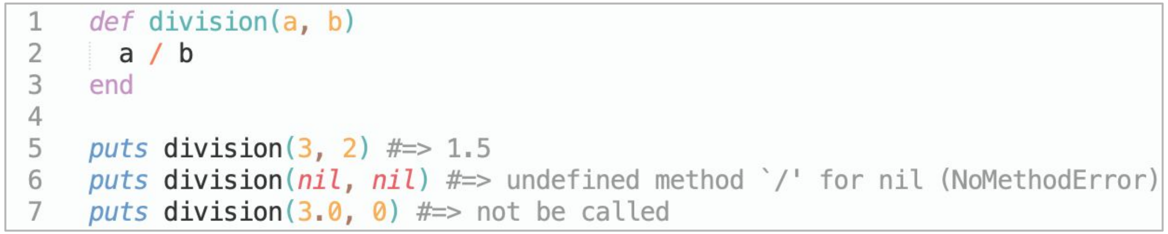
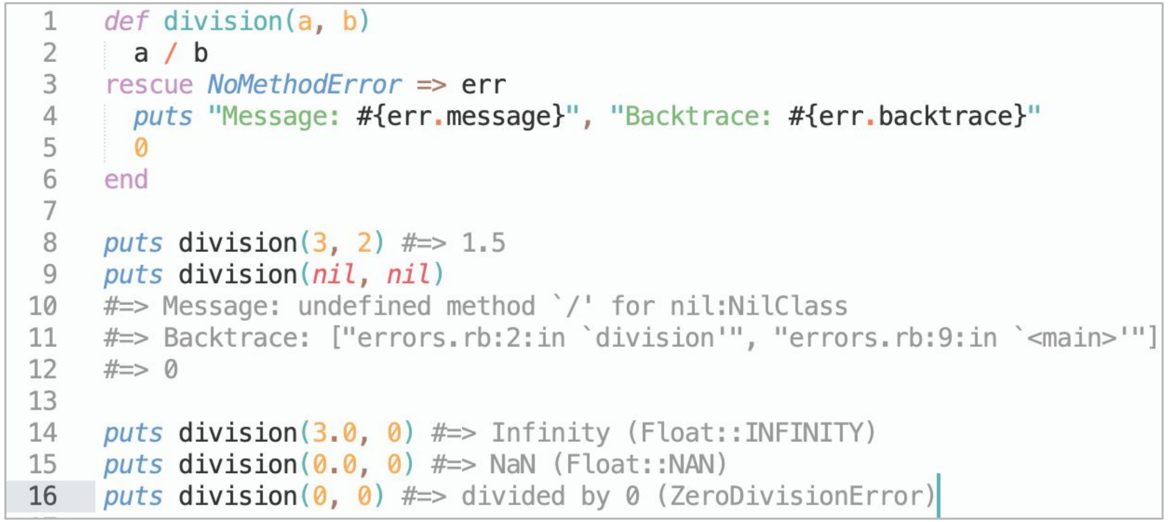
Обработка ошибок и блоков кода

Обработка различных ошибок, блоки, Proc и lambda объекты

Обработка исключений



```ruby
def division(a, b)
  a / b
end

puts division(3, 2) #=> 1.5
puts division(nil, nil) #=> undefined method `/' for nil (NoMethodError)
puts division(3.0, 0) #=> not be called
```



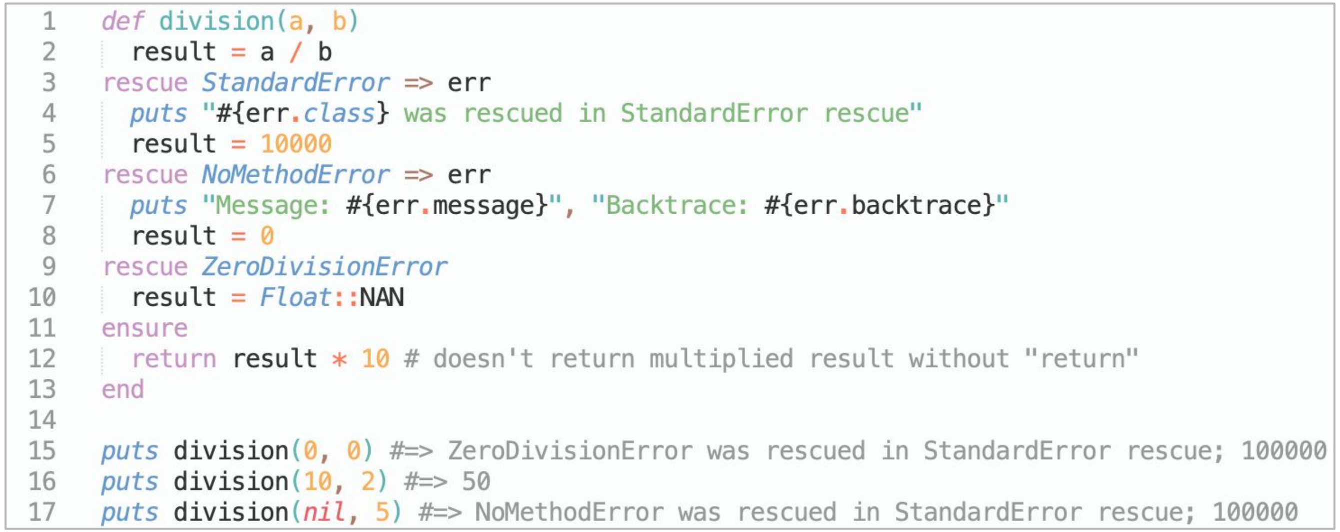
```ruby
def division(a, b)
  a / b
rescue NoMethodError => err
  puts "Message: #{err.message}", "Backtrace: #{err.backtrace}"
  0
end

puts division(3, 2) #=> 1.5
puts division(nil, nil)
#=> Message: undefined method `/' for nil:NilClass
#=> Backtrace: ["errors.rb:2:in `division'", "errors.rb:9:in `<main>'"]
#=> 0

puts division(3.0, 0) #=> Infinity (Float::INFINITY)
puts division(0.0, 0) #=> NaN (Float::NAN)
puts division(0, 0) #=> divided by 0 (ZeroDivisionError)
```

