	Regra	Exemplo *Correcto*	Exemplo Incorrecto	Classe	Secção
	use strict; use warnings;			Obrigatório	layout
	Não mais de 80 colunas			Obrigatório	layout
	Separar palavras de controlo de parêntesis	`for (@resuts) {  print;}` `if ( $not )`	`for(@results)` `if($not)`	Obrigatório	layout
	Não separar o nome da função do parêntesis	`func( $str );`	`func (@$str)`	Obrigatório	layout
76	Usar () na invocação de nossas funções, mas sem o &	`do_print()`	`do_print`	Obrigatório	subroutines

10	ponto e vírgula após todos "*statements*"	`while (<>) {`
		`  sleep 1;`
		`  print;`
		`}`
			`while (<>) {`
			`  sleep 1;`
			`  print`
			`}`
				Obrigatório	layout

14	Usar Espaços horizontais e verticais para formar *parágrafos* de código; Comentar início de parágrafo			Sugestão	layout

17	Alinhar itens correspondentes	`my %expansion = (`
		`  'um'  =>  1,`
		`  'dois' =>  2,`
		`  'tres' =>  3,`
		`);`
			`my %expansion = (`
			`  'um'=>1,`
			`  'dois'=>2,`
			`  'tres'=>3,`
			`);`
				Obrigatório	layout

19	Quebrar expressões longas antes de um operador e indentar o operador	`push @steps, $steps[-1]`
		`  + $radius + 1`
		`  + $orbita`
		`  * ( $pi + 3)`
		`  - $space`
		`  ;`
			`push @steps, $steps[-1]+`
			`  $radius + 1 +`
			`  $orbita * ( $pi + 3)`
			`  - $space`
			`  ;`
				Sugestão	layout

20	Quebrar atribuições antes do operador	`$pred = $avrg`
		`  + $one * $fuzzy;`
		`OU`
		`$pred`
		`  = $avrg + $one * $fuzzy;`
			`$pred = $avrg + $one`
			`  * $fuzzy;`
				Sugestão	layout

21	*underscore* para identificadores	`a_great_variable`	`aGreatVariable`	Obrigatório	naming
22	subrotinas e variáveis	`minúsculas, [a-z_]`		Obrigatório	naming
22	*packages* e classes	`Começa sempre com maiúscula, depois` `Mistura [a-zA-Z_]`		Obrigatório	naming
22	constantes	`MAIUSCULAS [A-Z_]`		Obrigatório	naming

25	subrotinas
	verb_noun
	verb_noun_preposition
	verb_noun_participle
		`sub get_record;`
		`sub get_record_for;`
		`sub build_profile_using;`
				Sugestão	naming

25	variáveis adjective_noun	`my $source;` `my $tree_node;`		Sugestão	naming

26	arrays plural hashes singular	`my @events;` `my %handler_for;`		Obrigatório	naming
27	*underscore* "_" apenas para uso interno			Obrigatório	naming
29	'' (plica, plica) para strings não interpoladas	`my $pong = 'pong';`		Obrigatório	Values
	não interpolar variáveis; usar sprintf			Sugestão	Values
30	não usar plica plica para string vazia	`q{}`		Sugestão	Values

36	Não usar *package variables*
	$_, @ARGV, $AUTOLOAD
	$1, $2, $3
	$a, $b, @a, @b (*imunes* ao use strict!!!)
		`package Customer;`
		`my %opt;`
		`sub terse { $opt{'terse'} = shift(@_); }`
		`...`
		`package main;`
		`$Customer::terse( 1 );`
			`package Customer;`
			`our %opt;`
			`...`
			`package main;`
			`$Customer::opt{'terse'} = 1;`
				Obrigatório	variables

37	usar *local*	`use YAML;` `local $YAML::Ident = 4;`	`use YAML;` `$YAML::Ident = 4;`	Obrigatório	variables
40	Cuidado com modificações ao $_ Quase sempre é um alias			Informativo	variables
41	usar índices negativos para aceder ao fim do array	`$ultimo = $arr[-1];`	`$ultimo = $arr[$#arr-1];`	Sugestão	variables
54	Não usar do ... while -> NÃO é um loop! (next; last não funcionam como esperados)			Obrigatório	control structure
57	Dar *label*s a todos os loops dos quais saltamos	`INPUT:` `while (...) { ....; next INPUT;}`		Sugestão	control structure
77	Não dar os mesmos nomes às rotinas das rotinas do sistema: nunca saberemos qual será invocada	`sub found_map {...}` `sub close_the {...}`	`sub map { print 'You found a map'}` `sub close { print "The @_ is close`	Obrigatório	subroutines
78	sempre expandir o @_	`my ($text, $cols) = @_;` `my $gap = $cols – length($text);`	`my $gap = $_[1] - $_[0];`	Sugestão	subroutines

