

Ejercicios Tema 3

Arrays, Formularios y Cadenas



Germán Fuentes Ripoll
a 7 de Noviembre del 2021

Cifrado del César	2
Manual de Usuario	2
Manual Técnico	7
Minijuego: Encuentra el tesoro	15
Manual de Usuario	15
Manual Técnico	17

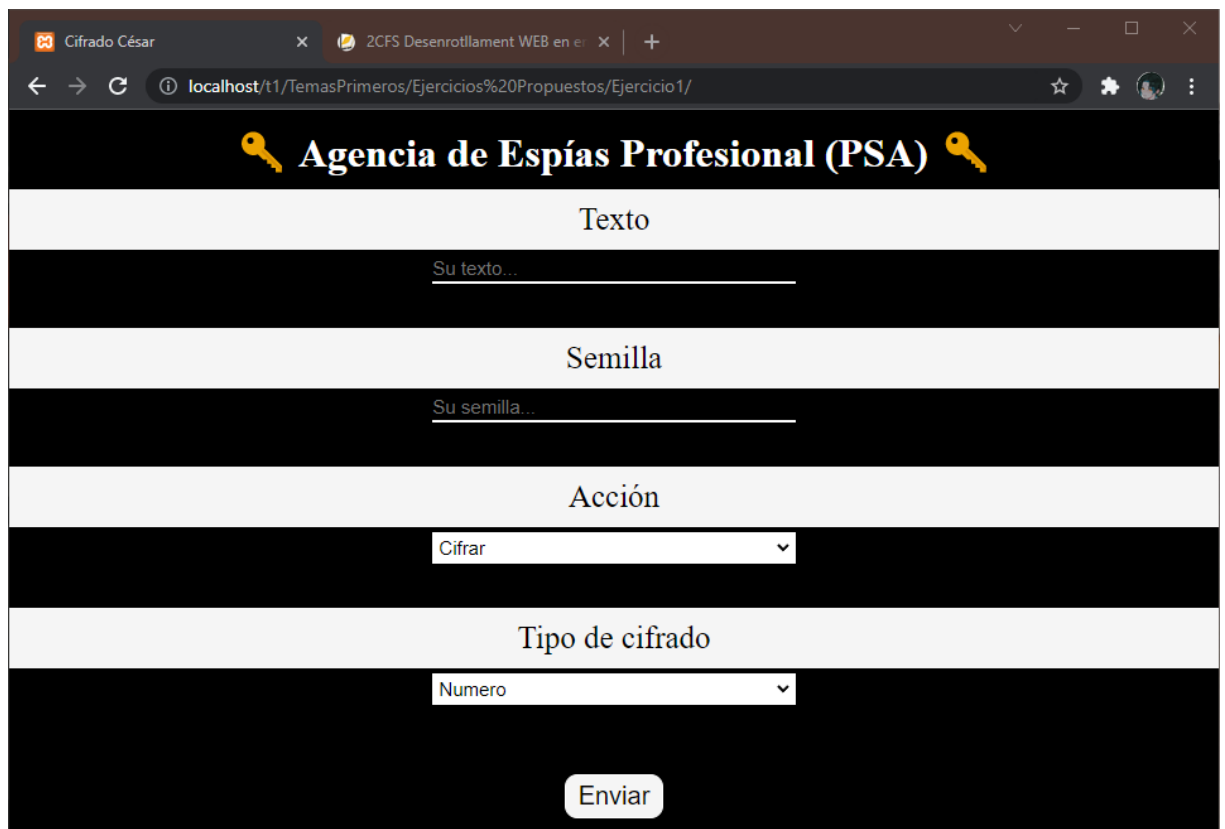
1. Cifrado del César

Esta aplicación consiste en el cifrado de textos mediante el método del César. Este método se basa en el desplazamiento de cada carácter del abecedario en un número determinado de posiciones o, por el contrario, desplazamientos aleatorios en base a una cadena “semilla”.

1.1. Manual de Usuario

La aplicación solicitará al usuario un texto que cifrar o descifrar, una semilla (numérica o cadena de texto dependiendo del tipo de cifrado), la acción de cifrar o descifrar y el tipo de cifrado (semilla) que utilizaremos.

El inicio de página de la aplicación mostrará por pantalla lo siguiente:





The screenshot shows a web browser window with the title 'Cifrado César'. The address bar shows the URL 'localhost/t1/TemasPrimeros/Ejercicios%20Propuestos/Ejercicio1/'. The page has a dark theme with a header bar containing the text 'Agencia de Espías Profesional (PSA)' flanked by two yellow key icons. Below the header, there are four main sections, each with a label and a corresponding input field or dropdown menu:



- Texto:** A text input field with the placeholder 'Su texto...'.
- Semilla:** A text input field with the placeholder 'Su semilla...'.
- Acción:** A dropdown menu with 'Cifrar' selected.
- Tipo de cifrado:** A dropdown menu with 'Numero' selected.

