Relación de prácticas de la asignatura
METODOLOGÍA DE LA PROGRAMACIÓN
Segundo Cuatrimestre
Curso 2017-2018

1º Grado en Informática

Práctica 3: Recursividad, Ficheros y Argumentos en Línea de órdenes

Objetivos

- Practicar conceptos básicos sobre recursividad, ficheros de texto y ficheros binarios en C.
- Usar argumentos en línea de órdenes para que los programas puedan recibir argumentos en el momento de ser ejecutados.

Recomendaciones

- Pasa a tus funciones el nombre del fichero como parámetro y ábrelo y ciérralo dentro de la función.
- Organiza los ejercicios 9 y 12 en al menos tres ficheros .c (*main.c, ficheros.c, util.c*) y dos ficheros *.h* (*ficheros.h y util.h*).

Temporización recomendada

3 sesiones de prácticas

- 1ª sesión: Ejercicios de recursividad
- 2ª sesión: Ejercicios de ficheros de texto
- 3ª sesión: Ejercicios de ficheros binarios

¿Qué hay que entregar?

El análisis y diseño de la función del ejercicio 3 y de la función *contarLibrosDiferentes* (ejer. 9).
El análisis consistirá en el estudio del problema que plantea el ejercicio.

- Qué datos de entrada necesita y de qué tipo son.
- Cómo van a llegar esos datos.
- Qué resultado se va a obtener y de qué tipo es.
- Cómo se obtiene el resultado a partir de los datos de entrada.
- Cómo se va a presentar al usuario el resultado final.
- Ejemplo de que la solución propuesta funciona, utilizando los nombres dados a los datos.

El diseño incluirá un algoritmo en pseudocódigo o diagrama de flujo que resuelva el problema y que servirá como base para la posterior codificación. Recordad que **el diseño es independiente del lenguaje de programación utilizado.**

¿Cuándo hay que entregar el ejercicio 3 ?

Grupo	GM1-GM2	GM3 -GM4	GM5
Fecha	10/04/18	11/04/18	13/04/18

¿Cuándo hay que entregar el análisis y el diseño de la función función contarLibrosDiferentes?

Grupo	GM1-GM2	GM3 -GM4	GM5
Fecha	17/04/18	18/04/18	20/04/18

Recursividad

1. La siguiente función recursiva se encarga de multiplicar los números comprendidos entre [inicio,limite):

```
int funcionRecursiva (int inicio, int limite) {
int retorno;
if (inicio > limite)
  retorno = -1;
else
  if (inicio == limite)
    retorno = 1;
  else
    retorno = inicio * funcionRecursiva(inicio+1, limite);
return retorno;
}
```

Se pide:

   1. Identificar el caso de parada de la función *funcionRecursiva.*
   2. Realizar un esquema sobre las variaciones en la pila de control.
   3. ¿Qué resultado daría la llamada *funcionRecursiva(14,10)*?
   4. ¿Qué resultado daría la llamada *funcionRecursiva(4,7)*?

2. Escribe una solución recursiva que calcule el algoritmo de *Euclides*, usado para calcular el máximo común divisor de dos enteros. El algoritmo de *Euclide*s se describe del siguiente modo:

```
mcd(x, y) = x, si y = 0
mcd(x, y) = mcd (y, mod(x,y)) si y > 0  y x >= y
```

3. Construye una función recursiva que calcule la división entera de dos números mediante el métodos de restas sucesivas. Implementa un pequeño programa para probarla.

4. Dada una cadena *c,* diseña una función recursiva que cuente la cantidad de veces que aparece un carácter *x* en *c*.
   Ejemplo*: para c = "elementos de programación" y x = 'e', el resultado es 4*.

5. Codifica una función recursiva que permita sumar los dígitos de un número. Implementa un programa para probarla. Ejemplo: Entrada: 124 Resultado: 7.

Ficheros de texto

6. Codifica un programa en C que, utilizando funciones, abra un fichero en modo **texto**, cuyo nombre se pedirá al usuario, y genere un nuevo fichero llamado "mayusculas-nombreFicheroDeEntrada.txt", que tenga el mismo contenido que el fichero original pero en mayúsculas.

7. Codifica un programa en C que, utilizando funciones, cree un fichero **texto** con números enteros generados aleatoriamente en un intervalo especificado por el usuario. El programa le preguntará al usuario el nombre del fichero a crear, el número de elementos que contendrá el fichero y los extremos superior e inferior del intervalo.

8. Codifica un programa en C que, utilizando funciones, calcule la media de los elementos pares que hay en un fichero de texto generado por el ejercicio **7**. El nombre del fichero se pasará como argumento en la línea de órdenes.

9. Construye un programa que gestione mediante ficheros de **texto** el *stock* de libros de una librería. Para cada libro se almacenarán tres líneas en un fichero de texto (*stock*):
   - en la primera línea el *título*,
   - en la segunda línea el *autor,*
   - y en la tercera línea *el precio* y las *unidades* disponibles del libro.

   El programa contará con un menú que permitirá realizar las siguientes operaciones:
   - Comprobar la existencia de un determinado libro en el *stock* buscando por su título.
   - Introducir un nuevo libro en el *stock*.
   - Contar el número de libros (títulos) diferentes que hay en el *stock*.
   - Listar los libros almacenados en el *stock* almacenándolos previamente en un vector dinámico.
   - Vender un libro buscándolo por su título.
   - Borrar aquellos registros con 0 unidades disponibles.
   - Salir.

   El nombre del fichero que almacena el *stock* de la librería se pasará como argumento en la línea de órdenes.

Ficheros binarios

10. Codifica un programa en C que, utilizando funciones, cree un fichero **binario** con números enteros generados aleatoriamente en un intervalo especificado por el usuario. El programa guardará los números en un vector dinámico antes de volcarlo a disco. El programa recibirá 4 parámetros como argumentos en la línea de órdenes: nombre del fichero a crear, número de elementos que contendrá el fichero y los extremos superior e inferior del intervalo.

11. Codifica un programa en C que, utilizando funciones, lea números enteros desde un fichero binario generado en el ejercicio anterior, almacene sus valores en un vector dinámico y, a continuación, calcule la media de los números pares. El nombre del fichero se le preguntará al usuario.

12. Construye un programa que gestione, mediante ficheros **binarios**, el *stock* de libros de una librería. Para cada libro se almacenará la siguiente estructura:

```
struct libro {
  char titulo[100];
  char autor[50];
  float precio;
  int unidades;
}
```

El programa contará con un menú que permitirá realizar las siguientes operaciones:

- Comprobar la existencia de un determinado libro buscando por su título.
- Introducir un nuevo libro en el *stock*.
- Contar el número de libros (títulos) diferentes que hay en el *stock*.
- Listar los libros almacenados en el *stock* almacenándolos previamente en un vector dinámico
- Vender un libro buscándolo por su título sin cargar el fichero en memoria.
- Borrar aquellos registros con 0 unidades disponibles.
- Salir.

El nombre del fichero que almacena el *stock* de la librería se pasará como argumento en la línea de órdenes.