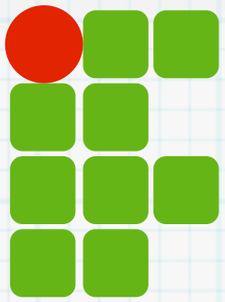


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Programação de Computadores

Escopo de variáveis

O que veremos hoje?

* Introdução
* Escopo de variáveis
* Escopo local
* Escopo global
* Escopo de parâmetros

Escopo

* É a faixa de código na qual cada variável existe
* É usado para definir visibilidade de variáveis

```
n = gets.to_i
m = gets.to_i
soma = 0
...
x = soma*2
...
puts soma
```

Escopo

* É a faixa de código na qual cada variável existe
* É usado para definir visibilidade de variáveis

A variável soma existe até o final do escopo em que é declarada

```
n = gets.to_i
m = gets.to_i
soma = 0
...
x = soma*2
...
puts soma
```

Tipos de escopo

* Local
  * A variável existe apenas na função/método em que foi declarada
* Global
  * A variável é visível em todo o programa
* Existem outros...
  * Veremos mais adiante

```ruby
var = "Successo"
class Teste
  # quero imprimir var aqui ...

  def metodo
    # ..e aqui
  end

end
```

Escopo local

* Uma função possui um escopo próprio
  * Variáveis locais, declaradas na função, são visíveis apenas na função
  * Em Ruby, variáveis declaradas fora da função não são visíveis na função

```
x = gets.to_i
y = gets.to_i

def dif_maior_menor(a,b)
  if (a>b) then
    c = a-b
  else
    c = b-a
  end
end

dif_maior_menor(x,y)
puts c
```

Escopo local

- Uma função possui um escopo próprio
  - Variáveis locais, declaradas na função, são visíveis apenas na função
  - Em Ruby, variáveis declaradas fora da função não são visíveis na função

A variável c deixa de existir no final do escopo d da função

```
x = gets.to_i
y = gets.to_i

def dif_maior_menor(a,b)
  if (a>b) then
     c = a-b
  else
     c = b-a
  end
end

dif_maior_menor(x,y)
puts c
```

Escopo local

* Uma função possui um escopo próprio
  * Variáveis locais, declaradas na função, são visíveis apenas na função
  * Em Ruby, variáveis declaradas fora da função não são visíveis na função

A variável c deixa de existir no final do escopo d da função

```
x = gets.to_i
y = gets.to_i

def dif_maior_menor(a,b)
  if (a>b) then
     c = a-b
  else
     c = b-a
  end
end

dif_maior_menor(x,y)
puts c
```

```
ruby — jorgiano@jorgianoIFRN: ~ — bash — 51×5
10
20
teste_escopo_02.rb:14:in `<main>': undefined local
variable or method `c' for main:Object (NameError)
cnat102982:ruby jorgiano$ _
```

Escopo local

O que é impresso pelo programa?

```
c = 0
def dif_maior_menor(a,b)
  if (a>b) then
    c = a-b
  else
    c = b-a
  end
end
x = gets.to_i
y = gets.to_i
dif_maior_menor(x,y)
puts c
```

Escopo local

O que é impresso pelo programa?

```
ruby — jorgiano@jorgianoIFRN: ~ — bash — 51×5
cnat102982:ruby jorgiano$ ruby teste_escopo_03.rb
10
20
0
cnat102982:ruby jorgiano$ _
```

```
c = 0
def dif_maior_menor(a,b)
   if (a>b) then
      c = a-b
   else
      c = b-a
   end
end
x = gets.to_i
y = gets.to_i
dif_maior_menor(x,y)
puts c
```

Apesar de possuir o mesmo nome, a variável 'c' declarada no programa está em escopo diferente da variável 'c' da função

Escopo local

O que é impresso pelo programa?

```
c = 0
def dif_maior_menor(a,b)
   if (a>b) then
      c = a-b
   else
      c = b-a
   end
end
x = gets.to_i
y = gets.to_i
dif_maior_menor(x,y)
puts c
```

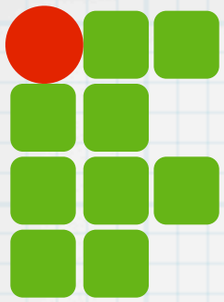
```
ruby — jorgiano@jorgianoIFRN: ~ — bash — 51×5
cnat102982:ruby jorgiano$ ruby teste_escopo_03.rb
10
20
0
cnat102982:ruby jorgiano$ _
```

Apesar de possuir o mesmo nome, a variável 'c' declarada no programa está em escopo diferente da variável 'c' da função

Escopo global

* Uma variável global tem seu escopo definido para todo o programa
* É visível em qualquer parte do programa
* Em Ruby uma variável é definida como global quando usamos o prefixo $

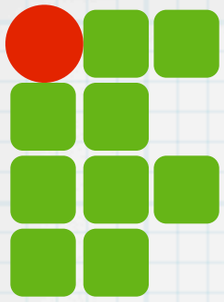
```
$soma = 0
```



Escopo global

```
def dif_maior_menor(a,b)
  if (a>b) then
    $c = a-b
  else
    $c = b-a
  end
end

x = gets.to_i
y = gets.to_i
dif_maior_menor(x,y)
puts $c
```