At the bottom of the form is a button labeled 'Enviar'.

Ahora procederemos a cifrar la cadena “Esta es mi cadena” mediante un cifrado numérico con valor 1:

 Agencia de Espías Profesional (PSA) 	
Texto	
<u>Esta es mi cadena</u>	
Semilla	
<u>1</u>	
Acción	
Cifrar	
Tipo de cifrado	
Numero	
Enviar	
Su mensaje cifrado es:	
FTUB FT NJ DBEFÑB	

Ahora cifraremos la misma cadena mediante un cifrado por cadena, utilizando la cadena “aba”:

 Agencia de Espías Profesional (PSA) 	
Texto	
<u>Esta es mi cadena</u>	
Semilla	
<u>aba</u>	
Acción	
Cifrar	
Tipo de cifrado	
Cadena	
Enviar	
Su mensaje cifrado es:	
FUUB FT NJ DBFFÑC	

Ahora procederemos a descifrar ambas cadenas para obtener las originales. Para ello deberemos de utilizar los mismos datos pero seleccionando la acción descifrar y utilizando la cadena cifrada:

🔑 Agencia de Espías Profesional (PSA) 🔑

Texto

FUUB FT NJ DBFFÑC

Semilla

aba

Acción

Descifrar ▾

Tipo de cifrado

Cadena ▾

Enviar

Su mensaje descifrado es:

ESTA ES MI CADENA

De nuevo utilizamos la cadena cifrada anteriormente con el mismo número:

🔑 Agencia de Espías Profesional (PSA) 🔑

Texto

FTUB FT NJ DBEFÑB

Semilla

1

Acción

Descifrar ▾

Tipo de cifrado

Numero ▾

Enviar

Su mensaje descifrado es:

ESTA ES MI CADENA

Para más comprobaciones le recomendamos al usuario que ejecute el programa por su cuenta debido a la ingente cantidad de combinaciones de esta.

Por el contrario mostraré los errores de inputs del programa:



- Se introduce un número como semilla e intenta cifrar con cadena:

The screenshot shows the 'Agencia de Espías Profesional (PSA)' interface. The 'Texto' field contains 'cadena'. The 'Semilla' field contains '25'. The 'Acción' dropdown is set to 'Cifrar'. The 'Tipo de cifrado' dropdown is set to 'Cadena'. The 'Enviar' button is visible. At the bottom, a red error message reads: 'La semilla debe de contener solo caracteres regulares.'

- Se introduce una cadena como semilla e intenta cifrar con número:

The screenshot shows the 'Agencia de Espías Profesional (PSA)' interface. The 'Texto' field contains 'cadena'. The 'Semilla' field contains 'cadena'. The 'Acción' dropdown is set to 'Cifrar'. The 'Tipo de cifrado' dropdown is set to 'Numero'. The 'Enviar' button is visible. At the bottom, a red error message reads: 'La semilla debe de ser un número entero positivo'.

- No se introducen valores:

 **Agencia de Espías Profesional (PSA)** 

Texto

Su texto...

Semilla

Su semilla...

Acción

Cifrar

Tipo de cifrado

Numero

Enviar

Introduzca algo antes de darle al botón

1.2. Manual Técnico

Lo primero de todo es incluir todos los métodos de mi archivo:

```
1  <?php
2  include_once('../funciones.php');
3
```

A continuación, si se ha pulsado el botón se recogen los datos de todos los campos y, en orden, se comprueba lo siguiente:

1. Campos no vacíos.
2. Acción realizada.
3. Cifrado utilizado.
4. Comprobaciones de errores en la pertinente cadenas o semilla.

```
if ($cadena === "" && $semilla === "") {
    $imprimirError = "Introduzca algo antes de darle al botón";
    $imprimir = "";
} else {
    if($accion == "cifrar") {
        if ($tCifrado == "V1") {
            if (ctype_digit($semilla) && $semilla >= 0 ) {
                $imprimir .= cifrarCesarV1($cadena,$semilla);
            } else {
                $imprimirError = "La semilla debe de ser un número entero positivo";
                $imprimir = "";
            }
        } else {
            if (soloLetras($semilla)) {
                $imprimir .= cifrarCesarV2($cadena,$semilla);
            } else {
                $imprimirError = "La semilla debe de contener solo caracteres regulares.";
                $imprimir = "";
            }
        }
    } else {
        if ($tCifrado == "V1") {
            if (ctype_digit($semilla) && $semilla >= 0) {
                $imprimir .= descifrarCesarV1($cadena,$semilla);
            } else {
                $imprimirError = "La semilla debe de ser un número entero positivo";
                $imprimir = "";
            }
        } else {
            if (soloLetras($semilla) ) {
                $imprimir .= descifrarCesarV2($cadena,$semilla);
            } else {
                $imprimirError = "La semilla debe de contener solo caracteres regulares.";
                $imprimir = "";
            }
        }
    }
}
```


