Introdução ao Ruby on Rails

by Jackson Pires

História e Características

História

- https://pt.wikipedia.org/wiki/Ruby_(linguagem_de_programa%C3%A7%C3%A3o)

História

- https://pt.wikipedia.org/wiki/Ruby_(linguagem_de_programa%C3%A7%C3%A3o)

- 1993 - 1995 (Japão)
- 2000 (Programming Ruby - Livro em inglês)

Características

- https://www.ruby-lang.org/pt/about/

- Open-source
- De propósito geral
- Orientada a objetos (tudo em ruby é objeto)
- É uma linguagem de scripting
- Sintaxe limpa e fácil

Compilada vs Interpretada

Compilada

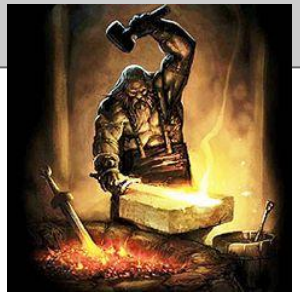
- Arquivos .exe / binários

Compilada

- Arquivos .exe / binários

Código-Fonte → Compilador → Executável

```c
#include <stdio.h>      // incluindo as bibliotecas
#include <stdlib.h>

void main( void ){      // inicio da função main
        int num, resultado;     // declaração de variáveis

        printf ("\nInforme o número a ser calculado: ");
        scanf ("%d",&num);
        // lendo o valor digitado e armazenando em num

        resultado = num * num;
        printf ("O quadrado de %d eh %d", num, resultado);
        // imprimindo o resultado
}

                                                    // Prog_C-01
```



Interpretada

- Arquivos de "script"

Interpretada

- Arquivos de "script"

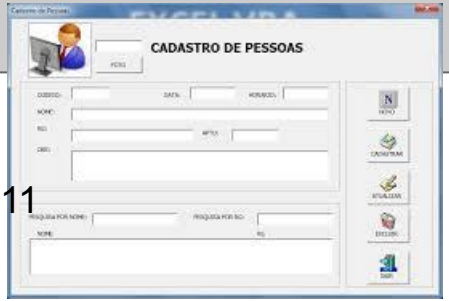
Interpretada

- Arquivos de "script"

Código-Fonte → Interpretador → Executável

```c
#include <stdio.h>      // incluindo as bibliotecas
#include <stdlib.h>

void main( void ){      // inicio da função main
        int num, resultado;     // declaração de variáveis

        printf ("\nInforme o número a ser calculado: ");
        scanf ("%d",&num);
        // lendo o valor digitado e armazenando em num

        resultado = num * num;
        printf ("O quadrado de %d eh %d", num, resultado);
        // imprimindo o resultado
}

                                                // Prog_C-01
```

010001011101011



Interpretada

- Ruby
  - MRI

REPL

REPL

- https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop

REPL

- https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop
- https://replit.com/

REPL

- https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop
- https://replit.com/

- **IRB** (Interactive Ruby) é o REPL do Ruby!

Olá Mundo!

Olá Mundo!

Arquivo:

```
# ola_mundo.rb

puts "Olá mundo!"
```

Prompt:

```
$> ruby ola_mundo.rb
```

RVM

RVM

Ruby Version Manager

- https://rvm.io/

```
$> rvm use <version> --default
```

Documentação

Documentação

- https://ruby-doc.org/
- https://rubyreferences.github.io/rubyref/

Tipos primitivos e variáveis

Tipos primitivos

- Números (integer, float)
    - Inteiros: `4, 76, 156`
    - Ponto flutuantes: `1.3, 4.5`

Tipos primitivos

- Caracteres (string)
    - `'A', '@', 'Curso de Ruby',`

Tipos primitivos

- Booleano
  - `true`
  - `false`

Tipos primitivos

- Primitivos?
  - `1.class`
  - `'a'.class`
  - `true.class`
  - `1.class.ancestors`

Variáveis

- O que é uma variável?

Constante

- O que é uma constante?

