



TRABAJO PRÁCTICO OBLIGATORIO:

Cumplir con las siguientes condiciones es parte de la evaluación del Trabajo Final:

En las clases previas a la defensa del trabajo práctico deben mostrar avances de la resolución y realizar consultas a los docentes. El día de la defensa, el programa debe funcionar: no debe arrojar errores de sintaxis/ejecución.

El trabajo final es grupal: Según las indicaciones de la cátedra.

Luego de descargar todos los archivos del trabajo práctico final publicado en PEDCO, el grupo deberá reunirse y **seguir el instructivo "TPFinal_GITHUB.pdf"**. El instructivo explica que cada integrante del grupo deberá crear una cuenta en GITHUB (<https://github.com/>). Uno de los integrantes del grupo deberá crear un repositorio público llamado "wordix" y agregar como colaboradores al resto de los integrantes del grupo. Cada integrante deberá tener instalado GIT en su pc (<https://git-scm.com>).

Para cumplimentar el trabajo final, cada grupo deberá entregar tres (3) archivos:

- i) Un archivo "**DisenioEstructuras<Apellidos>.pdf**" que contenga apellidos, nombres, legajos, mails, carrera de los integrantes del grupo, nombre usuario github, la url del repositorio wordix de github. Este archivo deberá realizar la representación de las estructuras de datos utilizadas y los datos almacenados en las estructuras. (*<Apellidos> debe ser reemplazado con los apellidos de los integrantes*).
- ii) El archivo "**wordix.php**"
- iii) El archivo "**programa<Apellidos>.php**". (*<Apellidos> debe ser reemplazado con los apellidos de los integrantes*).

Uno de los integrantes del grupo será el encargado de subir la resolución a una tarea en el **curso de PEDCO antes** de la defensa. Habrá otra tarea post defensa, sólo si el docente solicita modificaciones y re-entregar.



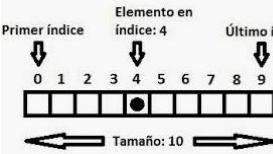





Además los archivos deben ser subidos al repositorio público "wordix" de **GITHUB**. Creado por el grupo utilizando el instructivo "TPFinal_GITHUB.pdf".

Consideraciones:

- Debe representar las estructuras de datos indicando índices/claves y valores. Esto le ayudará a identificar cómo crear, acceder y recorrer las estructuras, además de utilizar dicha representación para consultar a los docentes. Puede utilizar una planilla de cálculo o Diagrams.net en drive para crearla.
- Serán valoradas las soluciones sencillas, el código prolijo y legible.
- El trabajo será desarrollado íntegramente en PHP.
- Recuerde que es un Trabajo Final que será evaluado, utilice un vocabulario apropiado tanto en el código fuente como en los mensajes que mostrará en pantalla.
- Antes de comenzar a codificar, leer el enunciado completamente con el fin de hacer un buen análisis del problema a resolver: datos de entrada, objetivos, estructuras de datos que debe utilizar (aprovechar esta instancia para completar el archivo **DisenioEstructuras<Apellidos>.pdf**), posibles funciones a implementar, etc.
- Cada **función** implementada debe ser debidamente documentada (como fue visto en los apuntes de teoría utilizando `/** */`, especificando descripción de la función, tipo y nombre de los parámetros, tipo del retorno). Los nombres de las funciones y de los parámetros formales deben ser acorde a la funcionalidad que resuelven. Utilizar notación lowerCamelCase.

- El día de la defensa debe explicar la ejecución del programa y el código fuente, además de responder las preguntas que realizará el docente.

Resumen:

 <p>Leer atentamente todo el enunciado del trabajo práctico final.</p> <p>Para identificar todo lo que hay que hacer y establecer los pasos para resolverlo.</p>	 <p>Descargar los archivos de pedco: Enunciados y archivos php.</p> <p>Renombrar los archivos como indica el enunciado reemplazando <apellidos>.</p> <p>Juntarse en grupo para seguir la guía GIT/GITHUB:</p> <p>TPFinal_GITHUB.pdf</p>	 <p>Identifique las estructuras de datos diseñadas para resolver el problema, qué tipos de datos almacenan.</p>  <p>Represéntelas en papel según lo visto en la teoría, ¡Le ayudará a entender para qué se usan y cómo recorrerlas y accederlas!</p> <p>Genere el archivo “diseñoEstructuras”</p>	 <p>Relacionar el enunciado con lo programado en los archivos wordix.php y programaApellidos.php, para determinar qué está programado y qué falta. Esto le ayudará a reusar funciones y tener que codificar menos.</p> 	 <p>¡Manos a la obra! Comenzar a completar el código del archivo programaApellidos.php</p> 
---	---	---	---	--

ENUNCIADO DEL PROBLEMA A RESOLVER:



¡ JUGUEMOS WORDIX!

