3 Comandos

Onde o conceito de variável é introduzido e é mostrado como estas são manipuladas através do uso de comandos.

Pré-requisito(s):	Objetivos (ao final você deverá ser capaz de):
• Saber o que é um valor e associá-lo a um tipo • Conhecer os tipos de expressões existentes e os operadores associados a cada um dos tipos • Saber criar uma expressão e representá-la em pseudocódigo e na linguagem Java	• Saber o que é uma variável e como ela representa endereços de memória • Conhecer os tipos de comandos possíveis em programação • Aprender a manipular variáveis através de comandos • Escrever e entender um código Java por completo

Linguagens de programação podem ser classificadas segundo diferentes visões. No capítulo vimos, por exemplo, que elas podem ser de alto ou de baixo nível. Vimos também que elas podem ser compiladas, interpretadas ou compiladas e interpretadas, que é o caso da linguagem Java. Entretanto, existe ainda uma terceira forma muito utilizada para classificar as linguagens: o paradigma de programação onde ela se enquadra. Um paradigma de linguagem é um termo abstrato que descreve o comportamento daquela linguagem em relação ao processador e à forma com a qual permite que programas sejam expressados. As linguagens podem ser imperativas, funcionais ou lógicas, por exemplo. Neste curso estamos lidando com o paradigma imperativo. No paradigma imperativo, os programas são criados através da combinação de comandos. Comandos indicam exatamente o que o processador deve fazer e como deve fazer. Este capítulo mostra os tipos de comandos existentes mais comuns que são utilizados para escrita de programas. O capítulo inicia definindo variáveis e como elas são utilizadas para representar logicamente um endereço de memória que armazena

valores. Variável é a abstração mais importante em uma linguagem de programação imperativa e qualquer tipo de comando faz uso dessa abstração. Na última seção do capítulo é mostrado como utilizar variáveis e representar os diversos tipos de comando na linguagem Java.

3.1 Variáveis

Como vimos no capítulo 1, a memória principal (RAM) do computador pode ser entendida como uma seqüência finita de caixas (posições de memória) que guardam algum tipo de dado ou instrução. O processador precisa localizar este dado ou instrução na memória. Fisicamente, cada posição de memória, possui um endereço, um número que indica onde o dado ou instrução está localizado. Este número é representado através da notação hexadecimal, tendo o tamanho de quatro, ou mais *bytes*.

O endereçamento das posições de memória através de números hexadecimais é facilmente compreendido pela máquina, mas não para nós humanos.

Para facilitar a identificação dessas posições de memórias por parte dos programadores, as linguagens de programação permitem que se associe livremente nomes diferentes a cada posição de memória. Dessa forma, os programadores ficam livres da tarefa de lidar com endereços físicos (números hexadecimais) e passam a trabalhar com **endereços lógicos** (os nomes atribuídos).

É importante ressaltar que o conteúdo desses endereços pode variar. Ou seja, o valor armazenado numa determinada posição de memória pode ser variações ao longo da execução de um programa. Por esta razão, esses endereços lógicos são chamados de forma genérica de **variáveis**.

Podemos dizer então que uma variável é uma posição de memória representada por um nome simbólico atribuído pelo programador, que contém algum valor em um determinado instante. Esse nome simbólico é chamado de **identificador da variável** e segue as mesmas regras definidas para especificação de identificadores de constantes definidas no capítulo 2.

A figura 3.1 mostra um exemplo de associação de um nome a um endereço físico e manipulação do valor armazenado.

O cálculo do volume de uma esfera ilustra o uso de variável. O volume é calculado por $4/3*\pi*R^3$, onde π tem valor constante igual a 3,1416..., que não varia independentemente de qual seja a esfera; já o valor do raio R depende da esfera e portanto é variável.

