Язык программирования Ruby, занятие 2

CS 212

6 ноября 2018 г.

План

1. Массивы
2. Условные операторы
3. Циклы
4. Хэши
5. Обработка исключений
6. Регулярные выражения
7. Файлы

Массивы

```
1 ary = Array.new  #=> []
2 Array.new(3)     #=> [nil, nil, nil]
3 Array.new(3, true) #=> [true, true, true]
4 Array.new(4) { Hash.new } #=> [{}, {}, {}, {}]
5 arr = ['a', 'b', 'c', 'd', 'e', 'f']
6 arr.fetch(100) #=> IndexError: index 100 outside
      of array bounds: -6...6
7 arr.fetch(100, "oops") #=> "oops"
```

Листинг 1: Массивы

Размер массива

```
1  arr = [1, 2, 3, 4, 5, 6,]
2  arr.take(3) #=> [1, 2, 3]
3  arr.drop(3) #=> [4, 5, 6]
4  arr = [1, 1, 2, 3, 5, 6, 8]
5  arr = [1, 1, 2, 3, 5, 6, 8]
6  arr.length = 7
7  arr.size = 7
8  arr.count = 7
9  arr.count(&:even?) # => 3
10 arr.count(1) # => 2
```

Листинг 2: Массивы 2

Добавление в массив

```
1 arr = [1, 2, 3, 4]
2 arr.push(5) #=> [1, 2, 3, 4, 5]
3 arr << 6   #=> [1, 2, 3, 4, 5, 6]
4 arr.unshift(0) #=> [0, 1, 2, 3, 4, 5, 6]
5 arr.insert(3, 'apple') #=> [0, 1, 2, 'apple', 3,
      4, 5, 6]
```

Листинг 3: Массивы 3

Изъятие из массива

```
1 arr = [1, 2, 3, 4, 5, 6]
2 arr.pop #=> 6
3 arr #=> [1, 2, 3, 4, 5]
4 arr.shift #=> 1
5 arr #=> [2, 3, 4, 5]
6 arr.delete_at(2) #=> 4
7 arr #=> [2, 3, 5]
8 arr = [1, 2, 2, 3]
9 arr.delete(2) #=> [1, 3]
```

Листинг 4: Массивы 4

compact, uniq

```
1 arr = ['foo', 0, nil, 'bar', 7, 'baz', nil]
2 arr.compact #=> ['foo', 0, 'bar', 7, 'baz']
3 arr         #=> ['foo', 0, nil, 'bar', 7, 'baz',
     nil]
4 arr.compact! #=> ['foo', 0, 'bar', 7, 'baz']
5 arr         #=> ['foo', 0, 'bar', 7, 'baz']
6 arr = [2, 5, 6, 556, 6, 6, 8, 9, 0, 123, 556]
7 arr.uniq #=> [2, 5, 6, 556, 8, 9, 0, 123]
```

Листинг 5: Массивы 5

Массивы

```
1  arr = [1, 2, 3, 4, 5]
2  arr.each { |a| print a -= 10, " " }
3  # prints: -9 -8 -7 -6 -5
4  #=> [1, 2, 3, 4, 5]
5  words = %w[rats live on no evil star]
6  str = ""
7  words.reverse_each { |word| str +=
       "#{word.reverse} " }
8  str #=> "rats live on no evil star "
9  arr.map { |a| 2*a } #=> [2, 4, 6, 8, 10]
10 arr                 #=> [1, 2, 3, 4, 5]
11 arr.map! { |a| a**2 } #=> [1, 4, 9, 16, 25]
12 arr                 #=> [1, 4, 9, 16, 25]
```

Листинг 6: Массивы 6

Массивы

```
1 arr = [1, 2, 3, 4, 5, 6]
2 arr.select { |a| a > 3 }  #=> [4, 5, 6]
3 arr.reject { |a| a < 3 }  #=> [3, 4, 5, 6]
4 arr.drop_while { |a| a < 4 } #=> [4, 5, 6]
5 arr                       #=> [1, 2, 3, 4, 5, 6]
```

