



Краткий справочник по языку программирования **Ruby**

О языке:

- ◆ Название: **Ruby** (англ. ruby ['ru:bi] [ру́би]— рубин). Название навеяно языком Perl, многие особенности синтаксиса и семантики из которого заимствованы в Ruby: англ. pearl — «жемчужина», ruby — «рубин».
- ◆ Когда создан: 1995 год.
- ◆ Создатель: японский программист **Юкихиро «Matz» Мацумото** (松本行弘 = まつもとゆきひろ).
- ◆ Характеристика: интерпретируемый, динамический, сценарный, высокоуровневый полностью объектно-ориентированный (мультипарадигменный) язык программирования.
- ◆ Лицензия: Ruby License. Кросс-платформенная реализация интерпретатора языка является полностью свободной.
- ◆ Испытал влияние: Perl 5, Smalltalk, Eiffel, Ada, Lisp, Python, Dylan, CLU, C++, Lua.
- ◆ Повлиял на: Groovy, Amber, CoffeeScript, Perl 6, Elixir, Crystal.
- ◆ Принципы:
 - Ориентация на разработчика, на то, как ему лучше решать программистские задачи.
 - POLS = Principle Of Least Surprise = «Правило наименьшего удивления» — программные конструкции должны быть логичными, интуитивно понятными и ожидаемыми.
 - TIMTOWTDI [Tim Toady] = There Is More Than One Way To Do It ~ «Есть не один способ это сделать» — один и тот же результат можно получить несколькими различными способами.
 - Потребность программиста создавать полезные и красивые программы как причина программирования.

Формат программы:

- ◆ Тексты программ сохраняются в файлах с суффиксом (расширением) **.rb**
- ◆ Команды (операторы) обычно записываются по одной в строке.
- ◆ Если нужно записать несколько команд в строке, они разделяются точкой-с-запятой (;).
- ◆ Все команды последовательно выполняются интерпретатором, даже объявления.
- ◆ Имена обычно состоят из латинских букв, цифр и знаков подчёркивания.
- ◆ Имена констант и классов начинаются с заглавной буквы, обычно пишутся без подчёркиваний.
- ◆ Имена переменных и методов начинаются со строчной буквы, обычно разделяются подчёркиваниями.
- ◆ Пробелы можно не вставлять, если это не меняет смысл выражения, но лучше вставлять их для удобства чтения человеком.
- ◆ Круглые скобки при вызове метода можно не писать.
- ◆ Для выделения вложенности (программных конструкций) обычно делают отступ в 2 пробела.

Запуск программы:

Программу на **Ruby** можно выполнить в терминальном окне, для чего его нужно запустить интерпретатор командой:

ruby / путь/к/файлу/с/исходным/текстом/программы **.rb**

Интерактивный интерпретатор:

Любые команды **Ruby** можно выполнить в диалоговом интерпретаторе, для чего его нужно запустить в терминальном окне командой:

irb

Константы, литералы, переменные

Описание	Обозначение	Пример	Пояснения
Литералы			
целое число (в десятичной системе счисления)	<i>цифры</i> + <i>цифры</i>	1234567 +1234567	положительное число
	<i>-цифры</i>	-1234567	отрицательное число
	<i>десятичные_цифры</i>	1_234_567	подчёркивание игнорируется
целое число (в восьмеричной системе счисления)	0 <i>восьмеричные_цифры</i>	04553	цифры: 01234567
целое число (в шестнадцатеричной системе счисления)	0x <i>шестнадцатеричные_цифры</i>	0x96b	цифры: 0123456789ABCDEF
целое число (в двоичной системе счисления)	0b <i>двоичные_цифры</i>	0b1001_0110_1011	цифры: 01
дробное число (в десятичной системе счисления)	<i>цифры</i> . <i>цифры</i>	3.141592653 0.1234567	точка отделяет дробную часть
дробное число в «научной» записи (экспоненциальной нотации)	<i>цифры</i> E <i>цифры</i> <i>цифры</i> e <i>цифры</i>	123e4 123e+4 +123E4 123e-4	123000.0 123000.0 123000.0 0.0123
строка	' <i>строка с кавычками (")</i> ' " <i>строка с апострофом (')</i> " %Q{строка}	'строка в апострофах' "строка в кавычках" " '@' %Q{The Ruby language}	без подстановки #{v} с подстановкой #{v} пустая строка с длиной == 0 строка из 1 символа "The Ruby language"
символ	: <i>символ</i>	sensor = :DS18B20	:DS18B20 != 'DS18B20'
логические (булевы) значения	true # истина false # ложь	flag = false	
диапазоны	(число..число) (число...число)	(1..5) (1...5)	1,2,3,4,5 1,2,3,4
Константа: присваивание значения	<i>Константа</i> = <i>значение</i>	PI = 3.141592653	имя — с заглавной буквы
Переменная: присваивание значения	<i>переменная</i> = <i>значение</i>	pi = PI	имя — с маленькой буквы
Переменная: использование значения	<i>переменная</i>	print(pi)	имя: латинские буквы, цифры и подчёркивания