Выполнение общего ensure кода

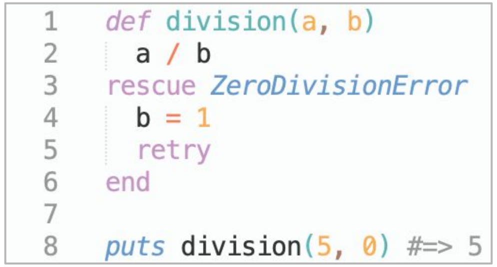
```ruby
def division(a, b)
  a / b
rescue NoMethodError => err
  puts "Message: #{err.message}", "Backtrace: #{err.backtrace}"
  0
rescue ZeroDivisionError
  Float::NAN
end

puts division(0, 0) #=> NaN
```

```ruby
def division(a, b)
  result = a / b
rescue NoMethodError => err
  puts "Message: #{err.message}", "Backtrace: #{err.backtrace}"
  result = 0
rescue ZeroDivisionError
  result = Float::NAN
ensure
  return result * 10 # doesn't return multiplied result without "return"
end

puts division(0, 0) #=> NaN
puts division(10, 2) #=> 50
puts division(nil, 5) #=> 0
```

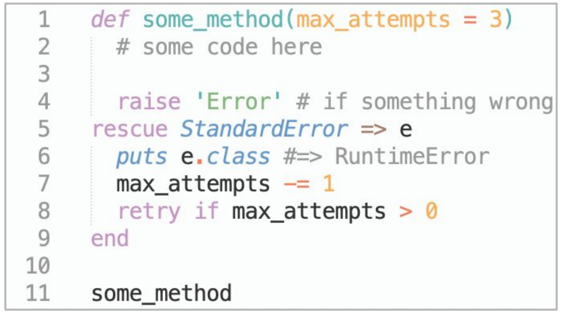
StrandardError и ключевое слово retry



```ruby
def division(a, b)
  result = a / b
rescue StandardError => err
  puts "#{err.class} was rescued in StandardError rescue"
  result = 10000
rescue NoMethodError => err
  puts "Message: #{err.message}", "Backtrace: #{err.backtrace}"
  result = 0
rescue ZeroDivisionError
  result = Float::NAN
ensure
  return result * 10 # doesn't return multiplied result without "return"
end

puts division(0, 0) #=> ZeroDivisionError was rescued in StandardError rescue; 100000
puts division(10, 2) #=> 50
puts division(nil, 5) #=> NoMethodError was rescued in StandardError rescue; 100000
```

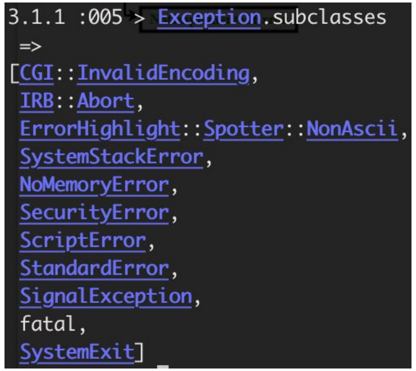


```ruby
def division(a, b)
  a / b
rescue ZeroDivisionError
  b = 1
  retry
end

puts division(5, 0) #=> 5
```

