



The University of North Carolina at Chapel Hill

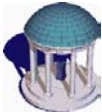
---

COMP 144 Programming Language Concepts  
Spring 2003

## And Yet More Scripting with Perl

David Stotts

1



## Block Comments

---

**To comment out a lengthy chunk of code**  
**Use POD capabilities (Plain Old Documentation)**

```
=comment
if ($thresh < 10) {
    $a = 5;
} elsif ($thresh <20) {
    $a = 16;
    print $a;
} else {
    die "oh no... I didn't expect this";
}
=cut
```

2



## GUI in Perl

---

### Modules

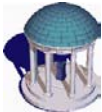
```
Use win32::GUI ; # imports code and names
```

### CPAN

### Comprehensive Perl Archive Network

<http://www.cpan.org>

3



## References

---

### “Pointers” (not bibliography)

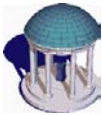
### References are scalar type

### References are means of addressing Perl entities

```
$a = "mamma mia" ;  
@arr = (10,20,30);  
%h2 = ("first", 25, "second", "2nd", "third", 3.14);  
$ra = \"a\" ;  
$rarr = \"@arr\" ;  
$rh2 = \"%h2\" ;
```

```
Print $a, $$ra ; # same thing twice  
$$rarr[0] = 15; # like $($rarr)[0]
```

4



## More References

---

**Arrays hold scalars**

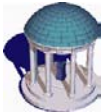
**References are scalar**

**So you can have an array of references**

– Multi-dimensional array

```
$rar = \@ar ;  
print $rar->[1] ;  
print $$rar[1] ;  
$ar[0] = \@b ; # assume @b has some integers in it...  
$ar[1] = \@c ;  
$ar[2] = \@d ;  
print $d[7], $ar[2][7] ; # both the same  
print $rar->[2]->[7], $$rar[2][7], $ar[2]->[7]; # all the same
```

5



## Yet More References

---

**Can pass refs as parameters to subroutines**

```
@ar1 = (1,2,3,4,5) ;  
@ar2 = (7,8,9,10) ;  
$s = addArrays(\@ar1, \@ar2);  
  
sub addArrays {  
    my ($rar1, $rar2) = @_ ;  
    $sum = $rar1[0] + $rar2[0] ;  
    # do whatever else...  
}
```

6