Informe de Laboratorio 08

Tema: Expresiones Regulares en Perl

Estudiante	Escuela	Asignatura
Luis Gustavo Sequeiros Condori lsequeiros@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web I Semestre: II

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 27 Diciembre 2023	Al 30 Diciembre 2023

1. Temas a Tratar:

- Expresiones regulares en Perl.
- CGI en Perl.

2. Actividades

2.1. Archivo *calculator.html*

Listing 1: Esta es la estructura de una calculadora simple

```html
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Calculadora</title>
  </head>
  <body>
    <div class="site-wrapper">
      <div class="title">
        <span>Calculadora</span>
      </div>
      <div class="subtitle">
        <span>Ingrese su operación:</span>
      </div>
      <div>
        <form action="cgi-bin/calcular.pl" method="POST">
          <input type="text" name="exp" class="solve-input" required>
          <input type="submit" value="Calcular" class="solve-button">
        </FORm>
      </div>
    </div>

  </body>
</html>
```

Se observa un contenedor general *site-wrapper*, dentro están los elementos como el él título, un pequeño texto guía y el formulario para ingresar la operación. El formulario se conecta con el archivo *calcular.pl*, el cgi que se encarga de las operaciones. Detro del formulario no hay más que una extrada de texto y un botón de solución.

2.2. Archivo *styles.css*

Listing 2: Hoja de estilos general

```
* {
  margin: 0;
  box-sizing: border-box;
}
body {
  font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}
a{
  text-decoration: none;
  color: white;
  width: 100%;
  height: 100%;
}
:root{
  --gray: rgb(180,200,255);
}
.site-wrapper {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
  flex-direction: column;
  row-gap: 2vh;
  background-color: aquamarine;
}
.title {
  font-size: 200%;
  padding: 1vh;
  background-color:dodgerblue;
  color: white;
  border-radius: 10px;
}
.subtitle {
  font-family:'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
}
form {
  display: flex;
  flex-direction: column;
  row-gap: 3vh;
  width: 40vw;
  align-items: center;
}
.solve-button {
  background-color:black;
  color: white;
  border: none;
  padding: 1% 3%;
  border-radius: 50em;
  font-size: 120%;
  cursor: pointer;
}
.solve-input {
  width: 50vw;
  height: 35px;
```

```
55    border: solid var(--gray) 3px;
56    border-radius: 50em;
57    background-color: var(--gray);
58    padding: 0 15px;
59    font-size: 120%;
60    display: flex;
61    align-items: center;
62  }
63  .resolution-container {
64    display: flex;
65    align-items: center;
66    background-color: var(--gray);
67    border-radius: 10px;
68    flex-direction: column;
69    padding: 3vh 5%;
70    row-gap: 20px;
71  }
72  .resolution {
73    display: flex;
74    align-items: center;
75    flex-direction: column;
76    row-gap: 5px;
77    font-size: 1.3em;
78  }
79  .back-button {
80    background-color: black;
81    text-align: center;
82    border: none;
83    padding: 2% 6%;
84    border-radius: 50em;
85    font-size: 120%;
86  }
```

Se observa unos estilos simples con dos fuentes para toda la página. Se normalizan algunos elementos y se crea una variable *gray*. Asimismo, se desarrolla un diseño simple para el contenedor raíz del sitio, a los contenedores internos se les concede algunos colores agradables. Al final de la hoja de estilos se encuentra el diseño para el cgi, es casi lo mismo que el diseño del html, pero con diferentes nombres de contenedores y una que otra diferencia para los estilos de letra.