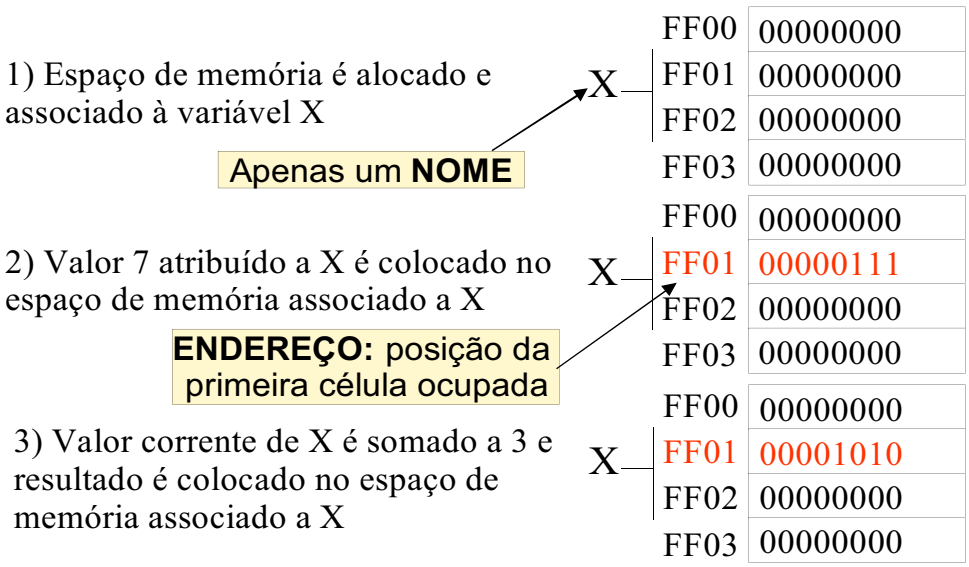

Figura 3.1 – Associação de um nome a um endereço físico de memória.

3.1.1 Declaração de variáveis

Uma determinada variável só pode armazenar valores de um mesmo tipo (numérico, lógico ou literal). Dessa forma, na elaboração de um algoritmo que faz uso de variáveis precisamos deixar claro que tipo de dado cada uma irá armazenar. Fazemos isso através de uma seção especial de declaração de variáveis, onde escolhemos um identificador e especificamos o seu tipo.

Exemplos de declaração de variáveis:

```
a) X, Y: inteiro;
b) Nome, Cidade: literal;
c) Altura, Preço: real;
d) Maioridade: lógico;
```

Maioridade, por exemplo, é um endereço de memória que só pode conter os valores *verdadeiro* ou *falso*. Já os identificadores X e Y são nomes para endereços de memória que só podem armazenar valores inteiros.

Para que uma variável possa ser usada em um programa ela obrigatoriamente deve ser declarada antes de seu primeiro uso.

3.2 Comando de Atribuição

Ao definirmos uma variável em um programa é natural que a utilizemos para atribuir valores. A forma utilizada pelas linguagens de programação para

armazenar um valor no endereço de memória identificado por uma variável é através de um **comando de atribuição direta**. Um comando de atribuição nos permite então fornecer um valor a uma variável ou alterar o valor de uma variável.

Em pseudocódigo, utilizamos a notação `V <- E` para representar uma atribuição; onde V representa a variável e E representa um valor (ou o resultado da avaliação de uma expressão, que também é um valor).

O pseudocódigo da figura 3.2 ilustra a declaração e o uso de variáveis com atribuição de valores.

```
1.  Algoritmo ExemploDeclaracao;
2.  Variáveis
3.     Numero: inteiro;
4.     Preco: real;
5.     Nome: literal;
6.  Início
7.     Numero <- 42;
8.     Preco <- 4.5;
9.     Nome <- "Maria";
10.    Numero <- Numero * Preco;
11.    Preco <- Preco * Numero;
12.    Preco <- Numero;
13.    Nome <- Preco;
14.    Numero <- Numero/3;
15. Fim.
```

Figura 3.2 – Declaração de variáveis e atribuição de valores.