Операции

Описание	Обозначение	Пример	Пояснения
Присваивание			
присваивание переменной	=	x = 1 x = 'Ruby'	переменная
параллельное присваивание		x, y, z = 100, 200, 500	
цепочка присваиваний		a = b = c = 3	
			результат – значение
Арифметические операции			
сложить, вычислить сумму	+	123.4 + 56.7 + 89	269.1
вычесть, вычислить разность	-	n - 1	n, уменьшенная на 1
умножить, найти произведение	*	2 * PI * r	длина окружности
разделить, вычислить частное от деления	/	42 / 5 42 / 5.0 42.0 / 5	8 8.4 8.4
вычислить остаток от деления	%	42 % 5	2
возвести в степень	**	2**5	32
изменение значения переменной	+= -= *= /= %= **=	n += 1 n -= 2 n *= 3 n /= 4 n %= 5 n **= 6	увеличить на 1 уменьшить на 2 увеличить в 3 раза уменьшить в 4 раза остаток от n/5 возвести n в 6-ю степень
Операции сравнения:			
равно	==	2*2 == 4	true
не равно	!=	2*2 != 4	false
больше	>	2*2 > 4	false
больше или равно	>=	2*2 >= 4	true
меньше	<	2*2 < 4	false
меньше или равно	<=	2*2 <= 4	true

Логические операции:			
И	&& and	true && true true and false	true false
ИЛИ	 or	true true true or false	true true
НЕ	! not	! true not false	false true
изменение с присваиванием	&&= =	variable &&= value variable = value	
Условная операция	<i>условие ? команда1 : команда2</i>	<code>z = (y != 0) ? x/y : 0</code>	
Операции со строками:			
повторение строки	<i>строка * число</i>	<code>print('Ypa! ' * 3)</code>	Ypa! Ypa! Ypa!
сцепление строк	<i>строка + строка</i>	<code>n=1; 'Значение N=' + n.to_s</code>	'Значение N=1'
дополнение	<i>строка << строка</i>	<code>s = "Ru"; s << "by"</code>	<code>s == "Ruby"</code>
подстановка в строку значения выражения	" #{выражение} "	<code>v=2; print "#{v}x#{v}=#{"v*v}"</code>	2x2=4
вычленение символа	<i>строка[индекс]</i>	<code>word[0]</code> <code>word[-1]</code>	1-й символ строки последний символ строки
выделение подстроки	<i>строка[индекс,длина]</i> <i>строка[диапазон]</i>	<code>word = 'Ruby'</code> <code>word[0, 1]</code> <code>word[1..2]</code> <code>'Ruby'[-2, 2]</code> <code>'Ruby'[-4..-3]</code>	# нумерация с 0 'R' 'ub' 'by' 'Ru'
замена символа	<i>строка[индекс]=символ</i>	<code>word[0] = 'r'</code> <code>word[-1]='ies'</code>	'ruby' 'rubies'
замена подстроки	<i>строка[строка]=новая_строка</i>	<code>word['R'] = 'ies'</code>	'Rubies'
длина строки	<i>строка.size</i> <i>строка.length</i>	<code>'Ruby'.size</code> <code>'Ruby'.length</code>	4 4
Преобразование в массив по разделителю	<i>массив = строка.split(разделитель)</i>	<code>'1;2;3'.split(';')</code> <code>'Ruby'.split "</code>	[1, 2, 3] ["R", "u", "b", "y"]
Операции побитовые:			
побитовое И	&		

побитовое ИЛИ		bits & mask	
побитовое исключительное ИЛИ	^	bits mask	
побитовое дополнение	~	bits ^ mask	
побитовый сдвиг влево	<<	bits << mask	
побитовый сдвиг вправо	>>	bits >> mask	
побитовое изменение с присваиванием	&= = ^= <<= >>=		

Преобразование значений:

преобразовать в строку	.to_s	42.to_s (25.5).to_s :symbol.to_s	"42" "25.5" "symbol"
преобразовать в целое	.to_i	(25.5).to_i 'один'.to_i	25 0
преобразовать в дробное	.to_f	25.to_f "1".to_f	25.0 1.0
преобразовать в тип «символ»	.to_sym	"symbol".to_sym (25.5).to_s.to_sym	:symbol :"25.5"
преобразовать в массив	.to_a	(1..12).to_a	[1,2,3,4,5,6,7,8,9,10,11,12]

