




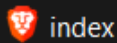


INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	PROGRAMACIÓN WEB 1				
TÍTULO DE LA PRÁCTICA:	Laboratorio 05 : Calculadora con expresiones regulares en Perl				
NÚMERO DE PRÁCTICA:	05	AÑO LECTIVO:	2024 - B	NRO. SEMESTRE:	II
FECHA DE PRESENTACIÓN	27/10/2024	HORA DE PRESENTACIÓN	23:59:00		
INTEGRANTE (s):  Layme Salas Rodrigo Fabricio (100%)				NOTA:	
DOCENTE: M.Sc. Richart Escobedo Quispe					

SOLUCIÓN Y RESULTADOS			
I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS			
Archivos usados:			
Utilice una estructura simple para organizar mis archivos, evité usar un archivo más de css haciendolo de forma incrustada			
<div><div><div> cgi-bin</div><div>10/19/2024 9:56 AM</div><div>Carpeta de archivos</div></div><div><div> html</div><div>10/19/2024 9:55 AM</div><div>Carpeta de archivos</div></div><div><div> Dockerfile</div><div>10/19/2024 10:10 AM</div><div>Archivo</div><div>3 KB</div></div><div><div> README</div><div>10/19/2024 10:18 AM</div><div>Text Document</div><div>1 KB</div></div></div>			
cgi-bin: Es solo uno, devuelve un html que puede ser reutilizado para una siguiente expresión			
<div><div> script</div><div>10/21/2024 1:47 PM</div><div>Archivo de origen ...</div><div>4 KB</div></div>			

*html: Solo es 1, además, pronto se cambiará por el html que trae perl*



index

10/19/2024 8:20 PM

Brave HTML Docu...

1 KB

*Dockerfile: Utilicé el mismo Dockerfile con el que hemos estado trabajando*

```
FROM bitnami/minideb
ENV DEBIAN_FRONTEND="noninteractive"
RUN apt-get update && \
    apt-get install -y apache2 perl openssl-server vim bash locales tree libcgi-pm-perl dos2unix && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
RUN echo -e 'LANG=es_PE.UTF-8\nLC_ALL=es_PE.UTF-8' > /etc/default/locale && \
    sed -i 's/^# *(es_PE.UTF-8)/\1/' /etc/locale.gen && \
    /sbin/locale-gen es_PE.UTF-8
RUN mkdir -p /home/pweb && \
    useradd pweb -m && echo "pweb:12345678" | chpasswd && \
    echo "root:12345678" | chpasswd && \
    chown pweb:www-data /usr/lib/cgi-bin/ && \
    chown pweb:www-data /var/www/html/ && \
    chmod 750 /usr/lib/cgi-bin/ && \
    chmod 750 /var/www/html/
RUN echo "export LC_ALL=es_PE.UTF-8" >> /home/pweb/.bashrc && \
    echo "export LANG=es_PE.UTF-8" >> /home/pweb/.bashrc && \
    echo "export LANGUAGE=es_PE.UTF-8" >> /home/pweb/.bashrc
RUN ln -s /usr/lib/cgi-bin /home/pweb/cgi-bin && \
    ln -s /var/www/html/ /home/pweb/html && \
    ln -s /home/pweb /usr/lib/cgi-bin/toHOME && \
    ln -s /home/pweb /var/www/html/toHOME
COPY ./cgi-bin/ /usr/lib/cgi-bin/
RUN dos2unix /usr/lib/cgi-bin/script.pl && \
    chmod +x /usr/lib/cgi-bin/script.pl
COPY /html/ /var/www/html/
RUN echo '<VirtualHost *:80>\n\
    ServerAdmin webmaster@localhost\n\
    DocumentRoot /var/www/html\n\
    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/\n\
    <Directory "/usr/lib/cgi-bin">\n\
        AllowOverride None\n\
        Options +ExecCGI\n\
        Require all granted\n\
    </Directory>\n\
    ErrorLog ${APACHE_LOG_DIR}/error.log\n\
    CustomLog ${APACHE_LOG_DIR}/access.log combined\n\
</VirtualHost>' > /etc/apache2/sites-available/000-default.conf
RUN a2enmod cgi
RUN sed 's@session@s*required@s*pam_loginuid.so@session optional pam_loginuid.so@g' -i /etc/pam.d/sshd
EXPOSE 80
EXPOSE 22
CMD ["bash", "-c", "service ssh start && apachectl -D FOREGROUND"]
```

