UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

FACULTAD DE INGENIERÍA, PRODUCCIÓN

Y SERVICIOS

INGENIERIA DE SISTEMAS

INFORME

CURSO: Introducción a la Computación

DOCENTE: Richard Smith Escobedo Quispe

GRUPO: 3

INTEGRANTES:

- Fernando Jesús Coyla Alvarez
- Carlos Daniel Umasi Cariapaza
- Diego Alonso Huamani Luque
- Jorge Gabriel Llerena Huanca
- Edson Joel López Quispe

FECHA DE ELABORACION: 30 de julio del 2021

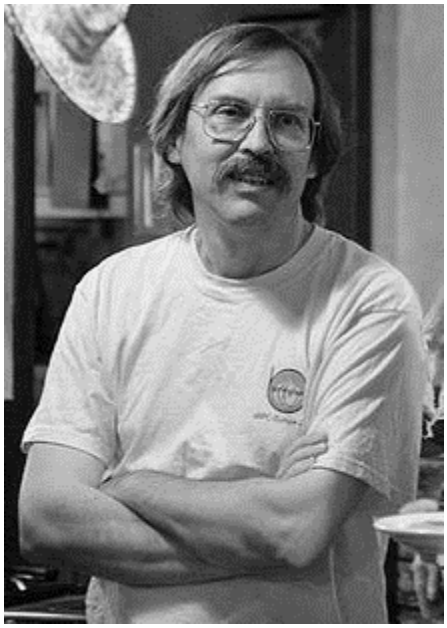
AREQUIPA

2021

EPIGRAFE

“Las nuevas versiones suceden”

Larry Wall



MOTIVACION

Estos conocimientos son buenos para todos, tanto para los programadores del futuro y para los programadores del presente, un lenguaje de programación tan innovador e interesante que nos abre la puerta del camino de un conocimiento emprendedor o también un lenguaje que lo podemos relacionar mucho con el famoso lenguaje conocido como C++, lenguaje usado por muchas empresas y conocedores de la programación, con esto buscamos que las personas aprendan del Perl ya que nos puede y nos ayudara a resolver muchos problemas o conflictos en la programación.

RESUMEN

A continuación, veremos el origen del lenguaje de programación, Perl, lenguaje de mucha ayuda y eficiencia, ejemplos sobre su aplicación en ejercicios básicos, intermedios, avanzados, también para explicar el funcionamiento de cada línea de código.

EXPECTATIVAS

Nuestras expectativas sobre el grupo y la investigación serán totalmente efectivas, donde cada integrante cumplirá con el rol asignado, en las reuniones que daremos en un futuro, si se presentan errores, en equipo serán solucionados así dando una participación efectiva de todos los integrantes.

ÍNDICE

I. INTRODUCCIÓN:

Larry Wall creador de Perl lo comenzó a trabajar en 1987 mientras trabajaba como programador en Unisys y anunció la versión 1.0 en un grupo de noticias comp. sources. misc el 18 de diciembre de 1987. El lenguaje se expandió rápidamente en los siguientes años. Después Perl 2, publicado en 1988, aportó un mejor motor de expresiones regulares. Y Perl 3, publicado en 1989, añadió soporte para datos binarios.

En 1991 debido a lo simple que era su documentación se publicó Programming Perl (el libro del camello) y se convirtió en la referencia de facto del lenguaje. Al mismo tiempo, el número de versión de Perl saltó a 4, no por marcar un gran cambio en el lenguaje, sino por identificar a la versión que estaba documentada en el libro. La versión 5 estable no apareció hasta el 17 octubre de 1994, y ha sido tan popular que todavía se usa. Fue casi una completa reescritura del intérprete y añadió muchas nuevas características al lenguaje, incluyendo objetos, referencias, paquetes y módulos.[1]

Perl tiene características que soportan varios paradigmas de programación, como pueden ser la imperativa, funcional y la orientada a objetos. Y Al mismo tiempo, el lenguaje Perl no nos obliga a seguir ningún paradigma en particular, ni obliga al programador a elegir alguna de ellas. No obstante, esta característica es solo accesible desde la versión 5.0. [2]

Para sus aplicaciones algunas de las más resaltantes serian [3]:

Empleado para la escritura de guiones de tipo CGI.

Crear plataformas web, así como para su debido desarrollo, como por ejemplo se encuentran Amazon, Ticket Master y otros.

Empleado para la liga de sistemas que no son creados con un objetivo específico, por lo que realizan una transformación de datos.

Uso constante en las administraciones de sistema.

