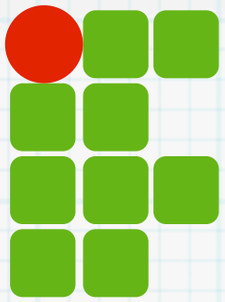
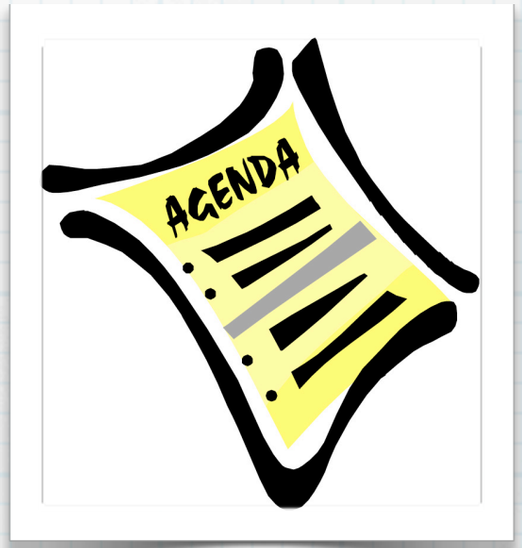


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Programação de Computadores

Iniciando em coleções: arrays (vetores)

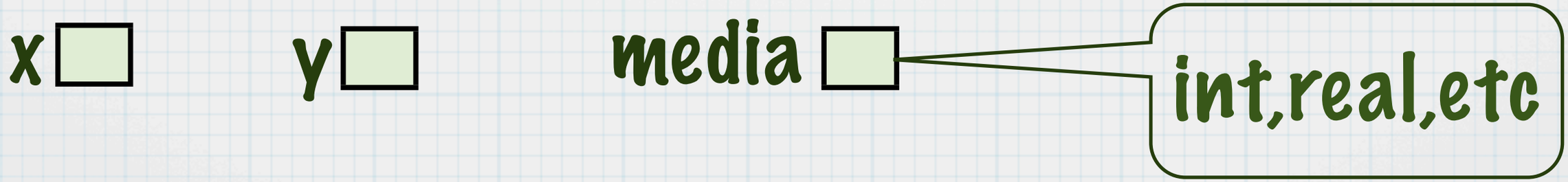
O que veremos hoje?



* Arrays
* Criação
* Acesso pelo índice
* Métodos
  * compact, size
* Mostrar elementos
* Ler elementos
* Mais métodos
* Exercícios