**Html:**

*Use un index para la presentación del formulario en el que se puede colocar la expresión. Con el css incrustado le di un buen acabado. Lo más resaltante es la etiqueta form en la línea 62, que me permite capturar datos que se ingresen con el método get.*

```
43     border-radius: 5px;
44     font-size: 18px;
45     cursor: pointer;
46 }
47 .footer {
48     padding: 10px;
49     margin: 0px;
50     left: 0;
51     bottom: 0;
52     width: 100%;
53     background-color: #f2f2f2;
54     color: #337ab7;
55     text-align: center;
56     font-size: 14px;
57 }
58 </style>
59 </head>
60 <body>
61 <div class="formulario">
62 <form action="../../cgi-bin/script.pl" method="get">
63 <h2>Ingrese alguna operación para calcularla:</h2>
64 <input type="text" name="expresion" placeholder="Ingrese alguna expresión aquí...">
65 <input type="submit" value="¡Calcular!">
66 </form>
67 <div class="footer">
68     Layme Salas, Rodrigo Fabricio &copy; 2024/10/26 - Programación Web Lab. Grupo D.
69 </div>
70 </div>
71 </body>
72 </html>
```

### Script Perl:

Para mi archivo Perl solo usé expresiones regulares, sin eval.

En este primer bloque usé unos controladores de errores y declaro mi objeto CGI para poder captar el parámetro en el formulario.

```
1  #!/usr/bin/perl -w
2  use strict;
3  use warnings;
4  use CGI;
5  my $q = CGI->new;
6  my $expresion = $q->param('expresion'); # Con el objeto CGI se guarda la expresión ingresada en el formulario
```

En este segundo bloque inicializo mi respuesta con un String vacío para evitar errores de terminal y luego reemplazarlo por el resultado que salga después de las operaciones con la función `resolver_expresion()`.

```
62 my $resultado = ''; # Inicializa el resultado pero vacío
63 if ($expresion) {
64     $resultado = resolver_expresion($expresion);
65 }
```

En este tercer bloque está la función base de todo el programa, mencionó usted que podíamos usar recursividad para esta actividad. Lo dividiré más para mejorar la explicación en las siguientes imágenes.

```

7  sub resolver_expresion {
8      my ($exp) = @_ ;
9      $exp =~ s/\s+//g; # Reemplaza los espacios ingresados por teclado por nada
10     while ($exp =~ /\(\([^\)]+\)\)/) { # Encuentra los paréntesis de la operación, los resuel
        resultado hasta que no quede ninguno
11         my $sub_expresion = $1;
12         my $resultado_sub = resolver_expresion($sub_expresion);
13         return 'error de indeterminacion' if $resultado_sub eq 'error de indeterminacion';
        división entre 0
14         $exp =~ s/\(\([^\)]+\)\)/$resultado_sub/;
15     }
16     my @expresion = split /(?!=[\d])([+|-*\/])|(?!=[-+])/, $exp; # Para el procesamiento de
        expresion en números y operadores
17     @expresion = multiplicacionYdivision(@expresion);
18     if (@expresion == 1 && $expresion[0] eq 'error de indeterminacion') {
19         return 'error de indeterminacion';
20     }
21     my $resultado = sumaYresta(@expresion);
22     return $resultado;
23 }

```

En esta parte de la función anterior se capta el parámetro en la Ln 8 y en la Ln 9 se elimina de forma global todos los espacios introducidos reemplazándolos por nada.

- Para permitir el uso de paréntesis correctamente, usar un bucle que seguirá hasta que no haya otro paréntesis dentro de la expresión.
- La Ln 11 almacena los primeros paréntesis encontrados en otra variable para trabajarla por separado
- La Ln 12 es la parte recursiva, con la nueva variable se vuelve a evaluar si hay otros paréntesis dentro.
- Si no existe otros paréntesis se continúa con el resto de la función (lo explico en la siguiente imagen)
- La Ln 13 verifica que no haya un error de indeterminación y la Ln 14 reemplaza la expresión entre paréntesis por su resultado si no hubo indeterminación.

```

8      my ($exp) = @_ ;
9      $exp =~ s/\s+//g; # Reemplaza los espacios ingresados por teclado por nada
10     while ($exp =~ /\(\([^\)]+\)\)/) { # Encuentra los paréntesis de la operación, los resuel
        resultado hasta que no quede ninguno
11         my $sub_exp = $1;
12         my $resultado_sub = resolver_expresion($sub_exp);
13         return 'error de indeterminacion' if $resultado_sub eq 'error de indeterminacion';
        división entre 0
14         $exp =~ s/\(\([^\)]+\)\)/$resultado_sub/;
15     }

```

En esta otra parte, la Ln 16 separa ya la expresión sin paréntesis dentro de un array, siendo cada elemento un número o un operador.

