

Re: nested "if"

Source: <http://coding.derkeiler.com/Archive/Perl/perl.beginners/2004-07/0136.html>

From: Randy W. Sims (ml-perl_at_thepierianspring.org)

Date: 07/03/04

Date: Sat, 03 Jul 2004 04:38:05 -0400
To: "Michael S. Robeson II" <popgen23@mac.com>

On 7/2/2004 10:25 PM, Michael S. Robeson II wrote:

> Well yeah, the indentation makes it much more clearer. However, this
> does not help me understand how the nested "if" statements are working.
> Which of the two "if" statements gets evaluated first? I am trying to
> figure out "in english" what the "if" statements are actually doing. Is
> it saying:
>
> "If a line begins with ">bla-bla" and if \$seq (which appears no
> where else in the code other than " \$seq="" ") exists assign it to the
> hash "pro" with the name "bla-bla"."
>
> So my question is how does the inner if statement work when seq="" is
> out side that "if" statement?
>
> Is the outer "if" statement evaluated first then the inner? Because
> how does the inner "if" statement know what "\$seq" is?
>
> I am probably not making any sense but I am trying to figure out
> mechanically how the perl interpreter knows what to do in the context of
> the nested if statements.

Tidied up a little more:

```
my( %pro, @names);
my( $name, $seq, $k );
while (defined( my $line = <DATA> )) {
    if ($line =~ /^>(.)/) {
        if ($seq) {
            $pro{$name} = $seq;
            $seq = "";
        }
        $name = $1;
        $name =~ s/s//g;
        push @names, $name;
        $k++;
    } else {
        chomp( $line );
    }
}
```

```
    $seq .= $line;  
  }  
}
```

This code deals with multi-line sequences, putting multiple lines together until a sequence is complete. The 'else' part of the outer 'if' does the accumulation of multiple lines into a sequence. The 'if' part determines that a sequence is complete, captures some type of name from the sequence, stores the complete sequence in the '%pro' hash, and pushes the name onto a '@names' array. I'm guessing '\$k' keeps tally of the number of sequences; it's not clear if that is necessary since `scalar @names` possibly will provide the same info. It's also unclear why there is a '@names' array that mostly duplicates `keys %pro`

I think where you're--understandably--getting confused is that most of those variables are global. That's made more explicit in my strictified rewrite above. It could probably be rewritten better if we knew the exact format of the data being read.

Randy.