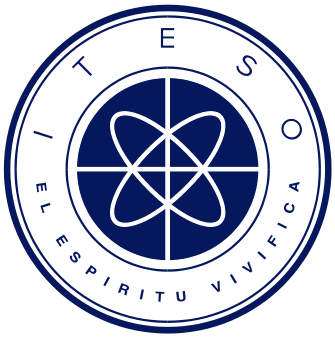
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 1. MANEJO DE APUNTADORES

Autor: Gallegos, José Isaac

24 de mayo de 2018. Tlaquepaque, Jalisco,

Presentación: 5 pts

Funcionalidad: 60 pts

Pruebas: 15 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 una 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 través 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 correspondía a profesores que imparten clases en Ingenierías y la segunda contenía a todos los profesores que imparten clases en el resto de las carreras.

Debido a que Denisse, le gusta programar, decidió 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 salió 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;

float averageArray(Profesor arrayR[], int n);
void readArray(Profesor arr[], int *n);
void mergeArrays(Profesor arr1[] , int* n1, Profesor arr2[], int* n2, Profesor
arrF[], int* nF);
void sortArray(Profesor arr[], int *n);
void printArray(Profesor arr[], int n);


int main(){                    //Error de sintaxis, se cambió a int
	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
	int n1, n2;               //Longitud de los arreglos
	int nF;                   //Longitud del arreglo final
	readArray(arr1,&n1);      //leer el primer arreglo
	readArray(arr2,&n2);      //leer el segundo arreglo
	//Fusionar los dos arreglos en un tercer arreglo
	mergeArrays(arr1,&n1,arr2,&n2,arrF,&nF);
	//Ordenar los elementos del tercer arreglo y promedia elementos repetidos
	sortArray(arrF,&nF);
	printArray(arrF,nF);   //Imprimir el resultado final

return 0;
}
float averageArray(Profesor arrayR[], int n){
	for(int k=1;k<n;k++)
		arrayR[0].calificacion+=arrayR[k].calificacion;
	arrayR[0].calificacion/=n;
	return(arrayR[0].calificacion);
}
void readArray(Profesor arr[], int *n){
	scanf("%d", n);
	for(int i=0;i<(*n);i++)
		scanf("%s %f", (arr[i].nombre), &(arr[i].calificacion));
}
```

```
void mergeArrays(Profesor arr1[] , int* n1, Profesor arr2[], int* n2, Profesor
arrF[], int* nF){
	for(int i=0; i<*n1; i++)
		arrF[i]=arr1[i];
	for(int i=0; i<*n2; i++)
		arrF[*n1+i]=arr2[i];
	*nF=*n1+*n2;
}
void sortArray(Profesor arr[], int *n){
	//Algoritmo para eliminar repetidos
	for(int i=0;i<(*n-1);i++){
		int repetidos=1;
		Profesor arrayR[20];
		arrayR[0]=arr[i];
		for(int j=i+1;j<*n;j++){
			if(strcmp(arr[i].nombre,arr[j].nombre)==0){
				//*Añadir al arreglo de repetidos
				arrayR[repetidos]=arr[j];
				repetidos++;
				//*Recorrer los elementos después del repetido
				for(int k=j;k<(*n-1);k++)
					arr[k]=arr[k+1];
				j--;
				--*n;  //*Restar 1 al total de elementos
			}
		}
		//Si hubo repetidos promediar la calificación
		if(repetidos>1)
			arr[i].calificacion=averageArray(arrayR,repetidos);
	}
	//Algoritmo para ordenar de mayor a menor, tipo ShellSort
	for (int inc=*n/2; inc>0;inc=(inc==2?1:(inc/2.2))){
		for(int i=inc;i<*n;i++){
			for(int j=i;
				j>=inc && arr[j-inc].calificacion<arr[j].calificacion;
					j-=inc){
				Profesor temp=arr[j];
				arr[j]=arr[j-inc];
				arr[j-inc]=temp;
			}
		}
	}
}
void printArray(Profesor arr[] , int n){
	for(int i=0;i<n;i++){
		printf("%s   %0.2f\n", arr[i].nombre, arr[i].calificacion);
	}
}
```

Ejecución:

```
9
Rosa 9
Ernesto 7
Victor 6
Javier 8
Rosa 4
Elena 10
Karina 7
Oscar 3
Karina 9
5
Mauricio 8
Susana 7
Norman 4
Mauricio 4
Mauricio 7
Elena           10.00
Javier          8.00
Karina          8.00
Ernesto         7.00
Susana          7.00
Rosa            6.50
Mauricio                6.33
Victor          6.00
Norman          4.00
Oscar           3.00
```

```
5
Mauricio 8
Susana 7
Norman 4
Susana 7
Susana 4
10
Javier 8
Rosa 4
Elena 10
Karina 7
Oscar 3
Elena 5
Oscar 4
Oscar 8
Elena 1
Javier 10
Javier          9.00
Mauricio                8.00
Karina          7.00
Susana          6.00
Elena           5.33
Oscar           5.00
Rosa            4.00
Norman          4.00
```

Conclusiones:

Esta tarea fue muy útil para practicar el funcionamiento básico de los apuntadores, utilizarlos en funciones y conocer algunas de sus posibles aplicaciones. No fue difícil realizarlo, lo que más se llegó a complicar fue cuidar la sintaxis a la hora de utilizar los apuntadores para referir al valor de las variables y la creación del algoritmo del programa sin cambiar las funciones y la estructura ya definida. Con esto pienso que el tema quedo bien aprendido.