

# 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: Manzano López de Ortigosa Luis Enrique

31 de mayo de 2018. Tlaquepaque, Jalisco,

Presentación: 10 pts

Funcionalidad: 55 pts

Pruebas: 25 pts

Todas las figuras e imágenes deben tener un título y utilizar 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;

//float averageArray(Profesor *lista, int n);
void readArray(Profesor *lista, int n);
void mergeArrays (Profesor *lista1, int n1, Profesor *lista2, int n2,
Profesor *listaf, int n3);
void sortArray(Profesor *lista, int n);
void printArray(Profesor *lista, int n );

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, n3; //Longitud de los arreglos

    scanf("%d", &n1);
    readArray(arr1, n1); //leer el primer arreglo
    scanf("%d", &n2);
    readArray(arr2, n2); //leer el segundo arreglo
    n3=n1+n2;
    mergeArrays(arr1, n1, arr2, n2, arrF, n3); //Fusionar los dos
arreglos en un tercer arreglo

    sortArray(arrF, n3); //Ordenar los elementos del tercer arreglo,
recuerde que pueden
    //existir profesores repetidos

    printArray(arrF, n3); //Imprimir el resultado final
```

```

}

void readArray (Profesor *lista, int n){
    short i;

    for (i=0; i<n ; i++){
        scanf("%s",lista[i].nombre);
        scanf("%f", &lista[i].calificacion);
    }
}

void mergeArrays (Profesor *lista1, int n1, Profesor *lista2, int n2,
Profesor *listaf, int n3){
    short i, j;

    for (i=0; i<n1; i++){
        strcpy(listaf[i].nombre, lista1[i].nombre);
        listaf[i].calificacion=lista1[i].calificacion;
    }
    j=i;
    for (i=0; i<n2; i++, j++){
        strcpy(listaf[j].nombre, lista2[i].nombre);
        listaf[j].calificacion=lista2[i].calificacion;
    }
}

/*void avarageArray (Profesor *lista, int *n){
    short i, j, ntemp;
    char temp[*n], repetir=0, repetir2=0;
    float ctemp;

    ntemp=*n;
    for (i=0; i<*n ; i++){
        for (j=i; j<*n; j++){
            if (strcmp(lista[i].nombre, lista[j].nombre)){
                temp[j]=1;
                repetir++;
            }
        }

        if (repetir){
            repetir2=repetir;
            ctemp=lista[i].calificacion;
            for (j=i; repetir>0; j++){
                if (temp[j]==1){
                    ctemp+=lista[j].calificacion;
                    lista[j]=lista[j+1];
                    *n =*n-1;
                }
            }
        }
    }
}

```

```

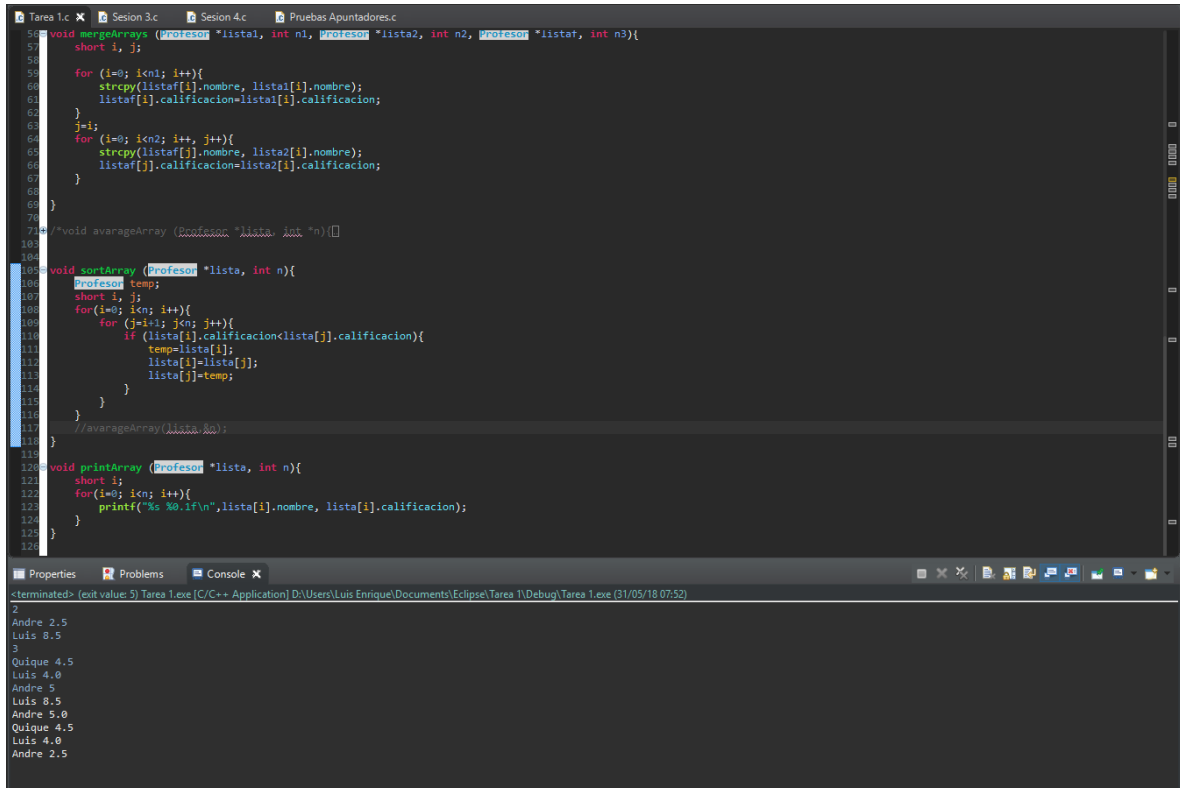
        }
        lista[i].calificacion=ctemp/repetir2;
        for (j=0; j<ntemp; j++){
            temp[j]=0;
        }
    }
}
*/

