Guía adicional: ejercicios integradores

Teoría

1. Conteste brevemente las siguientes preguntas:
    a. ¿Cómo se puede evitar que un parámetro pasado por referencia sea modificado?
    b. ¿Qué diferencia existe entre una unión y una estructura?
    c. ¿Qué relación existe entre las declaraciones **int vector[]** e **int *vector**?
    d. ¿Cuál es la utilidad de un puntero a función?
    e. ¿De qué manera está organizada la memoria en un sistema operativo? Detalle qué función cumple cada división y realice un diagrama.

2. Identifique si las siguientes afirmaciones son verdaderas o falsas y justifique en el caso de ser falsa.
    a. Las uniones se inicializan de la misma manera que las estructuras.
    b. Las estructuras se pasan siempre a las funciones por referencia.
    c. Los punteros de diferentes tipos no pueden ser asignados uno al otro.
    d. Las variables almacenadas en el stack son accesibles desde cualquier función.

3. Se realizan las siguiente declaraciones:
    int mat[3][4],
     *ptr = &mat,
    aux = 0;
    La matriz se inicializa de tal forma que cada uno de sus elementos sea igual al índice de la fila multiplicado por 10 más el índice de la columna. Qué valor adopta la variable aux al ejecutarse la sentencia:
    aux = *(ptr + 7);

4. Indique el valor que tendrá la variable dato antes de finalizar el programa:

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

int main()
{
   int dato = 5;
   int **pDoble;
   int mat[3][4] = { {9,  8,  7,  6}, {16,  15,  14,  13}, {24,  23, 22, 21} };
   pDoble = (int **)malloc(3*sizeof(int *));
   *(pDoble++) = (int *)&mat[0];
   *(pDoble++) = (int *)&mat[1];
   *(pDoble) = (int *)&mat[2];
```

```
   dato = *(*(pDoble - 1) + 2);
   return 0;
}
```

5. Indicar cuáles de las siguientes asignaciones son incorrectas y cómo se corregirían:

   a)
   ```
   int  *ptri,  a = 10;
    ptri = &a;
   double *ptrf,  x = 5.0;
   ptrf = &x;
   ptrf = ptri;
   ```

   b)
   ```
   char  *ptrc;
   *ptrc = 'a';
   ```

   c)
   ```
   int  n;
   int  *ptrn = &n;
   ptrn = 9;
   ```

   d)
   ```
   int  *ptrn = NULL;
   *ptrn = 9;
   ```

6. Se tiene el siguiente programa escrito en lenguaje C. Indique los valores que adoptan TODOS los campos de la variable de tipo *linea*.

```
#include <stdio.h>
#include <stdlib.h>
struct punto
{
   int coorx;
   int coory;
};
typedef struct
{
   int color;
   struct punto origen;
   struct punto *fin;
} linea;

int main()
{
```

```
    linea recta = { 15, {10} };
    return 0;
}
```

Práctica

1) Se cuenta con un archivo de texto que consta de un número determinado de líneas en cada una de las cuales se almacenan oraciones de diferente longitud. En la primera línea se encuentra sólo un entero que indica la cantidad de líneas que siguen. Cada línea comienza con un entero que representa la cantidad de caracteres de la oración, a continuación un espacio en blanco y luego la oración propiamente dicha. Escriba un programa que lea el archivo utilizando la menor cantidad de memoria posible, determine cuál es la línea de menor longitud e imprima en la consola y en un archivo *.txt todas las oraciones menos la de la línea identificada (No se deben mostrar los números). Los nombres de los archivos de entrada y salida deben especificarse en la línea de comandos.

   NOTA: Recordar que los parámetros del main se setean con: Project/Set programms' arguments… Ingresar argumentos en la ventana Program arguments.

2) En un archivo binario se encuentra grabada la información de una pantalla color de dimensiones variables. El primer dato del archivo es un entero que identifica la cantidad de filas de la imagen, el segundo es otro entero que identifica la cantidad de columnas; y el resto del archivo contiene una estructura por cada punto de la imagen. Cada una de ellas está formada por 3 enteros que representan las ponderaciones de los colores primarios (red, green, blue). Se requiere realizar un programa que guarde en memoria la información que contiene el archivo en la forma de una matriz[filas] [columnas]. La matriz debe ser pasada como parámetro a una función que la almacene en un archivo de texto suplantando cada estructura por un único valor que sea la suma de las tres componentes de color. Las columnas de cada fila deben estar separadas por un tabulador.

3) En un archivo binario se encuentra grabada información con el nombre y el año de nacimiento de un conjunto de personas. El programador organizó los datos en forma matricial donde cada elemento es una estructura compuesta por un entero (anio_nac) y un string de 10 caracteres (nombre). En el archivo se guardaron en forma consecutiva la cantidad de filas, la cantidad de columnas y el total de estructuras. Se debe realizar un programa que realice las siguientes operaciones:
   a) Lea los datos del archivo y los almacene en memoria.
   b) Genere un archivo de salida de tipo texto donde se guarden los datos en forma matricial empleando sólo 2 columnas. Cada dato tendrá que tener el siguiente formato: "nac = xxxx, nombre = yyyyyyyyyy (10 caracteres)\t\t".

c) Imprima en pantalla la edad y el nombre de la persona más joven y de la persona de mayor edad.

NOTA: los nombres de los archivos deben pasarse en la línea de comandos: Project/Set programms' arguments… Ingresar argumentos en la ventana Program arguments.

4) Se tiene un archivo de texto con coordenadas de puntos. Realice un programa que recupere la información agrupándolos de a pares y mediante una función que reciba todos los valores calcule y almacene TODAS las distancias euclidianas entre puntos consecutivos. Una segunda función debe imprimir en un archivo de texto las distancias encontradas y determinar e imprimir en pantalla cuál es la mayor. La primer fila del archivo especifica la cantidad de puntos almacenados. La información de cada par se guardó por fila de la forma 'coorx1, coory1; coorx2, coory2'.
NOTA: distancia = sqrt(pow(coorx1-coorx2, 2) + pow(coory1-coory2, 2)) en tipo double

5) Se cuenta con un archivo binario que tiene almacenados números del tipo float. La cantidad total de datos almacenados se agrega al final del archivo como un entero. Escriba un programa en C que almacene los datos en el HEAP mediante una única lectura, los ordene de menor a mayor y finalmente los guarde en un archivo de texto que debe tener el mismo nombre del archivo de entrada pero distinta extensión (*.txt). La función main() debe contener solamente la mínima declaración de variables y las llamadas a las funciones necesarias (modularidad). Recuerde realizar las verificaciones de que los archivos se abren correctamente y que la asignación de memoria es válida.

6) Escriba un programa que ejecute indefinidamente un menú para seleccionar una operación entre números imaginarios de tipo float. Tanto los operandos como el resultado deben almacenarse en una única variable (*sugerencia*: vector de estructuras) que **NO PUEDE** estar almacenada en el **STACK**. Las funciones que realicen las operaciones deben ser invocadas desde el main() y el programador debe asegurarse que las mismas no puedan modificar los valores pasados como parámetros. Las opciones del menú, así como la tecla que permite la selección de cada una deben ser leídas por la aplicación desde el archivo "ejercicio6.txt" y mostradas en pantalla con el mismo formato. En caso que el usuario seleccione la opción de salida, el programa debe solicitar la confirmación de la misma.