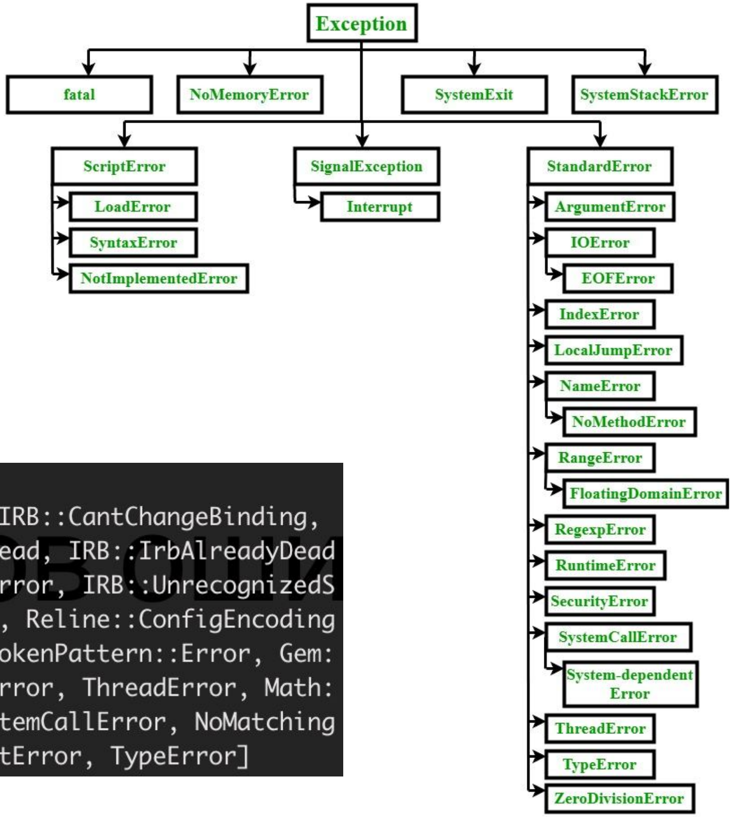


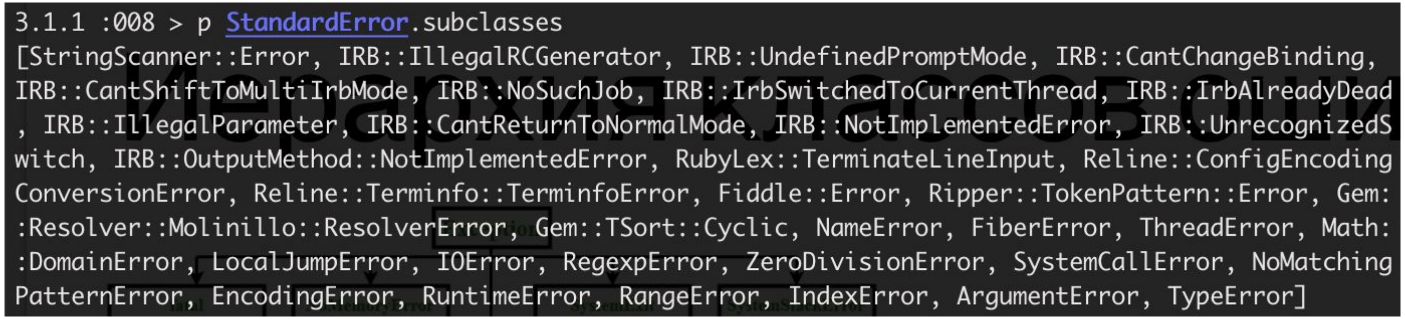
```ruby
def some_method(max_attempts = 3)
  # some code here

  raise 'Error' # if something wrong
rescue StandardError => e
  puts e.class #=> RuntimeError
  max_attempts -= 1
  retry if max_attempts > 0
end

some_method
```

Иерархия классов ошибок



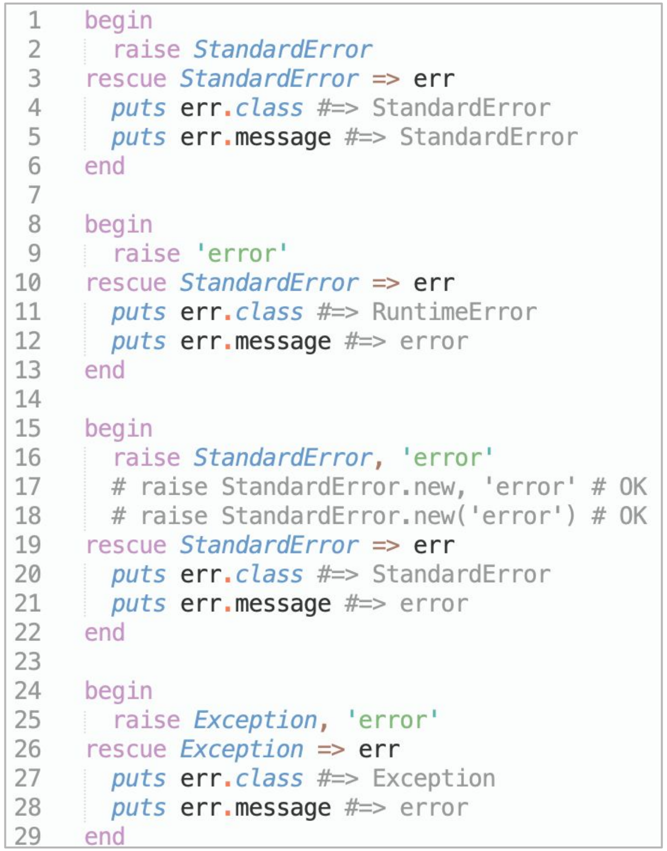
```
3.1.1 :005 > Exception.subclasses
 =>
[CGI::InvalidEncoding,
 IRB::Abort,
 ErrorHighlight::Spotter::NonAscii,
 SystemStackError,
 NoMemoryError,
 SecurityError,
 ScriptError,
 StandardError,
 SignalException,
 fatal,
 SystemExit]
```



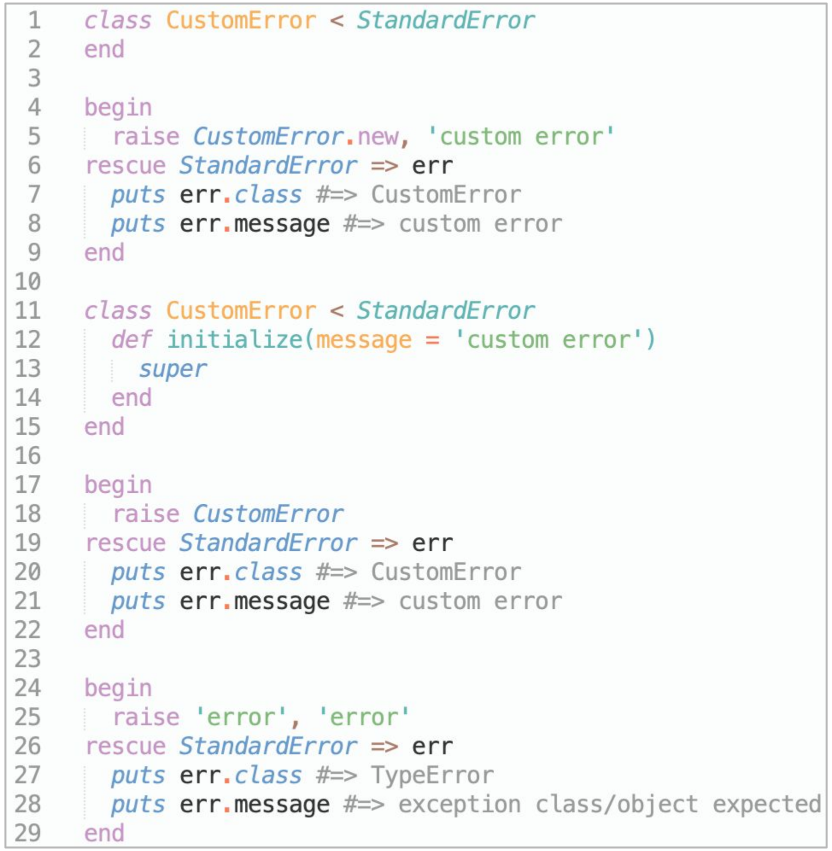


```
3.1.1 :008 > p StandardError.subclasses
[StringScanner::Error, IRB::IllegalRCGenerator, IRB::UndefinedPromptMode, IRB::CantChangeBinding,
IRB::CantShiftToMultiIrbMode, IRB::NoSuchJob, IRB::IrbSwitchedToCurrentThread, IRB::IrbAlreadyDead
, IRB::IllegalParameter, IRB::CantReturnToNormalMode, IRB::NotImplementedError, IRB::UnrecognizedS
witch, IRB::OutputMethod::NotImplementedError, RubyLex::TerminateLineInput, Reline::ConfigEncoding
ConversionError, Reline::Terminfo::TerminfoError, Fiddle::Error, Ripper::TokenPattern::Error, Gem:
:Resolver::Molinillo::ResolverError, Gem::TSort::Cyclic, NameError, FiberError, ThreadError, Math:
:DomainError, LocalJumpError, IOError, RegexpError, ZeroDivisionError, SystemCallError, NoMatching
PatternError, EncodingError, RuntimeError, RangeError, IndexError, ArgumentError, TypeError]
```