Logo após a seção de declaração das variáveis (linhas 3, 4, 5), cada uma das três posições de memória armazenam o valor *indefinido*. Após as linhas 7, 8 e 9, o valor nas posições de memória para `Numero`, `Preco` e `Nome` respectivamente são 42, 4.5 e "Maria".

É importante atentar que a atribuição realizada na linha 10 contém um erro: `Numero` é uma variável inteira e portanto não pode armazenar um valor do tipo real, resultado da avaliação da expressão `Numero*Preco`. Da mesma forma, a atribuição da linha 13 está errada uma vez que a variável Nome só pode armazenar valores do tipo literal.

O pseudocódigo da figura 3.3 mostra um exemplo válido do uso de variáveis. O pseudocódigo calcula o preço final de um automóvel através da soma do preço de fábrica com o preço dos impostos (45% do preço de fábrica) e a percentagem do revendedor (28% do preço de fábrica).

```
1.  Algoritmo Automovel;
2.  Variaveis
3.     Nome: literal;
4.     Precofabrica, Precofinal, Imposto: real;
5.  Inicio
```

```
6.      Nome <- "Gol"
7.      Precofabrica <- 18000,00
8.      Imposto <- Precofabrica * (0,45 + 0,28)
9.      Precofinal <- Precofabrica + Imposto
10.     Nome <- "VW "+ "Gol"
11. Fim.
```

Figura 3.3 – Exemplo de uso de variáveis em um pseudocódigo para cálculo do preco final de um automóvel.

No exemplo, ao final da execução, as variáveis `Nome`, `Precofabrica`, `Imposto e Precofinal` possuirão os seguintes valores, respectivamente: *"VW Gol"*, *18000,00*, *13140,00* e *31140,00.*

3.3 Comandos de Entrada e Saída de dados

Vimos no capítulo 1 que o computador possui alguns dispositivos periféricos que permitem a comunicação da CPU e Memória com o mundo exterior. Ou seja, existem dispositivos que permitem a entrada de dados para processamento e alguns outros que permitem a divulgação do resultado do processamento. O teclado é o maior representante do primeiro conjunto e o monitor é o maior representante do segundo conjunto.

Na programação de computadores, existem comandos dedicados para alimentar a memória com dados fornecidos pelo usuário através de um teclado, por exemplo, e comandos dedicados para exibir as informações resultantes de um processamento. Esses comandos são respectivamente chamados de **comandos de entrada** e **comandos de saída**.

Os comandos que iremos utilizar nos pseudocódigos são os comandos `Leia` e `Escreva`. O comando `Leia` é responsável por atribuir o valor de um dado fornecido através de um dispositivo de entrada à uma variável identificada. O comando `Escreva` por sua vez é responsável por exibir em um dispositivo de saída o valor de uma variável identificada [*Nota: por padrão, utilizamos o teclado como dispositivo de entrada e o monitor como dispositivo de saída.*].

O pseudocódigo da figura 3.4 representa um programa que exibe o cálculo do valor da velocidade média de um automóvel ao percorrer uma determinada distância durante um período de tempo. Os valores da distância e do tempo são fornecidos pelo usuário.

```
1.  Algoritmo VelocidadeMedia;
2.  Variaveis
3.     DV, DT, DS: real;
4.  Inicio
```

```
5.      Escreva ("Digite a distancia percorrida: ");
6.      Leia (DS);
7.      Escreva ("Digite o tempo de percurso: ");
8.      Leia (DT);
9.      DV <- DS / DT;
10.     Escreva ("A velocidade média é ", DV);
11. Fim.
```

Figura 3.4 – Leitura de valores fornecidos pelo usuário para cálculo da velocidade média de um automóvel e escrita do resultado desse cálulo..

