

CSE2395/CSE3395

Topic 5


## Hashes

*Learning Perl 2nd edition*  
chapter 5, pages 66-71

*Programming Perl 3rd edition*  
pages 76-78

*Programming Perl 2nd edition*  
pages 50-51

perl data manpage



1

## Last time

Covered in Topic 4

- Control structures
  - if, while, for, foreach, do-while
- True and false values
- Loop control
  - labels, next, last, redo
- Expression modifiers
- Program arguments
  - @ARGV array
- Perl defaults
  - \$\_ variable (standard argument)
  - <> filehandle (diamond operator)

2

## To be covered today

- Hashes
  - aka associative arrays
- Hash variables
- Hash literals
- Functions that operate on hashes
- Accessing Perl's environment

3

## Revision

Arrays

- Arrays are
  - ordered
  - indexed by number (integer)
  - if element *n* exists, so do elements 0 to *n*-1

@array	42	"dog"	-0.2	undef	42	0
indices	0	1	2	3	4	5

4

## When arrays aren't enough

Indexing students by ID number

- @marks is an array of students' marks
- Index by student ID numbers
  - from 10000000 up

@marks			89	undef	53	74
indices	0		12345678	12345679	12345680	12345681

over ten million empty elements in here!

5

## When arrays aren't enough

Indexing by strings

- Want to look up capital cities of countries
- Index is name of country (string)

	Lima	Tokyo	London	Moscow	Ottawa	Cairo
indices	Peru	Japan	UK	Russia	Canada	Egypt

This is a hash

6

## Hash

### Definition

- Array using strings as indices (keys)
  - keys must be unique in a hash
- Elements (values) are scalars
  - values may be duplicated for different keys
- Unordered
  - Perl stores key-value pairs in an uncontrollable order
- Also called
  - associative array
  - hash table
  - (lookup) table



*Llama2* page 66; *Camel3* pages 51, 76-77;  
*Camel2* pages 36, 50

CSE2395/CSE3395

7

## Hashes in Perl

### Variables

- Hash variables start with % symbol
  - %capitals
  - %marks
  - %ENV
- Hashes can be copied
  - %newmarks = %marks;
- Whole hashes are not interpolated into strings
  - print "%capitals"; # prints "%capitals"
- Hash, array and scalar names are separate
  - %days, @days and \$days are unrelated



*Llama2* pages 66-67; *Camel3* pages 76-78  
*Camel2* page 50

CSE2395/CSE3395

8

## Hash literals

### Initializing a hash

- Hashes can be unwound into lists
  - each key/value pair kept together
- Lists can be wound into hashes
  - even number of elements
  - order of key-value pairs is not retained
- Initialize a hash with a list literal
- %capitals = ("Peru", "Lima",  
"Japan", "Tokyo", "UK", "London",  
"Russia", "Moscow", "Canada",  
"Ottawa", "Egypt", "Cairo");



*Llama2* pages 67-68; *Camel3* pages 76-78  
*Camel2* page 50

CSE2395/CSE3395

9

## Example

### Reverse lookup of a hash

- %capitals, given a country (key), returns corresponding capital city (value)
- Hash cannot return a key (country) given its value (capital city)
- Can reverse hash to switch order of keys and values
  - %countries = reverse %capitals;
  - %countries = ("Cairo", "Egypt", "Ottawa",  
"Canada", "Moscow", "Russia", "London",  
"UK", "Tokyo", "Japan", "Lima", "Peru");
- Only useful if no hash values were duplicated

CSE2395/CSE3395

10

## Hash elements

### Accessing a single element of %hash

- \$hash{key}
  - \$capitals{"Egypt"} # equal to "Cairo"
- key is any scalar (string) expression
  - if key is a scalar variable, still need the \$ sign
    - \$capitals{\$mynation}
- Assigning to \$hash{key} replaces old value or creates new key-value pair
  - \$capitals{"Australia"} = "Canberra";



