Aula 05

Operadores aritméticos

Prof. Jalerson Lima

Apresentação

Nessa aula iremos começar a aprender como utilizar os operadores em Ruby, iniciando pelos operadores aritméticos. Para exercitar esses conhecimentos, vamos trabalhar com operadores e variáveis definidas pelo usuário, bem como produzir alguns *scripts* com problemas variados.

Objetivos

1. Compreender o conceito de operadores;
2. Aprender a utilizar os operadores aritméticos;
3. Aprender como utilizar os operadores aritméticos com variáveis;
4. Compreender como realizar operações aritméticas com variáveis definidas pelo usuário;
5. Aprender o conceito e como usar substituição de expressões.

1. Introdução

A linguagem Ruby, bem como outras linguagens, possui pelo menos três tipos de operadores: os aritméticos, os lógicos e os relacionais. Os operadores nos permitem realizar determinadas operações com os dados. Nessa aula iremos aprender sobre os operadores aritméticos.

2. Operadores aritméticos

Os operadores aritméticos são aqueles que aprendemos nas aulas de matemática: soma, subtração, multiplicação, divisão, entre outros. Em Ruby, nós temos seis operadores aritméticos a nossa disposição.

Tabela 1 - Operadores aritméticos em Ruby (POINT, 2015).

Nome	Símbolo	Descrição	Exemplo
Adição ou soma	+	Soma dois valores.	2 + 3 = 5
Subtração	-	Subtrai dois valores.	4 - 2 = 2
Multiplicação	*	Multiplica dois valores.	2 * 2 = 4
Divisão	/	Divide dois valores.	13 / 4 = 3
Módulo	%	Calcula o resto da divisão entre dois valores.	13 % 4 = 1
Expoente	**	Calcula a potência entre dois valores.	2 ** 3 = 8

Observe abaixo alguns exemplos de uso dos operadores aritméticos.

```
irb(main):001:0> 5 + 5
=> 10
irb(main):002:0> 3 - 5
=> -2
irb(main):003:0> 4 * 4
=> 16
irb(main):004:0> 22 / 11
=> 2
irb(main):005:0> 22 % 11
=> 0
irb(main):006:0> 3 ** 2
=> 9
irb(main):007:0> 12 / 0
ZeroDivisionError: divided by 0
        from (irb):7:in `/'
        from (irb):7
        from C:/Ruby22-x64/bin/irb:11:in `<main>'
```

Observe que quando tentamos calcular `12 / 0` o Ruby apresentou um erro: `ZeroDivisionError`. Esse erro ocorreu porque não é possível dividir nenhum valor por zero.

Atividade 5.1

Crie um *script* Ruby usando o Atom ou seu editor de código preferido. Nesse *script*, realize pelo menos uma operação com cada um dos operadores aritméticos apresentados.

3. Operadores aritméticos e variáveis

Lembra das variáveis? Também é possível utilizar os operadores aritméticos com variáveis. Confira abaixo o Exemplo de código 1.

```
1  a = 4
2  b = 2
3  puts a + b
4  puts a - b
5  puts a * b
6  puts a / b
7  puts a % b
8  puts a ** b
9  gets
```

Exemplo de código 1 - Usando operadores aritméticos com variáveis

Nas linhas 1 e 2 do Exemplo de código 1, criamos duas variáveis `a` e `b` e atribuímos valores a elas. Em seguida, usamos os operadores aritméticos para fazer operações matemáticas com os valores dessas variáveis. Observe que usamos o método `puts` para apresentar os resultados das operações.

Atividade 5.2

Crie um *script* Ruby usando o Atom ou seu editor de código preferido. Nesse *script*, realize pelo menos uma operação com cada um dos operadores aritméticos usando variáveis, conforme ilustrado no Exemplo de código 1.

4. Operadores aritméticos e variáveis definidas pelo usuário

Agora vamos um pouco além: vamos permitir que o usuário digite os valores das variáveis `a` e `b`, e em seguida vamos usar esses valores para realizar operações matemáticas usando os operadores aritméticos. Observe abaixo o Exemplo de código 2.

```
 1  puts "Digite um valor"
 2  a = gets.chomp.to_i
 3  puts "Digite outro valor"
 4  b = gets.chomp.to_i
 5  puts a + b
 6  puts a - b
 7  puts a * b
 8  puts a / b
 9  puts a % b
10  puts a ** b
11  gets
```

Exemplo de código 2 - Usando operadores com valores fornecidos pelo usuário

Observe que nas linhas 1 e 3 usamos o método `puts` para apresentar uma mensagem na tela solicitando que o usuário digite um valor. Nas linhas 2 e 4, usamos os métodos `gets.chomp.to_i` para permitir que o usuário digite um valor numérico no teclado. Esses valores digitados pelo usuário são guardados nas variáveis `a` e `b`.

Atividade 5.3

Crie um *script* Ruby usando o Atom ou seu editor de código preferido. Nesse *script*, digite o código apresentado no Exemplo de código 2, execute-o e verifique os resultados.

5. Apresentando melhor os resultados

Se você realizou a Atividade 5.3, deve ter observado que a execução do *script* apresentou algo parecido com o seguinte resultado.

```
Digite um valor
2
Digite outro valor
3
5
-1
6
0
2
8
```

O resultado apresentado pelo *script* não deixa claro para o usuário quais foram as operações matemáticas realizadas. Para ver quais foram as operações aritméticas realizadas, o usuário teria que ver o código do *script*. Contudo, é possível melhorar a apresentação dos resultados. Para isso, iremos utilizar a substituição de expressões (POINT, 2015).

