REFERENSI CEPAT PERL

Variabel, regex, file I/O, referensi, dan modul esensial

Dasar

Hello World

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

Jalankan Perl

```
perl script.pl           # jalankan file
perl -e 'print "hi\n"'   # jalankan inline
perl -ne 'print' file    # proses file baris per baris
```

Komentar & Dokumentasi

```
# komentar satu baris
=pod
Dokumentasi POD multi-baris
=cut
```

Variabel

Sigil

`$scalar`	Nilai tunggal (string, angka, referensi)
`@array`	Daftar scalar berurutan
`%hash`	Pasangan key-value
`$array[0]`	Akses satu elemen array (konteks scalar)
`$hash{key}`	Akses satu nilai hash (konteks scalar)

Variabel Scalar

```
my $name = "Perl";        # string
my $version = 5.40;       # angka
my $count = 42;           # integer
my $undef;                # tidak terdefinisi (undef)
my $combined = "$name v$version";  # interpolasi
```

Konteks

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

Variabel Spesial

`$_`	Variabel default (topik)
`@_`	Argumen subrutin
`$!`	Pesan error sistem
`$@`	Error dari eval
`$0`	Nama program
`@ARGV`	Argumen command-line
`%ENV`	Variabel environment

Operator

Operator Perbandingan

`==, !=, <, >, <=, >=`	Perbandingan numerik
`eq, ne, lt, gt, le, ge`	Perbandingan string
`<=>`	Spaceship numerik (mengembalikan -1, 0, 1)
`cmp`	Spaceship string
`=~`	Cocokkan / bind regex
`!~`	Kecocokan regex dinegasikan

Operator String

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

Operator Logika

`&& / and`	AND logika (prioritas rendah: `and`)
`|| / or`	OR logika (prioritas rendah: `or`)
`//`	Defined-or (mengembalikan kiri jika terdefinisi)
`! / not`	NOT logika
`? :`	Kondisional ternary

Alur Kontrol

Kondisional

```
if ($x > 0) { print "positif\n"; }
elsif ($x == 0) { print "nol\n"; }
else { print "negatif\n"; }
print "ya\n" if $condition;     # if postfix
print "tidak\n" unless $condition;  # unless postfix
```

Loop

```
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(); }
```

Kontrol Loop

`next`	Lanjut ke iterasi berikutnya (seperti `continue`)
`last`	Keluar dari loop (seperti `break`)
`redo`	Mulai ulang iterasi saat ini
`next LABEL`	Lanjut ke iterasi berikutnya pada loop berlabel
`last LABEL`	Keluar dari loop berlabel

Given / When

```
use feature 'switch';
given ($status) {
    when ("ok")    { say "sukses"; }
    when ("error") { say "gagal"; }
    default        { say "tidak diketahui"; }
}
```

Subrutin

Subrutin Dasar

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

Parameter Default & Named

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

Referensi Subrutin

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

Prototipe & Signature

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

Regex

Pencocokan

```
if ($str =~ /pattern/) { print "cocok\n"; }
if ($str =~ /(\d+)/) { print "angka: $1\n"; }
my @matches = ($str =~ /(\w+)/g);  # semua kecocokan
```

Substitusi

```
$str =~ s/old/new/;      # kemunculan pertama
$str =~ s/old/new/g;     # global (semua kemunculan)
$str =~ s/^\s+|\s+$//g;  # trim whitespace
(my $clean = $str) =~ s/\W//g;  # salinan non-destruktif
```

Modifier

`/i`	Case-insensitive
`/g`	Global (semua kecocokan)
`/m`	Multi-baris (`^` dan `$` cocok dengan batas baris)
`/s`	Satu baris (`.` cocok dengan newline)
`/x`	Extended (izinkan whitespace dan komentar)

Pola Umum

`\d, \D`	Digit / non-digit
`\w, \W`	Karakter kata / non-kata
`\s, \S`	Whitespace / non-whitespace
`\b`	Batas kata
`(?: ... )`	Grup non-capture
`(?<name> ... )`	Capture bernama (akses via `$+{name}`)

File I/O

Buka & Baca

```
open(my $fh, '<', 'data.txt') or die "Tidak bisa buka: $!";
while (my $line = <$fh>) {
    chomp $line;
    print "$line\n";
}
close($fh);
```

Tulis & Tambahkan

```
open(my $fh, '>', 'out.txt') or die "Tidak bisa buka: $!";
print $fh "Hello\n";
close($fh);
open(my $fh, '>>', 'log.txt') or die "Tidak bisa buka: $!";
print $fh "entry\n";
close($fh);
```

Slurp Seluruh File

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

Uji File

`-e $path`	File ada
`-f $path`	Adalah file biasa
`-d $path`	Adalah direktori
`-r / -w / -x`	Dapat dibaca / ditulis / dieksekusi
`-s $path`	Ukuran file dalam byte (0 jika kosong)
`-z $path`	File berukuran nol

Array & Hash

Array

```
my @arr = (1, 2, 3, 4, 5);
push @arr, 6;              # tambahkan ke akhir
my $last = pop @arr;       # hapus elemen terakhir
my $first = shift @arr;    # hapus elemen pertama
unshift @arr, 0;           # tambahkan ke depan
my @slice = @arr[1..3];    # slice
```

Fungsi Array

`scalar @arr`	Jumlah elemen
`push / pop`	Tambah/hapus dari akhir
`shift / unshift`	Hapus/tambah dari awal
`splice(@a, 2, 1)`	Hapus 1 elemen pada indeks 2
`sort @arr`	Urutkan secara alfabetis
`reverse @arr`	Balik urutan
`grep { /pat/ } @arr`	Filter berdasarkan pola
`map { $_ * 2 } @arr`	Transformasi setiap elemen
`join(',', @arr)`	Gabungkan menjadi string

Hash

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

Iterasi Hash

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

Referensi

Membuat Referensi

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

Dereference

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

Struktur Data Kompleks

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

Fungsi ref()

`ref($r) eq 'SCALAR'`	Referensi ke scalar
`ref($r) eq 'ARRAY'`	Referensi ke array
`ref($r) eq 'HASH'`	Referensi ke hash
`ref($r) eq 'CODE'`	Referensi ke subrutin

Modul

Menggunakan Modul

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

Membuat Modul

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

Modul Core Umum

`List::Util`	sum, max, min, reduce, any, all
`File::Basename`	basename, dirname, fileparse
`File::Path`	make_path, remove_tree
`Getopt::Long`	Parsing opsi command-line
`JSON`	encode_json, decode_json
`LWP::Simple`	get($url) — HTTP client sederhana
`Data::Dumper`	Debug dump struktur data
`Carp`	croak, confess — pesan error lebih baik

CPAN

```
cpan install Module::Name        # install dari CPAN
cpanm Module::Name               # cpanminus (lebih cepat)
perldoc Module::Name             # baca dokumentasi modul
```