

Curso Ruby on Rails - Aulas 1 a 16

Torne-se um programador (Danilo Aparecido)

<https://www.youtube.com/watch?v=Y8Yul1zDnHg&list=PLEdPHGYbHhldWUFs2Q-jSzXAv3NXh4wu0>

Resumo do curso feito por Roberto Pinheiro

1 Introdução

1.rb

```
10.times{|i| puts i.to_s + " - "}
```

```
C:\aulas-ror\1-introducao>ruby 1.rb
0 -
1 -
2 -
3 -
4 -
5 -
6 -
7 -
8 -
9 -
```

2.rb

```
10.times{|i| print i.to_s + " - "}
```

```
C:\aulas-ror\1-introducao>ruby 2.rb
0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 -
```

3.rb

```
10.times{|i| puts "#{i} - "}
```

```
C:\aulas-ror\1-introducao>ruby 3.rb
0 -
1 -
2 -
3 -
4 -
5 -
6 -
7 -
8 -
9 -
```

3 Variáveis e Data Types

1.rb

```
h = {}

print "Stringy string McString!": "
puts "Stringy string McString!".class
print "1 - class: "
puts 1.class
print "1 - class.superclass: "
puts 1.class.superclass
print "1 - class.superclass.superclass: "
puts 1.class.superclass.superclass

print "4.3 - class: "
puts 4.3.class
print "4.3 - class.superclass: "
puts 4.3.class.superclass

print "nil.class: "
puts nil.class
print "{}.class: "
puts h.class
print ":symbol.class: "
puts :symbol.class
print "[].class: "
puts [].class
print "(1..8).class: "
puts (1..8).class
```

```
C:\aulas-ror\3-variaveis-e-data-types>ruby 1.rb
'Stringy string McString!': String
1 - class: Integer
1 - class.superclass: Numeric
1 - class.superclass.superclass: Object
4.3 - class: Float
4.3 - class.superclass: Numeric
nil.class: NilClass
{}.class: Hash
:symbol.class: Symbol
[].class: Array
(1..8).class: Range
```

variável simples:

```
a = 1
```

variável de instância:

```
@a = 1
```

variável de classe:

```
@@a = 1
```

variável global:

```
$a = 1
```

constante:

```
A = 1
```

Usando uma variável simples

2.rb

```
a = 1  
puts a
```

```
C:\aulas-ror\3-variaveis-e-data-types>ruby 2.rb  
1
```

Usando uma variável de instância

3.rb

```
@a = 2  
  
def teste  
  puts @a  
end  
  
teste
```

```
C:\aulas-ror\3-variaveis-e-data-types>ruby 3.rb  
2
```

Usando uma variável de classe

4.rb

```
class teste
  @@a = 3

  def a
    puts @@a
  end
end
```

teste

```
C:\aulas-ror\3-variaveis-e-data-types>ruby 4.rb
3
```

Usando uma variável de instância

5.rb

```
class Teste
  def initialize
    @a = 4
  end

  def a
    puts @a
  end
end
```

Teste.new.a

Usando uma variável global

6.rb

```
$a = 5
```

```
def teste  
  puts $a  
end
```

```
teste
```

```
C:\aulas-ror\3-variaveis-e-data-types>ruby 6.rb  
5
```

Usando uma constante

7.rb

```
A = 6
```

```
def teste  
  puts A  
end
```

```
teste
```

```
C:\aulas-ror\3-variaveis-e-data-types>ruby 7.rb  
6
```

4 Condicionais

1.rb

a = 1

if a==1 then puts "O valor de a é #{a}" end

```
C:\aulas-ror\4-condicionais>ruby 1.rb  
O valor de a é 1
```

2.rb

a = 2

```
if a==1  
  puts "O valor de a é #{a}"  
else  
  puts "O valor de a não é 1"  
end
```

```
C:\aulas-ror\4-condicionais>ruby 2.rb  
O valor de a não é 1
```

3.rb

a = 3

```
if a==1  
  puts "O valor de a é #{a}"  
elsif a==3  
  puts "O valor de a é 3"  
else  
  puts "O valor de a não é 1 e nem 3"  
end
```

```
C:\aulas-ror\4-condicionais>ruby 3.rb  
O valor de a é 3
```

4.rb

a = 2

```
if a==1
  puts "O valor de a é #{a}"
elsif a==3
  puts "O valor de a é 3"
else
  puts "O valor de a não é 1 e nem 3"
end
```

```
C:\aulas-ror\4-condicionais>ruby 4.rb
O valor de a não é 1 e nem 3
```

5.rb

a = 1

```
case
  when a==1
    puts "O valor de a é 1"
  when a==3
    puts "O valor de a é 3"
  else
    puts "O valor de a não é 1 e nem 3"
end
```

```
C:\aulas-ror\4-condicionais>ruby 5.rb
O valor de a é 1
```

6.rb

a = 2

```
unless a==1
  puts "O valor de a é diferente de 1"
end
```

```
C:\aulas-ror\4-condicionais>ruby 6.rb
O valor de a é diferente de 1
```

7.rb

a = 3

puts "O valor de a é #{a}" if a == 3

```
C:\aulas-ror\4-condicionais>ruby 7.rb  
O valor de a é 3
```

Ternário

8.rb

a = 3

b = a == 3 ? 50 : 30

c = a == 1 ? 50 : 30

puts "O valor de b é #{b}"

puts "O valor de c é #{c}"

```
C:\aulas-ror\4-condicionais>ruby 8.rb  
O valor de b é 50  
O valor de c é 30
```