Ahora la carne del asador se encuentra en el archivo de funciones, donde echaremos un vistazo a cada una de las funciones creadas:

- **generaAbc().**

```
1  <?php
2  // Ejercicio 1
3  function generaAbc() {
4      $arrayAbc = [];
5      for ($i=65;$i<=91;$i++) {
6          if ($i==79) {
7              $arrayAbc[$i-65] = "Ñ";
8          } else if ($i >= 79){
9              $arrayAbc[$i-65] = chr($i-1);
10         } else {
11             $arrayAbc[$i-65] = chr($i);
12         }
13     }
14     return $arrayAbc;
15 }
```

En esta función generamos y devolvemos el array que contiene todas las letras del abecedario.

- **letraToNumero(\$letra).**

```
156 function letraToNumero($letra) {
157     $abc = generaAbc();
158     foreach($abc as $indice => $valor) {
159         if ($valor == $letra) {
160             return $indice+1;
161         }
162     }
163     return null;
164 }
```

En esta función recorremos el array generado del abecedario para encontrar y devolver el valor de desplazamientos al que equivale la letra pasada como parámetro.

- **stringToArray(\$cadena).**

```
165 function stringToArray($cadena) {
166     $array = [];
167     for($i=0;$i< strlen($cadena); $i++) {
168         $array[$i] = $cadena[$i];
169     }
170     return $array;
171 }
```

Esta función hace las veces de explode(), ya que para esta práctica en concreto no podíamos utilizar esta función.

- **soloLetras(\$in).**

```
172 function soloLetras($in){
173     if(preg_match('/^[a-zA-ZÑ\S]+$/i', $in)) return true;
174     else return false;
175 }
```

Esta función se utiliza para comprobar que el array suministrado sólo tenga los caracteres mostrados en el primer parámetro del preg_match().

Esto es posible utilizando expresiones regulares, como la mostrada en rojo en el código.

- **cifrarCesarV1(\$cadena,\$semilla).**

```
16 function cifrarCesarV1($cadena,$semilla) {
17     $cadena = mb_strtoupper($cadena);
18     $cifrado = "";
19     $abc = generaAbc();
20     $bandera = false;
21     for($i=0;$i<strlen($cadena);$i++) {
22         for($j=0;$j<count($abc) && !$bandera;$j++) {
23             if ($cadena[$i] == $abc[$j]) {
24                 $cifrado .= $j+$semilla>26?$abc[(($j+$semilla)%27)]:$abc[$j+$semilla];
25                 $bandera = true;
26             } if ($cadena[$i-1].$cadena[$i] == $abc[$j]) {
27                 $cifrado .= $j+$semilla>26?$abc[(($j+$semilla)%27)]:$abc[$j+$semilla];
28                 $bandera = true;
29             }
30         }
31         if (!$bandera && $cadena[$i] != chr(195)) {
32             $cifrado .= $cadena[$i];
33         }
34         $bandera = false;
35     }
36     return $cifrado;
37 }
```

Esta función se encargará de cifrar mediante desplazamiento numérico la cadena pasada por parámetro.

Lo primero es generar el array del abecedario, pasar la cadena a mayúscula para tratar mejor con esta y generar la cadena de cifrado vacía para ir rellenándola más tarde.

En el primer bucle recorre la cadena y por cada vuelta se recorre también el abecedario, si la letra del primero coincide con la del segundo se imprime, mediante expresión regular el valor actual de la letra más el de la semilla.

A este resultado se le hace módulo de 27 en base a si el número resultante supera el máximo valor del abecedario ($Z = 27$). Haciendo esto nunca sobrepasará este valor.

Este código es debido a la letra Ñ, ya que este carácter es especial y ocupa dos posiciones en el String:

```
} if ($cadena[$i-1].$cadena[$i] == $abc[$j]) {  
    $cifrado .= $j+$semilla>26?$abc[(($j+$semilla)%27]:$abc[$j+$semilla];  
    $bandera = true;  
}
```

Por último si ninguna letra coincide y se termina el bucle del abecedario la letra, tal cual se imprime en el resultado, a excepción de `chr(195)` que es el primer carácter de la Ñ:

```
if (!$bandera && $cadena[$i] != chr(195)) {  
    $cifrado .= $cadena[$i];  
}
```

