

СПБПУ ПЕТРА ВЕЛИКОГО

# RUBY

Выполнили:

Бабахина Софья Александровна  
Басалаев Даниил Александрович  
Липс Екатерина Константиновна

Группа:

5030102/10201

Преподаватель:

Иванов Денис Юрьевич

Санкт-Петербург  
2024

# ПЛАН

1. Ветвления
2. Глобальные методы
3. Циклы
4. Блоки
5. Итераторы
6. Классы

# ВЕТВЛЕНИЕ [1/2]

## КЛЮЧЕВЫЕ СЛОВА IF, ELSIF, ELSE

```
1 puts 'Начало программы'
2
3 if puts 'Эта строка выведется'
4   puts 'А эта уже нет'
5 end
6
7 k = 0
8 if k == 1
9   puts 'Содержимое if-конструкции'
10 elsif k == 2
11   puts 'Содержимое else'
12 else
13   puts 'Содержимое elif'
14 end
15
16 puts 'Завершение программы'
17
18 puts 'if-модификатор' if k == 0
```

Output:

Начало программы  
Эта строка выведется  
Содержимое elif  
Завершение программы  
if-модификатор

## ВЕТВЛЕНИЕ [2/2]

```
1 k = 1
2
3 puts (if k == 0
4       'if'
5     else
6       'else'
7     end)
8
9 var1 = if k == 1
10        'if'
11      else
12        'else'
13      end
14
15 if k == 2
16   var2 = 'if'
17 else
18   var2 = 'else'
19 end
20
21 puts var1, var2
```

Output:

```
else
if
else
```

```
1 if x = 1
2   y = 'hello'
3 end
4 puts x
5 puts y
6
```

Output:

```
1
hello
```

```
1 if false
2   y = 'Hello, world!'
3 end
4 p y
5 p z
6
7
```

Output:

```
nil
```

```
test.rb:8:in `<main>': undefined local variable or method `z'
```

# ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Оператор	Описание
<code>x &amp;&amp; y</code>	Логическое И возвращает <code>true</code> , если оба операнда: <code>x</code> и <code>y</code> — истинны, в противном случае возвращается <code>false</code>
<code>x and y</code>	Логическое И, отличающееся от оператора <code>&amp;&amp;</code> меньшим приоритетом
<code>x    y</code>	Логическое ИЛИ возвращает <code>true</code> , если хотя бы один из операндов: <code>x</code> и <code>y</code> — истинен. Если оба операнда ложны, оператор возвращает <code>false</code>
<code>x or y</code>	Логическое ИЛИ, отличающееся от оператора <code>  </code> меньшим приоритетом
<code>! x</code>	Возвращает либо <code>true</code> , если <code>x</code> ложен, либо <code>false</code> , если <code>x</code> истинен

# ЛОГИЧЕСКОЕ ОТРИЦАНИЕ

```
1 ▾ if RUBY_VERSION != '2.4.0'  
2   puts 'Некорректная версия'  
3 end  
4  
5 ▾ if not RUBY_VERSION == '2.4.0'  
6   puts 'Некорректная версия'  
7 end  
8  
9 ▾ if !(RUBY_VERSION == '2.4.0')  
10  puts 'Некорректная версия'  
11 end  
12  
13 ▾ unless RUBY_VERSION == '2.4.0'  
14  puts 'Некорректная версия'  
15 end
```

