Solved In Perl 6

Jonathan Worthington

Italian Perl Workshop 2009

Solved in Perl 6

Overview

- I'll take a range of everyday programming problems and for each one show…
  - The Perl 6 code that solves it
  - The output that code gives when run
- Hopefully, a good way for you to start to grasp some of the new syntax and features
- A chance to show off some of the cool stuff in Perl 6
- All examples shown today work in Rakudo

Solved in Perl 6

Problem

Say "Hello, world"

Solution

```
say "Hello, world!"
```

Output

```
Hello, world!
```

Solved in Perl 6

Problem

Read input from the console

Solution

```
print "Enter your name: ";
my $name = $*IN.get;
say "Ciao $name!";
```

Output

```
Enter your name: Jonathan
Ciao Jonathan!
```

Solved in Perl 6

Problem

Check a value is in a given range

Solution 1

```
loop {
    print "Enter a number from 1 to 10: ";
    my $num = $*IN.get;
    unless 1 <= $num <= 10 { say "Fail!" }
}
```

Output

```
Enter a number between 1 and 10: 3
Enter a number between 1 and 10: 42
Fail!
```

Problem

Check a value is in a given range

Solution 2

```
loop {
    print "Enter a number from 1 to 10: ";
    my $num = $*IN.get;
    unless $num ~~ 1..10 { say "Fail!" }
}
```

Output

```
Enter a number between 1 and 10: 3
Enter a number between 1 and 10: 42
Fail!
```

Solved in Perl 6

Problem

Add up a list of numbers

Solution

```
my @nums = 1, 5, 7, -2, 3, 9, 11, -6, 14;
say [+] @nums;
```

Output

```
42
```

Solved in Perl 6

Problem

Check if a list is sorted

Solution

```
my @a = 1, 1, 2, 3, 5, 8;
my @b = 9, 4, 1, 16, 36, 25;
if [<=] @a { say '@a is sorted' }
if [<=] @b { say '@b is sorted' }
```

Output

```
@a is sorted
```

Solved in Perl 6

Problem

Get a Perl-ish representation of a data structure (Data::Dumper style)

Solution

```
my @a = 1, 2, 3;
push @a, { x => 42, y => 100 };
say @a.perl;
```

Output

```
[1, 2, 3, {"y" => 100, "x" => 42}]
```

Solved in Perl 6

Problem

Iterate over a list

Solution

```
my @cities = <Naples Rome Florence Pisa>;
for @cities -> $city {
    say "I've been to $city";
}
```

Output

```
I've been to Naples
I've been to Rome
I've been to Florence
I've been to Pisa
```

Solved in Perl 6

Problem

Iterate over the keys and values of a hash

Solution

```
my %distances = Rome => 333, Naples => 567;
for %distances.kv -> $city, $distance {
    say "$city is $distance km away";
}
```

Output

```
Rome is 333 km away
Naples is 567 km away
```

Solved in Perl 6

Problem

Check if any of a list of test scores is a pass

Solution

```
my @a = 75, 47, 90, 22, 80;
my @b = 61, 77, 94, 82, 60;
my @c = 45, 59, 33, 11, 19;
if any(@a) >= 60 { say "Some passes in A" }
if any(@b) >= 60 { say "Some passes in B" }
if any(@c) >= 60 { say "Some passes in C" }
```

Output

```
Some passes in A
Some passes in B
```

Solved in Perl 6

Problem

Check if all of a list of test scores are passes

Solution

```
my @a = 75, 47, 90, 22, 80;
my @b = 61, 77, 94, 82, 60;
my @c = 45, 59, 33, 11, 19;
if all(@a) >= 60 { say "All passes in A" }
if all(@b) >= 60 { say "All passes in B" }
if all(@c) >= 60 { say "All passes in C" }
```

Output

```
All passes in B
```

Solved in Perl 6

Problem

Check if none of a list of test scores is a pass

Solution

```
my @a = 75, 47, 90, 22, 80;
my @b = 61, 77, 94, 82, 60;
my @c = 45, 59, 33, 11, 19;
if none(@a) >= 60 { say "No passes in A" }
if none(@b) >= 60 { say "No passes in B" }
if none(@c) >= 60 { say "No passes in C" }
```

Output

```
No passes in C
```

Solved in Perl 6

Problem

Get a random item from a list

Solution

```
my @drinks = <wine beer vodka>;
say "Tonight I'll drink { @drinks.pick }";
```

Output (results should vary ;-))

```
Tonight I'll drink vodka
```

Solved in Perl 6

Problem

Shuffle a list into a random order

Solution

```
my @competitors = <Tina Lena Owen Peter>;
my @order = @competitors.pick(*);
for @order { .say }
```

Output (results should vary ;-))

```
Peter
Lena
Owen
Tina
```

Problem

Write and call a subroutine with parameters

Solution

```
sub greet($name) {
    say "Ciao $name!";
}
greet("Patrick");
```

Output

```
Ciao Patrick!
```

Solved in Perl 6

Problem

Write a subroutine taking an array and a hash

Solution

```
sub example(@a, %h) {
    say @a.elems;
    say %h.keys;
}
my @nums = 42, 57, 74;
my %mapping = a => 1, b => 2;
example(@nums, %mapping);
```

