PRÁCTICA I

Ejercicio 1.1

Desarrollar un programa que permita registrar los datos de un campeonato de fútbol para N equipos (valor constante). Por cada equipo debe almacenar: El nombre, la cantidad de partidos ganados, la cantidad de partidos perdidos, la cantidad de partidos empatados, la cantidad de goles a favor y la cantidad de goles en contra. A continuación se mostrará una tabla ordenada (primero por puntos, luego por diferencia de gol y luego por goles a favor) de los equipos como se ejemplifica a continuación:

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

Ejercicio 1.2

En una matriz de 31 x 12 se registran las temperaturas medias de cada día del año. Programar las siguientes funciones:

a. `double promedio_general(double[][12])`, que calcula el promedio de temperatura de todo el año.

b. `void medias_mensuales(double[][12], double[])`, que calcula el promedio de temperatura de cada mes y asigna dicho valor en el índice correspondiente del arreglo pasado como segundo parámetro.

c. `double maxima_absoluta(double[][12])`, que retorna la máxima absoluta del año .

d. Alguien advirtió que no todos los meses del año tienen 31 días. Modifique las funciones definidas anteriormente, de manera que reciban como un parámetro adicional un arreglo de 12 elementos cuyo contenido indica la cantidad de elementos válidos de cada fila / mes (int limites[12] = {31,28,31,30,31,30,31,31,30,31,30,31};), y utilice dicho límite para que las funciones retornen resultados más acertados.

Ejercicio 1.3

Es conocido el algoritmo de ordenamiento de un arreglo de enteros:

```
void ordenar(int a[], int longitud)
{
    int i, j, aux;
    for (i = 0; i < longitud; i++)
    {
        for (j = i + 1; j < longitud; j++)
        {
            if (a[j] < a[i])
            {
                aux  = a[i];
                a[i] = a[j];
                a[j] = aux;
            }
        }
    }
}
```

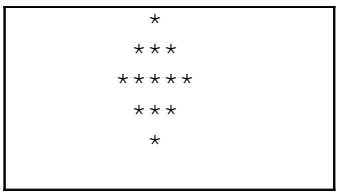
Basándose en esta idea, programe una función cuyo prototipo sea:
`void ordenar_nombres(char[][25]);` que permita ordenar una matriz de char donde cada fila contiene el nombre de una persona. Realice 3 versiones:

a. Sólo hace falta que considere la primer letra.
b. Tenga en cuenta todas las letras de cada nombre para determinar cuál va antes.
c. No distinga mayúsculas de minúsculas.

En todo caso considere que el '\0' indica el fin de cada fila.

Ejercicio 1.4

Programe una función que permita dibujar un "rombo" en una matriz de 5 x 5 char. Cada lugar vacío debe llenarse con un blanco y cada lugar ocupado por el rombo, con un asterisco ('*').



Ejercicio 1.5

Programe una función que reciba como parámetro una matriz cuadrada de N x N (siendo N una constante definida en el programa principal) y realice la transposición de la matriz. Esto es, los valores de cada par (i,j) deben ser intercambiados con los del par (j,i).

Ejercicio 1.6

Efectuar un programa en el que se ingresen notas (enteras) correspondientes a una determinada carrera universitaria. La carrera tendrá una cantidad fija de MATERIAS, una cantidad fija e igual de COMISIONES por cada materia, una cantidad fija e igual de ALUMNOS por cada comisión y una cantidad fija de PARCIALES por cada materia. El programa deberá permitir cargar las notas de un ciclo lectivo completo (todas los parciales de todos los alumnos de todas las comisiones de todas las materias), utilizando alguna de las siguientes opciones:

- Carga Masiva: se solicita al usuario que cargue todas las notas indicándole el número de Materia, Comisión, Alumno y Parcial. Por ejemplo:

  Ingrese nota para Materia 1 – Comisión 1 – Alumno 1 – Parcial 1:
  Ingrese nota para Materia 1 – Comisión 1 – Alumno 1 – Parcial 2:
  Ingrese nota para Materia 1 – Comisión 1 – Alumno 2 – Parcial 1:
  ...
  Ingrese nota para Materia 3 – Comisión 2 – Alumno 20 – Parcial 2:

- Carga Selectiva: se solicitan al usuario todos los datos:

  Ingrese Materia – Comisión – Alumno – Parcial – Nota – Ingrese "-1" para finalizar:
  …

A continuación presentar un menú con las siguientes opciones:

1- Ver promedio de una materia (*solicitará el código de la materia*)
2- Ver promedio de un alumno (*solicitará la comisión y el nro. del alumno*)
3- Ver promedio de una comisión (*solicitará la comisión*)
4- Ver promedio de un parcial (*solicitará la materia, la comisión y el parcial*)
5- Volver a cargar los datos (*se sobreescribirán los datos existentes*)
6- Salir.

Ejercicio 1.7

Realizar una función que muestre la representación binaria de una variable de tipo char, utilizando potencias de 2 para obtener cada bit. Aplicarla en un programa de ejemplo.

