



Introdução à linguagem Java

Introdução

Durante todo o dia nós nos deparamos com algoritmos, desde quando acordamos até quando estamos dormindo. Afinal, um algoritmo é uma sequência finita de ações que tem como objetivo produzir solução para um dado problema. Por exemplo, se lhe perguntar como você faz para escovar seus dentes, você me dará uma lista com um passo a passo que vai desde pegar o tubo do creme dental até enxaguar a boca, fechando a torneira e guardando todos os itens no final do processo.

Algoritmo então não é apenas programação, mas também pode ser utilizado na programação de computadores. Deste modo, nesta aula você compreenderá os conceitos essenciais para o desenvolvimento de algoritmos e programação com a linguagem Java. Também verá que existem diversas ferramentas para o desenvolvimento de códigos. No final, conhecerá os principais conceitos relacionados a programação.

Objetivos da aula

- Compreender o conceito de variáveis e os dados que elas armazenam.
- Reconhecer o tipo String e sua distinção com os demais tipos de dados.
- Compreender as operações de entrada e saída de dados.

- Conhecer os operadores aritméticos e como podem ser empregados em um algoritmo.



Resumo

Para entender corretamente como construir um algoritmo, primeiramente é necessário compreender a lógica de programação.

A lógica é uma parte da filosofia que tem como objetivo estudar e aplicar as leis do raciocínio humano, logo, podemos concluir que é a arte de pensarmos corretamente, colocando o pensamento em ordem. Por exemplo, considere estas duas declarações:

Todo mamífero é animal

Cavalo é um mamífero

Analisando as declarações, podemos concluir logicamente que cavalo é um animal, afinal ele é um mamífero e todo mamífero é um animal.

Agora, considere o exemplo em que lhe é dada a sequência numérica: **1 2 3 5 8 13 ?**.

Como definir qual será o próximo número da sequência? Neste caso, o próximo valor é o 21. Isso pode ser afirmado pois o valor atual é sempre a soma dos dois números anteriores:

- $3 + 2 = 5$;
- $5 + 3 = 8$;
- $5 + 8 = 13$;

- $13 + 8 = 21$.




Agora, considere a seguinte sequência numérica: **66 36 18 ?**. Como definir qual será o próximo número desta sequência? Neste caso, o próximo valor é o 8. Isso pode ser afirmado pois o valor seguinte é sempre o resultado da multiplicação da dezena pela unidade do número anterior, observe:

- Partindo do 66, temos $6 \times 6 = 36$;
- $3 \times 6 = 18$;
- $1 \times 8 = 8$.

Para estes dois últimos exemplos, observe que estipulamos (para cada um deles) uma **sequência fixa e finita de procedimentos** que nos leva até o resultado.

Podemos então concluir que um algoritmo deve possuir as seguintes características:

- Cada passo do algoritmo deve ser uma instrução possível de ser realizada: considere o exemplo da viagem de Campinas até São Paulo. Existe uma sequência lógica de instruções e todas elas precisam ser executadas, se uma destas instruções não ocorrer, o algoritmo não será executado corretamente;
- A ordem de cada uma das instruções deve ser respeitada: observe o segundo exemplo, nele o valor de um número X é obtido somando os dois valores anteriores a ele em uma sequência numérica. Se esta instrução não for obedecida (ou seja, se não somarmos o valor atual


com o valor anterior para obter o próximo da sequência), o resultado não será conforme o esperado; 

- O algoritmo deve ser finito: considere ainda o segundo exemplo. O algoritmo é: para obter um valor X devemos somar os dois valores imediatamente anteriores. Observe que, apesar de ser possível obter uma sequência infinita de valores, o algoritmo é finito em suas instruções. O mesmo ocorre com os demais exemplos apresentados.

Mas como podemos representar os algoritmos? Tudo vai depender da proposta. Podemos representar os algoritmos por pseudocódigo, que é uma forma de escrever código de programação, porém em um código fictício, geralmente escrito em português. Outra maneira é a narrativa, ou seja, se alguém lhe perguntar: como faço para ir de São Paulo até Belo Horizonte, você indicará o caminho a seguir. Logo, você está narrando um passo a passo, ou seja, uma sequência de instruções. Os algoritmos também podem ser representados no formato de fluxogramas, que são formas geométricas ligadas por uma linha.

Porém, na programação de computadores, os algoritmos são representados em uma linguagem própria chamada de linguagem de programação. Uma linguagem de programação é uma notação projetada para conectar instruções a uma máquina ou um computador (FORBELLONE, 2005). As linguagens de programação são usadas principalmente para expressar algoritmos, sendo que algumas delas são utilizadas para objetivos gerais e outras para fins específicos. Muitas destas linguagens precisam ser declaradas de forma imperativa, enquanto outras linguagens de programação utilizam a forma declarativa. O programa pode ser dividido em duas formas, como sintaxe e semântica.

Cada linguagem de programação tende a suportar um estilo (ou paradigma) particular de programação. Embora vários fatores possam

afetar a escolha da linguagem de programação para uma tarefa específica, incluindo preferência pessoal, política corpora  ou simplesmente a disponibilidade de conhecimento e experiência interna suficientes em uma linguagem específica, a linguagem selecionada deve ser a mais adequada para a tarefa em questão.

