Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Operaciones de conjuntos

Actividad 5

Materia:

Aplicaciones web

Integrantes del equipo:

Rosa Suarez Calderon

Ana Laura Torres Marin

Alfredo de Jesus Ramos Benavides

Profesor:

Pedro Bello López

Periodo:

Primavera 2022

Fecha de entrega:

08 de febrero de 2022

Desarrollo de trabajo.

CODIGO PARA LA ELABORACION DE FORMULARIO

Para la elaboración del trabajo fue necesario usar un formulario para que el usuario pudiera introducir los datos, para ello usamos la etiqueta `<form action= “” method=”post”>`. Dentro del formulario necesitamos dos “cajas de texto”, para ello usamos la etiqueta `<input=“” type=”text”>` y, usamos la etiqueta `<input=”submit” value=””>` para que al momento de que el usuario de clic al botón, lo dirija a una nueva pagina que mostrara los resultados. A continuación, se muestra el formulario elaborado:

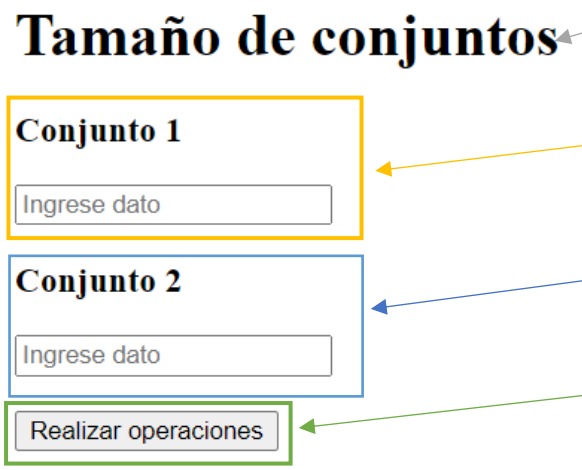
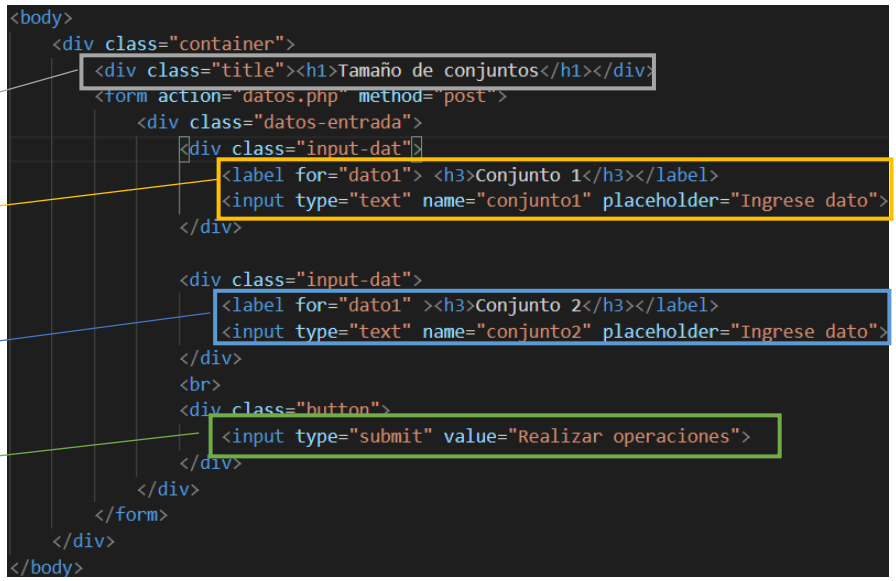
Resultado obtenido	Código implementado
	

Tabla 1. Formulario elaborado.

Como observamos en la Tabla 1, ocupamos las etiquetas antes mencionadas, si observamos mas a detalle, las etiquetas label, tienen un texto, y este texto esta encerrado en otras etiquetas como `<h1>` y `<h3>` estas nos ayudan a ponerle un estilo diferente al texto que se muestra.

Una vez que tenemos el código del formulario lo siguiente seria crear el archivo “datos.php” en el cual hacemos uso de los datos que el usuario nos ha dado.