Link para instalar en su celular: https://play.google.com/store/apps/details?id=word.daily.puzzle&hl=es_AR&gl=US

Wordix es un Juego de palabras muy adictivo en el que tendrás que adivinar palabras. Tu tarea consiste en resolver una palabra de cinco letras en seis intentos. ¡Ejercita tu mente intentando adivinar todas las palabras!

En Wordix tendrás seis oportunidades de adivinar una palabra de cinco letras al azar. Si adivinas una letra y esta está en el lugar correcto, se pinta de **verde**. Si adivinas una letra pero está en lugar incorrecto, se pinta de **amarillo**. Si no existe esa letra, se pinta de **rojo** y ya se sabe que no debería volver a usarse.

Una partida de Wordix (adivinar una palabra) es jugada por un jugador, pero del presente programa será llevar una estadística de diferentes jugadores.

Puntaje:

Si el jugador no adivina la palabra en 6 intentos, su puntaje será 0.

Si el jugador adivina la palabra en 6 o menos intentos, su puntaje se calcula:

Si adivina en el primer intento gana 6 puntos

Si adivina en el segundo intento gana 5 puntos

Si adivina en el tercer intento gana 4 puntos

Etc.

Además cada letra del abecedario tendrá un puntaje: las vocales 1 pto, las consonantes anteriores a la "M" (inclusive) son 2 ptos, y las consonantes posteriores a la "M" son 3 puntos. Por lo tanto, a los puntos del intento, se le sumará la el valor de cada letra de la palabra Wordix adivinada.

EXPLICACIÓN 1 (se enfoca en el punto de vista del Usuario)

En la materia de **Introducción a la Programación** especificaremos en lenguaje PHP un Programa con un menú de opciones, que permita jugar al WORDIX y mostrar información de las distintas partidas que jugaron los jugadores.



Aprovechando los beneficios de la modularización, un equipo de programadores ya se adelantó y especificó una librería llamada “wordix.php” que posee las funciones necesarias para jugar una (1) partida de wordix. Además de funciones que podremos utilizar (reusar).

Nuestra tarea como equipo será especificar el archivo “Programa<Apellidos>.php”. Donde <Apellidos> debe ser reemplazado por los apellidos de los integrantes del equipo.

El objetivo del programa es permitir a un usuario interactuar con el siguiente menú de opciones:

Menú de opciones:

- 1) Jugar al wordix con una palabra elegida
- 2) Jugar al wordix con una palabra aleatoria
- 3) Mostrar una partida
- 4) Mostrar la primer partida ganadora
- 5) Mostrar resumen de Jugador
- 6) Mostrar listado de partidas ordenadas por jugador y por palabra
- 7) Agregar una palabra de 5 letras a Wordix
- 8) salir

Cada opción del menú requiere que se invoquen una o varias funciones que serán descritas en la sección EXPLICACIÓN 3 del presente documento (también puede revisar si alguna función del archivo wordix.php puede ser utilizada).

A nivel general, de cada opción del menú, el usuario que ejecuta el programa espera lo siguiente:

- 1) Jugar al wordix con una palabra elegida: se inicia la partida de wordix solicitando el nombre del jugador y un número de palabra para jugar. Si el número de palabra ya fue utilizada por el jugador, el programa debe indicar que debe elegir otro número de palabra.
Luego de finalizar la partida, los datos de la partida deben ser guardados en una estructura de datos de partidas (ver la sección EXPLICACION 2, de Estructura de datos del presente enunciado)
- 2) Jugar al wordix con una palabra aleatoria: se inicia la partida de wordix solicitando el nombre del jugador. El programa elegirá una palabra aleatoria de las disponibles para jugar, el programa debe asegurarse que la palabra no haya sido jugada por el Jugador.
Luego de finalizar la partida, los datos de la partida deben ser guardados en una estructura de datos de partidas (ver la sección EXPLICACION 2, de Estructura de datos del presente enunciado)
- 3) Mostrar una partida: Se le solicita al usuario un número de partida y se muestra en pantalla con el siguiente formato:

