**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: Maris Mora, Michel

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.**

**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 5 pts
* 60% para la funcionalidad 60 pts
* 30% para las pruebas 20 pts

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:

<<Copie y pegue su código fuente aquí.>>

//Directivas al preprocesador

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//Variables Globales y nuevos tipos de datos

typedef struct {

char nombre[15];

float calificacion;

} Profesor;

int Acumu = 0;

//prototipos de funciones

void averageArray(Profesor \*, int, Profesor\*);

void readArray(Profesor \*, int);

void mergeArrays(Profesor \*, int, Profesor \*, int, Profesor \*, int);

void sortArray(Profesor \*, int);

void printArray(Profesor \*, int);

//Función principal

int main() {

//Configurando el buffer de salida

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, n12 = 0; //Longitud de los arreglos

Profesor \*arre1 = arr1;

Profesor \*arre2 = arr2;

Profesor \*arreF = arrF;

printf("Ingrese el número de profesores que ingresara: ");

scanf("%d", &n1);

readArray(arre1, n1); //leer el primer arreglo

printf("Ingrese el número de profesores que ingresara: ");

scanf("%d", &n2);

readArray(arre2, n2); //leer el segundo arreglo

mergeArrays(arre1, n1, arre2, n2, arreF, n12); //Fusionar los dos arreglos en un tercer arreglo

sortArray(arreF, Acumu); //Ordenar los elementos del tercer arreglo, recuerde que pueden

//existir profesores repetidos

printArray(arreF, Acumu);

//printArray(); //Imprimir el resultado final

return 0;

}

//Desarrollo de funciones

void readArray(Profesor \*Maestro, int NProfesor) {

printf("ingrese los datos de los profesores que ingresara\n");

for (int i = 0; i < NProfesor; i++) {

fflush(stdin);

scanf("%s %f", Maestro->nombre, &Maestro->calificacion);

Maestro++;

}

}

void mergeArrays(Profesor \*arre1, int n1, Profesor \*arre2, int n2,

Profesor \*arrF, int n12) {

n12 = 0;

Profesor ABC[40];

Profesor \*arrT;

arrT = ABC;

for (int i = 0; i < n1; i++) {

strcpy((arrT + n12)->nombre, (arre1 + i)->nombre);

(arrT + n12)->calificacion = (arre1 + i)->calificacion;

n12++;

}

for (int i = 0; i < n2; i++) {

strcpy((arrT + n12)->nombre, (arre2 + i)->nombre);

(arrT + n12)->calificacion = (arre2 + i)->calificacion;

n12++;

}

averageArray(arrF, n12, arrT);

}

void averageArray(Profesor \*arrF, int n12, Profesor \*arrT) {

float acum = 0;

int cont = 0;

char G[15] = "...............";

Profesor \*ARRF, ABC[40];

ARRF = ABC;

for (int i = 0; i < n12; i++) {

strcpy((ARRF + i)->nombre, (arrT + i)->nombre);

(ARRF + i)->calificacion = (arrT + i)->calificacion;

}

for (int i = 0; i < n12; i++) {

for (int j = 0; j < n12; j++) {

if (i != j) {

if (strcmp((arrT + i)->nombre, (arrT + j)->nombre) == 0) {

for (int z = 0; z < 15; z++) {

(arrT + j)->nombre[z] = '.';

}

(arrT + j)->calificacion = 20;

}

}

}

}

for (int i = 0; i < n12; i++) {

if ((arrT + i)->calificacion != 20) {

strcpy((arrF + Acumu)->nombre, (ARRF + i)->nombre);

Acumu++;

}

}

for (int l = 0; l < n12; l++) {

if (strcmp((arrT + l)->nombre, G) != 0) {

acum = 0;

acum += (ARRF + l)->calificacion;

cont = 1;

for (int j = 0; j < n12; j++) {

if (l != j) {

if (strcmp((ARRF + l)->nombre, (ARRF + j)->nombre) == 0) {

acum += (ARRF + j)->calificacion;

cont++;

}

}

}

for (int i = 0; i < n12; i++) {

if (strcmp((arrT + l)->nombre, (arrF + i)->nombre) == 0) {

if (cont != 1) {

(arrF + i)->calificacion = acum / cont;