Antes de crear el archivo “datos.php” crearemos el archivo que contiene a la clase “Conjuntos”, en esta clase se pondrán los métodos necesarios para que se pueda crear el conjunto, a continuación, se muestra el código de la clase Conjunto.

CODIGO DE LA CLASE CONJUNTO.

Para representar los conjuntos en PHP usamos los arreglos. Para realizar las operaciones de los conjuntos, necesitamos al menos dos conjuntos que contengan datos para que podamos realizar las operaciones, es por ello que, para crear los conjuntos necesitamos tener al menos su tamaño, por ello, el constructor de la clase debe tener como parámetro el tamaño del conjunto, como se muestra en la imagen siguiente:

```php
class Conjunto{
  private $tam;

  public function __construct($tam){
    $this->tam = $tam;
  }
```

Figura 1. Constructor de la clase Conjunto.

Una vez que recuperamos el tamaño del nuevo objeto de tipo Conjunto, debemos de crear el arreglo, llenarlo con datos aleatorios y después deberíamos de imprimir el arreglo para visualizar los datos, es por ello que se crearon dos funciones, una para llenar los conjuntos con datos aleatorios y la otra función para imprimir los datos, a continuación, se muestran dichas funciones:

```php
public function llenarArreglo(){
  $numeros = array();
  for($i=0; $i<$this->tam;$i++){
    $numeros[$i] =  rand(1,20);
  }
  return $numeros;
}

public function imprimirArreglo($m){
  echo"Tamaño del conjunto:$this->tam";
  echo "<table border align = 'center'>";
  for($i=0;$i<1;$i++){
    echo "<tr>";
      for ($j=0; $j < $this->tam; $j++) {
        echo "<td>";
        echo $m[$j];
        echo "</td>";
      }
      echo "</tr>";
  }
  echo "</table>";
}
```

Figura 2. Bloque de código que llena un arreglo e imprime el mismo.

Hay algo importante que debemos de mencionar y es que, en los ciclos for, usamos lo siguiente `&this->tam` esta asignación o referencia la usamos para que el objeto creado sea con el tamaño que el usuario a introducido, si no hacemos uso de esto, el tamaño del arreglo seria distinto al que el usuario haya indicado.

Para realizar las operaciones de los conjuntos escribimos tres funciones distintas, las operaciones que realizamos son: Unión, Intersección y Diferencia. El lenguaje de programación PHP tiene funciones especiales que hacen las operaciones de conjuntos, hicimos uso de ellos para realizar las operaciones, a continuación, se muestran las funciones antes mencionadas:

```php
public function union($c1,$c2){
  $aux_union = array_merge($c1,$c2);
  $union =  array_values(array_unique($aux_union));
  return $union;
}
```

Figura 3. Función que realiza la unión de los arreglos.

```php
public function interseccion($c1,$c2){
  $aux_inter = array_unique(array_intersect($c1,$c2));
  $interseccion = array_values($aux_inter);
  return $interseccion;
}
```

Figura 4. Función que realiza la intersección de arreglos.

```php
public function diferencia($c1,$c2){
  $aux_dif = array_unique(array_diff($c1,$c2));
  $diferencia = array_values($aux_dif);

  return $diferencia;

}
```

Figura 5. Función que realiza la diferencia.

Las funciones especiales que manejan las operaciones de conjuntos son: `array_merge()`, `array_intersect()` y `array_diff()`. La función `array_merge()` nos ayuda a mezclar dos arreglos, la función `array_intersect()` nos ayuda a identificar los datos que se repiten en ambos arreglos, almacenándolos en una nueva variable y la función `array_diff()` nos ayuda a hacer la diferencia de arrays.

En las Figuras anteriores observamos que hay mas funciones especiales de php, la función `array_unique()` nos ayuda a eliminar los datos que se repiten en el arreglo y la función `array_values()` nos ayuda a reindexar el arreglo, debido a que se eliminan los datos repetidos es necesario volver a hacer esto para que el tamaño del conjunto coincida, de no hacerlo el tamaño del arreglo sería diferente, lo que ocasionaría que se pase del rango.

