# PROGRAMACIÓN I

# GUÍA DE EJERCICIOS NO OBLIGATORIOS

***Objetivo de la guía*:** permitir al alumno usar con fluidez las siguientes estructuras de datos: arreglos, cadenas de caracteres, registros, archivos, estructuras dinámicas.
Lograr además que el alumno pueda :

- Elegír la estrategia de resolución.
- Identificar las funciones como mecanismos para la modularización de soluciones.
- Elegir las estructuras de datos apropiadas.
- Concebir la estructura general de un programa.

## *Unidades*

N° 1 Introducción: funciones - vectores
N° 2 Matrices
N° 3 Cadenas de Caracteres.
N° 4 Estructuras estáticas.
N° 5 Archivos binarios y de texto.
N° 6 Introducción al uso dinámico de memoria. Listas enlazadas.
N° 7 Recursión.

## *Bibliografía*

### *Básica*

Herbert Schildt. "*C Manual de Referencia*". 3a ed. Madrid: Osborne, McGraw Hill, 1995. xviii, 785 p. Serie McGraw Hill de informática. ISBN 84-481-28958.

Deitel & Deitel. "*Como programar en C/C++*". 4a ed. México: Pearson Educacion, 2003. 1320 p. ISBN 968-880-471-1

### *Complementaria*

Joyanes Aguilar e Ignacio Zahonero Martínez. "*Algoritmos y Estructuras de Datos, una perspectiva en C*", Luis Editorial MCGRAW-HILL. ISBN: 84-481-4077

Kernighan, Brian W.; Ritchie, Dennis. "*El lenguaje de programación C*", M. 2a ed. Naucalpan de Juárez: Prentice Hall Hispanoamericana, 1991. 294 p.

## *Autores:*

Mazzitelli, Patricia Silvia
Thompson, Ricardo
Rubín Aymá, Alejo Fedor

# ÍNDICE

# UNIDAD N° 1 – Introducción

**1.** Realizar una función que reciba tres valores enteros positivos y devuelva el menor estricto de los tres. En caso de no existir el menor estricto, devolver ***-1***. No usar operadores lógicos. Realizar un programa que permita probar la función.

**2.** Realizar una función que reciba tres valores enteros positivos y verifique si corresponden a una fecha gregoriana correcta - día, mes, año – En este caso debe devolver verdadero (1). De lo contrario que devuelva falso (0). Realizar un programa que permita probar la función.

**3.** Una persona desea llevar el control de los gastos realizados en subterráneo dentro de un mes. Para ello conoce la cantidad de viajes que realizó en dicho mes. Para esto utiliza las siguientes tarifas, que van disminuyendo según la cantidad de viajes realizados, de acuerdo con esta tabla:

| Cantidad de viajes | Precio por viaje |
|---|---|
| De 1 a 20 viajes | $4,50 |
| 21 a 30 viajes | $3,60 |
| 31 a 40 viajes | $3,15 |
| 41 viajes en adelante | $2,70 |

Realizar una función que calcule el gasto según la cantidad de viajes realizados y un programa que pruebe su funcionamiento.

**4.** Un banco necesita para sus cajeros automáticos un programa que lea las diferentes cantidades de dinero que le fue solicitado e informe la cantidad mínima de billetes que entregó, considerando que existen billetes de $100, $50, $10, $5 y $1. Realizar una función que calcule dichas cantidades y un programa que pruebe su funcionamiento. Se sabe que la solicitud de un importe menor a 0 finaliza la carga de datos.

**5.** Realizar cada una de las siguientes funciones y escribir un programa que permita verificar el funcionamiento de todas ellas:

- Cargar un vector con números de tres dígitos generados al azar.
- Buscar un valor dentro de un vector ordenado y devolver la posición en que se halla dicho valor dentro del vector. En caso que no lo encuentre devolver -1. Usar método de búsqueda binaria.
- Contar la cantidad de veces que un valor se encuentra en un vector.
- Eliminar todas las ocurrencias de un valor de un vector, dejando los valores no eliminados en el mismo orden que estaban y en forma consecutiva. Al finalizar la serie completar con un -1.
- Determinar si el contenido de un vector es capicúa. No usar vectores auxiliares.
- Invertir los elementos de un vector. No usar vectores auxiliares.