Собственные классы ошибок

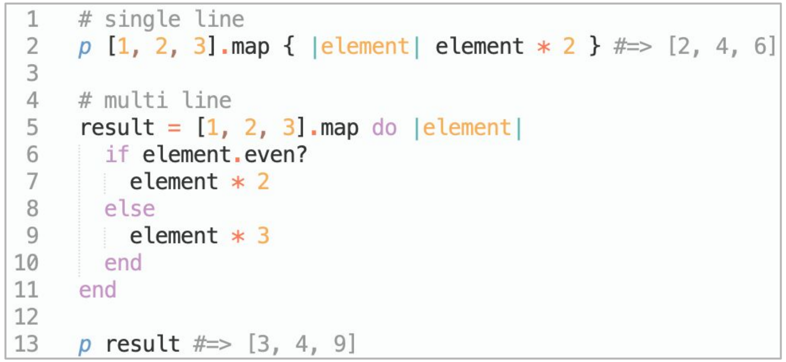


```ruby
begin
  raise StandardError
rescue StandardError => err
  puts err.class #=> StandardError
  puts err.message #=> StandardError
end

begin
  raise 'error'
rescue StandardError => err
  puts err.class #=> RuntimeError
  puts err.message #=> error
end

begin
  raise StandardError, 'error'
  # raise StandardError.new, 'error' # OK
  # raise StandardError.new('error') # OK
rescue StandardError => err
  puts err.class #=> StandardError
  puts err.message #=> error
end

begin
  raise Exception, 'error'
rescue Exception => err
  puts err.class #=> Exception
  puts err.message #=> error
end
```



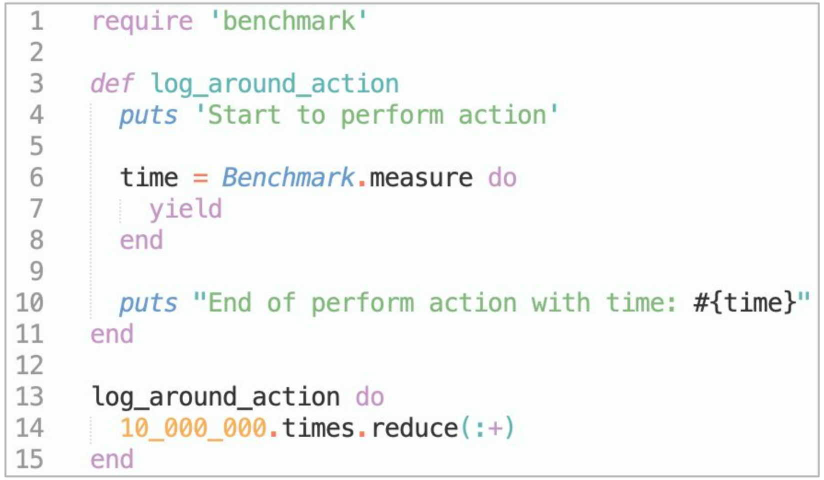
```ruby
class CustomError < StandardError
end

begin
  raise CustomError.new, 'custom error'
rescue StandardError => err
  puts err.class #=> CustomError
  puts err.message #=> custom error
end

class CustomError < StandardError
  def initialize(message = 'custom error')
    super
  end
end

begin
  raise CustomError
rescue StandardError => err
  puts err.class #=> CustomError
  puts err.message #=> custom error
end

begin
  raise 'error', 'error'
rescue StandardError => err
  puts err.class #=> TypeError
  puts err.message #=> exception class/object expected
end
```

Конструкция block и yield keyword

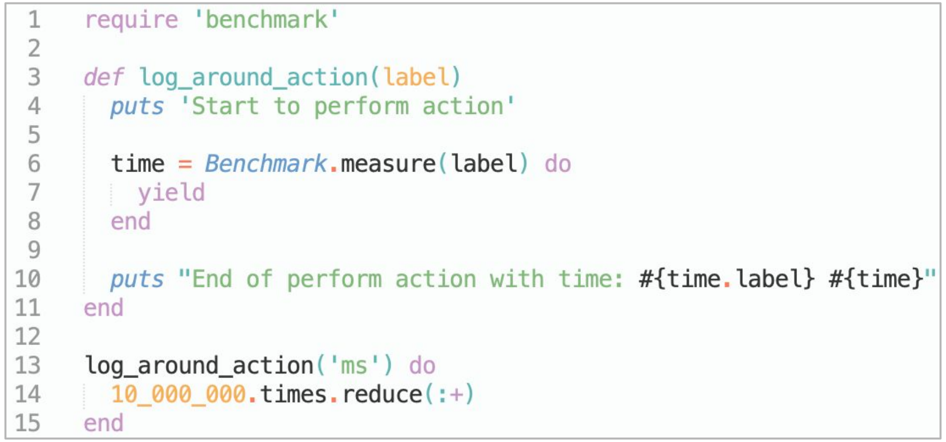


```ruby
# single line
p [1, 2, 3].map { |element| element * 2 } #=> [2, 4, 6]

# multi line
result = [1, 2, 3].map do |element|
  if element.even?
    element * 2
  else
    element * 3
  end
end

p result #=> [3, 4, 9]
```

- блок не является классом, это языковая конструкция;
- содержимое блоков заключается в конструкции do / end или {};
- в блоки можно передавать параметры.

