PERL

CLASE 2

PERL

Repaso clase anterior:

Ejemplo

```
#!/bin/perl
print "Hola Mundo\n"; # Imprime Hola Mundo
```

Tipos de datos

Escalares `$x=0.897;$var=”Hola”;`
Arreglos `@mezcla=("hola",23,"adios", 31.234);`
Hash `%cuota=("root",1000,"Juan",256,"Jose",4000);`

Como se referencia cada tipo de datos

```
print $var              # Variable escalar
print $mezcla[0]        # Elemento de un array
Print $cuota{root}      # Elemento de un hash
```

PERL

Repaso clase anterior:

Estructuras de Control

```
if (expresión) {
   instrucción o bloque de intrucciones 1;}
[else {
   instrucción o bloque de instrucciones 2;}

while (expresión) {instrucción o bloque de inst.; }

for (inicial_exp; test_exp; incremento_exp) {
    instrucción o bloque de intrucciones;}

foreach $variable (@lista) {
    instrucción o bloque de instrucciones;}
```

PERL

Repaso clase anterior:

Entradas y Salidas:

open (Manejador_de_fichero, Modo_y_NombreFichero).
Close(Manejador_de_fichero)

Ejemplo:

```
open (ENTRADA,"<$entrada.txt") || die "ERROR: No puedo
abrir el fichero $entrada\n";
open (SALIDA,">$salida.txt") || die "ERROR: No puedo abrir
el fichero $salida\n";
while ($linea=<ENTRADA>)
{
print SALIDA $linea;
}
close (ENTRADA);
close (SALIDA);
```

PERL

Lectura de directorios

opendir(manejador,directorio) (Devuelve verdadero si puede abrir el directorio)

readdir(manejador) (Dependiendo del contexto devuelve una lista o el siguiente archivo).

close(manejador) (Devuelve verdadero si puede abrir el directorio)

PERL

Lectura de directorios

Ejemplos

```
#!/usr/bin/perl
$dir=$ARGV[0];
if (opendir(DIRH,"$dir"))
{
@flist=readdir(DIRH);
closedir(DIRH);
}
foreach (@flist)
{
# ignorar . y .. :
next if ($_ eq "." || $_ eq "..");
if ( -x "$dir/$_" )
{print "$dir/$_\n";}
}
```

PERL

Lectura de directorios

Ejemplos

```
#!/usr/bin/perl
$dirname = "./";
opendir ( DIR, $dirname ) || die "Error in opening dir
$dirname\n";
while( $filename = readdir(DIR))
{
   if (-r $filename)
   {
      print("Tiene permiso de lectura $filename\n");
   }
}
closedir(DIR);
```

PERL

File Test Operator

Operator	Description
-d	Is name a directory?
-f	Is name an ordinary file?
-l	Is name a symbolic link?
-o	Is name owned by the user?
-r	Is name a readable file?
-w	Is name a writable file?
-x	Is name an executable file?
-z	Is name an empty file?

PERL

Operador diamante

Este operador toma diferentes comportamientos según el contexto.

- Pasandole un manejador

```
while ($linea=<ENTRADA>)
# Lee los registros del manejador ENTRADA
{print SALIDA $linea;}
```

- Pasandole un *

```
while ( <*> )
{print "$_\n";}
# Devuelve nombres de archivos del directorio
while ( <*.pl> )
{print "$_\n";}
# Devuelve nombres de archivos del directorio con extension
pl.
```

PERL

Operador diamante

- Sin parametro

El operador diamante tratará de leer @ARGV si no tiene elementos lee de la entrada standard.

```
While ( <> )# Lee de entrada standard
{print "$_\n";}
# Espera de entrada standard y la imprime.

While ( <> )# Lee de @ARGV
{print "$_\n";}
sript.pl arch1 arch2 arch3
# Muestra por salida las lineas de arch1, arch2 y arch3
```