Inicialmente (linha 3), os valores das variáveis `DV, DT, DS` estão indefinidos. No momento em que um comando `Leia` é executado, o processamento é interrompido até que o usuário forneça o valor pedido. Uma vez que o valor é digitado, ele é atribuído automaticamente à variável em questão. No exemplo, isso acontece por duas vezes, nas linhas 6 e 8. Finalmente, na linha 9, a expressão `DS / DT` é avaliada e o resultado é atribuído à variável `DV`. Os comandos Escreva são utilizados para orientar o usuário para entrada dos dados corretamente (linhas 5 e 7) e para exibir o resultado final do processamento (linha 10).

Uma simulação de execução do programa está ilustrada abaixo:

```
Digite a distancia percorrida: 100
Digite o tempo de percurso: 8
(calculo da velocidade)
A velocidade média é 12,5
```

3.4 Codificação de comandos em Java

A declaração de uma variável em Java é normalmente dividida em três partes: *modificadores*, *tipo do dado*, e *identificadores*. Os modificadores definem a visibilidade das variáveis e não são obrigatórios [*Nota: modificadores possuem importância particular quando utilizamos o conceito de orientação a objetos em Java. Como não é o caso, iremos usar apenas o modificador* `final` *quando definirmos constantes*]. O tipo do dado é um daqueles possíveis mostrados na figura 2.8. Os identificadores são os nomes criados pelo programador para representar as variáveis.

Java permite que valores iniciais possam ser atribuídos às variáveis assim que são declaradas. Na verdade, isso constitui uma boa prática de programação, uma vez que se evita problemas de compilação futuros ao se tentar, por exemplo, acessar o valor de uma variável que está com seu valor ainda indefinido.

A atribuição de valores a variáveis em Java é feita através do comando $V = E$, onde V é o identificador da variável e E representa o valor (ou expressão) a ser

atribuído à variável.

Para exibição de valores e resultados de processamento, a API padrão de Java disponibiliza dois comandos:

1) System.out.println(*valor*): escreve o valor e posiciona o cursor na linha seguinte;

2) System.out.print(*valor*): escreve o valor e mantém o cursor na mesma linha.

Infelizmente, os comandos para leitura de dados em Java não são tão simples ou fáceis de utilizar. Por esta razão, este livro acompanha uma classe especial chamada `System_in` criada exclusivamente para fornecer comandos simples de leitura de dados para quatro tipos definidos em Java. Para leitura de dados dos tipos inteiro, real, string ou leitura de apenas um caractere, utilizaremos respectivamente os comandos:
`System_in.readInt()`,`System_in.readFloat()`, `System_in.readString()`, e `System_in.readChar()`.

A figura 3.5 traz a codificação em Java para o problema do cálculo da velocidade média apresentado em pseudocódigo anteriormente.

```
1.  public class VelocidadeMedia {
2.     public static void main(String[] args) {
3.           float DV, DT, DS;
4.           System.out.print("Digite a distância percorrida: ");
5.           DS = System_in.readFloat();
6.           System.out.print("Digite o tempo de percurso: ");
7.           DT = System_in.readFloat();
8.           DV = DS / DT;
9.           System.out.print("A velocidade média é " + DV);
10.    }
11. }
```

Figura 3.5 – Exemplo de uso de variáveis e comandos de atribuição, leitura e escrita em Java.

Na linha 3, as três variáveis do programa são declaradas como sendo do tipo `real` (ponto flutuante em Java). Observe o uso dos comandos de escrita nas linhas 4, 6 e 9, e o uso dos comandos de leitura de dados nas linhas 5 e 7. Observe ainda o uso do simbolo = para representar o comando de atribuição nas linhas 5, 7 e 8.