5 Tratamento de strings

gsub

1.rb

```
a = "nossa aula de hoje"  
a = a.gsub("aula", "aula 2")
```

```
puts a
```

```
C:\aulas-ror\5-tratamento_strings>ruby 1.rb  
nossa aula 2 de hoje
```

2.rb

```
a = "nossa aula de hoje"  
a.gsub!("aula", "aula 2")
```

```
puts a
```

```
C:\aulas-ror\5-tratamento_strings>ruby 2.rb  
nossa aula 2 de hoje
```

Concatenação de strings

3.rb

```
b = "tratamento de strings"  
a = "aula de hoje - #{b}"
```

```
puts a
```

```
C:\aulas-ror\5-tratamento_strings>ruby 3.rb  
aula de hoje - tratamento de strings
```

4.rb

```
b = 2  
a = "aula de hoje - " + b.to_s
```

```
puts a
```

```
C:\aulas-ror\5-tratamento_strings>ruby 4.rb  
aula de hoje - 2
```

5.rb

```
b = '--- 2'  
a = 'aula de hoje ' + b
```

```
puts a
```

```
C:\aulas-ror\5-tratamento_strings>ruby 5.rb  
aula de hoje --- 2
```

6.rb

```
b = '--- 2'  
a = 'aula de hoje '
```

```
a << b
```

```
puts a
```

```
C:\aulas-ror\5-tratamento_strings>ruby 6.rb  
aula de hoje --- 2
```

substring

7.rb

```
a = 'nossa aula de hoje '
```

```
puts a[0,5]
```

```
C:\aulas-ror\5-tratamento_strings>ruby 7.rb  
nossa
```

8.rb

```
a = 'nossa aula de hoje '
```

```
puts a.scan(/nossa/)
```

```
C:\aulas-ror\5-tratamento_strings>ruby 8.rb  
nossa
```

split

9.rb

```
a = 'nossa aula de hoje '
```

```
puts a.split(" ")
```

```
C:\aulas-ror\5-tratamento_strings>ruby 9.rb  
nossa  
aula  
de  
hoje
```

upcase e downcase

10.rb

```
a = 'nossa aula de hoje '
```

```
puts a.upcase
```

```
C:\aulas-ror\5-tratamento_strings>ruby 10.rb  
NOSSA AULA DE HOJE
```

11.rb

```
a = 'NOSSA aula de hoje '
```

```
puts a.downcase
```

```
C:\aulas-ror\5-tratamento_strings>ruby 11.rb  
nossa aula de hoje
```

capitalize

12.rb

```
a = 'nossa aula de hoje '
```

```
puts a.capitalize
```

```
C:\aulas-ror\5-tratamento_strings>ruby 12.rb
Nossa aula de hoje
```

delete

13.rb

```
a = 'NOSSA aula de hoje '
```

```
puts a.delete("aula")
```

```
C:\aulas-ror\5-tratamento_strings>ruby 13.rb
NOSSA  de hoje
```

strip

14.rb

```
a = '   NOSSA aula de hoje   '
b = '---'
```

```
puts b + a.strip + b
puts b + a.lstrip + b
puts b + a.rstrip + b
```

```
C:\aulas-ror\5-tratamento_strings>ruby 14.rb
---NOSSA aula de hoje---
---NOSSA aula de hoje    ---
---   NOSSA aula de hoje---
```

include

15.rb

```
a = 'NOSSA aula de hoje'
```

```
puts a.include?("aula")
```

```
C:\aulas-ror\5-tratamento_strings>ruby 15.rb  
true
```

index

16.rb

```
a = 'NOSSA aula de hoje'
```

```
puts a.index("aula")
```

```
C:\aulas-ror\5-tratamento_strings>ruby 16.rb  
6
```

reverse

17.rb

```
a = 'NOSSA aula de hoje'
```

```
puts a.reverse
```

```
C:\aulas-ror\5-tratamento_strings>ruby 17.rb  
ejoh ed alua ASSON
```

6 Operadores lógicos

Operadores de igualdade

1.rb

Verifica se dois números tem o mesmo valor.

```
print "12 == 24/2: "
```

```
puts 12 == 24/2
```

Tal como o exemplo anterior compara os dois valores.

```
print "24.eql?(12*2): "
```

```
puts 24.eql?(12*2)
```

Verifica se dois números tem valor diferente.

```
print "12 != 14: "
```

```
puts 12 != 14
```

Verifica se dois números tem o mesmo valor e retorna 0 no caso afirmativo.

```
print "12 <=> 12: "
```

```
puts 12 <=> 12
```

Verifica se dois números tem o mesmo valor e retorna 1 caso o primeiro valor seja maior que o segundo.

```
print "12 <=> 10: "
```

```
puts 12 <=> 10
```

Verifica se dois números tem o mesmo valor e retorna -1 caso o primeiro valor seja menor que o segundo.

```
print "12 <=> 14: "
```

```
puts 12 <=> 14
```

```
C:\aulas-ror\6-operadores_logicos>ruby 1.rb
12 == 24/2: true
24.eql?(12*2): true
12 != 14: true
12 <=> 12: 0
12 <=> 10: 1
12 <=> 14: -1
```