Entrada e saída padrão

Entrada e saída padrão

- `gets`
- `puts`

Entrada e saída padrão

- `gets`
  - `.chomp` (retira a quebra de linha)
- `puts`

Comentários

Comentários

- Uma linha
  - `# uma linha`
- Multilinha
  - `=begin`
    - `xxx`
  - `=end`

Strings e interpolação de variáveis

Strings e interpolação de variáveis

- String
  - `'abc'`
  - `"abc"`
- Interpolação
  - `#{<código ruby>}`

Coerção/Cast

Coerção/Cast

- `.to_i`
- `.to_f`
- `.to_s`

Operadores aritméticos, relacionais e de atribuição

Operadores aritméticos

- `+, -, *, /, %, **`

Operadores relacionais

- `>`, `<`, `>=`, `<=`, `==`, `!=`

Operadores de atribuição

- =,+=, -=,*=, /=, %=, **==

Estruturas condicionais

Estruturas condicionais

- `if … elsif … else`
- `unless`
- `case ... when`

Operadores lógicos

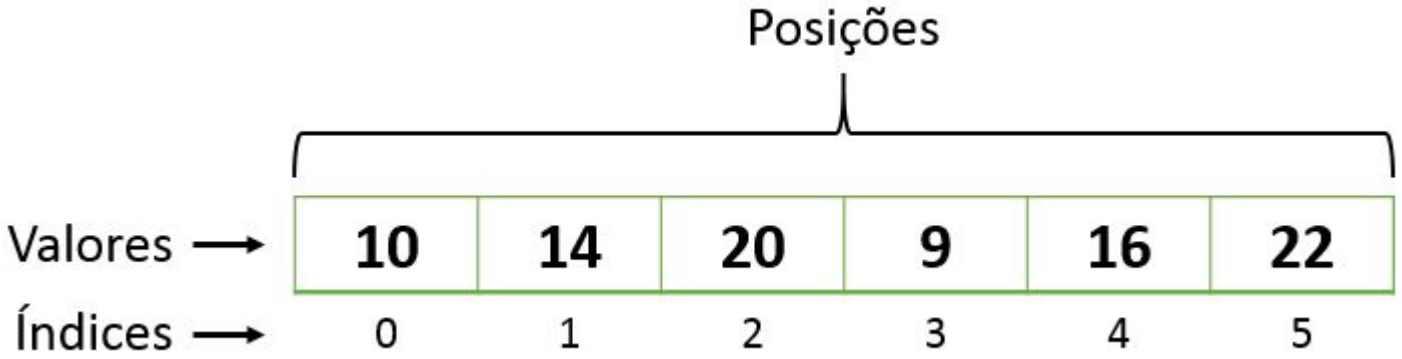
Operadores lógicos

- `&&, ||, !`
- `and, or, not`

Array, Hash, Símbolos e o iterador `each`

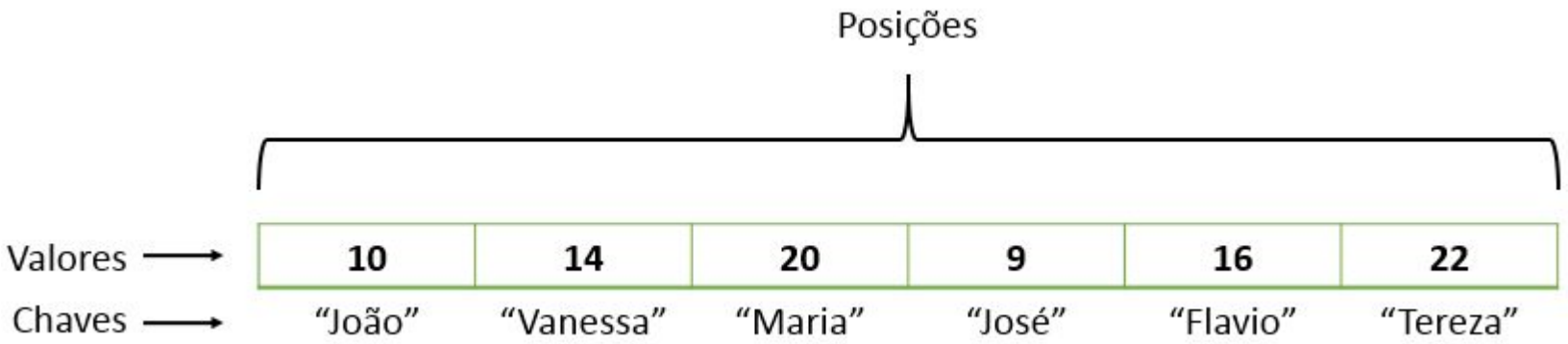
Array, Hash, Símbolos e o iterador `each`

- Array
  - `[ ]`



Array, Hash, Símbolos e o iterador `each`

- Hash
  - `{"key" => "value"}`



Array, Hash, Símbolos e o iterador `each`

- Símbolo
  - `:simbolo`

Array, Hash, Símbolos e o iterador `each`

- Iterador `.each`
  - `[93, 21, 35].each`

Exercícios

Faça um script que leia um número e imprima sua tabuada

- Dica: `.times`
- Ex:
    - Digite um número:
    - 5
    - ==========
    - Tabuada de 5
    - ==========
    - 5 x 0 = 0
    - 5 x 1 = 5
    - 5 x 2 = 10
    - …
    - 5 x 10 = 50

Faça um script que leia duas palavras e armazene em um array

- Dica: `.inspect`
- Ex:
    - Digite uma palavra:
    - jackson
    - Digite outra palavra:
    - pires
    - ==========
    - ['jackson', 'pires']
    - ==========

Faça um script que leia um número e imprima a tabuada em um hash

- Ex:
    - Digite um número:
    - 5
    - ==========
    - { "5x0" => 0, "5x1" => 5, "5x2" => 10, "5x3" => 15 … "5x10" => 50 }
    - ==========

Resolveremos o exercício na aula ao vivo