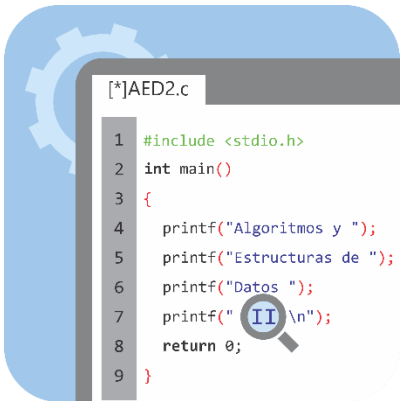

Algoritmos y Estructuras de Datos II

Trabajo Práctico 8 - Primera Parte
Archivos. Operaciones simples.

OBJETIVOS:

- Incorporar los conceptos básicos sobre almacenamiento de datos en archivos
- Procesar archivos de tipo "texto".
- Procesar archivos "binarios" con estructuras de registros.
- Aplicar técnicas de detección de máximos y mínimos.
- Incorporar buenas prácticas de programación que contribuyan a la legibilidad, mantenibilidad y extensibilidad del software.

COMPETENCIAS

- Identificar, formular y resolver problemas mediante programación.
- Utilizar de manera efectiva técnicas y herramientas de aplicación para desarrollar software.
- Desempeñarse de manera efectiva en equipos de trabajo.
- Aprender en forma continua, autónoma y de manera colaborativa.

METODOLOGÍA

- El alumno deberá resolver individualmente los ejercicios propuestos.
- El alumno deberá codificar las soluciones en el lenguaje de programación C.
- Realizar consultas a través del canal de slack/discord/whatsapp correspondiente a su comisión ó del aula virtual de la asignatura.

DURACIÓN

De acuerdo a la planificación de la asignatura, se deberá utilizar para la resolución de los ejercicios de esta serie, dos clases prácticas.

EJERCICIOS PROPUESTOS

Apertura de un archivo

- **fopen**(*nombre_archivo, modo_apertura*): la función fopen() abre una secuencia para que pueda ser utilizada, y la asocia a un archivo. Ejemplo:

```
FILE * archivoTexto;
archivoTexto = fopen("Notas.txt", "r");
```

Los modos de apertura permitidos son:

Modo	Significado
r	Abre un archivo de texto para lectura
w	Crea un archivo de texto para escritura
a	Abre un archivo de texto para añadir
rb	Abre un archivo binario para lectura
wb	Crea un archivo binario para escritura
ab	Abre un archivo binario para añadir

- **fclose**(*nombre_archivo*): cierra una secuencia que fue abierta mediante una llamada a fopen().
Ejemplo: fclose(archivoTexto);

- **feof**(*nombre_archivo*): se utiliza para detectar cuando no quedan más elementos en un archivo. Devuelve verdadero (true) si detecta fin de archivo. En cualquier otro caso, devuelve 0. Se aplica de la misma forma a archivos de texto y binarios.
Tener en cuenta que esta función no indica fin de fichero hasta que no se intenta volver a leer después de haber leído el último elemento del archivo. Por lo tanto, en el programa debe hacerse la lectura siguiendo estos pasos:

```
Leer (obtener) un registro del archivo
mientras (!feof(archivo)
        Procesar el registro leído
        Leer (obtener) un registro del archivo
fin-mientras
```

De este modo se lee/obtiene un registro y solo se procesa si se verifica que lo que se ha leído no es la marca de fin de archivo. Esta técnica se llama de lectura adelantada.

Procesando datos con archivos de texto (.txt)

En programación, en algunos casos es necesario procesar **archivos de texto**, para lo cual hay que acceder a lo que contienen (**leer** sus datos) y otras veces hay que generar estos archivos (**escribir** o **grabar**).
Las siguientes funciones son útiles a la hora de trabajar con archivos de texto:

- **getc**(*nombreArchivo*): permite leer un carácter del archivo de texto
Ejemplo: char letra = getc(archivoTexto);
- **fputs**(*cadena*, *nombreArchivo*): permite escribir/insertar una línea de texto plano sin tipo, en un archivo de texto
Ejemplo: fputs("Esto es una línea de texto", archivoTexto);

- **fprintf**(*nombreArchivo*, *cadena de control*): permite escribir/insertar una línea de texto en el archivo teniendo en cuenta el tipo de dato en cada caso

Ejemplo:

```
linea.dni = 50489254;
strcpy(linea.nombre, "Mariano");
fprintf( archivoTexto, "%d %s\n", linea.dni, linea.nombre);
/* linea es un registro o struct que tiene dos campos: dni y nombre */
```

- **fscanf**(*nombreArchivo*, *cadena de control*) permite leer una línea de texto del archivo teniendo en cuenta el tipo de dato en cada caso

Ejemplo:

```
fscanf(archivoTexto, "%d %s", &linea.dni, &linea.nombre);
/* linea es un registro o struct que tiene dos campos: dni y nombre */
```

1. El archivo Fix_you.txt contiene la letra de la canción de Coldplay del mismo nombre. Escribir un programa que lea este archivo y muestre la canción por pantalla.

2. Escribir un programa que permita grabar un archivo de texto con datos de alumnos contenidos en una línea de texto que se ingresa por teclado. Cada línea contiene los siguientes datos: DNI, nombre del alumno y finaliza con un salto de línea. Ingresar alumnos hasta que el usuario indique que no desea continuar ingresando datos. Informar al final la cantidad de registros grabados.

