

**Ej. 1:** Desarrollar una función que solicite al usuario el ingreso de un número real y luego muestre el Redondeo al entero más próximo de dicho número, el Piso y el Techo.

Definiciones:

- Redondeo: Si la parte decimal es mayor o igual a 0,5, se redondea hacia arriba. Caso contrario, se redondea hacia abajo.
- Piso: El entero más próximo hacia abajo.
- Techo: El entero más próximo hacia arriba.

Desarrollar el programa principal que solicite el ingresos de un número y luego llama a la función (una por cada uno de los conceptos a mostrar) pasándole el parámetro correspondiente. El resultado será devuelto por la función y mostrado por terminal desde el programa principal.


Ejemplos:

```
Ingrese un numero real: 5.3
Redondeo = 5
Piso = 5
Techo = 6
```

```
Ingrese un numero real: 8
Piso = 8
Techo = 8
Redondeo = 8
```

```
Ingrese un numero real: 8.92
Piso = 8
Techo = 9
Redondeo = 9
```

```
Ingrese un numero real: 8.6
Piso = 9
Techo = 8
Redondeo = 9
```

 <b>UCA</b> <small>PONTIFICIA UNIVERSIDAD CATÓLICA ARGENTINA</small>	<b>Facultad de Ingeniería y Ciencias Agrarias</b>	<b>PROGRAMACIÓN ESTRUCTURADA</b>
		<b>Practica N° 01</b>
	<i>Lenguaje C : Sintaxis básica, Variable, Operadores, Funciones, Estructuras condicionales, Ciclos de repetición, Arreglos Multidimensionales, Archivos de Texto</i>	

**Ej. 2:** Desarrollar la función `operacion` que reciba por parámetro dos números enteros y un carácter: [ + , - , \* , / ] el cual determinará una operación matemática. La función deberá imprimir el resultado de realizar con los números pasados por parámetros la operación correspondiente también pasada por parámetro. Por cada operación a realizar dentro de la función `operacion` se deberá invocar a la función [suma, resta, producto, division] (según corresponda), donde se le pasará por parámetro (los operandos), y cada función deberá retornar el resultado de la operación.

Desarrollar el programa principal que solicite el ingresos de dos números y un carácter que indica la operación, luego llamar a `operacion`, pasándole los parámetros, para que imprima el resultado.

```
Ingrese primer numero: 4
Ingrese segundo numero: 7
Ingrese una operación [+ - * /]: +


4 + 7 = 11
```

**Ej. 3:** Desarrollar un programa en el que se ingrese un texto de hasta 100 caracteres finalizando con "ENTER" (utilizar '\r' para detectar el ingreso de "ENTER"). Luego, el usuario deberá elegir un caracter al cual reemplazar y otro caracter con el cual reemplazará al anterior. El programa finalmente mostrará por pantalla el string modificado.

Para esto el programa principal deberá hacer uso de dos funciones, la función `leer` que guardará en un arreglo el string ingresado y la función `reemplazar` que reemplazará todas las ocurrencias del caracter indicado en el el string

Ejemplo:

```
Ingrese texto: La mar estaba serena
Ingrese carácter a reemplazar: a
Ingrese carácter de reemplazo: e
Resultado: Le mer estebe serene
```

	<b>Facultad de Ingeniería y Ciencias Agrarias</b>	<b>PROGRAMACIÓN ESTRUCTURADA</b>
		<b>Practica N° 01</b>
	<i>Lenguaje C : Sintaxis básica, Variable, Operadores, Funciones, Estructuras condicionales, Ciclos de repetición, Arreglos Multidimensionales, Archivos de Texto</i>	

**Ej. 4:** Desarrollar las funciones **cargarArr**, **imprimirArr** y **invertirArr**.

**cargarArr**: carga desde teclado un arreglo de números enteros pasado por parámetro. El cero determina el fin de los valores útiles del arreglo.

**imprimirArr**: imprime en pantalla el contenido del arreglo de números enteros pasado por parámetro.

**invertirArr**: invierte el arreglo de números enteros pasado por parámetro.

Luego, desarrollar el programa principal declarando dos arreglos (**arr1** y **arr2**) de números enteros de tamaño **T**. Uno de los arreglos deberá ser cargado, luego invertirlo y por último impreso. El otro arreglo deberá ser cargado, luego ordenado y por último impreso. Se debe utilizar las funciones que se solicita desarrollar.


