

# Perl 快速参考

变量、正则、文件 I/O、引用与模块要点

## 基础

### Hello World

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

### 运行 Perl

```
perl script.pl # 执行文件
perl -e 'print "hi\n"' # 内联执行
perl -ne 'print' file # 逐行处理文件
```

### 注释与文档

```
# 单行注释
=pod
多行 POD 文档
=cut
```

## 变量

### 符印 (Sigil)

<b>\$scalar</b>	标量 (字符串、数字、引用)
<b>@array</b>	有序列表
<b>%hash</b>	键值对
<b>\$array[0]</b>	访问数组单个元素 (标量上下文)
<b>\$hash{key}</b>	访问 hash 单个值 (标量上下文)

### 标量变量

```
my $name = "Perl"; # 字符串
my $version = 5.40; # 数字
my $count = 42; # 整数
my $undef; # 未定义 (undef)
my $combined = "$name v$version"; # 插值
```

### 上下文

```
my @arr = (1, 2, 3);
my $count = @arr; # 标量上下文: 3
my @copy = @arr; # 列表上下文: (1, 2, 3)
my $len = scalar @arr; # 强制标量上下文
```

### 特殊变量

<b>\$_</b>	默认变量 (话题变量)
<b>@_</b>	子程序参数
<b>!</b>	系统错误信息
<b>@\$</b>	eval 的错误
<b>\$0</b>	程序名
<b>@ARGV</b>	命令行参数
<b>%ENV</b>	环境变量

## 运算符

### 比较运算符

<b>==, !=, &lt;, &gt;, &lt;=, &gt;=</b>	数值比较
<b>eq, ne, lt, gt, le, ge</b>	字符串比较
<b>&lt;=&gt;</b>	数值飞船运算符 (返回 -1、0、1)
<b>cmp</b>	字符串飞船运算符
<b> =~ </b>	正则匹配/绑定
<b> !~ </b>	正则否定匹配

## 字符串运算符

```
my $full = "Hello" . " " . "World"; # 拼接
my $line = "-" x 40; # 重复
my $len = length($full); # ll
```

## 逻辑运算符

<b>&amp;&amp; / and</b>	逻辑与 (低优先级: <b>and</b> )
<b>   / or</b>	逻辑或 (低优先级: <b>or</b> )
<b>//</b>	已定义或 (左侧已定义则返回左侧)
<b>! / not</b>	逻辑非
<b>? :</b>	三目运算符

## 控制流

### 条件语句

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

### 循环

```
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(); }
```

### 循环控制

<b>next</b>	跳到下一次迭代 (类似 continue)
<b>last</b>	退出循环 (类似 break)
<b>redo</b>	重新执行当前迭代
<b>next LABEL</b>	跳到指定标签循环的下一迭代
<b>last LABEL</b>	退出指定标签循环

### given / when

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

## 子程序

### 基础子程序

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

### 默认参数与命名参数

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

### 子程序引用

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

## 原型与签名

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

## 正则表达式

### 匹配

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

### 替换

```
$str =~ s/old/new/; # 替换首个匹配
$str =~ s/old/new/g; # 全局替换
$str =~ s/^\s+|\s+$//g; # 去除首尾空白
(my $clean = $str) =~ s/\W//g; # 非破坏性副本
```

### 修饰符

<b>/i</b>	忽略大小写
<b>/g</b>	全局 (所有匹配)
<b>/m</b>	多行 (^ 和 \$ 匹配行边界)
<b>/s</b>	单行 (. 匹配换行符)
<b>/x</b>	扩展 (允许空白和注释)

### 常用模式

<b>\d, \D</b>	数字 / 非数字
<b>\w, \W</b>	单词字符 / 非单词字符
<b>\s, \S</b>	空白 / 非空白
<b>\b</b>	单词边界
<b>(?: ... )</b>	非捕获组
<b>(?&lt;name&gt; ... )</b>	命名捕获 (通过 <b>\${name}</b> 访问)

## 文件 I/O

### 打开与读取

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

### 写入与追加

```
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);
```

### 一次性读取整个文件

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

# Perl 快速参考

## 文件测试

<b>-e \$path</b>	文件存在
<b>-f \$path</b>	是普通文件
<b>-d \$path</b>	是目录
<b>-r / -w / -x</b>	可读/可写/可执行
<b>-s \$path</b>	文件大小 (字节, 空文件为 0)
<b>-z \$path</b>	文件大小为零

## 数组与 Hash

### 数组

```
my @arr = (1, 2, 3, 4, 5);
push @arr, 6;          # 追加
my $last = pop @arr;  # 移除末尾
my $first = shift @arr; # 移除开头
unshift @arr, 0;      # 在开头插入
my @slice = @arr[1..3]; # 切片
```

### 数组函数

<b>scalar @arr</b>	元素数量
<b>push / pop</b>	末尾添加/删除
<b>shift / unshift</b>	开头删除/添加
<b>splice(@a, 2, 1)</b>	删除索引 2 处的 1 个元素
<b>sort @arr</b>	字母序排序
<b>reverse @arr</b>	反转顺序
<b>grep { /pat/ } @arr</b>	按模式过滤
<b>map { \$_ * 2 } @arr</b>	对每个元素进行变换
<b>join(',', @arr)</b>	连接成字符串

### Hash

```
my %user = (name => "Alice", age => 30);
$user{email} = "a@b.com"; # 添加键值对
delete $user{age};        # 删除键值对
my @keys = keys %user;
my @vals = values %user;
```

### 遍历 Hash

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

## 引用

### 创建引用

```
my $scalar_ref = \$name;
my $array_ref  = \@arr;
my $hash_ref   = \%hash;
my $anon_arr   = [1, 2, 3]; # 匿名数组引用
my $anon_hash  = {a => 1, b => 2}; # 匿名 hash 引用
```

### 解引用

```
print $$scalar_ref; # 解引用标量
print $$array_ref->[0]; # 箭头语法
print $$hash_ref->{key}; # 箭头语法
my @copy = @$array_ref; # 解引用为数组
my %copy = %$hash_ref; # 解引用为 hash
```

## 复杂数据结构

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

### ref() 函数

<b>ref(\$r) eq 'SCALAR'</b>	标量引用
<b>ref(\$r) eq 'ARRAY'</b>	数组引用
<b>ref(\$r) eq 'HASH'</b>	hash 引用
<b>ref(\$r) eq 'CODE'</b>	子程序引用

## 模块

### 使用模块

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

### 创建模块

```
# MyModule.pm
package MyModule;
use Exporter 'import';
our @EXPORT_OK = qw(helper);
sub helper { return "help"; }
1; # 模块必须返回真值
```

### 常用核心模块

<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>	命令行选项解析
<b>JSON</b>	encode_json, decode_json
<b>LWP::Simple</b>	get(\$url) — 简易 HTTP 客户端
<b>Data::Dumper</b>	数据结构调试输出
<b>Carp</b>	croak, confess — 更友好的错误信息

### CPAN

```
cpan install Module::Name # 从 CPAN 安装
cpanm Module::Name        # cpanminus (更快)
perldoc Module::Name      # 查阅模块文档
```