3. Escribir un programa que lea el archivo de texto generado en el ejercicio anterior y visualice por pantalla cada registro del archivo.

4. Dado un archivo de texto clientes.txt que contiene, en cada línea, los siguientes datos de clientes de un banco: número de cuenta, nombre del cliente; realizar un programa que lea este archivo y copie el contenido en un nuevo archivo de texto clientes_backup.txt. Visualizar al final la cantidad de registros copiados.

Procesando datos con archivos binarios (.dat)

En otras ocasiones, en programación es útil manipular archivos de datos binarios. Es otra forma de persistir los datos o almacenarlos en memoria secundaria.

Las siguientes funciones son útiles a la hora de trabajar con archivos binarios:

- **fread**(*punteroAUnRegistro, tamaño del registro, cantidadRegistros, archivoDeDatos*): Obtiene del archivo de datos, habitualmente un registro del tamaño que se indica en el segundo parámetro y lo guarda en la variable registro referenciada por el puntero del primer parámetro.

Ejemplo: fread(&regPasajero, sizeof(tRegPasajeros), 1, archPasajeros);

- **fwrite**(*punteroAUnRegistro, tamaño del registro, cantidadRegistros, archivoDeDatos*): graba en un archivo de datos, los datos que se encuentran en la variable registro apuntada por el puntero del primer parámetro, habitualmente es 1 registro del tamaño que se indica en el segundo parámetro.

Ejemplo: fwrite(&regPasajero, sizeof(tRegPasajeros), 1, archPasajeros);

5. Dado un archivo de alumnos de FACENA cuyo diseño de registro es el siguiente: DNI, código de carrera, código de género y edad. Las carreras disponibles son: 18-Lic. en Sistemas de Información (LSI), 19-Lic. en Cs. Biológicas, 20-Agrimensura, 28-Bioquímica, 29-Ing. Electrónica; y los códigos de género: m-mujer, v-varón, x-otro. Se desea obtener un listado de los inscriptos a la carrear LSI, informar cuántos alumnos son y qué porcentaje representan sobre el total de alumnos inscriptos de toda la Facultad.
   **Nota: Tener en cuenta que se deberán grabar los datos antes de poder procesarlos, para lo que será necesario pensar un lote de datos de prueba representativo.*

6. Con los mismos datos del ejercicio 5, determinar cuántas mujeres menores de 25 años estudiarán la carrera Ing. Electrónica. Informar el total y el porcentaje que representan sobre el total de mujeres de todas las carreras. Detectar la mujer con menor edad, mostrar al final DNI y edad.

7. Con los mismos datos del ejercicio 5, contar la cantidad de alumnos inscriptos en las carreras Bioquímica, LSI y Agrimensura y determinar qué carrera tiene la mayor cantidad de alumnos.

8. Con los mismos datos del ejercicio 5, calcular el promedio de edades de los alumnos varones de la LSI. Informar el total de alumnos varones de la LSI y el promedio de edades.

9. Una agencia de turismo registra los siguientes datos de los pasajeros: DNI, Apellido y Nombre, Destino (1-Nacionales, 2-Extranjeros), Precio del viaje, Forma de pago (1-Efectivo, 2-Tarjeta). Se requiere un programa para obtener la siguiente información:
   - Datos de los pasajeros que viajaron a destinos extranjeros y pagaron con tarjeta.
   - Total de importe facturado en concepto de viajes al extranjero y porcentaje de esta recaudación con respecto al total de importes recaudados por la agencia.

10. Un registro de libros de una biblioteca posee los siguientes campos: ISBN (identificación del libro), título de la obra, autor/es, año de edición, editorial. Se requiere un programa para obtener la siguiente información:
    - Los datos de cada libro: ISBN, Título, Autor/es, Año de edición, Antigüedad del libro (Año actual - Año de edición).
    - El total de libros de la biblioteca, el total de libros recientes (hasta 2 años (inclusive) de antigüedad), el total de libros de más de 2 y menos de 5 años de antigüedad y el total de libros con 5 o más años de antigüedad.

11. La tienda virtual NETSHOES registra datos acerca de la venta de sus productos en las distintas regiones de argentina en el archivo netshoes.dat. Para ello dispone de un archivo con los datos de sus clientes y productos vendidos, el cual tiene la siguiente estructura:

Nro. Cliente	Apellido y Nombre	Cod. Región	Categoría producto	Cantidad productos comprados

COD Regiones: 0-Noroeste, 1- Cuyo, 2-Patagonia, 3-Mesopotamia, 4-llanura Pampeana, 5-Sierras Pampeanas, 6-Centro

Categoría de productos: 1-Calzado, 2-Indumentaria, 3-Accesorios.

Se desea escribir un programa, modularizado, que permita obtener la siguiente información:
- Mostrar por pantalla datos de los clientes de la región Patagónica que hayan comprado productos de la categoría Indumentaria.
- Informar por pantalla la cantidad total de productos que se hayan comprado de la categoría "Calzado".

- Generar un vector que contenga la cantidad de clientes por región. Mostrar por pantalla el vector generado.
- Detectar cuál es la región con menor cantidad de clientes. Para ello utilizar el vector generado en el punto anterior. Mostrar por pantalla la cantidad y el nombre de la región a la cual pertenece.