *c,int *cn){
	int k=0;
	int CN=*cn;
	int AN=*an;
	for(int i=0;i<CN;i++){
		if(i<AN){
			strcpy((c+i)->nombre,(a+i)->nombre);
			(c+i)->calificacion=(a+i)->calificacion;
		}
		else{
			strcpy((c+i)->nombre,(b+k)->nombre);
			(c+i)->calificacion=(b+k)->calificacion;
			k++;
		}
	}

}
void sortArray(Profesor *a,int *n){
	int N=*n;
	Profesor temp;
	Profesor *t;
	t=&temp;
	int c;
	//Acomodar por nombres
	for(int k=0;k<N;k++)
	{
		c=1;
		for(int l=k+1;l<N;l++)
		{
			if(strcmp((a+k)->nombre,(a+l)->nombre)==0)
```

```
                        {
                                strcpy(t->nombre,(a+k+c)->nombre);
                                t->calificacion=(a+k+c)->calificacion;
                                strcpy((a+k+c)->nombre,(a+l)->nombre);
                                (a+k+c)->calificacion=(a+l)->calificacion;
                                strcpy((a+l)->nombre,t->nombre);
                                (a+l)->calificacion=t->calificacion;
                                c++;
                        }
                }
                if(c>1)
                {
                        (a+k)->calificacion=averageArray((a+k),&c);
                        for(int i=k+1;i<N;i++)
                                        {
                                                strcpy((a+i)->nombre,(a+i+c-1)->nombre);
                                                (a+i)->calificacion=(a+i+c-1)-
>calificacion;
                                        }
                        N-=(c-1);
                }
        }

        for(int i=0;i<N;i++)
                for(int j=0;j<N-1;j++)
                {
                        if((a+j)->calificacion<(a+j+1)->calificacion)
                        {
                                strcpy(t->nombre,(a+j)->nombre);
                                t->calificacion=(a+j)->calificacion;
                                strcpy((a+j)->nombre,(a+j+1)->nombre);
                                (a+j)->calificacion=(a+j+1)->calificacion;
                                strcpy((a+j+1)->nombre,t->nombre);
                                (a+j+1)->calificacion=t->calificacion;
                        }
                }
        *n=N;
}
void printArray(Profesor *a,int *n){
        int N=*n;
        for(int i=0;i<N;i++){
                printf("%s\t%.2f\n",(a+i)->nombre,(a+i)->calificacion);
        }

}
```

Ejecución:

La primera lista tiene la suma mayor:

```
Profesores de Ingenierías
¿Cuántos elementos desea ingresar?4
Nombre: Omar
Calificación: 9.5
Nombre: Leticia
Calificación: 8
Nombre: Pablo
Calificación: 7.5
Nombre: Omar
Calificación: 10
Profesores de otras áreas
¿Cuántos elementos desea ingresar?5
Nombre: Elena
Calificación: 8
Nombre: Julieta
Calificación: 9
Nombre: Elena
Calificación: 7.5
Nombre: Pedro
Calificación: 8.2
Nombre: Julieta
Calificación: 9.2
Omar    9.75
Julieta 9.10
Pedro   8.20
Leticia 8.00
Elena   7.75
Pablo   7.50
```

La segunda lista tiene la suma mayor:

```
Profesores de Ingenierías
¿Cuántos elementos desea ingresar?4
Nombre: Omar
Calificación: 9.5
Nombre: Leticia
Calificación: 8
Nombre: Pablo
Calificación: 7.5
Nombre: Leticia
Calificación: 10
Profesores de otras áreas
¿Cuántos elementos desea ingresar?5
Nombre: Elena
Calificación: 9.8
Nombre: Julieta
Calificación: 10
Nombre: Elena
Calificación: 10
Nombre: Pedro
Calificación: 8.5
Nombre: Julieta
Calificación: 8
Elena   9.90
Omar    9.50
Leticia 9.00
Julieta 9.00
Pedro   8.50
Pablo   7.50
```

Conclusiones:

El uso de apuntadores puede resultar un tanto confuso de principio pues al trabajar con ellos en arreglos es necesario irlos desplazando; sin embargo, con la práctica se va comprendiendo mejor la forma en que esto debe hacerse, y esta tarea me ayudó a en ello, así como a aprender a emplearlos para cambiar dentro de una función el valor que contiene un apuntador utilizado en otras funciones.

Una dificultad que se me presentó tiene relación con esto último, pues al juntar varios datos de entrada en uno solo (es decir, poner los nombres repetidos y el promedio de estos en un solo espacio) la cantidad de datos totales a imprimir se reduce, así que la N principal que delimita el ciclo para imprimir uno por uno también se ve afectada. Para esto, reduje la N dentro del ciclo que comparaba los nombres y los juntaba en uno solo y

asigné ese valor al apuntador que usaban las siguientes funciones, modificando lo que había dentro.

Por otro lado, también encontré como obstáculo el juntar todos los nombres en un solo dato de la lista junto con el promedio, y posteriormente recorrer los demás datos para eliminar los repetidos. La solución que hallé fue comparar cada nombre con el resto de la lista, y acomodar todas las repeticiones juntas para así facilitar el obtener el promedio mediante una función a la que se le pasaba un apuntador al lugar donde se encontraba la primera repetición, y un contador c inicializado en 1 que incrementaba cada vez que se había hallado el nombre repetido. Esta función iba sumando los valores que se encontraban como calificación desde la dirección del apuntador hasta c-1, y lo dividía entre c devolviendo este valor. Finalmente, si c había aumentado, los datos a la derecha del que se estaba evaluando en primer lugar se recorrían c espacios a la izquierda.

Aparte de eso, no hubo dificultades mayores que no pude solucionar, únicamente pequeños errores al manejar el recorrer los datos que, tras hacer las pruebas, pude captar y solucionar, comprendiendo mejor la aritmética de apuntadores y en qué puede beneficiar utilizarlos.