

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN ELECTRÓNICA



PROGRAMACIÓN CON MEMORIA DINÁMICA TAREA 1. MANEJO DE APUNTADES

Autor: FLORES RUIZ, DAVID

Presentación: 5 pts

Funcionamiento: 60 pts

Pruebas: 15 pts

24 de mayo de 2018. Tlaquepaque, Jalisco,

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.

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
- 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 una 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:

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

```
/*
=====
Name       : -.c
Author      :
Version     :
Copyright   : Your copyright notice
Description : Hello World in C, Ansi-style
=====
*/

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

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

//float averageArray(Profesor _____ , int _____);
void readArray(Profesor arr[20], int n);
void mergeArrays(Profesor arr1[] , int n1, Profesor arr2[] , int n2, Profesor
arrF[]);
void sortArray(Profesor arrF[], int n3);
int comparaCadenas (char cadena1[], char cadena2[]);
void printArray(Profesor arrF[], int n3);

int main(void) {
    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, n3; //Longitud de los arreglos
    int valor;
    Profesor *apuntador;
```

```

printf("\nDe que tamaño es la *lista #1* de profesores a ingresar: ");
scanf("%d", &n1);
readArray(arr1, n1); //leer el primer arreglo

setbuf(stdin, 0);
printf("\nDe que tamaño es la *lista #2* de profesores a ingresar: ");
scanf("%d", &n2);
readArray(arr2, n2); //leer el segundo arreglo

n3 = n1 + n2;

mergeArrays(arr1, n1, arr2, n2, arrF); //Fusionar los dos arreglos en un
tercer arreglo
printf("El arreglo fusionado es: ");
printArray(arrF, n3);

apuntador = arrF;
int i=0;
int j=0;

for(i=0; i<n3; i++){
    for(j=i; j<n3; j++){
        if((i != j) && ((apuntador + i)->calificacion != -5) &&
((apuntador + j)->calificacion != -5))// Evitar que compare contra si mismo y
Revisar si no ha habido BORRADO Lógico
        {
            valor = comparaCadenas( (apuntador + i)->nombre ,
(apuntador + j)->nombre );
            printf("Valor Comparacion: %d ", valor);
            if(valor == 0) //Si en la comparacion son iguales
            {
                (apuntador + i)->calificacion = (
(apuntador + i)->calificacion + (apuntador + j)->calificacion) /2;
                //Le asignamos el promedio al nombre repetido en la posición que primero
comparo contra todas la demás borrando logicamente esas otras
                (apuntador + j)->calificacion = -5; //Borrado
lógico
            }
            printf("-----");
            printf("%d %d", i, j);
        }
    }
}

sortArray(arrF, n3); //Ordenar los elementos del tercer arreglo,
recuerde que pueden existir profesores repetidos
printf("El arreglo ORDENADO es: ");
printArray(arrF, n3);

printf("El arreglo ---FINAL--- es: ");
printArray(arrF, n3); //Imprimir el resultado final

return EXIT_SUCCESS;
}

```

```

//float averageArray(Profesor _____ , int _____){
//
//
//    return 3.5;
//}

void readArray(Profesor arr[20], int N){
    Profesor *apuntador = arr;
    char auxNombre[15];
    float auxCalif;

    printf("Introduce los profesores y su calificacion:");

    int i = 0;
    for (i=0; i < N; i++){
        setbuf(stdin, 0);
        gets(auxNombre);
        strcpy((apuntador + i)->nombre , "");
        strcpy((apuntador + i)->nombre , auxNombre);
        scanf("%f", &auxCalif);
        (apuntador + i)->calificacion = auxCalif;
    }
}

void mergeArrays(Profesor arr1[] , int n1, Profesor arr2[] , int n2, Profesor
arrF[]){    //n3 = n1 + n2
    Profesor *apuntador = arrF;
    int i, j, limit;

    for(i=0; i < n1; i++){
        strcpy((apuntador + i)->nombre , "");
        strcpy((apuntador + i)->nombre , (arr1 + i)->nombre);
        (apuntador + i)->calificacion = (arr1 + i)->calificacion;
    }

    limit = i;

    for(j=0; j < n2; j++){
        strcpy((apuntador + j + limit)->nombre , "");
        strcpy((apuntador + j + limit)->nombre , (arr2 + j)->nombre);
        (apuntador + j + limit)->calificacion = (arr2 + j)->calificacion;
    }
}

void sortArray(Profesor arrF[], int n3){    //Aqui deben llegar el arreglo
    Profesor *apunta = arrF;                //combinado de las 2 listas y su
tamaño
    Profesor structSwap;    //CREO Q PODRIA SER SOLO 1 VARIABLE
    Profesor *auxSwap = &structSwap;
    int i;
    int j;

    for(i = 0; i < n3; i++){