Operador	Função	É método? (redefinível)
::	Escopo	Não
[]	Referência (Array)	Sim
**	Expoenciação	Sim
-, +, !, ~	-, +, Neg., Compl. (unários)	Sim
*, /, %	Mult., Div., Mód.	Sim
+, -	Adic., Subtr.	Sim
<<, >>	Deslocamento	Sim
&&	"E" binário	Sim
, ^	"Ou", "Ou exclusivo"	Sim
>, >=, <, <=	Comparação	Sim
<=>, ==, ===, !=, ~=, !~	Igualdade	Sim*
&&	"E" lógico	Não
	"Ou" lógico	Não
.., ...	"Faixas" incl. e excl.	Não
?:	if-then-else	Não
=, +=, -=, etc.	Atribuição	Não*
defined?	Testa definição de um símbolo	Não
Not	"Não" lógico	Não
and, or	"E", "Ou" lógicos	Não
if, unless, while, until	Modificadores	Não
begin, end	Expressão em bloco	Não

7 Looping

while

1.rb

```
top = 4
now = 0
while now < top
  puts "while #{now} .. #{top}"
  now += 1
end
```

```
C:\aulas-ror\7-looping>ruby 1.rb
while 0 .. 4
while 1 .. 4
while 2 .. 4
while 3 .. 4
```

2.rb

```
top = 4
now = 1
while now <= top
  puts "while #{now} .. #{top}"
  now += 1
end
```

```
C:\aulas-ror\7-looping>ruby 2.rb
while 1 .. 4
while 2 .. 4
while 3 .. 4
while 4 .. 4
```

3.rb

```
top = 4
now = 0

puts "while #{now+=1} .. #{top}" while now < top
```



```
C:\aulas-ror\7-looping>ruby 3.rb
while 1 .. 4
while 2 .. 4
while 3 .. 4
while 4 .. 4
```

until

4.rb

```
top = 4
now = 0
```

```
puts "while #{now+=1} .. #{top}" until now >= top
```

```
C:\aulas-ror\7-looping>ruby 4.rb
while 1 .. 4
while 2 .. 4
while 3 .. 4
while 4 .. 4
```

looping

5.rb (looping eterno)

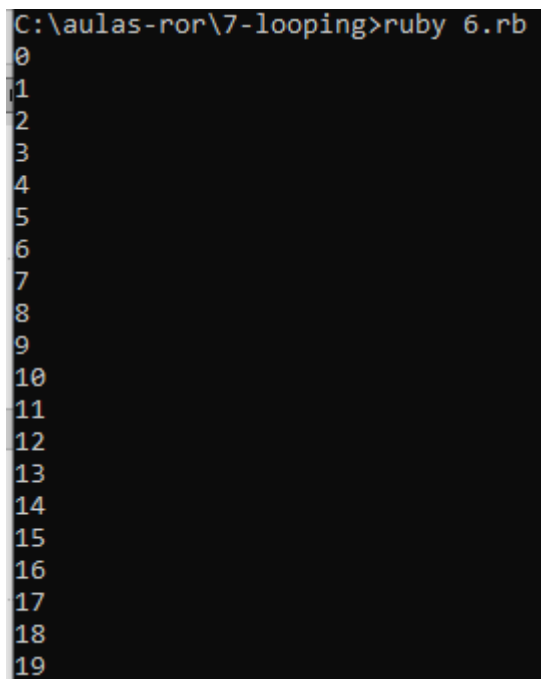
```
loop do
  puts "oi"
end
```

[illegible]

6.rb

```
index = 0
```

```
loop do
  puts index
  index += 1
  break if index >= 20
end
```

A terminal window with a black background and white text. The command 'C:\aulas-ror\7-looping>ruby 6.rb' is entered at the top. Below it, the numbers 0 through 19 are printed on separate lines, representing the output of the loop in 6.rb.

```
C:\aulas-ror\7-looping>ruby 6.rb
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

7.rb

```
index = 0
```

```
loop do
  index += 1
  next if index==5 or index==8
  puts index
  break if index >= 20
end
```

```
C:\aulas-ror\7-looping>ruby 7.rb
1
2
3
4
6
7
9
10
11
12
13
14
15
16
17
18
19
20
```

times

8.rb

```
5.times {|index| puts index}
```

```
C:\aulas-ror\7-looping>ruby 8.rb
0
1
2
3
4
```

9.rb

```
10.times do |index|
  puts index
end
```

```
C:\aulas-ror\7-looping>ruby 9.rb
0
1
2
3
4
5
6
7
8
9
```

10.rb

```
for i in 0..7
  next if i % 2 == 0
  puts i
end
```

```
C:\aulas-ror\7-looping>ruby 10.rb
1
3
5
7
```

each

11.rb

```
(1..7).each{|i| puts i}
```

```
C:\aulas-ror\7-looping>ruby 11.rb
1
2
3
4
5
6
7
```

12.rb

```
[1,2,3,4,5].each{|i| puts i}
```

```
C:\aulas-ror\7-looping>ruby 12.rb
1
2
3
4
5
```

select

13.rb

```
a = [1,2,3,4,5].select{|i| i >= 4}
puts a
```

```
C:\aulas-ror\7-looping>ruby 13.rb
4
5
```

14.rb

```
a = [1,2,3,4,5].select{|i| i >= 4}
puts a.to_s
```

```
C:\aulas-ror\7-looping>ruby 14.rb
[4, 5]
```

map

15.rb

```
a = [1,2,3,4,5].map{|i| "oi - #{i}"}
puts a.to_s
```

```
C:\aulas-ror\7-looping>ruby 15.rb
["oi - 1", "oi - 2", "oi - 3", "oi - 4", "oi - 5"]
```

16.rb

