Day 7: Data Structures

Suggested reading:

http://perldoc.perl.org/perlreftut.html

or

```
perldoc perlreftut
```

Turn In Homework

Homework #5 Comment

Please format code for readability!

Homework Review

So Far:

$ @ %

But What About … ?

Complex Data

'Japan' => ('lang' => 'Japanese', 'pop' => 127.56M)

Multidimensional Arrays

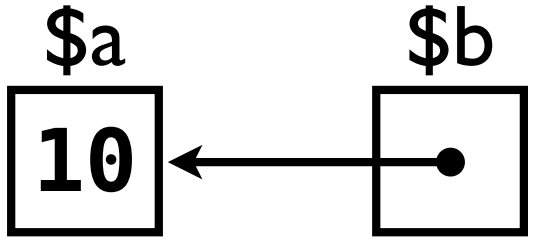
matrix[4][2] = 3.6354

Trees and Graphs

References

References

Scalar that refers to another scalar, list, hash, …




```
my $a = 10;
my $b = \$a;
${$b} = 5;
print $a; # => 5
```

Making a Reference to a Variable

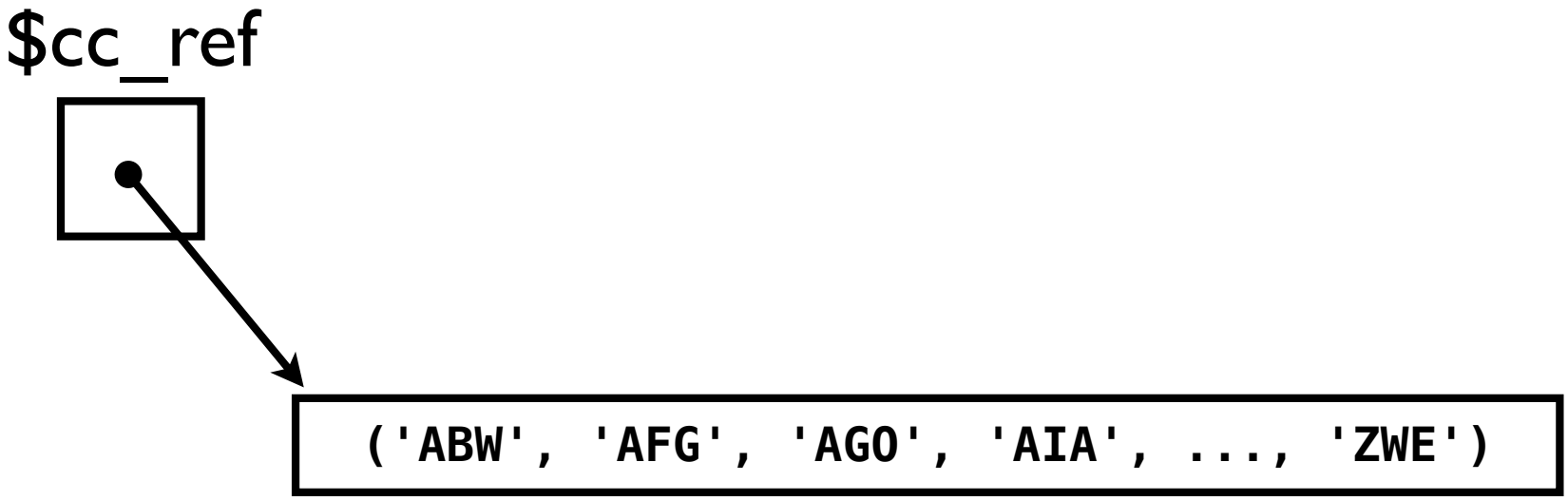
Prefix the variable with \

```
my $scalar_ref = \$region_string;
my $array_ref  = \@countries;
my $hash_ref   = \%country_code_map;
```

Making a Reference to Anonymous

Special syntax for unnamed arrays and hashes

```
my $cc_ref = [ 'ABW', 'AFG', ..., 'ZWE' ];
my $ch_ref = { 'ABW' => 'Aruba',
               'AFG' => 'Afghanistan' };
```



Using References

Use **{$*reference*}** in place of ***name*** of real thing

```
${$region_ref}        <=> $region
@{$country_ref}       <=> @country
${$country_ref}[1]    <=> $country[1]
%{$code_ref}          <=> %code
${$code_ref}{'ABW'} <=> $code{'ABW'}

print "Region: ${$region_ref}\n";
foreach my $country (@{$country_ref}) {
  print "${$code_ref}{$country}\n";
}
```

Reference Shortcuts I

- No ***need*** to use — **{$ref}** always works
- Use ***ref->*** to get one element of array or hash

```
$countries[2]

${$countries_ref}[2]
$countries_ref->[2]
$countries_ref[2]      # BAD! (in Perl 5.8)
```

```
$country_codes{'ABW'}

${$country_code_ref}{'ABW'}
$country_code_ref->{'ABW'}
$country_code_ref{'ABW'}    # BAD! (in 5.8)
```

Reference Shortcuts II

Can omit **->** between indices

```
${${$score_ref}{'Tim'}}[5]

$score_ref->{'Tim'}->[5]

$score_ref->{'Tim'}[5]
```

Data Structures

No References

- array
- (linked) list
- stack
- queue
- associative array / hashtable / dictionary / map
- set
- others?

