# **Estruturas de Controle**

### ****Conteúdo****

**Será abordado:**

* Estruturas de Controle
* Condicional
  + If/Else/Elsif
  + Unless
  + Case
* Iteração
  + For
  + While
  + Times
  + Do/While (Loop)
* Iteração + Condicional

### ****Estruturas de Controle****

Tratam-se de códigos que escrevemos em nossos programas para analisar dados e decidir qual caminho seguir. Divide-se em dois tipos, **Condicional** e **Iteração**. Nesta aula você aprenderá como utilizá-los.

### ****Condicional****

Tipo de estrutura de controle que executa um trecho de código dependendo do resultado de uma condição.  
Abaixo, veja as Instruções Condicionais **If, Else, Elsif, Unless** e**Case**.

#### ****If****

Expressão que verifica se uma condição é verdadeira(true), e a partir deste resultado determina se as instruções dentro de seu corpo serão ou não executadas.

1. Para entender como utiliza-lo crie um arquivo chamado**if.rb** e adicione o seguinte código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | day = 'Sunday'    if day == 'Sunday'    lunch = 'special'  end    puts "Lunch is #{lunch} today" |

Leia a condição da seguinte forma:  
Se o dia é Domingo, então o almoço é especial

2. Execute o programa (**ruby if.rb**) e veja que a condição é verdadeira, pois foi exibido na tela

|  |  |
| --- | --- |
| 1 | Today the lunch is special. |

#### ****Else****

Informa o que fazer quando a verificação de uma condição**if** for falsa.

1. Para utilizar o **else**, substitua o código do arquivo **if.rb** por

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | day = 'Saturday'      if day == 'Sunday'    lunch = 'special'  else    lunch = 'normal'  end    puts "Lunch is #{lunch} today" |

Leia a condição da seguinte forma:  
Se o dia é Domingo, então o almoço é especial  
Senão, o almoço é normal.

2. Execute o programa e veja que nossa condição é falsa. O bloco de código dentro do Else então é executado e por fim exibido na tela.

|  |  |
| --- | --- |
| 1 | Today the lunch is normal. |

#### ****Elsif****

Utilizado quando há a necessidade de verificar mais de uma condição em um **if**.

1. Substitua o código do arquivo **if.rb** por

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | day = 'Holiday'    if day == 'Sunday'    lunch = 'special'  elsif day == 'Holiday'    lunch = 'later'  else    lunch = 'normal'  end    puts "Lunch is #{lunch} today" |

Leia a condição da seguinte forma:  
Se o dia é Domingo, então o almoço é especial.  
Senão e se o dia é feriado, então o almoço é tarde.  
Senão, o almoço é normal.

Apenas nossa segunda condição é verdadeira.

2. Execute o programa e veja que o resultado será

|  |  |
| --- | --- |
| 1 | Today the lunch is normal. |

#### ****Unless****

Enquanto o**if** é executado quando sua condição é verdadeira, o **unless** ocorre de forma contrária. É executado apenas quando a condição é falsa.

1. Crie um arquivo chamado**unless.rb** e adicione o seguinte código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | product\_status = 'closed'    unless product\_status == 'open'    check\_change = 'can'  else    check\_change = 'can not'  end    puts "You #{check\_change} change the product" |

Leia da seguinte forma  
A menos que o estado do produto seja aberto, a troca é possível.  
Senão, a troca não é possível.

2. O produto não foi aberto. Execute o programa e veja que o resultado será

|  |  |
| --- | --- |
| 1 | You can change the product |

#### ****Case****

Instrução muito parecida com o **if**. Ele se enquadra muito bem a situações com diversas condições.