Una vez que se hacen las operaciones correspondientes retornamos el nuevo arreglo para que sea usado a nuestra conveniencia.

CODIGO QUE CREA Y LLAMA A LOS METODOS DE LA CLASE CONJUNTO (Archivo datos.php).

Una vez que tenemos la clase que contiene las funciones para las operaciones se procedió a crear los conjuntos para hacer las operaciones, a continuación, se explicaron los pasos que se siguieron para obtener los resultados deseados:

1. Usamos la sentencia `require_once` para expresar que requerimos del archivo donde se encuentra la clase Conjunto (Figura 6).

```
require_once "conjunto.php";
```

Figura 6. Llamado al archivo donde se encuentra la Clase Conjunto.

2. Obtenemos los datos que el usuario ingreso en el formulario, y con ello creamos dos nuevos conjuntos (Figura 7).

```
$cn1 = $_REQUEST['conjunto1'];
$cn2 = $_REQUEST['conjunto2'];


$conjunto1 = new Conjunto($cn1);
$conjunto2 = new Conjunto($cn2);
```

Figura 7. Creación de conjuntos a partir de los datos ingresados por el usuario

3. Una vez que hemos creado los objetos, procedemos con su respectivo tamaño, procedemos a hacer los llamados a funciones. Solo para los conjuntos creados de la Figura 7 se le asignaron datos aleatorios y después imprimimos los conjuntos (Figura 8).

```
echo"<h2> Conjunto 1</h2>";
$aux1=$conjunto1->llenarArreglo();
$conjunto1->imprimirArreglo($aux1);

echo"<br><br><h2> Conjunto 2</h2>";
$aux2=$conjunto2->llenarArreglo();
$conjunto2->imprimirArreglo($aux2);
```

Figura 8. Asignación de datos e impresión de los arreglos.

4. Teniendo los dos conjuntos necesarios con datos, se procedió a realizar las operaciones. Para realizar las operaciones se presentaron algunos problemas en cuanto al tamaño de los conjuntos para las operaciones.

Por ejemplo: Tenemos el conjunto 1 con {1,2,3,5,4,3,7} y el conjunto 2 con {3,4,5,9,8,4}, para la unión tendríamos un nuevo conjunto con los datos {1,2,3,5,4,3,7,3,4,5,9,8,4}, este nuevo conjunto tiene como tamaño 13, pero en la teoría, si hay un dato que se repite simplemente no lo volvemos a poner, pues que ya se encontró al menos un dato igual, por lo que el tamaño del arreglo cambia, es decir, el conjunto ahora tiene un nuevo tamaño que es 8 y ahora se vería así {1,2,3,5,4,7,9,8}, y es lo mismo para las demás operaciones, si hay datos repetidos seria necesario quitar esos elementos.
Por lo que, antes de ingresar el tamaño para los nuevos conjuntos que tendrían asignados las operaciones, fue necesario hacer el calculo para saber el tamaño del conjunto, sin datos repetidos.

5. Como mencionamos anteriormente, hay métodos en PHP que nos ayudan a realizar operaciones de conjuntos y son estos mismos métodos que nos ayudaran a sacar el tamaño para los nuevos conjuntos. Para la unión, usamos las funciones `array_merge()` y `array_unique()`, posteriormente se usa la función `sizeof()` para saber el número de elementos que tendrá el nuevo arreglo. Para la intersección y la diferencia de conjuntos es similar la manera que sacamos el tamaño del arreglo.

```
echo"<br><br><h2> Conjunto Unión</h2>";
$tam_un = array_merge($aux1,$aux2);
$aux_tam = array_unique($tam_un);
$tam_aux = sizeof($aux_tam);
$conjunto3 = new Conjunto($tam_aux);
$aux3 = $conjunto3->union($aux1, $aux2);
$conjunto3->imprimirArreglo($aux3);


echo"<br><br><h2> Conjunto Intersección</h2>";
$tam_aux2 = array_intersect($aux1,$aux2);
$tam_inter_aux = array_unique($tam_aux2);
$tam_in = sizeof($tam_inter_aux);
$conjunto4 = new Conjunto($tam_in);
$aux4 = $conjunto4->interseccion($aux1, $aux2);
$conjunto4->imprimirArreglo($aux4);
```

