## Trabajo Práctico N° 5

**TEMA:** Arreglos

"Un arreglo es una colección de variables ordenadas e indexadas, todas de idéntico tipo que se referencian usando un nombre común".

### Funciones de biblioteca del archivo de cabecera <string.h>

Las variables cad1 y cad2 son arreglos de caracteres.

| | | |
|---|---|---|
| ***int strlen***(cad1) | Retorna la longitud de cad1 | |
| ***int strcmp***(cad1, cad2) | Compara cad1 con cad2, carácter a carácter | SI (cad1[i] < cad2 [i] ) ENTONCES<br>Retorna entero< 0<br>SI (cad1[i] = cad2 [i] ) ENTONCES<br>Retorna 0<br>SI (cad1[i] > cad2 [i] ) ENTONCES<br>Retorna entero> 0 |
| **int strncmp(cad1,cad2,n)** | Compara hasta n caracteres de la cad1 con cad2 sin diferenciar mayúsculas de minúscula. | SI (cad1[i] < cad2 [i] ) ENTONCES<br>Retorna entero < 0<br>SI (cad1[i] = cad2 [i] ) ENTONCES<br>Retorna 0<br>SI (cad1[i] > cad2 [i] ) ENTONCES<br>Retorna entero > 0 |
| ***char *strcpy(cad1, cad2)*** | Copia cad2 a cad1, incluyendo el terminador "\0". Retorna cad1 | |
| ***char *strncpy(cad1*, cad2,n)** | Copia hasta n caracteres de la cad2 a cad1. Retorna cad1. Rellena con "\0" si cad2 tiene menos de n caracteres | |
| ***char *strcat(cad1*, cad2)** | Concatena la cad2 al final de cad1. Retorna cad1 | |

### Números aleatorios:

```
#include <stdlib.h>      // Incluye las funciones rand() y srand()
#include <time.h>        // Incluye las funciones time(). Time(null) devuelve el tiempo actual.
srand(time(NULL));       // Inicializar la semilla con la hora actual. Genera distintos aleatorios
                         en cada ejecución.
rand()                   // Genera aleatorio de 0 a 32767
rand() % 101;            // Genera aleatorio de 0 a 100
inicio + rand() % (fin - inicio + 1);     // Fórmula general para un rango [inicio,fin]
```

Resuelva el punto 1 con detenimiento, en el mismo se busca de forma sencilla aprender a manejar el recorrido en un arreglo.

## 1. Calentando Motores

Declare un arreglo de números enteros con un tamaño fijo de 15 elementos. A continuación, recorra el arreglo y llénelo con números aleatorios dentro del rango [10, 350].

a) Muestre todos los números almacenados en el arreglo.
b) Recorra el arreglo para encontrar y mostrar el número más grande.
c) Calcule y muestre el promedio de todos los números almacenados en las posiciones impares.
d) Reemplace cada número mayor a 300 por un número que ingrese el usuario por teclado.
e) Incremente en uno todos los números impares del arreglo.
f) Modifique el programa para que el número de valores aleatorios no sea fijo sino que se lea como entrada y sea como máximo 15. ¿Qué ocurre si indicamos más de 15?

Para la implementación, declare y defina las siguientes funciones:

```
void cargarArreglo(int arreglo[], int tama, int num1, int num2)
void mostrarArreglo(int arreglo[], int tama)
int buscarMayor(int arreglo[], int tama)
void mostrarPromedio(int arreglo[], int tama)
```

**Nota:** num1 y num2 representan los límites del rango en el que se deben generar los números aleatorios para llenar el arreglo

## 2. Verificando contraseñas

**Retomando el tp anterior,** escriba un programa en C que permita al usuario crear una contraseña y verifique si cumple con los siguientes requisitos:

- Debe tener al menos 8 caracteres.
- Debe contener al menos una letra mayúscula.
- Debe contener al menos una letra minúscula.
- Debe incluir al menos un número.

El usuario deberá ingresar la contraseña dos veces para confirmar que ambas coinciden. El programa debe mostrar un mensaje indicando si la contraseña es válida o especificar cuáles de los requisitos no se han cumplido. El proceso debe repetirse hasta que se ingrese una contraseña válida.