} else {

(arrF + i)->calificacion = acum;

}

}

}

}

}

}

void sortArray(Profesor \*arrF, int n12) {

int i, j;

Profesor t;

for (i = 0; i < n12; i++) {

for (j = 0; j < n12 - 1; j++) {

if ((arrF + j)->calificacion < (arrF + j + 1)->calificacion) {

t.calificacion = (arrF + j)->calificacion;

strcpy(t.nombre, (arrF + j)->nombre);

(arrF + j)->calificacion = (arrF + j + 1)->calificacion;

strcpy((arrF + j)->nombre, (arrF + j + 1)->nombre);

(arrF + j + 1)->calificacion = t.calificacion;

strcpy((arrF + j + 1)->nombre, t.nombre);

}

}

}

}

void printArray(Profesor \*arrF, int n12) {

for (int i = 0; i < n12; i++) {

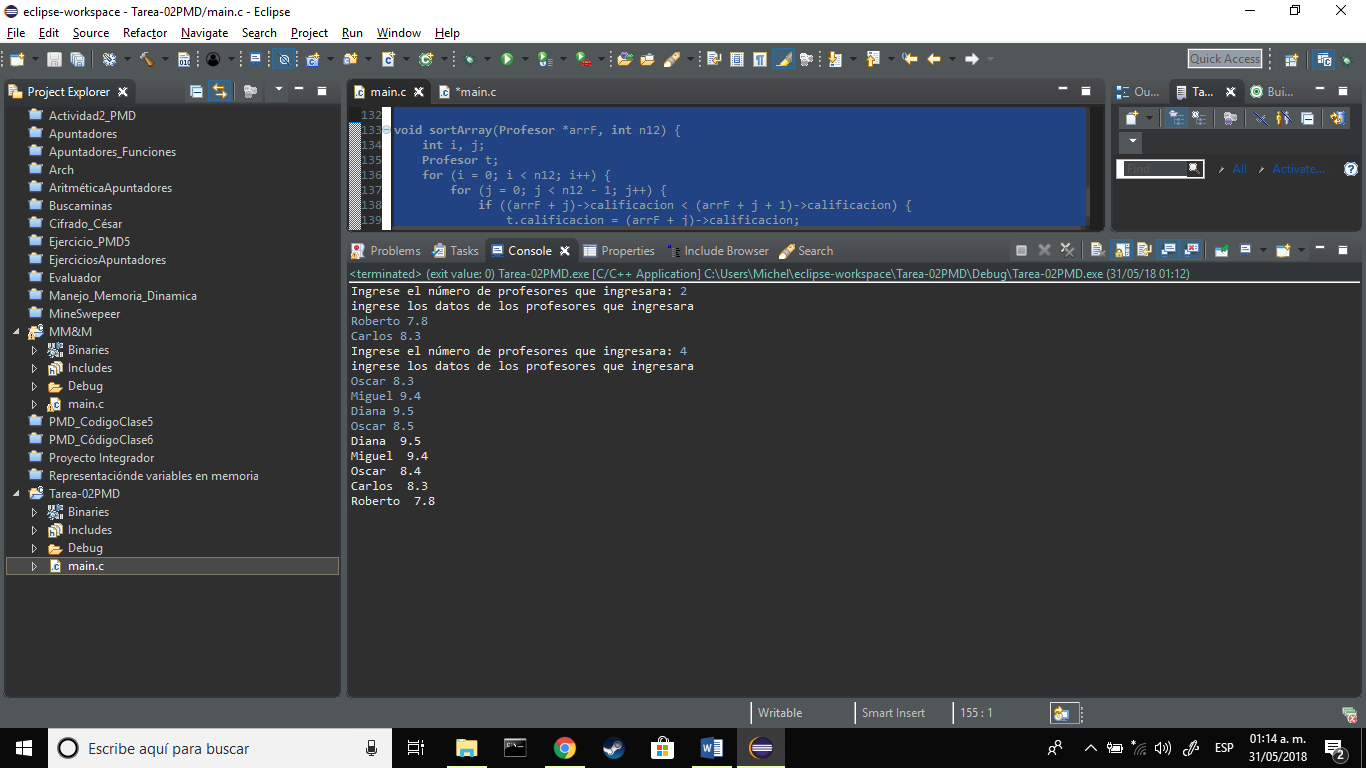
printf("%s %.1f\n", (arrF + i)->nombre, (arrF + i)->calificacion);