```
a = [1,2,3,4,5].map{|i| "oi - #{i}"}

a.each do |i|
  puts i
end
```

```
C:\aulas-ror\7-looping>ruby 16.rb
oi - 1
oi - 2
oi - 3
oi - 4
oi - 5
```

8 Hash

1.rb

```
a = {nome: "Danilo", telefone: "(75) 3725-8568"}  
b = {:nome=>"Danilo", :telefone=>"(75) 3725-8568"}
```

```
puts a  
puts b
```

```
C:\aulas-ror\8-hash>ruby 1.rb  
{:nome=>"Danilo", :telefone=>"(75) 3725-8568"}  
{:nome=>"Danilo", :telefone=>"(75) 3725-8568"}
```

2.rb

```
a = {nome: "Danilo", telefone: "(75) 3725-8568"}
```

```
puts a[:nome]
```

```
C:\aulas-ror\8-hash>ruby 2.rb  
Danilo
```

3.rb

```
a = {}
```

```
a[:nome] = "Danilo"  
a[:telefone] = "(75) 3725-8568"
```

```
puts a
```

```
C:\aulas-ror\8-hash>ruby 3.rb  
{:nome=>"Danilo", :telefone=>"(75) 3725-8568"}
```

4.rb

```
a = {}
```

```
a["nome"] = "Danilo"  
a["telefone"] = "(75) 3725-8568"
```

```
puts a
```

```
C:\aulas-ror\8-hash>ruby 4.rb  
{"nome"=>"Danilo", "telefone"=>"(75) 3725-8568"}
```

5.rb

```
a = {}  
  
a["nome".to_sym] = "Danilo"  
a["telefone".to_sym] = "(75) 3725-8568"  
  
puts a
```

```
C:\aulas-ror\8-hash>ruby 5.rb  
{:nome=>"Danilo", :telefone=>"(75) 3725-8568"}
```

6.rb

```
loop do  
  puts "Bem-vindo ao programa"  
  puts "Digite 0 para sair ou 1 para continuar"  
  valor = gets.to_i  
  
  break if(valor == 0)  
  
  alunos = []  
  
  3.times do  
    aluno = {}  
  
    puts "Digite o nome do aluno"  
    aluno[:nome] = gets  
  
    puts "Digite o telefone do aluno"  
    aluno[:telefone] = gets  
  
    alunos << aluno  
  end  
  
  alunos.each do |aluno|  
    puts "=====  
    puts "aluno: #{aluno[:nome]} - telefone: #{aluno[:telefone]}"  
  end  
  puts "=====  
end
```

```

C:\aulas-ror\8-hash>ruby 6.rb
Bem-vindo ao programa
Digite 0 para sair ou 1 para continuar
1
Digite o nome do aluno
Danilo
Digite o telefone do aluno
(75) 3725-8568
Digite o nome do aluno
Denilson
Digite o telefone do aluno
(79) 2774-0385
Digite o nome do aluno
Fabiane
Digite o telefone do aluno
(11) 2660-6684
=====
aluno: Danilo
- telefone: (75) 3725-8568
=====
aluno: Denilson
- telefone: (79) 2774-0385
=====
aluno: Fabiane
- telefone: (11) 2660-6684
=====
Bem-vindo ao programa
Digite 0 para sair ou 1 para continuar

```

7.rb

```

loop do
  puts "Bem-vindo ao programa"
  puts "Digite 0 para sair ou 1 para continuar"
  valor = gets.to_i

  break if(valor == 0)

  alunos = []

  3.times do
    aluno = {}

    puts "Digite o nome do aluno"
    aluno[:nome] = gets.delete("\n")

    puts "Digite o telefone do aluno"
    aluno[:telefone] = gets.delete("\n")

    alunos << aluno
  end
end

```



```

alunos.each do |aluno|
  puts "=====
  puts "aluno: #{aluno[:nome]} - telefone: #{aluno[:telefone]}"
end
puts "=====
end

```

7.rb

```

loop do
  puts "Bem-vindo ao programa"
  puts "Digite 0 para sair ou 1 para continuar"
  valor = gets.to_i

  break if(valor == 0)

  alunos = []

  3.times do
    aluno = {}

    puts "Digite o nome do aluno"
    aluno[:nome] = gets.delete("\n")

    puts "Digite o telefone do aluno"
    aluno[:telefone] = gets.delete("\n")

    alunos << aluno
  end

  alunos.each do |aluno|
    puts "=====
    puts "aluno: #{aluno[:nome]} - telefone: #{aluno[:telefone]}"
  end
  puts "=====
end

```

```
C:\aulas-ror\8-hash>ruby 7.rb
Bem-vindo ao programa
Digite 0 para sair ou 1 para continuar
1
Digite o nome do aluno
Danilo
Digite o telefone do aluno
(45) 2705-5727
Digite o nome do aluno
Denilson
Digite o telefone do aluno
(73) 3974-0164
Digite o nome do aluno
Fabiane
Digite o telefone do aluno
(18) 2856-8864
=====
aluno: Danilo - telefone: (45) 2705-5727
=====
aluno: Denilson - telefone: (73) 3974-0164
=====
aluno: Fabiane - telefone: (18) 2856-8864
=====
Bem-vindo ao programa
Digite 0 para sair ou 1 para continuar
```

9 Funções

1.rb