2.3. Archivo *calcular.pl*

Listing 3: CGI para el cálculo

```
1   #!/usr/bin/perl
2   use strict;
3   use warnings;
4   use CGI;
5
6   sub solveMul {
7     my $ec = $_[0];
8     my $mult;
9     if($ec =~ m/(.*?)\*(.*)/){
10      $mult = $1 * $2;
11    }
12    if($mult > 0){
13      $mult = "+".$mult;
14    }
15    return $mult;
16  }
17  sub solveDiv {
18    my $ec = $_[0];
```

```
  my $div;
  if($ec =~ m/(.*?)\/(.*)/){
    $div = $1 / $2;
  }
  if($div > 0){
    $div = "+".$div;
  }
  return $div;
}
sub solveExp {
  my $ec = $_[0];
  my $sol;
  if($ec =~ m/(.*?)\((.+)\)(.*?)(\(.+\))(.*)/){
    my @vars = ($1,$2,$3,$4,$5);
    #print "dobPar: $vars[0],$vars[1],$vars[2],$vars[3],$vars[4]\n";
    $sol = solveExp($vars[1]);
    $ec = $vars[0].$sol.$vars[2].$vars[3].$vars[4];
    #print "solvedobPar: ".$ec."\n";
    return solveExp($ec);
  }elsif($ec =~ m/^\((.+)\)$/){
    my $ecu = $1;
    return solveExp($ecu);
  }elsif($ec =~ m/(.*?)\((.+)\)(.*)/){
    my @vars = ($1,$2,$3);
    #print "nesPar: $vars[0],$vars[1],$vars[2]\n";
    $sol = solveExp($vars[1]);
    $ec = $vars[0].$sol.$vars[2];
    #print "solvenesPar: ".$ec."\n";
    return solveExp($ec);
  }else{
    my $solv;
    # print "Init without par\n";
    while($ec =~ m/(.*?)([\-\+]?[0-9]+(\.[0-9]+)?\*[\-\+]?[0-9]+(\.[0-9]+)?)/){
      $solv = solveMul($2);
      $ec = $1.$solv.$';
      #print "Mult: ".$ec."\n";
    }
    while($ec =~ m/(.*?)([\-\+]?[0-9]+(\.[0-9]+)?\/[\-\+]?[0-9]+(\.[0-9]+)?)/){
      $solv = solveDiv($2);
      $ec = $1.$solv.$';
      #print "Div: ".$ec."\n";
    }
    $ec =~ s/-/+-/g;
    my @mem = split('\+', $ec);
    my $sum = 0;
    for my $num(@mem){
      $sum = $sum + 0 + $num;
      #print "$sum\n";
    }
    $ec = $sum;
    #print "Final sum: $ec\n";
    return $ec;
  }
}
sub secureExp {
  my $ec = $_[0];
  $ec =~ s/ //g;
  if($ec =~ /[a-z]|[A-Z]/){
    return "No es una expresión válida\n";
  }
  return solveExp($ec);
}
my $q = CGI->new;
$q->charset('UTF-8');
my $exp = $q->param('exp');
```

```
84
85  my $result = secureExp($exp);
86
87  print $q->header('text/html');
88  print<<BLOCK
89  <!DOCTYPE html>
90  <html lang="es">
91    <head>
92      <meta charset="UTF-8">
93      <meta name="viewport" content="width=device-width, initial-scale=1.0">
94      <link rel="stylesheet" href="../styles.css">
95      <title>Calculadora</title>
96    </head>
97    <body>
98      <div class="site-wrapper">
99        <div class="title">
100         <a href="../calculator.html">Calculadora</a>
101       </div>
102       <div class="resolution-container">
103         <div class="resolution">
104           <p><b>La expresión es:</b></p>
105           <p>$exp</p>
106           <p><b>La respuesta es:</b></p>
107           <p>$result</p>
108         </div>
109         <div class="back-button">
110           <a href="../calculator.html">Volver</a>
111         </div>
112       </div>
113     </div>
114
115   </body>
116 </html>
117 BLOCK
```

Este CGI usa expresiones regulares para la resolución de operaciones simples. La subrutina principal de la que se vale este script es *solveExp*, esta identifica primeramente expresiones de paréntesis que pueden ser de tres tipos: ()(),(()), (allthecontent); la subrutina identifica estas expresiones y se llama recursivamente para resolverlas. Esto lo hace hasta no identificar paréntesis, luego, resuelve la expresión sin paréntesis y devuelve un solo número como resultado, la pila se queda sin elementos hasta que quede un solo número.

2.4. Recursos

- Video sobre el funcionamiento de la página `https://youtu.be/wo17Kdz0Ov4`.
- Repositorio de GitHub para commits `https://github.com/gusCreator/pweb-course/tree/main/lab08`.