JUNIO 2010

NOTAS IMPORTANTES.-

Debes tener durante todo el examen tu dni o pasaporte sobre la mesa.

- **Debes entrar en la cuenta que figura en tu examen (*login* y *password*). No debes salir de esta cuenta durante el examen y sólo tienes permitido utilizar un editor, el compilador y el depurador. Lee con mucha atención antes de comenzar a trabajar.**
- **Debes crear un fichero en tu cuenta cuyo nombre será: *loginuco.txt* (ej. *i92romeo.txt*) y contendrá tu nombre, apellidos y dni. En caso contrario no se corregirá tu examen.**
- **Debes realizar cada ejercicio en su directorio correspondiente. En caso contrario no se evalurá.**
- **El ejercicio 3 lo tenéis que hacer todos. Si no tienes aprobadas las prácticas resumen, además, tienes que realizarlo correctamente para que se te corrija el examen práctico.**
- **La puntuación indicada en los ejercicios es orientativa. El profesor podrá puntuar el ejercicio en conjunto para tener en cuenta la comprensión global de la asignatura.**

TIEMPO: Primer parcial (1 + ½ hora) – Segundo parcial (3 horas)

PRIMER PARCIAL

(4 puntos) Ejercicio 1 - (Primer parcial)

Escribe un programa en C que, usando funciones, lea una cadena de caracteres, la codifique, carácter a carácter, siguiendo las siguiente reglas y muestre el resultado por pantalla.

- Si el carácter es una letra o dígito, será reemplazado por el siguiente carácter en el conjunto de caracteres, excepto '*Z*' que será reemplazado por '*A*', '*z*' por '*a*' y '*9*' por '*0*'. Por tanto, '*1*' se transforma en '*2*', '*C*' en '*D*', '*p*' en '*q*', etc.
- Cualquier carácter que no sea letra o dígito será reemplazado por un carácter de subrayado ('_').

Por ejemplo, si la cadena de entrada es " *12@9 splash_*", la cadena de salida será "*_23_1_ tqmbti_*".

(6 puntos) Ejercicio 2 - (Primer parcial)

Escribe un programa que implemente las siguientes funciones sobre matrices estáticas de tipo *float* (considera que el programa podrá trabajar como máximo con matrices de 20x20):

(0.75 puntos) void rellenaMatriz(float matriz[][MAX], int nFil, int nCol). Función que rellene una matriz con valores positivos leídos desde teclado. Los valores deberán estar en el intervalo [1-50].

- **(0.25 puntos) void imprimeMatriz(float matriz[][MAX], int nFil, int nCol)**. Función que imprima una matriz por pantalla.
- **(1.25 punto) float calculaMedia(float matriz[][MAX], int nFil, int nCol)**. Función que calcule la media de todos los elementos de la matriz.
- **(1.75 punto) void rotarColumna (float matriz[][MAX], int nFil, int n)**. Función que rote hacia arriba la columna *n* de la matriz. La operación se hará sobre la misma matriz.
- **(2 puntos) int minimoMaximosFilas(float matriz[][MAX], int nFil, int nCol)**. Función que calcule el mínimo de los máximos cada fila.

Deberás estructurar tu programa en dos ficheros, *funciones.c* (contendrá las funciones descritas anteriormente) y *funciones.h* (contendrá los prototipos de las funciones). El fichero *main.c* que se encuentra en tu cuenta (en el directorio ejercicio2), contiene las llamada a las funciones que has implementado; utilízalo para probar tu código.

SEGUNDO PARCIAL

(1.5 punto) Ejercicio 3 – (Segundo parcial) -ELIMINATORIO

Dada la siguiente estructura:

```
    susvdu!dpnqmfkp
!!!|!
!   !!!gmpbu!sfbm<!
!   !!!gmpbu!jnh<!
!   ~!
```

escribe un programa que la use para representar números complejos. El programa leará dos números complejos, realizará la suma de ellos y mostrará el resultado por pantall. Para ello se implementarán, al menos, las siguientes funciones:

- **LeeComplejo.** Función que lee por teclado un número complejo (su parte real y su parte imaginaria), lo almacene en una estructura y lo devuelve haciendo un *return*.
- **SumaComplejos**. Función que recibe como parámetros dos números complejos y devuelve su suma en una nueva estructura pasada *por referencia*. La suma de números complejos se obtiene sumando las partes reales y las partes imaginarias.
  - (a+b.i) + (c+d.i) = (a+c)+(b+d).i
  - (2,3)+(5,6)=(7,9)
- **EscribeComplejo**. Función que recibe un número complejo como parámetro y lo escribe en pantalla (su parte real y su parte imaginaria).

(4 puntos) Ejercicio 4 – (Segundo parcial)

Un fichero binario almacena datos sobre los productos en *stock* de un almacén. Para cada producto, el fichero almacena nombre, codigo, precio y unidades usando la siguiente estructura:

```
    susvdu!qspevdup|
!   !!!dibs!opncsf¥61^<!
!   !!!jou!dpe<!
!   !!!gmpbu!qsfdjp<!
!   !!!jou!vojebeft<!
    }
```

Para gestionar los productos almacenados escribe un programa, sin menú, con las siguientes características:

a) **(0.25 puntos)** Recibirá tres **argumentos en la línea de órdenes** en el momento de ser ejecutado. El primer argumento será el fichero binario con los datos de los productos, el segundo, el fichero de texto que almacenará la salida del programa y el tercer un entero que indicará el sentido de la ordenación (0 para la ordenación ascendente y 1 para la ordenación descendente)

b) **(0.75 puntos).** Actualizará a *30* el número de unidades del producto con código *8*. Para ello se implementará una función que, de forma genérica, actualice en el fichero binario (sin cargarlo en memoria) el número de unidades de un producto dado su código.