- descifrarCesarV1(\$cadena,\$semilla).

```

38 function descifrarCesarV1($cadena,$semilla) {
39     $cadena = mb_strtoupper($cadena);
40     $descifrado = "";
41     $abc = generaAbc();
42     $bandera = false;
43     for($i=0;$i<strlen($cadena);$i++) {
44         for($j=0;$j<count($abc) && !$bandera;$j++) {
45             if ($cadena[$i] == $abc[$j]) {
46
47                 $descifrado .= $j-$semilla<0?
48                     $abc[(($j-($semilla%27)))]
49                     :
50                     $abc[$j-$semilla];
51
52                 $bandera = true;
53             } if ($cadena[$i-1].$cadena[$i] == $abc[$j]) {
54
55                 $descifrado .= $j-$semilla<0?
56                     $abc[(($j-($semilla%27)))]
57                     :
58                     $abc[$j-$semilla];
59                 $bandera = true;
60             }
61         }
62         if (!$bandera && $cadena[$i] != chr(195)) {
63             $descifrado .= $cadena[$i];
64         }
65         $bandera = false;
66     }
67     return $descifrado;
68 }

```

Esta función descifra la cadena mediante una semilla numérica. Realmente es muy parecida a la anterior cambiando lo siguiente:

```

    $descifrado .= $j-$semilla<0?
        $abc[(($j-($semilla%27)))]
        :
        $abc[$j-$semilla];

```

Esta parte resta la semilla al valor de la letra actual, si este resultado es menor que 0 se hace módulo de 27 a la semilla. Este módulo es por el mismo motivo que la función anterior.

La parte de la Ñ queda igual a esta.

- cifrarCesarV2(\$cadena,\$cadenaSemilla).

```

69 function cifrarCesarV2($cadena,$cadenaSemilla) {
70     $cadena = mb_strtoupper($cadena);
71     $cadenaSemilla = mb_strtoupper($cadenaSemilla);
72     $cadenaSemilla = stringToArray($cadenaSemilla);
73     $cifrado = "";
74     $abc = generaAbc();
75     $bandera = false;
76     for($i=0;$i<strlen($cadena);$i++) {
77         for($j=0;$j<count($abc) && !$bandera;$j++) {
78             if ($cadena[$i] == $abc[$j]) {
79                 if(letraToNumero($cadenaSemilla[$i% count($cadenaSemilla)]) == null) {
80                     $semilla = letraToNumero("Ñ");
81                     array_splice($cadenaSemilla,$i,1);
82                 } else {
83                     $semilla = letraToNumero($cadenaSemilla[$i% count($cadenaSemilla) ]);
84                 }
85                 $cifrado .= $j+$semilla>26?$abc[(($j+$semilla)%27):$abc[$j+$semilla];
86                 $bandera = true;
87             } else if ($cadena[$i-1].$cadena[$i] == $abc[$j]) {
88                 if(letraToNumero($cadenaSemilla[$i% count($cadenaSemilla)]) == null) {
89                     $semilla = letraToNumero("Ñ");
90                     array_splice($cadenaSemilla,$i,1);
91                 } else {
92                     $semilla = letraToNumero($cadenaSemilla[$i% count($cadenaSemilla) ]);
93                 }
94                 $cifrado .= $j+$semilla>26?$abc[(($j+$semilla)%27):$abc[$j+$semilla];
95                 $bandera = true;
96             }
97         }
98         if (!$bandera && $cadena[$i] != chr(195)) {
99             $cifrado .= $cadena[$i];
100         }
101         $bandera = false;
102     }
103     return $cifrado;
104 }

```

Esta función se encarga de cifrar una cadena en base a otra, de esta manera se consigue un desplazamiento no fijo como pasaba en la versión 1 del César.

Por partes, dentro de los bucles (misma estructura de siempre) se comprueba si las letras coinciden, si es así se comprueba si la letra es la Ñ, si es así se fuerza a letraToNumero() a devolver ese valor y se utiliza *array_splice()* para quitar el segundo carácter de la ñ.

Si no es así simplemente se devuelve la semilla de la letra pertinente.

A continuación en la línea 85 se asigna la nueva letra al String que estamos montando, de nuevo controlando que no se pase de la longitud máxima del abecedario.

La segunda parte del primer if se centra en el tratamiento de la Ñ.