Partida WORDIX <numero>: palabra <palabra>
Jugador: <nombre>
Puntaje: <puntaje> puntos
Intento: No adivinó la palabra | Adivinó la palabra en <X> intentos

Ejemplo al visualizar el juego número 13:

```
*****
Partida WORDIX 13: palabra MELON
Jugador: majo
Puntaje: 0 puntos
Intento: No adivinó la palabra
*****
```

Si el número de partida no existe, el programa deberá indicar el error y volver a solicitar un número de partida correcto.

- 4) Mostrar la primer partida ganadora: Se le solicita al usuario un nombre de jugador y se muestra en pantalla el primer juego ganado por dicho jugador. Por ejemplo si el usuario ingresa el nombre “Majo”

```
*****
Partida WORDIX 5: palabra MUJER
Jugador: majo
Puntaje: 11 puntos
Intento: Adivinó la palabra en 5 intentos
*****
```

En caso que el jugador no ganó ninguna partida, se debe indicar: “El jugador majo no ganó ninguna partida”.

- 5) Mostrar Estadísticas Jugador: Se le solicita al usuario que ingrese un nombre de jugador y se muestra la siguiente información:

```
*****
Jugador: majo
Partidas: 15
Puntaje Total: 105
Victorias: 10
Porcentaje Victorias: 66%
Adivinadas:
    Intento 1: 0
    Intento 2: 0
    Intento 3: 3
    Intento 4: 2
    Intento 5: 4
    Intento 6: 1
*****
```

- 6) Mostrar listado de partidas ordenadas por jugador y por palabra: Se mostrará en pantalla la estructura ordenada alfabéticamente por jugador y por palabra , utilizando la función predefinida **uasort** de php, y la función predefinida **print_r**. (En el código fuente documentar qué hace cada una de estas funciones predefinidas de php, utilizar el manual php.net). (Este es el único menú de opciones que debe utilizar la función **print_r** para mostrar la estructura de datos)
- 7) Agregar una palabra de 5 letras a Wordix: Debe solicitar una palabra de 5 letras al usuario y agregarla en mayúsculas a la colección de palabras que posee Wordix, para que el usuario pueda utilizarla para jugar.
- 8) Salir: Sale del programa.

EXPLICACIÓN 2 (Estructuras de datos)

La presente sección deberá ser utilizada para representar las estructuras de datos en el archivo “**DiseñoEstructuras<Apellidos>.pdf**” indicando los tipos de estructuras (indexado, asociativo o multidimensional), de qué tipo de datos son los índices y mostrar un ejemplo de la estructura (para representar las estructuras puede utilizar una planilla de cálculo o el software diagrams.net)

Archivo wordix.php:

Deben revisar el código del archivo **wordix.php** e identificar qué estructuras de datos se utilizan y representarlas (dibujarlas).

Archivo programaApellidos.php:

- a) Utilizará un arreglo de palabras (Colección de Palabras), de donde se elegirá una palabra para jugar la partida WORDIX.
- b) Utilizará una estructura indexada de arreglos asociativos que almacene información de las partidas que se jugaron. Cada arreglo asociativo tendrá el siguiente formato: (**"palabraWordix"=> "palabraX"**, **"jugador"=> "nombre"**, **"intentos"=> nroIntento**, **"puntaje"=> ptos**)

Por ejemplo:

```
coleccionPartidas[0] = ["palabraWordix" => "QUESO", "jugador" => "majo", "intentos"=> 0, "puntaje" => 0]
coleccionPartidas[1] = ["palabraWordix" => "CASAS", "jugador" => "rudolf", "intentos"=> 3, "puntaje" => 14]
coleccionPartidas[2] = ["palabraWordix" => "QUESO", "jugador" => "pink2000", "intentos"=> 6, "puntaje" => 10]
```

c) Utilizará una estructura asociativa para almacenar el resumen de un jugador que tendrá los siguientes datos: jugador, partidas, puntaje, victorias, intento1, intento2, intento3, intento4, intento5, intento6.