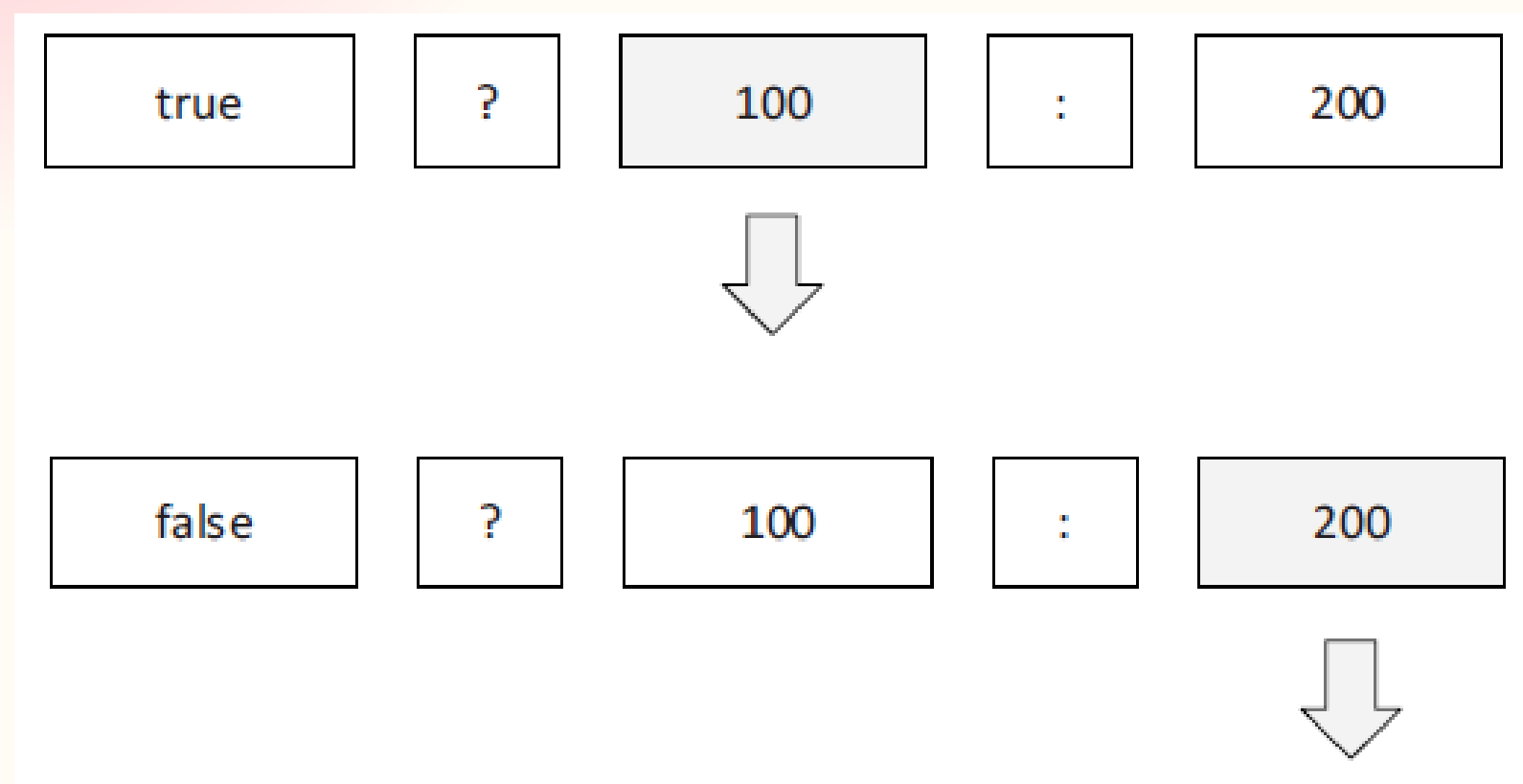
Output:

Некорректная версия  
Некорректная версия  
Некорректная версия  
Некорректная версия

```
puts 'Некорректная версия' unless RUBY_VERSION == '2.4.
```

# УСЛОВНЫЙ ОПЕРАТОР

## $X ? Y : Z$



```
1 var = if rand(0..1) == 0
2     100
3 else
4     200
5 end
6
7 var = rand(0..1) == 0 ? 100 : 200
```



## КЛЮЧЕВОЕ СЛОВО CASE

```
1 number = rand(0..2)
2 puts number
3
4 ▾ case number
5 ▾ when 0
6   puts 'Ноль';
7 ▾ when 1
8   puts 'Единица'
9 ▾ else
10  puts 'Два'
11 end
```

Output:

2

Два



## КЛЮЧЕВОЕ СЛОВО CASE

```
1 number = rand(0..2)
2 puts number
3
4 ▾ case number
5 ▾ when 0
6   puts 'Ноль';
7 ▾ when 1
8   puts 'Единица'
9 ▾ else
10  puts 'Два'
11 end
```

Output:

2

Два

# ГЛОБАЛЬНЫЕ МЕТОДЫ [1/5]

**def** + название метода в snake-режиме + телом метод (любое Ruby-выражение) + **end**

```
3 ▾ def convert
4   5 * 1000
5   end
6   puts convert
```

Output:

5000

```
22 ▾ def convert(value)
23   value * 1000
24   end
25   puts convert(11)
```

Output:

11000

```
38 ▾ def multi_params(x, y, *params)
39   p x # 1
40   p y # 2
41   p params # [3, 4]
42   end
43   multi_params(1, 2, 3, 4)
```

Output:

1

2

[3, 4]

## ГЛОБАЛЬНЫЕ МЕТОДЫ [2/5]

Ruby поддерживает специальное ключевое слово **return** для возврата значений из методов.

```
106 ▾ def convert(value, factor)
107   return value * factor
108   end
109   puts convert(11, 1024)
110
```

11264

```
106 ▾ def convert(value, factor)
107   value * factor
108   end
109   puts convert(11, 1024)
110
```

Output:

11264

Дело в том, что метод и так **возвращает результат, полученный в последнем выражении метода**.