- descifrarCesarV2(\$cadena,\$cadenaSemilla).

```
105 function descifrarCesarV2($cadena,$cadenaSemilla) {
106     $cadena = mb_strtoupper($cadena);
107     $cadenaSemilla = mb_strtoupper($cadenaSemilla);
108     $cadenaSemilla = stringToArray($cadenaSemilla);
109     $descifrado = "";
110     $abc = generaAbc();
111     $bandera = false;
112     $volver = 0;
113     for($i=0;$i<strlen($cadena);$i++) {
114         for($j=0;$j<count($abc) && !$bandera;$j++) {
115             if ($cadena[$i] == $abc[$j]) {
116                 if(letraToNumero($cadenaSemilla[$i% count($cadenaSemilla)]) == null) {
117                     $semilla = letraToNumero("Ñ");
118                     //Ocupaba dos espacios por eso SPLICE quita la siguiente posicion a la actual
119                     array_splice($cadenaSemilla,$i,1);
120                 } else {
121                     $semilla = letraToNumero($cadenaSemilla[$i% count($cadenaSemilla) ]) +$volver;
122                 }
123
124                 $descifrado .= $j-$semilla<0?
125                     $abc[(($j-($semilla%27)) + count($abc))]
126                     :
127                     $abc[$j-$semilla];
128
129                 $bandera = true;
130             } else if ($cadena[$i % strlen($cadena)].$cadena[(($i+1) % strlen($cadena))] == $abc[$j]) {
131                 if(letraToNumero($cadenaSemilla[$i% count($cadenaSemilla)]) == null) {
132                     $semilla = letraToNumero("Ñ");
133                     //Ocupaba dos espacios por eso SPLICE quita la siguiente posicion a la actual
134                     array_splice($cadenaSemilla,$i,1);
135                 } else {
136
137                     $semilla = letraToNumero($cadenaSemilla[$i% count($cadenaSemilla)]);
138                 }
139                 $descifrado .= $j-($semilla+1)<0?
140                     $abc[(($j-($semilla%27)) + count($abc))]
141                     :
142                     $abc[$j-$semilla];
143
144                 $bandera = true;
145                 $i++;
146                 $volver++;
147             }
148         }
149         if (!$bandera && $cadena[$i] != chr(195)) {
150             $descifrado .= $cadena[$i];
151         }
152     }
153     $bandera = false;
154     return $descifrado;
155 }
```

Por partes, esta función se encarga de descifrar mediante cadena un texto pasado por parámetro y para ello realiza lo siguiente:

Dentro de los bucles, se comprueba que sea Ñ o no la letra cifrar para conseguir el valor de la semilla. Se realiza lo mismo con el `array_splice()` y creación del mensaje descifrado.

La segunda parte del `if` principal consiste en saber si los siguientes caracteres juntos equivalen a la Ñ, en caso de ser así se toma el valor de la Ñ, si la cadena que cifra es esta letra, u otro en base a la letra pertinente.

Por último se incrementa el valor del bucle de la cadena a descifrar en 1 adicional (cosa que no me gusta haber hecho así, pero bueno) en caso de ser la Ñ la palabra a descifrar. Esto es debido a que deberemos ignorar el siguiente carácter de la Ñ o recibiremos este fantástico rombo .

2. Minijuego: Encuentra el tesoro

Esta aplicación es básicamente un minijuego de busca el tesoro. En este podremos recorrer una mazmorra a oscuras para tratar de encontrar un enorme botín. Durante nuestra ciega travesía podremos encontrarnos a enemigos y luchar con ellos hasta la muerte o peor, morir presos de la locura si pasamos mucho tiempo explorando.

2.1. Manual de Usuario

Lo primero que veremos al cargar la aplicación será lo siguiente:



En este juego tenemos dos medidores:

- Cordura, la cual descenderá en 1 por cada paso a la oscuridad que demos, o en 2 por cada enemigo que nos encontremos.
- Vidas, la cual descenderá en 1 por cada enemigo encontrado, al cual derrotamos y podremos seguir avanzando, a menos que tengamos 0, donde perderemos el combate y terminará el juego.

A continuación mostraré algunas de las posibilidades de desenlace del juego:

- El jugador encuentra la casilla con el tesoro.



- El jugador muere por locura o por vidas.



En este último caso se revelará la posición del tesoro y de los enemigos para cabrear al usuario.

2.2. Manual Técnico

Lo primero de todo es ver el array que contiene el tablero original vacío, el cual rellenaremos más tarde:

```
//Creacion del arrayTablero sin eventos
$arraytablero = [
  1 => [ ...
    1,
    2 => [
      1 => [ "wallL" => "wall", ],
      2 => [ "floor" => "floor", ],
      3 => [ "floor" => "floor", ],
      4 => [ "floor" => "floor", ],
      5 => [ "floor" => "floor", ],
      6 => [ "floor" => "floor", ],
      7 => [ "floor" => "floor", ],
      8 => [ "floor" => "floor", ],
      9 => [ "floor" => "floor", ],
      10 => [ "floor" => "floor", ],
      11 => [ "wallR" => "wall", ],
    ],
  3 => [ ...
    1
```

Cada celda consiste en dos valores, el primero, que es el valor que se va a imprimir, ya sea en formulario o en imagen, y el segundo, el cual indica el tipo de celda (por temas de movimiento).

