**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: Ramírez Enríquez, Karla Patricia.

Pruebas: 20 pts

Funcionalidad: 30 pts

Presentación: 5 pts

24 de mayo de 2018. Tlaquepaque, Jalisco,

**La documentación de pruebas implica:**

**1) Descripción del escenario de cada prueba**

**2) Ejecución de la prueba**

**3) Descripción y análisis de resultados.**

**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.**

**No se cumplió con los requerimientos funcionales de la tarea.**

**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 califación). 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>

**#define** N 40

**typedef** **struct**{

**char** nombre[15];

**float** calificacion;

} Profesor;

**float** **averageArray**(Profesor \*profe3 , **int** n){ //De los repetidos, hace un promedio

**int** i,j;

**float** aux;

**for** (i=0; i<n; i++){

**for** (j=n-1; j>=0; j--){

**if** (profe3[i].nombre == profe3[j].nombre)

aux= (profe3[i].calificacion+profe3[j].calificacion)/2;

**else**

aux= profe3[i].calificacion;

}

}

**return** aux;

}

**void** **readArray**(Profesor \*profe1, **int** \*pN){

**int** i;

**scanf**("%d",pN);

**for** (i=0; i< (\*pN); i++){

**scanf**("%s", profe1[i].nombre);

**scanf**("%f", &profe1[i].calificacion);

}

}

**void** **printArray**(Profesor \* profe1, **int** n){

**int** i;

**for**(i=0; i<n; i++){

**printf**("%s\n", profe1->nombre);

**printf**("%.2f\n", averageArray(profe1,n));

}

}

**void** **mergeArrays**(Profesor \*profe1, **int** \*a,Profesor \*profe2, **int** \*b, Profesor \*profe3, **int** c){

**int** i,j;

**for** (i=0; i<(\*a); i++){

**strcpy**(profe3[i].nombre, profe1[i].nombre);

profe3[i].calificacion = profe1[i].calificacion;

}

**for** (j= (\*a); j<c; j++){

**strcpy**(profe3[j].nombre, profe2[j].nombre);

profe3[j].calificacion= profe2[j].calificacion;

}

}

**void** **sortArray**(Profesor \*profe3, **int** a){

Profesor arr; //Arreglo auxiliar

**int** i;

**for** (i=0; i<(a); i++){

**if** (profe3[i].calificacion > profe3[i+1].calificacion){

arr= profe3[i];

**strcpy**(profe3[i].nombre, profe3[i+1].nombre); //Variable destino, Variable a copiar

profe3[i].calificacion = profe3[i+1].calificacion;

profe3[i+1]=arr;

}

}

}

**int** **main**(**void**){

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(arr1, &n1); //Leer el primer arreglo

readArray(arr2, &n2); //Leer el segundo arreglo

mergeArrays(arr1, &n1, arr2, &n2, arrF, N); //Fusionar los dos arreglos en un tercer arreglo

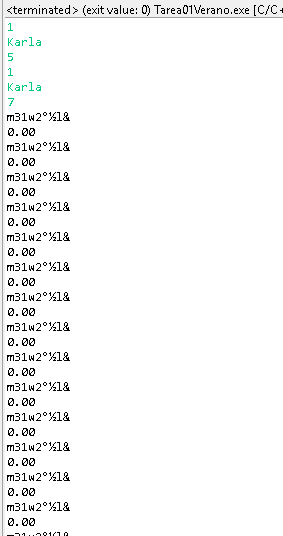
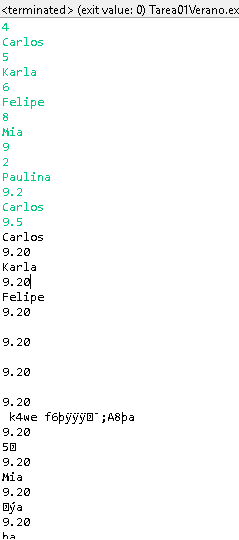
sortArray(arrF, N); //Ordenar los elementos del tercer arreglo, recuerde que pueden existir profesores repetidos

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

**return** 0;

}

Ejecución:

Conclusiones (obligatorio):

Anteriormente ya había llevado la materia y por ello mismo la tarea no se me hizo tan complicada. Tuve varios problemas en cuanto a los apuntadores cuando escaneaba la calificación y para comprender que era lo que se debía hacer en cada función.

Recordé el uso de apuntadores y aprendí a que siempre se debe leer todo el documento porque en la función de mergeArrays no comprendía por qué se pasaban 3 estructuras Profesor en lugar de dos.

Me costó trabajo y no pude solucionar que de repente se imprimían valores muy extraños y que no concordaban con lo esperado. Supongo que tengo algún error en cuanto a lo que devuelven las funciones pero no lo logro identificar hasta el momento.

Así mismo, me di cuenta que es muy importante comprender la aritmética de apuntadores ya que es completamente necesario para el uso de arreglos y ver que está apuntando a qué.