Figura 9. Creación del Conjunto Unión e Intersección.

```php
echo"<br><br><h2> Conjunto Diferencia (C1 - C2)</h2>";
$tam_aux3 = array_diff($aux1,$aux2);
$aux_tam2 = array_unique($tam_aux3);
$tam_dif1 = sizeof($aux_tam2);
$conjunto5 = new Conjunto($tam_dif1);
$aux5 = $conjunto5->diferencia($aux1,$aux2);
$conjunto5->imprimirArreglo($aux5);



echo"<br><br><h2> Conjunto Diferencia (C2 - C1)</h2>";
$tam_aux4 = array_diff($aux2,$aux1);
$aux_tam3 = array_unique($tam_aux4);
$tam_dif2 = sizeof($aux_tam3);
$conjunto6 = new Conjunto($tam_dif2);
$aux6 = $conjunto6->diferencia($aux2,$aux1);
$conjunto6->imprimirArreglo($aux6);
```

Figura 10. Diferencia de conjuntos.

FUNCIONAMIENTO DEL PROGRAMA.

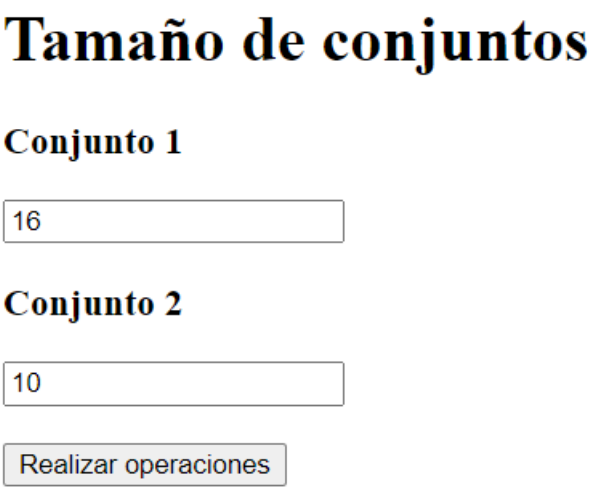


Figura 11. Datos ingresados por el usuario.