Листинг 7: Массивы 7

Инициализация объектами

```
1 a = Array.new(2, Hash.new)
2 # => [{}, {}]
3
4 a[0]['cat'] = 'feline'
5 a # => [{"cat"=>"feline"}, {"cat"=>"feline"}]
6
7 a[1]['cat'] = 'Felix'
8 a # => [{"cat"=>"Felix"}, {"cat"=>"Felix"}]
```

Листинг 8: Подвох с объектом в массиве

Умножение массивов

```
1 [ 1, 2, 3 ] * 3   #=> [ 1, 2, 3, 1, 2, 3, 1, 2, 3 ]
2 [ 1, 2, 3 ] * "," #=> "1,2,3"
```

Листинг 9: Подвох с умножением

Продвинутые трюки с массивами

```
1 a = [1, 2, 3, 4]
2 a.combination(1).to_a #=> [[1],[2],[3],[4]]
3 a.combination(2).to_a #=>
    [[1,2],[1,3],[1,4],[2,3],[2,4],[3,4]]
4 a.combination(3).to_a #=>
    [[1,2,3],[1,2,4],[1,3,4],[2,3,4]]
5 a.combination(4).to_a #=> [[1,2,3,4]]
6 a.combination(0).to_a #=> [[]] # one combination
    of length 0
7 a.combination(5).to_a #=> [] # no combinations of
    length 5
```

Листинг 10: Комбинации вхождений

Продвинутые трюки с массивами

```
1 a = [1, 2, 3]
2 a.permutation.to_a  #=>
    [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
3 a.permutation(1).to_a #=> [[1],[2],[3]]
4 a.permutation(2).to_a #=>
    [[1,2],[1,3],[2,1],[2,3],[3,1],[3,2]]
5 a.permutation(3).to_a #=>
    [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
6 a.permutation(0).to_a #=> [[]] # one permutation
    of length 0
7 a.permutation(4).to_a #=> [] # no permutations of
    length 4
```

Листинг 11: Комбинации с перестановками

Множество

```
1 require 'set'
2 s1 = Set[1, 2]                    #=> #<Set: {1,
      2}>
3 s2 = [1, 2].to_set                #=> #<Set: {1,
      2}>
4 s1 == s2                          #=> true
5 s1.add("foo")                     #=> #<Set: {1,
      2, "foo"}>
6 s1.merge([2, 6])                  #=> #<Set: {1,
      2, "foo", 6}>
7 s1.subset?(s2)                    #=> false
8 s2.subset?(s1)                    #=> true
```

Листинг 12: Множества

Условные операторы

```
1  x = 1
2  if x > 2
3    puts 'x is greater than 2'
4  elsif x <= 2 and x!=0
5    puts 'x is 1'
6  else
7    puts "I can't guess the number"
8  end
9
10 puts 'x > 1' if x > 1
11 puts 'x <= 1' unless x > 1
12 if x == 3 then puts "x is 3" end
```

Листинг 13: Условные операторы

Условные операторы

```
1  true ? "this is true" : "this is not true"
2
3
4  case a
5  when 5
6    puts "a is 5"
7  when 6
8    puts "a is 6"
9  else
10   puts "a is neither 5, nor 6"
11 end
```

Листинг 14: Условные операторы 2

Циклы

```ruby
loop do
  puts "This will keep printing until you hit Ctrl + c"
end

i = 0
loop do
  i += 1
  puts i
  break        # this will cause execution to exit the loop
end
```

Листинг 15: Циклы

Циклы

```
1 while x >= 0
2   puts x
3   x -= 1
4 end
5
6 until x < 0
7   puts x
8   x -= 1
9 end
```

Листинг 16: Циклы 2

Циклы

```ruby
begin
  puts "Do you want to do that again?"
  answer = gets.chomp
end while answer == 'Y'

for i in 1..x do
  puts i
end

x = [1, 2, 3, 4, 5]

for i in x do
  puts i
end
```

Листинг 17: Циклы 3

Циклы

```
1 5.times do
2
3 end
4
5 n.times do |i|
6 end
7 # i = 0,1,2,3,4
```

Листинг 18: Циклы 4

next, redo

next - переходит к следующему шагу самого внутреннего цикла без проверки условий redo - начинает заново самый внутренний цикл без проверки условий