**6.** Dado el siguiente programa indicar qué valores se obtienen en pantalla. Justificar sin ejecutar el programa en computadora.

```
void funcion1(int, int);
void funcion2(int*, int*);
int main(void){
     int a=3, b=7;
     printf("a: %d – b: %d\n", a, b);
     funcion1(a, b);
     printf("a: %d – b: %d\n", a, b);
     funcion2(&a, &b);
     printf("a: %d – b: %d\n", a, b);
     return 0;
 }

void funcion1(int a, int b){
     int aux=a;
     a=b;
     b=aux;
}
void funcion2 (int*a, int*b){
```

```
     int aux=*a;
     *a=*b;
     *b=aux;
}
```

**7.** Dado el siguiente programa indicar si es correcto y qué valores se obtendrían en pantalla. Justificar sin ejecutar el programa en computadora.

```
int main(void){
     int a=3, b=7, aux;
     int*p;
     printf("a: %d – b: %d\n", a, b);
     p=&a;
     aux=a;
     a=b;
     b=aux;
     printf("a: %d – b: %d\n", a, b);
     printf("%p - %d\n", p, *p);
     p=p+1;
     printf("%p - %d\n", p, *p);
     return 0;
 }
```

**8.** La siguiente función permite averiguar el día de la semana para una fecha determinada. La fecha se pasa en forma de tres parámetros enteros y la función devuelve 0 para domingo, 1 para lunes, etc.
Escribir un programa para imprimir por pantalla el calendario de un mes completo, correspondiente a un mes y año cualquiera basándose sobre la función suministrada. Considerar que la semana comienza en domingo.

```
int diadelasemana(int dia, int mes, int anio) {
     int siglo, anio2, diasem;
     if (mes<3) {
          mes=mes+10;
          anio--;
     } else
          mes=mes-2;
     siglo=anio/100;
     anio2=anio%100;
     diasem=(((26 * mes-2)/10)+dia+anio2+(anio2/4)+(siglo/4)-(2*siglo))%7;
     if (diasem<0)
          diasem=diasem+7;
     return diasem;
}
```

# UNIDAD N° 2 - Matrices

**1.** Realizar cada una de las siguientes funciones y escribir un programa que permita verificar el funcionamiento de todas ellas

- Cargar números enteros en una matriz de N x N, ingresando los datos desde teclado. ¿En qué cambiaría si la matriz contuviera otro tipo de datos y/o tuviese otras dimensiones?
- Ordenar en forma ascendente cada una de las filas de una matriz.
- Cargar una matriz con números enteros de tres dígitos, generados al azar.
- Intercambiar dos filas dadas.
- Intercambiar dos columnas dadas.
- Intercambiar una fila por una columna dadas.
- Transponer la matriz sobre si misma. (intercambiar cada elemento $a_{ij}$ por $a_{ji}$)
- Calcular el promedio de los elementos de una fila dada.
- Calcular el porcentaje de elementos con valor impar en una columna dada.
- Determinar si la matriz es simétrica con respecto a su diagonal principal.
- Determinar si la matriz es simétrica con respecto a su diagonal secundaria.
- Determinar si todas las columnas de una matriz son palíndromos.

Ejemplos:

| | | | |
|---|---|---|---|
| 1 | 2 | 2 | 1 |
| 2 | 3 | 3 | 4 |
| 2 | 3 | 3 | 4 |
| 1 | 2 | 2 | 1 |

Para esta matriz la función devuelve verdadero

| | | | |
|---|---|---|---|
| 1 | 2 | 2 | 1 |
| 2 | 3 | 3 | 2 |
| 3 | 3 | 3 | 3 |
| 1 | 1 | 1 | 1 |

Para esta matriz la función devuelve falso

Nota: el tamaño de la matriz se deberá declarar mediante la directiva #define.

