

Perl — best practices

perl.it <http://www.perl.it/>

Italian Perl Workshop 2008

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie

Grazie a Damian Conway

Questo intervento è ispirato a “Perl Best Practices”, di Damian Conway, edizioni O’Reilly

Perl — best practices

perl.it

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie

Obiettivi

- ▶ Robustezza
- ▶ Efficienza
- ▶ Manutenibilità

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie

Scrivi sempre codice come se
la persona che lo deve
mantenere fosse un violento
psicopatico che sa dove abiti

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie

```
1 my@result=(1,2,3);
2 for my$result (@results) {print_sep;print ($result)}
3 while(my_func $param,$param2) {
4 my$stuff=$x->{$keys[($x-$y)/2]};
5 if($stuff<$target) {$x=$y} else {$y=$x}
```

Spaziatura

```
1 my @result = (  
2     1,  
3     2,  
4     3,  
5 );  
6  
7 for my $result (@results) {  
8     print_sep();  
9     print $result;  
10 }  
11  
12 while (my_func($param,$param2)) {  
13     my $stuff = $x->{ $keys[ ($x-$y)/2 ] };  
14     if ($stuff < $target) {  
15         $x = $y;  
16     }  
17     else {  
18         $y = $x;  
19     }  
20 }
```

Perl — best
practices

perl.it

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie



- ▶ date nomi comprensibili e non ambigui
(`$totale_finora`, `$utente_attuale`, `&è_valido`)
- ▶ non abbiate paura dei nomi lunghi
- ▶ separate con `_` e non CamelCase
- ▶ hash al genitivo singolare, array al nominativo plurale
- ▶ non abbreviate troppo
- ▶ cominciate i “nomi privati” con `_`

- ▶ usate `'virgolette_semplici'` tranne quando `"dovete_$interpolare\n"`
- ▶ non abbiate paura di `q()` e `qq()`
- ▶ in particolare, `q[]`, `q[]`, `q[,]`, `qq[\t]`
- ▶ usate “here-doc” per blocchi di testo

- ▶ dichiarate sempre le variabili con **my**
- ▶ non usate **local** se non sapete *esattamente* perché
- ▶ non toccate le variabili globali speciali (in particolare @ARGV)
- ▶ **use** English '-no_match_vars'
- ▶ *usate local* su `$_` e altre variabili “punteggiatura”
- ▶ `$array[-1]`, non `$array[$#array]`

- ▶ usate `if` posposto solo per semplici istruzioni, tipicamente controlli nei cicli:

```
next CICLO if !mi_serve($elemento);
```

- ▶ usate ancora meno gli altri controlli posposti (**unless**, **while**, **for**)

- ▶ evitate il `for`(`init`; `cond`; `step`) alla C

- ▶ date sempre un nome alla variabile di iterazione:

```
for my $persona_attuale (@persone) { ... }
```

- ▶ usate `map` per trasformare le liste (ma non in-place!)

- ▶ usate `grep` e `List::Util::first` per estrarre elementi

- ▶ usate *dispatch tables* invece che cascate di `if`

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie

Dispatch tables

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie

```
1 {
2 my %comandi=(
3     start => \&avvia_processo_di_calcolo,
4     stop  => \&termina_processo_di_calcolo,
5     status => \&esamina_stato_processo,
6 );
7
8 sub esegui_comando {
9     my ($comando)=@_;
10    if (exists $comandi{$comando}) {
11        $comandi{$comando}->();
12    }
13    else {
14        carp qq{Comando "$comando" ignoto.\n};
15    }
16 }
17 }
```

Controlli di flusso (2)

- ▶ usate i comandi di controllo cicli (**next**, **last**, **continue**, **redo**)
- ▶ etichettate sempre i cicli in cui usate quei comandi:

```
1 CLIENTE:  
2 while (my $cliente_corrente =  
3         $clienti->next()) {  
4     next CLIENTE  
5     if ! $cliente_corrente->attivo;  
6     usa_cliente($cliente_corrente);  
7 }
```

Documentazione e commenti

- ▶ usate POD per la documentazione utente
- ▶ usate i commenti per la documentazione interna
- ▶ spiegate gli algoritmi
- ▶ annotate i trucchi e le finezze
- ▶ “incremento il contatore” non è documentazione

