CSE2395/CSE3395

Topic 2

Scalars

Learning Perl 2nd edition
chapter 2, pages 31-47
Programming Perl 3rd edition
chapter 2, pages 6-8, 58-67
Programming Perl 2nd edition
pages 4-6, 38-44
`perldata` manpage

1

CSE2395/CSE3395

Last time

Covered in Topic 1

- Introduction to Perl
- Introduction to subject
- Outline of topics
  - What's examinable
- Simple Perl programs
  - and how to run them

2

CSE2395/CSE3395

To be covered today

- Scalar values
  - numbers
  - strings
- Scalar variables
- Scalar operators
- Console input/output
  - printing to the screen
  - reading from the keyboard
- Interpolating into strings

3

CSE2395/CSE3395

What is a scalar?

Definition

- A single value existing by itself
- Opposite is a vector
  - called list or array in computing

4

CSE2395/CSE3395

Scalars in Perl

Perl considers these to be scalars

- numbers
  - double-precision floating-point (C `double`)
  - e.g., `87`, `-1.06E20`, `0`
- strings
  - basic data type
  - not an array of characters
  - of arbitrary length
  - no pointers needed
  - e.g., `"frotz"`, `'nitfol'`, `"Hello world!\n"`
- references
  - covered in Topic 11

Llama2 pages 31-35

5

CSE2395/CSE3395

Scalar variables

How to store a scalar value

- Scalar values are stored in scalar variables
- Variables are global by default
- All Perl scalar variables begin with `$` character
  - `$apples`
  - `$text2`
  - `$_`
- Perl doesn't care if a scalar variable contains a number or a string
  - numbers and strings are converted back and forth as needed
  - `$a = 87` and `$a = "87"` are almost identical in Perl

6

String literals

A fixed string in your Perl script

- Single-quote delimited
  - all characters within string are literal, except
    - `\'` (becomes `'`)
    - `\\` (becomes `\`)
  - does not interpolate variables
  - `'hello'` (hello)
  - `'$35.40'` ($35.40)
  - `'it\'s'` (it's)
  - `'\n'` (backslash followed by n)

Llama2 pages 33-35; *Camel3* pages 60-64; *Camel2* pages 39-44

String literals

A fixed string in your Perl script

- Double-quote delimited
  - usual C backslash-rules (e.g., `\n`) apply
  - `"hello"` (hello)
  - `"it's \$35.40"` (it's $35.40)
  - `"\n"` (newline character)
  - interpolates scalar and array variables
  - `"hello $name"` (becomes hello Tiger if `$name` contains Tiger)
- Other string literal representations exist
  - to be introduced as they are met

Llama2 pages 33-35; *Camel3* pages 60-63; *Camel2* pages 39-44

Scalar operators

Things to do with numbers

- Most familiar numeric operators available
  - `+`, `-`, `++`, `%`, etc.
  - `=` (assignment)
  - `/` (floating-point division)
  - `<`, `<=`, `>`, `>=`, `==`, `!=` (numeric comparison)
  - `**` (exponentiation)
- New string operators
  - `.` (join two strings)
    - `"cat" . "fish"` (produces string "catfish")
  - `lt`, `le`, `gt`, `ge`, `eq`, `ne` (string comparison)

Llama2 pages 35-40; *Camel3* pages 86-110 *Camel2* pages 76-95; `perlop` manpage

String operators

The need for `gt` as well as `>`

- Perl converts between numbers and strings as needed
- Numerically, is 7 greater than 30?
  - Use `>` for numeric comparisons
  - Values on both sides of `>` are converted to numbers
  - `7 > 30` is false (zero)
- Alphabetically, is "7" greater than "30"?
  - "7" comes after "3" in ASCII
  - Use `gt` for string comparisons
  - Values on both sides of `gt` are converted to strings
  - `7 gt 30` is true (non-zero)
- Need to use the correct comparison operator