```
def teste(count)
  return 1 if count == 2

  1 + 420 - 3
end
```

```
puts teste 2
puts teste 1
```

```
C:\aulas-ror\9-funcoes>ruby 1.rb
1
418
```

2.rb

```
def teste(count=1)
  return 1 if count == 2

  1 + 420 - 3
end
```

```
puts teste
```

```
C:\aulas-ror\9-funcoes>ruby 2.rb
418
```

3.rb

```
def teste(*parametros)
  return 1 if parametros.include? 2

  1 + 420 - 3
end
```

```
a = teste 1,2,3,4,5,6
```

```
puts a
```

```
C:\aulas-ror\9-funcoes>ruby 3.rb
1
```

4.rb

```
def teste(*parametros)
  a = "aa" + 1
  return 1 if parametros.include? 2
  1 + 420 - 3

rescue Exception => e
  puts "Opa! Aconteceu um erro"
  puts e.message
end

a = teste 1,2,3,4,5,6

puts a
```

```
C:\aulas-ror\9-funcoes>ruby 4.rb
Opa! Aconteceu um erro
no implicit conversion of Integer into String
```

5.rb

```
def captura_aluno
  aluno = {}

  puts "Digite o nome do aluno"
  aluno[:nome] = gets.delete("\n")

  puts "Digite o telefone do aluno"
  aluno[:telefone] = gets.delete("\n")

  aluno
end

def mostrar_alunos(alunos)
  alunos.each do |aluno|
    puts "=====
    puts "aluno: #{aluno[:nome]} - telefone: #{aluno[:telefone]}"
  end
  puts "=====
end

loop do
  puts "Bem-vindo ao programa"
  puts "Digite 0 para sair ou 1 para continuar"
  valor = gets.to_i

  break if(valor == 0)
```

```
alunos = []  
  
3.times do  
  alunos << captura_aluno  
end
```

```
  mostrar_alunos(alunos)
```

```
end
```

```
C:\aulas-ror\9-funcoes>ruby 5.rb  
Bem-vindo ao programa  
Digite 0 para sair ou 1 para continuar  
1  
Digite o nome do aluno  
Danilo  
Digite o telefone do aluno  
111111111  
Digite o nome do aluno  
Denilson  
Digite o telefone do aluno  
222222222  
Digite o nome do aluno  
Fabiane  
Digite o telefone do aluno  
333333333  
=====  
aluno: Danilo - telefone: 111111111  
=====  
aluno: Denilson - telefone: 222222222  
=====  
aluno: Fabiane - telefone: 333333333  
=====  
Bem-vindo ao programa  
Digite 0 para sair ou 1 para continuar
```

10 Funções recursivas

1.rb

@i = 1

```
def captura_servico
  puts "Implementar o dado para buscar da API #{@i}"
  @i += 1
  return if @i > 3
  captura_servico
end
```

captura_servico

```
C:\aulas-ror\10-funcoes_recursivas>ruby 1.rb
Implementar o dado para buscar da API 1
Implementar o dado para buscar da API 2
Implementar o dado para buscar da API 3
```

2.rb

```
def captura_servico(i=1)
  puts "Implementar o dado para buscar da API #{i}"
  return if i > 3
  captura_servico(i+1)
end
```

captura_servico

```
C:\aulas-ror\10-funcoes_recursivas>ruby 2.rb
Implementar o dado para buscar da API 1
Implementar o dado para buscar da API 2
Implementar o dado para buscar da API 3
Implementar o dado para buscar da API 4
```

11 POO - Classes, instâncias e construtores

1.rb

```
class Carro
  # Setter
  def nome=(value)
    @nome = value
  end

  # Getter
  def nome
    @nome
  end

  def mostrar
    puts @nome
  end
end

carro = Carro.new
carro.nome = "Palio"
puts carro.nome
```

```
C:\aulas-ror\11-poo_classes>ruby 1.rb
Palio
```

2.rb

```
class Carro
  # Setter
  def nome=(value)
    @nome = value
  end

  # Getter
  def nome
    @nome
  end

  def mostrar(marca)
    puts "Marca: #{marca} - Modelo: #{self.nome}"
  end
end
```

```
carro = Carro.new
carro.nome = "Palio"
carro.mostrar("Fiat")
```

```
C:\aulas-ror\11-poo_classes>ruby 2.rb
Marca: Fiat - Modelo: Palio
```

3.rb

```
class Carro
  # Setter
  def nome=(value)
    @nome = value
  end

  # Getter
  def nome
    @nome
  end

  def mostrar(marca="Marca Padrão")
    puts "Marca: #{marca} - Modelo: #{self.nome}"
  end
end

carro = Carro.new
carro.nome = "Palio"
carro.mostrar
carro.mostrar("Fiat")
```

```
C:\aulas-ror\11-poo_classes>ruby 3.rb
Marca: Marca Padrão - Modelo: Palio
Marca: Fiat - Modelo: Palio
```

4.rb

```
class Carro

  # Construtor
  def initialize(nome)
    @nome = nome
  end

  # Setter
  def nome=(value)
    @nome = value
  end
```



```

# Getter
def nome
  @nome
end

def mostrar(marca="Marca Padrão")
  puts "Marca: #{marca} - Modelo: #{self.nome}"
end
end

carro = Carro.new("Fiesta")
carro.mostrar("Ford")

```

```

C:\aulas-ror\11-poo_classes>ruby 4.rb
Marca: Ford - Modelo: Fiesta

```

5.rb

```

class Carro

  # Construtor
  def initialize(nome="Modelo Padrão")
    @nome = nome
  end

  # Setter
  def nome=(value)
    @nome = value
  end

  # Getter
  def nome
    @nome
  end

  def mostrar(marca="Marca Padrão")
    puts "Marca: #{marca} - Modelo: #{self.nome}"
  end
end