*Llama2* page 67; *Camel3* page 52  
*Camel2* page 37

CSE2395/CSE3395

11

## Hash elements

### Accessing a single element of %hash

- \$hash{key}
- One hash value is of scalar type
  - thus \$ to indicate one element of %hash
    - \$days{"1"} is one element of hash %days
    - \$days[1] is one element of array @days
    - \$days is the scalar variable \$days
  - %hash{key} produces a compilation error
- Hash slices are possible
  - @capitals{"Canada", "UK"} # equal to list ("Ottawa", "London")
- Using nonexistent keys produces undef
  - \$capitals{"Atlantis"} # equal to undef

CSE2395/CSE3395

12

## Functions for hashes

### Getting keys and values

- `keys %hash`
  - ▶ returns a list containing all keys in `%hash`
  - ▶ order is indeterminate but same as `values` function
  - ▶ every key is unique
  - ▶ `keys %capitals # returns List ("Canada", "UK", "Egypt", "Japan", "Peru", "Russia")`
- `values %hash`
  - ▶ returns a list containing all values in `%hash`
  - ▶ order is same as `keys` function
  - ▶ `values %capitals # returns List ("Ottawa", "London", "Cairo", "Tokyo", "Lima", "Moscow")`



*Llama2* pages 68-69; *Camel3* pages 733-734, 824  
*Camel2* pages 181-182, 239; perlfunc manpage

CSE2395/CSE3395

13

## Example

### Printing out a hash

```
# Initialize the hash.
# The => operator is just a prettier-looking
# synonym for the , operator that also quotes
# the word on its left side. Great for hashes.
%capitals = (Peru => "Lima", Japan => "Tokyo",
            UK => "London", Russia => "Moscow",
            Canada => "Ottawa", Egypt => "Cairo");

# Iterate over the hash, once per nation.
# Countries could be printed sorted by saying
# foreach $nat (sort keys %capitals)
foreach $nat (keys %capitals)
{
    # Single hash elements can be interpolated.
    print "Capital of $nat is $capitals{$nat}\n";
}
```

CSE2395/CSE3395

14

## Functions for hashes

### Getting keys and values with the `each` function

- `each %hash`
  - ▶ returns a two-element list containing the first key-value pair of `%hash`
  - ▶ subsequent calls to `each` return the following key-value pair
  - ▶ when all key-value pairs have been returned, `each` returns an empty list (`()`)
  - ▶ state is kept associated with the hash
  - ▶ typical use:
    - `while (($key, $value) = each %hash) { ... }`
  - ▶ more space-efficient than `keys` for large hashes



*Llama2* page 69; *Camel3* pages 703-704  
*Camel2* pages 159-160; perlfunc manpage

CSE2395/CSE3395

15

## Example

### Using `each` to print out a hash

```
# Initialize the hash.
%capitals = (Peru => "Lima", Japan => "Tokyo",
            UK => "London", Russia => "Moscow",
            Canada => "Ottawa", Egypt => "Cairo");

# Iterate over the hash using each function.
# No provision for sorting the countries here
# as we see only one key-value pair at a time.
while (($nat, $cap) = each %capitals)
{
    print "Capital of $nat is $cap\n";
}
```

CSE2395/CSE3395

16

## Changing a hash

### Adding and removing elements

- To add a key-value pair to a hash
  - ▶ assign a value to `$hash{key}`
  - ▶ `$capitals{"Ukraine"} = "Kiev";`
- To remove a key-value pair from a hash
  - ▶ use `delete` function
  - ▶ `delete $capitals{"USSR"};`
  - ▶ assigning `undef` doesn't work
- To clear an entire hash
  - ▶ Assign the empty list to `%hash`
  - ▶ `%capitals = ();`



*Llama2* pages 67, 69-70; *Camel3* pages 699-700  
*Camel2* page 156; perlfunc manpage