Output

```
3
ab
```

Solved in Perl 6

Problem

Write a subroutine that only takes a number

Solution

```
sub double(Num $n) { 2 * $n }
say double(21);
say double("oh no I'm not a number");
```

Output

```
42
Parameter type check failed; expected Num,
but got Str for $n in call to double
```

Solved in Perl 6

Problem

Use multi-subs to react differently by type

Solution

```
multi double(Num $n) { 2 * $n }
multi double(Str $s) { $s x 2 }
say double(21);
say double("boo");
```

Output

```
42
booboo
```

Solved in Perl 6

Problem

Compute factorial (recursively)

Solution

```
multi fact($n) { $n * fact($n - 1) }
multi fact(0)  { 1 }
say fact(1);
say fact(10);
```

Output

```
1
3628800
```

Solved in Perl 6

Problem

Compute factorial (using a meta-operator)

Solution

```
sub fact($n) { [*] 1..$n }
say fact(1);
say fact(10);
```

Output

```
1
3628800
```

Solved in Perl 6

Problem

Add a new factorial operator (so 10! works)

Solution

```
sub postfix:<!>($n) { [*] 1..$n }
say 1!;
say 10!;
```

Output

```
1
3628800
```

Solved in Perl 6

Problem

Declare a class with attributes and a method

Solution

```
class Product {
    has $.name;   # Attr + accessor
    has $!price;  # Attr only
    has $.discount is rw;
                  # Attr + lvalue accessor
    method get_price {
        return $!price - $!discount;
    }
}
```

Problem

Instantiate a class and call a method on it

Solution

```
my $prod = Product.new(
    name        => "Beer",
    price       => 500,
    discount    => 60
);
say $prod.get_price;
```

Output

```
440
```

Solved in Perl 6

Problem

Get/set attributes through accessors

Solution

```
say $prod.name;
$prod.discount = 40;
say $prod.get_price;
$prod.name = 'Wine';
```

Output

```
Beer
460
Cannot assign to readonly variable.
```

Solved in Perl 6

Problem

Call a method on every object in a list

Solution

```
my @products =
  Product.new(name => 'Beer', price => 500),
  Product.new(name => 'Wine', price => 450),
  Product.new(name => 'Vodka', price => 1600);
my @uc_names = @products>>.name>>.uc;
for @uc_names { .say }
```

Output

```
BEER
WINE
VODKA
```

Solved in Perl 6

Problem

Introspect a class to find its methods

Solution

```
my @meths = Product.^methods(:local);
for @meths>>.name {  .say }
```

Output

```
get_price
discount
name
```

Solved in Perl 6

Problem

Sort an array of objects by result of a method

Solution (Example 1)

```
my @products =
  Product.new(name => 'Beer', price => 500),
  Product.new(name => 'Wine', price => 450),
  Product.new(name => 'Vodka', price => 1600);
my @sorted = @products.sort(*.name);
for @sorted { .name.say }
```

Output (Example 1)

```
Beer
Vodka
Wine
```

Problem

Sort an array of objects by result of a method

Solution (Example 2)

```
my @products =
  Product.new(name => 'Beer', price => 500),
  Product.new(name => 'Wine', price => 450),
  Product.new(name => 'Vodka', price => 1600);
my @sorted = @products.sort(*.get_price);
for @sorted { .name.say }
```

Output (Example 2)

```
Wine
Beer
Vodka
```

Problem

Find minimum and maximum values from a list

Solution (Example 1)

```
my @temperatures = -3, 5, 7, 2, -1, -4, 0;
say "Minimum was " ~ @temperatures.min;
say "Maximum was " ~ @temperatures.max;
```

Output (Example 1)

```
Minimum was -4
Maximum was 7
```

Problem

Find minimum and maximum values from a list

Solution (Example 2)

```
my @products =
  Product.new(name => 'Beer', price => 500),
  Product.new(name => 'Wine', price => 450),
  Product.new(name => 'Vodka', price => 1600);
say "Cheapest: " ~ @products.min(*.get_price).name;
say "Costliest: " ~ @products.max(*.get_price).name;
```

Output (Example 2)

```
Cheapest: Wine
Costliest: Vodka
```

Problem

Paper, Scissor, Stone game

Solution (Part 1)

```
class Paper { }
class Scissor { }
class Stone { }
multi win(Paper,   Stone)   { "Win" }
multi win(Scissor, Paper)   { "Win" }
multi win(Stone,   Scissor) { "Win" }
multi win(::T,     T)       { "Draw" }
multi win(Any,     Any)     { "Lose" }
```

Solved in Perl 6

Problem

Paper, Scissor, Stone game

Solution (Part 2)

```
say win(Paper, Paper);
say win(Scissor, Stone);
say win(Stone, Scissor);
```

Output

```
Draw
Lose
Win
```

Solved in Perl 6

Want to play with Perl 6?

- Rakudo – the most actively developed Perl 6 compiler – is available from: http://www.rakudo.org/
- Lots of Perl 6 resources can be found at: http://www.perl6.org/
- Join the friendly IRC channel: #perl6 on irc.freenode.org
- Write modules, write applications, jump into the evolving Perl 6 community and make your mark on it ☺

Grazie

Questions?