Более того, соглашения, принятые в Ruby-сообществе, требуют исключать `return` в последней строчке метода. Ключевое слово `return` находит применение в случае, когда необходимо досрочно покинуть метод

## ГЛОБАЛЬНЫЕ МЕТОДЫ [3/5]

Ruby предоставляет специальное ключевое слово **alias**, при помощи которого можно создавать для методов псевдонимы.

<pre>155 ▾ def convert(value:, factor: 1000) 156   value * factor 157   end 158   alias kg_to_grams convert 159   alias kb_to_bytes convert 160   puts kg_to_grams(value: 5) 161   puts kb_to_bytes(value: 11, factor: 1024) 162</pre>	<p>Output:</p> <p>5000</p> <p>11264</p>
--	---

Важно помнить, что **alias** — это ключевое слово, а не метод.  
Поэтому между первым и вторым аргументами запятая не ставится.

# ГЛОБАЛЬНЫЕ МЕТОДЫ [4/5]

## Рекурсия

При помощи ключевого слова **undef** можно удалить метод.

```
def convert(value:, factor: 1000)
  value * factor
end

puts convert(value: 5) # 5000

undef convert

puts convert(value: 5), # undefined method `convert'
```

0	def factorial(number)	Output:
1	return 1 if number <= 1	
2	number * factorial(number - 1)	120
3	end	
4	puts factorial(5)	
5		

## Список predefined методов Ruby:

- puts, p и print
- local\_variables, global\_variables, instance\_variables и class\_variables
- require и require\_relative
- sprintf и format



# ГЛОБАЛЬНЫЕ МЕТОДЫ [5/5]

Для остановки программы предназначены методы **exit** и **abort**:

210 at_exit { puts 'Завершение программы' } 211 puts 'Эта строка выводится.' 212 abort 'Аварийная остановка программы' 213 puts 'А эта уже нет!' 214 215 216 ...	Output:  Эта строка выводится. Завершение программы  Аварийная остановка программы
---	---

Метод **sleep**:

```
puts 'Подождем 5 секунд'  
sleep 5  
puts 'А теперь еще 3.5 секунды'  
sleep 3.5  
puts 'Работа программы завершена'
```

**STDIN** и глобальная переменная **\$stdin**

```
print 'Введите, пожалуйста, имя файла '  
filename = gets.chomp  
  
$stdin = File.open(filename)  
  
puts gets('$')
```

**Чтение входного потока**

val = gets  
val = gets.to\_i  
и другие.

# ЦИКЛЫ [1/1]

```
1 print 'Пожалуйста, введите количество повторов: '
2 max_iterates = gets.to_i
3 i = 0
4 while i < max_iterates do
5   puts 'Hello, world!'
6   i += 1
7 end
8
9 i = 0
10 begin
11   puts 'Hi, world!'
12   i += 1
13 end while i < max_iterates
14
15 i = 1
16 size = (max_iterates * max_iterates).to_s.size + 1
17 while i <= max_iterates
18   j = 1
19   while j <= max_iterates
20     print format("% #{size}d ", i * j)
21     j += 1
22   end
23   i += 1
24   puts
25 end
26
```

```
27 i = 0
28 while i < max_iterates do
29   puts 'Hello, world! only 1 time'
30   i += 1
31   break if i >= 1
32 end
33
34 while i < max_iterates do
35   i += 1
36   next if i.even?
37   puts 'Hello, world! only if no even'
38 end
39
40 until i >= max_iterates
41   puts 'Hello, world! [until]'
42   i += 1
43 end
44
45 for x in [1, 2, 3, 4]
46   puts x
47 end
```

```
Пожалуйста, введите количество повторов: 3
Hello, world!
Hello, world!
Hello, world!
Hi, world!
Hi, world!
Hi, world!
 1  2  3
 2  4  6
 3  6  9
Hello, world! only 1 time
Hello, world! only if no even
1
2
3
4
```