Управляющие конструкции

Описание	Обозначение	Пример	Пояснения
Комментарии			
однострочный	# текст	# Программа: led.rb led = LED.new # светодиод	На всю строку от # и до конца строки
многострочный	=begin комментарии комментарии комментарии =end	=begin Программа: led.rb Автор: ШМВ Описание: «мигалка» =end	

специальный (shebang): имя программы-интерпретатора для выполнения	<i>#!/путь/к/интерпретатору/ruby</i>	<i>#!/usr/bin/ruby</i>	
Последовательность	<i>команда1 команда2</i>	<i>print "Привет, " print "Ruby!\n"</i>	Привет, Ruby!
	<i>команда1; команда2</i>	<i>s="рубины!"; puts s</i>	
Блоки			
Блок без параметров	<i>{ команда1; команда2 } do команда1 команда2 end</i>	<i>3.times { print("Ruby! ") }</i>	
Блок с параметром	<i>{ параметр команда1; команда2 } do параметр команда1 команда2 end</i>	<i>sorted = array.sort do a, b a <=> b end</i>	
Ветвление			
Ветвление по условию истинности	<i>if условие1 команды1 elsif условие2 команды2 else команды3 end</i>	<i>t = sensor.temperature() if (t > low_t) && (t < high_t) fan.switch_off() elsif (t > high_t) fan.switch_on() elsif (t > max_t) system('poweroff') else log.write(t) end</i>	
	<i>if условие1 then команды1 else команды2 end</i>	<i>if (t > max_t) then system('poweroff') end</i>	
Ветвление по условию ложности	<i>unless условие1 команды1 else команды2 end</i>	<i>unless (command=='stop') send(command) else stop_sending() end</i>	

	unless условие1 then команды1 else команды2 end		
Ветвление по значению выражения	case выражение when значение1 команды1 when значение2 команды2 else команды3 end	t = sensor.temperature() case t when t < high_t log.write(t) when t > high_t fan.switch_on() when t > max_t system('poweroff') end	
Повторение			
Цикл с условием продолжения (повторять, пока условие истинно)	while условие do команды end	n = 0 while n < 50 do n += 10 end	
	begin команды end while условие	n = 0 begin n += 10 end while n < 50	
Цикл с условием окончания (повторять, пока условие ложно)	until условие do команды end	n = 0 until n == 50 do n += 10 end	
	begin команды end until условие	n = 0 begin n += 10 end until n == 50	
Бесконечный цикл с оператором прерывания	loop do команды break if условие end	i = 0 loop do i += 1 break if i == 10 end	
Цикл перебора элементов списка	for переменная in список do команды end	for i in 0..9 { print(i) } for j in 0..9 do print(j)	0123456789

		end	
Итераторы	объект. итератор блок		
выполнить заданное количество раз	число. times блок	n.times { led.on; led.off }	Выполнится n раз
выполнять с увеличением от начального до конечного значения	число. upto число блок	0.upto n do i print(i) end	Выполнится с увеличением i от 0 до n
выполнять с уменьшением от начального до конечного значения	число. downto число блок	n.downto 0 do i print(i) end	Выполнится с уменьшением i от n до 0
	объект. each { элемент команды }	(1..5).each { print('Ruby') } [1,2,3].each { e print(e) }	Выполнится 5 раз Печать <i>каждый</i> элемент массива
Модификаторы команд:			
Выполнение команды при условии	команда if условие	system('halt') if (t > max_t)	
Выполнение команды не при условии	команда unless условие	print('OK') unless t > max_t	
Выполнение команды пока условие	команда while условие	work while dow != 'суббота'	
Выполнение команды до условия	команда until условие	work until dow == 'суббота'	

Агрегаты:

Описание	Обозначение	Пример	Пояснения
Массивы (списки)			
Литерал	[элемент0, элемент1, элемент2]	empty = [] list = [0, 'один', [2], :three]	Пустой массив Массив со значениями
Создание массива	массив = Array.new (размер, значение)	empty = Array.new array = Array.new(3) a = Array.new(3, rand())	[] [nil, nil, nil] Массив с 3-мя случайными
Добавление элемента в конец массива	массив << значение	a = []; a << 3.14; a << 2.718	[3.14, 2.718]
Изменение значения одномерного массива	массив[индекс] = значение	list[4] = 'IV' list[100] = 99+1	