Um outro exemplo está ilustrado na figura 3.6. O programa Java em questão calcula o valor das raízes de uma equação do segundo grau.

```
1.  public class SegundoGrau {
2.     public static void main ( String args[] ) {
3.        double a, b, c, delta, x1 = 0, x2 = 0;
4.        //Em 2X*X+3X-10=0, "a" é o 2, "b" é o 3 e c é o -10.
5.        System.out.print("Digite o valor de a:");
6.        a = System_in.readFloat();
```

```
7.          System.out.print("Digite o valor de b:");
8.          b = System_in.readFloat();
9.          System.out.print("Digite o valor de c:");
10.         c = System_in.readFloat();
11.         delta = ((b*b)-(4*a*c));
12.         x1 = ( ( -b + (Math.sqrt (delta) ) ) / ( 2*a ) );
13.         x2 = ( ( -b + (Math.sqrt (delta) ) ) / ( 2*a ) );
14.         System.out.println("x1 = " + x1);
15.         System.out.println("x2 = " + x2);
16.     }
17. }
```

Figura 3.6 – Programa Java para cálculo das raízes de uma equação de segundo grau.

Esse exemplo acrescenta algumas observações importantes. A linha 4, por exemplo, mostra o uso da sintaxe Java para escrita de comentários sobre trechos de programa. Um comentário, como o próprio nome sugere, é apenas uma observação feita pelo autor do código para facilitar o entendimento do mesmo por parte do leitor do código e em nada interfere no processamento. Ou seja, tudo que vier escrito após o uso de duas barras `//` é ignorado pela JVM ao executar o código.

É possível observar também que todas as variáveis do programa foram definidas com o tipo `double`. Como vimos no capítulo anterior, o tipo `double` tem capacidade de armazenamento de valores maiores que a do tipo `float`, por exemplo. Isso significa que é possível armazenar valores do tipo `float` em variáveis do tipo `double`. Observe que é exatamente isso que fazemos nas linhas 6, 8 e 10, ao utilizarmos o comando `System_in.readFloat()` para ler os valores de a, b e c. O contrário, entretanto não é possível; não podemos armazenar valores `double` em variáveis `float` ou `int`, por exemplo.

Finalmente, observe que as variáveis x1 e x2 foram inicializadas com o valor 0 (zero) exatamente no momento em que foram declaradas, para garantir que não permaneçam com valores indefinidos que potencializam erros de execução futuros.

O programa da figura 3.7 faz uso dos métodos da classe String mostrados na seção 2.6 para mudar o formato de apresentação de uma data fornecida como entrada por um usuário. A data é fornecida no formato, por exemplo, “29/novembro/2008” e é transformada para o formato “29 de novembro de 2008”. O entendimento do funcionamento desse programa é um bom exercício para o aprendizado de manipulação de valores do tipo string.

```
1.  public class Data {
2.     public static void main(String args[]){
3.        String data, dataProcessada, dia, mes, ano;
4.        int inicio, fim;
5.        System.out.print("Informe a data (ex: 01/janeiro/2009):
6.  ");
```

```
7.          data = System_in.readString();
8.          dia = data.substring(0,2);
9.          inicio = data.indexOf('/');
10.         fim = data.indexOf('/', inicio+1);
11.         mes = data.substring(inicio+1,fim);
12.         ano = data.substring(fim+1,data.length());
13.         dataProcessada = dia + " de " + mes + " de " + ano;
14.         System.out.println(dataProcessada);
15.     }
16. }
```

Figura 3.7 – Uso de métodos da classe String de Java.

Na linha 7, a data fornecida pelo usuário é lida em formato de string e atribuída á variável `data`. Na linha 8, o método `substring` copia da variável `data` os seus dois primeiros caracteres e armazena em `dia`. A etapa mais complicada de todo o processamento é copiar o nome do mês da string original. Como os nomes dos meses do ano possuem comprimentos diferentes e como não sabemos qual o mês fornecido pelo usuário, temos que utilizar métodos para identificação correta da posição da primeira letra do nome do mês e da última letra. Isso é feito nas linhas 9 e 10 respectivamente, através da identificação das posições dos dois caraceres `'/'` na string original lida. A variável `mes` recebe a substring compreendida entre esses dois caracteres (linha 11) e a variável `ano` recebe o restante da string (linha 12).

Finalmente, a data processada é escrita na linha 14.