En el área de bioinformática que también abarca las actividades financieras, ya que el mismo exhibe una velocidad de desarrollo alta, por lo que puede ser usado en aplicaciones con facilidad.

Manejar una alta cantidad de datos.

Como que también el lenguaje de programación Perl posee grandes características nombrando algunas de las ventajas de este serian [4]:

Es un lenguaje de alto nivel, así que su curva de aprendizaje es suave.

Es eficiente a la hora de tratar un gran volumen de datos. Un ejemplo es que se usa en el mercado de finanzas y bioinformática.

Es de propósito general: puede usarse en desarrollo web, pero también en otros entornos como la administración de sistemas.

Es multiplataforma, además, viene con la propia instalación de cualquier sistema operativo Linux/Unix.

Es una buena alternativa a C, ya que no tienes que trabajar con punteros.

Pero también tendría algunas desventajas [4]:

Aunque es un lenguaje interpretado, un programa hecho en Perl se compila al principio de su ejecución, por lo que puede ser lento comparado con otros lenguajes similares.

Su código no es muy legible, comparado con otros lenguajes como Python.

No tiene control de excepciones, y los posibles errores suelen tener una dificultad media encontrarlos.

La mayoría de las tareas solo requieren conocer un pequeño subconjunto del lenguaje Perl. Por lo que se podría decir que la curva de aprendizaje de Perl tiene

una pendiente suave (es fácil de aprender) y larga (si desea seguir profundizando, puede aprender a hacer muchas cosas).[5]

II. EJEMPLOS

A. Hola mundo

Para realizar el "hola mundo" en Perl, simplemente se llama al método print y luego se pone lo que se quiere imprimir, con punto y coma al finalizar. El uso de la directiva strict es opcional y sirve para generar errores al programa en caso de que se encuentre lo que pueda considerar "programación insegura"

holamundo.pl

```
1  #!/usr/local/bin/perl -w
2  use strict;
3
4  print "Hola mundo";
```

```
D:\Dwimperl\perl\bin\perl.exe holamundo.pl
Hola mundoPresione una tecla para continuar . . .
```

B. Variables

Para las variables no se declara el tipo de dato (int, double, etc.). Por ejemplo, para declarar un dato escalar se pone el símbolo $, para un arreglo se pone el signo @, etc. Cabe resaltar que no hay barewords para true o false, solo las siguientes sentencias son falsas: 0, '0', undef, '', ( ), ('').Cualquier otra será verdadera

variables.pl

```
1
2  use strict;
3  #Creando un real
4  my $qw=425.26;
5  #Creando un string
6  my $nombre="Juan Perez";
7  #Creando un entero
8  my $entero=9;
9  #Creando un arreglo
10 my @arreglo=(2,5,6,9);
11 print $entero." ".$nombre;
```

```
D:\Dwimperl\perl\bin\perl.exe variables.pl
9 Juan PerezPresione una tecla para continuar . . .
```

C. Estructuras de control

Las estructuras de control son un conjunto de instrucciones que permiten controlar el desarrollo de un programa, permiten ejecutar un conjunto de instrucciones cuando se verifica una condición o ejecutar iterativamente un bloque de instrucciones mientras una expresión sea válida.

1.-IF-ELSE

estructuras de control.pl

```
1
2  use strict;
3  #La sintaxis del if-else es muy parecida a otros lenguajes
4  #EJEMPLO
5  print "Introduzca un numero del cero al dos y pulse Enter:\n";
6  my $var  =  <STDIN>;
7  if ($var == 0) {
8     print "Esta es la opcion 0\n";
9  }
10 elsif ($var == 1) {
11    print "Esta es la opcion 1\n";
12 }
13 elsif ($var == 2) {
14    print "Esta es la opcion 2\n";
15 }
16 else {
17    print "No existe al opción tecleada\n";
18 }
19
20
```

```
D:\Dwimperl\perl\bin\perl.exe  estructuras de control.pl
Introduzca un numero del cero al dos y pulse Enter:
2
Esta es la opcion 2
Presione una tecla para continuar . . .
```

2.-WHILE

estructuras de control.pl

```
1
2  use strict;
3  #La instrucción while ejecuta iterativamente un bloque de instrucciones
4  # mientras una expresión sea válida, evaluando la comprobación en cada iteración
5  #La sintaxis del while es muy parecida a otros lenguajes
6  print "Teclea \"1\" para salir:\n";
7  my $rpta = 0;
8  while ($rpta !=1) {
9     $rpta = <STDIN> ;
10    print "Has escrito $rpta\n";
11 }
12 print "Salida.\n"
13
14
```

```
D:\Dwimperl\perl\bin\perl.exe estructuras de control.pl
Teclea "1" para salir:
2
Has escrito 2

4
Has escrito 4

1
Has escrito 1

Salida.
Presione una tecla para continuar . . .
```