Выборка значения одномерного массива	<i>значение = массив</i> [индекс]	one = list[1] print list[0]	
Создание многомерного массива (массива массивов)	<i>куб = </i> [массив1, массив2, массив3]	cube = [[11,12], [21,22], [31,33]]	
Изменение значения многомерного массива	<i>массив</i> [i1][i2] = значение	matrix[i][j] = rand()	
Выборка значения многомерного массива	<i>значение=массив</i> [i1][i2]	cell = matrix[i][j]	
Перебор элементов массива	<i>массив</i> .each do элемент <i>команды</i> end	[1, 2, 3].each do e print "#{e} " end	1 2 3
Перебор индексов массива	<i>массив</i> .each_index do индекс <i>команды</i> end	[1, 2, 3].each_index do i print "a[#{i}]=#{a[i]}; " end	a[0]=1; a[1]=2; a[2]=3;
Перебор элементов массива с индексами	<i>массив</i> .each_with_index do <i>элемент,индекс </i> <i>команды</i> end	[4, 5, 6].each_with_index do e, i print "arr[#{i}]=#{e}; " end	a[0]=4; a[1]=5; a[2]=6;
Преобразование элементов массива в строку с разделителем	<i>строка = массив</i> .join(разделитель)	s = [1,2,3].join(';')	'1;2;3'

Хэши (ассоциативные массивы, словари)

Литерал	{ <i>ключ1=>значение1,</i> <i>ключ2=>значение2</i> } { <i>ключ1:значение1,</i> <i>ключ2:значение2</i> }	hash = {'Ruby' => 1995, :author => 'Matz', 0.95 => '1995-12-21'}	точка отделяет дробную часть
Создание хэша	<i>хэш=</i> Hash.new	hash = Hash.new	
Изменение значения: ассоциация значения с ключом	<i>хэш</i> [ключ] = значение	hash[1.0] = '1996-12-25'	{1.0 => '1996-12-25' }
Выборка значения	<i>значение = хэш</i> [ключ]	released = hash[1.0]	
Удаление пары по ключу	<i>хэш</i> .delete <i>ключ</i>	hash.delete 'Python'	

Модули, классы, методы

Описание	Обозначение	Пример	Пояснения
----------	-------------	--------	-----------

Классы			
Описание класса	class <i>ИмяКласса</i> атрибуты методы end	class Sensor end	Класс описывает для объектов набор характеристик (свойства) и набор действий (методы).
Наследование	class <i>ПодКласс</i> < <i>НадКласс</i> end	class DHT11 < Sensor end	Подкласс наследует все описания надкласса и добавляет свои.
Атрибут (свойство) объектов	@ <i>имя_атрибута</i>	class Sensor @value = 0.0 end	У всех объектов одного класса один и тот же набор атрибутов, но у каждого объекта своё состояние (набор значений) атрибутов.
Объявления атрибутов	attr_reader : <i>имя_атрибута</i> attr_writer : <i>имя_атрибута</i> attr_accessor : <i>имя_атрибута</i>	attr_reader :ro_value attr_writer :wo_value attr_accessor :rw_value	v = object.ro_value object.wo_value = new_value v = object.rw_value object.rw_value = new_value
Метод объектов	def <i>имя_метода</i> (<i>параметры</i>) команды end	class DHT11 def read() @value = read_dht11() end end	Методы — это органы управления объектом.
Инициализатор для конструктора объекта	def initialize (<i>параметры</i>) end	def initialize(pin=18) @pin = pin @value = read() end	initialize() вызывается при создании объекта, ему передаются параметры Класс.new()
Вызов одноимённого метода надкласса	super	def method() super end	
Создание объекта класса	<i>объект</i> = <i>Класс</i> . new	sensor = Sensor.new(17)	
Вызов метода объекта	<i>объект</i> . метод()	t = sensor.read()	
Модули			
Описание модуля	class <i>ИмяМодуля</i> end	module TemperatureSensor end	

Наследование	class ПодМодуль < НадМодуль end		

Ввод-вывод: файлы, каталоги

Описание	Обозначение	Пример	Пояснения
Файлы			
Вывод в STDOUT (на экран)	print список, выражений	print "Перевод строки\n"	
Вывод в STDOUT (на экран) с новой строки	puts список, выражений	puts 'Ошибка чтения!'	
Отладочный вывод в STDOUT (на экран)	p список, выражений	p object	
Форматированный вывод в STDOUT	printf "формат вывода", список, выражений	printf "T=%5.2f°C\n", temp	
Вывод в STDERR (на экран)	STDERR.команда сообщение	STDERR.print(error_msg)	
Чтение строки из STDIN (с клавиатуры)	переменная = gets	mode = gets()	
Чтение всех строк из STDIN	while (переменная = STDIN.gets) do команды end	while (line = STDIN.gets) do puts line end	
Чтение из файла	File.open (файл, 'r').each do переменная команды end	File.open('in.txt', 'r').each do line puts line end	
Запись в файл	File.open (имя_файла, 'w') do file f.print выражение f.puts выражение f.write выражение end	File.open("out.txt", "w") do f 5.times do f.write rand(100) end end	

Учебник по Ruby для начинающих:

/home/pi/Documents/books/Learn_To_Program-Ch.Pine-ru.pdf

<<https://ru.wikipedia.org/wiki/Ruby>>

<<http://www.ruby-lang.org/ru/>>

<<http://ruby-doc.org/>>