void sortArray (Profesor *lista, int n){
    Profesor temp;
    short i, j;
    for(i=0; i<n; i++){
        for (j=i+1; j<n; j++){
            if (lista[i].calificacion<lista[j].calificacion){
                temp=lista[i];
                lista[i]=lista[j];
                lista[j]=temp;
            }
        }
    }
    //avarageArray(lista,&n);
}

void printArray (Profesor *lista, int n){
    short i;
    for(i=0; i<n; i++){
        printf("%s %0.1f\n",lista[i].nombre, lista[i].calificacion);
    }
}

```

## Ejecución:

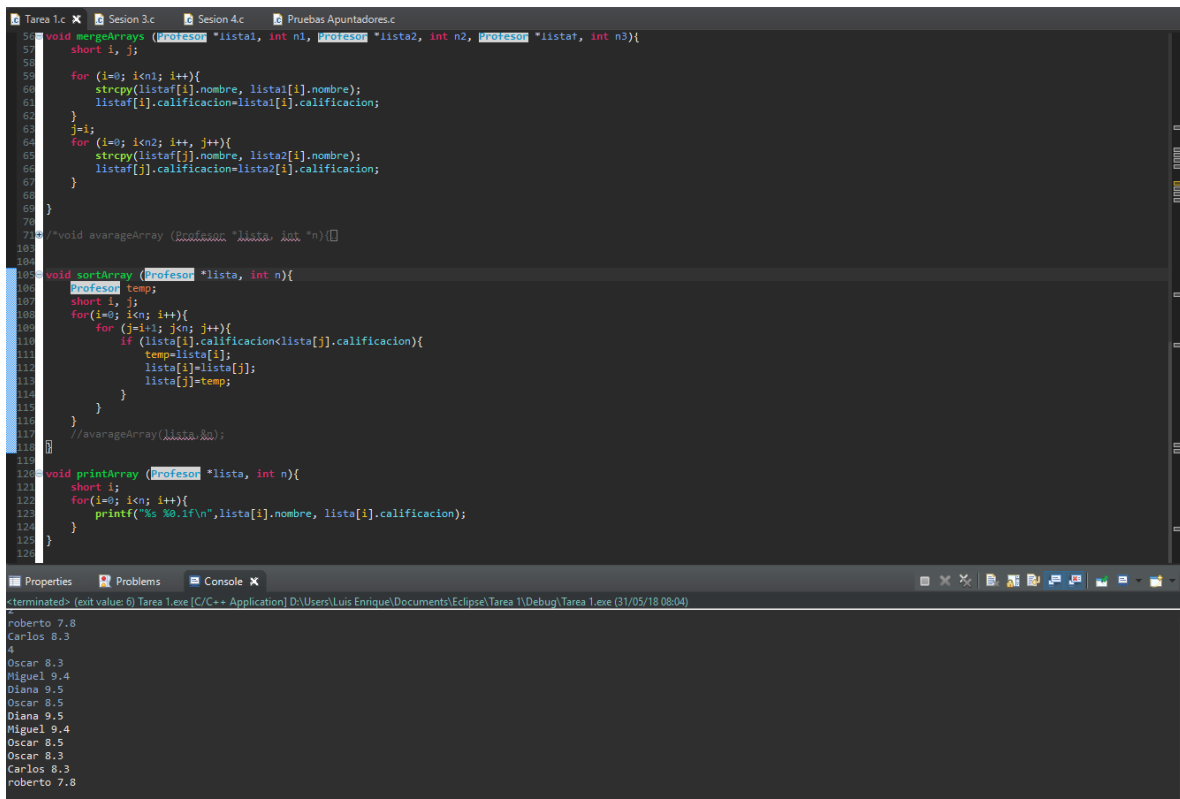