c) **(0.25 puntos)** Calculará el número de productos que hay en el fichero binario sin recorrerlo.

d) **(1 punto)** Almacenará los registros del fichero binario en un vector dinámico y lo mostrará por pantalla.

e) **(1.25 punto)** Ordenará el vector anterior por el método que se quiera, usando como campo clave el precio. Para determinar el sentido de la ordenación (de menor a mayor ó de mayor a menor) se pasará como parámetro un puntero a la función de comparación. En caso de no utilizar punteros a funciones restarás **0.5 puntos** de la nota del apartado.

f) **(0.5 punto)** Almacenará el vector ordenado en un fichero de texto con el siguiente formato.

```
Opncsf!Dpejhp!Qsfdjp!Vojebeft!
.........................................!
Cbcfsp!!!!!!!2!     !     !4/:6!!     !!!4!
```

Divqfuf! !!3! ! !6/61!! !!!9!

Estructura tu programa en los siguientes ficheros, *principal.c* (contendrá el *main*), *ficheros.c* (funciones relacionadas con los ficheros), *varias.c.* (resto de funciones, ordenación, reserva, ...) y los correspondientes .h (*ficheros.h* y *varias.h*).

(3.25 puntos) Ejercicio 5 – (Segundo parcial)
Escribe un programa en C que dado un fichero de texto con una palabra en cada línea, almacene en una lista enlazada cada palabra del fichero y su frecuencia. Cada nodo de la lista almacenará una de las palabras del fichero y el número de veces que aparce en el fichero, usando la siguiente estructura:

```
!  tusvdu!opep!
!!!|!
!!!!!!dibs!qbmbcsb¥61^<!
!!!!!!jou!wfdft<!
!!!!!!tusvdu!opep+!tjh<!
!!!~<!
```

El programa deberá realizar las siguientes operaciones de forma secuencial: Leer un fichero de texto y crear una lista enlazada con las palabras del fichero y el número de veces que aparecen. A continuación, se escribirá la lista en pantalla y se borrará de la lista el primer elemento con frecuencia 5 mostrando el resultado en pantalla.

Para ello, implementa, al menos, las siguientes operaciones:
(0.75 puntos) *creaListaPalabras*. Función que lee palabras de un fichero de texto con la estructura descrita anteriormente y devuelve una lista en la que cada nodo contiene una palabra diferente del fichero y el número de veces que aparece en el fichero.
(0.25 puntos) *insertaPalabra*. Función que inserta una nueva palabra en la lista. Recuerda que la lista no tiene palabras repetidas.
(0.5 puntos) *actualizaNodo*. Función que recibe una palabra y actualiza el nodo de la lista que almacena esa palabra, aumentando en 1 el número de veces que aparece la palabra en el fichero.
(0.5 puntos) *pertenece*. Función que comprueba si una palabra ya está en la lista.
(0.5 puntos) *escribeLista*. Función que muestra por pantalla los elementos de la lista (recursivamente).
(0.75 puntos) *borraElemento*. Función que borra el primer elemento de la lista cuya frecuencia (nº de veces que aparece la palabra en el fichero) sea la que se indica como parámetro.

Estructurara tu programa en dos ficheros, listas.c y cabecera.h.

(1.25 puntos) Ejercicio 6 – (Segundo parcial)
Para el desarrollo de un proyecto sobre pasatiempos, se tienen los siguientes ficheros:
reservaMemoria.c
funciones para la reserva de memoria de diferentes estructuras de datos
liberaMemoria.c
funciones para liberar memoria
memoria.h
Prototipos de las funciones de reserva y liberación de memoria
fichero.c – fichero.h
funciones relacionadas con la E/S de datos en archivos y sus prototipos
crucigrama.c – crucigrama.h
main, funciones específicas para la creación de crucigramas y sus prototipos
sopaLetras.c – sopaLetras.h
main, funciones específicas para la creación de sopas de letras y sus prototipos

El resultado final del proyecto serán dos ejecutables (*crucigrama.x y sopaLetras.x*) que permitirán la creación de crucigramas y sopas de letras. Crea un fichero *makefile* con las siguientes características:

- **(0.25 puntos**) Construirá una biblioteca (*libMemoria.a*) a partir de los ficheros objetos de *reservaMemoria.c* y *liberaMemoria.c.*
- **(0.25 puntos)** Construirá el ejecutable *crucigrama.x* a partir de la biblioteca y los ficheros objetos de *fichero.c y crucigrama.c*
- **(0.25 puntos)** Construirá el ejecutable *sopaLetras.x* a partir de la biblioteca y los ficheros objetos de *sopaLetras.c y fichero.c.*
- **(0.25 puntos)** Permitirá generar los dos ejecutables con una única llamada de la orden *make.*
- **(0.25 puntos)** Permitirá eliminar los ficheros objetos generados.

El fichero *makefile* que construyas debe funcionar correctamente. Para probarlo puedes utilizar los ficheros que se encuentran en el directorio **ejercicio6**.

Notas:

Para que los ejercicios sean puntuados, han de funcionar correctamente y han de seguir las especificaciones reseñadas en cada uno de ellos.

- **Excepto en las funciones específicas de entrada/salida y el *main*, no deben pedirse o mostrarse valores de variables dentro de las funciones. En caso contrario no se evaluará el ejercicio aunque funcione correctamente.**
- **La salida por pantalla será clara y ordenada, en caso contrario restarás un punto a tu nota final.**
- **Si los nombres de los archivos no se corresponden con los indicados en el enunciado del examen restarás un punto de tu nota final.**
- **Aquellos apartados que se implementen sin usar funciones no puntuarán, aunque funcionen correctamente.**
- **No se podrá hacer uso de variables globales.**
- **No olvides firmar tu examen y poner la hora de entrega.**