# Estructuras de Datos y Algoritmos

## Práctico de máquina 1 - Año 2022

**Fecha de entrega: Lunes 18 de Abril de 2022 hasta las 8 hs.**

Un importador de electrodomésticos necesita informatizar el listado de artículos existentes en su depósito. Cada artículo tiene un *código* que lo identifica y la información asociada al mismo es: *tipo de artículo, marca, descripción, precio unitario en dólares* y *cantidad existente*.

Se necesita diseñar una aplicación que permita resolver los requerimientos del importador y para ello se cuenta con las siguientes estructuras de datos para almacenar la información mencionada:

**a)** Lista Vinculada Desordenada (LVD).

**b)** Lista Vinculada Ordenada (LVO).

**c)** Lista Secuencial Ordenada (LSO).

**d)** Lista Secuencial Ordenada con Búsqueda Binaria por Bisección(LSOBB).

La aplicación deberá presentar un menú de opciones principal con las opciones: **Administrar Estructuras** y **Comparar Estructuras**

Al elegir la opción **Administrar Estructuras:** se deberá presentar un nuevo menú que permita seleccionar la estructura con la que se desea trabajar y para cada una de ellas un nuevo menú que muestre las opciones para administrarla. Este menú debe presentar por pantalla las siguientes opciones: **ingreso de nuevos artículos, eliminación de artículos existentes** (con confirmación por pantalla) y **consulta de artículos**. Además, debe contener las opciones **Mostrar Estructura** y **Memorización Previa**.
La opción **Memorización Previa** es una rutina que permite guardar en una estructura la información incluida en el archivo de texto *"Articulos.txt"* provisto por la cátedra. Esta rutina debe leer desde el archivo la información correspondiente a un artículo e insertarla en la estructura correspondiente.
La opción **Mostrar Estructura** debe mostrar **sólo las posiciones con elementos**.

**Comparación de Estructuras:** Esta opción debe realizar y mostrar una comparación adecuada de lo que cuesta, en cada una de las estructuras, **realizar ingresos, eliminaciones y consultas de un artículo dado**. En el análisis debe considerar el peor escenario y el comportamiento esperado en cada caso. Una vez finalizada esta operación, deberá realizar un análisis de los resultados obtenidos y sacar una conclusión de los mismos; dicha conclusión deberá quedar plasmada al principio de su programa principal (donde se encuentra el main) como comentario (incluir resultados de la comparación).

Para el cálculo de los costos de ingreso y eliminación: en las listas secuenciales cada corrimiento de nupla tiene costo **1,5 (uno coma cinco)**. En las listas vinculadas se considerará un costo de **0,5 (cero coma cinco)** por cada modificación de punteros. Para las consultas el costo se determinará en celdas consultadas para todas las estructuras (1 punto por cada celda).

Para comparar las estructuras se utilizará una secuencia de operaciones detallada en el archivo de texto *"Operaciones.txt"* provisto por la cátedra. Esta secuencia de operaciones se deberá realizar sobre cada una de las estructuras, asegurando que las mismas **no contengan ningún dato inicialmente**. Una vez finalizada la secuencia de operaciones se mostrarán por pantalla los costos obtenidos para cada estructura. Además una vez terminada la comparación en las estructuras **deben quedar** los datos resultantes de efectuar las operaciones del archivo para ser alcanzados desde la administración de cada una de ellas.

En el archivo *"Operaciones.txt"*, se describen las operaciones a realizar mediante un código de operación, seguido por los datos necesarios para la misma. Así, el código de operación de ingreso de artículo (1) estará seguido por el código

del artículo a ingresar y todos sus datos (nupla completa), el código de eliminación (2) será seguido por el código del artículo a eliminar y todos sus datos (nupla completa), y el código de consulta (3) sólo estará seguido por el código del artículo a consultar.