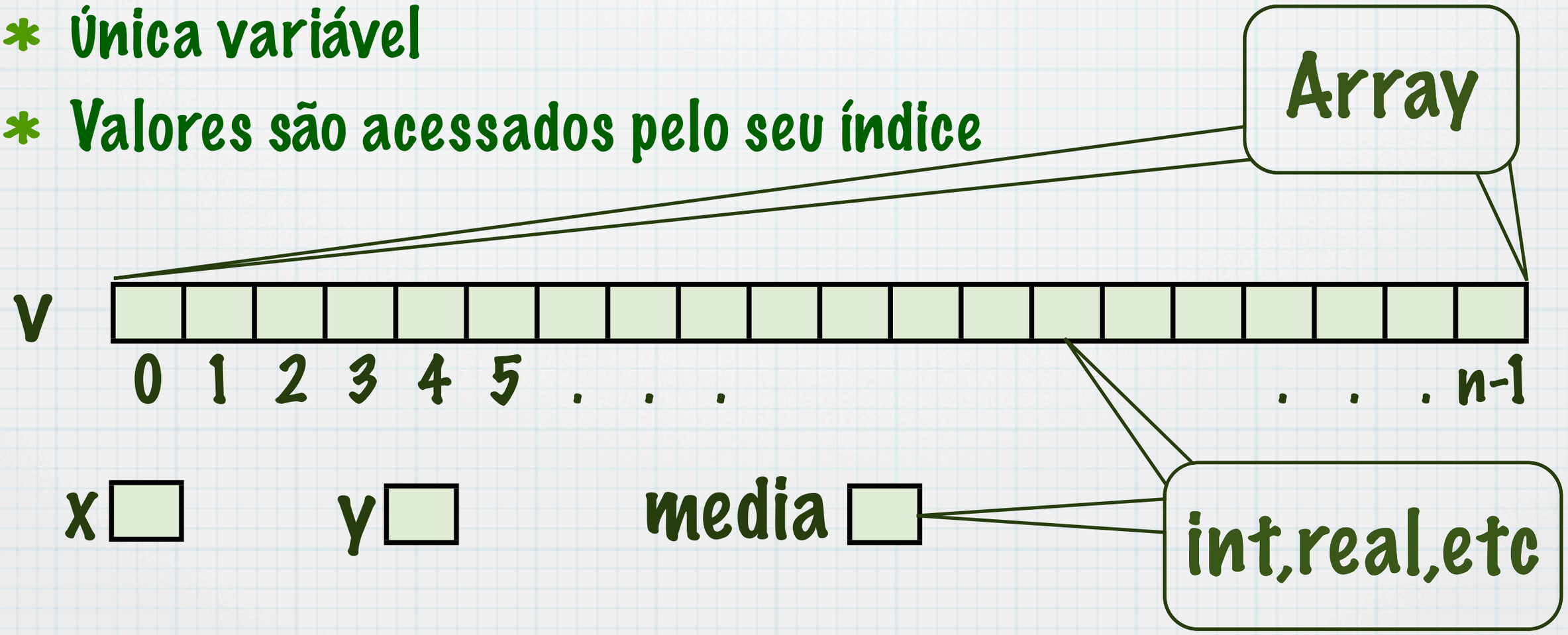
Introdução

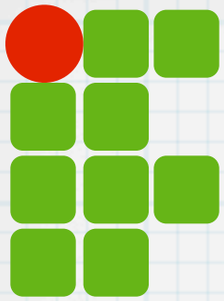
* O que são arrays?
  * Um agregado de elementos
  * Capacidade de armazenar uma coleção de valores
  * Única variável
  * Valores são acessados pelo seu índice



Introdução

* O que são arrays?
  * Um agregado de elementos
  * Capacidade de armazenar uma coleção de valores
  * Única variável
  * Valores são acessados pelo seu índice

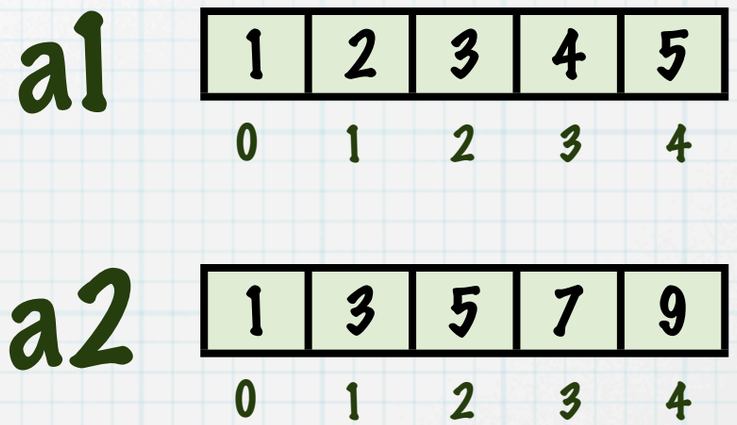




Arrays

* Criação de arrays
  * Elementos separados por vírgulas
  * Entre colchetes ('[' e ']')

```
a1 = [1,2,3,4,5]
a2 = [1,3,5,7,9]
```



* Criar array vazio

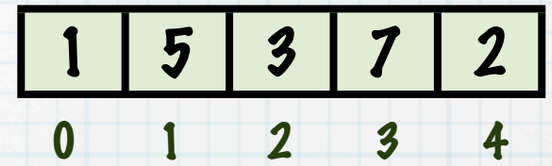
```
a3 = []
```

a3 []

Arrays

* Acesso aos elementos
  * nomeVar[indice]

numeros




```
numeros = [1,5,3,7,2]
puts numeros[0]
puts numeros[4]
```

O que este programa mostra?