String interpolation

Substituting a variable's value into a string

- In C, use `printf` or `sprintf` function to insert variable's value into a string
  - `printf ("The sum is %d\n", total);`
- In Perl, place variable name inside double-quoted string
  - `print "The sum is $total\n";`
- Can also use string concatenation operator
  - `print "The sum is " . $total . "\n";`

Llama2 pages 44-45; *Camel3* pages 62-63; *Camel2* pages 40-41

String interpolation

Pitfalls and notes

- Single-quoted strings do not interpolate
  - `print 'the sum is $total\n'` prints "the sum is $total\n"
- If variable name is ambiguous, use braces
  - want to print "Today is the 6th" when `$day` is 6
  - `print "Today is the $dayth\n";`
    - wrong, inserts value of `$dayth` (possibly undefined)
  - `print "Today is the ${day}th\n";`
    - right, uses value of `$day`
- Braces can be used this way in any situation
  - used for accessing complex nested data structures (Topic 11)

Input and output

Communicating with Perl

CSE2395/CSE3395

- Output to screen with
  - `print` function
  - `printf` function
  - `print "testing\n";`
- Input from keyboard with
  - `<STDIN>` operator
  - reads from standard input up to (and including) the first newline character
  - `$line = <STDIN>;`
  - `chomp` function can be used to remove newline

Llama2 pages 45-46; *Camel3* pages 80-83; *Camel2* pages 53-55

Example

Calculating the circumference of a circle

CSE2395/CSE3395

```
#!/usr/bin/perl -w

# Set $pi.
$pi = 3.1415926535898;

# Read a number from the user.
print "Please enter radius: ";
$radius = <STDIN>;
# Remove the newline from $radius.
chomp $radius;

# Calculate the circumference.
$around = $radius * 2 * $pi;

# Print the result.
print "The circumference is $around\n";
```

Example

Calculating the maximum of two numbers

CSE2395/CSE3395

```
#!/usr/bin/perl -w

# Read two numbers.
print "Enter a number: ";
$num1 = <STDIN>; chomp $num1;
print "Enter another number: ";
$num2 = <STDIN>; chomp $num2;

# Calculate the bigger value.
if ($num1 > $num2) {
  $max = $num1; }
else { $max = $num2; }

# Print result
print "Maximum is $max\n";
```

undef

The undefined value

CSE2395/CSE3395

- Scalar variables used without being defined contain `undef`
- Often used as `NULL` is in C.
- `undef` returned by some functions and operators on out-of-range input
  - `<STDIN>` operator on end of file
  - use `defined` function to test an expression
- `undef` is converted to
  - empty string (`""`) in string context
  - zero in numeric context

Llama2 page 46; *Camel3* pages 818-819; *Camel2* page 235

Some scalar functions

Useful tools for working on scalars

CSE2395/CSE3395

- `chomp $string`
  - removes a newline from the end of `$string`
- `length $string`
  - returns length of `$string` in characters
- `uc|lc|ucfirst|lcfirst $string`
  - returns a version of $string entirely in uppercase/lowercase (or with just first letter changed)
- `rand $number`
  - returns pseudorandom number from 0 to `$number`

Camel3 chapter 29, pages 677-830; *Camel2* chapter 3, pages 141-242; `perlfunc` manpage

Covered today

CSE2395/CSE3395

- Scalar values
  - numbers
  - strings
- Scalar variables
  - always start with `$` character
- Scalar operators
- Console input/output
  - printing to the screen
  - reading from the keyboard
- Interpolating into strings

Going further

More things related to today’s topic

- References
  - the “other kind” of scalar value
  - Topic 11
- Unicode
  - support for national character sets
  - *Camel3* pages 401-410
- Operator overloading
  - providing new behaviour for builtin operators
  - *Camel2* pages 463-469; *Camel3* pages 347-362

Next time

To be covered in Topic 3

- Lists
- Arrays
  - variables that contain lists
- List and array functions
  - sorting lists
  - adding and removing array elements
- Context
  - scalar versus list

Reading:
Learning Perl 2nd edition chapter 3
Programming Perl 3rd edition pages 69-76
Programming Perl 2nd edition pages 45-49