

Ejercicio calificable 1.2

⭐ Ejercicio 1 (El Arquitecto de Marcos)



Imagina que eres un **constructor de marcos** y quieres diseñar el marco de los cuadros para tus amigos artistas ("Leonardo da Vinci", "Sandro Botticelli", "Diego Velázquez", "Francisco de Goya", "Claude Monet", etc).

Tu misión es crear una **una función (que pudiera llamar a otras funciones)** que lea las medidas de un **fichero (marco.txt)** y, con la ayuda de los algoritmos, transforme esos números en marcos dibujados con caracteres.

Así podrás calcular las proporciones del marco, visualizarlo en pantalla y también guardar el resultado en un fichero de salida.

¡Como un auténtico arquitecto del arte! 🖌️ ✨ .

Dispondrás de:

- **marco.txt** → Contiene las dimensiones de los marcos (altura, largo, ancho).
- **marco.php** → Programa en PHP que lee el fichero, genera el marco y lo muestra en pantalla.
- **salida.txt** → Archivo de salida con los marcos generados.

🚀 ¿Qué debes de hacer?

1. Lee un fichero **marco.txt** que contiene la siguiente contenido de ejemplo:

```
4 7 1
```

1. Ejecutar el script en PHP:

```
php marco.php
```

1. El programa:

- Leerá cada línea de **marco.txt**.
 - Dibujará en pantalla el marco.
 - Guardará todos los marcos en el fichero **salida.txt**.
-

Ejemplos de entrada y salida

Ejemplo 1

Entrada (marco.txt):

```
4 7 1
```

Salida en pantalla y en salida.txt:

```
*****
*      *
*      *
*      *
*****
```

Ejemplo 2

Entrada (marco.txt):

```
3 10 2
```

Salida:

```
*****
*****
**      **
**      **
*****
```

Ejemplo 3

Entrada (`marco.txt`):

```
5 5 1
```

Salida:

```
*****  
* *  
* *  
* *  
* *  
* *  
*****
```

⭐ Ejercicio 2 (El Taller de los Pintores mejorado)

Mejora el diseño del algoritmo del **ejercicio 1**.

Cada pintor te dará las medidas de su marco en un fichero, y tu misión será crear un programa que:

1. **Lea las medidas y el nombre del pintor** desde un fichero de texto.
2. **Genere el marco** usando caracteres (*) para el marco y espacios para el interior).
3. **Muestre el resultado en pantalla**.
4. **Guarde cada marco en un fichero independiente** con el nombre del pintor (`Leonardo.txt`, `Picasso.txt`, `Monet.txt`, etc.).

¡Serás el arquitecto de los grandes pintores! 🖌️🌟

📁 Archivos necesarios

- `pintores.txt` → Fichero de entrada con las dimensiones y el nombre del pintor.
- `pintores.php` → Programa en PHP que lee el fichero y genera los marcos.
- `Leonardo.txt`, `Picasso.txt`, `Monet.txt`, etc. → Archivos de salida generados automáticamente.

🚀 Instrucciones

1. Crear el fichero `pintores.txt` con este contenido:

```
4 7 1 Leonardo  
3 10 2 Picasso  
5 5 1 Monet
```

2. Ejecutar el script en PHP:

```
php pintores.php
```

3. El programa:

- Leerá cada línea de **pintores.txt**.
 - Dibujará el marco en ASCII en pantalla.
 - Guardará cada marco en un fichero independiente con el nombre del pintor.
 -
-

Ejemplos de entrada y salida

Ejemplo 1 – Leonardo

Entrada (**pintores.txt**):

```
4 7 1 Leonardo
```

Salida en pantalla y en **Leonardo.txt**:

```
*****  
*      *  
*      *  
*      *  
*****
```

Ejemplo 2 – Picasso

Entrada (**pintores.txt**):

```
3 10 2 Picasso
```

Salida en pantalla y en **Picasso.txt**:

```
*****  
*****  
**      **  
**      **  
*****  
*****
```

Ejemplo 3 – Monet

Entrada ([pintores.txt](#)):