3.-FOR

estructuras de control.pl

```
1
2  use strict;
3  #La instrucción for permite ejecutar iterativamente
4  #un conjunto de instrucciones. La sintaxis de la instrucción for es:
5  #for (inicial_exp; test_exp; incremento_exp) {
6      #instrucción o bloque de intrucciones;}
7  #donde:
8  #inicial_exp es la instrucción que inicializa el bucle. Consiste generalmente
9  #en la asignación de un valor a una variable que permite controlar el número de iteraciones.
10 #test_exp es la expresión evaluada en cada iteración.
11 #Cuando esta expresión es verdadera, el bloque de instrucciones se ejecuta.
12 #incremento_exp es la expresión que permite la actualización de
13 #la variable que controla el número de iteraciones.
14
15 #EJEMPLO
16 print "10 Iteraciones\n";
17 for (my $i=0; $i<10; $i++) {
18         print "Iteracion numero $i\n";
19 }
20
21
22
```

```
D:\Dwimperl\perl\bin\perl.exe  estructuras de control.pl
10 Iteraciones
Iteracion numero 0
Iteracion numero 1
Iteracion numero 2
Iteracion numero 3
Iteracion numero 4
Iteracion numero 5
Iteracion numero 6
Iteracion numero 7
Iteracion numero 8
Iteracion numero 9
Presione una tecla para continuar . . .
```

III. EJERCICIOS BÁSICOS

OPERACIONES BÁSICAS

Este problema pide al usuario que ingrese dos números e imprime las operaciones básicas.

```perl
#! /usr/bin/perl
use strict;
use warnings;
print "Ingrese el primer numero: ";
#Pedimos los numeros por teclado
my $numero1=<STDIN>;
#El STDIN es para almcenar una variable que se pide por teclado
#Almacenamos los numeros ingresados en variables
print "Ingrese el segundo numero: ";
my $numero2=<STDIN>;
#Esto se hace 2 veces para pedir 2 numeros
my $suma=$numero1 + $numero2;
my $resta=$numero1 - $numero2;
my $producto=$numero1 * $numero2;
my $division=$numero1 / $numero2;
my $modulo=$numero1 % $numero2;
#Ponemos variables que almacenen las operaciones basicas de los 2 numeros
chomp($numero1);
chomp($numero2);
#Ponemos el chomp para que no salte a la siguiente linea una vez se usen las variables para las operaciones
print "La suma de $numero1 y $numero2 es $suma\n";
print "La resta de $numero1 y $numero2 es $resta\n";
print "La multiplicacion de $numero1 y $numero2 es $producto\n";
print "La division de $numero1 y $numero2 es $division\n";
print "El modulo de $numero1 y $numero2 es $modulo\n";
#Imprimimos todas las operaciones y colocamos el \n para saltar a la linea siguiente
#Esto imprimira las operaciones en orden y sera mejor a la vista
exit;
```

EJECUCIÓN:

```
C:\Users\ACER\Desktop\perlArchivos>perl NivelBasicoOperaciones.pl
Ingrese el primer numero: 30
Ingrese el segundo numero: 6
La suma de 30 y 6 es 36
La resta de 30 y 6 es 24
La multiplicacion de 30 y 6 es 180
La division de 30 y 6 es 5
El modulo de 30 y 6 es 0
```

PEDIR NOMBRE

Este problema nos pide que ingresemos nuestro nombre y nos imprimirá un “Hola + el nombre que hayamos puesto.

```
1   #! /usr/bin/perl
2   use strict;
3   use warnings;
4   print "Ingrese el tu nombre: ";
5   #Pedimos el nombre por teclado
6   my $nombre=<STDIN>;
7   #Almacenamos una variable con STDIN que se pide desde teclado
8   print "Hola $nombre";
9   #Imprime un Hola junto al nombre que se pidio
10  exit;
```

EJECUCIÓN

```
C:\Users\ACER\Desktop\perlArchivos>perl NivelBasicoSaludo.pl
Ingrese el tu nombre: Ernesto
Hola Ernesto
```