```
56 void mergeArrays (Profesor *lista1, int n1, Profesor *lista2, int n2, Profesor *lista3, int n3){
57     short i, j;
58
59     for (i=0; i<n1; i++){
60         strcpy(lista3[i].nombre, lista1[i].nombre);
61         lista3[i].calificacion=lista1[i].calificacion;
62     }
63     j=i;
64     for (i=0; i<n2; i++, j++){
65         strcpy(lista3[j].nombre, lista2[i].nombre);
66         lista3[j].calificacion=lista2[i].calificacion;
67     }
68 }
69
70
71 //void averageArray (Profesor *lista, int n){
72
73
74
75 void sortArray (Profesor *lista, int n){
76     Profesor temp;
77     short i, j;
78     for(i=0; i<n; i++){
79         for (j=i+1; j<n; j++){
80             if (lista[i].calificacion<lista[j].calificacion){
81                 temp=lista[i];
82                 lista[i]=lista[j];
83                 lista[j]=temp;
84             }
85         }
86     }
87     //averageArray(lista, &n);
88 }
89
90 void printArray (Profesor *lista, int n){
91     short i;
92     for(i=0; i<n; i++){
93         printf("%s %0.1f\n", lista[i].nombre, lista[i].calificacion);
94     }
95 }
```

Properties Problems Console

<terminated> (exit value: 5) Tarea 1.exe [C:/C++ Application] D:\Users\Luis Enrique\Documents\Eclipse\Tarea 1\Debug\Tarea 1.exe (31/05/18 07:52)

```
2
Andre 2.5
Luis 8.5
3
Quique 4.5
Luis 4.0
Andre 5
Luis 8.5
Andre 5.0
Quique 4.5
Luis 4.0
Andre 2.5
```



```
56 void mergeArrays (Profesor *lista1, int n1, Profesor *lista2, int n2, Profesor *lista3, int n3){
57     short i, j;
58
59     for (i=0; i<n1; i++){
60         strcpy(lista3[i].nombre, lista1[i].nombre);
61         lista3[i].calificacion=lista1[i].calificacion;
62     }
63     j=i;
64     for (i=0; i<n2; i++, j++){
65         strcpy(lista3[j].nombre, lista2[i].nombre);
66         lista3[j].calificacion=lista2[i].calificacion;
67     }
68 }
69
70
71 //void averageArray (Profesor *lista, int n){
72
73
74
75 void sortArray (Profesor *lista, int n){
76     Profesor temp;
77     short i, j;
78     for(i=0; i<n; i++){
79         for (j=i+1; j<n; j++){
80             if (lista[i].calificacion<lista[j].calificacion){
81                 temp=lista[i];
82                 lista[i]=lista[j];
83                 lista[j]=temp;
84             }
85         }
86     }
87     //averageArray(lista, &n);
88 }
89
90 void printArray (Profesor *lista, int n){
91     short i;
92     for(i=0; i<n; i++){
93         printf("%s %0.1f\n", lista[i].nombre, lista[i].calificacion);
94     }
95 }
```

Properties Problems Console

<terminated> (exit value: 6) Tarea 1.exe [C:/C++ Application] D:\Users\Luis Enrique\Documents\Eclipse\Tarea 1\Debug\Tarea 1.exe (31/05/18 08:04)

```
roberto 7.8
Carlos 8.3
4
Oscar 8.3
Miguel 9.4
Diana 9.5
Oscar 8.5
Diana 9.5
Miguel 9.4
Oscar 8.5
Oscar 8.3
Carlos 8.3
roberto 7.8
```