**2.** Para cada una de las siguientes matrices realizar una función que la genere y escribir un programa que muestre el funcionamiento de todas ellas

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 0 | 3 | 0 | 0 |
| 0 | 0 | 5 | 0 |
| 0 | 0 | 0 | 7 |

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 27 |
| 0 | 0 | 9 | 0 |
| 0 | 3 | 0 | 0 |
| 1 | 0 | 0 | 0 |

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 0 | 2 | 2 | 2 |
| 0 | 0 | 3 | 3 |
| 0 | 0 | 0 | 4 |

| | | | |
|---|---|---|---|
| 4 | 0 | 0 | 0 |
| 3 | 3 | 0 | 0 |
| 2 | 2 | 2 | 0 |
| 1 | 1 | 1 | 1 |

| | | | |
|---|---|---|---|
| 1 | 25 | 23 | 21 |
| 3 | 27 | 33 | 19 |
| 5 | 29 | 31 | 17 |
| 7 | 11 | 13 | 15 |

| | | | |
|---|---|---|---|
| 8 | 8 | 8 | 8 |
| 4 | 4 | 4 | 4 |
| 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 1 |

| | | | |
|---|---|---|---|
| 1 | 3 | 9 | 27 |
| 1 | 3 | 9 | 27 |
| 1 | 3 | 9 | 27 |
| 1 | 3 | 9 | 27 |

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 2 |
| 3 | 0 | 4 | 0 |
| 0 | 5 | 0 | 6 |
| 7 | 0 | 8 | 0 |

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 12 | 13 | 14 | 5 |
| 11 | 16 | 15 | 6 |
| 10 | 9 | 8 | 7 |

| | | | |
|---|---|---|---|
| 1 | 2 | 6 | 7 |
| 3 | 5 | 8 | 13 |
| 4 | 9 | 12 | 14 |
| 10 | 11 | 15 | 16 |

**3.** 2048 es un juego que se desarrolla en una cuadrícula de 4 x 4, donde en cada celda hay una baldosa con un número 2 o 4, generados al azar. Se utilizan las teclas de dirección para mover las baldosas: ***4*** izquierda, ***6*** derecha, ***8*** arriba y ***2*** abajo, las cuales pueden deslizarse por el tablero. Si dos baldosas con el mismo número "colisionan" durante un movimiento, se combinarán en una nueva baldosa, cuyo número será el equivalente a la suma de los números de las dos baldosas originales, es decir, si dos baldosas con el número 4 colisionan, se combinarán en una baldosa con el número 8. Sin embargo, la baldosa resultante no podrá combinarse con otra en una misma jugada. Después de realizar una jugada, indicando la baldosa (fila, columna) y dirección en que se mueve, aparecerá una baldosa nueva en el lugar vacío del tablero, conteniendo un número 2 o un 4. El juego finaliza cuando se obtiene una baldosa con el número 2048. Escribir un programa que permita jugarlo.

# UNIDAD N° 3 – Cadenas de caracteres

**1.** Realizar una función que invierta los caracteres de una cadena, sin utilizar vectores auxiliares y escribir un programa que permita verificar su funcionamiento.

**2.** Realizar una función que determine si una cadena de caracteres es capicúa, sin utilizar vectores auxiliares y escribir un programa que permita verificar su funcionamiento.

**3.** Realizar una función que calcule y devuelva la cantidad de palabras (separadas por uno o más guiones) que tiene una frase y escribir un programa que permita verificar su funcionamiento.

**4.** Realizar una función que reciba una cadena y coloque en mayúscula la primera letra siguiente a un espacio eliminando dichos espacios. Escribir un programa que permita verificar su funcionamiento. Ejemplo:
Cadena original:
*Platero es pequeño, peludo, suave; tan blando por fuera, que se diría todo de algodón, que no lleva huesos. Sólo los espejos de azabache de sus ojos son duros cual dos escarabajos de cristal negro.*
Cadena final:
*PlateroEsPequeño,Peludo,Suave;TanBlandoPorFuera,QueSeDiríaTodoDeAlgodón,QueNoLlevaHuesos.SóloLosEspejosDeAzabacheDeSusOjosSonDurosCualDosEscarabajosDeCristalNegro.*

**5.** Realizar una función que convierta un número entero entre 0 y 999 en un número romano y escribir un programa que permita verificar su funcionamiento. ¿En qué cambiaría la función si el rango de valores fuese diferente?

**6.** Realizar una función que extraiga una sub-cadena, indicando la posición inicial y la cantidad de caracteres y escribir un programa que permita verificar su funcionamiento. Ejemplo, dada la secuencia: "*El número de teléfono es 4356-7890.* " extraer la sub-cadena que comienza en la posición 26, y tiene 9 caracteres, resultando la subcadena "*4356-7890*".

**7.** Realizar una función que inserte una sub-cadena en una cadena a partir de una posición dada, para cada uno de los siguientes casos:
- Sin usar funciones de biblioteca standard
- Usando funciones de biblioteca standard

Escribir un programa que permita verificar el funcionamiento de ambas funciones.