Dentre os principais paradigmas de linguagem de programação, temos (MANZANO, 2016):

- **Paradigma imperativo ou estrutura sequencial:** O paradigma consiste em várias instruções e, após a execução de todas elas, o resultado é armazenado em variáveis, que são espaços na memória reservados para guardar dados. Neste paradigma, uma lista de instruções é apresentada ao computador para que ele possa executá-las uma a uma, passo a passo;
- **Paradigma procedural:** Trata-se de um paradigma de programação estruturado baseado no conceito de chamada de procedimento, os comandos podem se apresentar como sub-rotinas, métodos ou funções. As principais linguagens são: C, C++, Java e Pascal;
- **Paradigma orientado a objetos (POO):** Trata-se de um paradigma de programação que usa objetos para modelar entidades de dados. Embora a maioria das linguagens de programação tenha adotado características orientadas a objetos até certo ponto, as linguagens atualmente em destaque incluem: Java, Python, Ruby e C++;
- **Paradigma de processamento paralelo:** Neste paradigma, o processamento das instruções do programa é dividido entre vários processadores. Alguns exemplos de linguagens são o NESL (um dos mais antigos), C e C++;

- **Paradigma declarativo:** Este paradigma permite que o programador instrua o computador sobre o que ele precisa fazer, sem se preocupar em instruí-lo como alcançar um objetivo. Logo, trata-se de um paradigma em que o programador define o que precisa ser realizado pelo programa sem definir como ele deve ser implementado;
- **Paradigma de lógica:** O paradigma da programação lógica não é composto de instruções - é composto de fatos e cláusulas. Ele usa tudo o que sabe e tenta criar o mundo onde todos esses fatos e cláusulas



- **Paradigma funcional:** Trata-se de um paradigma que está no centro das atenções há um tempo por causa do JavaScript, uma linguagem de programação funcional que ganhou mais popularidade recentemente. Exemplos de linguagens: JavaScript, Scala e Haskell.

Cabe destacar que não existe o melhor paradigma ou o paradigma certo. Tudo dependerá da finalidade do projeto o qual você está desenvolvendo.

Independentemente do paradigma escolhido ou até mesmo da linguagem de programação, alguns conceitos são fundamentais e merecem uma atenção a mais. A começar pelas variáveis.

Ao criarmos nossos programas, é necessário que os valores permaneçam em memória, de modo que estes dados possam ser reaproveitados no código. Este é o conceito de variável.

Uma variável representa um local da memória que é usado para armazenar um valor, o qual pode ser do tipo numérico, caractere ou lógico. O valor da variável na memória pode ser manipulado a qualquer instante, podendo ser alterada durante a execução do nosso programa.

Pense na memória do computador como sendo uma rua. Agora, considere que nesta rua nós temos diversas casas, uma ao lado da outra. As casas são como variáveis: você pode entrar em uma casa, morar nela e também pode sair. Se você quiser saber quem mora na casa número 85 da rua memória, basta você bater na porta e perguntar: “quem está aí?”. Com as variáveis é a mesma coisa: você pode chamá-la em seu código e manipular seu valor interno.


Essa manipulação pode ser realizada via entrada do usuário, seja digitando em um teclado ou escolhendo uma opção em um menu touchscreen no celular. Por outro lado, a saída é o resultado de uma operação ou de uma transação. Observe que, seja na entrada ou na saída, estamos lidando com dados.

Os dados são os tipos brutos que manipulamos durante as transações. Você pode solicitar em seu programa que o usuário informe o ano de nascimento, por exemplo. Com este dado, você é capaz de realizar um cálculo e saber quantos anos ele tem ou disparar um e-mail o cumprimentando pelo seu aniversário.

Em linguagens como o Java, por exemplo, podemos lidar com números, valores booleanos e também com cadeias de caracteres.

Na linguagem Java podemos encontrar números de ponto flutuante e inteiros. Os números inteiros são definidos como `int`, por exemplo 35, 832, 421, entre outros. Os números de ponto flutuante são definidos como `double` ou `float`, por exemplo 0.0, 88.5, 64.33, entre outros. Existe mais um tipo de dados nesta categoria, que é o `long`, o qual é usado para armazenar números inteiros mais longos.

Estes dados numéricos armazenados em memória podem ser manipulados, principalmente, por meio de operações aritméticas (assim como na matemática).

Na programação, podemos utilizar os operadores de soma (+), subtração (-), divisão (/), multiplicação (*) e de módulo (%). 

Observe que os operadores utilizados na linguagem Java devem seguir a mesma ordem de precedência da matemática, logo, em uma operação como $8 - 4 * 3 / 2 + 6$, a primeira operação será a de multiplicação, seguida da operação de divisão e, por fim, as operações de soma e subtração. Se o programador desejar que a operação de soma seja a primeira a ser realizada, o indicado é que se coloque a expressão entre parênteses.

Assim como em outras linguagens, em Java as strings são identificadas como uma sequência de caracteres representados entre aspas. Um número entre aspas é um caractere do tipo string, porém, um número sem aspas se torna um tipo numérico (inteiro ou ponto flutuante).

Tópicos avançados

Algoritmos, abstração, decomposição e reconhecimento de padrões são pilares do chamado Pensamento Computacional. Pensar computacionalmente não é o mesmo que pensar como um computador, mas sim, pensar racionalmente para resolver problemas. Você pode compreender mais sobre pensamento computacional lendo o artigo Pensamento Computacional na educação básica acessando o link: <http://ojs.sector3.com.br/index.php/rbie/article/view/v29p604>.

Referência Bibliográfica

FORBELLONE, A. L. V.; EBERSPACHER, H. F. **Lógica de programação: a construção de algoritmos e estruturas de dados**. São Paulo: Prentice Hall, 2005.

MANZANO, J. A. N. G. **Algoritmos: lógica para desenvolvimento de programação de computadores**. 28. ed. São Paulo: Érica, 2016

Ir para questão