

Perl Control Structures

The HTML lecture notes have more details and examples

EIW - Perl Control Structures 1

vs. C++

- Perl supports many of the same basic control structures as C/C++:
 - if, if-else
 - while
 - do while
 - for loop
- Perl also supports a few more...
- Perl does not support **switch**

EIW - Perl Control Structures 2

Perl `if`

```
if (expression) {  
    statement;  
}
```

- The value of *expression* determines whether or not *statement* is executed.
- The curly braces are **not** optional – they are required even if there is only one statement!

EIW - Perl Control Structures 3

True and False

- In Perl, an expression that evaluates to a non-zero number is considered TRUE.
- The following are FALSE:
 - 0
 - ""
 - "0"
- Everything else is considered TRUE

EIW - Perl Control Structures

4

The special Perl value *undef*

- undef is also FALSE
 - converted to a string the value is "".

```
if ($age{"Jones, Mary"} ) {  
    print "Age is found\n";  
}
```

EIW - Perl Control Structures

5

If – else in action

```
print "Enter a number greater than 0\n";  
$num = <STDIN>;  
if ($num<=0) {  
    print "Not a valid number!\n";  
} else {  
    print "Congratulations!\n";  
    print "You entered the number $num\n";  
}
```

EIW - Perl Control Structures

6

If elsif

- There is a shorthand for a sequence of if-else statements:

```
if ($num<10) {  
    print("You think to small\n");  
} elsif ($num<100) {  
    print("You picked a good number\n");  
} else {  
    print("Way too big, be reasonable!\n");  
}
```

EIW - Perl Control Structures

7

unless Control Structure

- Sometimes you want to leave off the if part, and just do something in the else.
 - **unless** can do this for you:

```
unless ($num > 0) {  
    print "I said greater than 0\n";  
}
```

EIW - Perl Control Structures

8

while is just like C/C++

```
print "Enter a number greater than 0\n";  
$num = <STDIN>;  
  
while ($num<0) {  
    print "Not valid, try again!\n";  
    print "Enter a number greater than 0\n";  
    $num = <STDIN>;  
}  
print "Congratulations!\n";  
print "You entered the number $num\n";
```

EIW - Perl Control Structures

9

do while is also like C/C++

```
do {  
  print "Enter a number greater than 0\n";  
  $num = <STDIN>;  
} while ($num <= 0);  
  
print "Congratulations!\n";  
print "You entered the number $num\n";
```

EIW - Perl Control Structures

10

until loop

```
$num = 0;  
  
until ($num > 0) {  
  print "Enter a number greater than 0\n";  
  $num = <STDIN>;  
}  
  
print "Congratulations!\n";  
print "You entered the number $num\n";
```

EIW - Perl Control Structures

11

Looping over input

- <STDIN> returns the value *undef* when it reaches End-of-File.
 - *undef* is FALSE

```
while ($line = <STDIN>) {  
  print $line;  
}
```

EIW - Perl Control Structures

12

|| and && operators (just like C/C++)

```
print "Enter a number between 1 and 100\n";
$num=<STDIN>;

if ($num<1 || $num>100) {
    print "Dummy\n";
} else {
    print "Great job! you entered $num\n";
}
```

EIW - Perl Control Structures

13

for loop (just like C/C++)

```
for ($j=0;$j<10;$j++) {
    print "$j\n";
}

for ($x=100;$x>0;$x=$x-10) {
    print "$x\n";
}
```

EIW - Perl Control Structures

14

foreach loop

- **foreach** iterates over the elements in an array (a list). A scalar variable assumes the value of each element in the list (iteratively):

```
foreach $x (2,4,6) {
    print "x is $x\n";
}
```

Output is:

```
x is 2
x is 4
x is 6
```

EIW - Perl Control Structures

15

foreach with no scalar

- You don't need to specify the scalar variable:

```
foreach (2,4,6) {  
    statement;  
}
```

- In this case, perl assumes you want to use the *default scalar variable*.
 - The default scalar variable is named `$_`

EIW - Perl Control Structures

16

Default variable name?

- That's right! Perl has a default variable.
- Many operators use `$_` if no scalar is supplied:

```
print;    # prints $_  
chop;    # chops $_
```

EIW - Perl Control Structures

17

Fun with `$_`

```
$total=0;  
while (<STDIN>) {  
    chop;  
    print "you entered";  
    print;  
    $total = $total + $_;  
}  
print "the total is $_\n";
```

EIW - Perl Control Structures

18

Some Exercises

- Write a perl program that reads in a number and prints that number 10 times.
- Write a perl program that reads in a number and prints an ASCII square of that size. For example, if the number is 4, the program prints:

```
@@@@
@  @
@  @
@@@@
```

EIW - Perl Control Structures

19

More fun practice

- Write a Perl program that generates an HTML table of the first 10 squares:

```
<TABLE>
  <TR> <TD>1</TD> <TD>1</TD> </TR>
  <TR> <TD>2</TD> <TD>4</TD> </TR>
  <TR> <TD>3</TD> <TD>9</TD> </TR>
  <TR> <TD>4</TD> <TD>16</TD> </TR>
  . . .
```

EIW - Perl Control Structures

20