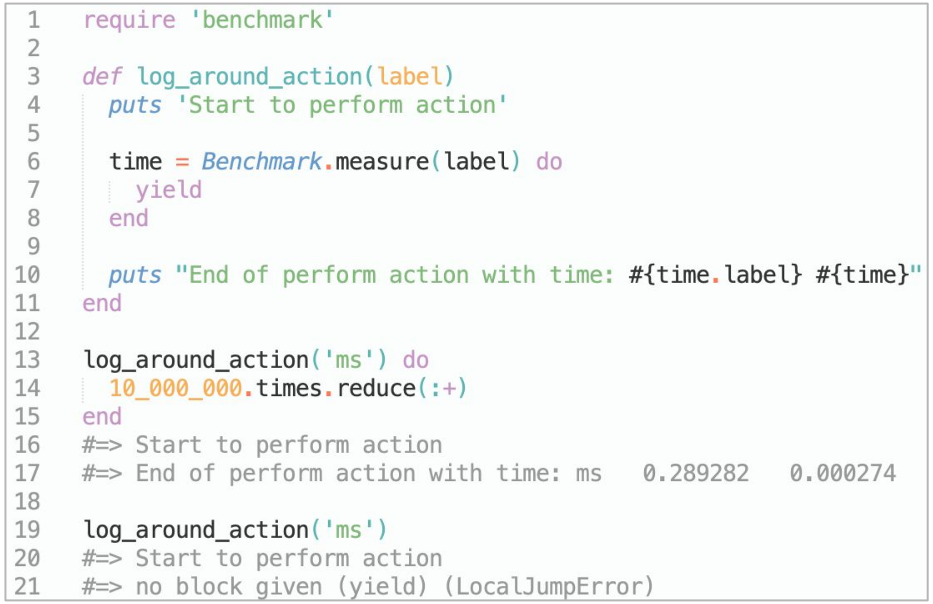


```ruby
require 'benchmark'

def log_around_action
  puts 'Start to perform action'

  time = Benchmark.measure do
    yield
  end

  puts "End of perform action with time: #{time}"
end

log_around_action do
  10_000_000.times.reduce(:+)
end
```

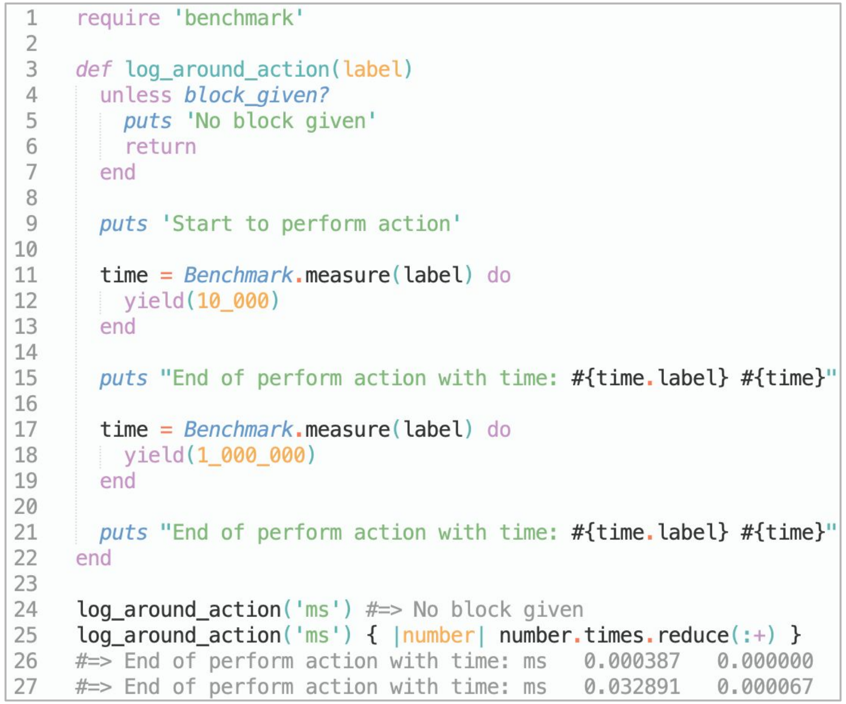


```ruby
require 'benchmark'

def log_around_action(label)
  puts 'Start to perform action'

  time = Benchmark.measure(label) do
    yield
  end

  puts "End of perform action with time: #{time.label} #{time}"
end

log_around_action('ms') do
  10_000_000.times.reduce(:+)
end
```

Проверка наличия блока



```ruby
require 'benchmark'

def log_around_action(label)
  puts 'Start to perform action'

  time = Benchmark.measure(label) do
    yield
  end

  puts "End of perform action with time: #{time.label} #{time}"
end

log_around_action('ms') do
  10_000_000.times.reduce(:+)
end
#=> Start to perform action
#=> End of perform action with time: ms   0.289282   0.000274

log_around_action('ms')
#=> Start to perform action
#=> no block given (yield) (LocalJumpError)
```