Arrays

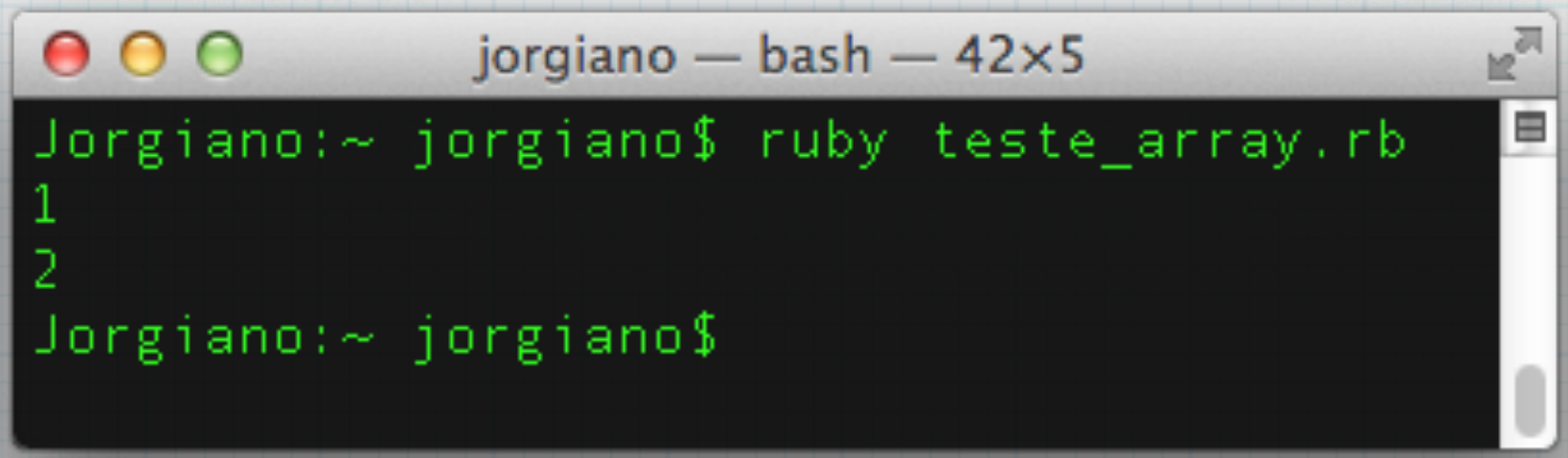
* Acesso aos elementos
  * nomeVar[indice]

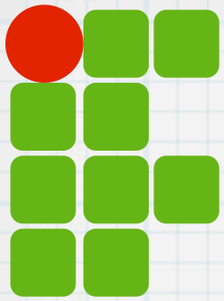
numeros

1	5	3	7	2
0	1	2	3	4

```
numeros = [1,5,3,7,2]
puts numeros[0]
puts numeros[4]
```

O que este programa mostra?





