# PRACTICA DEL MODULO 5

## Ejercicios

### Breve descripción

Ejercicios de programación para el módulo 5.

**Versión 1.1**

Martin Jerman
Martin.jerman@inspt.utn.edu.ar

# Ejercicios con archivos de texto

Importante: se asume que los archivos `.txt` son de texto y los archivos `.dat` son binarios.

1. Preguntas
   - La función `fclose()`, utiliza el nombre externo de un archivo o la variable de este?
   - Y la función `fprintf()`?
   - ¿Cuál es la función que usa ambos nombres, el externo y el de la variable asociada?
   - Siendo `arch` una variable `FILE`, explique la diferencia entre:
     ```
     arch = fopen("archivo.txt","rt");
     arch = fopen("archivo.txt","wt");
     ```
   - La instrucción `arch = fopen("archivo.txt","wt");` presenta algún peligro potencial?

2. Se tiene el archivo `pro1.txt`. Escriba las instrucciones para abrirlo en modo lectura y cerrarlo.

3. Desarrolle una función `contarVocales` que reciba un archivo de texto ya abierto y devuelva la cantidad de vocales que hay en el archivo.

4. Desarrolle un programa que abra el archivo `cuento.txt` y lo muestre por pantalla.

5. Desarrolle una función `fLog()` que reciba una cadena y grabe en un archivo de texto `debug.log` el día, la hora y la cadena. La función valida si el archivo existe y lo crea si es necesario. Para trabajar el día y la hora, revisar este link.

6. Desarrolle una función que lea el archivo `config.txt` que tiene el siguiente contenido:
   ```
   Archivo de configuración
   Tamaño de arrays: 25
   Año: 2019
   Máximo de líneas: 150
   ```
   Y devuelva los datos leídos en un `struct`. La función **no debe** recibir datos por parámetro.

7. Desarrolle un programa que pida por teclado el numero de legajo de un alumno y dos notas; que calcule el promedio y guarde toda la información en un archivo de texto `alumnos.txt`. El ingreso de datos finaliza con un legajo igual a `-1`.
8. Modifique el programa anterior para que lea el archivo `alumnos.txt` y lo muestre por pantalla.

9. Desarrolle la función `mostrarArchivoTokenizado()` que recibe un archivo de texto ya abierto y muestre por pantalla el contenido del archivo, una palabra por línea. NOTA: los separadores son todos los signos de puntuación. Luego impleméntela en un programa.

10. Basado en el ejercicio anterior, desarrolle la función `tokenizarArchivo()` que reciba un archivo de texto ya abierto y cree uno nuevo llamado `archTokenizado.txt` que contenga el contenido del primero pero con una palabra por línea.

11. Dado un archivo `promedios.txt` con la siguiente forma:
    ```
    11111  4.50
    11112  8.70
    33331  1.20
    ```

Desarrolle un programa lea dicho archivo y guarde en otro nuevo `aprobados.txt` aquellas líneas cuyo promedio es mayor a `7`. El nuevo archivo debe contener el titulo `Los aprobados son`.

12. Escribe un programa para actualizar, mensualmente, el archivo de `sueldos.txt` de una empresa. Cada grupo de datos en el archivo contiene el nombre del empleado, el sueldo semanal, y finalmente los meses y años de antigüedad en la empresa. Al fin del mes, cuando se ejecuta el programa, se debe incrementar la antigüedad en meses (y en años si corresponde), y deben darse aumentos. Si el empleado tiene al menos 5 años, recibe un aumento del 30%, y si supera los 10 años, se le aumenta un 35% adicional.

