2 Expressões

Onde o conceito de "expressão" em programação é introduzido e como os diversos tipos de expressões são representadas em pseudocódigo e em Java.

Pré-requisito(s):	Objetivos (ao final você deverá ser capaz de):
• Saber identificar um problema como computacional • Saber compilar e executar um programa em Java	• Saber o que é uma expressão e conhecer seus diferentes tipos • Conhecer os tipos de dados básicos • Escrever a solução de um problema simples de expressões em linguagem de pseudocódigo • Estabelecer um paralelo entre a sintaxe de pseudocódigo e a sintaxe de Java

Você certamente já ouviu falar de expressões na matemática. Uma expressão na matemática é uma sentença composta por valores numéricos e operadores aritméticos e relacionais que, aplicados sobre esses valores, resultam em um novo valor. Exemplos de operadores aritméticos bastante conhecidos são os operadores de adição, subtração, divisão e multiplicação. Maior, menor, igualdade e diferença são exemplos de operadores relacionais. Pois bem, o conceito de "expressão" pode ser generalizado para qualquer sentença – não apenas aquelas que lidam com valores numéricos – composta de elementos e de operadores que combinados de forma coerente podem ser avaliados e resultarem em um novo valor. Um programa de computador faz uso de expressões a todo instante. Além de expressões aritméticas, um programa pode utilizar expressões lógicas e expressões literais. Este capítulo mostra como definir e utilizar os diversos tipos de expressões em programação. O capítulo mostra ainda que diferentes tipos de expressões lidam com diferentes tipos de dados, mostra quais os tipos de dados mais comuns existentes e como fazer para utilizá-los em um programa.

2.1 Valores e Tipos de Dados

Já vimos que instruções, dados e informação são elementos de composição de um algoritmo.

Um dado que é processado durante uma computação possui um **valor**. O valor é de fato o componente mais básico da computação. Durante uma computação, o valor de um dado pode ser avaliado, armazenado, incorporado em estruturas de dados, passado como parâmetro, retornado como resultado, entre outras ações.

Todo valor possui um **tipo de dado**. O tipo de um valor define as operações que podem ser executadas sobre aquele valor, e valores de um determinado tipo devem exibir comportamento uniforme em relação às operações sobre o tipo.

Os tipos de dados que trataremos nesse capítulo são os chamados **tipos primitivos**, e agrupam valores atômicos (que não podem ser decompostos em valores menores). Os tipos primitivos podem ser divididos em numéricos, literais e lógicos.

Os tipos numéricos, por sua vez, podem ser:

- **Inteiro**: qualquer valor numérico que pertença ao conjunto dos números inteiros (positivo, negativo, ou nulo). Exemplo: idade de uma pessoa.
- **Real**: qualquer valor numérico que pertença ao conjunto dos números reais (positivo, negativo, ou nulo), podendo dessa forma ter componente fracionária. Exemplo: preço de um produto no supermercado.

O tipo **Literal** é composto por letras, dígitos numéricos e/ou símbolos especiais. Normalmente, são assinalados por aspas ou apóstrofos. O comprimento N de um literal L consiste no número N de caracteres apresentados por L. Exemplo: o literal “casa” tem comprimento 4, e o literal “Av. 15 de março” tem comprimento 15.

O tipo **Lógico** pode assumir os valores *verdadeiro* ou *falso*. Também é conhecido como Booleano. Exemplo: a declaração *10 é menor que 5* é falsa.

A figura 2.1 ilustra alguns valores de dados e seus respectivos tipos:

213	**Inteiro**
213,0	**Real**
-213	**Inteiro**
“213”	**Literal**
verdadeiro	**Lógico**
10^{15}	**Inteiro**
10^{-5}	**Real**
“ICC 2009.1”	**Literal**

Figura 2.1 – Tipos de dados.

2.2 Expressões Aritméticas

Uma expressão aritmética é aquela que gera como resultado de sua computação um valor numérico. Ela possui **operadores aritméticos** (figura 2.2) que são aplicados a valores numéricos, os operandos da expressão.

OPERAÇÃO	OPERADOR
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Exponenciação	**

Figura 2.2 – Operadores aritméticos.

Para resolução dessas expressões são aplicadas as operações conforme a ordem determinada pela prioridade definida classicamente pela matemática:

1) Sinais;

2) Exponenciação;

3) Divisão e multiplicação;

4) Adição e subtração.

