

# Referencia Rápida de Perl

Variables, regex, E/S de archivos, referencias, módulos esenciales

## Fundamentos

### Hola Mundo

```
#!/usr/bin/perl
use strict;
use warnings;
print "Hello, World!\n";
say "Hello, World!"; # con use feature 'say';
```

### Ejecutar Perl

```
perl script.pl # ejecutar un archivo
perl -e 'print "hi\n"' # ejecutar en línea
perl -ne 'print' file # procesar archivo línea a línea
```

### Comentarios y Documentación

```
# comentario de una línea
=pod
Documentación POD multilinea
=cut
```

## Variables

### Sigils

**\$scalar** Un solo valor (cadena, número, referencia)  
**@array** Lista ordenada de escalares  
**%hash** Pares clave-valor  
**\$array[0]** Acceder a un elemento del array (contexto escalar)  
**%hash{key}** Acceder a un valor del hash (contexto escalar)

### Variables Escalares

```
my $name = "Perl"; # cadena
my $version = 5.40; # número
my $count = 42; # entero
my $undef; # indefinido (undef)
my $combined = "$name v$version"; # interpolación
```

### Contexto

```
my @arr = (1, 2, 3);
my $count = @arr; # contexto escalar: 3
my @copy = @arr; # contexto lista: (1, 2, 3)
my $len = scalar @arr; # forzar contexto escalar
```

### Variables Especiales

**\$\_** Variable predeterminada (tópico)  
**@\_** Argumentos de subrutina  
**\$\_!** Mensaje de error del sistema  
**\$\_@** Error de eval  
**\$\_0** Nombre del programa  
**@ARGV** Argumentos de línea de comandos  
**%ENV** Variables de entorno

## Operadores

### Operadores de Comparación

**==, !=, <, >, <=, >=** Comparación numérica  
**eq, ne, lt, gt, le, ge** Comparación de cadenas  
**<=>** Nave espacial numérico (retorna -1, 0, 1)  
**cmp** Nave espacial de cadenas  
**==** Coincidencia / enlace de regex  
**!~** Coincidencia negada de regex

### Operadores de Cadenas

```
my $full = "Hello" . " " . "World"; # concatenación
my $line = "-" x 40; # repetición
my $len = length($full); # 11
```

## Operadores Lógicos

**&& / and** AND lógico (baja precedencia: **and**)  
**|| / or** OR lógico (baja precedencia: **or**)  
**//** Definido-o (retorna izquierda si está definido)  
**! / not** NOT lógico  
**? :** Condicional ternario

## Flujo de Control

### Condicionales

```
if ($x > 0) { print "positivo\n"; }
elsif ($x == 0) { print "cero\n"; }
else { print "negativo\n"; }
print "sí\n" if $condition; # if postfijo
print "no\n" unless $condition; # unless postfijo
```

### Bucles

```
for my $i (0..9) { print "$i\n"; }
foreach my $item (@array) { print "$item\n"; }
while ($line = <STDIN>) { chomp $line; }
until ($done) { last if check(); }
```

### Control de Bucle

**next** Saltar a la siguiente iteración (como **continue**)  
**last** Salir del bucle (como **break**)  
**redo** Reiniciar iteración actual  
**next LABEL** Saltar a la siguiente iteración del bucle etiquetado  
**last LABEL** Salir del bucle etiquetado

### Given / When

```
use feature 'switch';
given ($status) {
    when ("ok") { say "success"; }
    when ("error") { say "failed"; }
    default { say "unknown"; }
}
```

## Subrutinas

### Subrutina Básica

```
sub greet {
    my ($name) = @_;
    return "Hello, $name!";
}
my $msg = greet("Alice");
```

### Parámetros Predeterminados y con Nombre

```
sub connect {
    my (%opts) = @_;
    my $host = $opts{host} // "localhost";
    my $port = $opts{port} // 5432;
    return "$host:$port";
}
connect(host => "db.example.com", port => 3306);
```

### Referencias a Subrutinas

```
my $double = sub { return $_[0] * 2; };
print $double->(5); # 10
my @sorted = sort { $a <=> $b } @nums;
```

### Prototipos y Firmas

```
use feature 'signatures';
sub add($a, $b) { return $a + $b; }
sub greet($name, $greeting = "Hello") {
    return "$greeting, $name!";
}
```

## Regex

### Coincidencia

```
if ($str =~ /pattern/) { print "coincidió\n"; }
if ($str =~ /(\\d+)/) { print "número: $1\n"; }
my @matches = ($str =~ /(\\w+)/g); # todas las coincidencias
```

### Sustitución

```
$str =~ s/old/new/; # primera ocurrencia
$str =~ s/old/new/g; # global (todas las ocurrencias)
$str =~ s/^\s+|\s+$//g; # eliminar espacios en blanco
(my $clean = $str) =~ s/\\W//g; # copia no destructiva
```

### Modificadores

**/i** Sin distinción de mayúsculas  
**/g** Global (todas las coincidencias)  
**/m** Multilínea (**^** y **\$** coinciden con límites de línea)  
**/s** Línea simple (**.** coincide con salto de línea)  
**/x** Extendido (permite espacios en blanco y comentarios)