13. Cada grupo de datos para el archivo `clientes.txt` contiene el nombre del cliente, dirección, género y saldo de la cuenta en un supermercado. Escribe un programa que:
    - Cree dos archivos separados, `mujer.txt` y `varon.txt` con todos los datos de los clientes almacenados en el archivo correcto.
    - Muestre en pantalla la siguiente información:
        i. Número total de clientes
        ii. Número total de varones
        iii. Número total de mujeres
        iv. Promedio de saldos para hombres
        v. Promedio de saldos para mujeres

14. Desarrolle un programa que pida por teclado un legajo, nombre, edad guarde la información en un archivo de texto con la siguiente forma:
    Registro #`unNum`
    Legajo: `unLegajo`
    Nombre: `unNombre`
    Edad: `unaEdad`
    `Línea vacía`
    El ingreso de datos se finaliza con un legajo igual a cero. Los datos ingresados se guardan en el archivo `datos.txt`.
    Luego desarrollar otra función que lea dicho archivo y lo muestre por pantalla con el formato `unNum-unLegajo|unNombre|unaEdad`. Implementar la función al final del programa para que muestre el contenido del archivo.

# Ejercicios con archivos binarios

15. Escribe la declaración del tipo estructurado que se guardará en un archivo de datos de los socios del club "San Antonio de Padua". Cada registro contendrá los siguientes campos (elementos de la estructura): nombre, género, edad, fecha último pago de abono.

16. Para el archivo del ejercicio anterior realiza un programa que permita crear el archivo y guardar registros. Luego, construye una función `mostrarData()` que abra el archivo para lectura, muestre sus registros en pantalla y cierre el archivo.

17. Desarrolle un programa que genere un archivo binario `personas.dat` cuya estructura interna sea: `identificador(int)`, `nombre(15)`, `apellido(15)`, `genero(char)`, `fechaNacimiento(int DDMMAAAA)`, `localidad(15)`, `salario(float)`. Una vez creado el archivo, el programa debe mostrarlo por pantalla. Utilizaremos este archivo en ejercicios siguientes.

18. Utilizando el archivo `personas.dat` del ejercicio 16, desarrolle una función que reciba el archivo abierto y una posición, y devuelva el registro que se encuentra en dicha posición.

19. Utilizando el archivo `personas.dat` del ejercicio 16, desarrolle una función que reciba el archivo abierto y devuelva el ultimo registro del archivo.

20. Utilizando el archivo `personas.dat` del ejercicio 16, desarrolle una función que reciba el archivo abierto y devuelva los últimos 5 registros del archivo.

21. Utilizando el archivo `personas.dat` del ejercicio 16, desarrolle una función que reciba el archivo abierto y una posición y devuelva el registro de dicha posición y los 5 registros siguientes.

22. Utilizando el archivo `personas.dat` del ejercicio 16, desarrolle un programa (con funciones) que guarde en el archivo `adultos.dat` a todos los mayores de edad. NOTA: para saber la fecha actual, ingresarla por teclado como un int.

23. Utilizando el archivo `personas.dat` del ejercicio 16, desarrolle un programa que incremente el salario de cada persona en un 10% (modificando el archivo original).

24. Utilizando el archivo `personas.dat` del ejercicio 16, desarrolle una función devuelva el tamaño del archivo en bytes.

25. Utilizando el archivo `personas.dat` del ejercicio 16, desarrolle una función devuelva la cantidad de registros en el archivo.

26. El archivo binario `estudia.dat`, contiene registros con los siguientes datos de los estudiantes de una universidad:
    - Nombre
    - Número de Legajo
    - Cantidad de materias aprobadas
    - Promedio de calificaciones

    El archivo se encuentra ordenado en orden ascendente por número de legajo. El programa debe presentar un menú con las siguientes opciones:
    - Mostrar todos los datos de un estudiante, cuyo nro. de legajo se ingresa por teclado.
    - Mostrar todos los datos de los estudiantes cuyo promedio sea mayor o igual al ingresado por el usuario, y además que tengan no menos de 10 materias aprobadas.

    NOTA: Previamente deberá realizar un programa auxiliar para la creación del archivo de estudiantes.

