Aula 08

Estruturas de controle de fluxo condicionais

Prof. Jalerson Lima

Apresentação

Nessa aula vamos dar início aos estudos das estruturas de controle de fluxo em Ruby. Essas estruturas nos permitem controlar como a execução do código é feita. Vamos iniciar com as estruturas de controle condicionais: *if*, *else*, *elsif*, *unless* e *case*. Elas nos permitem definir trechos de código que são executados dependendo de uma condição.

Objetivos

1. Compreender o conceito de estruturas de controle condicionais;
2. Aprender como utilizar as estruturas condicionais em Ruby: *if*, *else*, *elsif*, *unless* e *case*;
3. Exercitar os conhecimentos através de atividades práticas.

1. Introdução

Daremos início ao estudo das estruturas de controle de fluxo, que são comandos e instruções que nos permitem controlar como a execução do código será realizada. Os códigos que produzimos até o presente momento não incluíram nenhuma estrutura de controle de fluxo, e, portanto, a execução deles é simplesmente linear, ou seja, inicia na primeira linha, segue a execução das linhas seguintes e termina na última, sem nenhum trecho de código condicional ou repetição.

Nessa aula iremos estudar apenas uma categoria das estruturas de controle de fluxo: as estruturas condicionais. Essas estruturas permitem que trechos de código sejam executados dependendo de uma condição (expressão lógica). Essa condição normalmente é construída utilizando operadores lógicos e relacionais.

Para ilustrar como funciona uma das estruturas de controle de fluxo condicionais, vamos usar um exemplo do cotidiano: uma receita de brigadeiro adaptado do site Como Fazer Brigadeiro (http://www.comofazerbrigadeiro.com.br/). Confira a receita abaixo.

1. Aqueça a panela em fogo médio;
2. Acrescente 1 colher de sopa de manteiga;
3. Acrescente todo o leite condensado;
4. Se desejar fazer brigadeiro de chocolate, acrescente 4 colheres de sopa de chocolate em pó;
5. Mexa sem parar até que o brigadeiro esteja desgrudando da panela;
6. Unte um recipiente na qual a mistura será despejada;
7. Molde os brigadeiros com as mãos, passando-os no chocolate granulado.

Observe que a receita é uma sequência simples de instruções, mas o passo 4 é uma estrutura de controle condicional, pois você só irá acrescentar 4 colheres de sopa de chocolate em pó se desejar fazer brigadeiro de chocolate. Numa linguagem de programação, essa estrutura poderia ser reescrita da seguinte forma.

```
se desejar fazer brigadeiro de chocolate
  acrescente 4 colheres de sopa de chocolate em pó
fim
```

Observe que, na primeira linha, escrevemos a condição (se desejar fazer brigadeiro de chocolate), e, na segunda linha, escrevemos a instrução que deverá ser "executada" apenas se a condição da primeira linha for verdadeira. Na última linha escrevemos "fim" para marcar o fim da estrutura condicional. Observe que a estrutura condicional segue o formato apresentado abaixo.

```
se <condição>
  <instruções a serem executados caso a condição seja verdadeira>
fim
```

Um outro detalhe a ser observado nos exemplos de código acima é o espaçamento dado antes da segunda linha, de forma que a linha de código fica deslocada para a direita. Essa técnica é chamada de indentação de código (WIKIPÉDIA, 2015) e ela visa melhorar a legibilidade do código, ou seja, a facilidade de leitura e compreensão.

A indentação permite identificar visualmente que um trecho de código está dentro de uma estrutura. Nos exemplos apresentados anteriormente é possível identificar que a segunda linha está dentro da estrutura do "se". Vale salientar, contudo, que a indentação de código é opcional em Ruby, portanto seu código terá o mesmo comportamento se você usar a indentação ou não. Contudo, indentar seu código é altamente recomendado, tendo em vista que essa técnica melhora muito a legibilidade do código, especialmente em *scripts* complexos e longos.

2. Estruturas de controle condicionais em Ruby

2.1 Estruturas condicionais: if e else

A primeira estrutura condicional que vamos estudar é o *if*, que significa "se" em inglês. Na linguagem Ruby, o *if* é construído conforme a sintaxe ilustrada no Exemplo de código 1.

```
1  if <condição>
2    <trecho de código a ser executado quando a condição for  verdadeira>
3  end
```

Exemplo de código 1 – Sintaxe do `if`

Vale salientar que devemos sempre substituir o código que aparece entre `<` e `>` pelo que desejamos. Confira o Exemplo de código 2 que ilustra o exemplo de uso do `if`.

```
1  puts "Digite a sua idade:"
2  idade = gets.chomp.to_i
3
4  if idade >= 18
5    puts "Você é maior de idade!"
6  end
7
8  gets
```

Exemplo de código 2 – Exemplo de uso do `if`

Na linha 4 do Exemplo de código 2 escrevemos a instrução `if` seguido da condição `idade >= 18`. A linha 5 só será executada caso a condição seja verdadeira, ou seja, caso o valor da variável `idade` seja maior ou igual a 18. Na linha 6 escrevemos a instrução `end` que marca o fim da estrutura condicional. Caso seja dada uma idade inferior a 18 anos, a execução do código irá literalmente pular a linha 5.

