

# Referência Rápida de Perl

Variáveis, regex, E/S de arquivos, referências, módulos essenciais

## Básico

### Hello World

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

### Executar Perl

```
perl script.pl # run a file
perl -e 'print "hi\n"' # run inline
perl -ne 'print' file # process file line by line
```

### Comentários e Documentação

```
# single-line comment
=pod
Multi-line POD documentation
=cut
```

## Variáveis

### Sigils

<b>\$scalar</b>	Valor único (string, número, referência)
<b>@array</b>	Lista ordenada de escalares
<b>%hash</b>	Pares chave-valor
<b>\$array[0]</b>	Acessar elemento único de array (contexto escalar)
<b>%hash{key}</b>	Acessar valor único de hash (contexto escalar)

### Variáveis Escalares

```
my $name = "Perl"; # string
my $version = 5.40; # number
my $count = 42; # integer
my $undef; # undefined (undef)
my $combined = "$name v$version"; # interpolation
```

### Contexto

```
my @arr = (1, 2, 3);
my $count = @arr; # scalar context: 3
my @copy = @arr; # list context: (1, 2, 3)
my $len = scalar @arr; # force scalar context
```

### Variáveis Especiais

<b>\$_</b>	Variável padrão (tópico)
<b>@_</b>	Argumentos de sub-rotina
<b>\$_!</b>	Mensagem de erro do sistema
<b>\$_@</b>	Erro de eval
<b>\$_0</b>	Nome do programa
<b>@ARGV</b>	Argumentos da linha de comando
<b>%ENV</b>	Variáveis de ambiente

## Operadores

### Operadores de Comparação

<b>==, !=, &lt;, &gt;, &lt;=, &gt;=</b>	Comparação numérica
<b>eq, ne, lt, gt, le, ge</b>	Comparação de strings
<b>&lt;=&gt;</b>	Spaceship numérico (retorna -1, 0, 1)
<b>cmp</b>	Spaceship de strings
<b>==~</b>	Correspondência / ligação com regex
<b>!~</b>	Correspondência negada com regex

### Operadores de String

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

## Operadores Lógicos

<b>&amp;&amp; / and</b>	AND lógico (baixa precedência: <b>and</b> )
<b>   / or</b>	OR lógico (baixa precedência: <b>or</b> )
<b>//</b>	Defined-or (retorna esquerda se definido)
<b>! / not</b>	NOT lógico
<b>? :</b>	Ternário condicional

## Controle de Fluxo

### Condicionais

```
if ($x > 0) { print "positive\n"; }
elsif ($x == 0) { print "zero\n"; }
else { print "negative\n"; }
print "yes\n" if $condition; # postfix if
print "no\n" unless $condition; # postfix unless
```

### Laços

```
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(); }
```

## Controle de Laço

<b>next</b>	Pular para próxima iteração (como <b>continue</b> )
<b>last</b>	Sair do laço (como <b>break</b> )
<b>redo</b>	Reiniciar iteração atual
<b>next LABEL</b>	Pular para próxima iteração do laço rotulado
<b>last LABEL</b>	Sair do laço rotulado

## Given / When

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

## Sub-rotinas

### Sub-rotina Básica

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

### Parâmetros Padrão e Nomeados

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

### Referências de Sub-rotinas

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

### Protótipos e Assinaturas

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

## Regex

### Correspondência

```
if ($str =~ /pattern/) { print "matched\n"; }
if ($str =~ /(\d+)/) { print "number: $1\n"; }
my @matches = ($str =~ /(\w+)/g); # all matches
```

### Substituição

```
$str =~ s/old/new/; # first occurrence
$str =~ s/old/new/g; # global (all occurrences)
$str =~ s/^\s+|\s+$//g; # trim whitespace
(my $clean = $str) =~ s/\W//g; # non-destructive copy
```

