Linguagem Ruby

PUC Minas
Poços de Caldas

Luiz Alberto Ferreira Gomes

Curso de Ciência da Computação

18 de setembro de 2020

Ruby

1 Arrays

2 Hashes

Arrays (1)

- Coleção de objetos (auto-expandível)
- Indexado pelo operador (método) [ ]
- Pode ser indexado por números negativos ou intervalos
- Tipos heterogêneos são permitidos em um mesmo array
- %{str1 str2} pode ser utilizado para criar um array de strings

Arrays (2)

Listing 1: arrays.rb

```ruby
heterogeneo = [1, "dois", :tres]
puts heterogeneo[1] # => dois (indice comeca em 0)
palavras = %w{ olhe que grande dia hoje! }
puts palavras[-2] # => dia
puts "#{palavras.first} - #{palavras.last}" # => olha - hoje!
p palavras[-3, 2] # => ["grande", "dia"] (volta 3 and pega 2)
p palavras[2..4] # => ["grande", "dia", "hoje!"]
puts palavras.join(',') # => olhe,que,grande,dia,hoje!
```

Arrays (3)

- Modificando arrays:
  - criação: $= [\ ]$
  - inclusão: push ou $<<$
  - remoção: pop ou shift
- Extração randômica de elementos com sample
- Classificação ou inversão com sort! ou reverse!

Arrays (4)

Listing 2: arrays2

```
1  pilha = []; pilha << "um"; pilha.push ("dois")
2  puts pilha.pop # => dois
3
4  fila = []; fila.push "um"; fila.push "dois"
5  puts fila.shift # => um
6
7  a = [5,3,4,2].sort!.reverse!
8  p a # => [5,4,3,2]
9  p a.sample(2) # => extrai dois elementos
10
11 a[6] = 33
12 p a # => [5, 4, 3, 2, nil, nil, 33]
```

Arrays (5)

- Métodos úteis
  - each - percorre um array
  - select - filtra por seleção
  - reject - filtra por rejeição
  - map - modifica cada elemento do array

Arrays (6)

Listing 3: arrays2

```
a = [1, 3, 4, 7, 8, 10]
a.each { |num| print num } # => 1347810
puts # => (nova linha)
novo = a.select { |num| num > 4 }
p novo # => [7, 8, 10]
novo = a.select { |num| num < 10 }
            .reject{ |num| num.even? }
p novo # => [1, 3, 7]
# Multiplica cada elemento do array produzindo
# um novo array
novo = a.map {|x| x * 3}
p novo # => [3, 9, 12, 21, 24, 30]
```

Recapitulando

- A API de arrays é flexível e poderosa
- Existem diversas formas de processar um elemento do array

Ruby

1 Arrays

2 Hashes

Hashes (1)

- Coleção indexada de objetos
- Criados com {} ou Hash.new
- Também conhecidos como arrays associativos
- Pode ser indexado com qualquer tipo de dados
  - não apenas com inteiros
- Acessados utilizando o operador [ ]
- Atribuição de valores poder feita usando:
  - => (criação)
  - [] (pós-criação)

Hashes (2)

Listing 4: hashes.rb

```ruby
propriedades = { "font" => "Arial", "size" => 12, "color" => "red"}

puts propriedades.length # => 3
puts propriedades["font"] # => Arial
propriedades["background"] = "Blue"
propriedades.each_pair do |key, value|
  puts "Key: #{key} value: #{value}"
end
# => Key: font value: Arial
# => Key: size value: 12
# => Key: color value: red
# => Key: background value: Blue
```

Hashes (3)

- E se tentarmos acessar um valor em Hash que não existe?
  - nil é retornado
- Se o Hash é criado com Hash.new(0) 0 é retornado.

Listing 5: word_frequency.rb

```
1 frequencias = Hash.new(0)
2 sentenca = "Chicka chicka boom boom"
3 sentenca.split.each do |word|
4   frequencias[word.downcase] += 1
5 end
6 puts frequencias # => {"chicka" => 2, "boom" => 2}
```

Hashes (4)

- A partir da versão 1.9
  - A ordem de criação do Hash é mantida
  - A sintaxe simbolo: pode ser utilizada, se símbolos são utilizados como chave
  - Se o Hash é o último argumento, {} são opcionais

Hashes (5)

Listing 6: more_hashes.rb

```ruby
familia = {oldest: "Jim", older: "Joe", younger: "Jack"}
familia[:youngest] = "Jeremy"
p familia
# => {:oldest=>"Jim",:older=>"Joe",:younger=>"\Jack
# =>   ,:youngest => \"Jeremy}

def ajusta_cores (props = {foreground: "red",background: "white"})
  puts "Foreground: #{props[:foreground]}" if props[:foreground]
  puts "Background: #{props[:background]}" if props[:background]
end
ajusta_cores # => foreground: red
               # => background: white
ajusta_cores ({ :foreground => "green" }) # => foreground: green
ajusta_cores background: "yella" # => background: yella
ajusta_cores :background => "magenta" # => background: magenta
```

Recapitulando

- Hashes são coleções indexadas
- Usado de forma similar aos arrays

Hora de Colocar as Mãos na Massa (1)

1. Modifique o jogo de adivinhação para permitir que o sistema me informe, antes de uma jogada, os números que já foram chutados e me impeça o usuário de digitar um número repetido.
2. Modifique o jogo de adivinhação para permitir que o usuário escolha o nível de dificuldade que poderá variar de 1 (fácil) a 5 (difícil). A faixa de números aleatórios que o sistema gerará variará de acordo com o nível: 1(1-30), 2(1-60), 3(1-100), 4(1-150) e 5(1-200).