Atividade 8.1

Crie um *script* em Ruby e digite o código apresentado no Exemplo de código 2.

Ao realizar a Atividade 8.1, você deve ter percebido que, ao digitar uma idade inferior a 18 anos, o *script* não apresenta nenhuma resposta. Idealmente ele deveria apresentar uma mensagem informando que o usuário é menor de idade. Para fazer isso, vamos utilizar uma outra instrução chamada `else`, que significa "senão" em inglês. Confira a sintaxe do `else` no Exemplo de código 3.

```
1  if <condição>
2    <trecho de código a ser executado quando a condição for  verdadeira>
3  else
4    <trecho de código a ser executado quando a condição for falsa>
5  end
```

Exemplo de código 3 – Sintaxe do `else`

O primeiro trecho de código será executado quando a condição for verdadeira e o segundo será executado quando a condição for falsa. Vale salientar que o `else` só pode ser usado acompanhado de um `if`, porque o `else` depende da condição do `if`. Agora confira o Exemplo de código 4, que ilustra o exemplo apresentado anteriormente usando o `else`.

```
1  puts "Digite a sua idade:"
2  idade = gets.chomp.to_i
3
4  if idade >= 18
5    puts "Você é maior de idade!"
6  else
7    puts "Você é menor de idade!"
8  end
9
10 gets
```

Exemplo de código 4 – Exemplo da maioridade usando o `else`

No Exemplo de código 4, se atribuirmos uma idade inferior a 18 na variável `idade`, o *script* irá apresentar a mensagem `"Você é maior de idade!"`, caso contrário, o *script* irá apresentar `"Você é menor de idade!"`. Observe que as duas mensagens nunca serão apresentadas ao mesmo tempo porque a condição só pode dar um resultado lógico: verdadeiro ou falso.

Observe novamente que a indentação de código nos permite identificar facilmente que o comando `puts` (linha 5) está dentro da estrutura do `if` e, portanto, só será executado se a condição for verdadeira. Além disso, é fácil ver que o outro `puts` (linha 7) está dentro da estrutura do `else`, e, portanto, será executado quando a condição for falsa.

Atividade 8.2

Crie um *script* em Ruby que permita que o usuário digite um número inteiro e verifique se esse número é par ou ímpar.

Uma forma mais simples de escrever o `if` é ilustrada Exemplo de código 5.

```
1  <trecho de código> if <condição>
```

Exemplo de código 5 – Sintaxe do `if` simplificado

Apesar da estrutura simplificada, o funcionamento permanece o mesmo: o trecho de código só será executado se a condição for verdadeira. Observe um exemplo de uso no Exemplo de código 6.

```
1  puts "Digite a sua idade:"
2  idade = gets.chomp.to_i
3
4  puts "Você é maior de idade" if idade >= 18
5
6  gets
```

Exemplo de código 6 – Exemplo de uso do `if` simplificado

2.2 Estrutura condicional: elsif

Outra estrutura de controle condicional é o `elsif`, que é uma combinação de um `if` com um `else`. Para melhor ilustrar a utilidade do `elsif`, tente resolver o exercício proposto na Atividade 8.3.

Atividade 8.3

Crie um *script* em Ruby que permita que o usuário digite uma nota de 0 a 100 (número inteiro) e calcule o conceito relativo à nota. Se a nota for entre 0 e 20, o conceito é “E”; entre 21 e 40, conceito “D”; entre 41 e 60, conceito “C”; entre 61 e 80, conceito “B”; entre 81 e 100, conceito “A”.