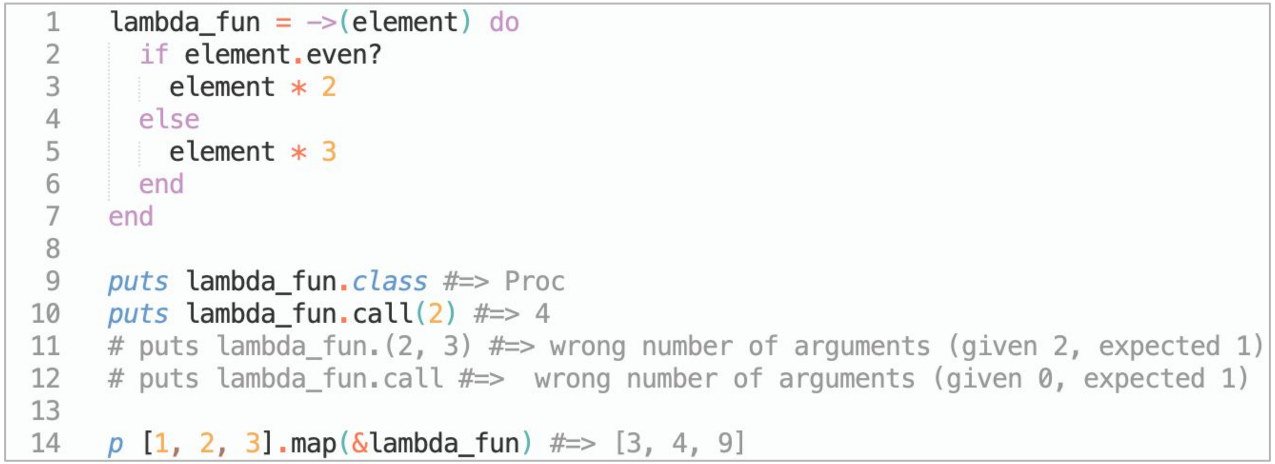
```ruby
require 'benchmark'

def log_around_action(label)
  unless block_given?
    puts 'No block given'
    return
  end

  puts 'Start to perform action'

  time = Benchmark.measure(label) do
    yield(10_000)
  end

  puts "End of perform action with time: #{time.label} #{time}"

  time = Benchmark.measure(label) do
    yield(1_000_000)
  end

  puts "End of perform action with time: #{time.label} #{time}"
end

log_around_action('ms') #=> No block given
log_around_action('ms') { |number| number.times.reduce(:+) }
#=> End of perform action with time: ms   0.000387   0.000000
#=> End of perform action with time: ms   0.032891   0.000067
```

Proc-объект

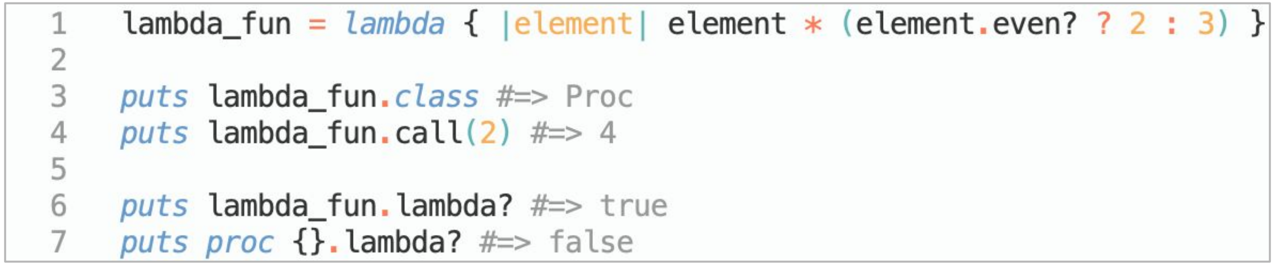
```ruby
proc_fun = Proc.new do |element|
  if element.even?
    element * 2
  else
    element * 3
  end
end

puts proc_fun.class #=> Proc
puts proc_fun.call(2) #=> 4
puts proc_fun.(2, 3) #=> 4
# puts proc_fun.call #=> undefined method `even?' for nil

p [1, 2, 3].map(&proc_fun) #=> [3, 4, 9]
```

```ruby
proc_fun = proc { |element| element * (element.even? ? 2 : 3) }

puts proc_fun.class #=> Proc
puts proc_fun.call(2) #=> 4
```

Lambda-объект

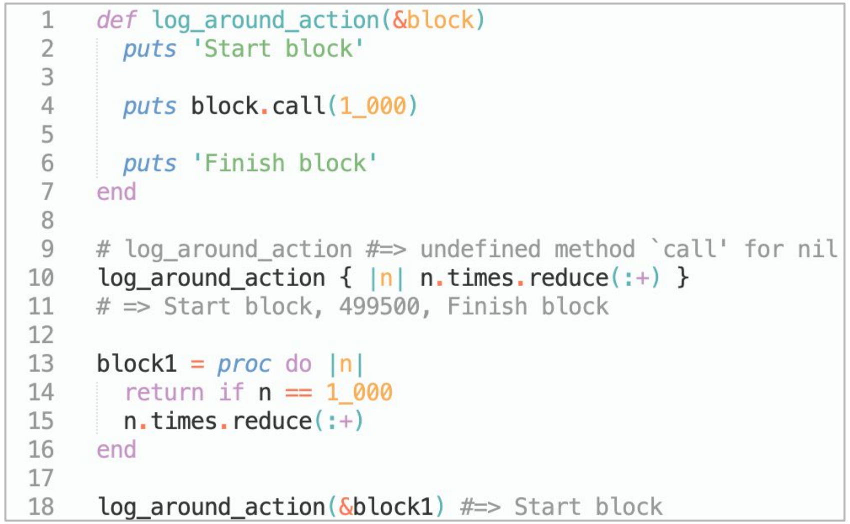


```ruby
lambda_fun = ->(element) do
  if element.even?
    element * 2
  else
    element * 3
  end
end

puts lambda_fun.class #=> Proc
puts lambda_fun.call(2) #=> 4
# puts lambda_fun.(2, 3) #=> wrong number of arguments (given 2, expected 1)
# puts lambda_fun.call #=>  wrong number of arguments (given 0, expected 1)

p [1, 2, 3].map(&lambda_fun) #=> [3, 4, 9]
```