Carro.new.mostrar

```

```

C:\aulas-ror\11-poo_classes>ruby 5.rb
Marca: Marca Padrão - Modelo: Modelo Padrão

```

12 POO Accessors

1.rb

```
class Carro

  # Construtor
  def initialize(nome="Modelo Padrão")
    @nome = nome
  end

  # Setter
  def nome=(value)
    @nome = value
  end

  # Getter
  def nome
    @nome
  end

  # Setter
  def pneu=(value)
    @pneu = value
  end

  # Getter
  def pneu
    @pneu
  end

  # Setter
  def porta=(value)
    @porta = value
  end

  # Getter
  def porta
    @porta
  end

  # Setter
  def painel=(value)
    @painel = value
  end

  # Getter
  def painel
    @painel
  end
end
```

```

end

# Setter
def roda=(value)
  @roda = value
end

# Getter
def roda
  @roda
end

end

carro = Carro.new

carro.nome = "Palio"
puts 'Nome: ' + carro.nome

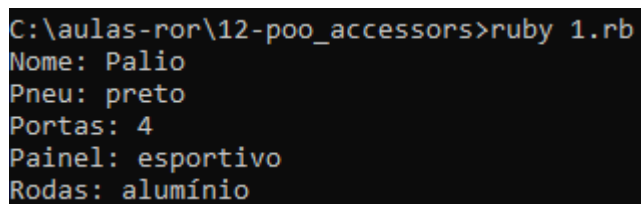
carro.pneu = "preto"
puts 'Pneu: ' + carro.pneu

carro.porta = 4
puts 'Portas: ' + carro.porta.to_s

carro.painel = "esportivo"
puts 'Painel: ' + carro.painel

carro.roda = "alumínio"
puts "Rodas: " + carro.roda

```



```

C:\aulas-ror\12-poo_accessors>ruby 1.rb
Nome: Palio
Pneu: preto
Portas: 4
Painel: esportivo
Rodas: alumínio

```

2.rb

```

class Carro

  # Construtor
  def initialize(nome="Modelo Padrão")
    @nome = nome
  end

  attr_accessor :nome, :pneu, :porta, :painel, :roda

end

```

```
carro = Carro.new

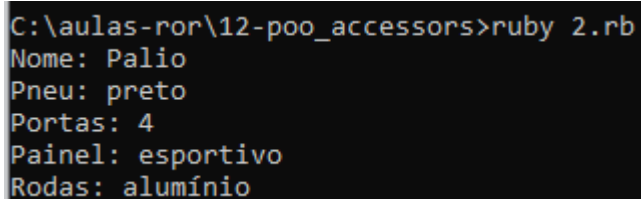
carro.nome = "Palio"
puts 'Nome: ' + carro.nome

carro.pneu = "preto"
puts 'Pneu: ' + carro.pneu

carro.porta = 4
puts 'Portas: ' + carro.porta.to_s

carro.painel = "esportivo"
puts 'Painel: ' + carro.painel

carro.roda = "alumínio"
puts "Rodas: " + carro.roda
```



```
C:\aulas-ror\12-poo_accessors>ruby 2.rb
Nome: Palio
Pneu: preto
Portas: 4
Painel: esportivo
Rodas: alumínio
```

3.rb

```
class Carro

  # Construtor
  def initialize(nome="Modelo Padrão")
    @nome = nome
  end

  attr_accessor :nome, :porta, :painel
  attr_writer :pneu
  attr_reader :roda

end

carro = Carro.new

carro.nome = "Palio"
puts 'Nome: ' + carro.nome

carro.pneu = "preto"
# puts 'Pneu: ' + carro.pneu

carro.porta = 4
puts 'Portas: ' + carro.porta.to_s

carro.painel = "esportivo"
```

```
puts 'Painel: ' + carro.painel

# carro.roda = "alumínio"
if carro.roda == nil
  puts "Rodas: Não definido"
else
  puts "Rodas: " + carro.roda
end
```

```
C:\aulas-ror\12-poo_accessors>ruby 3.rb
Nome: Palio
Portas: 4
Painel: esportivo
Rodas: Não definido
```

13 POO Herança

1.rb

```
require_relative 'carro'
require_relative 'fiesta'
require_relative 'golf'

carro = Carro.new
fiesta = Fiesta.new
golf = Golf.new

fiesta.cor = "branco"
fiesta.roda = "alumínio"
golf.painel = "esportivo"

puts 'Fiesta - Cor: ' + fiesta.cor
puts 'Fiesta - Rodas: ' + fiesta.roda
puts 'Golf - Painel: ' + golf.painel

Carro.new.mostrar
```

carro.rb

```
class Carro

  # Construtor
  def initialize(nome="Modelo Padrão")
    @nome = nome
  end

  attr_accessor :nome, :porta, :painel, :roda
  attr_reader :pneu

  def mostrar(marca="Marca Padrão")
    puts "Marca: #{marca} - Modelo: #{self.nome}"
  end

end
```

fiesta.rb

```
class Fiesta < Carro
  attr_accessor :cor
end
```

golf.rb

```
class Golf < Carro
```

```
end
```

```
C:\aulas-ror\13-poo_heranca>ruby 1.rb  
Fiesta - Cor: branco  
Fiesta - Rodas: alumínio  
Golf - Painel: esportivo  
Marca: Marca Padrão - Modelo: Modelo Padrão
```

14 POO - Public, Private e Protected

1.rb