Resumo

- Variável é uma representação lógica de um endereço físico da memória principal do computador;
- Variáveis são representadas e um programa através do uso de um identificador.
- Uma variável só armazena dados de um mesmo tipo (inteiro, real, literal, lógico). Esse tipo é definido estaticamente no momento da declaração da variável.
- Comandos são instruções que ordenam ações do processador.
- Existem vários tipos de comandos. Os comandos de atribuição, de leitura, e de escrita são os mais utilizados nos programas.
- Um comando de atribuição atribui um valor a uma variável, ou seja, atualiza o valor armazenado no endereço de memória representado logicamente pelo identificador de variável.
- Um comando de leitura capta dados fornecidos pelo usuário do programa através

do teclado.

- Um comando de escrita exibe um valor ou resultado da avaliação de uma expressão para que o usuário do programa possa ter conhecimento.
- A linguagem Java implementa todos os três tipos de comandos listados acima.

Na próxima aula...

… veremos diferentes formas de organizar seqüência de comandos para permitir caminhos de computação alternativos ou repetição de ações.

Referências e Sugestões de Leitura

Uso de variáveis em pseudocódigos é discutido no capítulo 2 de

LEITE, M., Técnicas de Programação: uma Abordagem Moderna, BRASPORT, 2006.

Variáveis e comandos em pseudocódigos também apresentados no capítulo 2 de

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

Para um aprofundamento teórico sobre os conceito de variáveis, tipos e comandos, sugiro os capítulos 5, 6 e 7 de

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

e os capítulos 4 e 5 de

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

Para aprender sobre variáveis e comandos em Java, capítulo 2 de

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

Exercícios Propostos

3.1 – Considere duas variáveis X e Y, é possível armazenar o conteúdo de X em Y e o de Y em X? Como?

Elabore algoritmos em pseudocódigo para solucionar os problemas abaixo:

3.2 – A partir do diâmetro de um círculo que será digitado, calcular e exibir sua área.

3.3 – Calcular e exibir o volume de uma esfera a partir do valor de seu diâmetro que será digitado ($V = \pi D3/6$).

3.4 – Calcular e exibir o volume de um cone a partir dos valores da altura e do raio da base que serão digitados ($V = \pi R2h/3$) .

3.5 – Calcular e exibir a média aritmética de quatro valores quaisquer que serão digitados.

3.6 – Sabendo que uma milha marítima equivale a 1852 metros e que um quilômetro possui mil metros, fazer um programa para converter milhas marítimas em quilômetros.

Utilize a linguagem Java para resolver os seguintes problemas abaixo:

3.7 – Calcular e exibir a tensão de um determinado circuito eletrônico a partir dos valores da resistência e corrente elétrica que serão digitados. Utilize a lei de Ohm. ($I = V/R$)

3.8 – Calcular e exibir a velocidade final (em km/h) de um automóvel, a partir dos valores da velocidade inicial (em m/s), da aceleração (m/s^2) e do tempo de percurso (em s) que serão digitados ($V=V0 + A*T$ e $Vkm/h = 3,6 * Vm/s$).

3.9 – Entrar via teclado com dois valores quaisquer “X” e “Y”. Calcular e exibir o cálculo X^Y (“X” elevado a “Y”). [*Nota: pesquisar sobre a função Math.pow(...,...) de Java*].

3.10 – Entrar via teclado com o valor de cinco produtos. Após as entradas, calcular e exibir o valor total. Depois deve-se digitar o valor do pagamento e exibir o troco que deverá ser devolvido.

3.11 – Entrar via teclado com o valor da cotação do dólar atual e uma certa quantidade de dólares. Calcular e exibir o valor correspondente em Reais (R$).

3.12 – Tomando 3(três) palavras, com 4(quatro) letras cada, como segue:

CAMA

ASAS

LARA

É possível obter vocábulos tanto no sentido horizontal quanto no vertical. Escreva um programa para ler as três palavras (horizontais) e exibir os vocábulos dispostos na vertical.