Los valores pueden oscilar entre los siguientes:

- WallL, WallR, WallT, WallB
 - Estas representan las paredes rectas de los bordes del tablero, no se modifican nunca.
- CornerTL, CornerTR, CornerBL, CornerBR
 - Estas representan las esquina del tablero, no se modifican nunca.
- floor, floorC
 - Estas representan las casillas por las que has pasado (floorC) o aquellas a oscuras (floor).
- Treasure, Enemy, EnemyDefeated
 - Estas representan los eventos del tablero, donde treasure pone fin al juego directamente y Enemy también en caso de perder tu última vida en él.

Las imágenes utilizadas se nombran igual para facilitar el uso de estas.

Ahora deberemos saber cómo narices se ha creado esto. Para ello deberemos echar un vistazo a la función `generarTablero()`, la cual se encarga de generar “aleatoriamente” el mapa de la mazmorra.

Entre comillas por temas que comentaré al final y que (probablemente?) podría solucionar en un futuro.

Lo primero es lo primero, deberemos de generar un array multidimensional en el cual guardaremos las posiciones en las cuales los eventos podrán hacer spawn:

```
function generarTablero($arrayTablero,$filaInicial,$columnaInicial,$cordura,$vidas,$send) {  
    $arraySpawnsValidos = [  
        2 => [0,0,0,0,2,3,4,5,6,7,8,9,10],  
        3 => [0,0,0,0,2,3,4,5,6,7,8,9,10],  
        4 => [0,0,0,0,2,3,4,5,6,7,8,9,10],  
        5 => [0,0,0,0,2,3,4,5,6,7,8,9,10],  
        6 => [0,0,0,0,2,3,4,5,6,7,8,9,10],  
        7 => [0,0,0,0,2,3,4,5,6,7,8,9,10],  
        8 => [0,0,0,0,2,3,4,8,9,10],  
        9 => [0,0,0,0,2,3,4,8,9,10],  
    ];  
}
```

Esto se debe a que no podemos permitir que los eventos se creen en las paredes ni en las cercanías a la entrada.

El primer valor del array es la fila del tablero, la 1 y la 10 no aparecen porque la 1 es todo pared y la 10 es pared e inicio de personaje.

El resto de celdas se permiten a excepción de los siguientes:

- La primera casilla (1) de cada fila, debido a que esta es siempre pared.
- La última casilla (11) de cada fila, debido a lo mismo.
- Las posiciones 5,6,7 de las filas 8 y 9 porque son las cercanías de la entrada.

Por último comentar que los 0s significan que no se generará nada (esto lo veremos ahora).

A continuación haremos un bucle:

```
538     do {
539         $spawns = [
540             2 => $arraySpawnsValidos[2][array_rand($arraySpawnsValidos[2])],
541             3 => $arraySpawnsValidos[3][array_rand($arraySpawnsValidos[3])],
542             4 => $arraySpawnsValidos[4][array_rand($arraySpawnsValidos[4])],
543             5 => $arraySpawnsValidos[5][array_rand($arraySpawnsValidos[5])],
544             6 => $arraySpawnsValidos[6][array_rand($arraySpawnsValidos[6])],
545             7 => $arraySpawnsValidos[7][array_rand($arraySpawnsValidos[7])],
546             8 => $arraySpawnsValidos[8][array_rand($arraySpawnsValidos[8])],
547             9 => $arraySpawnsValidos[9][array_rand($arraySpawnsValidos[9])],
548         ];
549     } while ($spawns[2] === 0 && $spawns[3] === 0 && $spawns[4] === 0 &&
550         $spawns[5] === 0 && $spawns[6] === 0 && $spawns[7] === 0 &&
551         $spawns[8] === 0 && $spawns[9] === 0);
552
```

Donde crearemos un array para las casillas de los eventos, para cada línea se obtiene una celda al azar de esta.

Esto se repetirá el número de veces que sea necesario hasta generar al menos 1 evento (el del tesoro).

Esto es lo que comentaba de “aleatoriedad” ya que por cada fila solo se puede generar 0 o 1 evento, por lo que si vences a un enemigo en una fila, sabes que a lo largo de esta no habrá ni enemigo ni tesoro, por lo que realmente se puede trucar el juego si lo conoces por dentro. Pero bueno, la vida es dura.