CSE2395/CSE3395

17

## Example

### Counting identical lines in a file

```
# Clear hash (redundant).
%count = ();

# Read each line into $_.
while (<>)
{
    # Increment the frequency of this line.
    # The string making up the line is the key,
    # and the frequency is the value.
    $count{$_}++;
}

# Print results.
while (($key, $value) = each %count)
{
    print "$value: $key";
}
```

CSE2395/CSE3395

18

## Testing for a hash element

exists and defined

- To test if a hash key exists, use `exists` function
  - ▶ `exists $capitals{"Canada"} # true`
  - ▶ `exists $capitals{"Atlantis"} # false`
- To test if an existing hash key has a defined value, use `defined` function
  - ▶ `defined $capitals{"Canada"} # true`
  - ▶ `defined $capitals{"Vatican City"} # false: key exists but value is undef`
  - ▶ `defined $capitals{"Atlantis"} # false: key does not exist`



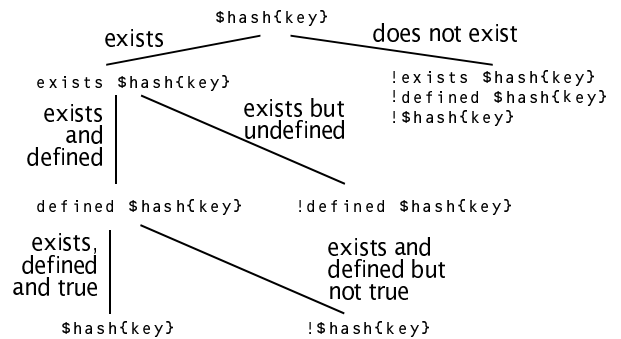
*Camel3* pages 697-698, 710-711;  
*Camel2* pages 155-156, 164; perlfunc manpage

CSE2395/CSE3395

19

## Testing for a hash element

exists, defined and truth value



CSE2395/CSE3395

20

## Environment

Accessing Perl's environment through `%ENV`

- Environment variables are set by shell
  - ▶ `[/bin/sh] NAME=value; export NAME`
  - ▶ `[/bin/csh] setenv NAME value`
  - ▶ `[COMMAND.COM] set NAME=value`
- Shell's environment is inherited by child processes
  - ▶ including any Perl script it runs
- Perl can access the environment through `%ENV` hash
  - ▶ `print "Search path is $ENV{'PATH'}\n";`
  - ▶ changes made to `%ENV` are inherited by children
  - ▶ changes made to `%ENV` are not returned to the shell

CSE2395/CSE3395

21

## Covered today

- Hashes
- Hash variables
- Hash literals
- Functions that operate on hashes
  - ▶ `keys`, `values`, `each`, `delete`, `exists`
- Accessing Perl's environment
  - ▶ the `%ENV` variable

CSE2395/CSE3395

22

## Going further

More things related to today's topic

- Tying
  - ▶ treat an external file (or any other object) like an internal hash (or any other variable)
  - ▶ *Camel3* pages 363-398; *Camel2* pages 301-315
- Databases
  - ▶ talking to databases with Perl
  - ▶ *Programming the Perl DBI* by Alligator Descartes & Tim Bunce, O'Reilly 2000
- Shells
  - ▶ the Unix command-line interface
  - ▶ `man sh`

23

## Next time

To be covered in Topic 6

- Regular expressions
  - ▶ pattern matching
- Functions using regular expressions
  - ▶ `split`, `join`



**Reading:**  
*Learning Perl 2nd edition* chapter 7, pages 76-91  
*Programming Perl 3rd edition* pages 139-195  
*Programming Perl 2nd edition* pages 58-76  
perlre manpage

CSE2395/CSE3395

24

CSE2395/CSE3395 lecture notes copyright  
© 2000-2001 Deborah Pickett.  
Reproduction of this presentation for  
nonprofit study use is permitted. All other  
reproduction, including for other  
educational courses, must be authorized in  
writing by the author.