PERL

Operador diamante

- Asignando a un arreglo

```
@arreglo =<*> ;
for ($i=0;$i<=$#arreglo;$i++)
{print "$arreglo[$i] \n";}
# Recorre archivos del directorio.

open(MYFILE,”<pgm.pl”) || die “Error”;
@array = <MYFILE>;
# Asigna al arreglo todos los registros del archivo.
foreach (@array){
$i=index($_,"print",1); # busca el string “print”
if ($i > 0){
   print "$_\n";
   };
};
```

PERL

Como obtener la hora del sistema

`localtime`

Contexto array

Devuelve los siguientes elementos:
-Segundos
-Minutos
-Hora
-Día
-Mes (0-11)
-Año desde 1900
-Día de la semana (Domingo = 0)
-Numero de día del año desde 0.
-Verdadero en el caso de que la fecha esté dentro del horario de verano

PERL

Como obtener la hora del sistema

Ejemplo:

```
($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,
$isdst)=localtime;
$year+=1900;
$mon++;
print "anio=$year\nmes=$mon \ndia=$mday\nhora=$hour
\nmin=$min \nsegundos= $sec \n";
print "$mday/$mon/$year $hour:$min:$sec \n";
```

```
SALIDA:
anio=2011
mes=10
dia=3
hora=11
min=30
segundos=30
3/10/2011 11:30:30
```

PERL

Como obtener la hora del sistema

Contexto escalar

```
$fecha=localtime;
Print $fecha.”\n”;

SALIDA
Sat Apr  7 19:49:46 2012
```

Como obtener usuario

La función `getlogin` devuelve el user ID del usuario que esta logueado.

```
$logname = getlogin();
print "El usuario es $logname \n";
```

PERL

Llamadas al sistema operativo

Para pedir al sistema operativo que ejecute otros programas desde un programa en Perl podemos hacerlo de varias maneras. Al hacerlo estamos en realidad llamando al sistema operativo y esperando su respuesta:

Ejemplos:

```
$resultado=`ls -l *.sh`;# muestra todos los archivos .sh
del directorio actual.
$resultado = `df -k`;  # muestra el espacio libre en los
sistemas de ficheros de tu sistema,
$resultado = `cat archivo.pdb | grep ATOM`;  # imprime las
líneas 'ATOM' de un PDB

$resultado = system("cat archivo.pdb | grep ATOM");
$resultado = system("pwd");    # devuelve el directorio
actual
```

PERL

SPLIT

La funcion split() divide una cadena de acuerdo a un patrón pasado como parámetro.

```
split(PATRÓN,CADENA);
```

Ejemplo

```
$informacion = "Juan,Gomez,29";
@data = split(",", $informacion);
($nombre,$apellido,$edad) = split(",", $informacion);
print "$nombre $edad  $apellido \n";
print $data[1] . "\n";
```

PERL

JOIN

La funcion join() permite unir diferentes partes en una sola, introduciendo un separador entre ellas, es decir, hace lo contrario de split..

```
join(STRING,LISTA);
```

Ejemplo

```
push (@numeros,$i)  while ($i++ <= 10);
$cadena=join(",",@numeros);

$cadena=join("\n",@numeros);
print $cadena . "\n";
```

PERL

Expresiones regulares

Las expresiones regurales son caracteres especiales que juntos o por separado actúan como patrones de búsqueda, comparación o sustitución en una cadena. Estas expresiones se colocan entre 2 barras (/expr/).

Caracteres especiales

^	Inicio de la linea
$	fin de la linea
a*	es una a cero o más veces.
a+	es una a una o más veces.
a?	es una a una o ninguna veces.
a{n}	es una a n veces.
a{n,}	es una a n o más veces.
a{n,m}	es una a entre n y m veces, incluidos ambos valores.

PERL

Expresiones regulares

Caracteres especiales

\d es equivalente a [0-9].
\s es equivalente a [\ \t\r\n\f] (espacio, tabulador y retornos de carro).
\w es equivalente a [0-9a-zA-Z_].
\D es el negado de \d; Representa cualquier caracter que no sea un digito [^0-9].
\S es el negado de \s; Represnta cualquier cosa que no sea un espacio en blanco [^\s].
\W es el negado de \w; Representa cualquier cosa que no sean cifras, letras o _ [^\w].
\n newline.