**8.** Realizar una función que elimine una sub-cadena a partir de una posición y un tamaño dado, para cada uno de los siguientes casos:
- Sin usar funciones de biblioteca standard
- Usando funciones de biblioteca standard

Escribir un programa que permita verificar el funcionamiento de ambas funciones.

**9.** Realizar una función que obtenga una subcadena de caracteres con los últimos <n> caracteres de la cadena. La función no hará nada si <n> es menor que cero o mayor que la longitud de la cadena. Escribir un programa que permita verificar su funcionamiento.

**10.** Realizar una función que cuente cuántas veces se encuentra una sub-cadena en otra cadena, los caracteres de la subcadena no necesariamente deben estar en forma consecutiva pero si respetando el orden de los caracteres y escribir un programa que permita verificar su funcionamiento. NO diferenciar mayúsculas de minúsculas. Ejemplo:
Cadena:
*Platero es pequeño, peludo, suave; tan blando por fuera, que se diría todo de algodón, que no lleva huesos. Sólo los espejos de azabache de sus ojos son duros cual dos escarabajos de cristal negro.*
Sub-cadena: *UADE*

*Platero es pe**q**u**e**ño, peludo, su**a**ve; tan blan**d**o por fu**e**ra, q**u**e se diría to**d**o d**e** algodón, q**u**e no lle**v**a huesos. Sólo los espejos **de** azabache de s**u**s ojos son duros cu**a**l **d**os **e**scarabajos de cristal negro.*

Cantidad encontrada: 4

# UNIDAD N° 4 – Estructuras estáticas

**1.** Definir las estructuras para almacenar los siguientes tipos de datos:

a. Fracción (numerador, denominador)
b. Fecha (día, mes, año)
c. Horario (horas, minutos)
d. Datos de una película
   - nombre (cadena de caracteres)
   - duración (horas, minutos)
   - fecha de estreno (día, mes, año)
   - costo de filmación

**2.** Realizar las siguientes funciones utilizando la estructura definida anteriormente según corresponda:

- Ingresar una fracción desde teclado
- Simplificar una fracción
- Ingresar una fecha desde teclado, validando que corresponda a una fecha gregoriana.
- Sumar n días a una fecha
- Ingresar un horario desde teclado, validando que sea correcto
- Calcular la diferencia entre dos horas cualquieras. En el caso que el primer horario sea mayor al segundo, considerar que la primera hora corresponde a la hora del día anterior. La diferencia en horas no supera las 24 horas.

Escribir un programa que permita verificar el buen funcionamiento de cada una de las funciones realizadas.

**3.** Una empresa comercial de entretenimiento que suministra streaming multimedia bajo demanda por Internet, desea contar con un programa que le permita registrar los datos de las películas que pueden solicitar sus clientes. En su biblioteca no existen más de 1000 películas distintas. Realizar un programa ABM utilizando las estructuras definidas en el ejercicio 1. Se deberán desarrollar en el programa las siguientes funciones:

- **Alta**: los parámetros recibidos serán el vector de estructuras y la cantidad de películas disponibles hasta el momento. La tarea de esta función será cargar películas hasta que se llene el vector o hasta que ingresen un nombre vacío en el campo **nombre**.
- **Baja():** los parámetros recibidos serán el vector y la cantidad de películas disponibles hasta el momento. La tarea de esta función será ingresar, en forma parcial o total, un nombre de película y eliminar todos los datos de la misma si el usuario aprueba las películas seleccionadas por el usuario.
- **Mostrar()**: los parámetros recibidos serán el vector y la cantidad de películas disponibles. La tarea de esta función será mostrar los datos de todas las películas disponibles, ordenadas por **nombre de película**.

# UNIDAD N° 5 – Archivos. Archivo binario

**1.** Realizar las siguientes funciones :

- **grabar**: deberá generar un archivo con números reales en el rango 0.00 y 99.99 generados al azar. La cantidad de datos deberá ser generada al azar.
- **mostrar**: deberá mostrar todos los datos del archivo.
- **agregarAcumulado**: deberá agregar al final del archivo el resultado de sumar todos los valores que éste contiene.
- **ordenar**: deberá ordenar los elementos del archivo. Implementarla usando el método optimizado de ordenamiento por burbujeo.
- **buscar**: deberá buscar un valor recibido por parámetro y devolver la posición en que éste se encuentra en el archivo. En caso de no encontrarlo devolver un valor negativo. Implementarla usando el método de búsqueda binaria. La primera posición es la número 0.
- **EliminarL**: deberá eliminar en forma lógica un valor recibido por parámetro, cambiándole de signo.
- **EliminarF**: deberá eliminar en forma física un valor recibido por parámetro.