```
require_relative 'carro'
require_relative 'fiesta'
require_relative 'golf'

carro = Carro.new
fiesta = Fiesta.new
golf = Golf.new

fiesta.cor = "branco"
fiesta.roda = "alumínio"
golf.painel = "esportivo"

puts 'Fiesta - Cor: ' + fiesta.cor
puts 'Fiesta - Rodas: ' + fiesta.roda
puts 'Golf - Painel: ' + golf.painel

Carro.new.mostrar

puts fiesta.mostrar_protected
puts fiesta.mostrar_protected_new

puts fiesta.mostrar_private
puts fiesta.mostrar_private_new
```

Carro.rb

```
class Carro

  # Construtor
  def initialize(nome="Modelo Padrão")
    @nome = nome
  end

  attr_accessor :nome, :porta, :painel, :roda
  attr_reader :pneu

  def mostrar(marca="Marca Padrão")
    puts "Marca: #{marca} - Modelo: #{self.nome} - #{algo_mais_private}"
  end

  private
```



```

def algo_mais_private
  "Este é um método privado para retornar algo a mais."
end

protected

def algo_mais_protected
  "Este é um método protegido para retornar algo a mais"
end

public

def algo_mais_public
  "Este é um método público para retornar algo a mais"
end

```

end

fiesta.rb

```

class Fiesta < Carro
  attr_accessor :cor

  def mostrar_protected
    algo_mais_protected
  end

  def mostrar_protected_new
    Carro.new.algo_mais_protected
  end

  def mostrar_private
    algo_mais_private
  end

  def mostrar_private_new
    Carro.new.algo_mais_private
  end

```

end

```

C:\aulas-ror\14-poo_metodos_acesso>ruby 1.rb
Fiesta - Cor: branco
Fiesta - Rodas: alumínio
Golf - Painel: esportivo
Marca: Marca Padrão - Modelo: Modelo Padrão - Este é um método privado para retornar algo a mais.
Este é um método protegido para retornar algo a mais
Este é um método protegido para retornar algo a mais
Este é um método privado para retornar algo a mais.
Traceback (most recent call last):
  1: from 1.rb:25:in `<main>'
C:/aulas-ror/14-poo_metodos_acesso/fiesta.rb:17:in `mostrar_private_new': private method `algo_mais_private' called for
#<Carro:0x0000000002e549d8 @nome="Modelo Padrão"> (NoMethodError)
Did you mean?  algo_mais_protected

```

15 POO - Polimorfismo e Overload

O Ruby não tem overload.

1.rb

```
require_relative 'carro'  
require_relative 'fiesta'  
require_relative 'golf'
```

```
carro = Carro.new  
fiesta = Fiesta.new  
golf = Golf.new
```

```
fiesta.cor = "branco"  
fiesta.roda = "alumínio"  
golf.painel = "esportivo"
```

```
puts 'Fiesta - Cor: ' + fiesta.cor  
puts 'Fiesta - Rodas: ' + fiesta.roda  
puts 'Golf - Painel: ' + golf.painel
```

```
puts golf.mostrar
```

carro.rb

```
class Carro
```

```
  # Construtor  
  def initialize(nome="Modelo Padrão")  
    @nome = nome  
  end
```

```
  attr_accessor :nome, :porta, :painel, :roda  
  attr_reader :pneu
```

```
  def mostrar(marca="Marca Padrão")  
    puts "Marca: #{marca} - Modelo: #{self.nome} - #{algo_mais_private}"  
  end
```

```
  private
```

```
  def algo_mais_private  
    "Este é um método privado para retornar algo a mais."  
  end
```


protected

```
def algo_mais_protected
  "Este é um método protegido para retornar algo a mais"
end
```

public

```
def algo_mais_public
  "Este é um método público para retornar algo a mais"
end
```

end

golf.rb

```
class Golf < Carro
```

end

```
C:\aulas-ror\15-poo_polimorfismo_overload>ruby 1.rb
Fiesta - Cor: branco
Fiesta - Rodas: alumínio
Golf - Painel: esportivo
Marca: Marca Padrão - Modelo: Modelo Padrão - Este é um método privado para retornar algo a mais.
```

golf.rb

```
class Golf < Carro
```

```
  def mostrar
    puts "Este é o método mostrar da classe golf."
  end
```

end

```
C:\aulas-ror\15-poo_polimorfismo_overload>ruby 1.rb
Fiesta - Cor: branco
Fiesta - Rodas: alumínio
Golf - Painel: esportivo
Este é o método mostrar da classe golf.
```

golf.rb

```
class Golf < Carro
```

```
  def mostrar
    mostrar_original = super
    puts "Este é o método mostrar da classe golf. - #{mostrar_original}"
  end
```

end

```
C:\aulas-ror\15-poo_polimorfismo_overload>ruby 1.rb
Fiesta - Cor: branco
Fiesta - Rodas: alumínio
Golf - Painei: esportivo
Marca: Marca Padrão - Modelo: Modelo Padrão - Este é um método privado para retornar algo a mais.
Este é o método mostrar da classe golf. -
```

1.rb

```
require_relative 'carro'
require_relative 'fiesta'
require_relative 'golf'

carro = Carro.new
fiesta = Fiesta.new
golf = Golf.new

fiesta.cor = "branco"
fiesta.roda = "alumínio"
golf.painel = "esportivo"

puts 'Fiesta - Cor: ' + fiesta.cor
puts 'Fiesta - Rodas: ' + fiesta.roda
puts 'Golf - Painei: ' + golf.painel