- ▶ studiate la “Orcish manoeuvre” e la “Schwartzian transform”
- ▶ **reverse** è molto utile
`(for my $countdown (reverse 0..100) { ... })`
- ▶ spaccettare dati:
 - ▶ **unpack** se lunghezza fissa
 - ▶ **split** se lunghezza variabile
 - ▶ `Text::CSV_XS` se più complicati
- ▶ è quasi sempre possibile evitare **eval** 'stringa'
- ▶ usate **glob** ('*'), non `<*>`
- ▶ studiate e usate `Scalar::Util`, `List::Util`, `List::MoreUtils`

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie

- ▶ `fai_qualcosa(@argomenti)`, **non**
`&fai_qualcosa @argomenti`
- ▶ per prima cosa, spaccettate `@_`
- ▶ se la sub prende più di un paio di argomenti,
passateli per hash
- ▶ attenti al contesto nei vostri **return**
- ▶ **return;** garantisce che il risultato sia falso, anche in
contesto lista
- ▶ mettete sempre un qualche **return** alla fine delle
sub

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie



```
1 open my $fh, '<', $nomefile  
2 or croak qq{Impossibile aprire "$nomefile": $!};
```

invece che

```
1 open FH, $nomefile;
```



```
1 open my $fh, '<', $nomefile  
2 or croak qq{Impossibile aprire "$nomefile": $!};
```

invece che

```
1 open FH, $nomefile;
```

- ▶ filehandle lessicali
- ▶ **open** a 3 argomenti
- ▶ controllate sempre il valore di ritorno

- ▶ **while** (**my** \$line=<\$fh>), **non for**
- ▶ evitate il più possibile di caricare l'intero file in memoria
- ▶ se proprio dovete, usate `File::Slurp`
- ▶ **print** {\$logfile} \$messaggio, **non print** \$logfile \$messaggio
- ▶ studiate `IO::Handle` e famiglia

- ▶ non usate *mai* reference simboliche
`("${nome}_tipo"=5)`
- ▶ `$list_ref->[$indice]`, **non**
`$$list_ref[$indice]`
- ▶ `@{$hoa{$chiave}}`, **non** `@$hoa{$chiave}`

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie

Espressioni regolari

- ▶ usate sempre `/xms`
- ▶ `\A` e `\z` per inizio e fine *stringa*
- ▶ `^` e `$` per inizio e fine *riga*
- ▶ `m{...}smx`, non `/.../`
- ▶ `\p{Uppercase}\p{Alphabetic}`, non `[A-Z] [a-zA-Z]`
- ▶ attenti alla differenza tra `.*` e `.*?`

```
1 my ($nome, $valore) =
2   ($linea =~ m{\A (\w+) \s*=\s* (.*) \z}smx)
```

invece che

```
1 $linea =~ m{^( \w+) \s*=\s* (.*) $};
2 my ($nome, $valore) = ($1, $2)
```

- ▶ **studiate** `Regexp::Common`

- ▶ lanciate eccezioni, non impostate flag o valori di ritorno
- ▶ studiate `autodie`
- ▶ studiate `Exception::Class` o simili
- ▶ usate `Carp`

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie

- ▶ **usate** Moose
- ▶ **non usate oggetti dove non servono**
- ▶ **usateli *sempre* dove servono**
- ▶ `$oggetto->metodo(@args)`, **non**
`metodo $oggetto @args`
- ▶ **evitate** AUTOLOAD

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie

- ▶ non ripetete mai il codice: fattorizzate *sempre*
- ▶ nascondete le implementazioni dentro moduli
- ▶ attenti ai nomi che esportate
- ▶ evitate di esporre variabili o altro stato globale
- ▶ non scrivete moduli: usate quelli in CPAN

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie

- ▶ scrivete test, e usateli
- ▶ usate un debugger
- ▶ usate un sistema di controllo delle versioni
- ▶ non fate ottimizzazioni alla cieca – misurate i tempi
(`Benchmark`, `Devel::NYTProf`)
- ▶ non fate ottimizzazioni alla cieca – misurate l'occupazione di memoria (`Devel::Size`)

Spaziatura

Nomi

Valori

Variabili

Flusso

Documentazione

Built-in

Subroutine

I/O

References

Regex

Eccezioni

OOP

Moduli

Varie