A substituição de expressões nos permite incluir expressões dentro de *Strings*. Observe abaixo o Exemplo de código 3.

```
1  puts "Digite um valor"
2  a = gets.chomp.to_i
3  puts "Digite outro valor"
4  b = gets.chomp.to_i
5  puts "Calculando a soma entre #{a} e #{b}..."
6  puts "O resultado da soma é #{a + b}"
7  gets
```

Exemplo de código 3 – Exemplo com substituição de expressões

Atividade 5.4

Crie um *script* Ruby usando o Atom ou seu editor de código preferido. Nesse *script*, digite o código apresentado no Exemplo de código 3, execute-o e verifique os resultados.

Ao realizar o exercício proposto na Atividade 5.4, você deve ter observado um resultado parecido com o apresentado abaixo.

```
Digite um valor
2
Digite outro valor
3
Calculando a soma entre 2 e 3...
O resultado da soma é 5
```

Observe que o resultado apresentado não inclui `#{a}`, `#{b}` e o `#{a + b}` que aparecem nas linhas 5 e 6 do Exemplo de código 3. Na realidade, `#{a}` foi substituído pelo valor da variável `a`, `#{b}` foi substituído pelo valor da variável `b` e `#{a + b}` foi substituído pelo resultado da soma entre `a` e `b`.

Compreendido como funciona a substituição de expressões, agora podemos melhorar o nosso código. Observe a seguir o Exemplo de código 4.

```
 1  puts "Digite um valor"
 2  a = gets.chomp.to_i
 3  puts "Digite outro valor"
 4  b = gets.chomp.to_i
 5  puts "Você digitou os valores #{a} e #{b}, calculando..."
 6
 7  puts "A soma entre #{a} e #{b} é #{a + b}"
 8  puts "A subtração entre #{a} e #{b} é #{a - b}"
 9  puts "A multiplicação entre #{a} e #{b} é #{a * b}"
10  puts "A divisão entre #{a} e #{b} é #{a + b}"
11  puts "O resto da divisão entre #{a} e #{b} é #{a % b}"
12  puts "A potência entre #{a} e #{b} é #{a ** b}"
13  gets
```

Exemplo de código 4 – Melhorando nosso código

Atividade 5.5

Crie um *script* Ruby usando o Atom ou seu editor de código preferido. Nesse *script*, digite o código apresentado no Exemplo de código 4, execute-o e verifique os resultados.

Ao realizar o exercício proposto na Atividade 5.5, você deve visualizar resultados parecidos com esses apresentados abaixo.

```
Digite um valor
2
Digite outro valor
3
Você digitou os valores 2 e 3, calculando...
A soma entre 2 e 3 é 5
A subtração entre 2 e 3 é -1
A multiplicação entre 2 e 3 é 6
A divisão entre 2 e 3 é 5
O resto da divisão entre 2 e 3 é 2
A potência entre 2 e 3 é 8
```

Bem melhor, concorda? Observe que usamos bastante a substituição de expressões no código, para que o Ruby substituísse os nomes das variáveis pelos seus respectivos valores, bem como as operações matemáticas pelos seus respectivos resultados.

Atividade 5.6

a) Crie um *script* em Ruby que leia um número inteiro e mostre seu sucessor.
b) Crie um *script* em Ruby que leia um número inteiro e mostre seu antecessor.
c) Crie um *script* em Ruby que leia um número inteiro e mostre o dobro desse número.
d) Crie um *script* em Ruby que leia um número inteiro e mostre a metade desse número.
e) Crie um *script* em Ruby que lê dois números inteiros, X e Y, e mostre o quociente e o resto da divisão de X e Y.
f) Crie um *script* em Ruby que leia uma idade e calcula quantos dias essa pessoa já viveu.
g) Crie um *script* em Ruby que lê dois números reais, calcula e mostra a soma deles, o produto e o quociente.
h) Crie um *script* em Ruby que lê dois números, X e Y, calcula X elevado a Y e mostra o resultado.
i) Crie um *script* em Ruby que lê dois números, X e Y, e mostra o resto da divisão entre eles.
j) Crie um *script* em Ruby que lê o salário de um funcionário, reajusta o salário em 7% e mostra o resultado.
k) Crie um *script* em Ruby que lê um valor real em dólar, e converte o valor para reais. Considere que a cotação é US$ 1 = R$ 1,82.
l) Crie um *script* em Ruby que leia uma distância (em Km) entre dois pontos e o preço da gasolina em reais. Depois, calcule e mostre quantos litros de gasolina o carro irá consumir e quanto será o gasto em reais. Considere que o carro consegue percorrer 12 km com um litro de gasolina.
m) Crie um *script* em Ruby que leia as variáveis inteiras n1 e n2 e troque o valor destas variáveis. Isto é, n1 deve ficar com o valor de n2 e n2 deve ficar com o valor de n1. Mostre os valores depois da troca.

Resumindo

Nessa aula começamos a aprender sobre os operadores em Ruby, iniciando pelo compreensão e uso dos operadores aritméticos. Realizamos operações matemáticas com variáveis definidas pelo usuário, além de realizar um tratamento usando substituição de expressões para apresentar melhor os resultados dos nossos *scripts*.

Referências

POINT, T. Ruby Operators. **Tutorials Point**, 2015. Disponivel em: <http://www.tutorialspoint.com/ruby/ruby_operators.htm>. Acesso em: 03 nov. 2015.

POINT, T. Ruby Strings. **Tutorials Point**, 2015. Disponivel em: <http://www.tutorialspoint.com/ruby/ruby_strings.htm>. Acesso em: 04 nov. 2015.

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