```
Tarea 1.c x Sesión 3.c Sesión 4.c Pruebas Apuntadores.c
59 for (i=0; i<n1; i++){
60     strcpy(listaf[i].nombre, lista1[i].nombre);
61     listaf[i].calificacion=listaf[i].calificacion;
62 }
63 j=i;
64 for (i=0; i<n2; i++, j++){
65     strcpy(listaf[j].nombre, lista2[i].nombre);
66     listaf[j].calificacion=listaf[i].calificacion;
67 }
68 }
69 }
70
71 void averageArray (Profesor *lista, int n){
72
73 }
74
75 void sortArray (Profesor *lista, int n){
76     Profesor temp;
77     short i, j;
78     for(i=0; i<n; i++){
79         for (j=i+1; j<n; j++){
80             if (lista[i].calificacion<lista[j].calificacion){
81                 temp=lista[i];
82                 lista[i]=lista[j];
83                 lista[j]=temp;
84             }
85         }
86     }
87 }
88 //averageArray(lista, n);
89 }
90
91 void printArray (Profesor *lista, int n){
92     short i;
93     for(i=0; i<n; i++){
94         printf("%s %s %.1f\n", lista[i].nombre, lista[i].calificacion);
95     }
96 }
97 }
```

Properties Problems Console x

<terminated> (exit value: 7) Tarea 1.exe [C/C++ Application] D:\Users\Luis Enrique\Documents\Eclipse\Tarea 1\Debug\Tarea 1.exe (31/05/18 08:06)

3

Luis 8.5  
enrique 9.5  
Manzano 10  
4  
Frank 3.5  
Lopez 2.5  
Obrador 4.0  
Meade 7.0  
Manzano 10.0  
Enrique 9.5  
Luis 8.5  
Meade 7.0  
Obrador 4.0  
Frank 3.5  
Lopez 2.5

```
Tarea 1.c x Sesión 3.c Sesión 4.c Pruebas Apuntadores.c
59 for (i=0; i<n1; i++){
60     strcpy(listaf[i].nombre, lista1[i].nombre);
61     listaf[i].calificacion=listaf[i].calificacion;
62 }
63 j=i;
64 for (i=0; i<n2; i++, j++){
65     strcpy(listaf[j].nombre, lista2[i].nombre);
66     listaf[j].calificacion=listaf[i].calificacion;
67 }
68 }
69 }
70
71 void averageArray (Profesor *lista, int n){
72
73 }
74
75 void sortArray (Profesor *lista, int n){
76     Profesor temp;
77     short i, j;
78     for(i=0; i<n; i++){
79         for (j=i+1; j<n; j++){
80             if (lista[i].calificacion<lista[j].calificacion){
81                 temp=lista[i];
82                 lista[i]=lista[j];
83                 lista[j]=temp;
84             }
85         }
86     }
87 }
88 //averageArray(lista, n);
89 }
90
91 void printArray (Profesor *lista, int n){
92     short i;
93     for(i=0; i<n; i++){
94         printf("%s %s %.1f\n", lista[i].nombre, lista[i].calificacion);
95     }
96 }
97 }
```

Properties Problems Console x

<terminated> (exit value: 5) Tarea 1.exe [C/C++ Application] D:\Users\Luis Enrique\Documents\Eclipse\Tarea 1\Debug\Tarea 1.exe (31/05/18 08:06)

2

Venustiano 5.5  
Carranza 3.8  
3  
Adolfo 8.5  
Mateos 7.0  
Anaya 9  
Anaya 9.0  
Adolfo 8.5  
Mateos 7.0  
Venustiano 5.5  
Carranza 3.8



## Conclusiones:

Con este trabajo pude emplear diversos conocimientos que se aprendieron a través del curso, como:

- ✓ Accesar al valor de un apuntador.
- ✓ Poder hacer modificaciones a los valores, directamente de un puntero.
- ✓ Usar diversos métodos de estructuras, funciones iterativas, entre otros.

Además, mis competencias fueron estiradas al límite al momento de que las siguientes situaciones se presentaron:

- ✓ El ordenamiento de los profesores conforme a su calificación. Y esto pudo ser resuelto gracias a el empleo de variables temporales para así almacenar su contenido, pasarlo e ir acomodando con respecto a su calificación.

Por último, cabe destacar que desafortunadamente en este ejercicio no se pudieron completar las siguientes tareas:

- ✓ Debido a la complejidad de la actividad en sí, traté el programar la función *avarageArray* pero no tuve la capacidad de poder eliminar en el arreglo aquellos profesores que se repetían. Por ende, mi código ordena de forma descendiente, pero no remueve los que se repiten.