Хэши

```ruby

grades = { "Jane Doe" => 10, "Jim Doe" => 6 }
options = { :font_size => 10, :font_family =>
    "Arial" }
options = { font_size: 10, font_family: "Arial" }
options[:font_size] # => 10
grades = {}
grades = Hash.new(0)
grades = {"Timmy Doe" => 8}
grades.default = 0
```

Листинг 19: Инициализация хешей

Хэши

```ruby

h = { "n" => 100, "m" => 100, "y" => 300, "d" =>
    200, "a" => 0 }
h.invert  #=> {0=>"a", 100=>"m", 200=>"d",
    300=>"y"}
h = { "a" => 100, "b" => 200 }
h.has_key?("a") #=> true
h.has_key?("z") #=> false
h = { "a" => 100, "b" => 200 }
h.has_value?(100) #=> true
h.has_value?(999) #=> false
```

Листинг 20: Методы, которых нет у массивов

Слияние хэшей

```
1
2 h1 = { "a" => 100, "b" => 200 }
3 h2 = { "b" => 254, "c" => 300 }
4 h1.merge!(h2) #=> {"a"=>100, "b"=>254, "c"=>300}
5
6 h1 = { "a" => 100, "b" => 200 }
7 h2 = { "b" => 254, "c" => 300 }
8 h1.merge!(h2) { |key, v1, v2| v1 }
```

Листинг 21: Слияние

Rehash

```
1 a = [ "a", "b" ]
2 c = [ "c", "d" ]
3 h = { a => 100, c => 300 }
4 h[a]       #=> 100
5 a[0] = "z"
6 h[a]       #=> nil
7 h.rehash   #=> {["z", "b"]=>100, ["c", "d"]=>300}
8 h[a]       #=> 100
```

Листинг 22: Обработка исключений

Регулярные выражения

```
1 /hay/ =~ 'haystack' #=> 0
2 /y/.match('haystack') #=> #<MatchData "y">
3 /hay/ =~ 'haystack' #=> 0
4 'haystack' =~ /hay/ #=> 0
5 /a/   =~ 'haystack' #=> 1
6 /u/   =~ 'haystack' #=> nil
```

Листинг 23: Регулярные выражения

Спецсимволы

```ruby
/1 \+ 2 = 3\?/.match('Does 1 + 2 = 3?') #=>
    #<MatchData "1 + 2 = 3?">
place = "test"
/#{place}/.match("Go to test")#=> #<MatchData
    "test">
/W[aeiou]rd/.match("Word") #=> #<MatchData "Word">
/[0-9a-f]/.match('9f') #=> #<MatchData "9">
/[9f]/.match('9f')   #=> #<MatchData "9">
```

Листинг 24: Примеры использования

https://ruby-doc.org/core-2.4.1/Regexp.html

Файлы

```ruby
File.open('test.csv', 'r') do |f|
  while line = f.gets
    puts line
  end
end
File.readlines('foo.bar')

File.open('test.txt', 'w') do |f|
  # use "\n" for two lines of text
  f.puts "It will be \n two lines"
  f.write 'Just a string'
end

File.write('/path/to/file', 'Some glorious
    content')
```

Файлы

```ruby
File.absolute_path('ex5.rb')
File.atime('ex5.rb')
File.birthtime 'ex5.rb'
File.ctime 'ex5.rb'
File.mtime('ex5.rb') #=> Tue Apr 08 12:58:04 CDT 2003
File.utime(aTime, mTime, filename)
File.basename("/some/path/ruby.rb")      #=> "ruby.rb"
File.basename(/some/path/ruby.rb", ".rb") #=> "ruby"
File.basename("/some/path/ruby.rb", ".*") #=> "ruby"
File.zero? text.csv #exists and has zero size
```

Листинг 26: Методы класса File

Файлы

```
1 File.exist filename.txt
2 File.exists filename.txt #deprecated
3 File.extname #extension
4 File.identical?("test1", "test2")
5 File.join("usr", "mail", "me") #=> "usr/mail/me"
6 File.rename("afile", "afile.bak")
7 File.split("/home/me/.profile") #=> ["/home/me",
    ".profile"]
```

Листинг 27: Методы класса File 2