Introduction to Perl: Part II

Pat Morin

COMP 2405

Carleton
UNIVERSITY
Canada's Capital University

Outline

- References
- Subroutines

Carleton
UNIVERSITY
Canada's Capital University

References in Perl

- Recall that a scalar value can be either a number, a string, or a reference
- A reference is really a pointer to a memory location
- Arrays and hashes can only store scalar values
  - To make complicated data structures in Perl, we need references

Referencing and Dereferencing

- To get a reference to an object we use the \ (backslash) operator
- To dereference a variable we use the {} operator
- We use ${}, @{}, or %{}, depending on whether the reference refers to a scalar, an array, or a hash


```
my $data = 23;
my $ref = \$data;
print("At memory location $ref ".
      "is stored the value '${$ref}'.\n");
```

Reference Examples

```
my @a = (1, 2, 3, 4);
my $s = "This is a scalar";
my %h = ('a' => 1, 'b'=> 2);

my $arrayRef = \@a;
my $scalarRef = \$s;
my $hashRef = \%h;

print(@{$arrayRef});    # print the array
print(${$arrayRef}[1]); # print 2
print(keys(%{$hashRef})); # print 'a' 'b'
print(${$hashRef}{'a'}); # print 1
```

Carleton UNIVERSITY
Canada's Capital University

Data Structures

- Arrays and Hashtables can store references

```
my @bday1 = (1879, 3, 14);
my @bday2 = (1898, 6, 17);
my @bday3 = (1906, 4, 28);

my %birthdays =
  ('einstein' => \@bday1,
   'escher' => \@bday2,
   'goedel' => \@bday3 );

my ($year, $month, $day)
  = @{$birthdays{'einstein'}};
print("Albert Einstein was born on ".
      "$year-$month-$day.\n");
```

Anonymous References

- We can make anonymous array and hash references using the [] operators and the {} operators

```
my %birthdays =
  ('einstein' => [1879, 3, 14],
   'escher' => [1898, 6, 17],
   'goedel' => [1906, 4, 28] );

my ($year, $month, $day)
  = @{$birthdays{'einstein'}};
print("Albert Einstein was born on ".
      "$year-$month-$day.\n");
```

Carleton UNIVERSITY
Canada's Capital University

Subroutines in Perl

- In Perl, we can define subroutines
- Syntax: sub *SubName* (*params*) { *code* }

```
sub PrintSomething ()
{
  print("Something\n");
}

...

PrintSomething();
```

Carleton UNIVERSITY
Canada's Capital University

Subroutines: Parameters

- Perl subroutines do not have named parameters
- Parameters are passed as an array named @_
- You should not assign new values to the elements of @_ (use references to achieve the same effect)

```
sub PrintStrings(@) {
  while ($s = shift(@_)) {
    print("$s\n");
  }
}
```

Carleton UNIVERSITY
Canada's Capital University

Parameter Lists and Prototypes

- You should specify explicitly what the parameters to a subroutine are
  - $ is a scalar
  - \@ is a reference to an array
  - \% is a reference to a hash
  - @ is an array (that eats everything passed to it)

```
sub sum (@) {
  my $sum = 0;
  foreach my $x (@_) {
    $sum = $sum + $x;
  }
  return $sum;
}
```

Carleton UNIVERSITY
Canada's Capital University

Parameter Lists (Cont'd)

```
sub AddtoHash (\%$$) {
  my ($hashRef, $s1, $s2) = @_;
  ${$hashRef}{$s1} = $s2;
}

...

my %table;
AddtoHash(%table, 'red', '#ff0000');
AddtoHash(%table, 'green', '#00ff00');
AddtoHash(%table, 'blue', '#0000ff');
```

Parameter Lists with @

- Using @ in a parameter list consumes all other input
  - ($@) is a parameter list that consists of one scalar followed by a variable number of arguments (evaluated in the array context)
  - (@$) doesn't make sense
- Rule of Thumb: only use @ as the last element of a parameter list

Subroutines: Return Values

- Subroutines return a value with the return keyword
- Subroutines can return a scalar (including reference) or an array of scalars

```
sub distSquared ($$) {
  my ($x, $y) = @_;
  return ($x*$x + $y*$y);
}
```

Carleton UNIVERSITY
Canada's Capital University

Summary

- References
  - Create a reference using the \ operator
  - Dereference a reference using the {} operator
  - Create anonymous references using the [] and {} operators
- Subroutines
  - Subroutine parameters are passed as an array called @_
  - Subroutine parameter lists can include scalars ($), references to hashes and arrays (\@ and \%) and a single greedy array (@)
  - Subroutines can return scalars or arrays
- See Chapters 5 and 6 of the Perl 5 Tutorial for more details