

CS306: Introduction to Perl

Control Structures (Chapter 3)

U. of Alabama at Birmingham
Dept. of Computer & Information Sciences

Slide 1

Control Structures

Conditional structures Looping structures

Slide 2

Conditional Structures - *if..elsif..else*

- `if (some condition) {`
 `...`
 `} elsif (some other condition) {`
 `...`
 `} else {`
 `...`
 `}`

Slide 3

conditions

- All the usual comparators
 - Numeric: `>` `<` `>=` `<=` `==` `!=`
 - String: `gt` `lt` `ge` `le` `eq` `ne`
- `if ($number > 10) { ... }`
- `if ($word eq 'secret') { print "You guessed it!"; }`

Slide 4

More conditions

- Ultimately, a condition is simply an expression that evaluates into either a true value or a false value. This is handy when using functions...
- `if (defined($num)) { # work with $num }`
- `if (! (open FILE, "file.txt")) {
 print "Can't open file.";
 exit(-1);
}`

Slide 5

Complex Conditions

- `if ($num > 10) && ($word ne $password) {
 print "Too many incorrect attempts.\n";
 exit;
}`
- `&&`, `||`, `!`
- `and`, `or`, `not`
- `(($num > 10) and ($word ne $password))`
- `($num > 10 and $word ne $password)`

Slide 6

Cascading Conditions

- `if ($num > 10) { # what's wrong?
 print "That's a big number.";
}`
`if ($num > 0 and $num < 10) {
 print "Not too bad.\n";
}`
`if ($num < 0) {
 print "I wish the number were bigger.\n";
}`

Slide 7

Cascading Conditions

- `if ($num > 10) { # better
 print "That's a big number.";
} elsif ($num > 0 and $num < 10) {
 print "Not too bad.\n";
} elsif ($num < 0) {
 print "I wish the number were bigger.\n";
}`
- Don't make your program do extra work

Slide 8

Conditional Structures – *unless*

- `unless ($timeleft == 0) {
 print "Keep playing.\n";
}`
- `unless (condition)` is the same thing as saying `if (not condition)`
- “If I don't get at least a 70, I'll fail the class.” vs.
“Unless I get at least a 70, I'll fail the class.”

Slide 9

Expression Modifiers

- But how would we say the following in Perl?

“I'll fail the class if I don't get at least a 70.” or
“I'll fail the class unless I get at least a 70.”

Slide 10

Expression Modifiers

- `print “I failed.” unless $num >= 70;`
- `print “I failed.” if not $num > 70;`
- CAREFUL OF PRECEDENCE!! (page 36)
- `print “I failed.” if ! $num > 70; # doesn't work`
- `print “I failed.” if not $num > 70; #works`
- `print “I failed.” if ! ($num > 70); #works`

Slide 11

Looping Structures – *while*

- `while (condition) {
 action
}`
- `my $num = 10;
while ($num > 0) {
 print “$num\n”;
 $num--;
}`

Slide 12

while and <STDIN>

- `$input = <STDIN>;`
 - Gets a single line of input from user
- `while ($input = <STDIN>) {`
 - `print $input;`
 - `}`
 - Will keep doing this a line at a time until end-of-file.
That's Ctrl-D for *nix, Ctrl-Z for Windows.

Slide 13

while and <STDIN>

- `while ($input = <STDIN>) {`
 - `print $input;`
 - `}`
- Very common pattern so Perl gives us a shortcut:
- `while (<STDIN>) { # puts STDIN into $_`
 - `print $_;`
 - `}`
- `$_` is magic – the “default variable”

Slide 14

while and <STDIN>

- `while (<STDIN>) { # puts STDIN into $_`
 - `print $_;`
 - `}`
- And this is so common, that `print()` defaults to printing `$_` if nothing provided to it.
- `while (<STDIN>) {`
 - `print;`
 - `}`

Slide 15

while and <STDIN>

- `while (<STDIN>) {`
 - `chomp $_; # let's do some cleanup`
 - `print $_;`
 - `}`
- `chomp` acts just like `print`, defaulting to `$_`;
- `while (<STDIN>) {`
 - `chomp;`
 - `print;`
 - `}`

Slide 16

while and <STDIN>

- while (\$line = <STDIN>) {
 chomp \$line;

}
- This common idiom often becomes:
- while (<STDIN>) {
 chomp;

}

Slide 17

C/C++/Java style *for*

- for (\$i = 0; \$i < 10; \$i++) {
 # this code will run 10 times
 ...
}
- Not so common, because it is equivalent to:
 - \$i = 0;
 while (\$i < 10) {
 ...
 \$i++;
 }

Slide 18

Perl's *foreach* list iterator

- foreach \$num (2..10) { # generic list
 print "I have \$num apples.\n";
}
- foreach \$day (@daysofweek) { # array
 print "My favorite day is \$day.\n";
}
- The keywords *for* and *foreach* are interchangeable

Slide 19

do..while and *do..until*

- do { action } while (condition);
- do { action } until (condition);
- do {
 print "Guess a number from 1-10: ";
 \$num = <STDIN>;
} until (\$num == \$secret);

Slide 20

next and last

- Use next and last to break out of loops
- while (\$i < 10) {
 ...
 if (\$done_early) { last;}
}
- foreach \$line (@lines) {
 if (\$line eq "don't print me please") { next; }
 print "\$line\n";
}