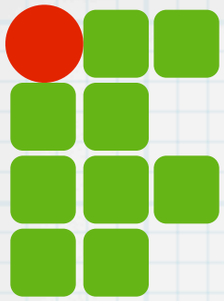


Escopo global

Mesmo que a variável $c tenha sido definida em um método/função, ela continua existindo durante toda a execução do programa

```
def dif_maior_menor(a,b)
  if (a>b) then
    $c = a-b
  else
    $c = b-a
  end
end

x = gets.to_i
y = gets.to_i
dif_maior_menor(x,y)
puts $c
```



Escopo global

* Existe durante todo o programa e em todas as partes do programa

```
def acumula(x)
  $total = $total + x
end

$total = 10
acumula(20)
$total = $total + 10
puts $total
```

Escopo global

* Existe durante todo o programa e em todas as partes do programa

É importante observar o fluxo de processamento do programa

```
def acumula(x)
  $total = $total + x
end

$total = 10
acumula(20)
$total = $total + 10
puts $total
```

Parâmetros

* Os parâmetros possuem escopo local
  * Existem durante a execução do método
  * O valor é atribuído ao parâmetro no momento da chamada

```
def dif_maior_menor(a,b)
  if (a>b) then
    c = a-b
  else
    c = b-a
  end
  return c
end
a = gets.to_i
b = gets.to_i
puts dif_maior_menor(a,b)
```

Parâmetros

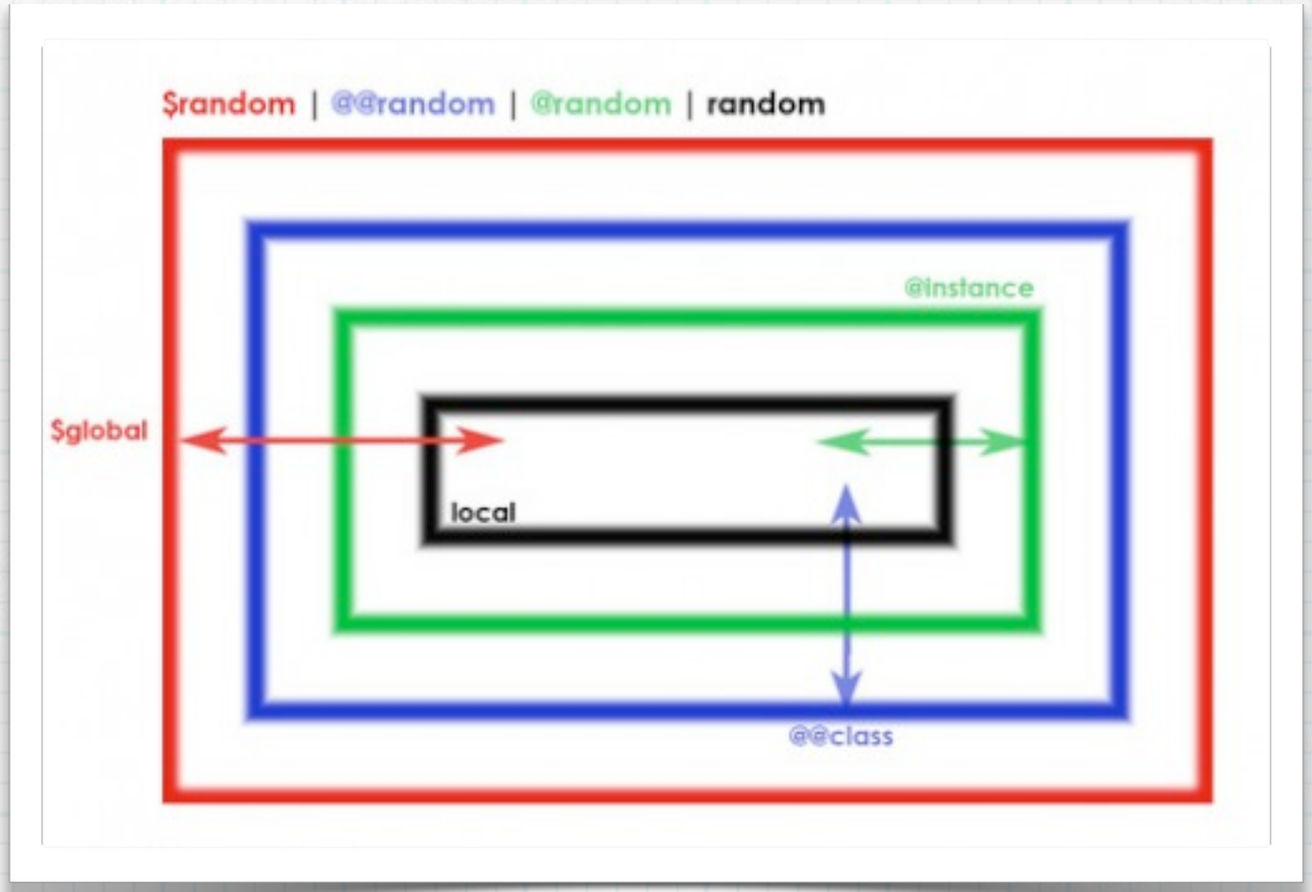
* Os parâmetros possuem escopo local
  * Existem durante a execução do método
  * O valor é atribuído ao parâmetro no momento da chamada

```
def dif_maior_menor(a,b)
  if (a>b) then
    c = a-b
  else
    c = b-a
  end
  return c
end
a = gets.to_i
b = gets.to_i
puts dif_maior_menor(a,b)
```