# ИТЕРАТОРЫ [1/4]

Итератор **each** эквивалентен циклу **for** — он пробегает от первого до последнего элемента массива и на каждой итерации в блок передается очередной элемент массива.

```
1 rainbow = %w[red orange yellow green gray indigo violet]
2 for color in rainbow
3   puts color
4 end
```

```
1 rainbow = %w[red orange yellow green gray indigo violet]
2 rainbow.each { |color| puts color }
```

red  
orange  
yellow  
green  
gray  
indigo  
violet

СПБПУ ПЕТРА ВЕЛИКОГО

# ИТЕРАТОРЫ [2/4]

- Итератор **times** применяется к целым числам и позволяет выполнить блок указанное в числе количество раз
- Для итерирования от одного числа к другому применяется итератор **upto**
- Итератор **downto** позволяет пробегать числа с шагом -1

```
1 5.times { |i| puts i }
```

```
0  
1  
2  
3  
4
```

```
1 5.upto(10) { |i| puts i }
```

```
5  
6  
7  
8  
9  
10
```

```
1 10.downto(5) { |i| puts i }
```

```
10  
9  
8  
7  
6  
5
```

## ИТЕРАТОРЫ [3/4]

**map** итератор возвращает новую коллекцию, количество элементов в которой совпадает с количеством элементов в исходной коллекции.

Однако каждый элемент заменен на результат вычисления в блоке.

```
1 result = [1, 2, 3, 4, 5].map { |x| x + 1 }  
2 p result
```

```
[2, 3, 4, 5, 6]
```

# ИТЕРАТОРЫ [4/4]

Для того чтобы отфильтровать содержимое массива используются итераторы:

**select** и **reject**.

Для этих итераторов также предусмотрены синонимы:

**find\_all** и **delete\_if** соответственно.

```
1 p [1, 2, 3, 4, 5].select { |x| x.even? } # [2, 4]
2 p [1, 2, 3, 4, 5].reject { |x| x.even? } # [1, 3, 5]
```

СПБПУ ПЕТРА ВЕЛИКОГО

## БЛОКИ [1/4]

Для того чтобы метод мог принимать блок, в нем необходимо воспользоваться ключевым словом **yield**

```
1 ▾ def my_loop  
2   puts 'Начало метода'  
3   yield  
4   puts 'Завершение метода'  
5 end  
6 my_loop { puts 'Hello, world!' }
```

```
Начало метода  
Hello, world!  
Завершение метода
```



## БЛОКИ [2/4]

После ключевого слова **yield** можно указывать произвольное количество аргументов.  
Аргументы подставляются в соответствующие параметры блока

```
1 ▾ def my_loop
2   n = 0
3   yield n += 1
4   yield n += 1
5   yield n += 1
6 end
7 my_loop { |i| puts "#{i}: Hello, world!" }
```

```
1: Hello, world!
2: Hello, world!
3: Hello, world!
```

СПБПУ ПЕТРА ВЕЛИКОГО

## БЛОКИ [3/4]

Блоки тоже возвращают значение, которое можно получить как результат вызова **yield**

```
1 ▾ def greeting
2   name = yield
3   "Hello, #{name}!"
4 end
5
6 ▾ hello = greeting do
7   print 'Пожалуйста, введите имя '
8   gets.chomp
9 end
10 puts hello # Hello, Katya!
```



## БЛОКИ [4/4]

Объект **lb**, полученный при помощи метода **lambda**, может использоваться в качестве параметров блока

```
> lb = lambda { |n| n * n }  
=> #<Proc:0x00007f987108f438@(irb):3 (lambda)>  
> lb.call(3)  
=> 9
```

Чтобы подчеркнуть, что перед нами именно **лямбда**, можно использовать специальный синтаксис

```
> lb = ->(x) { x * x }  
=> #<Proc:0x00007febd50e0768@(irb):1 (lambda)>  
> lb.call(3)  
=> 9  
> (->(x) { x * x }).call(3)  
=> 9
```

# КЛАССЫ [1/3]

```
CustomClasses.rb +
1 class HelloWorld_1
2   def say
3     'hello'
4   end
5 end
6
7 greeting = HelloWorld_1.new
8 hello = HelloWorld_1.new
9
10 puts greeting.say
11 puts hello.say
12
```

Output:

hello  
hello  
hello  
hello

```
40 # переопределение методов класса
41
42 class HelloWorld
43   def say
44     'Определяем метод say в первый раз'
45   end
46
47   def say
48     'Определяем метод say во второй раз'
49   end
50 end
51
52 hello = HelloWorld.new
53
54 puts hello.say
```

Output:

Определяем метод say во второй раз

```
57 class HelloWorld
58   def say
59     'Определяем метод say в первый раз'
60   end
61 end
62
63 class HelloWorld
64   def say
65     'Определяем метод say во второй раз'
66   end
67 end
68
69 hello = HelloWorld.new
70
71 puts hello.say
```

