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:

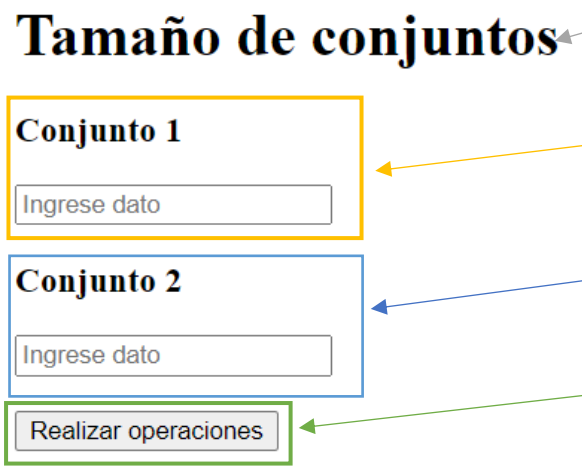
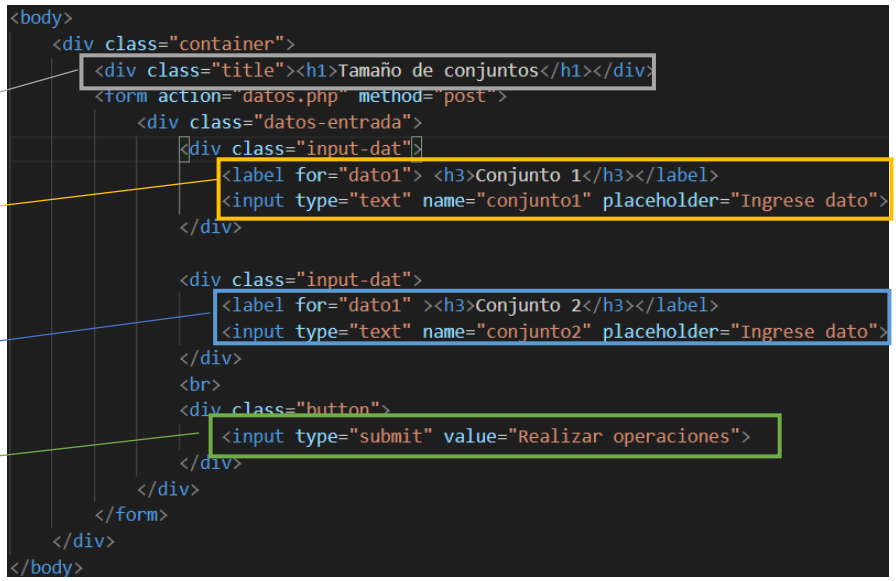
Resultado obtenido	Código implementado
	

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.

La clase conjunto se encargará de crear un nuevo Conjunto, para ello se necesita tener una variable de tipo array que vaya almacenando los conjuntos. Para crear el conjunto necesitaríamos su tamaño, es por ello que en el constructor de esta clase tendrá como parámetro el tamaño del conjunto que se creará. Dentro del constructor se puede realizar la acción de llenar el arreglo con datos aleatorios, a continuación, se muestran las líneas de código:

```
private $conjunto;

public function __construct($tam){
  $this->tam = $tam;
  for($i=0; $i<$tam; $i++){
    $this->conjunto[$i] = rand(1,20);
  }
}
```

Figura 1. Creación del arreglo con datos aleatorios.

Debido a que se crearan varios conjuntos con diferentes tamaños, es necesario saber el tamaño del arreglo, para ello, creamos un método el cual va obteniendo el numero de datos que hay en el arreglo, se usa la función *count()* para obtener el numero de datos, a continuación, se muestra su implementación:

```
public function getTam(){
  return count($this->conjunto);
}
```

Figura 2. Función para obtener el tamaño del arreglo.

Sin la función de la Figura 2, el tamaño del arreglo seria 0, por lo que, no se crearía ningún conjunto y a la hora de hacer el llamado de la función que imprima el arreglo no se vería nada.

Para visualizar los datos que se han almacenado en los arreglos es necesario tener al menos el conjunto que queremos visualizar, se puede usar un *foreach()* para imprimir el arreglo sin necesidad de tener un contador. De igual manera, se puede usar *for()* para hacer el uso de la variable que contiene el tamaño, pero solo usamos el ciclo *foreach()* para imprimir su contenido, a continuación, se muestra su implementación:

```
public function imprimirArreglo(){
  echo"Tamaño del conjunto: ".$this->getTam()."<br>";
  echo"<br>{";
    foreach($this->conjunto as $elem){
      echo $elem.", ";
    }
  echo"}";
}
```

Figura 3. Impresión del arreglo.