- La Ln 17 llama a otra función para poder hacer las operaciones de multiplicación y división pasando como parámetro el array. (Aquí se verán los errores de indeterminación para acabar con el bucle).
- La Ln 21 después de verificar que no haya indeterminación, continúa con la función de suma y resta donde no habrá posibles errores para luego retornar el valor sin paréntesis.

```

16 my @expresion = split /(?!=[\d])([+\\-*\\/])(?!=[-+])/, $exp; # Para el proces
    expresion en números y operadores
17 @expresion = multiplicacionYdivision(@expresion);
18 if (@expresion == 1 && $expresion[0] eq 'error de indeterminacion') {
19     return 'error de indeterminacion';
20 }
21 my $resultado = sumaYresta(@expresion);
22 return $resultado;
23 }

```

Para este cuarto bloque, está la siguiente función, que trabajará con el array creado de operadores y números.

- En la Ln 26 se crea un nuevo array que almacenará los valores operados ya que se eliminarán mientras se desarrolla el bucle con shift.
- El condicional modificará el nuevo array si es un operador el valor evaluado, sino, lo agrega al nuevo array sin cambios.
- Con números en el nuevo array y siendo un operador evaluado se trae el último número almacenado en el nuevo array (el operando derecho) y el primer número del viejo array (el operando izquierdo) para luego operarlos con el operador.
- Además se evalúa si genera una indeterminación en la Ln 33, si no lo es se agrega al nuevo array la multiplicación o la división y lo retorna al array inicial.

```

24 sub multiplicacionYdivision {
25     my @expresion = @_;
26     my @nueva_expresion;
27     while (@expresion) {
28         my $elemento = shift @expresion;
29         if ($elemento eq '*' || $elemento eq '/') { # Se pregunta
30             my $anterior = pop @nueva_expresion;
31             my $siguiente = shift @expresion;
32             if ($elemento eq '/' && $siguiente == 0) { # Verifica
33                 return ('error de indeterminacion'); # Devuelve el
34             }
35             if ($elemento eq '*') { # Evalua si el operador es mul
36                 push @nueva_expresion, $anterior * $siguiente;
37             } elsif ($elemento eq '/') { # Evalua si el operador e
38                 push @nueva_expresion, $anterior / $siguiente;
39             }
40         } else {
41             push @nueva_expresion, $elemento;
42         }
43     }
44     return @nueva_expresion;
45 }

```

En este quinto bloque está la función sumaYresta, que es la más sencilla ya que no presentaré indeterminaciones.

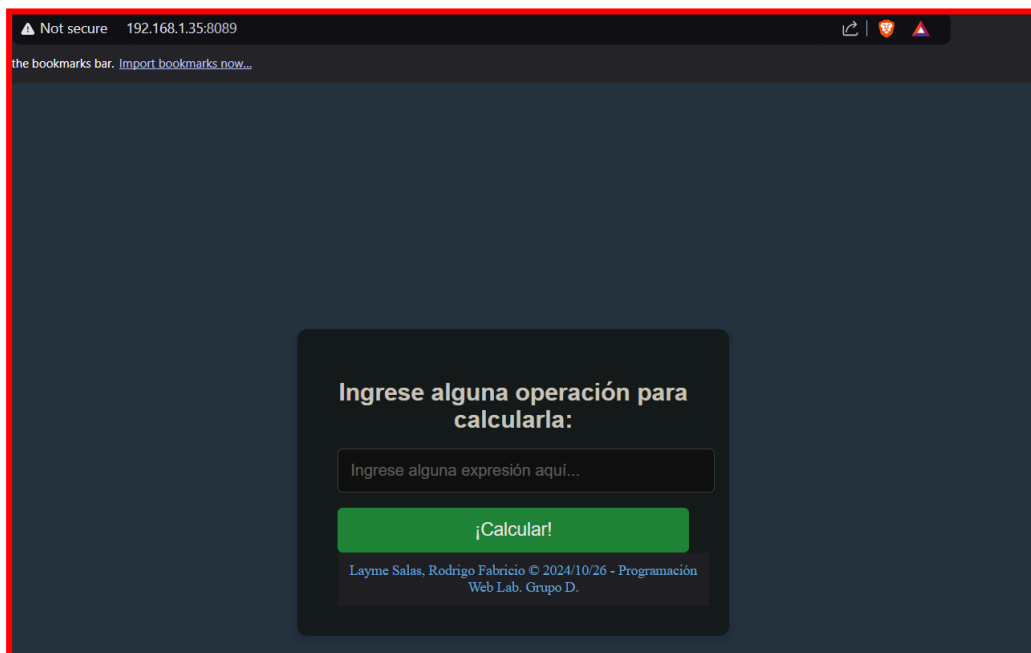
- En la Ln 48 se guarda y elimina el primer operando, luego el operador y finalmente el segundo operando.
- Los condicionales evalúan al operador y realiza la operación para luego retornarla.

```
46  sub sumaYresta { # Por orden en operaciones se
47      my @expresion = @_ ;
48      my $resultado = shift @expresion;
49
50      while (@expresion) {
51          my $operador = shift @expresion;
52          my $siguiente = shift @expresion;
53
54          if ($operador eq '+') { # Evalua si el
55              $resultado += $siguiente;
56          } elsif ($operador eq '-') { # Evalua
57              $resultado -= $siguiente;
58          }
59      }
60      return $resultado;
61  }
```