**2.** Una fábrica de productos perecederos posee un archivo cuyos registros contienen los siguientes datos:
**Descripción del producto:** 32 caracteres
**Código de barras:** 21 caracteres
**Lote:** 5 caracteres
**Costo unitario:** número real
**Stock:** Número entero. Cantidad de unidades almacenadas
**Mes, Año:** Enteros. Mes y año de vencimiento

Un producto puede tener varios lotes según día de envasado que no pueden estar repetidos. Realizar uno o más programas que permitan:

- Generar el archivo.
- Realizar la baja física del archivo de los lotes de aquellos productos que se encuentren vencidos e informarlos. Un lote está vencido cuando la fecha de vencimiento es anterior a la fecha del día. La fecha del día puede ser ingresada por teclado. Listarlos ordenados en forma ascendente por descripción y por vencimiente en forma descendente.
  **Ejemplo:**

  Listado de lotes vencidos a la fecha corriente.

```
LOTE  CODIGO                    DESCRIPCION                Mes/año
00005 12345678901234567890      Dulce de ciruela           11/2010
00001 12345678901234567890      Dulce de ciruela           10/2010
00007 10001                     Dulce de naranja           11/2010
..... ....................      ..............             ........
```

- Mostrar la cantidad en stock de cada producto sin importar a qué lote pertenezca.
  **Ejemplo:**

  Listado de líneas de dulces no vencidos

```
CODIGO                    DESCRIPCION                CANTIDAD
12345678901234567890      Dulce de ciruela           xxxx
10001                     Dulce de naranja           yyyy
.....                     ..............             ....
```

# UNIDAD N° 6 – Archivos. Archivo de texto.

**1.** Realizar una función que elimine todos los comentarios de un programa fuente escrito en lenguaje “C” y escribir un programa que permita verificar su funcionamiento. Recordar que los comentarios en “C” pueden ocupar más de una línea.

**2.** Realizar una función que muestre todas las directivas para el preprocesador de un programa fuente escrito en lenguaje “C”. Escribir un programa que permita verificar su funcionamiento. Considerar sólo las directivas #define e #include. Considerar que puede haber uno o más espacios antes y/o después del ‘#’. Ejemplo:

```
   #  include <stdio.h>
#define MAX 50
```

**3.** Para una simulación se solicita generar al azar la cantidad de lluvia caída para cada uno de los días de un año. Considerar que los meses son de 30 días y que la cantidad máxima de lluvia caida en un mes no es mayor a 999 mm. No utilizar matrices.
Guardar los datos generados en un archivo de texto a razón de una línea por mes, separando cada dato con uno o más espacios.
Leer luego el archivo y mostrar los datos por pantalla en forma matricial.
Mostrar además el promedio de lluvia caída en los meses pares.
Repetir el ejercicio separando los datos por ‘;’ en lugar de espacios.

**4.** Realizar un programa que permita grabar un archivo los datos de lluvia caída durante un año únicamente en los días que haya llovido, con el siguiente formato por cada línea *(<dia>*; <mes>; <lluvia caída en mm>). La cantidad de lluvia caida no supera los 999 mm por día. Mostrar en formato matricial tal que la columna indica el mes y las filas corresponden a los dias del mes. Además mostrar la lluvia promedio caida en los meses pares.

**5.** Realizar un programa que permita verificar el buen funcionamiento de las siguientes funciones:

- **grabarRangoAlturas()** Graba en un archivo el rango de alturas de los alumnos de distintas materias. Cada dato se debe grabar en línea distinta.

  Ejemplo:
  materia
  alturaMinima
  alturaMaxima

- **grabarPromedio()** Graba en un archivo los promedios de las alturas de los alumnos del archivo generado en el paso anterior. Tanto la materia como el promedio se graban en diferentes líneas.
- **mostrarMasAltos()** Muetra las materias cuyos alumnos superan la estatura promedio general. Tomar los datos del segundo archivo.