79	preferir atribuição de listas em vez de shift
	EXCEPTO para class instance
		`my ($um,$dois) = @_;`
		`my $s = shift;`
		`my $c = shift;`
			`my $um =shift;`
			`my $dois = shift;`
					subroutines

88	colocar _ref em variáveis que são referências	`my ($text, $arg_ref) = @_;`		Obrigatório	variables
89	terminar uma rotina SEMPRE com um return;			Obrigatório	subroutines

89	usar um return vazio e nunca um return undef	é undef em *scalar context*
		é lista vazia em *list context*
			`sub ret_u { return undef;  }`
			`for my $k ( ret_u() ) {`
			`  print "oinc\n" ;`
			`}`
				Obrigatório	subroutines

91	open SEMPRE com 3 argumentos Usar File::IO	`open $in, '<', $in_file or croak;` `open $out, '>', $out_file or croak;`	`open ">file.txt"`	Obrigatório	I/O
93	usar while (<>) e não for (<>) (o for é em contexto de lista, lê logo ficheiro)			Obrigatório	I/O
	atribuir "nomes" às variáveis do search	`my $primeiro = $1;`		Obrigatório	
12	TAB a 4 espaços			Obrigatório	layout
13	Não usar hard TABs			Obrigatório	layout

16	else na linha a seguir	`if ($ok) {`
		`  ...`
		`}`
		`else {`
		`  ...`
		`}`
			`if ($ok) {`
			`  ...`
			`} else {`
			`  ...`
			`}`
				Obrigatório	layout

	use croak em vez de die				

34	Capitalizar identificadores SEM espaços para heredocs	`print <<"END_LIST";`
		`get name`
		`gez size`
		`END_LIST`
				Informativo	values

34	Usar heredocs para strings de multiplas linhas			Informativo	values

35	Não usar palavras sem aspas (barewords).
	Causam ambiguidade
		`print $sqrt{'N'}`	`for my $n (2,3,4) {`
			`  print $sqrt{n};`
			`}`
			`print Hello . World;`
			`# => HelloWorld`
				Obrigatório	values

45	Evitar ciclos for tipo c	`for my $n (4..$max) {`
		`....`
		`}`
			`for (my $i=4; $n<=$max; $n+=2) {`
			`...`
			`}`
				Obrigatório	control structure

54	usar operador terenário em formato tabular em vez de if - else em cascata	`$s = $name eq $EMPTY_STR  ? 'Customer'`
		`  : $name =~ m/(Mr|Miss)/ ? $1`
		`  : $name =~ m/(Phd|Dr)/  ? $1`
		`  :                          $name`
		`  ;`
				Informativo	control structure

	Documentação				
60	Documentação pod no mesmo ficheiro que o código			Obrigatório	documentation
60	Documentação pod inline			Obrigatório	documentation

62	Usar =for para comentários grandes	`=for Rationale:`
		`  (não existem tags pod com ":",`
		`portanto são ignorados)`
				Informativo	documentation

	Markov Language			Obrigatório	documentation
70	Evitar evals sobre strings `(BEGIN, END e DATA não é repetido!)`			Informativo	Built-in function
75	Usar List::Util e Scalar::Util List::MoreUtils			Informativo	Built-in function
100	Não esquecer do IO::Prompt para programas de linha de comando	`my $passwd = prompt 'Passwd: '` `  , echo=>'*';`		Informativo	I/O
101	Smart::Comments			Informativo	I/O
103	Quando possível, usar -> para de-referenciar	`$list_ref->[0];`	`${$list_ref}[0];`	Obrigatório	References
	Perlcritic -3	**vim:** <ESC> _pc **Linha de comando:** perlcritic, perlcritic5, perlcritic8,perlcritic11		Obrigatório	

	perltidy	vim: <ESC> _t Funciona em modo visual (blocos)		Obrigatório	
	Usar sempre a flag /x (ignora espaços)	m{ ' #opening single quote [^\\']* #any non-special char (?: #all of \\ . # escaped char [^\\']* #any non-speacial)* #zero or more ' #closing single quote }x		Informativo	Regexes
	Usar sempre a flag /m Muda comportamento de ^ e $			Informativo	Regexes
	Usar sempre o /s o ponto “.” apanha mesmo TODOS os chars (sem, não apanha o \n \r)			Informativo	Regexes
	usar (...) quando se quer mesmo o resultado senão (?:)			Informativo	Regexes
	Tentar NUNCA usar o .* e .*? Muito ineficiente	$src =~ m{\G ([^;]+) ;}gcx (apanha tudo até primeiro ;)		Informativo	Regexes
	Não usar $` $& $' (impactam todas as regex)			Obrigatório	Regexes