*La parte del programa netamente perl acaba ahí, terminado con la recursión, sin paréntesis y sin elementos en el array y el resultado almacenado en una variable para luego mostrarla en la parte de html.*

### **Pruebas:**

*Este es el primer vistazo a la página web, como se pide, funciona con expresiones regulares y solo hay un botón de calcular*



**Sumas y Restas:** Con estas operaciones no hay problemas, tampoco si están con paréntesis.

Ingrese alguna operación para calcularla:

$(4+5-(6-2))$

¡Calcular!

Layme Salas, Rodrigo Fabricio © 2024/10/26 - Programación Web Lab. Grupo D.

Ingrese alguna operacion para calcularla:

$(4+5-(6-2))$

Calcular

Resultado:

5

Layme Salas, Rodrigo Fabricio © 2024/10/26 - Programacion Web Lab. Grupo D.

**Multiplicación y división:** Aquí se agrega el error por indeterminación, también funciona con paréntesis normales o anidados. Además, respeta la ley de signos.

Ingrese alguna operacion para calcularla:

$-5*-5$

Calcular

Resultado:

5

Layme Salas, Rodrigo Fabricio © 2024/10/26 - Programacion Web Lab. Grupo D.

Ingrese alguna operacion para calcularla:

$-5*-5$

Calcular

Resultado:

25

Layme Salas, Rodrigo Fabricio © 2024/10/26 - Programacion Web Lab. Grupo D.

Aquí se hace uso del manejo de error con un mensaje diferente para evitar el error 500 de perl.

Ingrese alguna operacion para calcularla:

$3*(7+2)/0$

Calcular

Resultado:

25

Layme Salas, Rodrigo Fabricio © 2024/10/26 - Programacion Web Lab. Grupo D.

Ingrese alguna operacion para calcularla:

$3*(7+2)/0$

Calcular

Resultado:

error de indeterminacion

Layme Salas, Rodrigo Fabricio © 2024/10/26 - Programacion Web Lab. Grupo D.

Ingrese alguna operacion para calcularla:

$(2*(5-5)+(2*(6+7)))$

Calcular

Resultado:

error de indeterminacion

Layme Salas, Rodrigo Fabricio © 2024/10/26 - Programacion Web Lab. Grupo D.

Ingrese alguna operacion para calcularla:

$(2*(5-5)+(2*(6+7)))$

Calcular

Resultado:

26

Layme Salas, Rodrigo Fabricio © 2024/10/26 - Programacion Web Lab. Grupo D.

*Las operaciones ingresadas se vuelven a ver dentro del cuadro como placeholder. ESO ES TODO EL LABORATORIO, en el readme están los comandos para crear la imagen con el puerto que me corresponde.*

**RÚBRICA DE CALIFICACIÓN**

**(SIGUIENTE PÁGINA)**



ÍTEM	DESCRIPCIÓN	EXCELENTE	PROCESO	DEFICIENTE
<b>Código fuente</b>	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	4		
<b>Ejecución</b>	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente hasta llegar al código final del requerimiento del laboratorio.	4		
<b>Pregunta</b>	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación). Si no se le entregó pregunta, usted recopile información relevante para el laboratorio desde diferentes medios, <u>referenciándola correctamente</u> (máximo 2 caras).	4		
<b>Ortografía</b>	El documento no muestra errores ortográficos.	4		
<b>Madurez</b>	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).		2	
	<b>CALIFICACIÓN</b>	<b>16</b>	<b>2</b>	
	<b>TOTAL</b>	<b>18</b>		

## II. SOLUCIÓN DEL CUESTIONARIO

*Esta práctica no tuvo cuestionario.*

## III. CONCLUSIONES

*Perl me parece un lenguaje, sinceramente no tan bueno como otros, pero merece la pena aprender solo sus fuertes, procesamiento de textos y archivos, y cgi. Aprendí bastante de la sintaxis para procesar textos y es la mejor de todos los lenguajes que he probado.*

### RETROALIMENTACIÓN GENERAL

Si tengo que enfrentarme a un nuevo lenguaje como este debo de no enfocarme en entender su sintaxis en general, por que es casi igual a la de C y Java, sino en sus puntos fuertes o en lo que hace diferente al lenguaje.

### REFERENCIAS Y BIBLIOGRAFÍA

Wall, L., Christiansen, T., & Orwant, J. (2000). *Programming perl*. "O'Reilly Media, Inc."

Wall, L., & others. (1994). *The Perl programming language*. Prentice Hall Software Series.