EXPLICACIÓN 3 (desde el punto de vista del Programador)

En wordix.php deberá buscar qué funciones están incompletas o tienen la documentación incompleta, y completarla. Como ayuda busque el texto ******COMPLETAR******. Es probable que alguna/as de las funciones ya implementadas en wordix.php puedan ser invocadas desde programaApellido.php

Como mínimo deberán implementar (Alguna función ya puede estar incluida en el archivo wordix.php):

1. Una función llamada cargarColeccionPalabras, que inicialice una estructura de datos con ejemplos de Palabras de 5 letras en mayúsculas y que retorne la colección de palabras descrita en la sección EXPLICACION 2. Mínimo debe cargar 15 palabras.
2. Una función llamada cargarPartidas, que inicialice una estructura de datos con ejemplos de Partidas y que retorne la colección de partidas descrita en la sección EXPLICACION 2. Mínimo debe cargar 10 partidas donde vayan variando los jugadores, las palabras, los intentos y los puntajes, en algunos casos los jugadores se deben repetir.
3. Para visualizar el menú de opciones (que siempre es el mismo), una función **seleccionarOpcion** que muestre las opciones del menú en la pantalla (ver sección EXPLICACION 1), le solicite al usuario una opción válida (si la opción no es válida vuelva a solicitarla en la misma función hasta que la opción sea válida), y retorne **el número** de la opción. La última opción del menú debe ser "Salir".
4. Una función que le pida al usuario ingresar una palabra de 5 letras, y retorne la palabra.
5. Una función que solicite al usuario un número entre un rango de valores. Si el número ingresado por el usuario no es válido, la función se encarga de volver a pedirlo. La función retorna un número válido.
6. Una función que dado un número de partida, muestre en pantalla los datos de la partida como lo indica la sección EXPLICACIÓN 1.
7. Una función agregarPalabra cuya entrada sea la colección de palabras y una palabra, y la función retorna la colección modificada al agregarse la nueva palabra. (Estructura c de la sección EXPLICACIÓN 2)
8. Una función que dada una colección de partidas y el nombre de un jugador, retorne el índice de la primer partida ganada por dicho jugador. Si el jugador ganó ninguna partida, la función debe retornar el valor -1. (debe utilizar las instrucciones vistas en la materia, no utilizar funciones predefinidas de php).
9. Una función que dada la colección de partidas y el nombre de un jugador, retorne el resumen del jugador utilizando la estructura c) de la sección EXPLICACIÓN 2.
10. Una función solicitarJugador sin parámetros formales que solicite al usuario el nombre de un jugador y retorne el nombre en minúsculas. La función debe asegurar que el nombre del jugador comience con una letra. (Utilice funciones predefinidas de string).
11. Una función sin retorno que, dada una colección de partidas, muestre la colección de partidas ordenada por el nombre del jugador y por la palabra. Utilice la función predefinida uasort de php y print_r.
12. Un Programa Principal que deberá seguir los siguientes pasos:
 - a. Precargar las estructuras de partidas.
 - b. Precargar la estructura de palabras.
 - c. Repetir el menú de opciones mientras la opción seleccionada no sea la opción Salir.
 - d. Cuando el usuario selecciona la opción del menú, debe invocar a la/s función/es necesarias. Salvo algunas excepciones, debe contar con funciones con parámetros formales y retorno. Asesorarse con la Cátedra para implementar las funciones correctamente de modo que los resultados de las funciones puedan ser reusados.
 - e. Investigar la instrucción **switch** en el manual de PHP. ¿a qué tipo de estructura de control vista en teoría corresponde? Escriba un comentario sobre la instrucción en el código fuente.
13. En el desarrollo utilizar funciones predefinida por PHP (strtolower, strtoupper, strlen, etc.). En la defensa se le preguntará que funciones predefinidas utilizaron, qué hacen dichas funciones y para qué las utilizaron.

Ayuda para organizar al grupo:

- a) El grupo deberá reunirse para leer todo el enunciado y resolver el instructivo de github
- b) Luego de identificar en qué consiste todo el enunciado la cátedra les recomienda llevar un documento compartido con el siguiente listado:
<https://docs.google.com/spreadsheets/d/1urpZKJyRKZUdsEy8KMGmyYjF4B6J76LTiGzfHLNmOwU/edit?usp=sharing> para dimensionar todo lo que hay que hacer y quién se puede ir encargando de cada tema. O bien utilizar algun programa para organizar las tareas como [trello](#).