Exemplos de expressões aritméticas:

```
a)  2,1 * 2 + 8 / 4
4,2 + 2
6,2
b) -5 * 2 ** 3 + 1
-5 * 8 + 1
-40 + 1
-39
```

Os parênteses têm o "poder" de redefinir a prioridade de resolução das operações. Exemplo:

```
a) 2,1 * (2 + 8) / 4
2,1 * 10/4
21,0/4
5,5
b) (-5 * 2) ** (3 + 1)
-10 ** 4
10000
```

A figura 2.3 traz um pseudocódigo que exibe o resultado do cálculo da área de um triângulo retângulo com os valores da base e da altura já fornecidos.

```
1.  Início
2.     escreva ("base = 5");
3.     escreva ("altura = 10");
4.     escreva ("area = ", 5*10);
5.  Fim.
```

Figura 2.3 – Cálculo da área de um triângulo.

2.3 Expressões Lógicas

Uma expressão lógica gera como resultado de sua computação é verdadeiro (V) ou falso (F), um valor lógico. Ela pode ser formada por **operadores lógicos** (figura 2.4) ou **operadores relacionais** (figura 2.5).

OPERAÇÃO	OPERADOR	PRECEDÊNCIA
Negação	não	1°
Conjunção	e	2°
Disjunção	ou	3°

Figura 2.4 – Operadores lógicos.

OPERAÇÃO	OPERADOR
Igual a	=
Maior que	>
Menor que	<
Maior ou igual a	>=
Menor ou igual a	<=
Diferente de	<>

Figura 2.5 – Operadores relacionais.

O valor do resultado de uma expressão lógica depende dos valores de seus operandos, segundo as regras a seguir:

1. Negação: o resultado é o oposto do valor do operando
2. Conjunção: o resultado é verdadeiro somente se os operandos forem verdadeiros.
3. Disjunção: o resultado é verdadeiro se pelo menos um dos operandos forem verdadeiros.

A tabela da figura 2.6 ilustra o resultado das respectivas operações lógicas para todas as combinações de valores entre dois operandos.

Op1	Op2	não Op1	Op1 e Op2	Op1 ou Op2
V	V	F	V	V
V	F	F	F	V
F	V	V	F	V
F	F	V	F	F

Figura 2.6 – Resultados de operações lógicas.

Os operadores relacionais são utilizados para se realizar comparações entre dois valores de um mesmo tipo primitivo. O resultado de uma operação relacional também é um valor lógico.

Exemplos de operações relacionais:

```
a)  2 > 8
falso (F)
b) -5 <= 0
verdadeiro (V)
c) -1 <> 1
verdadeiro (V)
```

Exemplos de expressões lógicas com operadores lógicos e relacionais:

```
a) 2 < 5 ou 15/3 = 5
V ou V
V
b) não (5 <> 10/2 ou V e 2 - 5 > 5 -2 ou V)
não (5 <> 5 ou V e -3 > 3 ou V)
não (F ou V e F ou V)
não (F ou F ou V)
não (V)
F
```

Neste último exemplo é possível perceber que a ordem de precedência que deve ser respeitada entre os três tipos de operadores é a seguinte:

1) parênteses

2) operadores aritméticos

3) operadores relacionais

4) operadores lógicos

2.4 Expressões Literais

Uma expressão literal é formada por operandos que são valores literais. O

resultado de sua computação é também um valor literal.

Existe apenas um operador exclusivo para expressões literais, que é o operador de concatenação. Utiliza-se o símbolo + para representar a concatenação de dois literais.

Exemplos de expressões literais:

```
a) "Machado" + " de " + "Assis"
"Machado de Assis"
b) "ECO" + ", " + "Humberto"
"ECO, Humberto"
c) "Minha senha é: " + "2009rxt123@"
"Minha senha é: 2009rxt123@"
```

2.5 Identificadores e Constantes

Vimos que todos os dados processados por uma máquina possuem um valor associado e que esse valor deve ser de um determinado tipo. Vimos também alguns tipos de expressões que podem ser formadas com a manipulação direta esses valores.

Para que se possa organizar e entender melhor um algoritmo (e um programa) é interessante em muitos casos representar esses valores através de um nome que represente intuitivamente o valor. Esse nome é chamado de **identificador**.

Os identificadores possuem uma regra de formação mais ou menos universal:

1) Devem começar por uma letra; e
2) Podem ser seguidos por mais letras ou dígitos.

Exemplos de identificadores:

Válidos: X, Y, Nota1, Nota2, Media, BASE, ALTURA, xyz1234

Inválidos: 5x, A:B:C, m(4), Pink&Blue

Com o uso de identificadores, podemos amarrar um valor a um nome intuitivo para facilitar a leitura e o entendimento de expressões. Quando esse valor não é modificado durante toda a execução do programa, dizemos que o nome representa uma **constante**.