PERL

Expresiones regulares

Expresiones regulares de comparación.

valor a comparar =~ patrón de búsqueda

```
if ($linea =~ /html/) {
   print $linea; }

$string = "Mi frase no muy larga";
if($string =~ /frase/){
print "Se encontró la palabra: frase.\n";
}
$string = "Mi frase no muy larga";
if($string !~ /jugo/){
print "No se encontró la palabra: jugo.\n";
}
```

PERL

Expresiones regulares

Expresiones regulares de comparación.

Ejemplos

```
$cadena="Hola Mundo";

($cadena =~  /^Hola/) # que comience con Hola
($cadena =~  /Mundo$/)# que termine con Mundo
($cadena =~  /M*/)    # que contenga 0 o mas M
($cadena =~  /M?ndo/) # que contenga cualquier carácter
                        en el lugar de la u.
```

PERL

Expresiones regulares

Expresiones regulares de sustitución.

`variable =~ s/patrón búsqueda/patrón sustitución/opciones`

```
$string = "Hoy es Lunes";
$string =~ s/Lunes/Martes/;
print $string ."\n"

$var="Hola Mundo";
$var=~ s/o/a/g;
print $var; # Imprime Hala Munda

$var=~ s/\s/-/g;
print $var; # Imprime Hala-Munda
```

PERL

Variables especiales

```
$_ Contiene el contenido del último registro leído.
$0 Contiene el nombre del script perl.
$$ Contiene el PID dscript perl.
$. Contiene el número de la última línea leída.
$/ Separador de campo para la entrada.
$[ Contiene el valor del primer índice de una array (0)
$] Contiene la version de perl ("perl -v").
$< Uid real del proceso actual.
$@ Contiene el mensaje de error de sintaxis de perl del
último comando evaluado.
$( Gid real del proceso actual.
@ARGV Contiene los parámetros pasados a nuestro script
%ENV: Array asociativo que contiene las variables de
entorno bajo el que se ejecuta nuestro script Perl.(ej.
$ENV{HOME})
```

PERL

Ejemplos

```
##!/usr/bin/perl
print "Ingrese numero desde: "; $desde= <STDIN>;
print "Ingrese numero hasta: "; $hasta= <STDIN>;
chop ($desde); chop ($hasta);
print "Los numeros primos desde $desde hasta $hasta
son:\n";
foreach $i ($desde .. $hasta)
{
    $resul=1;
    foreach $j (2 .. $i-1)
    {
      if ($i % $j == 0) {$resul=0; }
     }
     print "===== $i =====\n" if $resul!=0;
}
```

PERL

Ejemplos

```
#! /usr/bin/perl
# localiza un programa que va a ejecutar

die "Uso: cual programa\n" if ($#ARGV < 0);
$programa= $ARGV[0];
@lpath= split(/:/, $ENV{PATH});

foreach $dir (@lpath)
{
  next if ($dir =~ /^\.+$/);
  if (-x "$dir/$programa")
  {
     print "$dir/$programa\n";
  }
}
```

PERL

Ejemplos

```
#! /usr/bin/perl
# Busca ficheros de tipo .pdf y muestra su tamaño

opendir(DIR,$ARGV[0]) || die "No se puede abrir el
directorio \n";
@indice= readdir(DIR) ;
closedir(DIR);

foreach $f (@indice)
{
  next unless ( $f =~ /\.pdf$/ );
  system("wc $f");
  `wc $f `;
}
```

PERL

Ejemplo

Archivo Facturas:nro de factura, fecha,sucursal, importe

```
open (ENT,"<facturas.txt") || die "ERROR: No puedo abrir
el fichero facturas\n";
open (SAL,">sucursales.txt")|| die "ERROR: No puedo abrir
el fichero sucursales\n";
while (<ENT>)
{
chomp($_);
@reg=split(”,”,$_);
$suc{$reg[2]}+=$reg[3];
}
open (SAL,">sucursales.txt");
foreach (keys(%suc))
{
print SAL $_.",".$suc{$_} ."\n";
};
close (SAL);
close (ENT);
```