Ahora deberemos obtener el número de eventos, que nos hará falta más tarde:

```
554     $numEventos = 0;
555     foreach ($spawns as $fila => $columna) {
556         if ($columna !== 0) {
557             $numEventos++;
558         }
559     }
560
```

Ahora la parte final:

```
561     if ($numEventos === 1) {
562         foreach ($spawns as $fila => $columna) {
563             if ($columna !== 0) {
564                 $arrayTablero[$fila][$columna] = ["treasure"=>"floor"];
565             }
566         }
567     } else {
568         $treasure = rand(1,$numEventos);
569         $i = 1;
570         // Recorro arraySpawns para añadir los eventos al arrayTalero
571         foreach ($spawns as $fila => $columna) {
572
573
574             // Si no es un 0, es decir, si se tiene que generar algo en dicha linea
575             if ($columna !== 0) {
576                 // Comprobar si la posición del tesoro es la del evento a incluir
577                 if ($i === $treasure) {
578                     // si se cumple se añade el tesoro
579                     $arrayTablero[$fila][$columna] = ["treasure" => "floor"];
580                 } else {
581                     // en caso contrario seguimos añadiendo enemigos
582                     $arrayTablero[$fila][$columna] = ["enemy" => "floor"];
583                 }
584                 $i++;
585             }
586
587         }
588     }
589 }
590 }
```

Primero comprobaremos cuántos eventos se han generado:

- Si solo se ha generado 1 evento.

Se asigna a este el valor del tesoro.

- Si se han generado más de 1 evento

Se selecciona una posición al azar del número de eventos generado para signársela al tesoro.

Se recorre el array de spawns para añadir los eventos a las posiciones del tablero que este contenga.

Si la fila actual coincide con la del tesoro se asigna este a la casilla del tablero.

Sino se asignará un enemigo a esta.

Por último en esta función se llama a `tablaTablero()` la cual se encarga de generar la tabla con los pertinentes formularios o imágenes en base a la posición del jugador:

```
591     $imprimir = tablaTablero($arrayTablero,$filaInicial,$columnaInicial,$cordura,$vidas,$end);
592     return $imprimir;
593 }
```

Ahora vamos con **tablaTablero()**, la cual primero se encarga de serializar el array del tablero a una cadena, la cual será asignada como hidden a los formularios de movimiento:

```
477 // Ejercicio 8
478 function tablaTablero($arrayTablero,$filaActual,$columnaActual,$cordura,$vidas,$end) {
479
480     $cadenaTablero = array_a_cadenaurl($arrayTablero);
481     $cadena = "<table>";
482 }
```

Ahora deberemos recorrer las filas del array tablero, por cada fila recorreremos sus columnas y por cada columna recorreremos sus datos.

Ahora presta atención:

- Si la **filaActual** es igual a la fila en la que estás y la columna es la siguiente o la anterior.
- O por el contrario, la **columnaActual** es igual a la columna en la que estás y la fila es la superior o inferior inmediata.
- Y la casilla no es pared.
- Y el número de vidas es mayor o igual a 1.
- Y la cordura es mayor o igual a 1.

Entonces crearemos para dicha celda un formulario hacia sí mismo.

La imagen la asignamos en base al valor de esta misma posición en el array del tablero. Donde deberemos de comprobar si este es "floor".

En caso de serlo pondremos dicha imagen, y en caso contrario pondremos la imagen con el valor pertinente ("floorC","enemyDefeated").

El resto del formulario son hiddens para poder llevarme todos los valores con cada reenvío a sí mismo.

Si no es así deberemos de seguir la misma lógica pero rellenando la celda con una imagen en vez de un formulario.

Después de eso cerramos toda la tabla y devolvemos el valor.

Ahora vamos con el archivo del juego donde iniciaremos el array mostrado al principio e iniciaremos los valores del juego:

```
};  
// Inicialización de valores  
$filaInicial = 10;  
$columnaInicial = 6;  
$cordura = 40;  
$vidas = 4;
```

Los dos siguientes for son para construir las barras de cordura y corazones respectivamente:

```
$score = "<p>Cordura: ";  
for($i=0;$i<$cordura;$i++) {  
    $score .= "<img src='images/staminaChunk.png' width='18' height='50'>";  
}  
$score .="</p>  
<p>Vidas: ";  
for($i=0;$i<$vidas;$i++) {  
    $score .= "<img src='images/heart.png' width='50' height='50'>";  
}  
$score .="</p>";
```

Por último se llama a generarTablero(), la cual tiene tablaTablero() integrada dentro de sí:

```
$imprimir = generarTablero($arraytablero,$filaInicial,$columnaInicial,$cordura,$vidas,$end);
```

Con esto ya tendríamos lo primero que veríamos al entrar al juego.

Ahora toca manejar el resto de veces que carguemos la página, donde ya habremos pulsado una casilla para movernos.