A figura 2.7 ilustra o mesmo pseudocódigo da figura 2.3 com o uso de nomes para identificar os valores constantes.

```
1.  Início
2.     constantes
3.        BASE = 5;
4.        ALTURA = 10;
5.     escreva ("AREA = ", BASE*ALTURA);
6.  Fim.
```

Figura 2.7 – Cálculo da área de um triângulo com uso de constantes.

2.6 Codificação de Expressões em Java

Para que seja possível iniciar o entendimento de codificação usando a sintaxe da linguagem Java, esta seção mostrará como representar os tipos de dados primitivos, as expressões e o uso de identificadores em Java.

2.6.1 Tipos de Dados Primitivos

A figura 2.8 relaciona os tipos de dados primitivos em Java, a memória necessária para armazenar os valores de cada um dos tipos e os limites de valores respectivos.

Nome do Tipo	Tipo de Valor	Memória usada	Limite de Valores
byte	inteiro	1 byte	-128 a 127
short	inteiro	2 bytes	-32768 a 32767
int	inteiro	4 bytes	-2,147,483,648 a 2,147,483,647
long	inteiro	8 bytes	-9,223,372,036,854,775,808 a 9,223,374,036,854,775,808
float	real - vírgula flutuante	4 bytes	+/- 3.4028... x 10^{+38} a +/- 1.4023... x 0^{-45}
double	real - vírgula flutuante	8 bytes	+/- 1.767... x 10^{+308} a +/- 4.940... x 0^{-324}
char	único caracter (Unicode)	2 bytes	todos os caracteres Unicode
boolean	*true* ou *false*	1 bit	não é aplicável

Figura 2.8 – Tipos de dados primitivos em Java.

O tipo *char* armazena um único caractere. Os valores deste tipo aparecem entre apóstrofos. O trecho de código Java abaixo, por exemplo, exibiria a letra y como resultado.

```
public static final char resposta = `y`;
System.out.println(resposta);
```

O tipo *boolean* descreve os valores lógicos *true* ou *false*. Expressões lógicas em Java fazem uso dos operadores lógicos e operadores relacionais ilustrados na figura 2.9.

Operador	Significado
&, &&	e
|, ||	ou
!	não

Operador	Significado
==	igual a
!=	diferente de
>	maior que
>=	maior ou igual a
<	menor que
<=	menor ou igual a

Figura 2.9 – Operadores lógicos e relacionais em Java.

A figura 2.10 relaciona os operadores aritméticos utilizados pela linguagem Java.

Operador	Significado
+	adição
-	subtracção
*	multiplicação
/	divisão (inteira, se ambos os argumentos o forem)
%	resto da divisão inteira

Figura 2.10 – Operadores aritméticos em Java.

A figura 2.11 mostra a precedência de alguns dos operadores mais comuns em Java:

Precedência	Operador	Tipo
14	++ -- + - !	(unários)
13	(tipo)	conversão
12	* / %	aritméticos
11	+ -	aritméticos
9	< <= > >=	relacionais
8	== !=	relacionais
7	&	lógico
6	^	lógico
5	|	lógico
4	&&	lógico
3	||	lógico
2	? :	condicional
1	= += -= /= %=	atribuição

Figura 2.11 – Precedência de operadores em Java.

2.6.2 O Tipo String

Em Java, o tipo *literal* que definimos anteriormente para representar um valor textual é codificado como uma **string**. Dessa forma, uma string em Java é uma

seqüência de caracteres.

A API de Java possui uma classe denominada *String* que possui métodos para manipular esse tipo de valor. [*Nota: veremos mais adiante do que se tratam métodos em Java. Por enquanto, basta saber que representam um conjunto de instruções que manipulam um determinado valor e que podem retornar resultados de alguma computação efetuada*].

Abaixo estão ilustrados alguns exemplos de constantes do tipo String em Java. Note que em Java um identificador é associado a uma constante através de uma sentença na forma `final` *`tipo identificador = valor.`*

```
final String CIENTISTA = "Isaac Newton";
final String CUMPRIMENTO = "Bom dia a todos";
CUMPRIMENTO.charAt(0) retorna o caracter B;
CUMPRIMENTO.charAt(2) retorna o caracter m;
CUMPRIMENTO.substring(4,7) retorna a substring dia;
CUMPRIMENTO.lenght() retorna o valor 15;
CUMPRIMENTO.equals(CIENTISTA);
```

Uma vez que o tipo String em Java armazena uma seqüência de caracteres, o índice de um caractere numa string é um inteiro que indica a posição do caractere e começa em 0 para o primeiro caractere. Observe a figura 2.12 que ilustra essa propriedade para a constante CUMPRIMENTO definida anteriormente.

B	o	m		d	i	a		a		t	o	d	o	s
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Figura 2.12 – String como uma seqüência de caracteres indexados.