**6.** Hacer un programa que permita ingresar una lista de apellidos y nombres en formato "Apellido, Nombre" y guarde en el archivo ARMENIA.TXT, los nombres de aquellas personas cuyo apellido termina con la cadena "IAN", en el archivo ITALIA.TXT los terminados en "INI" y en el archivo ESPAÑA.TXT los terminados en “EZ”. Descartar el resto. Ejemplo:

```
Arslanian, Gustavo --> ARMENIA.TXT
Rossini, Giuseppe  --> ITALIA.TXT
Pérez, Juan        --> ESPAÑA.TXT
Smith, John        --> descartar
```

**7.** Un organismo de defensa del consumidor dispone de dos archivos, con distintos formatos, con una lista de productos de precios cuidados con la siguiente información: descripción (63 caracteres), marca (31 caracteres), cantidad (16 caracteres) y categoría (32 caracteres). En el primer archivo sus campos están

separados por '#' y en el segundo archivo cada uno de los campos comienzan en una posición determinada de la línea. En ambos ar chivos, después de la descripción, puede faltar alguno de los datos indicados. Dicha organización requiere de dos programas. Cada uno de ellos deberá leer archivos de diferente formatos, pero ambos deberán obtener los mismos resultados. Para ello deberá:

- Leer el archivo (un programa lee el archivo con formato 1 el otro programa leerá el archivo con formato 2)
- Listar aquellos productos a los que no les falta la información de marca y cantidad.
- Generar un archivo binario con aquellos productos con información faltante de marca y/o cantidad
- Listar los productos con información faltante de marca o cantidad ordenados alfabéticamente por **descripción**, indicando: descripción, marca y cantidad. En caso que la cantidad esté expresada en **l** o **Kg** mostrarla expresada en **cc** o **g** respectivamente.

**Formato 1**

```
ALIMENTO EN POLVO VAINILLA BATIDO#NESTUM#200 GR#ALIMENTO BEBE
AMARGO SERRANO BLANCO PET#SIBSAYA#1500 CC#AMARGOS
AMERICANO#LUSERA#950 CC#APERTIVOS CON ALCOHOL
ANANA LIGHT#CANALE#800 GR#
Arroz Lucchetti LF x Kg#LUCCHETTI#1 KG#
ARVEJAS SECAS REMOJADAS, 300 gr#NOEL#300 GR#CONCERVAS DE VERDURAS
AZUCAR . DOMINO BSA 1 KGM#DOMINO#1 KG#AZUCAR
BEBIDA POWERADE CITRUS PET#PET#500 CC#BEBIDAS
BIZCOCHUELO DE COCO#EXQUISITA##BIZCOCHUELO
Blancaflor 0000#BLANCAFLOR##HARINA
BOCADITO BA¥ADO CHOCOLATE NUCREM#NUCREM#8 UNI#GOLOSINAS
ESENCIA VAINILLA DOS ANCLAS PET 100 CC#DOS ANCLAS#100 CC#ESPECIAS
FANTA POMELO DESCARTABLE PET#FANTA#1,5 LT#
FERNET#LUSERA#935 CC#APERTIVOS CON ALCOHOL
FIDEO SPAGUETTI X 12UNI X 500 GR#LA SALTENA#500 GR#FIDEOS
FLAN DE DULCE DE LECHE#ROYAL#40 GR#FLAN
```

**Formato 2**

```
ALIMENTO EN POLVO VAINILLA BATIDO                    NESTUM                  200
GR          ALIMENTO BEBE
AMARGO SERRANO BLANCO PET                            SIBSAYA                 1500
CC         AMARGOS
AMERICANO                                            LUSERA                  950
CC          APERTIVOS CON ALCOHOL
ANANA LIGHT                                          CANALE                  800
GR
Arroz Lucchetti LF x Kg                              LUCCHETTI               1 KG
ARVEJAS SECAS REMOJADAS, 300 gr                      NOEL                    300
GR          CONCERVAS DE VERDURAS
AZUCAR . DOMINO BSA 1 KGM                            DOMINO                  1 KG
AZUCAR
BEBIDA POWERADE CITRUS PET                           PET                     500
CC          BEBIDAS
BIZCOCHUELO DE COCO                                  EXQUISITA
BIZCOCHUELO
Blancaflor 0000                                      BLANCAFLOR
HARINA
BOCADITO BA¥ADO CHOCOLATE NUCREM                     NUCREM                  8
UNI           GOLOSINAS
ESENCIA VAINILLA DOS ANCLAS PET 100 CC               DOS ANCLAS              100
CC          ESPECIAS
FANTA POMELO DESCARTABLE PET                         FANTA                   1,5
LT
FERNET                                               LUSERA                  935
CC          APERTIVOS CON ALCOHOL
FIDEO SPAGUETTI X 12UNI X 500 GR                     LA SALTENA              500
GR          FIDEOS
FLAN DE DULCE DE LECHE                               ROYAL                   40
GR           FLAN
```