Não são a mesma variável. Apesar do mesmo nome, estão em escopos diferentes

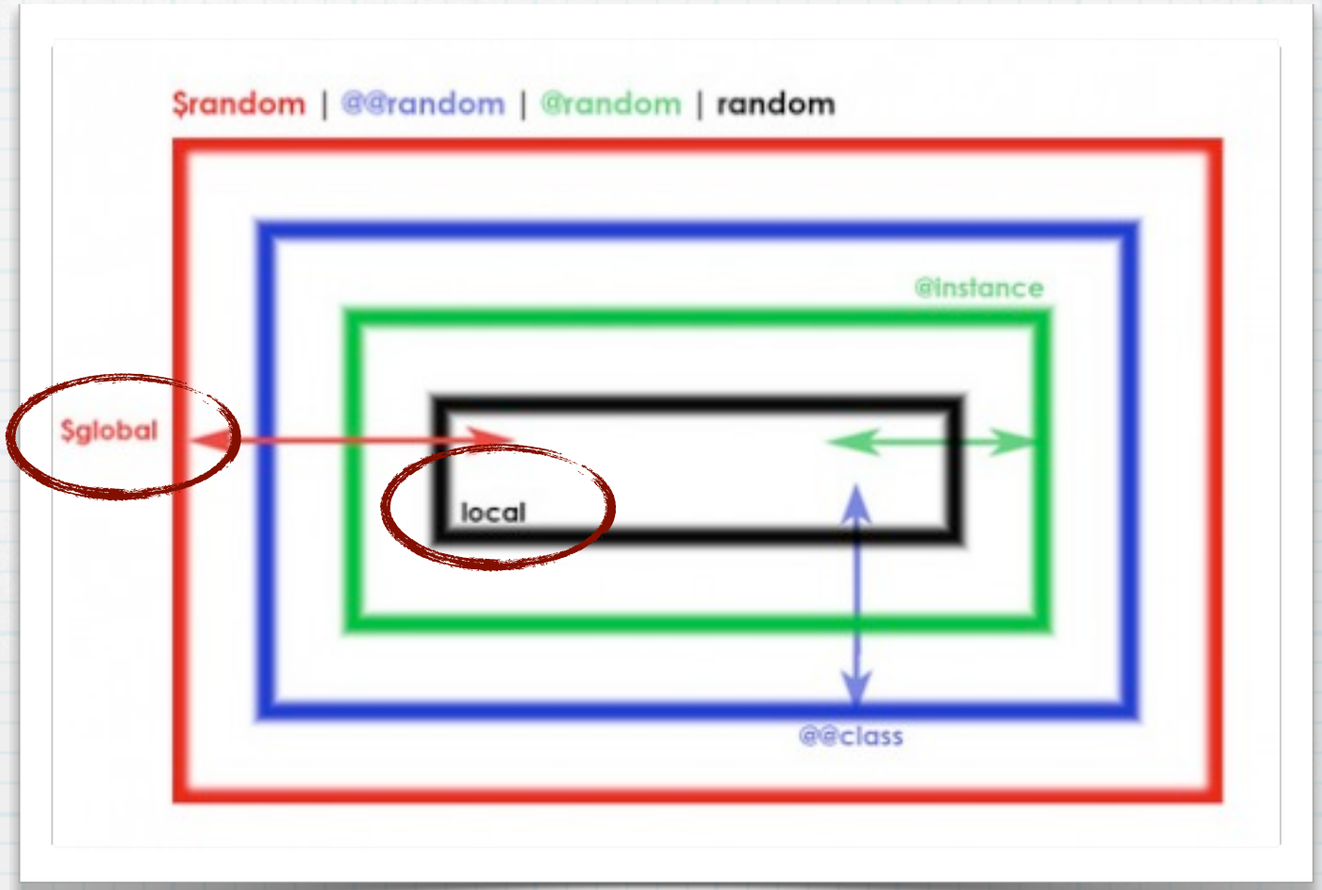
Observações

* NORMALMENTE é preferível usar variáveis locais
  * Fácil gerenciamento de modificações no valor da variável
* É importante conhecer o escopo de cada variável para entender o comportamento do programa



Observações

* NORMALMENTE é preferível usar variáveis locais
  * Fácil gerenciamento de modificações no valor da variável
* É importante conhecer o escopo de cada variável para entender o comportamento do programa



Dúvidas?