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）

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

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

运算符

比较运算符

`==, !=, <, >, <=, >=`	数值比较
`eq, ne, lt, gt, le, ge`	字符串比较
`<=>`	数值飞船运算符（返回 -1、0、1）
`cmp`	字符串飞船运算符
`=~`	正则匹配/绑定
`!~`	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);             # 11
```

逻辑运算符

`&& / and`	逻辑与（低优先级：`and`）
`|| / or`	逻辑或（低优先级：`or`）
`//`	已定义或（左侧已定义则返回左侧）
`! / not`	逻辑非
`? :`	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

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

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

常用模式

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

文件测试

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

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

`ref($r) eq 'SCALAR'`	标量引用
`ref($r) eq 'ARRAY'`	数组引用
`ref($r) eq 'HASH'`	hash 引用
`ref($r) eq 'CODE'`	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

`List::Util`	sum, max, min, reduce, any, all
`File::Basename`	basename, dirname, fileparse
`File::Path`	make_path, remove_tree
`Getopt::Long`	命令行选项解析
`JSON`	encode_json, decode_json
`LWP::Simple`	get($url) — 简易 HTTP 客户端
`Data::Dumper`	数据结构调试输出
`Carp`	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
```