# UNIDAD N° 7 – Estructuras dinámicas de datos: listas

**1.** Una empresa comercial de entretenimiento que proporciona streaming multimedia bajo demanda por Internet, desea contar con un programa que le permita registrar los datos de las películas que pueden solicitar sus clientes. En su biblioteca no existen más de 1000 películas distintas. Realizar un programa ABM utilizando la estructura de puntero a estructura definida en la unidad 4 (estructuras)**,** realizando las siguientes funciones**:**

- **Alta()**: los parámetros recibidos serán el vector y la cantidad de películas disponibles hasta el momento. Esta función deberá permitir a la empresa cargar películas hasta que se llene el vector o hasta que ingresen un nombre vacío en el campo **nombre**.
- **Baja():** los parámetros recibidos serán el vector y la cantidad de películas disponibles hasta el momento. Esta función deberá permitir ingresar, en forma parcial o total, un nombre de película, buscar todas las coincidencias y eliminar todos los datos de las películas encontradas, siempre que el usuario apruebe la eliminación de cada una de las películas seleccionadas.
- **Mostrar()**: los parámetros recibidos serán el vector y la cantidad de películas disponibles. Esta función deberá permitir mostrar los datos de todas las películas disponibles ordenadas por **nombre de película**.

**2.** Realizar un programa que permita verificar el buen funcionamiento de las siguientes funciones de pila que deberán implementarse sobre una lista enlazada.

**apilar**(DADA, e): apila el elemento “e” en la pila DADA.
**desapilar** (DADA): desapila un elemento de la pila DADA.
**tope**(DADA): devuelve la información que se encuentra en el tope de la pila DADA.
**pilaVacia** (DADA): devuelve verdadero si la pila DADA está vacía, y falso en caso contrario.
**inicializarPila**(DADA): inicializa la pila DADA vacía.
Probar las funciones cargando y mostrando una pila de elementos enteros.

**3.** Realizar un programa que permita verificar el buen funcionamiento de las siguientes funciones de cola que deberán implementarse sobre una lista enlazada.

**acolar**(DADA, e): agrega el elemento “e” en la cola DADA.
**desacolar**(DADA): retira el primer elemento de la cola DADA.
**primero**(DADA) : devuelve el primer elemento de la cola DADA.
**colaVacia**(DADA): devuelve verdadero si la cola DADA está vacía, y falso en caso contrario.
**inicializarCola**(DADA): inicializa la cola DADA vacía.
Probar las funciones cargando y mostrando una pila de elementos enteros

**4.** Realizar una función que determine si dos listas poseen los mismos elementos. Las listas pueden tener elementos repetidos y pueden estar desordenados. Nota: no usar malloc ni free en la función.

**5.** Realizar una función que liste la cantidad de veces que aparece cada palabra en una frase, utilizando una lista enlazada para almacenar las palabras y sus repeticiones.

**6.** Realizar una función que invierta el orden de los nodos de una lista, sin destruir y crear nodos nuevos y sin intercambiar la información contenida en los mismos. Nota: no usar malloc ni free en la función. No se permite manipular los datos contenidos en cada nodo.

**7.** Realizar una función que intercale los nodos de una lista en otra lista, sin destruir ni crear nodos nuevos. Las listas pueden tener distinta cantidad de elementos. No se permite manipular los datos contenidos en cada nodo.