Arrays

* Ler array
  * Um elemento a cada vez

O array deve existir antes de usar algum índice

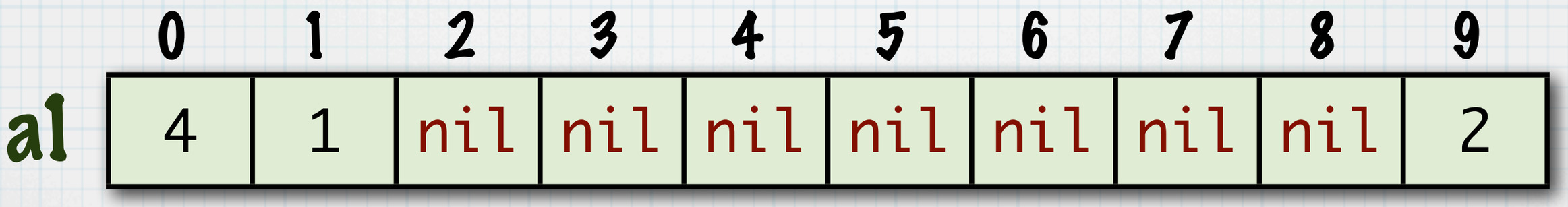
```
a1 = []
a1[0] = gets.to_i
a1[1] = gets.to_i
a1[2] = gets.to_i
...
```

Arrays

* Elementos nulos

```
a1 = []
a1[0] = 4
a1[1] = 1
a1[9] = 2
```

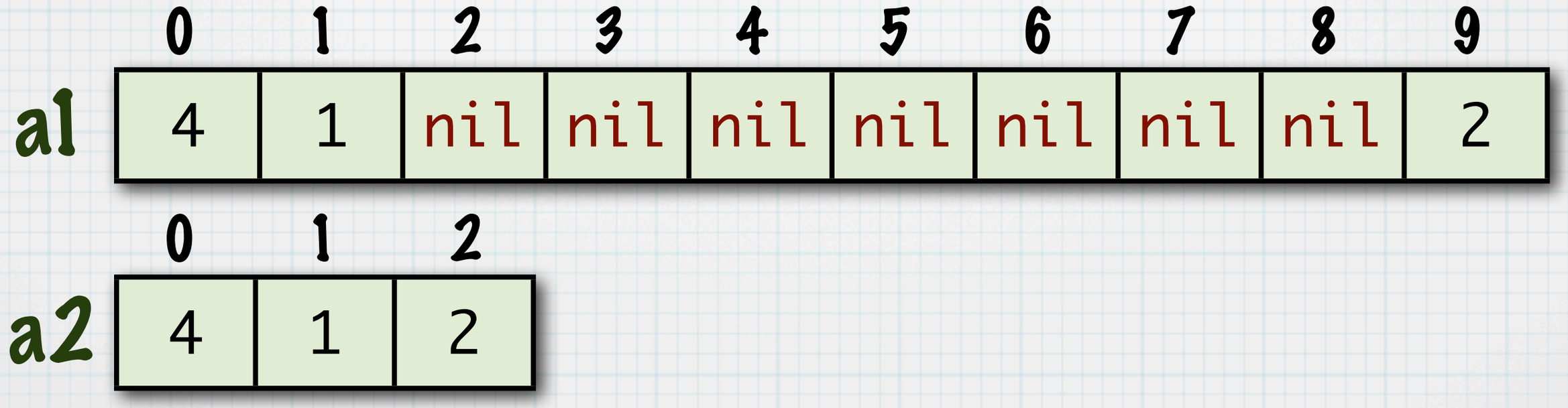
Elementos não atribuídos recebem o valor `nil` (nulo)

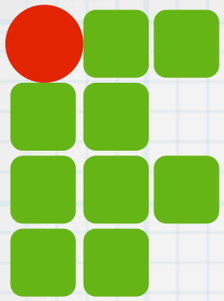


Arrays

* O método compact
  * Retorna um novo array, Eliminando os elementos nulos, mantendo todos os não-nulos no início do array

```
a2 = a1.compact
```





Tamanho

* método size
  * Informa quantidade de índices usados

```
tamanho_a1=a1.size
```