Consideraciones a tener en cuenta:

- **NO** se utilizaran elementos ficticios de ningún tipo.
- Se esperan 350 artículos.
- Para **Búsqueda por Bisección** se utilizará la siguiente consigna: límite inferior inclusivo, límite superior exclusivo, segmento más grande a la izquierda y testigo a derecha.
- Sobre los datos:
  - El código de artículo es una secuencia de 8 caracteres.
  - El tipo de artículo puede contener un máximo de 20 caracteres.
  - La marca es una secuencia de a lo más 30 caracteres.
  - La descripción puede contener un máximo de 100 caracteres.
  - El valor del artículo es un valor real positivo.
  - La cantidad es un número entero positivo.
- El ingreso de datos **no debe ser sensible a mayúsculas y minúsculas**, esto significa que al buscar un código de artículo deberá ser reconocido independientemente de cómo se ingresen las letras del mismo. Ejemplo: los códigos AAAB534A, aaAb534a, AaaB534a, Aab534A son consideradas iguales.
- El programa deberá desarrollarse en Lenguaje C (estándar), utilizando como herramienta para tal fin **Code::Blocks** (disponible en www.codeblocks.org).

**Nota Importante**: La entrega del práctico se realiza por medio de la página de la materia y se debe enviar el archivo fuente del programa. El nombre del archivo deberá estar conformado de la siguiente manera: ***P****nroP*-***Grupo****nroG* donde *nroP* es reemplazado por el número de práctico que se entrega y *nroG* por el número del grupo al que pertenece el programa. Por ejemplo, el nombre P1-Grupo22.c corresponde al práctico de máquina 1 enviado por el grupo 22. **Los programas cuyos nombres no respeten estas reglas de conformación no serán aceptados**.

## Ejemplo de rutina para Memorización Previa

El código que se presenta a continuación es una guía para programar una rutina que permita leer datos desde un archivo de texto. **Deberá adaptarlo a la situación planteada.**

```
int Memorización_Previa()
{   .... //declaraciones
        FILE *fp;
        if (( fp = fopen ( ''Articulos.txt'' , ''r'') )==NULL)
                return 0;
        else {
                while(!(feof(fp))){
                        fgets (CodArt ,10 , fp);
                        CodArt[8]='\0';
                        fgets (TipoArt ,21 , fp );
                        TipoArt[strlen(TipoArt)-1]='\0';
                        fgets (Marca ,31 , fp );
                        Marca[strlen(Marca)-1]='\0';
                        fgets (Desc ,101 , fp );
                        Desc[strlen(Desc)-1]='\0';
                        fscanf(fp,''%f'',&Valor);
                        fscanf(fp, ''%c'', &barraene );
                        fscanf(fp,''%d'',&Cant);
                        fscanf(fp, ''%c'', &barraene );

            /* Donde CodArt , TipoArt, Marca, Descripción, Valor,
                Cant corresponden a la información guardada en el
                archivo Artículos.txt en la posición corriente y barraene
                es una variable auxiliar para consumir el caracter nueva
                línea después de los valores numéricos */
                        .....
                /* Invocar los procedimientos que correspondan */
                        .....
                        }
                return 1;
        }
        fclose(fp);
}
```

## Ejemplo de rutina para Lectura de Operaciones

El código que se presenta a continuación es una guía para programar una rutina que permita leer datos desde un archivo de texto. **Deberá adaptarlo a la situación planteada.**

```
int Lectura_operaciones()
{   .... //declaraciones
        FILE *fp;
        if (( fp = fopen ( ''Operaciones.txt'' , ''r'') )==NULL)
                return 0;
        else {
                while(!(feof(fp))){

            fscanf(fp, ''%d'', &codigoOperador);
            fscanf(fp, ''%c'', &barraene );
            fscanf(fp, ''%[^\n]'', &aux.CodArt );


            if(codigoOperador==1||codigoOperador==2){
                fscanf(fp,'' %[^\n]'',aux.TipoArt);
                fscanf(fp,'' %[^\n]'',aux.Marca);
                fscanf(fp,'' %[^\n]'',aux.Desc);
                fscanf(fp,''%f'',&aux.Valor);
                fscanf(fp,''%d'',&aux.Cant);

                //llamar al operador correspondiente (Alta o Baja)
                //en todas las estructuras

            }else if(codigoOperador==3){
                //llamar a Evocar en todas las estructuras
            }else{
            //error, código operación no reconocido
            }
            codigoOperador=0;

    }
                return 1;
        }
        fclose(fp);
}
```