

CS306: Introduction to Perl

HW1 Review

Elegant Loops

- Problem: You want to write a loop that you guarantee runs at least once
- Common mistake:

```
do it once here          # copy 1 here
while (not finished)
  keep doing it         # copy 2 here
ask if finished
end while
```

- Leads to duplicated code!

Elegant Loops

- One better way:

```
finished = 0
while (not finished)
  run the code
  ask if finished
end while
```

Elegant Loops

- Another better way:

```
do
  code goes here
ask if finished
until (finished)
```

Elegant Loops

- Another loop problem, infinite loops:

```
while (1)
  do something
  rely on internal code to break out
end while
```

- It works, but is hard for code readers to see your intent

Elegant Loops

- One better way:

```
while (exit condition not met)
  do something
end while
```

Avoid case/switch-like code

- Problem: A user has several choices, and you want to do a similar task with each choice
- Example: The exchange rate problem

7

Avoid case/switch-like code

- Common mistake:
if (choice is EUR)
 print output for EUR
elsif (choice is CAN)
 print output for CAN
elsif (choice is JPN)
 print output for JPN
and so on.....

8

Avoid case/switch-like code

- Now for actual Perl code representing the mistake
if (\$choice eq 'canada') {
 print "\$us USD is \$conv CND\n";
} elsif (\$choice eq 'europe') {
 print "\$us USD is \$conv EUR\n";
} and so on....

9

Avoid case/switch-like code

- If you see that pattern emerge in your code, **refactor your code!**
- One better way:

```
print "$us USD is " .  
      ($us * $rates{$choice}) .  
      " $choice.\n";
```
- No loop required at all. What if we had all 178 currency exchange rates?!

10

Predeclaring variables

- Completely unnecessary
- Common extra work:

```
$finished = 0;  
while (! $finished) {  
  do something  
  if (some condition) $finished = 1  
}
```

11

Predeclaring variables

- Completely unnecessary
- Common extra work:

```
$finished = 0;  
while (! $finished) {  
  do something  
  if (some condition) $finished = 1  
}
```

12

Predeclaring variables

- Completely unnecessary
- Common extra work:

```
while (! $finished) {  
  do something  
  if (some condition) $finished = 1  
}
```
- Works exactly the same

13

Predeclaring variables

- Another common extra work:

```
$myvariable;  
.....  
# later on...  
  
$myvariable = somevalue;
```
- Why bother with the predeclare?

14

Case Insensitivity

- Use `lc()` or `uc()` to provide case-insensitivity
- One way:

```
my $rates = (usd => 1, can => 0.77);  
chomp (my $choice = <STDIN>);  
$choice = lc($choice);
```

15

Answers to Questions

- Everyone pretty good on the first one
- Many people forgot to mention that arrays are ordered whereas hashes have no particular order

16

Answers to Questions

- On the second one, parts c and d caused trouble
- Part c: list on left, scalar value on right

```
($foo, $bar) = 5;
```
- \$foo gets 5, \$bar is undef. Multiple people reported a compiler error!

17

Answers to Questions

- Part d: single scalar variable on left, list of values on right

```
$foo = (1,3,5);
```
- \$foo gets 5
- Why? Just because. :-)

18