**Ej. 5:** Desarrollar las funciones **cargarArrPNR**, **estaEnArr**, y **esPos**.

**cargarArrPNR**: carga desde teclado un arreglo de números enteros pasado por parámetro. El cero determina el fin de los valores útiles del arreglo. Sólo cargará números positivos y no repetidos (no debe haber números repetidos dentro del arreglo). Llamará a la función **estaEnArr** para determinar si un número está dentro del arreglo. Y llamará a la función **esPos** para determinar si un número es positivo.

**estaEnArr**: recibe por parámetro un arreglo de números enteros, y un número entero. La función retorna 1 si el número se encuentra dentro del arreglo, o retorna 0 si el número no está dentro del arreglo.

**esPos**: Recibe por parámetro un número entero y retorna 1 si el número es mayor e igual que 0 (cero) y retorna 0 caso contrario.

Desarrollar el programa principal que utilizando la función **cargarArrPNR** cargue un arreglo de números enteros de tamaño **T** con números enteros positivos. Utilizar la función **imprimirArr** del ejercicio 4 para imprimir el arreglo en pantalla y así verificar que no hay elementos repetidos en el arreglo.

	Facultad de Ingeniería y Ciencias Agrarias	PROGRAMACIÓN ESTRUCTURADA
		Practica N° 01
	Lenguaje C : Sintaxis básica, Variable, Operadores, Funciones, Estructuras condicionales, Ciclos de repetición, Arreglos Multidimensionales, Archivos de Texto	


- Ej. 6:** Desarrollar las funciones **cargarText**, **imprimirText**, y **normalizar**.
- cargarText:** carga desde teclado un arreglo de char pasado por parámetro, hasta ingresar un “*enter*” (utilizar el carácter ‘\n’ para detectarlo). El texto debe contener ‘\0’ como carácter final.
- imprimirText:** imprime en pantalla el contenido de un arreglo de char pasado por parámetro.
- normalizar:** recibe por parámetro un arreglo de char y debe verificar que:
- La primer letra del texto sea mayúscula.
  - Que haya un sólo espacio entre palabras
  - Y que el texto termine con un punto (“.”).
- En caso de encontrar alguna diferencia deberá ser corregida (*modificar arreglo*) .

Desarrollar el programa principal que invocando a las funciones mencionadas, cargue el arreglo, lo imprima, luego lo normalice y por último lo vuelva a imprimir para verificar el texto.

- Ej. 7:** Desarrollar las funciones **cargarMat**, **imprimirMat**, y **promMat**.
- cargarMat:** carga desde teclado una matriz de int de tamaño **F x C** pasada por parámetro. La matriz se debe cargar completa.
- imprimirMat:** imprime en pantalla el contenido una matriz de int de **F x C** pasada por parámetro.
- promMat:** recibe por parámetro una matriz de int de tamaño **F x C** y retorna un número real que representa el promedio de los valores de la matriz.

Desarrollar el programa principal que invocando a las funciones mencionadas, cargue una matriz, imprima la matriz e imprima el promedio de los valores de la matriz.

- Ej. 8:** Desarrollar la función **transponer**, que recibe por parámetro una matriz de int de tamaño **F x C** (*matriz cuadrada, F=C*) y la modifique por su transpuesta.
- Desarrollar el programa principal que cargue una matriz, luego la muestre en pantalla. A continuación invocar a **transponer** y luego volver a imprimir para verificar su transpuesta. Utilizar **cargarMat**, **imprimirMat**, desarrolladas anteriormente.