IV. EJERCICIOS INTERMEDIOS

SERIE DE FIBONACCI
El ejercicio le pide al usuario que ingrese la cantidad de números de fibonacci que desea ver, posteriormente el programa los muestra.

```
1   #Serie Fibonacci
2
3   #variables
4   my $n;
5   my $suma=0;
6   my $aux;
7   my $num=1;
8
9   #Interaccion con el usuario
10  print "Cuantos numeros de Fibonacci desea obtener? \n";
11  $n = <STDIN>;   #Una serie de "n" numeros
12
13  #Se hace unso de un bucle "for" para sumar el termino anterior
14  for(my $i=0;$i<$n;$i++){
15          $aux=$suma;
16          $suma = $suma + $num;
17          $num=$aux;
18          print "$suma ";#Se imprimi uno por uno los resultados
19  }
20  print "\n"; #Salto de linea para la estetica
21
```

EJECUCIÓN:

```
Cuantos numeros de Fibonacci desea obtener?
15
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
Presione una tecla para continuar . . .
```

NÚMEROS PRIMOS

Este problema le pide al usuario la cantidad de números primos que desea ver y los imprime

*numeros primos.pl

```
1
2  #Ejercicio intermedio-Números primos
3  #Inicializamos un contador de números primos en 0
4  my $cont=0;
5  print "Ingrese cantidad de numeros primos que desea ver: ";
6  #Le pedimos al usuario que ingrese por teclado
7  my $cantidad=<STDIN>;
8  #Buscamos los divisores de un número desde el 3 y de 2 en 2
9  #ya que el 2 es el unico primo par,todos los demás serán impares
10 for(my $i=3;$cont<$cantidad-1;$i+=2){
11    my $divisores=0;
12    #Si encontramos un divisor ,lo contamos
13    for(my $j=1;$j<=$i;$j++){
14       if($i%$j==0){
15         $divisores++;
16       }
17    }
18    #El 2 es el unico primo par,por lo tanto lo imprimiremos aparte
19    if($i==3){
20         print 2;
21         print " ";
22      }
23     #Cuando haya solo 2 divisores entre 1 y "$i",el numero sera primo y por lo tanto
24     #contamos ese primo en nuestro contador principal($cont) y lo imprimimos
25    if($divisores==2){
26      $cont++;
27      print $i." ";
28      #Para una mejor visualización,se separarán de 10 en 10
29      #excepto la primera fila que tendrá 11
30      if($cont%10==0){
31         print "\n";
32      }
33    }
34 }
35
```

EJECUCIÓN

```
D:\Dwimperl\perl\bin\perl.exe  numeros primos.pl
Ingrese cantidad de numeros primos que desea ver: 100
2 3 5 7 11 13 17 19 23 29 31
37 41 43 47 53 59 61 67 71 73
79 83 89 97 101 103 107 109 113 127
131 137 139 149 151 157 163 167 173 179
181 191 193 197 199 211 223 227 229 233
239 241 251 257 263 269 271 277 281 283
293 307 311 313 317 331 337 347 349 353
359 367 373 379 383 389 397 401 409 419
421 431 433 439 443 449 457 461 463 467
479 487 491 499 503 509 521 523 541 Presione una tecla para continuar . . .
```

V. EJERCICIO AVANZADO

FACTORIAL RECURSIVO

CODIGO:

```
1   print "Ingrese un numero ";
2   $n=<STDIN>;
3   chop($n);
4   $i=1;
5   print &factorial ($n);
6   sub factorial
7   {
8           my $n=$_[0];
9           if(($n==0)||($n==1))
10          {$fact=1;
11          }
12          else
13          {$fact=$n*factorial($n-1);
14          }
15  }
```

EJECUCION

```
C:\Dwimperl\perl\bin\perl.exe FACTORIAL.pl
Ingrese un numero 7
5040Presione una tecla para continuar . . .
```

MATRIZ TRANSPUESTA

El ejercicio pide al usuario filas y columnas para formar una matriz e al final imprime la matriz y su transpuesta

```
#Algoritmo para una matriz transpuesta
print "Ingrese el numero de filas de la matriz: ";
$m=<STDIN>;
chop($m);
print "Ingrese el numero de columnas de la matriz: ";
$n=<STDIN>;
chop($n);
for ($i=0;$i<$m;$i++){

        for ($j=0;$j<$n;$j++)
        { $valor=<STDIN>;
        chop ($valor);
        $R[$i][$j]=$valor;
        }
}
for($i=0;$i<$n;$i++){

        for($j=0;$j<$m;$j++)
        {
         $T[$i][$j]=$R[$j][$i];
        }
}
print "Matriz Original\n";
for($i=0;$i<$m;$i++)
{
        for($j=0;$j<$n;$j++)
        {print $R[$i][$j]," ";
        }
        print "\n";
}
print "Matriz Transupesta\n";
for ($i=0;$i<$n;$i++)
{
        for($j=0;$j<$m;$j++)
        {print $T [$i][$j]," ";
        }
        print "\n";
}

```

EJECUCIÓN:

```
Ingrese el numero de filas de la matriz: 2
Ingrese el numero de columnas de la matriz: 3
7
18
3
-6
0
12
Matriz Original
7 18 3
-6 0 12
Matriz Transupesta
7 -6
18 0
3 12
Presione una tecla para continuar . . .
```

VI. APLICACIONES

SIMULADOR DE LANZAMIENTOS DE 2 DADOS.

