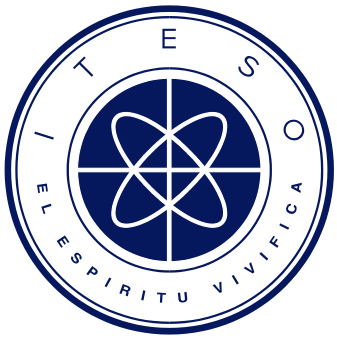
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 1. MANEJO DE APUNTADORES

Autor: Méndez Méndez Sergio Alejandro

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

24 de mayo de 2018. Tlaquepaque, Jalisco,

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:

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

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

void averageArray(Profesor array[], int tam);
void readArray(Profesor array[], int tam);
void mergeArrays(Profesor array1[], int tam1, Profesor array2[], int tam2, Profesor
arrayF[], int tamF);

void sortArray(Profesor array[], int tam);
void printArray(Profesor array[], int tam);

int nF;

int 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
	scanf("%d", &n1);
	readArray(arr1, n1); //leer el primer arreglo
	printf("\n");
	scanf("%d", &n2);
	readArray(arr2, n2); //leer el segundo arreglo
	nF = n1 + n2;
	printf("\n");
	mergeArrays(arr1, n1, arr2, n2, arrF, nF);  //Fusionar los dos arreglos en un tercer

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

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

	return 0;
}

void averageArray(Profesor array[], int tam){
	Profesor* pArray;
	pArray = array;
	float valores[20];
	for(int i=0; i<(tam-1); i++){
```

```
                int contador = 0;
                float valorPromedio = (pArray+i)->calificacion;
                for(int j=(i+1); j<tam; j++){
                        if(strcmp((pArray+i)->nombre, (pArray+j)->nombre) == 0){
                                valores[contador] = (pArray+j)->calificacion;
                                for(int k=j; k<(tam-1); k++) *(pArray+k) = *(pArray+k+1);
                                contador++;
                                tam--;
                                j--;
                        }
                }
                if(contador>0){
                        for(int m=0; m<contador; m++){
                                valorPromedio += valores[m];
                        }
                        valorPromedio /= (contador+1);
                        (pArray+i)->calificacion = valorPromedio;
                }
        }
        nF = tam;
}

void sortArray(Profesor array[], int tam){
        Profesor* apuntador;
        apuntador = array;
        Profesor* aux;
        Profesor arreglo[40];
        aux = arreglo;
        averageArray(array, tam);
        tam = nF;
        printf("\n%d\n", tam);
        for(int i=1; i<tam; i++){
                for(int j=0; j<(tam-i); j++){
                        if((apuntador+j)->calificacion < (apuntador+j+1)->calificacion){
                                *aux = *(apuntador+j+1);
                                *(apuntador+j+1) = *(apuntador+j);
                                *(apuntador+j) = *aux;
                        }
                }
        }
}

void readArray(Profesor array[], int tam){
        Profesor* apuntador;
        apuntador = array;
        for(int i=0; i<tam; i++){
                scanf("%s %f", (apuntador+i)->nombre, &(apuntador+i)->calificacion);
        }
}

void printArray(Profesor array[], int tam){
        Profesor* p;
        p = array;
        for(int x=0; x<tam; x++){
                printf("%s\t", (*(p+x)).nombre);
                printf("%.1f\n", (*(p+x)).calificacion);
        }
}
```

```
void mergeArrays(Profesor array1[], int tam1, Profesor array2[], int tam2, Profesor
arrayF[], int tamF){
        Profesor* pArray1;
        Profesor* pArray2;
        Profesor* pArrayF;
        pArray1 = array1;
        pArray2 = array2;
        pArrayF = arrayF;
        for(int i=0; i<tam1; i++){
                *(pArrayF + i) = *(pArray1 + i);
        }
        pArrayF = (pArrayF + tam1);
        for(int j=0; j<tam2; j++){
                *(pArrayF + j) = *(pArray2 + j);
        }

}
```

Ejecución:

Input

```
8
Jesus 7.8
Sergio 9.8
Marco 7.1
Jose 4.9
Jesus 7.1
Jesus 6.3
Marco 8.3
Carlos 5.7

6
Karla 7.4
Laura 8.5
Kevin 6.8
Andres 9.1
Andres 8.6
Karla 8.6
```

Output

```
9
Sergio  9.8
Andres  8.9
Laura   8.5
Karla   8.0
Marco   7.7
Jesus   7.1
Kevin   6.8
Carlos  5.7
Jose    4.9
```

Input

```
5
Kevin 7.4
Luis 8.1
Jose 6.2
Kevin 6.7
Jose 7.2

7
Laura 9.8
Ricardo 7.7
Laura 9.6
Laura 8.9
Andres 8.3
Karla 6.1
Ricardo 8.3
```

Output

```
7
Laura   9.4
Andres  8.3
Luis    8.1
Ricardo 8.0
Kevin   7.1
Jose    6.7
Karla   6.1
```

Conclusiones:

Para esta practica aprendi como se realiza la implementación de apuntadores a datos, arreglos y estructuras; esto me permitio comprender mejor como se utilizan estas herramientas y poder desarrollar la capacidad de usarlas en las futuras implementaciones. Esto va a ser de vital importancia ya que, según lo visto en la introducción de memoria dinamica, los apuntadores seran lo que más usaremos.

Ya conocia el como realizar registros y poderlos manipular según lo que se pedia, sin embargo, no conocia como se realizaba esto con apuntadores ni como se hacia la sintaxis para poder usar estos apuntadores. Pero esto lo pude aprender de las clases, ademas de la bibliografía con la que contamos, por lo que no fue tanto desafio.

Un problema que me surgió durante el desarrollo fue el como eliminar un elemento de un arreglo, pero esto lo solucione recorriendo los valores del arreglo un espacio, sin emargo esto me llevó a un problema que no pude solucionar, que fue el como usar apuntadores dentro de una funcion para poder modificar los valores de la variable a la que se le asigno el apuntador desde la funcion que la llamo, esto para poder manipular las veces que se ejecuta el ciclo for, esto no pude solucionarlo, quiza porque no implemente de forma correcta el apuntador, sin embargo encontre una solución parcial que fue usar una variable global a la que se le asigna el valor que necesito, no es la mejor solución pero fue la única con la que me funciono el programa, por lo que considero que no fue tan mala opción.

En general, considero que la tarea me brindo los conocimientos para poder manejar los apuntadores hasta un cierto nivel, claro que esto se puede mejorar y para esto es que la practica ayudara.