10

	0	1	2	3	4	5	6	7	8	9
a1	4	1	nil	nil	nil	nil	nil	nil	nil	2

	0	1	2
a2	4	1	2

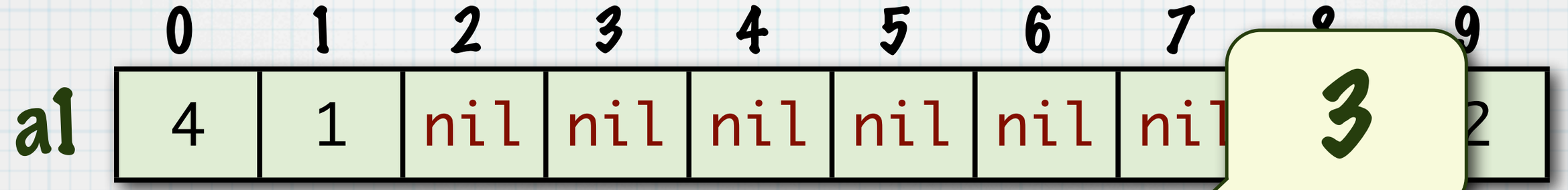
```
tamanho_a2=a2.size
```

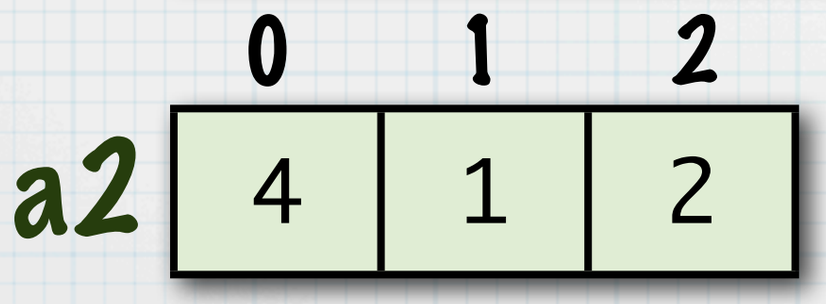
Tamanho

* método size
  * Informa quantidade de índices usados

```
tamanho_a1=a1.size
```

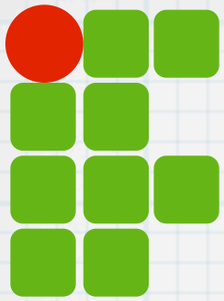
10





3

```
tamanho_a2=a2.size
```



Exemplos

* Ler 5 nomes e mostra-los na ordem em que foram lidos
  * Muito repetitivo
  * Propenso a erros
  * E se forem 10.000 nomes?

```
a1 = []

a1[0] = gets.chomp
a1[1] = gets.chomp
a1[2] = gets.chomp
a1[3] = gets.chomp
a1[4] = gets.chomp

puts a1[0]
puts a1[1]
puts a1[2]
puts a1[3]
puts a1[4]
```

Exemplos

* Ler 5 nomes e mostra-los na ordem em que foram lidos
  * Muito repetitivo
  * Propenso a erros
  * E se forem 10.000 nomes?

```
a1 = []

a1[0] = gets.chomp
a1[1] = gets.chomp
a1[2] = gets.chomp
a1[3] = gets.chomp
a1[4] = gets.chomp

puts a1[0]
puts a1[1]
puts a1[2]
puts a1[3]
puts a1[4]
```

Mesmo código, muda apenas índice do array

Arrays

* Mostrar todos os elementos

```
puts a1[0]
puts a1[1]
puts a1[2]
puts a1[3]
puts a1[4]
```

Mesmo que

```
puts a1
```

Arrays

* Mostrar todos os elementos

```
puts a1[0]
puts a1[1]
puts a1[2]
puts a1[3]
puts a1[4]
```

Mesmo que →

```
puts a1
```

E com o print?