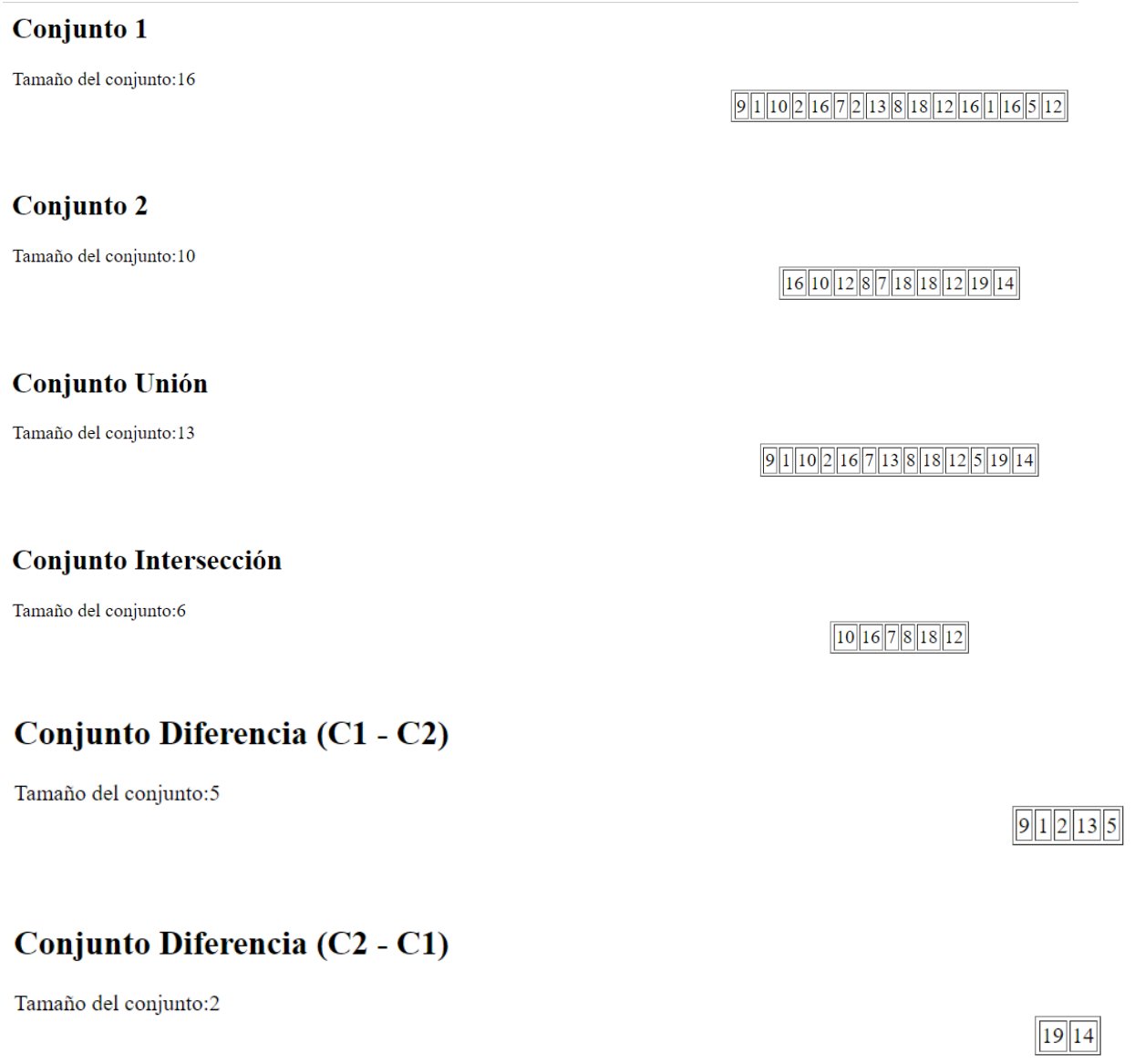


Figura 12. Resultados obtenidos.

Como observamos en la Figura 12, los arreglos obtenidos no muestran datos repetidos y cada uno tiene su propio tamaño. Las funciones de PHP nos fueron de gran utilidad y nos ayudaron a no tener un código robusto. Se pudo implementar las operaciones de conjuntos con los ciclos, pero se decidió usar los métodos de PHP para saber un poco más de este lenguaje.