1. Em um novo arquivo chamado **case.rb** adicione o código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | puts 'Digite o número do mês em que você nasceu?'    month = gets.chomp.to\_i    case month  when 1..3    puts 'Você nasceu no começo do ano'  when 9..12    puts 'Você nasceu no final do ano'  when 4..6    puts 'Você nasceu na primeira metade do ano'  when 7..9    puts 'Você nasceu na segunda metade do ano!'  else    puts 'Não foi possível identificar'  end |

Leia a condição da seguinte forma  
Caso o mês informado  
esteja no intervalo entre 1 e 3, você nasceu no começo do ano  
esteja no intervalo entre 9 e 12, você nasceu no final do ano  
esteja no intervalo entre 4 e 6, você nasceu na primeira metade do ano  
esteja no intervalo entre 7 e 9, você nasceu na segunda metade do ano  
Senão, não foi possível identificar o mês

### ****Iteração****

Tipo de estrutura de controle que gerencia quantas vezes um trecho de código será executado.  
Abaixo, veja as instruções de iteração **For, While, Times, Do/While.**

#### ****For****

Usado para percorrer uma coleção de elementos.

1. Crie um programa chamado **for.rb** com o seguinte código

|  |  |
| --- | --- |
| 1  2  3  4  5 | fruits = ['Maçã', 'Uva', 'Morango']    for fruit in fruits    puts fruit  end |

No exemplo, a instrução **for** percorrerá todos os elementos da lista fruits. Em cada iteração, podemos acessar o elemento atual através da variável fruit.

2. Execute o programa e veja o nome da fruta cada vez que a repetição é executada.

|  |  |
| --- | --- |
| 1 | ruby for.rb |

3. Agora, substitua o código de **for.rb** por

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | fruits = ['Maçã', 'Uva', 'Morango']    fruit = "Laranja"    for fruit in fruits    puts fruit  end    puts fruit |

**CUIDADO**  
**Ao executar o programa, note que a variável de iteração pode sobrescrever outra que esteja fora da estrutura de repetição.**

#### ****While****

Instrução que repete um bloco de código enquanto sua condição é verdadeira.

1. Crie um programa chamado **while.rb** com o seguinte código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | x = 1    while x < 10    puts x    # Adiciona + 1 ao valor de x    x += 1  end |

Quando este programa é executado, a instrução **while** é repetida enquanto o valor de x for menor que 10.

#### ****Times****

Executa uma repetição por um especificado número de vezes.

1. Em um novo programa chamado **times.rb** adicione o seguinte código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | 2.times do    puts 'Estou aprendendo times!'  end    names = ['João', 'Alfredo', 'Juca']  # Igual ao array, o times começa com índice 0  3.times do |index|    puts names[index]  end |

Execute o programa e perceba que a estrutura times:  
Exibe a frase “Estou aprendendo times” 2 vezes  
Exibe um índice do array name por 3 vezes

#### ****Do/While****

Na verdade, no Ruby utilizamos uma estrutura de repetição chamada **loop** que faz o mesmo que o **do/while** em outras linguagens de programação. Ele cria um laço de repetição que só é parado quando uma instrução break for verdadeira.

1. Em um arquivo chamado **loop.rb** adicione o seguinte código.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | count = 1  loop do    puts count    break if count == 10    # Incrementa a variável count    count += 1  end |

Foi criado uma estrutura de repetição que só será parada quando o valor da variável count for igual a 10.  
Execute o programa e veja que ele contará de 1 a 10.

### ****Iteração + Condicional****

1. Veja um exemplo de como utilizar estas duas estruturas em conjunto, em **age.rb**, coloque:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | result = ''  loop do    puts result    puts 'Selecione uma das seguintes opções'    puts '1- Descobrir a idade de uma pessoa'    puts '0- Sair'    print 'Opção: '      option = gets.chomp.to\_i      if option == 1      print 'Digite o ano de nascimento: '      year\_of\_birth = gets.chomp.to\_i      print 'Digite o ano atual: '      current\_year = gets.chomp.to\_i      age = current\_year - year\_of\_birth      result = "Quem nasceu no ano de #{year\_of\_birth}, tem #{age} anos em #{current\_year}"    elsif option == 0      break if option == 0    else      result = 'Opção inválida'    end    # Comando que limpa o console    system "clear"  end |

Neste programa é criado uma estrutura de repetição que permite ao usuário descobrir a idade de alguma pessoa.  
O programa é executado até que a opção 0 que significa sair seja escolhida.

### ****Missões especiais****

Utilizando as estruturas de iteração e condição, **crie uma calculadora** que ofereça ao usuário a opção de **Multiplicar, Dividir, Adicionar ou Subtrair dois números**. Não se esqueça de também permitir que o usuário feche o programa.