## 3. Concatenando cadenas

a. Dado el nombre y el apellido de una persona por separado, únalos en una sola cadena con el formato: 'Apellido, Nombre'. No utilice la función strcat().

b. A partir del nombre completo de una persona (al menos un apellido y dos nombres), compactar la frase en el mismo arreglo; es decir, eliminar los espacios en blanco. Además, agregue un punto al final.
Por Ejemplo:
Entrada: “Héctor Eduardo Reglero Montaner”
Salida: “HéctorEduardoRegleroMontaner.”

## 4. Suma de Matrices

Escriba un programa en C que permita al usuario ingresar dos matrices cuadradas y calcular su suma. Las tres matrices deben mostrarse en pantalla.
Desarrolle funciones para:

- Cargar datos a una matriz
- Mostrar una matriz
- Sumar dos matrices.

Una matriz A es cuadrada si el número de filas es igual al número de columnas.
Siendo A y B matrices de igual dimensión, la suma de A y B se obtiene sumando los términos situados en el mismo lugar.

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \qquad A + B = \begin{pmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}$$

## 5. Adivinando la Palabra

Programar un prototipo para un juego llamado Adivinando la Palabra. El juego funciona así:

- Primero se debe ingresar una palabra. Luego, basándose en la longitud de la palabra ingresada, el programa muestra guiones incógnitos que representan cada carácter de la palabra.
- Un usuario tiene que adivinar la palabra ingresando alguna letra.
    - Si la letra está en la palabra, el programa actualiza los guiones incógnitos, revelando las letras adivinadas.
    - Si el jugador introduce una letra que no está en la palabra, se le resta un intento. El jugador tiene un límite de 3 intentos.

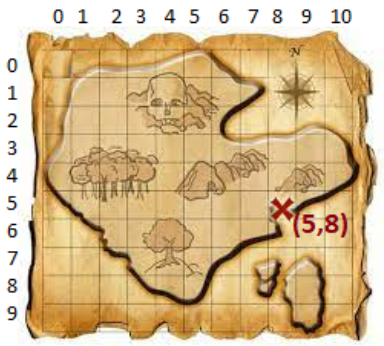
El juego continúa hasta que el jugador adivine la palabra o agote sus 3 intentos. Si el jugador adivina la palabra, el programa muestra un mensaje de felicitación. Si el jugador agota sus 3 intentos sin adivinar la palabra, el programa muestra un mensaje de derrota y revela la palabra correcta.

Ejemplo:

1- Jugador 1: Ingresa palabra: **programacion**
2- Muestra en pantalla: - - - - - - - - - - - -
3- Jugador 2: Ingresa una letra: **a**
4- Muestra en pantalla: - - - - - a - a - - - - Intentos: **3**
5- Jugador 2: Ingresa una letra: **e** (No hay letra e en la palabra)
6- Muestra en pantalla: - - - - - a - a - - - - Intentos: **2**

## 6. Cazador de tesoros

Programe un juego con las siguientes reglas:
El juego se desarrolla en un mapa cuadriculado, representado por una matriz, donde se oculta un tesoro en una ubicación aleatoria.
El objetivo del jugador es **adivinar las coordenadas (X, Y)** en dónde se encuentra el tesoro en el mapa y para ello **cuenta con cinco intentos** para encontrarlo.
Cada celda del mapa es inicialmente desconocida. El jugador puede explorar una celda ingresando coordenadas y el programa marcará esa celda como explorada.
El programa calculará y mostrará la distancia total desde las coordenadas ingresadas por el jugador hasta la ubicación del tesoro después de cada intento.
Si el jugador adivina las coordenadas del tesoro, el programa mostrará un mensaje de felicitación y el número de intentos que le tomó al jugador para encontrar el tesoro.
Si el jugador agota sus intentos sin encontrar el tesoro, el programa mostrará un mensaje de derrota y revelará las coordenadas del tesoro.




**Implementación**

El programa utiliza una matriz bidimensional para representar el mapa y las ubicaciones que serán exploradas por el jugador.
Las celdas no exploradas se representan con " **.** ", las exploradas con "**O**". La ubicación del tesoro se mantiene oculta hasta finalizar y se mostrará con "**X**".
Luego de cada selección de coordenadas debe mostrar nuevamente el mapa con los puntos seleccionados.

Ejemplo: Si el tesoro se encuentra en la posición (5,8) y el jugador supone que está en la posición (3,9), el programa debe indicar que se encuentra a 3 celdas de distancia y descontar un intento.

Realice al menos una función para inicializar el mapa y una función para mostrarlo.