```

```

        for(j = 0; j < n3-1; j++){
            if( (apunta + j)->calificacion < (apunta + j + 1)-
>calificacion )
            {
                strcpy( (auxSwap)->nombre, "");
                strcpy( (auxSwap)->nombre, (apunta + j)-
>nombre ); //Pasar al AUX
                (auxSwap)->calificacion = (apunta + j)-
>calificacion; //Pasar al AUX
                strcpy( (apunta + j)->nombre, (apunta + j +
1)->nombre );// Inicia el SWAP
                (apunta + j)->calificacion = (apunta + j + 1)-
>calificacion;//Inicia el SWAP
                strcpy( (apunta + j + 1)->nombre,
(auxSwap)->nombre );//FIN de SWAP
                (apunta + j + 1)->calificacion = (auxSwap)-
>calificacion;//FIN de SWAP
            }
        }
    }
}

```

```

int comparaCadenas (char cadena1[], char cadena2[]){ //Deben ser de MAX. 15
caracteres
    char *apCad1 = cadena1;
    char *apCad2 = cadena2;
    int i=0;
    int bandera1 = 1, bandera2 = 1; // Se inicializan banderas en 1
    int mayor = 1;
    int iguales = 0;
    int menor = -1;

    while (i<15)
    {
        if( *(apCad1 + i) == '\0' )
            bandera1 = 0; //Se apaga bandera1

        if( *(apCad2 + i) == '\0' )
            bandera2 = 0; //Se apaga bandera2

        if ( *(apCad1 + i) > *(apCad2 + i) ) // s1>s2
            return mayor;

        if ( *(apCad1 + i) < *(apCad2 + i) ) // s1<s2
            return menor;

        if( bandera1 == 0 && bandera2 == 1)
            return menor;

        if( bandera1 == 1 && bandera2 == 0)
            return mayor;
    }
}

```



```

        if( bandera1 == 0 && bandera2 == 0)
            return iguales;

        i++;
    }
    return 8;    // SI regresa 8 es error
}

void printArray(Profesor arrF[], int n3){
    Profesor *dezplazaArr = arrF;

    printf("\n\nLa lista ordenada de profesores, sin repetir ninguno es:\n");

    int i = 0;
    for (i=0; i<n3; i++){
        if( (dezplazaArr + i)->calificacion != -5)//Si hubo borrado
        lógico no lo imprime
        {
            printf("%s ->\t\t", (dezplazaArr + i)->nombre );
            printf("%f\n", (dezplazaArr + 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.>>

```
De que tamaño es la *lista #1* de profesores a ingresar: 2
Introduce los profesores y su calificación:Roberto
7.8
Carlos
8.3

De que tamaño es la *lista #2* de profesores a ingresar: 4
Introduce los profesores y su calificación:Oscar
8.3
Miguel
9.4
Diana
9.5
Oscar
8.5
El arreglo ---FINAL--- es:

La lista ordenada de profesores, sin repetir ninguno es:
Diana ->          9.500000
Miguel ->         9.400000
Oscar ->          8.400000
Carlos ->         8.300000
Roberto ->        7.800000
|
```

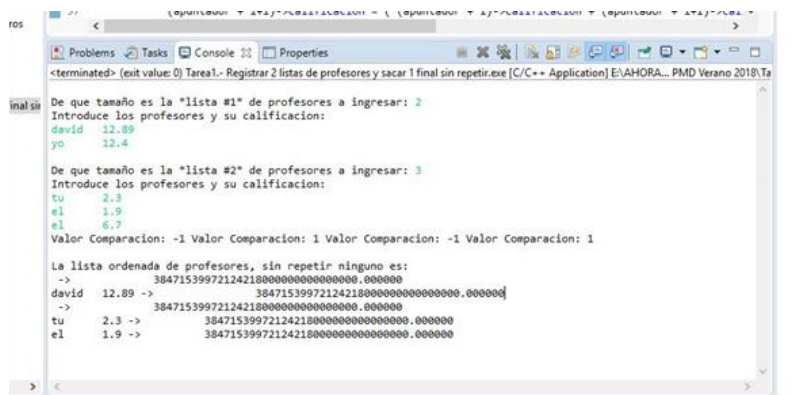
Conclusiones (obligatorio):

- ✓ Lo que aprendí con esta práctica. Lo que ya sabía.

Aprendí a declarar apuntadores teniendo en cuenta al tipo de dato que van a apuntar en este caso un arreglo de estructuras y a manejarlo dentro de ciclos para imprimir y guardar los datos dentro de mi arreglo de estructuras.

A hacer la unión en 1 arreglo más grande de otros 2 de tamaño posiblemente diferentes o iguales usando 2 ciclos for; y también logré hacer la función strcmp sin usar la de la librería string; fue implementada en un ciclo while y muchas condiciones así como los posibles returns para saber la comparación de las 2 cadenas.

✓ Lo que me costó trabajo y como lo solucioné.



```
<terminated> (exit value: 0) Tarea1.- Registrar 2 listas de profesores y sacar 1 final sin repetir.exe [C:/C++ Application] E:\AHORA... PMD Verano 2018.Ta

De que tamaño es la "lista #1" de profesores a ingresar: 2
Introduce los profesores y su calificación:
david 12.09
yo 12.4

De que tamaño es la "lista #2" de profesores a ingresar: 3
Introduce los profesores y su calificación:
tu 2.3
el 1.9
el 6.7
Valor Comparacion: -1 Valor Comparacion: 1 Valor Comparacion: -1 Valor Comparacion: 1

La lista ordenada de profesores, sin repetir ninguno es:
-> 38471539972124218000000000000000.000000
david 12.09 -> 38471539972124218000000000000000.000000
-> 38471539972124218000000000000000.000000
tu 2.3 -> 38471539972124218000000000000000.000000
el 1.9 -> 38471539972124218000000000000000.000000
```

Llegar a esta primera ejecución en consola fue el primer paso pensé que era el problema con el buffer; y fui rastreando paso por paso función por función hasta que encontré que el problema era que ni siquiera se guardaban los datos en el arreglo de estructuras; ahí empecé a poner líneas del `setvbuf` para limpiarlo antes de cada `gets`(para el nombre) y `scanf`(calificación) hasta que por fin con dar enter entre un nombre y calificación se empezaron a guardar los datos.

Luego salio todo bien con el `bubblesort` y mi último problema es que mi programa a la hora de detectar los repetidos se ciclaba en un `while` que tenía como condición dentro de 2 `for` comparar contra todos los nombres excepto el mismo, y los que tuvieran un borrado lógico tampoco.

Hasta que se corrigió por que aunque la condición estaba bien debía ser en un `if` y no en un `While`, fue cuando por fin vi la solución por primera vez solamente faltaba volver a ordenar o cambiar los procesos es decir primero sacar los repetidos y luego ordenar para imprimir todos los que no tengan borrado lógico (-5).

Aclarar solo que para la función de unir los arreglos un use el parámetro que era para el tamaño del arreglo resultante porque $n3 = n1 + n2$.

Y también omití la función de average porque para mí estaba de más; es decir cuando comparaba un elemento contra todos los demás de la lista si eran iguales al elemento que iteraba mas lento le asignaba el promedio de sumarlo a el con el otro y entre 2 (osea "promedio") y al otro le asignaba como calif un -5 que era un borrado lógico y esos al final no los imprimia también esto último fue algo nuevo que aprendí.

✓ Lo que no pude solucionar.

Esta vez con suerte y gracias a las asesorías y además con ayuda de los mtros. Pude solucionar todo el ejercicio.

Todavía me gustaría que pudiera leer el arreglo con mas presentación es decir, usando un tabulador para nombre y calificación y luego enter cosa que de plano no logré al contrario tuve que poner varios setvbuf y usar enter entre un dato y otro desde el principio.