EJECUCION Y EXPLICACION EN VIDEO

```
#SIMULADOR DE LANZAMIENTO DE 2 DADOS

#Para los sistemas operativos que usan dos caracteres (Windows) para separar líneas dentro de archivos de texto
#pero no tiene ningún efecto en los sistemas operativos que usan caracteres únicos (Unix, Mac OS, QNX)
use strict;
binmode (STDOUT,":encoding(cp850)");
binmode (STDIN,":encoding(cp850)");
#Variables Declaradas
my @frecuencias=(0,0,0,0,0,0,0,0,0,0,0);    #Arreglo para la frecuencia de las sumas posibles
my @sumaPosible = (2 ,3 ,4 ,5 ,6,7,8,9,10,11,12);   #Arreglo de las sumas posibles
my $suma;
my $dado1;
my $dado2;
#Código
print "Ingrese numero de lanzamientos de los 2 dados: ";  #Se pide un número de lanzamientos
$a=<STDIN>;   #Dicho número de lanzamientos se almacena aquí
for(my $i=0;$i<$a;$i++){        #Bucle for para los datos de los dados
   $dado1= 1 + int(rand(6));  #Genera aleatoriamente un numero entre 1-6
   $dado2= 1 + int(rand(6));  #Genera aleatoriamente un numero entre 1-6
   $suma=$dado1+$dado2;  #Se suman ambos resultados
   $frecuencias[$suma-2]++;  #Se aumenta en 1 a la variable de la posición, suma-2
}
for(my $i=0;$i<11;$i++){
  print "$frecuencias[$i] ";  #Se imprime las frecuencias para corroborar lo del gráfico
}
#Gnuplot
open (ARCHIVO,'>','archivo.dat'); #Nombre del archivo
foreach my $x (0..@sumaPosible-1){
  print ARCHIVO $sumaPosible[$x]; # Eje vertical
  print ARCHIVO " ";
  print ARCHIVO $frecuencias[$x]; # Eje Horizontal
  print ARCHIVO "\n";
 }
close(ARCHIVO);

my $resultado = `C:\\Users\\Fernando\\Downloads\\prueba3.gp`; #Dirección del archivo
print $resultado; #Se muestra la grafica
```

VII. RECOMENDACIONES:

En nuestra experiencia se podría decir que el lenguaje de programación Perl fue bastante interesante ya que comparte mucho con otros lenguajes lo que lo hace de más fácil entendimiento y muy intuitivo si ya trabajaste con algún otro lenguaje además de que nos dejó con una sensación gran satisfactoria con los resultados.

También tuvimos algunos inconvenientes que seria bueno resaltar como que al trabajar con otros lenguajes nos pareció un poco raro trabajar con la consola de la computadora algo que no estábamos acostumbrados, además de que la comunicación fue otra gran dificultad para nosotros puesto que si alguien trabaja en grupos tenga presente que la comunicación es muy importante.

VIII. CONCLUSONES

En primer lugar, tener conocimiento de lenguaje programación permite comprender como se desarrolla una aplicación que es creada por esta. El estudiante o interesado en esta tecnología que conozca uno o más lenguajes de programación podrá aprender autodidactamente otros lenguajes de programación. Conocer más de un lenguaje de programación nos hace más competitivo en el mercado.

Como Perl que nos facilita a los programadores la integración de interfaces o componentes de terceros que no son compatibles entre sí.

IX. ENTREGABLES

URL Git:
https://github.com/elopezqu/IC/blob/main/README.md
URL Video:
https://www.youtube.com/watch?v=oey6pvKENRQ

X. REFERENCIAS:

[1]https://trabajodeprogramacionperl.blogspot.com/p/historia.html
[2]https://trabajodeprogramacionperl.blogspot.com/p/caracteristicas.html?m=1
[3]https://tecnoinformatic.com/c-programacion/perl-en-lenguaje-de-programacion/
[4]https://lenguajesdeprogramacion.net/perl/
[5]https://metacpan.org/dist/POD2-ES/view/lib/POD2/ES/perlfaq1.pod
[6]https://citas.in/autores/larry-wall/