Una vez que tenemos los dos arreglos creamos las funciones que harán la: Unión, Intersección y Diferencia de los conjuntos. En el lenguaje de PHP hay funciones que hacen las operaciones de los conjuntos, a continuación, se muestra su implementación:

```php
public function union($c1,$c2){
  $this->conjunto = array_values(array_unique(array_merge($c1->conjunto,$c2->conjunto)));
}
```

Figura 4. Función con la operación Unión.

```php
public function interseccion($c1,$c2){
  $this->conjunto = array_values(array_unique(array_intersect($c1->conjunto,$c2->conjunto)));
}
```

Figura 5. Función con la operación Intersección.

```php
public function diferencia($c1,$c2){
  $this->conjunto = array_values(array_unique(array_diff($c1->conjunto,$c2->conjunto)));
}
```

Figura 6. Función con la operación Diferencia

Las funciones 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.

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 7).

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

Figura 7. 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 8). Después los mandamos a imprimir para visualizar los datos que tiene los arreglos.

```
echo"<h2> Conjunto 1</h2>";
$conjunto1 = new Conjunto($cn1);
$conjunto1->imprimirArreglo();

echo"<br><h2> Conjunto 2</h2>";
$conjunto2 = new Conjunto($cn2);
$conjunto2->imprimirArreglo();
```

Figura 8. Creación de conjuntos a partir de los datos ingresados por el usuario

3. Teniendo los 2 conjuntos con los que aremos las operaciones, procedemos a crear los nuevos objetos que tendrán el conjunto unión, intersección, diferencia del conjunto C1-C2 y diferencia de conjunto C2-C3 (Figura 9).

```
$conjunto3 = new Conjunto(0);    //Conjunto union
$conjunto4 = new Conjunto(0);    //Conjunto interseccion
$conjunto5 = new Conjunto(0);    //Conjunto difenrencia (c1,c2)
$conjunto6 = new Conjunto(0);    //Conjunto difenrencia (c2,c1)
```

Figura 9. Creación de los nuevos conjuntos.

Observando la Figura 9 vemos que al crear el conjunto tiene como tamaño 0, y esto es debido a que el tamaño lo obtenemos cuando se realiza la operación ya que le quitamos los elementos repetidos, lo que hace que el tamaño sea diferente al que se le pase como parámetro.

4. Una vez que tenemos los nuevos conjuntos creados, solo resta hacer el llamado a las funciones que hemos creado, a continuación, se muestra su implementación:

```
echo"<br><h2> Conjunto Unión</h2>";
$conjunto3->union($conjunto1, $conjunto2);
$conjunto3->imprimirArreglo();

echo"<br><h2> Conjunto Intersección</h2>";
$conjunto4->interseccion($conjunto1,$conjunto2);
$conjunto4->imprimirArreglo();

echo"<br><h2> Conjunto Diferencia (C1 - C2)</h2>";
$conjunto5->diferencia($conjunto1,$conjunto2);
$conjunto5->imprimirArreglo();

echo"<br><h2> Conjunto Diferencia (C2 - C1)</h2>";
$conjunto6->diferencia($conjunto2, $conjunto1);
$conjunto6->imprimirArreglo();
```

Figura 10. Llamado a los métodos.

FUNCIONAMIENTO DEL PROGRAMA.

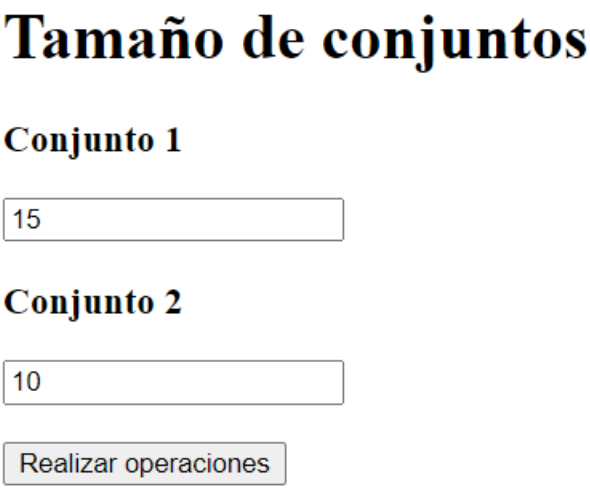


Figura 11. Datos ingresados por el usuario.