 PONTIFICIA UNIVERSIDAD CATÓLICA ARGENTINA	<b>Facultad de Ingeniería y Ciencias Agrarias</b>	<b>PROGRAMACIÓN ESTRUCTURADA</b>
		<b>Practica N° 01</b>
	<i>Lenguaje C : Sintaxis básica, Variable, Operadores, Funciones, Estructuras condicionales, Ciclos de repetición, Arreglos Multidimensionales, Archivos de Texto</i>	

**Ej. 9:** Desarrollar las funciones **cargarMatText**, **imprimirMatText**, y **ordenarMatText**.

**cargarMatText:** carga desde teclado una matriz de char de  $F \times C$  pasada por parámetro. Por cada fila carga un texto el cual debe contener el '\0' al final de los caracteres útiles (concepto de string). Para marcar el fin de las filas útiles deberá agregar en la siguiente fila (a la última útil) un '\0' ( en la columna cero) [*nota1*].

**imprimirMatText:** Imprime la matriz de char de  $F \times C$  pasada por parámetro.

**ordenarMatText:** recibe una matriz de char de  $F \times C$  . La función deberá ordenar la matriz en orden alfabético ascendente. Tener en cuenta que los contenidos de la matriz pueden tener tanto letras mayúsculas como minúsculas.

Desarrollar el programa principal que cargue una matriz, luego la muestre en pantalla. A continuación ordenar la matriz y luego volver a imprimirla para verificar el orden.

*Nota1: (Método). Un '\0' en una fila (y cualquier columna) determinará el fin de los caracteres útiles de dicha fila. Y un '\0' en columna 0 (cero) en una fila cualquiera, determinará el fin de las filas útiles de la matriz.*

*Este método (ad hoc) adoptado no permite identificar en la matriz cadena de caracteres vacías (sólo la última).*

**Ej. 10:** Crear un archivo de texto, "equipos.txt", utilizando un editor de texto plano, que contenga nombres de equipos de fútbol, cada uno en una línea diferente (*es decir una abajo del otro*).

Desarrollar la función **cargarMatTexDeArch** que recibe por parámetro una matriz de  $F \times C$  (vacía) y un texto que indica el nombre del archivo. La función deberá cargar en la matriz pasada por parámetro, los nombres de los equipos que se encuentran guardados en el archivo de texto. Para cargar los datos en la matriz utilizar el mismo método descrito en ejercicio anterior.

Desarrollar el programa principal que invocando a las función mencionada, pasándole el parámetro de la matriz y el nombre del archivo realice la carga. Luego imprimir la matriz en pantalla (*utilizar alguna función realizada anteriormente*).

Ejemplo: (de salida)

```
Milan
Real Madrid
Ajax
Arsenal
Bayern Munich
```

**Ej. 11:** Desarrollar una función cuyo prototipo es **ordenarTexto(char[][N])**, donde  $N = 25$ , que ordene la matriz de char generada en el ejercicio anterior. Como bien se sabe, cada fila contiene un string.

Desarrollar el programa principal que cargue una matriz e imprima en pantalla su orden original, luego invocar a **ordenarTexto** para que ordene la matriz. Luego imprimir la matriz ordenada. Para cargar e imprimir la matriz utilizar las funciones desarrolladas en ejercicios anteriores.

Aclaración: Para realizar comparaciones y copia de cadena de caracteres se sugiere utilizar las funciones **strcmp()** y **strcpy()** de la librería **string.h**

Importante: Una vez terminado el ejercicio, considerar desarrollar usted mismo las funciones **strcmp** y **strcpy**.

**Ej. 12:** Crear el archivo "puntos.txt" cuyo contenido es CSV (del inglés *Comma Separated Values*) utilizando un editor de texto plano. Cada línea del archivo contiene un conjunto de datos correspondientes para cada uno de los equipos del ejercicio del ejercicio anterior. Este conjunto contiene los siguientes datos:

*Puntos, Partidos Jugados, Partidos Ganados, Partidos Empatados, Partidos Perdidos, Goles a favor, Goles en Contra, Diferencia de Goles.*

Desarrollar las funciones:

**cargarMatNumDeArch** recibe por parámetro una matriz de números enteros de  $F \times C$  y el nombre de un archivo de texto. La función debe cargar en la matriz los valores numéricos que se encuentran guardados en el archivo.

**imprimirMatNum** recibe por parámetro una matriz de enteros y dos valores enteros (**fil** y **col**) que indican el tamaño de la fila y de la columna respectivamente. La función deberá imprimir los valores de la matriz con el formato que se muestra en el ejemplo.