**8.** Una empresa comercial de entretenimiento que proporciona streaming multimedia bajo demanda por Internet, desea contar con un programa que le permita registrar los datos de las películas que pueden solicitar sus clientes. Se desconoce la cantidad de películas que hay en su biblioteca. Realizar un programa ABM utilizando la estructura indicada en la unidad 4 (estructuras) realizando las siguientes funciones**:**

- **Alta()**: el parámetro recibido será una lista. Esta función deberá permitir a la empresa cargar películas hasta que ingresen un nombre vacío en el campo **nombre**.
- **Baja():** el parámetro recibido será una lista. Esta función deberá permitir ingresar, en forma parcial o total un nombre de película y eliminar todos los datos de la misma si el usuario aprueba las películas seleccionadas por el usuario.
- **Mostrar()**: el parámetro recibido será una lista. Esta función deberá permitir mostrar los datos de todas las películas disponibles ordenadas por **nombre de película**.

**Nota:** utilizar en el programa una lista enlazada simple.

**9.** Modificar el ejercicio anterior de tal manera que en lugar de usar una lista simple ordenada use una lista doble.

**10.** Normalmente las listas enlazadas deben ser recorridas secuencialmente a fin de buscar datos. Esto puede provocar grandes demoras en listas que contengan una cantidad considerable de nodos. Para paliar este problema existen las *listas indexadas*, que son listas enlazadas simples que se relacionan con otra lista enlazada que hace las veces de índice.
Por ejemplo, una lista ordenada de clientes podría tener una lista índice de 26 elementos, uno para cada letra del alfabeto. Cada nodo de la lista índice cuenta no sólo con un puntero al siguiente nodo de la misma sino también con un puntero adicional que indica a partir de qué nodo de la lista principal aparecen los apellidos con esa letra. Las estructuras de los nodos podrían tener el siguiente aspecto:

```
typedef struct snodoprincipal {
   char nombre[40];
   char domicilio[50];
   double saldo;
   struct snodoprincipal *ant,*sig;
} tnodoprincipal;
typedef struct snodoindex {
   char letra;
   struct snodoprincipal *comienzo;
   struct snodoindex *sig;
} tnodoindex;
```

Escribir un programa que permita insertar nodos en orden en una lista indexada con las características descritas precedentemente, y que pueda realizar búsquedas aprovechando el índice. Implementar también una función para eliminar nodos de la lista principal de acuerdo al nombre del cliente. Utilizar un archivo binario adecuado para cargar la lista y para resguardar su contenido luego de las operaciones realizadas sobre la misma.
**Nota**: El habitual puntero a la cabeza de la lista principal aquí no se utiliza, ya que es reemplazado por un puntero al primer nodo de la lista índice.

# UNIDAD N° 8 - Recursividad

Por cada ejercicio realizar un programa que permita verificar el buen funcionamiento de las funciones solicitadas en cada uno de ellos.

**1.** La función de Ackermann int A(int m, int n) se define de la siguiente forma:
n+1 si m = 0
A(m-1,1) si n = 0
A(m-1,A(m,n-1)) de otro modo
Realizar un cuadro con los valores que adopta la función para valores de m entre 0 y 3 y de n entre 0 y 4.

**2.** Dados dos parámetros de tipo entero, devolver el resultado de calcular el Máximo Común Divisor de dos enteros no negativos, basándose en las siguientes fórmulas matemáticas:
MCD(X,X) = X
MCD(X,Y) = MCD(Y,X)
Si X>Y => MCD(X,Y) = MCD(X-Y,Y).
Utilizando la función MCD anterior calcular el MCD de todos los elementos de una cola dada, sabiendo que:
MCD(X,Y,Z) = MCD(MCD(X,Y), Z).

**3.** Realizar una función recursiva que muestre los N primeros números naturales en forma ascendente.

**4.** Realizar una función recursiva que devuelva la suma de los N primeros números naturales.

**5.** Realizar una función recursiva que devuelva el producto de dos números enteros, por sumas sucesivas.

**6.** Realizar una función recursiva que devuelva el resto de dos números enteros, utilizando restas sucesivas.

**7.** Realizar una función recursiva que devuelva la cantidad de dígitos de un número entero. No utilizar cadenas.

**8.** Para cada uno de los siguientes casos realice funciones que reciban como parámetro al menos una lista y

- muestre los valores en orden inverso .
- devuelva la suma de todos.
- devuelva la cantidad de elementos.
- devuelva el elemento menor.
- devuelva el número de nodo en que se haya un valor dado; en caso de no encontrarlo devolver -1.

**9.** Realizar las siguientes funciones que reciban como parámetro al menos una lista y permitan

- insertar un dato al final de la lista.
- eliminar un dato de la lista.
- invertir una lista sin tener que eliminar ni generar nuevos nodos.