Uma das soluções possíveis para o exercício proposto na Atividade 8.3 é o código ilustrado no Exemplo de código 7.

```
1  puts "Digite uma nota de 0 a 100:"
2  nota = gets.chomp.to_i
3
4  if nota < 0 or nota > 100
5    puts "Nota inválida! Digite uma nota entre 0 e 100!"
6  else
7    if nota <= 20
8      puts "Conceito E"
9    else
10     if nota <= 40
11       puts "Conceito D"
12     else
13       if nota <= 60
14         puts "Conceito C"
15       else
16         if nota <= 80
17           puts "Conceito B"
18         else
19           puts "Conceito A"
20         end
21       end
22     end
23   end
24 end
25
26 gets
```

Exemplo de código 7 – Solução da Atividade 8.3

Observe, no Exemplo de código 7, que o código se desloca para a direita por causa da indentação. Uma opção para melhorar a legibilidade é usar o `elsif`, conforme ilustra o Exemplo de código 8.

```
1  puts "Digite uma nota de 0 a 100:"
2  nota = gets.chomp.to_i
3
4  if nota < 0 or nota > 100
5    puts "Nota inválida! Digite uma nota entre 0 e 100!"
6  elsif nota <= 20
7    puts "Conceito E"
8  elsif nota <= 40
9    puts "Conceito D"
10 elsif nota <= 60
11   puts "Conceito C"
12 elsif nota <= 80
13   puts "Conceito B"
14 else
15   puts "Conceito A"
16 end
17
18 gets
```

Exemplo de código 8 – Exemplo de uso do `elsif`

Observe que o código ficou mais curto pois com o uso do `elsif` foi possível eliminar a repetição de vários `if` e `else`.

Atividade 8.4

Crie um *script* em Ruby, digite o código apresentado no Exemplo de código 8 e o execute.

A construção do `elsif` sempre segue o formato apresentado no Exemplo de código 9.

```
1  if <condição_1>
2    <trecho código a ser executado quando a condição_1 for verdadeira>
3  elsif <condição_2>
4    <trecho de código a ser executado quando a condição_1 for falsa e a
5    condição_2 for verdadeira>
6  elsif <condição_3>
7    <trecho de código a ser executado quando a condição_1 e a condição_2
8    forem falsas e a condição_3 for verdadeira>
9  end
```

Exemplo de código 9 – Sintaxe do `elsif`

Atividade 8.5

Crie um *script* em Ruby que calcule o IMC (Índice de Massa Corporal) de uma pessoa. O *script* deve ler dois números reais. O primeiro número é o peso de uma pessoa em Kg, e o segundo é a altura de uma pessoa em metros. A fórmula para calcular o IMC é a seguinte:

$$IMC = \frac{peso}{altura^2}$$

- Se o IMC for abaixo de 17: apresentar “Muito abaixo do peso”;
- Se o IMC for entre 17 e 18,49: apresentar “Abaixo do peso”;
- Se o IMC for entre 18,5 e 24,99: apresentar “Peso normal”;
- Se o IMC for entre 25 e 29,99: apresentar “Um pouco acima do peso”;
- Se o IMC for entre 30 e 34,99: apresentar “Obeso”;
- Se o IMC for entre 35 e 39,99: apresentar “Obesidade severa”.

2.3 Estrutura condicional: unless

O `unless` é o inverso do `if`. O trecho de código abaixo do `if` é executado quando a condição é verdadeira. O trecho de código abaixo do `unless` é executado quando a condição for falsa. Observe a sintaxe do `unless` no Exemplo de código 10.

```
1  unless <condição>
2    <trecho de código a ser executado quando a condição for falsa>
3  end
```

Exemplo de código 10 – Sintaxe do `unless`

No Exemplo de código 10, o trecho de código da linha 2 só será executado quando a condição do `unless` for falsa. Para exemplificar o uso do `unless`, vamos usar um método chamado `empty?`. O método `empty?` é executado sobre uma variável que guarda uma *string* e verifica se ela está vazia ou não, portanto seria o equivalente a testar `variavel == “”`. Esse método retorna `true` quando a *string* está vazia, e `false` caso contrário.

```
1  puts "Digite o seu nome:"
2  nome = gets.chomp
3
4  if nome.empty?
5    puts "Você não digitou seu nome."
6  end
7
8  gets
```

Exemplo de código 11 – Exemplo de uso do `empty?`

O Exemplo de código 11 permite que o usuário digite o seu nome (*string*), que será guardado na variável `nome`. Em seguida, o *script* verifica, usando o método `empty?`, se o usuário

de fato digitou algo. Mas e se quiséssemos executar um trecho de código apenas quando o usuário digitar algo? É nesse tipo de situação que o `unless` pode ser útil, conforme ilustra Exemplo de código 12.

```
1  puts "Digite o seu nome:"
2  nome = gets.chomp
3
4  unless nome.empty?
5    puts "Olá #{nome}! Seja bem-vindo(a)!"
6  end
7
8  gets
```

Exemplo de código 12 – Exemplo de uso do `unless`

É claro que, ao invés de ter usado o `unless`, nós poderíamos ter usado apenas `if nome != “”` na condição da linha 4, mas a intenção aqui era apenas ilustrar um exemplo do `unless`.