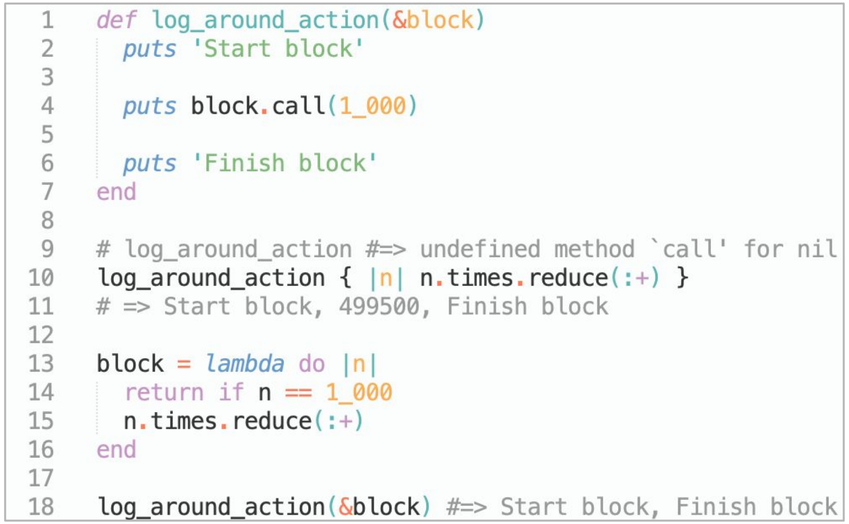


```ruby
lambda_fun = lambda { |element| element * (element.even? ? 2 : 3) }

puts lambda_fun.class #=> Proc
puts lambda_fun.call(2) #=> 4

puts lambda_fun.lambda? #=> true
puts proc {}.lambda? #=> false
```

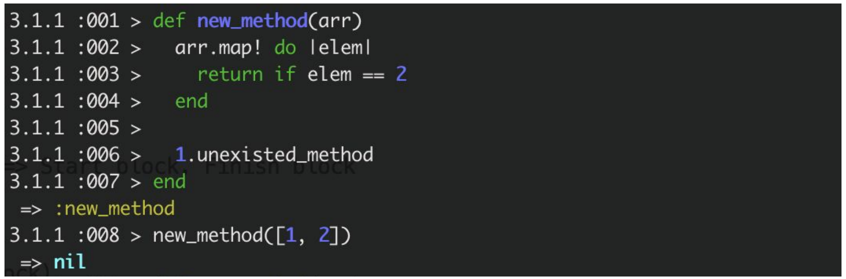
Поведение proc и lambda объектов



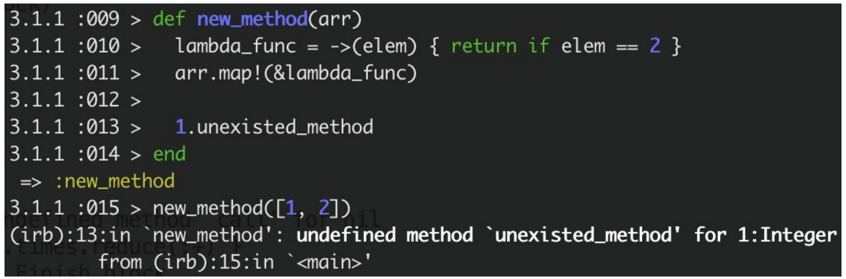
```ruby
def log_around_action(&block)
  puts 'Start block'

  puts block.call(1_000)

  puts 'Finish block'
end

# log_around_action #=> undefined method `call' for nil
log_around_action { |n| n.times.reduce(:+) }
# => Start block, 499500, Finish block

block1 = proc do |n|
  return if n == 1_000
  n.times.reduce(:+)
end

log_around_action(&block1) #=> Start block
```



```ruby
def log_around_action(&block)
  puts 'Start block'

  puts block.call(1_000)

  puts 'Finish block'
end

# log_around_action #=> undefined method `call' for nil
log_around_action { |n| n.times.reduce(:+) }
# => Start block, 499500, Finish block

block = lambda do |n|
  return if n == 1_000
  n.times.reduce(:+)
end

log_around_action(&block) #=> Start block, Finish block
```

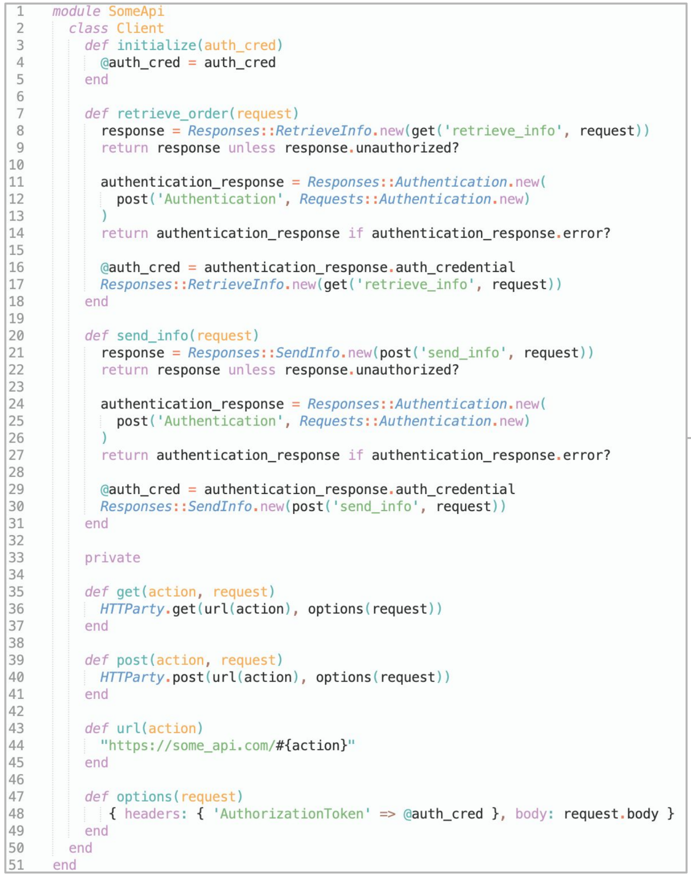


```
3.1.1 :001 > def new_method(arr)
3.1.1 :002 >   arr.map! do |elem|
3.1.1 :003 >     return if elem == 2
3.1.1 :004 >   end
3.1.1 :005 >
3.1.1 :006 >   1.unexisted_method
3.1.1 :007 > end
 => :new_method
3.1.1 :008 > new_method([1, 2])
 => nil
```