Arrays

* Ler os elementos

Mesmo que

```
a1[0] = gets.chomp
a1[1] = gets.chomp
a1[2] = gets.chomp
a1[3] = gets.chomp
a1[4] = gets.chomp
```

```
a1 = 5.times.map do gets.chomp end
```

Arrays

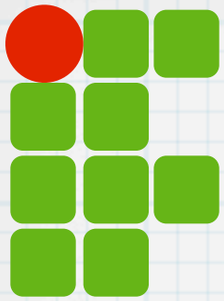
* Ler os elementos

Mesmo que

```
a1[0] = gets.chomp
a1[1] = gets.chomp
a1[2] = gets.chomp
a1[3] = gets.chomp
a1[4] = gets.chomp
```

```
a1 = 5.times.map do gets.chomp end
```

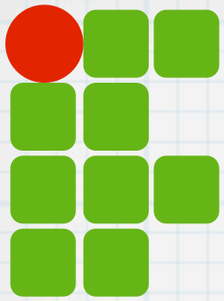
Futuramente detalharemos esta operação



Exemplo

* Ler e mostrar 100 números na ordem e que foram lidos

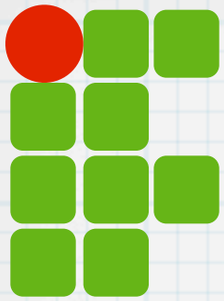
```
numeros = 100.times.map do gets.to_i end

puts numeros
```



Exemplo

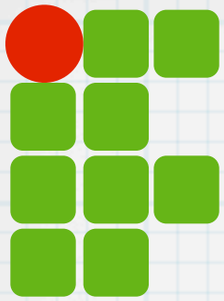
* Ler 10 números e somar os dos índices pares

```
n = 10.times.map do gets.to_i end
soma = n[0]+n[2]+n[4]+n[6]+n[8]
puts soma
```



Métodos

* empty: Informa se array está vazio (true or false)
* size (length): Retorna o tamanho do array
* first: Retorna o primeiro elemento
* last: Retorna o último elemento
* drop (n): Remove os n primeiros elementos
* reverse: Inverte a ordem dos elementos
* compact: Elimina os índices nulos
* sort: Ordena os elementos do array



Exemplo

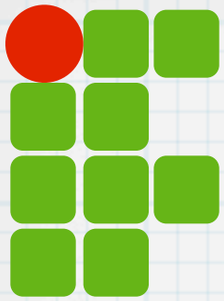
* Ler um array de 100 elementos e inverter se o último elemento for menor que o primeiro

```
n = 100.times.map do gets.to_i end
if (n.first > n.last) then
  n = n.reverse
end
puts n
```

Exemplo

* Ler 1000 elementos inteiros, ordenar e mostrar

```
n = 1000.times.map do gets.to_i end
n = n.sort
puts n
```



Exemplo

* Ler 20 nomes e mostra-los na ordem invertida em que foram lidos

```
nomes = 20.times.map do gets.chomp end
nomes = nomes.reverse
puts nomes
```

Exemplo

* Ler 1.000 notas, ordenar e mostrar a primeira, a do meio e a última

```
notas = 1000.times.map do gets.to_f end
notas = notas.sort
indice_meio = notas.size/2
puts notas[0]
puts notas[indice_meio]
puts notas[999]
```

Exemplo

* Ler um array de 100 elementos inteiros e mostrar o conteúdo do índice indicado pelo primeiro elemento do array

```
n = 100.times.map do gets.to_i end
indice = n[0]
puts n[indice]
```

Exemplo

* Ler um array de 100 elementos inteiros e mostrar o conteúdo do índice indicado pelo primeiro elemento do array

```
n = 100.times.map do gets.to_i end
indice = n[0]
puts n[indice]
```

```
n = 100.times.map do gets.to_i end
puts n[n[0]]
```

Exemplo

* Ler um array de 1000 elementos, um número inteiro e mostrar o conteúdo do índice indicado pelo número lido

```
n = 100.times.map do gets.to_i end
indice = gets.to_i
puts n[indice]
```

Exemplo

* Ler 'x' números e mostrar na ordem em que foram lidos

```
quantidade_de_numeros = gets.to_i
numeros = quantidade_de_numeros.times.map do gets.to_i end
puts numeros
```

Dúvidas?