Array of Arrays

Matrix (equal-sized sub-arrays) or not

```
my @countries = (
  [ 'ABW', 'Aruba', 193, 0.1 ],
  [ 'AFG', 'Afghanistan', 647500, 28.2 ],
  # ...
);

print "The size of Aruba is " .
      $countries[0][2] .
      " km^2.\n";

# or, $countries[0]->[2] if that helps.
```

Array of Arrays

Matrix (equal-sized sub-arrays) or not

```
my @countries = (
  [ 'ABW', 'Aruba', 193, 0.1 ],
  [ 'AFG', 'Afghanistan', 647500, 28.2 ],
  # ...
);

my $population = 0;
foreach my $country (@countries) {
    $population += $country->[3];
}
print "$population thousand people!\n";
```

Array of Arrays

Matrix (equal-sized sub-arrays) or not

```
my @countries = (
   [ 'ABW', 'Aruba', 193, 0.1 ],
   [ 'AFG', 'Afghanistan', 647500, 28.2 ],
   # ...
);

push @countries,
     ['---', 'South Sudan', 619745, 8.2];
```

General Structured Data

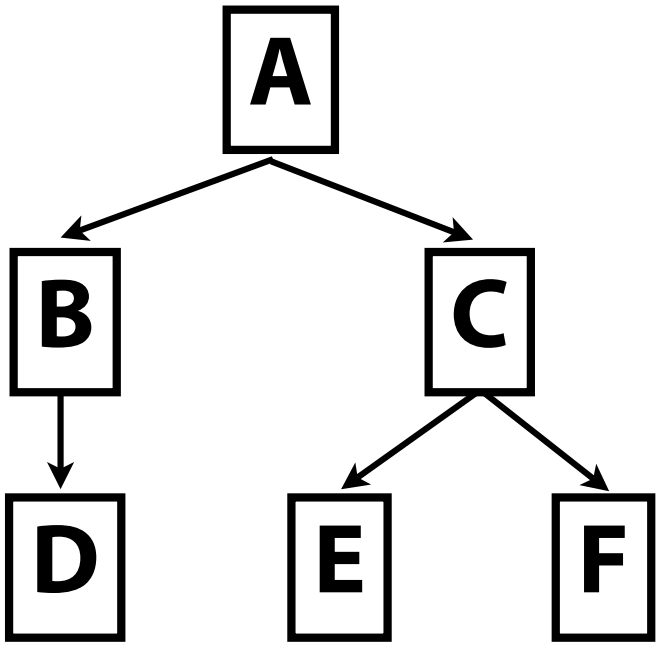
- No class or struct
- Just nest arrays and hashes as needed

```
$countries{'Brazil'}{'Pop'} = 192.8;
$countries{'India'}{'Lang'}[0] = 'Hindi';

$countries{'New'} = {};
$countries{'New'}{'Lang'} = [];
push @{$countries{'New'}{'Lang'}}, '???';
if (exists $countries{'New'}{'Pop'}) {
  $world_pop += $countries{'New'}{'Pop'};
}
```

Trees and Graphs

- General solution may be complicated
- One approach: Hash the parent-child relations




```
$parent{'B'} = 'A';
$parent{'C'} = 'A';
$parent{'D'} = 'B';
$parent{'E'} = 'C';
```

or

```
$child{'A'} = ['B', 'C'];
$child{'B'} = ['D'];
$child{'C'} = ['E', 'F'];
```

Subroutines and Data Structures

The Problem

Perl flattens arrays and hashes:

```
my @array_1 = (1, 2);
my @array_2 = (3, 4);
my %hash = ( 'foo' => 'bar' );

my_function(@array_1, @array_2, %hash);

sub my_function {
  print join(', ', @_) . "\n";
}

# 1, 2, 3, 4, foo, bar
```

The Solution

Use references!

```
my @array_1 = (1, 2);
my @array_2 = (3, 4);
my %hash = ( 'foo' => 'bar' );

my_function(\@array_1, \@array_2, \%hash);

sub my_function {
  my ($a1_ref, $a2_ref, $h_ref) = @_;
  my @array_1 = @{$a1_ref};
}
```

Almost Done!

Other Scripting Languages

- Nested data structures generally just work
- Typically: *object.member*
- Common data structures may have direct support

Homework

- Real country data, 1960–2002
- Read the assignment carefully! Lots of hints…
- Pick ***just one*** report
- Must use good data structure(s)