# Ejercicios integradores

27. Explica las siguientes funciones, aportando ejemplos de cada una: `feof()`, `fscanf()`, `fseek()` y `fread()`.

28. Dado un archivo de texto, escribir un programa que cuente el número de palabras que aparezcan en dicho archivo.

29. Escribe un programa que copie en un archivo otros dos archivos, uno a continuación de otro.

30. Se tiene el archivo binario `acciones.dat` cuya estructura de registros es `idAccion(int), valor(float)`. Se pide un programa que implemente una función que reciba el archivo abierto y devuelva un vector con los precios máximos y mínimos de cada acción.

31. Tenemos un archivo de registros cuya estructura están definidas según la declaración:

```
struct registro1{
      char partido[60];
      char localidad[60];
      int candidatos;
      }
```

Se quiere sustituir la estructura del registro por el siguiente:

```
struct registro2{
      char partido[60];
      char localidad[60];
      int candidatos;
      int elegidos;
      }
```

Escribe un programa que modifique la estructura y traslade toda la información del archivo primitivo al nuevo, inicializando a cero el nuevo campo creado.

32. Dada una palabra ingresada por teclado, el programa debe informar si esta palabra se encuentra escondida en el texto del archivo `cuento.txt`. Una palabra esta escondida cuando sus letras se encuentran en el texto (no necesariamente consecutivamente), por ejemplo, la palabra "elefante" está escondida en la frase "**el** h**e**rmano de **F**r**an**cisco **t**uvo su**e**rte" (ver negritas). De encontrarse, debe informarse la cantidad de caracteres entre los que se encuentra escondida.

33. Dado el archivo abreviaturas.txt cuyo formato es `palabra:abreviatura`, desarrollar un programa que lea el contenido del archivo `cuento.txt` y genere uno nuevo `cuentoAbreviado.txt` en el que se reemplacen las palabras que pueden ser abreviadas.

34. Dado un valor entero por teclado, elaborar un programa que busque en un archivo (`numeros.dat`) de enteros ese valor, y en caso de encontrarlo, que diga en qué posición o posiciones del archivo se encuentra.

35. Se define el tipo siguiente:

```
struct alumno {
      char apellidos[60];
      char nombre[20];
      float notajunio,notasept,notaprac;
      }
```

En un archivo con datos de esta estructura disponemos de los alumnos matriculados en una asignatura. Se quiere obtener un archivo de texto en el que aparezca en cada línea el nombre de un alumno con el formato apellidos, nombre, nota junio.

36. En un archivo se tiene grabada una sucesión de números enteros positivos o nulos, correspondientes a las puntuaciones de varios jueces sobre un mismo ejercicio. Escriba un algoritmo para calcular la media aritmética de los valores estrictamente positivos. También debe calcular la cantidad de ceros.

37. Escribe un programa que permita llevar los gastos e ingresos menores a $25000 de una empresa. El programa presentará un menú como el siguiente:

```
CONTROL DE GASTOS
==================
1. Ingresos
2. Gastos
3. Salida
==================
```

Donde las funciones realizan las siguientes operaciones:

- Menú muestra el menú en la pantalla.
- lee_opcion lee la opción del teclado.
- Ingresos lee nuevos ingresos y los suma al saldo actual.
- Gastos lee nuevos gastos y los resta del saldo actual (que nunca podrá ser negativo).
- Salida muestra el saldo actual (ingresos - gastos).

38. Dados dos archivos binarios y secuenciales: "A.DAT" y "B.DAT" con el siguiente formato de registros:

```
struct Treg {
      int Clave: integer;
      char Dato[10];
      }
```

Ambos archivos están clasificados en forma ascendente por el campo Clave y dentro de un mismo archivo no hay claves duplicadas.
Escribe cada una de las siguientes funciones:

- Unión: se genera un archivo C resultado de la unión entre A y B, es decir el archivo C contendrá todos los registros consignados en los archivos de entrada. En caso de existir claves repetidas en el Archivo A y el B, se grabará un solo registro en C utilizando los datos del archivo B.
- Intersección: se generará el archivo C conteniendo solo los registros del Archivo A cuyas claves también aparezcan en el Archivo B.
- Diferencia: el archivo C tendrá los registros que estén en el primer archivo que se pase como parámetro y cuyas claves no aparezcan en el archivo que se pase como segundo parámetro.