# Carro.new.mostrar

puts golf.mostrar
puts golf.andar
puts golf.andar 1,2,3,4,5,6,7,8,9,10
```

golf.rb

```
class Golf < Carro

  def mostrar
    mostrar_original = super
    puts "Este é o método mostrar da classe golf. - #{mostrar_original}"
  end

  def andar(*parametros)
    puts "Andar com parâmetros opcionais #{parametros}"

    parametros.each do |param|
      puts param
    end
    puts ""
  end

end
```

```
C:\aulas-ror\15-poo_polimorfismo_overload>ruby 1.rb
Fiesta - Cor: branco
Fiesta - Rodas: alumínio
Golf - Painei: esportivo
Marca: Marca Padrão - Modelo: Modelo Padrão - Este é um método privado para retornar algo a mais.
Este é o método mostrar da classe golf. -

Andar com parâmetros opcionais []

Andar com parâmetros opcionais [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
1
2
3
4
5
6
7
8
9
10
```

16 POO - Interface, Abstração e Singleton

Interface e abstração

Interface é uma classe que não possui implementação.

Não é possível criar uma instância de uma interface ou de uma abstração.

aula.rb

```
require_relative 'interface'  
require_relative 'klass'
```

```
Interface.new
```

interface.rb

```
class Interface  
  def initialize  
    raise "Classe não pode ser instanciada, somente herdada e implementada interface"  
  end  
  def teste1  
    raise "Método para ser implementado"  
  end  
  def teste2  
    raise "Método para ser implementado"  
  end  
end
```

klass.rb

```
class Klass < Interface  
  
end
```

```
C:\aulas-ror\16-poo_interface_abstracao_singleton>ruby aula.rb  
Traceback (most recent call last):  
  2: from aula.rb:7:in `'  
  1: from aula.rb:7:in `new'  
C:/aulas-ror/16-poo_interface_abstracao_singleton/interface.rb:3:in `initialize': Classe não pode ser instanciada, somente herdada e implementada interface (RuntimeError)
```

klass.rb

```
class Klass < Interface  
  def initialize  
  end
```

end

aula.rb

```
require_relative 'interface'  
require_relative 'klass'
```

Klass.new.test1

```
C:\aulas-ror\16-poo_interface_abstracao_singleton>ruby aula.rb  
Traceback (most recent call last):  
  1: from aula.rb:8:in `<main>'  
C:/aulas-ror/16-poo_interface_abstracao_singleton/interface.rb:6:in `test1': Método para ser implementado (RuntimeError)
```

klass.rb

```
class Klass < Interface  
  
  def initialize  
    end  
  
  def test1  
    puts "test1 implementado."  
  end  
  
  def test2  
    puts "test2 implementado."  
  end  
  
end
```

aula.rb

```
require_relative 'interface'  
require_relative 'klass'
```

```
k1 = Klass.new  
k1.test1  
k2 = Klass.new  
k2.test2
```

```
C:\aulas-ror\16-poo_interface_abstracao_singleton>ruby aula.rb  
test1 implementado.  
test2 implementado.
```


abstrata.rb

```
class Abstrata
  def initialize
    raise "Classe não pode ser instanciada, somente herdada e implementada interface"
  end
  def test1
    raise "test1 não implementado"
  end
  def test2
    "Este é um teste 2"
  end
end
```

klass2.rb

```
class Klass2 < Abstrata

  def initialize
  end

end
```

aula.rb

```
require_relative 'interface'
require_relative 'abstrata'
require_relative 'klass2'
```

Abstrata.new

```
C:\aulas-ror\16-poo_interface_abstracao_singleton>ruby aula.rb
Traceback (most recent call last):
  2: from aula.rb:16:in `<main>'
  1: from aula.rb:16:in `new'
C:/aulas-ror/16-poo_interface_abstracao_singleton/abstrata.rb:3:in `initialize': Classe não pode ser instanciada, somente herdada e implementada interface (RuntimeError)
```

aula.rb

```
require_relative 'interface'
require_relative 'abstrata'
require_relative 'klass2'
```

```
k2 = Klass2.new
k2.test2
```

```
C:\aulas-ror\16-poo_interface_abstracao_singleton>ruby aula.rb
test2 implementado.
```

aula.rb

```
require_relative 'interface'
require_relative 'abstrata'
require_relative 'klass2'
```

```
k1 = Klass2.new
k1.test1
```

```
C:\aulas-ror\16-poo_interface_abstracao_singleton>ruby aula.rb
Traceback (most recent call last):
  1: from aula.rb:12:in `
```

Classe Singleton

A classe singleton possui uma instância única.

aula2.rb

```
require_relative 'instancia_unica'

instancia = InstanciaUnica.instance

InstanciaUnica.new
```

instancia_unica.rb

```
require 'singleton'

class InstanciaUnica
  include Singleton

  def test1
    "Este é um test 1"
  end

  def test2
    "Este é um test 2"
  end
end
```

```
C:\aulas-ror\16-poo_interface_abstracao_singleton>ruby aula2.rb
Traceback (most recent call last):
aula2.rb:5:in `<main>': private method `new' called for InstanciaUnica:Class (NoMethodError)
```

aula2.rb

```
require_relative 'instancia_unica'
```

```
instancia = InstanciaUnica.instance
```

```
iu = InstanciaUnica.instance
```

```
puts iu.test1
```

```
puts iu.test2
```

```
C:\aulas-ror\16-poo_interface_abstracao_singleton>ruby aula2.rb
Este é um test 1
Este é um test 2
```