Abaixo estão ilustradas exemplos de expressões Java que podem ser formadas através do uso de métodos pré-definidos da classe String e que utilizam essa propriedade referente aos caracteres numa string.

```
CUMPRIMENTO.charAt(0) retorna o caractere 'B';
CUMPRIMENTO.charAt(2) retorna o caractere 'm';
CUMPRIMENTO.substring(4,7) retorna a substring "dia";
CUMPRIMENTO.lenght() retorna o valor 15 (tamanho da string);
CUMPRIMENTO.equals(CIENTISTA) retorna false (testa se as duas
strings possuem o mesmo valor);
```

A concatenação de Strings é realizada através do operador +. Uma curiosidade a respeito do uso do operador + é que é possível se concatenar um valor do tipo String com um valor numérico, por exemplo. Nesse caso, o resultado da concatenação é uma string. A figura 2.13 traz a versão Java para o pseudocódigo

da figura 2.7. Observe o uso do operador + para concatenar o valor string "AREA = " com o valor inteiro 50, resultado da expressão BASE*ALTURA. Será exibido como resultado do programa a string "AREA = 50".

```
1.  public class AreaTriangulo {
2.     public static void main(String[] args) {
3.         final int BASE = 5;
4.         final int ALTURA = 10;
5.         System.out.print("AREA = " + BASE*ALTURA);
6.     }
7.  }
```

Figura 2.13 – Uso de constantes em Java.

Resumo

- Todo dado a ser processado possui um valor e um tipo associado.
- Os tipos de dados mais básicos são os tipos inteiro (números), real (números com casa decimal), lógico (verdadeiro ou falso), literal (valores alfanuméricos).
- Esses dados são submetidos a expressões durante um programa.
- Expressões são sentenças compostas por valores e operadores, que aplicados sobre esses valores retornam um novo valor.
- Existem pelo menos três tipos de expressões em programação: expressões aritméticas, expressões lógicas e expressões literais.
- Expressão aritmética é composta por valores numéricos, operadores aritméticos (soma, subtração, multiplicação, divisão).
- Uma expressão lógica é composta de valores lógicos com operadores lógicos (e, ou, não) ou numéricos com operadores relacionais (comparação de superioridade, inferioridade, igualdade e diferença).
- Uma expressão literal lida com valores tipo texto e é composta fundamentalmente pelo operador de concatenação, que unifica dois pedaços de texto.
- Com o intuito de facilitar a escrita e leitura de um algoritmo, valores podem ser associados a um termo chamado de identificador.
- Um identificador que representa um valor imutável é chamado de "constante".
- Todos os tipos de expressões citados podem ser implementadas na linguagem Java.

Na próxima aula...

… veremos o conceito de um "comando" em programação de computadores e aprenderemos a fazer leitura e escrita de valores em um programa.

Referências e Sugestões de Leitura

O uso de expressões em algoritmos é abordado no capítulo 2 de

FORBELLONE, A., EBERSPÄCHER, H. Lógica de Programação. Prentice Hall, 3ed, 2005.

Para o aluno interessado em aprofundar conhecimento sobre a semântica de *Expressões* no contexto de linguagens de programação, o capítulo 7 de

SEBESTA, R. Conceitos de Linguagens de Programação, Bookman, 5ed, 2005.

é altamente recomendado.

O capítulo 3 de

VAREJÃO, F. Linguagens de Programação: Conceitos e Técnicas, Campus, 2004.

também é recomendado para o aluno que deseja aprofundar os conhecimentos teóricos acerca dos conceitos de Valor e Tipo de dado em programação.

O capítulo 2 de

DEITEL, H., DEITEL, P. Java como Programar, Prentice-Hall, 2005.

mostra como construir os diversos tipos de expressões em Java.

Exercícios Propostos

2.1 – Identifique uma situação problema cuja solução implique no uso de um algoritmo. Escreva o algoritmo usando pseudocódigo e assinale neste os tipos de dados manipulados.

2.2 – Determine os resultados obtidos na avaliação das expressões seguintes, sabendo que A, B, C contém respectivamente *2, 7* e *3,5* e que existe uma variável lógica L cujo valor é *falso*:

a) 3 ** 2/3 + 15 – 35+7

b) B = A * C e (L ou *verdadeiro*)

c) B/A = C ou B/A <> C

d) L ou B**A <= C * 10 + A * B

e) “Macedo” + “, “ + “H” + “.”

2.3 – Escreva um algoritmo em pseudocódigo para transformar para Fahrenheit o valor de uma temperatura em graus Celsius. Considere a constante T = 28, que representa o valor em Celsius.

2.4 – Passe o algoritmo anterior para Java, compile e execute