39. Un club cuenta con un archivo maestro de acceso secuencial denominado `socios.dat`. Este archivo se encuentra ordenado por código de socio y tiene los siguientes campos: `Código de socio`, `Nombre` y `apellido`, `Dirección`, `Deuda` (array de 12 posiciones, cada posición contendrá la deuda correspondiente a ese mes).
Se cuenta además con un archivo `bajas.dat` de acceso secuencial ordenado, con los códigos de socio a ser dados de baja. Se desea:

- Actualizar el archivo SOCIOS con el de BAJAS.
- Emitir un listado ordenado por código de socio con: código de socio, nombre y apellido, dirección y total adeudado de aquellos socios que hayan sido dados de baja.

Se supone que no hay duplicados en el archivo Socios, pero si puede haberlo en el de Bajas.

Se cuenta además con un archivo `pagos.dat` de acceso secuencial, con los datos de los pagos realizados por los socios. Este archivo tiene registros con la siguiente estructura:

- Código de socio
- Mes

- Monto abonado

Este archivo está ordenado en forma ascendente por código de socio.
Se desea:

- Actualizar el archivo SOCIOS con la información del archivo Pagos.
- Emitir un listado ordenado por código de socio con: código de socio, nombre y apellido, total adeudado con aquellos socios que mantengan deudas con el club.

Se supone que no hay duplicados en el archivo Socios, pero si puede haber varios registros para un mismo socio en el archivo de pagos.

40. Dados dos archivos binarios `a.dat` (acceso directo y secuencial) y `b.dat` (acceso secuencial) con el siguiente formato de registros:

```
struct Treg {
   int Clave: integer;
   char Dato[10];
   }
```

Se desea realizar un algoritmo que actualice los datos del archivo A con los del B. Teniendo en cuenta que la clave en A coincide con el número de registro. Toda clave en B estará en A.

41. Desarrollar un algoritmo que permita la actualización de un archivo de solicitudes de planes de ahorro para compra de automóviles. Se cuenta con:
    - Un archivo secuencial ordenado en forma ascendente por el campo `nroSolicitud` donde están registradas todas las solicitudes vigentes. Los registros de este archivo tienen la siguiente estructura:

```
registroSolicitud:
      nroSolicitud: entero
      titular: cadena de 40 caracteres
      marca: cadena de 20 caracteres (12 marcas diferentes)
      modelo: carácter (válidos: A,B,C,D)
      color: entero (1 a 8)
```

    - Un archivo también secuencial que informa las bajas a procesar. Este archivo está ordenado en forma ascendente por el campo nroSolicitudBaja y sus registros tienen la siguiente estructura:

```
registroBaja
      nroSolicitudBaja: entero
      causa: carácter (R,F,O)
```

Se quiere generar un nuevo archivo de solicitudes resultado de efectuar las bajas correspondientes. (Este archivo tendrá la misma estructura que el archivo de solicitudes.)

Se podrán rechazar bajas por causa inválida, o baja a registro inexistente. Las bajas rechazadas serán listadas indicando el porqué del rechazo.

Al final del proceso se emitirá un informe indicando para cada marca, la cantidad de solicitudes existentes por modelo y color.

También al final del proceso se indicará la cantidad de registros leídos en cada uno de los archivos de entrada, la cantidad de registros grabados en el archivo actualizado, la cantidad de bajas rechazadas y listadas y la/s marca/s con mayor cantidad de solicitudes vigentes (en el actualizado).