Linguagem Ruby

PUC Minas
Poços de Caldas

Luiz Alberto Ferreira Gomes

Curso de Ciência da Computação

8 de agosto de 2019

Ruby

1 Blocos

2 Strings

3 Arrays

4 Hashes

Blocos (1)

- Um "Trecho" de código
  - escrito entre chaves($\{\}$) ou entre do e end
  - passado para métodos como o último parâmetro
- **Convenção**
  - use chaves($\{\}$) quanto o bloco contém uma linha
  - use do e end quando o bloco contém múltiplas linhas
- Frequentemente utilizado em iteração

Blocos (2)

Listing 1: times.rb

```
1.times { puts "Hello World!" }
# => Hello World!
2.times do |index|
  if index > 0
    puts index
  end
end
# => 1
2.times { |index| puts index if index > 0 }
# => 1
```

Utilizando Blocos

- Duas técnicas para utilizar blocos nos métodos
- **Implicitamente:**
  - use `block_given?` para checar se o bloco foi passado
  - use `yield` para chamar o bloco
- **Explicitamente:**
  - use `&` como prefixo do último parâmetro
  - use `call` para chamar o bloco

Técnica Implícita (1)

- Necessário checar com `block_given?`
  - se não uma excessão será lançada

Listing 2: implicit_blocks.rb

```
1  def totaliza(valores)
2    return "Nenhum bloco foi passado" unless block_given?
3    total = 0
4    for valor in valores
5      total += valor
6      yield(total)
7    end
8  end
9
10 totaliza([ 20, 30, 40, 10 ]){| resultado | puts resultado }
11 totaliza([ 20, 30, 40, 10 ]) do | resultado |
12   resultado = resultado * 0.25
```

Técnica Implícita (2)

```
13    puts "#{resultado}"
14  end
15
16  puts totaliza ([ 20, 30, 40, 10 ]) # => Nenhum bloco foi passado
```

Técnica Explícita (1)

- Necessário checar com `nil?`

Listing 3: implicit_blocks.rb

```
1  def totaliza(valores, &um_bloco)
2    return "Nenhum bloco foi passado" if um_bloco.nil?
3    total = 0
4    for valor in valores
5      total += valor
6      um_bloco.call(total)
7    end
8  end
9
10 totaliza([ 20, 30, 40, 10 ]){| resultado | puts resultado }
11 totaliza([ 20, 30, 40, 10 ]) do | resultado |
12   resultado = resultado * 0.25
13   puts "#{resultado}"
```

Técnica Explícita (2)

```
14  end
15
16  puts totaliza ([ 20, 30, 40, 10 ]) # => Nenhum bloco foi passado
```

Recapitulando

- Blocos são apenas trechos de códigos que podem ser passados para métodos
- Tanto explicitamente quanto implicitamente

Ruby

1 Blocos

2 Strings

3 Arrays

4 Hashes

Strings (1)

- Strings com aspas simples
  - permitem a utilização de ' com \
  - mostra a string como foi escrita
- Strings com aspas duplas
  - interpreta caracteres especiais como \n e \t
  - permite a interpolação de strings, evitando concatenação

Strings (2)

Listing 4: strings.rb

```
1  aspas_simples = 'D\' Silva Filho\n programa em Ruby!'
2  aspas_duplas  = "D\' Silva Filho\n programa em Ruby!"
3  puts aspas_simples # => D' Silva Filho\n programa em Ruby!
4  puts aspas_duplas  # => D' Silva Filho\n
5                     # =>  programa em Ruby!
6  def multiplica (um, dois)
7    "#{um} multiplicado por #{dois} = #{um * dois}"
8  end
9  puts mutiplica(5, 3)
10 # => 5 multiplicado por 3 = 15
```

Strings (3)

- Métodos terminados com ! modificam a string
  - a maioria retorna apenas um novo string
- Permite o uso do %Q{textos longos com multiplas linhas}
  - o mesmo comportamento de strings com aspas duplas
- É essencial dominar a API de Strings do Ruby

Strings (4)

Listing 5: more_strings.rb

```
1  nome = " tim"
2  puts nome.lstrip.capitalize # => Tim
3  p nome # => " tim"
4  nome.lstrip! # remove os espacos do inicial (modifica)
5  nome[0] = 'K' # substitui o primeiro caracter
6  puts nome # => Kim
7
8  clima = %Q{O dia esta quente la fora
9                pegue os guarda\-chuva}
10
11 clima.lines do |line|
12   line.sub! 'quente', 'chuvoso' # substitui 'quente' with 'chuvoso'
13   puts "#{line.strip}"
14 end
15 # =>  dia esta quente la fora
16 # =>       pegue os guarda\-chuvas
```

Símbolos

- :simbolo – string altamente otimizadas
  - ex. :domingo, :dolar, :calcio, :id
- Constantes que não precisam ser pré-declaradas
- Garantia de unicidade e imutabilidade
- Podem ser convertidos para uma String com to_s
  - ou de String para Símbolo com to_sym

Recapitulando

- A interpolação evita a concatenação de strings
- Strings oferecem uma API muito útil

Ruby

1 Blocos

2 Strings

3 Arrays

4 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 6: arrays.rb

```
1  heterogeneo = [1, "dois", :tres]
2  puts heterogeneo[1] # => dois (indice comeca em 0)
3  palavras = %w{ olhe que grande dia hoje! }
4  puts palavras[-2] # => dia
5  puts "#{palavras.first} - #{palavras.last}" # => olha - hoje!
6  p palavras[-3, 2] # => ["grande", "dia"] (volta 3 and pega 2)
7  p palavras[2..4] # => ["grande", "dia", "hoje!"]
8  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 7: 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 8: arrays2

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

2 Strings

3 Arrays

4 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 9: 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 10: word_frequency.rb

```ruby
frequencias = Hash.new(0)
sentenca = "Chicka chicka boom boom"
sentenca.split.each do |word|
  frequencias[word.downcase] += 1
end
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 11: 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

Para Saber Mais

- ⟨https://www.ruby-lang.org/en/⟩
  - referência oficial da linguagem Ruby onde a toda a sua documentação está disponível para ser consultada.
- ⟨http://rubyonrails.org/⟩
  - referência oficial do framework Rails onde a toda a sua documentação está disponível para ser consultada.
- ⟨http://www.codecademy.com/pt/tracks/ruby⟩
  - curso iterativo em portugês sobre a linguagem Ruby.
- ⟨https://gorails.com/setup/ubuntu/16.04⟩
  - guia para instalação do Ruby on Rails no Ubuntu e no Mac OSX.