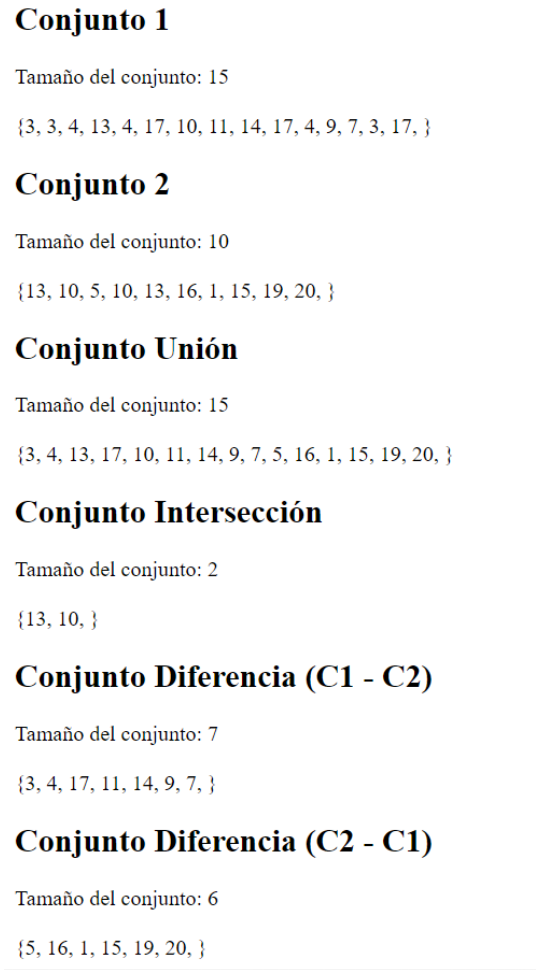


Figura 12. Resultados obtenidos.

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'];

        echo"<h2> Conjunto 1</h2>";
        $conjunto1 = new Conjunto($cn1);
        $conjunto1->imprimirArreglo();

        echo"<br><h2> Conjunto 2</h2>";
```

```
        $conjunto2 = new Conjunto($cn2);
        $conjunto2->imprimirArreglo();

        $conjunto3 = new Conjunto(0);   //Conjunto union
        $conjunto4 = new Conjunto(0);   //Conjunto interseccion
        $conjunto5 = new Conjunto(0);   //Conjunto difenrencia (c1,c2)
        $conjunto6 = new Conjunto(0);   //Conjunto difenrencia (c2,c1)

        echo"<br><h2> Conjunto Unión</h2>";
        $conjunto3->union($conjunto1, $conjunto2);
        $conjunto3->imprimirArreglo();

        echo"<br><h2> Conjunto Intersección</h2>";
        $conjunto4->interseccion($conjunto1,$conjunto2);
        $conjunto4->imprimirArreglo();

        echo"<br><h2> Conjunto Diferencia (C1 - C2)</h2>";
        $conjunto5->diferencia($conjunto1,$conjunto2);
        $conjunto5->imprimirArreglo();

        echo"<br><h2> Conjunto Diferencia (C2 - C1)</h2>";
        $conjunto6->diferencia($conjunto2, $conjunto1);
        $conjunto6->imprimirArreglo();


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

ARCHIVO formulario.php

```
<!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="titulo"> <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="dato1"> <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
class Conjunto{

  private $conjunto;

  public function __construct($tam){
    $this->tam = $tam;
    for($i=0; $i<$tam; $i++){
      $this->conjunto[$i] = rand(1,20);
    }
  }

  public function imprimirArreglo(){
    echo"Tamaño del conjunto: ".$this->getTam()."<br>";
    echo"<br>{";
      foreach($this->conjunto as $elem){
        echo $elem.", ";
      }
    echo"}";
  }

  public function getTam(){
    return count($this->conjunto);
  }

  public function union($c1,$c2){
```

```
    $this->conjunto = array_values(array_unique(array_merge($c1-
>conjunto,$c2->conjunto)));
  }


  public function interseccion($c1,$c2){
    $this->conjunto = array_values(array_unique(array_intersect($c1-
>conjunto,$c2->conjunto)));
  }


  public function diferencia($c1,$c2){
    $this->conjunto = array_values(array_unique(array_diff($c1-
>conjunto,$c2->conjunto)));
  }


}
?>
```