# Segundo parcial

Parte teórica (2 ptos)

1. (0.4 ptos)¿Cuál es la relación entre punteros y arreglos?

*El nombre de un arreglo es un puntero CONSTANTE a un conjunto de bytes en memoria. En cambio, un puntero puede cambiar el lugar al que apunta. En ambos casos, el acceso a esos elementos es similar, pueden usarse índices.*

1. (0.4 ptos)¿Cuál es la relación entre una *unión* y una *struct*?

*Son iguales salvo que los campos de la unión comparten la memoria. El tamaño de una unión es igual al campo de mayor tamaño, en cambio el struct ocupa la suma de lo que ocupa cada campo.*

1. (0.2 ptos)¿Qué pasa si se invoca la función *free* pasándole un puntero a un arreglo creado estáticamente?

*Resultados impredecibles.*

1. (0.5 ptos)Explique cómo se comporta la función *realloc* según los diferentes valores de sus parámetros. (NOTA: Recuerde la tabla vista en la teoría)

|  |  |  |
| --- | --- | --- |
| *Buffer* | *Size en bytes* | *Comportamiento* |
| *NULL* | *0* | *Asigna 0 bytes (Idem malloc)* |
| *NULL* | *<> 0* | *Asigna nbytes igual a malloc. Si no es posible retorna NULL* |
| *<> NULL* | *0* | *Devuelve NULL y libera el bloque original* |
| *<> NULL* | *<> 0* | *Reasigna nbytes. El contenido del espacio conservado no cambia. Si la reasignación no es posible, devuelve NULL y el bloque original no cambia.* |

1. (0.4 ptos)¿Qué son los flujos (*streams*) y para qué sirven?

*Los flujos son estructuras de datos intermedias entre nuestro programa y los periféricos externos. Abstraen los detalles del manejo de los dispositivos y permite optimizar los tiempos del intercambio de información (diferencia notable de velocidades)*

*Los programas almacenan e intercambian información leyendo y escribiendo en ficheros(archivos,files).Un fichero se considera como un flujo de datos(stream) de E/S, cuyo destino puede ser una impresora, un monitor o un disco.*

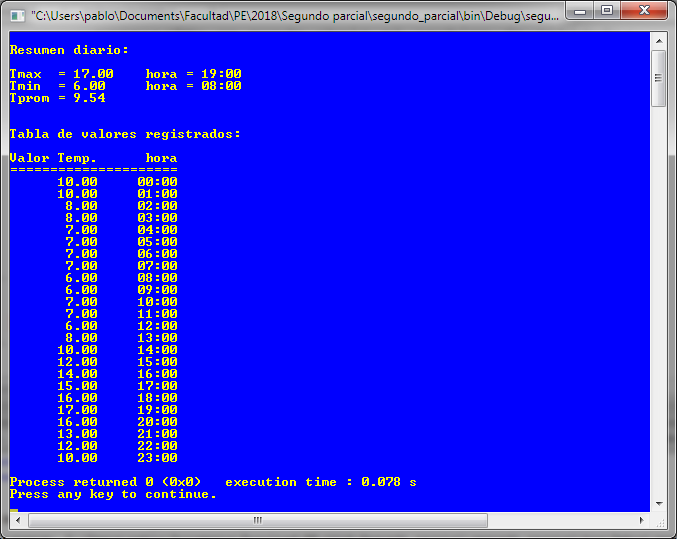
1. (0.1 ptos)¿En qué caso/s es necesario manipular programáticamente la estructura FILE?

*Nunca hay que manipular programáticamente los campos de la estructura FILE.*

Parte práctica (8 ptos)

El prototipo hardware de una estación meteorológica cuenta con un sensor de temperatura, entre otros. El firmware que controla el sensor de temperatura genera diariamente un archivo de texto “csv” que contiene los valores de temperatura registrados a intervalos de 1h.

Se pide realizar un programa en lenguaje C que permita leer todo el contenido de este archivo y lo cargue en la memoria HEAP. Los datos deberán estar estructurados en memoria de manera conveniente, para que la implementación del algoritmo resulte ser simple. Dicho algoritmo deberá calcular, desde los datos cargados en memoria, las temperaturas máxima y mínima, junto con la hora en que esos valores se registraron y la temperatura promedio. La salida estándar deberá ser la que se muestra a continuación



Además, se pide que el programa guarde esta información en un archivo binario, que posteriormente, será leído por otro programa. Para garantizar que la información contenida en el archivo binario puede ser leída, deberá liberar la memoria carga desde el archivo de texto (entrada) y cargarla nuevamente desde el archivo binario (salida) e imprimirla en pantalla nuevamente, reutilizando la función de impresión en pantalla.

## Aclaraciones

* Lea atentamente el enunciado del ejercicio. Durante la primera h se contestarán las dudas que surjan de los enunciados. Recuerde que primero deberá saber QUE hacer y luego pensar COMO hacerlo.
* Deberá entregar junto al proyecto de CodeBlocks este mismo documento, con las respuestas a las preguntas teóricas y el diagrama de flujo (COMPLETO).
* Si rinde el examen en una PC de escritorio recuerde guardar el proyecto de CodeBlocks en la unidad D: y que el path que no contenga espacios ni caracteres especiales como la ñ o las vocales acentuadas. Esto último es igual para las notebooks.
* Los prototipos de función y sus implementaciones deberán estar en archivos separados. Por supuesto que el programa deberá compilar sin errores ni advertencias y deberá comportarse de forma predecible durante su ejecución.
* Junto a este documento se entregará un proyecto de codeblocks parcialmente implementado. Se recomienda leerlo cuidadosamente, hacerlo compilar primero y luego completar las declaraciones e implementaciones necesarias.
* El nombre del archivo de entrada y el de salida se deberán suministrar al programa a través de los argumentos de la función principal *main*().
* El programa deberá liberar todos los recursos utilizados durante el tiempo de ejecución.

**Exitos! =)**

**CRITERIO DE CORRECCIÓN: (8 ptos en total)**

1. Compila sin errores ni advertencias 0.33 p
2. Error en tiempo de ejecución 0.33 p
3. Control al abrir y cerrar archivos0.33 p
4. Control al asignar memoria de manera dinámica y liberación 1p
5. Modularidad 1p
6. Definición de estructura de datos adecuada (anidación de estructuras) 1p
7. Cálculos usando puntero a función para máximo y mínimo 2p
8. Lectura archivo .csv 1p
9. Escritura archivo binario 0.5 p
10. Lectura archivo binario 0.5p