### Modificadores

<b>/i</b>	Insensível a maiúsculas
<b>/g</b>	Global (todas as correspondências)
<b>/m</b>	Multilinha (^ e \$ correspondem a limites de linha)
<b>/s</b>	Linha única (. corresponde a nova linha)
<b>/x</b>	Estendido (permite espaços em branco e comentários)

### Padrões Comuns

<b>\d, \D</b>	Dígito / não dígito
<b>\w, \W</b>	Caractere de palavra / não palavra
<b>\s, \S</b>	Espaço em branco / não espaço
<b>\b</b>	Limite de palavra
<b>(? ... )</b>	Grupo não capturante
<b>(?&lt;name&gt; ... )</b>	Captura nomeada (acessar via <b>\${+name}</b> )

## E/S de Arquivos

### Abrir e Ler

```
open(my $fh, '<', 'data.txt') or die "Cannot open: $!";
while (my $line = <$fh>) {
    chomp $line;
    print "$line\n";
}
close($fh);
```

### Escrever e Acrescentar

```
open(my $fh, '>', 'out.txt') or die "Cannot open: $!";
print $fh "Hello\n";
close($fh);
open(my $fh, '>>', 'log.txt') or die "Cannot open: $!";
print $fh "entry\n";
close($fh);
```

### Ler Arquivo Inteiro

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

### Testes de Arquivo

<b>-e \$path</b>	Arquivo existe
<b>-f \$path</b>	É um arquivo regular
<b>-d \$path</b>	É um diretório
<b>-x / -w / -x</b>	Legível / gravável / executável
<b>-s \$path</b>	Tamanho do arquivo em bytes (0 se vazio)
<b>-z \$path</b>	Arquivo tem tamanho zero

# Referência Rápida de Perl

## Arrays e Hashes

### Arrays

```
my @arr = (1, 2, 3, 4, 5);
push @arr, 6;           # append
my $last = pop @arr;   # remove last
my $first = shift @arr; # remove first
unshift @arr, 0;       # prepend
my @slice = @arr[1..3]; # slice
```

### Funções de Array

<b>scalar @arr</b>	Número de elementos
<b>push / pop</b>	Adicionar/remover do final
<b>shift / unshift</b>	Remover/adicionar do início
<b>splice(@a, 2, 1)</b>	Remover 1 elemento no índice 2
<b>sort @arr</b>	Ordenar alfabeticamente
<b>reverse @arr</b>	Inverter ordem
<b>grep { /pat/ } @arr</b>	Filtrar por padrão
<b>map { \$_ * 2 } @arr</b>	Transformar cada elemento
<b>join(',', @arr)</b>	Juntar em string

### Hashes

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

### Iteração de Hash

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

## Referências

### Criando Referências

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

### Desreferenciação

```
print $$scalar_ref; # dereference scalar
print $array_ref->[0]; # arrow notation
print $hash_ref->{key}; # arrow notation
my @copy = @$array_ref; # dereference to array
my %copy = %$hash_ref; # dereference to hash
```

### Estruturas de Dados Complexas

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

### Função ref()

<b>ref(\$r) eq 'SCALAR'</b>	Referência a escalar
<b>ref(\$r) eq 'ARRAY'</b>	Referência a array
<b>ref(\$r) eq 'HASH'</b>	Referência a hash
<b>ref(\$r) eq 'CODE'</b>	Referência a sub-rotina

## Módulos

### Usando Módulos

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

### Criando um Módulo

```
# MyModule.pm
package MyModule;
use Exporter 'import';
our @EXPORT_OK = qw(helper);
sub helper { return "help"; }
1; # module must return true
```

### Módulos Core Comuns

<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>	Análise de opções da linha de comando
<b>JSON</b>	encode_json, decode_json
<b>LWP::Simple</b>	get(\$url) — cliente HTTP simples
<b>Data::Dumper</b>	Dump de depuração de estruturas de dados
<b>Carp</b>	croak, confess — melhores mensagens de erro

### CPAN

```
cpan install Module::Name # install from CPAN
cpanm Module::Name       # cpanminus (faster)
perldoc Module::Name     # read module docs
```