### Patrones Comunes

**\d, \D** Dígito / no dígito  
**\w, \W** Carácter de palabra / no palabra  
**\s, \S** Espacio en blanco / sin espacio  
**\b** Límite de palabra  
**(? ... )** Grupo sin captura  
**(?<name> ... )** Captura con nombre (acceder con **\${name}**)

## E/S de Archivos

### Abrir y Leer

```
open(my $fh, '<', 'data.txt') or die "No se puede abrir: $!";
while (my $line = <$fh>) {
    chomp $line;
    print "$line\n";
}
close($fh);
```

### Escribir y Añadir

```
open(my $fh, '>', 'out.txt') or die "No se puede abrir: $!";
print $fh "Hello\n";
close($fh);
open(my $fh, '>>', 'log.txt') or die "No se puede abrir: $!";
print $fh "entrada\n";
close($fh);
```

### Leer Archivo Completo

```
use File::Slurp;
my $content = read_file('data.txt');
my @lines = read_file('data.txt', chomp => 1);
```

### Pruebas de Archivo

**-e \$path** El archivo existe  
**-f \$path** Es un archivo regular  
**-d \$path** Es un directorio  
**-r / -w / -x** Legible / escribible / ejecutable  
**-s \$path** Tamaño del archivo en bytes (0 si vacío)  
**-z \$path** El archivo tiene tamaño cero

# Referencia Rápida de Perl

## Arrays y Hashes

### Arrays

```
my @arr = (1, 2, 3, 4, 5);
push @arr, 6;           # agregar al final
my $last = pop @arr;   # eliminar el último
my $first = shift @arr; # eliminar el primero
unshift @arr, 0;       # agregar al inicio
my @slice = @arr[1..3]; # segmento
```

### Funciones de Array

<b>scalar @arr</b>	Número de elementos
<b>push / pop</b>	Agregar/eliminar del final
<b>shift / unshift</b>	Eliminar/agregar del inicio
<b>splice(@a, 2, 1)</b>	Eliminar 1 elemento en el índice 2
<b>sort @arr</b>	Ordenar alfabéticamente
<b>reverse @arr</b>	Invertir el orden
<b>grep { /pat/ } @arr</b>	Filtrar por patrón
<b>map { \$_ * 2 } @arr</b>	Transformar cada elemento
<b>join(',', @arr)</b>	Unir en una cadena

### Hashes

```
my %user = (name => "Alice", age => 30);
$user{email} = "a@b.com"; # agregar par
delete $user{age};        # eliminar par
my @keys = keys %user;
my @vals = values %user;
```

### Iteración de Hash

```
while (my ($k, $v) = each %hash) {
    print "$k => $v\n";
}
for my $key (sort keys %hash) {
    print "key: $hash{$key}\n";
}
```

## Referencias

### Crear Referencias

```
my $scalar_ref = \$name;
my $array_ref  = \@arr;
my $hash_ref   = \%hash;
my $anon_arr   = [1, 2, 3]; # ref de array anónimo
my $anon_hash  = {a => 1, b => 2}; # ref de hash anónimo
```

### Desreferenciar

```
print $$scalar_ref; # desreferenciar escalar
print $array_ref->[0]; # notación de flecha
print $hash_ref->{key}; # notación de flecha
my @copy = @$array_ref; # desreferenciar a array
my %copy = %$hash_ref; # desreferenciar a hash
```

### Estructuras de Datos Complejas

```
my @users = (
    { name => "Alice", age => 30 },
    { name => "Bob",   age => 25 },
);
print $users[0]->{name}; # "Alice"
```

### Función ref()

<b>ref(\$r) eq 'SCALAR'</b>	Referencia a escalar
<b>ref(\$r) eq 'ARRAY'</b>	Referencia a array
<b>ref(\$r) eq 'HASH'</b>	Referencia a hash
<b>ref(\$r) eq 'CODE'</b>	Referencia a subrutina

## Módulos

### Usar Módulos

```
use strict;
use warnings;
use List::Util qw(sum max min);
use File::Basename;
use Cwd qw(abs_path);
```

### Crear un Módulo

```
# MyModule.pm
package MyModule;
use Exporter 'import';
our @EXPORT_OK = qw(helper);
sub helper { return "help"; }
1; # el módulo debe retornar verdadero
```

### Módulos Básicos Comunes

<b>List::Util</b>	sum, max, min, reduce, any, all
<b>File::Basename</b>	basename, dirname, fileparse
<b>File::Path</b>	make_path, remove_tree
<b>Getopt::Long</b>	Análisis de opciones de línea de comandos
<b>JSON</b>	encode_json, decode_json
<b>LWP::Simple</b>	get(\$url) — cliente HTTP simple
<b>Data::Dumper</b>	Volcado de depuración de estructuras de datos
<b>Carp</b>	croak, confess — mejores mensajes de error

### CPAN

```
cpan install Module::Name # instalar desde CPAN
cpanm Module::Name        # cpanminus (más rápido)
perldoc Module::Name      # leer docs del módulo
```