Atividade 8.6

Crie um *script* em Ruby, digite o código apresentado no Exemplo de código 12 e o execute.

Assim como o `if`, o `unless` também tem uma forma mais simples de ser escrita, conforme ilustra o Exemplo de código 13.

```
1  <trecho de código> unless <condição>
```

Exemplo de código 13 – Sintaxe do `unless` simplificado

O trecho de código só será executado caso a condição seja falsa. Observe um exemplo de uso do `unless` simplificado no Exemplo de código 14.

```
1  puts "Digite o seu nome:"
2  nome = gets.chomp
3
4  puts "Olá #{nome}! Seja bem-vindo(a)!" unless nome.empty?
5
6  gets
```

Exemplo de código 14 – Exemplo de uso do `unless` simplificado

2.4 Estrutura condicional: case

O `case` é uma estrutura condicional interessante que nos permite substituir vários testes com `if`. Observe a sintaxe do `case` no Exemplo de código 15.

```
1  case <variável>
2  when <valor_1>
3    <trecho de código 1>
4  when <valor_2>
5    <trecho de código 2>
6  else
7    <trecho de código 3>
8  end
```

Exemplo de código 15 – Sintaxe do `case`

O `case` funciona da seguinte forma: uma variável é colocada ao lado do `case` (linha 1). O valor dessa variável é comparado com os valores aparecem ao lado dos `when` (linhas 2 e 4). Caso o valor da variável seja igual ao valor que aparece no `when`, o trecho de código correspondente é executado. Caso o valor da variável não combine com nenhum dos valores, o trecho de código do `else` é executado. Vale salientar, contudo, que o `else` é opcional.

Vamos a alguns exemplos para melhor ilustrar o uso do `case`.

Atividade 8.7

Crie um *script* em Ruby que permita ao usuário digitar um número inteiro que representa um mês do ano (1 para janeiro, 2 para fevereiro, 3 para março, etc.). O *script* deve apresentar o nome do mês e quantos dias ele tem.

Uma solução para o exercício proposto na Atividade 8.7 é ilustrado no Exemplo de código 16.

```
 1  puts "Digite o mês do ano:"
 2  mes = gets.chomp.to_i
 3
 4  case mes
 5  when 1
 6    puts "Janeiro tem 31 dias"
 7  when 2
 8    puts "Fevereiro tem 28 dias"
 9  when 3
10    puts "Março tem 31 dias"
11  when 4
12    puts "Abril tem 30 dias"
13  when 5
14    puts "Maio tem 31 dias"
15  when 6
16    puts "Junho tem 30 dias"
17  when 7
18    puts "Julho tem 31 dias"
19  when 8
20    puts "Agosto tem 30 dias"
21  when 9
22    puts "Setembro tem 31 dias"
23  when 10
24    puts "Outubro tem 30 dias"
25  when 11
26    puts "Novembro tem 31 dias"
27  when 12
28    puts "Dezembro tem 30 dias"
29  else
30    puts "Digite um número entre 1 e 12!"
31  end
32
33  gets
```

Exemplo de código 16 – Solução do exercício proposto na Atividade 8.7

Quando o valor da variável `mes` for igual a 1, a linha 6 será executada. Quando o valor da variável `mes` for igual a 2, a linha 8 será executada, e assim sucessivamente. Caso o valor da variável não seja igual ao valor de nenhum `when`, o trecho de código do `else` será executado, nesse caso, a linha 30.

Atividade 8.8

Resolva o exercício proposto na Atividade 8.3 usando a estrutura condicional `case`.

O Exemplo de código 17 apresenta uma solução para o exercício proposto na Atividade 8.8.

```
1  puts "Digite uma nota de 0 a 100:"
2  nota = gets.chomp.to_i
3
4  case nota
5  when 0..20
6    puts "Conceito E"
7  when 21..40
8    puts "Conceito D"
9  when 41..60
10   puts "Conceito C"
11 when 61..80
12   puts "Conceito B"
13 when 81..100
14   puts "Conceito A"
15 else
16   puts "Digite uma nota de 0 a 100!"
17 end
18
19 gets
```

Exemplo de código 17 – Solução do exercício proposto na Atividade 8.3 usando `case`

Observe, no Exemplo de código 17, que também é possível usar faixas de valores ao lado do `when`. Dessa forma, se o valor da variável `nota` estiver entre 0 e 20, a linha 6 será executada; se estiver entre 21 e 40, a linha 8 será executada; se estiver entre 41 e 60, a linha 10 será executada, e assim sucessivamente. Se a nota não estiver entre 0 e 100, a linha 16 será executada.