Output:

Определяем метод say во второй раз

# КЛАССЫ [2/3]

```

74 puts 'abc'.reverse
75
76 class String
77   def reverse
78     'Изменяем поведение метода'
79   end
80 end
81
82 puts 'abc'.reverse
83
84
85 class String
86   def hello
87     "Hello, #{self}!"
88   end
89 end
90
91 puts 'world'.hello
92 puts 'Ruby'.hello
93 puts 'Igor'.hello

```

Output:

```

cba
Изменяем поведение метода
Hello, world!
Hello, Ruby!
Hello, Igor!

```

```

134 p Car::Engine      Car::Engine
135
136 p Engine           CustomClasses.rb:136:in `':
137

```

CustomClasses.rb:136:in `': uninitialized constant Engine (NameError)

```

98 class Car
99   def title
100     'BMW X7'
101   end
102   def description
103     'Новый BMW X7 с окраской кузова...'
104   end
105   def engine
106     @engine
107   end
108   def build
109     @engine = Engine.new
110   end
111   class Engine
112     def cylinders
113       6
114     end
115     def volume
116       3
117     end
118     def power
119       250
120     end
121   end
122 end
123
124 car = Car.new
125 car.build
126
127 puts car.title
128 puts car.description
129 puts car.engine.cylinders
130 puts car.engine.volume
131 puts car.engine.power

```

Output:

```

BMW X7
Новый BMW X7 с окраской кузова...
6
3
250

```

## КЛАССЫ [3/3]

```
140 ▾ class Ticket
141 ▾   def set_price(price)
142     @price = price
143   end
144 ▾   def price
145     @price
146   end
147 end
148
149 first = Ticket.new
150 second = Ticket.new
151
152 first.set_price(500)
153 second.set_price(600)
154
155 puts "Цена билета first: #{first.price}"
156 puts "Цена билета second: #{second.price}"
157
```

Output:

Цена билета first: 500  
Цена билета second: 600

```
174 ▾ class Ticket
175 ▾   def initialize(date:, price: 500)
176     @price = price
177     @date = date
178   end
179 ▾   def price
180     @price
181   end
182 ▾   def date
183     @date
184   end
185 end
186
187 first = Ticket.new(date: Time.mktime(2019, 5, 10, 10, 20))
188 puts "Дата билета first: #{first.date}"
189 puts "Цена билета first: #{first.price}"
190 second = Ticket.new(date: Time.mktime(2019, 5, 11, 10, 20), price: 600)
191 puts "Дата билета second: #{second.date}"
192 puts "Цена билета second: #{second.price}"
193
```

```
def new
  allocate
  initialize
end
```

Output:

Дата билета first: 2019-05-10 10:20:00 +0000  
Цена билета first: 500  
Дата билета second: 2019-05-11 10:20:00 +0000  
Цена билета second: 600





# GETTER, SETTER, ACCESSOR

```
218 ▾ class Ticket
219 ▾   def initialize(date:, price: 500)
220     @price = price
221     @date = date
222   end
223 ▾   def price=(price)
224     @price = price
225   end
226 ▾   def price
227     @price
228   end
229 ▾   def date=(date)
230     @date = date
231   end
232 ▾   def date
233     @date
234   end
235 end
236
237 ticket = Ticket.new(
238   date: Time.new(2019, 5, 10, 10, 20),
239   price: 500
240 )
241 ticket.price = 600
242 ticket.date = Time.new(2019, 5, 11, 10, 20)
243
244 puts "Цена билета: #{ticket.price}"
245 puts "Билет на дату: #{ticket.date}"
246
```

```
247 ▾ class Ticket_acc
248   attr_accessor :date, :price
249 ▾   def initialize(date:, price: 500)
250     @price = price
251     @date = date
252   end
253 end
254
255 ticket_acc = Ticket_acc.new(
256   date: Time.new(2019, 5, 10, 10, 20),
257   price: 500
258 )
259 ticket_acc.price = 600
260 ticket_acc.date = Time.new(2019, 5, 11, 10, 20)
261
262 puts "Цена билета ticket_acc: #{ticket_acc.price}"
263 puts "Билет на дату ticket_acc: #{ticket_acc.date}"

```

Output:

```
Цена билета: 600
Билет на дату: 2019-05-11 10:20:00 +0000
Цена билета ticket_acc: 600
Билет на дату ticket_acc: 2019-05-11 10:20:00 +0000
```