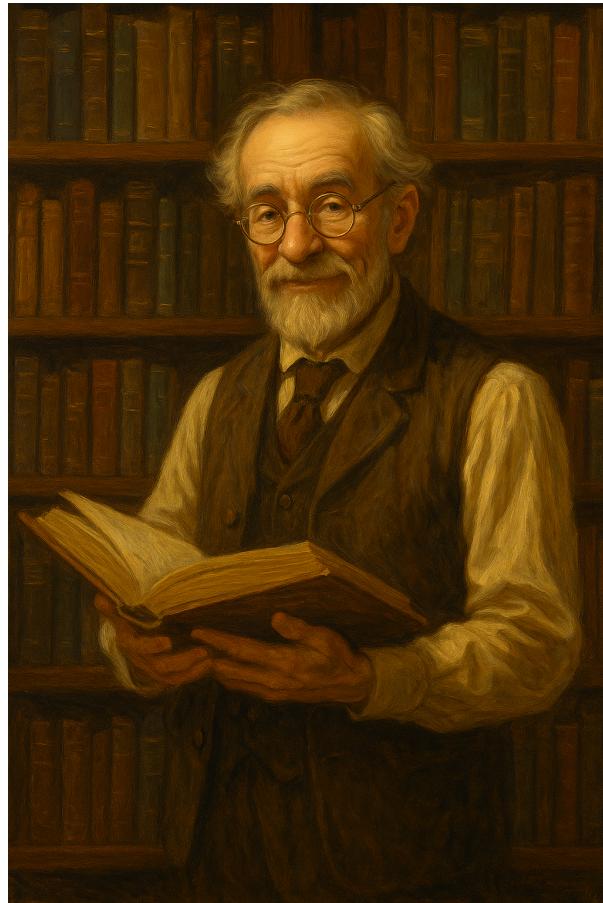
```
5 5 1 Monet
```

Salida en pantalla y en [Monet.txt](#):

```
*****  
*      *  
*      *  
*      *  
*      *  
*      *  
*****
```

NOTA: Reutiliza y mejora el código anterior si fuera necesario.

⭐ Ejercicio 3 (Bibliotecario de Palabras 📚)



Imagina que eres un **bibliotecario digital** y acabas de recibir un fichero lleno de palabras. Tu misión es organizar y analizar esas palabras para que tus amigos escritores puedan usarlas en sus obras.

Archivos necesarios

- **palabras.txt** → Fichero de entrada que contiene una lista de palabras (una por línea).
 - **palabras.php** → Programa en PHP que lee el fichero y realiza las operaciones.
 - **resultados.txt** → Archivo de salida donde se guardan los resultados del análisis.
-

Instrucciones

1. Crea un fichero **palabras.txt** con el siguiente contenido de ejemplo:

```
sol  
luna  
estrella  
cielo  
mar  
luna  
sol  
nube  
cielo
```

2. Crea el script **palabras.php** que:

- Lea todas las palabras del fichero **palabras.txt**.
- Muestre en pantalla y guarde en **resultados.txt** las siguientes operaciones:
 1. Número total de palabras.
 2. Número de palabras únicas.
 3. La palabra más larga.
 4. La palabra más corta.
 5. Listado de palabras ordenadas alfabéticamente.
 6. Conteo de repeticiones de cada palabra.

Ejemplo de salida esperada (**resultados.txt**):

```
total 9  
únicas 5  
+larga estrella  
+corta: mar  
  
orden:  
cielo  
cielo  
estrella  
luna  
luna  
mar  
nube  
sol
```

```
sol
```

```
numero palabras:  
cielo: 2  
estrella: 1  
luna: 2  
mar: 1  
nube: 1  
sol: 2
```

Rúbrica de Evaluación — Ejercicio calificable 1.2

Ejercicio	No Funciona/Ejecuta 0 puntos	Funciona Correctamente (Aprobado mínimo)	Se crea una función y se restringen los tipos de datos	Se encuentra documentada la función	El algoritmo es Óptimo
Ejercicio - 1 (4 pts)	0 pts	2.0 pts	2.5 pts	3.0 pts	4.0 pts
Ejercicio - 2 (1,5 pts)	0 pts	0.75 pts	1.0 pts	1.25 pts	1.5 pts
Ejercicio - 3 (4,5 pts)	0 pts	2.25 pts	3.0 pts	3.75 pts	4.5 pts

Explicación

- **No Funciona/Ejecuta (0 puntos)** → El código no se ejecuta o no genera salida válida.
- **Funciona Correctamente (50% de la nota del ejercicio)** → El algoritmo funciona con los ejemplos básicos y genera la salida esperada.
- **Se crea una función y se restringen los tipos de datos (+12,5%)** → Uso de funciones bien definidas con parámetros/retornos tipados (`declare(strict_types=1)`).
- **Se encuentra documentada la función (+12,5%)** → Cada función tiene phpDoc y comentarios explicativos.
- **El algoritmo es Óptimo (100%)** → Código eficiente, modular, validaciones de entrada, manejo de errores y buenas prácticas.

Recursos

- Documentación **oficial Php**.
- Ejercicios resueltos por el alumnado.