Ejercicio 1.8

Realizar una función que muestre la representación binaria de una variable de tipo char, utilizando los operadores SHIFT (desplazamiento a izquierda y a derecha). Aplicarla en un programa de ejemplo.

Ejercicio 1.9

Realizar una función que efectúe el SHIFT A DERECHA completando con 0 desde la izquierda. Aplicarla en un programa de ejemplo.

Ejercicio 1.10

Realizar una función que cuente la cantidad de bits que están en 1 en una variable de tipo unsigned int.

Ejercicio 1.11

Realizar una función que reciba como parámetros 4 unsigned char y que coloque cada uno de ellos en una variable de tipo unsigned int, que devolvera en su valor de retorno.

Ejercicio 1.12

Realizar una función que reciba como parámetro una letra y que la cambie a mayúscula (si está en minúscula) o que la cambie a minúscula (si está en mayúscula).
Dicha cambio consiste en cambiar un solo bit del char. Determinar cuál es ese bit, qué ***operador*** hay que aplicar sobre el char original y con qué ***máscara***. La función tendrá una única línea:

```
return letra OPERADOR MASCARA;
```

Ejercicios Adicionales (no se evaluarán en parcialitos. Usarlos para practicar para parciales y finales)

LABERINTO

Desarrollar un programa que contenga un laberinto dibujado en texto (de 20 x 20 caracteres). A continuación, desarrollar una función que, dada la posición de entrada (que se debe enviar como parámetro a la función), sea capaz de recorrer el laberinto hasta encontrar la salida (se considera salida cuando se llega a alguno de los bordes del laberinto – que no sea la entrada-). Una forma sencilla de encontrar la salida de un laberinto es doblar a la derecha siempre que se pueda. Tener en cuenta que cuando uno camina hacia el sur, a su derecha tiene el oeste pero cuando camina hacia el norte, a la derecha está el este. Cuando se llega a un punto muerto (es decir que no se puede seguir avanzando), el mecanismo de "doblar a la derecha" permite volver sobre los mismos pasos. El programa deberá imprimir el recorrido hecho desde la entrada hasta la salida (por ejemplo, marcándolo con puntos).

TA-TE-TI

Realizar un juego de TA-TE-TI para dos jugadores. Cada jugador ingresará en qué casillero quiere jugar y, a continuación, se mostrará por pantalla el dibujo del tablero. Finalizado el juego se debe informar por pantalla qué jugador ganó o si ninguno ganó.

BUSCAMINAS

Programar el juego "Buscaminas" según las siguientes reglas:

- El tablero será de 10 x 10 casilleros.
- Habrá 5 minas ubicadas en posiciones al azar.
- Los casilleros que no contengan minas, contendrán la cantidad de minas que hay en los casilleros adyacentes (los que están pegados arriba, abajo, a los costados y en diagonal) al mismo.

El programa deberá generar un tablero con todos los datos y luego comenzar el juego solicitando al jugador que ingrese las coordenadas del casillero que desee "explorar". Luego de cada elección, se mostrará el estado actual del tablero mostrando los números de aquellos casilleros descubiertos o bien un '#' en los casilleros no explorados. Luego de cada elección se verificará lo siguiente:

- En caso de que se elija un casillero que no contiene minas, se deberá mostrar el número de minas que hay alrededor.
- En caso de que se elija un casillero que contenga el 0, se revelarán todos los casilleros adyacentes a éste.
- Cuando se elija un casillero que contenga una mina, el juego finalizará y el jugador habrá perdido.
- Cuando se hayan elegido todos los casilleros y sólo queden sin descubrir los que contengan minas, el jugador habrá ganado.

Al finalizar el juego se mostrará el tablero completo con los números e indicando con '*' las ubicaciones de las bombas, y con guiones los lugares vacíos.

SOPA DE LETRAS

Efectuar un programa que genere una sopa de letras al azar de 20 x 20 letras que se mostrará por pantalla. Luego, se ingresará por teclado una palabra y el programa deberá buscarla en forma horizontal, vertical y diagonal (todos en ambas direcciones). Si la palabra es encontrada, volverá a imprimir el tablero con dicha palabra en mayúscula. No se permitirá ingresar 2 veces la misma palabra. Cuando el usuario ingrese una palabra en blanco, finalizará el juego y se imprimirá por pantalla cuántas palabras acertó.

4 EN LÍNEA

Programar un juego de 4 en línea para 2 jugadores humanos. Se dispone de un tablero de 10 x 10 casilleros en los cuales se ubicarán las fichas de los 2 jugadores. Para identificarlas, un jugador utilizará fichas “X” y el otro fichas “O”. En turnos alternados cada uno elegirá en qué columna ubicará su ficha, la cual deberá posicionarse en la última fila que encuentre vacía de dicha columna. El juego finalizará inmediatamente cuando un jugador logre ubicar 4 de sus fichas seguidas en forma horizontal, vertical o diagonal. Considere que puede haber empate en caso de que todos los casilleros del tablero están completos y ninguno de los jugadores haya logrado ubicar 4 fichas en línea.