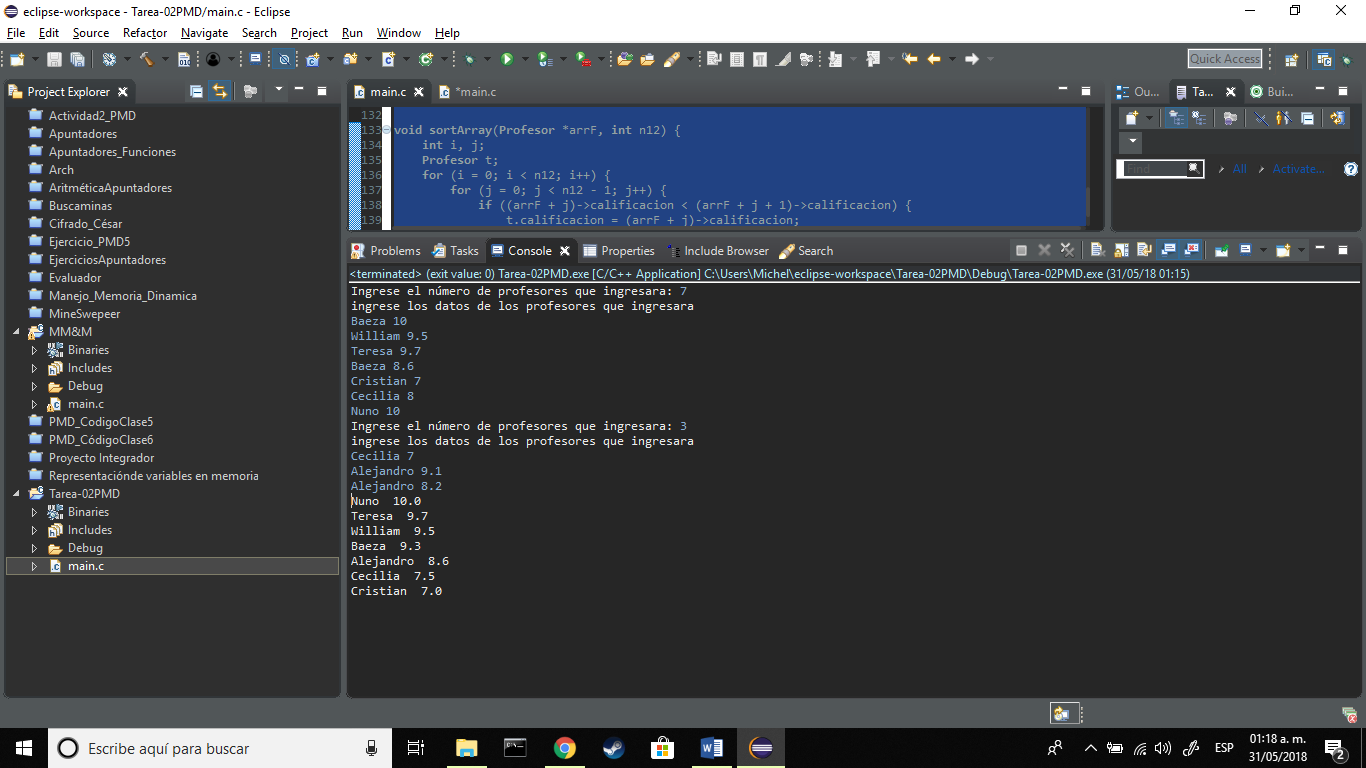
}

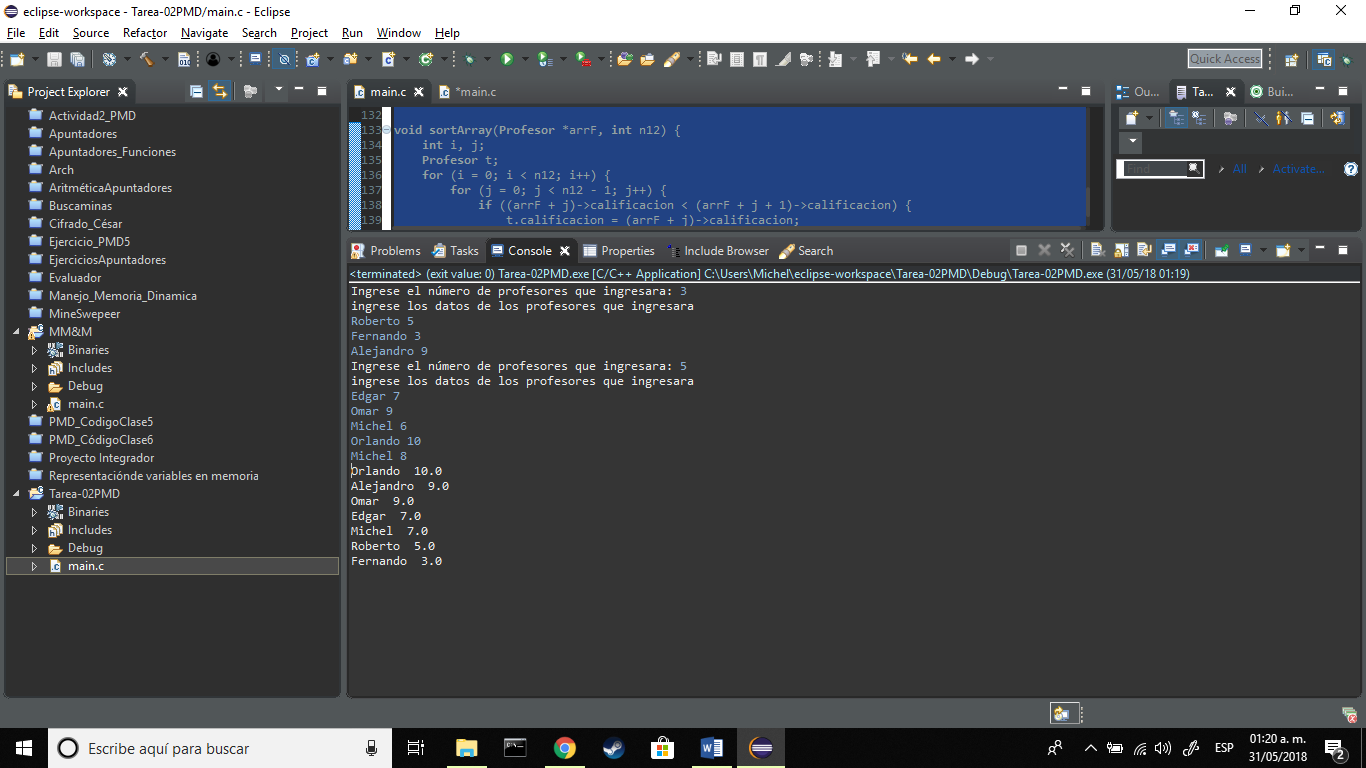
}

Ejecución:

<<Inserte capturas de pantalla de una ejecución donde la primera lista tenga la suma mayor y de otra donde la segunda lista tenga la suma mayor.>>







Conclusiones (obligatorio):

* Lo que aprendí con esta práctica. Lo que ya sabía.
  + ¿Qué aprendí en esta práctica?: Fueron distintas cosas las que aprendí, pero más que nada reforcé después de su uso en clase, una de las cosas que aprendí fue a eliminar datos repetidos en un arreglo, y lo que reforcé fue el uso de sintaxis de apuntadores con estructuras, lo cual, a pesar de tener distintas formas de ser utilizado, ya puedo identificar de una forma fácil cuándo y cómo se esta utilizando.
* Lo que me costó trabajo y cómo lo solucioné.
  + Lo que me costó más trabajo se encontraba en la función de promediar datos y eliminar los repetidos de estos, esto debido a que se dificultaba la comparación de caracteres y no variables, lo que hice para solucionarlo fue a base de prueba y error y llamando la función para imprimir datos e imprimiendo variables utilizadas para conocer sus valores canónicos durante la ejecución, también fue de ayuda el trabajar en compañía para así conocer errores de codificación y el volver a visitar códigos antiguos para reanimar antiguos códigos que pudiesen ser de ayuda.
* Lo que no pude solucionar.
  + Al final pude solucionar todos los problemas de ejecución, sintaxis y lógica