Primero recogemos hasta a mi perro:

```
334 } else {
335     //Recogida de datos o inicialización de estos
336     $cadenaTablero = $_POST["array"];
337     $arraytablero = cadenaurl_a_array($cadenaTablero);
338     $filaActual = $_POST["filaActual"];
339     $columnaActual = $_POST["columnaActual"];
340     $filaAnterior = $_POST["filaAnterior"];
341     $columnaAnterior = $_POST["columnaAnterior"];
342     $cordura = $_POST["cordura"];
343     $vidas = $_POST["vidas"];
344     $end = false;
345     $imagen = "";
346     $mensaje = "";
```

Todos estos valores son necesarios para más tarde, así como la inicialización de otros.

Pasemos al siguiente fragmento de código:

```
357 $cambio = "currentPoss";
358 // Condicional de cambio de casilla actual
359 $typeActual;
360 foreach ($arraytablero[$filaActual][$columnaActual] as $type => $value) {
361     if ($type == "enemy") {
362         $cordura -= 2;
363         $vidas--;
364         if ($vidas > 0) {
365             $cambio = "clashVictory";
366         } else {
367             $cambio = "clashDefeat";
368         }
369     } else if($value == "floor") {
370         $cordura -= 1;
371     }
372     $typeActual = $type;
373 }
374 $arraytablero[$filaActual][$columnaActual] = [$cambio => $cambio];
375
```

Aquí recorreremos los valores de la casilla a la que nos hemos desplazado.

Si el tipo de la casilla era un enemigo nuestra cordura bajará en 2 y perderemos 1 vida. Si tras restar esa vida estas son superiores a 0, se mostrará la imagen en celda de "Victoria", en caso contrario se mostrará la imagen en celda de "Derrota".

Por el contrario si la celda no es un enemigo y es suelo sin más sólo perderemos 1 de cordura.

Ahora guardaremos el valor de esta celda a la que nos vamos a mover, ya que deberemos de actualizarla cuando la abandonemos.

Por último asignamos a la posición a la que nos vamos a mover el valor de **\$cambio**, donde será el cartel de victoria o derrota si nos hemos encontrado con un enemigo o el jugador si solo había suelo.

Para la casilla anterior deberemos de coger el valor de columna anterior y, en caso de ser el combate ganado contra un enemigo imprimimos en esa unos huesos indicando que ahí hemos vencido a un enemigo.

En caso contrario simplemente se imprime floorC (floor Cleared).

```
376     $cambio = "floorC";
377     // Condicional de cambio de casilla anterior
378     foreach ($arraytablero[$filaAnterior][$columnaAnterior] as $type => $value) {
379         if ($type == "clashVictory") {
380             $cambio = "defeatedEnemy";
381         }
382     }
383     $arraytablero[$filaAnterior][$columnaAnterior] = [$cambio => $cambio];
384
```

Ahora imprimimos las vidas y cordura actuales, solo tras haberlas restado anteriormente:

```
385     // Imprimir contador de vidas y cordura
386     $score = "<p>Cordura: ";
387     for($i=0;$i<$cordura;$i++) {
388         $score .= "<img src='images/staminaChunk.png' width='18' height='50'>";
389     }
390     $score .="</p>";
391     $score .="<p>Vidas: ";
392     for($i=0;$i<$vidas;$i++) {
393         $score .= "<img src='images/heart.png' width='50' height='50'>";
394     }
395     $score .="</p>";
396
```

Por último en este ejercicio deberemos de comprobar cuando el usuario ha ganado o ha perdido:

```
397 // Comprobar si ha ganado el juego, si lo ha perdido o si el juego continua
398 if($vidas == 0 || $cordura == 0) {
399     // $imagen = "<img src='images/gameDefeat.png' id='endGame'>";
400     $send = true;
401     $mensaje = "<p>Has Perdido!</p>";
402     $imprimir = tablaTablero($arraytablero,$filaActual,$columnaActual,$cordura,$vidas,$send);
403 }
404 if($typeActual == "treasure") {
405     $imagen = "<img src='images/gameVictory.png' id='endGame'>";
406     $mensaje = "<p>Has Ganado!</p>";
407     $imprimir = "";
408 } else {
409     $imprimir = tablaTablero($arraytablero,$filaActual,$columnaActual,$cordura,$vidas,$send);
410 }
411
```

Para ello tenemos que comprobar primero que la cordura y las vidas sean distintas de 0, porque sino \$send será true (no imprimiendo más formularios, solo imágenes).

Comprobaremos luego si la casilla a la que nos hemos movido es el tesoro, por lo que directamente hemos ganado, no mostramos tablero por lo que no hay que modificar nada de \$send y mostramos un mensaje de victoria para el usuario.

En el caso contrario, si la casilla no es treasure y se tiene cordura y vidas, se sigue jugando.

No hay botón de “Volver a Jugar” porque no me gusta crear ludópatas.

FIN