En el programa principal realizar la carga y la impresión de la matriz, invocando a las funciones anteriormente descritas.

Ejemplos: (de salida)

10	4	3	1	0	12	3	+9
9	4	3	0	1	11	2	+9
4	4	1	1	2	8	8	0
3	4	0	3	1	5	11	-6
1	4	0	1	3	2	14	-12

- Ej. 13: Desarrollar un programa que a partir de la información que se encuentra en tres archivos de texto la cargue en matrices (tres matrices distintas), la ordene por alguna columna de los datos numéricos, y luego desde el programa imprimir en pantalla como se muestra a en el ejemplo. Utilizar en el programa las funciones que se detallan más abajo.

Ejemplo: (de salida)

Equipo	PTS	PJ	PG	PE	PP	GF	GC	DIF
Milan	10	4	3	1	0	12	3	+9
Real Madrid	9	4	3	0	1	11	2	+9
Ajax	4	4	1	1	2	8	8	0
Arsenal	3	4	0	3	1	5	11	-6
Bayern Munich	1	4	0	1	3	2	14	-12

ARCHIVOS (deberán ser creados con editor de texto):

**cabeceras.txt:** Contiene palabras, una por renglón. Es la información que se encuentra como título en el ejemplo. (Equipo PTS.....etc)

**items.txt:** Contiene palabras, una por renglón. Es la información que se encuentra en la primer columna. En este caso contendrá los equipos (Milan .....etc)

**datos.txt:** Contiene numeros, conjunto de ocho números enteros separados por coma. Cada conjunto está en una línea distinta. Corresponde a los valores numéricos que se pueden observar en el ejemplo.

FUNCIONES:


Reutilizar las funciones de carga e impresión de los ejercicios anteriores de acuerdo al criterio que se detalla al pie de la consigna (*criterio1*).

**ordenarTabla:** recibe cuatro parámetro, la matriz de texto (equipos), la matriz de datos, un valor entero que indica el número de una columna de la matriz de datos y un valor numérico (0 | 1) 0 es ascendente o 1 es descendente. La función deberá ordena ambas matrices de acuerdo al criterio pasado por parámetro.

**imprimirTabla:** recibe por parámetro las tres matrices e imprime el contenido como se muestra en el ejemplo.

**criterio1:** Modificar las funciones para que puedan ser utilizadas por todos los ejercicios que actualmente la están usando y también por este que se está desarrollando (es decir que las funciones o subfunciones sean compatibles).



	<b>Facultad de Ingeniería y Ciencias Agrarias</b>	<b>PROGRAMACIÓN ESTRUCTURADA</b>
		<b>Practica N° 01</b>
	<i>Lenguaje C : Sintaxis básica, Variable, Operadores, Funciones, Estructuras condicionales, Ciclos de repetición, Arreglos Multidimensionales, Archivos de Texto</i>	

- Ej. 14: A Crear un archivo CSV, "temperaturas.txt", utilizando un editor de texto plano, que contenga 12 líneas (una debajo de otra). Cada línea contiene valores numéricos reales separados por comas correspondientes a las temperaturas medias de cada día del mes. Guardando en la primer línea, las correspondientes a los días de enero, en la segunda las correspondientes a los días de febrero y así sucesivamente. Desarrollar una función que cargue en una matriz de 31x12, los valores que se encuentran guardados en el archivo CSV. En un programa de prueba invoque a la función, pasando la matriz como parámetro y luego imprímala por pantalla.
- Además, crear otro archivo CSV, "diasMedidos.txt", que contiene una línea con 12 números separados por coma, por ej: 31,28,31,30,31,30,31,31,30,31,30,31. Cada valor indica la cantidad de días que se han tomado mediciones.

Desarrollar un programa que imprima por pantalla los siguientes datos:

- Valor de temperatura mínima anual, máxima anual y promedio anual.
- Valores promedio mensual de temperatura (una por cada mes).
- Mes más caluroso (en promedio) y valor de dicho promedio.
- Mes más frío (en promedio) y valor de dicho promedio.
- Cantidad total de días medidos

Utilizar funciones, bien definidas por su funcionalidad y reutilizables.