

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 APUNTADES

Autor: Aguilar Aguirre Alfonso

24 de mayo de 2018. Tlaquepaque, Jalisco,

Instrucciones para entrega de tarea

Es **IMPRESINDIBLE** 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 40 pts
- 30% para las pruebas 10 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 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 primera 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 <string.h>
typedef struct{
    char nombre[15];
    float calificacion;
} Profesor;

void readArray(Profesor *P,int n);
void printArray(Profesor *P,int n);
void mergeArrays(Profesor P1[20],int n1,Profesor *P2,int
n2,Profesor P3[40],int n3);

int main(int argc, const char * argv[]) {
    Profesor arr1[20];
    Profesor arr2[20];
    Profesor arrF[40];
    Profesor *d1=arr1,*d2=arr2,*d3=arrF;
    int n1,n2;
    scanf("%d",&n1);
    readArray(d1,n1);
    scanf("%d",&n2);
    readArray(d2,n2);
    mergeArrays(arr1,n1,d2,n2,arrF,n1+n2);
    printArray(d3,n1+n2);

    return 0;
}
void readArray(Profesor *P,int n){
```

```

    int i;
    char temp;
    float t;
    for(i=0;i<n;i++){
        scanf("%c",&temp);
        scanf("%[^\\n]",(P+i*5)->nombre);
        scanf("%f",&t);
        (P+i*5)->calificacion=t;
    }
}

void printArray(Profesor *P,int n){
    int i;
    for(i=0;i<n;i++)
        printf("%s  %f\\n", (P+i*5)->nombre, (P+i*5)->calificacion);
}

void mergeArrays(Profesor P1[20],int n1,Profesor *P2,int
n2,Profesor P3[40],int n3){
    int i,j,cont,k;
    for(i=0;i<n1;i++){
        cont=i+1;
        for(j=cont;j<n1;j++)
            //c1=&P1.nombre;
            //c1=(P1+i*5)->nombre;
            //c2=(P1+j*5)->nombre;
            if(strcmp(P1[i].nombre,P1[j].nombre)==0){
                //(P3+i*5)->nombre=c1;
                printf("equal\\n");
                strcpy(P1[i].nombre,P3[i].nombre);
                break;
            }
            else{
                printf("different\\n");
                strcpy(P1[i].nombre,P3[i].nombre);
            }
        for(k=0;k<n2;k++){
            if(strcmp(P1[i].nombre,P2[k].nombre)==1){
                strcpy(P1[i].nombre,P3[i].nombre);
                break;
            }
            else{
                printf("different\\n");
                strcpy(P1[i].nombre,P3[i].nombre);
            }
        }
    }
}
}


```



Ejecución:

```
2
lo
2
lo
2
2
lo
2
lo
2
diferent
diferent
diferent
diferent
diferent
0.000000
0.000000
0.000000
0.000000
Program ended with exit code: 0
```

Conclusiones (obligatorio):

- 
- ✓ Lo que aprendí con esta práctica. Lo que ya sabía.
Lo que aprendí fue el uso de apuntadores a estructuras con lo cual podemos facilitar algunas tareas
 - ✓ Lo que me costó trabajo y cómo lo solucioné.
Algo que me costó trabajo fue al inicio del problema no podía recuperar el primer dato ingresado ya que la dirección de memoria inicial aparentemente no era la misma que necesitaba pero cambiando a que la función recibiera un apuntador de la estructura en lugar de la estructura soluciono el problema
 - ✓ Lo que no pude solucionar.
No pude hacer la comparación lógica para no repetir los datos en el merge, no pude lograr que dentro del if regresara como verdadera la igualdad, siempre fue desigualdad supongo que porque nunca tomé el valor como cadena o realmente lo desconozco, eso me frenó a seguir avanzando con las demás funciones.