Atividade 8.9

Resolva o exercício proposto na Atividade 8.5 usando a estrutura condicional `case`.

Atividade 8.10

a) Crie um *script* em Ruby que lê três valores, 'a', 'b' e 'c' e dizer se estes valores podem ser os lados de um triângulo. Para um triângulo ser formado, a soma de dois lados deve ser maior do que o terceiro lado: a + b > c e a + c > b e b + c > a;
b) Modifique a questão anterior para contemplar o seguinte: quando os lados do triângulo forem válidos, o algoritmo deve informar qual é o tipo de triângulo formado pelos lados. O triângulo equilátero é formado quando os três lados são iguais. O triângulo isósceles é formado quando dois lados quaisquer são iguais, e o triângulo escaleno é formado quando os três lados são diferentes entre si;
c) Desenvolva um *script* em Ruby que, dada uma nota de 0 a 10, mostre o conceito relativo à nota, segundo a tabela a seguir (0 a 2 = E, 2 a 4 = D, 4 a 6 = C, 6 a 8 = B e 8 a 10 = A);
d) Crie um *script* em Ruby que permita o usuário digitar um ano. Em seguida, o *script* deve informar se o ano (informado pelo usuário) é ou não bissexto. Dica: um ano é bissexto se ele for divisível por 400; ou se ele for divisível por 4 e não por 100;
e) Crie um *script* em Ruby que receba três notas, calcule e mostre a média e, além disso, mostre a situação do aluno (aprovado, recuperação ou reprovado). Se a média for maior ou igual a 7, o aluno está aprovado; se for menor que 7 e maior ou igual a 5, o aluno está de recuperação; se for menor que 5, o aluno está reprovado;
f) Desenvolva um *script* em Ruby que leia a velocidade máxima permitida em uma avenida e a velocidade com que o motorista estava dirigindo nela. Calcule e mostre a multa que uma pessoa vai receber, sabendo que são pagos: R$ 50 reais se o motorista ultrapassar em até 10km/h a velocidade permitida; R$ 100 reais, se o motorista ultrapassar de 11 a 30 km/h a velocidade permitida; e R$ 200 reais, se estiver acima de 31km/h da velocidade permitida;
g) O imposto de renda de uma pessoa varia segundo uma tabela. Se o salário for menor do que R$ 1.000,00, não há imposto, se for entre R$ 1.000,00 e R$ 2.200,00, o imposto é de 13%, se for maior do que 2.200,00, o imposto é de 22%. Crie um *script* em Ruby que, dado um valor em reais informado pelo usuário, correspondente a um salário, informe o valor que será recebido (total menos imposto);
h) Desenvolva um *script* em Ruby que informe se uma data é válida ou não. O *script* deverá ler 3 números inteiros, que representem o dia, o mês e o ano da data;
i) Faça um *script* em Ruby que calcule o IMC (Índice de Massa Corporal) de uma pessoa. O *script* deve receber dois números reais, representando o peso da pessoa em Kg e a altura da pessoa em metros. O *script* deve calcular o IMC (peso / altura^2) e mostrar a situação da pessoa:
    - Se o resultado for abaixo de 17: muito abaixo do peso;
    - Se o resultado for entre 17 e 18,49: abaixo do peso;
    - Se o resultado for entre 18,5 e 24,99: peso normal;
    - Se o resultado for entre 25 e 29,99: um pouco acima do peso;
    - Se o resultado for entre 30 e 34,99: um pouco obeso;
    - Se o resultado for entre 35 e 39,99: obesidade severa;
    - Se o resultado for acima de 40: obesidade mórbida.

Resumindo

Essa aula deu início aos estudos das estruturas de controle de fluxo em Ruby. Tais estruturas são fundamentais para controlar como a execução do código é realizada. Iniciamos aprendendo sobre as estruturas de controle condicionais: *if*, *else*, *elsif*, *unless* e *case*.

Referências

RANGEL, E. **Conhecendo Ruby**. [S.l.]: Leanpub, 2014.

SOUZA, L. **Ruby - Aprenda a programar na linguagem mais divertida**. 1ª. ed. São Paulo: Casa do Código, v. I, 2012.

WIKIPÉDIA. Identação. **Wikipédia**, 2015. Disponivel em: <https://pt.wikipedia.org/wiki/Indenta%C3%A7%C3%A3o>. Acesso em: 24 nov. 2015.