REFERENCIAS

https://www.php.net/manual/es/function.array-values.php

https://www.php.net/manual/es/function.array-unique.php

https://www.php.net/manual/es/function.array-diff.php

https://www.php.net/manual/es/function.array-merge.php

https://www.php.net/manual/es/function.array-intersect.php

http://www.matematicas.ciencias.uchile.cl/juaco/section-2.html

https://lineadecodigo.com/php/generar-una-lista-de-numeros-aleatorios-en-php/

CODIGO FUENTE DEL PROGRAMA

ARCHIVO datos.php

```
<html>
<head>
    <title>Datos recibidos</title>
</head>
<body>
    <?php
        require_once "conjunto.php";
        $cn1 = $_REQUEST['conjunto1'];
        $cn2 = $_REQUEST['conjunto2'];


        $conjunto1 = new Conjunto($cn1);
        $conjunto2 = new Conjunto($cn2);

        echo"<h2> Conjunto 1</h2>";
        $aux1=$conjunto1->llenarArreglo();        //Llenado del arreglo con
numeros aleatorios
        $conjunto1->imprimirArreglo($aux1);       //Impresion del arreglo

        echo"<br><br><h2> Conjunto 2</h2>";
        $aux2=$conjunto2->llenarArreglo();      //Llenado del arreglo con
numeros aleatorios
        $conjunto2->imprimirArreglo($aux2);     //Impresion del arreglo


        echo"<br><br><h2> Conjunto Unión</h2>";
        $tam_un =
array_merge($aux1,$aux2);                                  //Acciones
necesarias para
        $aux_tam =
array_unique($tam_un);                                    //sacar el nuevo
tamaño del conjunto
        $tam_aux =
sizeof($aux_tam);                                         //union (no se
repiten datos)
        $conjunto3 = new Conjunto($tam_aux);
```

```
        $aux3 = $conjunto3->union($aux1, $aux2);
        $conjunto3->imprimirArreglo($aux3);


        echo"<br><br><h2> Conjunto Intersección</h2>";
        $tam_aux2 =
array_intersect($aux1,$aux2);                              //Acciones
necesarias para
        $tam_inter_aux =
array_unique($tam_aux2);                                 //sacar el tamaño del
conjunto interseccion
        $tam_in =
sizeof($tam_inter_aux);                                     //sin datos
repetido
        $conjunto4 = new Conjunto($tam_in);
        $aux4 = $conjunto4->interseccion($aux1, $aux2);
        $conjunto4->imprimirArreglo($aux4);



        echo"<br><br><h2> Conjunto Diferencia (C1 - C2)</h2>";
        $tam_aux3 = array_diff($aux1,$aux2);
        $aux_tam2 = array_unique($tam_aux3);
        $tam_dif1 = sizeof($aux_tam2);
        $conjunto5 = new Conjunto($tam_dif1);
        $aux5 = $conjunto5->diferencia($aux1,$aux2);
        $conjunto5->imprimirArreglo($aux5);



        echo"<br><br><h2> Conjunto Diferencia (C2 - C1)</h2>";
        $tam_aux4 = array_diff($aux2,$aux1);
        $aux_tam3 = array_unique($tam_aux4);
        $tam_dif2 = sizeof($aux_tam3);
        $conjunto6 = new Conjunto($tam_dif2);
        $aux6 = $conjunto6->diferencia($aux2,$aux1);
        $conjunto6->imprimirArreglo($aux6);


    ?>
</body>
</html>
```

ARCHIVO formulario.php

```html
<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Datos</title>
</head>
<body>
    <div class="container">
        <div class="title"><h1>Tamaño de conjuntos</h1></div>
        <form action="datos.php" method="post">
            <div class="datos-entrada">
                <div class="input-dat">
                    <label for="dato1"> <h3>Conjunto 1</h3></label>
                    <input type="text" name="conjunto1" placeholder="Ingrese
dato">
                </div>

                <div class="input-dat">
                    <label for="dato2" ><h3>Conjunto 2</h3></label>
                    <input type="text" name="conjunto2" placeholder="Ingrese
dato">
                </div>
                <br>
                <div class="button">
                    <input type="submit" value="Realizar operaciones">
                </div>
            </div>
        </form>
    </div>
</body>
</html>
```

ARCHIVO conjunto.php

```php
<?PHP
class Conjunto{
  private $tam;

  public function __construct($tam){
    $this->tam = $tam;
  }
```

```php
  public function llenarArreglo(){
    $numeros = array();
    for($i=0; $i<$this->tam;$i++){
      $numeros[$i] =  rand(1,20);
    }
    return $numeros;
  }

  public function imprimirArreglo($m){
    echo"Tamaño del conjunto:$this->tam";
    echo "<table border align = 'center'>";
    for($i=0;$i<1;$i++){
      echo "<tr>";
        for ($j=0; $j < $this->tam; $j++) {
          echo "<td>";
          echo $m[$j];
          echo "</td>";
        }
        echo "</tr>";
    }
    echo "</table>";
  }

  public function union($c1,$c2){
    $aux_union = array_merge($c1,$c2);
    $union =  array_values(array_unique($aux_union));
    return $union;
  }

  public function interseccion($c1,$c2){
    $aux_inter = array_unique(array_intersect($c1,$c2));
    $interseccion = array_values($aux_inter);
    return $interseccion;
  }

  public function diferencia($c1,$c2){
    $aux_dif = array_unique(array_diff($c1,$c2));
    $diferencia = array_values($aux_dif);

    return $diferencia;

  }

}
?>
```