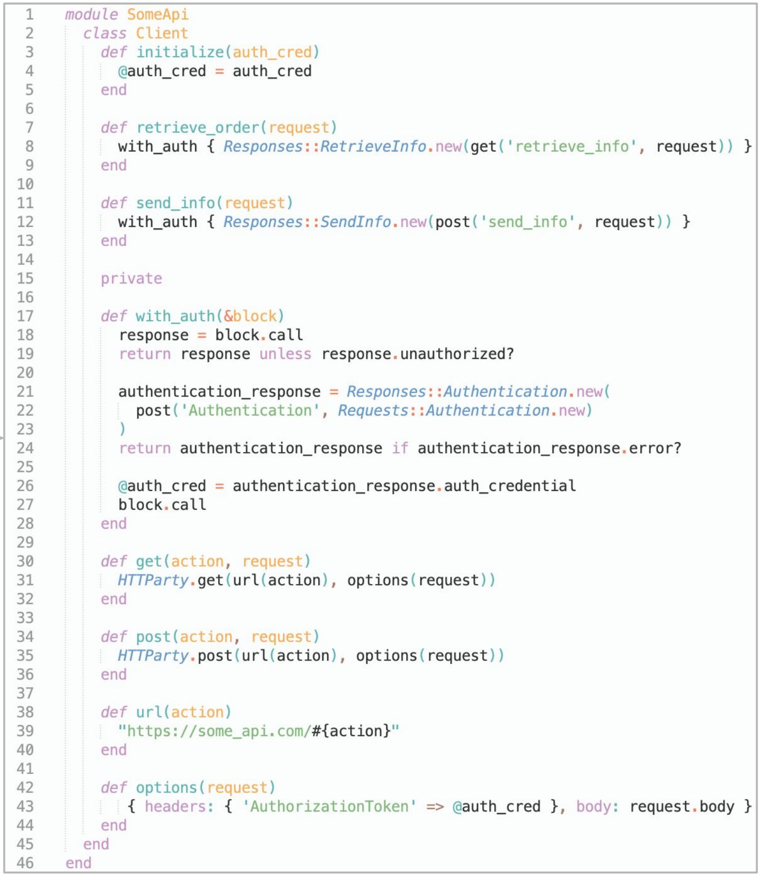


```
3.1.1 :009 > def new_method(arr)
3.1.1 :010 >   lambda_func = ->(elem) { return if elem == 2 }
3.1.1 :011 >   arr.map!(&lambda_func)
3.1.1 :012 >
3.1.1 :013 >   1.unexisted_method
3.1.1 :014 > end
 => :new_method
3.1.1 :015 > new_method([1, 2])
(irb):13:in `new_method': undefined method `unexisted_method' for 1:Integer
        from (irb):15:in `<main>'
```

Пример работы с блоком



```ruby
module SomeApi
  class Client
    def initialize(auth_cred)
      @auth_cred = auth_cred
    end

    def retrieve_order(request)
      response = Responses::RetrieveInfo.new(get('retrieve_info', request))
      return response unless response.unauthorized?

      authentication_response = Responses::Authentication.new(
        post('Authentication', Requests::Authentication.new)
      )
      return authentication_response if authentication_response.error?

      @auth_cred = authentication_response.auth_credential
      Responses::RetrieveInfo.new(get('retrieve_info', request))
    end

    def send_info(request)
      response = Responses::SendInfo.new(post('send_info', request))
      return response unless response.unauthorized?

      authentication_response = Responses::Authentication.new(
        post('Authentication', Requests::Authentication.new)
      )
      return authentication_response if authentication_response.error?

      @auth_cred = authentication_response.auth_credential
      Responses::SendInfo.new(post('send_info', request))
    end

    private

    def get(action, request)
      HTTParty.get(url(action), options(request))
    end

    def post(action, request)
      HTTParty.post(url(action), options(request))
    end

    def url(action)
      "https://some_api.com/#{action}"
    end

    def options(request)
      { headers: { 'AuthorizationToken' => @auth_cred }, body: request.body }
    end
  end
end
```



```ruby
module SomeApi
  class Client
    def initialize(auth_cred)
      @auth_cred = auth_cred
    end

    def retrieve_order(request)
      with_auth { Responses::RetrieveInfo.new(get('retrieve_info', request)) }
    end

    def send_info(request)
      with_auth { Responses::SendInfo.new(post('send_info', request)) }
    end

    private

    def with_auth(&block)
      response = block.call
      return response unless response.unauthorized?

      authentication_response = Responses::Authentication.new(
        post('Authentication', Requests::Authentication.new)
      )
      return authentication_response if authentication_response.error?

      @auth_cred = authentication_response.auth_credential
      block.call
    end

    def get(action, request)
      HTTParty.get(url(action), options(request))
    end

    def post(action, request)
      HTTParty.post(url(action), options(request))
    end

    def url(action)
      "https://some_api.com/#{action}"
    end

    def options(request)
      { headers: { 'AuthorizationToken' => @auth_cred }, body: request.body }
    end
  end
end
```

Полезные ссылки

https://www.tutorialspoint.com/ruby/ruby_exceptions.htm# – Описание способов отлавливания ошибок и создание своих классов для ошибок;

https://dev.to/okuramasafumi/be-sure-ensure-doesn-t-return-value-implicitly-8gp – Описание особенности возврата значения из метода из ensure блока;

https://ruby-doc.org/core-2.5.1/Exception.html –Документация по классу Exception и производным от него ошибкам;

https://www.rubyguides.com/2016/02/ruby-procs-and-lambdas/ – Описание блоков, Proc-ов и лямбд.

Конец! Спасибо!