# METODOLOGÍA DE LA PROGRAMACIÓN II

## ENTRADA/SALIDA EN C++

1. Haz un programa que lea un texto desde consola y que lo escriba en un fichero de disco.
2. Haz un programa que reciba como parámetro un fichero y que lo muestre en consola (de forma similar a como lo hace la orden *cat* de linux).
3. Haz un programa que lea una serie de números por teclado y que los almacene en un fichero de texto.
4. Haz un programa que lea el fichero escrito por el programa del ejercicio 3 y que muestre en consola la suma de todos esos números.
5. Haz un programa que reciba como parámetros los nombres de dos ficheros binarios y que copie el contenido del primero en el segundo.
6. Haz un programa que cuente el número de letras 'a' que hay en un fichero de texto.
7. Haz un programa que cuente el número de líneas que hay en un fichero.
8. Haz un programa que cuente el número de palabras que hay en un fichero.
9. Lee un fichero de texto y cópialo en otro sin poner las vocales.
10. Lee un fichero de texto y muéstralo por consola en orden inverso usando las funciones de posicionamiento.
11. Haz una función que calcule el tamaño de un fichero.
12. Haz un programa que lea un fichero de texto y lo muestre en consola saltándose aquellas líneas que comiencen por el carácter '#'.
13. Hacer un programa que permita formar el nombre de un fichero de la forma "salidaXXX.Z.dat" a partir de dos números que le damos a través de la línea de comandos. XXX es un número de 3 dígitos (se rellena con ceros a la izquierda si es necesario) y Z es un número con cualquier cantidad de dígitos. A continuación tienes varios ejemplos de ejecución (nuestro programa se llamará *componer*):
    ```
    componer 45 6          generaría el nombre de fichero "salida.045.6.dat"
    componer 5 67          generaría el nombre de fichero "salida.005.67.dat"
    ```
    Nota: usa flujos de tipo *stringstream*.
14. Dada la siguiente clase[1]:
    ```
    enum Ficha {Blanca, Negra};
    class Jugador {
     private:
        char nombre[30];
        float puntos;
        int ganadas, perdidas, empatadas;
        Ficha color_de_ficha;
    };
    ```
    *a*) Implementa un método público al que le pasemos como argumento un string con el nombre de un fichero y que añada los datos de la clase al final del fichero (este fichero podrá almacenar datos de varios jugadores).
    *b*) Implementa otro método al que le demos un string con el nombre de un fichero y un número N y que cargue los datos del jugador N-ésimo que está almacenado en el fichero.
    *c*) Haz un tercer método al que le indiquemos el string con el nombre del fichero y un número N y que modifique los datos del jugador N-ésimo del fichero con los que tenga el objeto que llama al método.
    *d*) Modifica las funciones anteriores para optimizar el espacio en disco de forma que sólo se almacenen los caracteres útiles del nombre (Si el nombre es "Juan", almacenaremos sólo esos 4 caracteres y no los 26 restantes hasta ocupar un total de 30 bytes).
15. Haz un programa que permita encriptar y desencriptar el contenido de un fichero de texto. Para encriptar sustituiremos cada letra por la letra que está 3 posiciones mas adelante en el alfabeto (para las últimas letras ciclamos el alfabeto). Los caracteres que no sean letras se quedarán igual.
16. Haz un programa que reciba como parámetros 3 nombres de ficheros. Los dos primeros ficheros contienen números y están ordenados. El programa tomará los datos de esos ficheros y los irá metiendo en el tercer fichero de forma que al finalizar la secuencia esté también ordenada.
17. Aunque sea ineficiente, implementa el TDA lista enlazada usando un fichero para almacenar los datos en lugar de usar memoria dinámica.

[1] Para los 3 primeros apartados puedes considerar que el nombre tiene un número fijo de caracteres (30). El nombre podrá contener espacios en blanco