1 Introdução

Muitas vezes nossos programas possuem trechos de código cuja execução deve ser repetida por um número determinado de vezes ou mesmo de maneira indefinida. Considere, por exemplo, um programa que permite a um professor informar as notas de seus alunos, uma a uma, a fim de calcular a média de cada um e, talvez, ao final, calcular também a média da turma. O programa deverá repetir a execução da entrada de dados. O que resta saber é como isso pode ser feito, já que copiar e colar código não é uma boa prática e pode haver casos em que o número de alunos sequer é conhecido.

2 Estruturas de repetição e suas representações

A linguagem de programação Java possui três estruturas de repetição: **while**, **for** e **do/while**. Como ocorre com as demais estruturas, elas podem ser representadas de várias formas diferentes. Vamos ilustrar seu uso por meio de um programa que permite ao professor informar a nota de 5 alunos, calculando a média de cada um e exibindo.

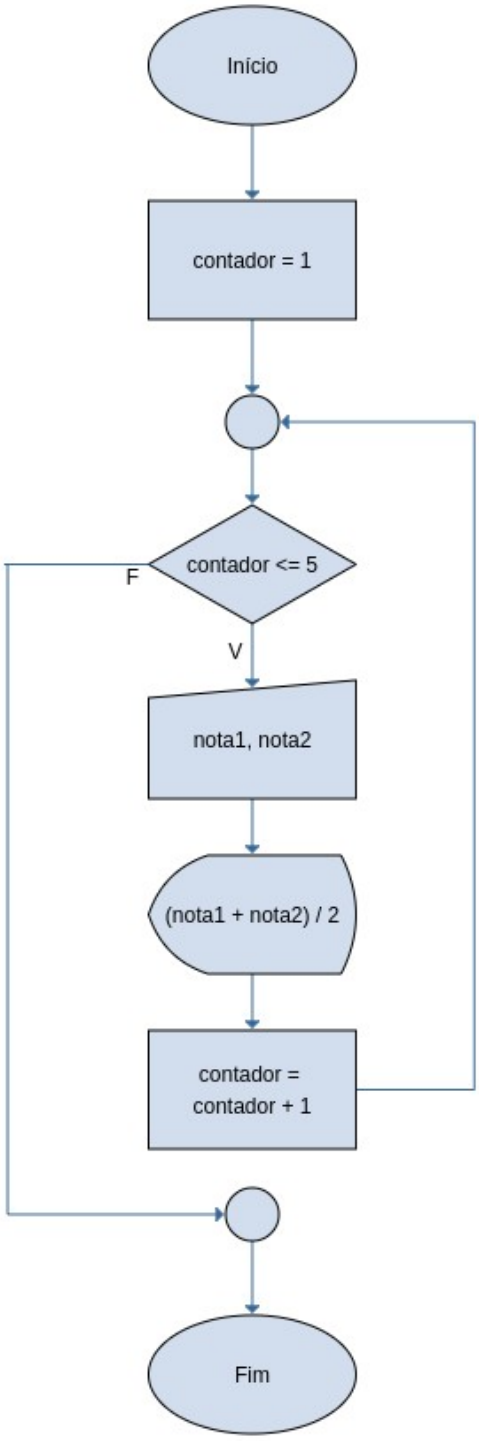
2.1 (A estrutura while) A representação por pseudocódigo da estrutura de repetição **while** é exibida na Listagem 2.1.1.

Listagem 2.1.1

```
var nota1, nota2: real;
Início
   var contador: inteiro;
   contador = 1;
   Enquanto contador <= 10 Faça
      Ler nota1;
      Ler nota2;
      Escrever (nota1 + nota2)/2;
      contador = contador + 1;
   Fim Enquanto
Fim
```

O mesmo programa pode ser representado por um fluxograma, como na Figura 2.1.1.

Figura 2.1.1



2.2 (A estrutura for) A estrutura for também pode ser utilizada para implementar o algoritmo de exemplo. Veja sua representação como pseudocódigo na Listagem 2.2.1. Note que ela é muito parecida com a estrutura **while**. Ocorre que as três partes de controle são especificadas de uma única vez em uma, potencialmente simplificando a leitura do código.

Listagem 2.2.1

```
var nota1, nota2: real;
Início
   Para contador = 1 Até 5 Faça
      Ler nota1;
      Ler nota2;
      Escrever (nota1 + nota2) / 2;
   Fim Para
Fim
```

O fluxograma da estrutura **for** é análogo ao da estrutura **while.**

Importante: Note que ambas estruturas for e while têm uma característica em comum: elas possuem um bloco a ser repetido e uma condição que precisa ser verdadeira para que o bloco repita mais uma vez. Ocorre que essa condição, em ambos os casos, é avaliada antes de o bloco ser executado. Isso quer dizer que, caso a condição seja avaliada como falsa logo de cara, o bloco não vai ser repetido nenhuma vez.

Isso é importante pois é o principal aspecto que difere as estruturas **for** e **while** da estrutura **do/while**. A estrutura **do/while** também possui um bloco de código a repetir e uma condição que precisa ser avaliada como verdadeira para que a repetição ocorra. Porém, o bloco é executado uma primeira vez e, somente depois disso a condição é avaliada.

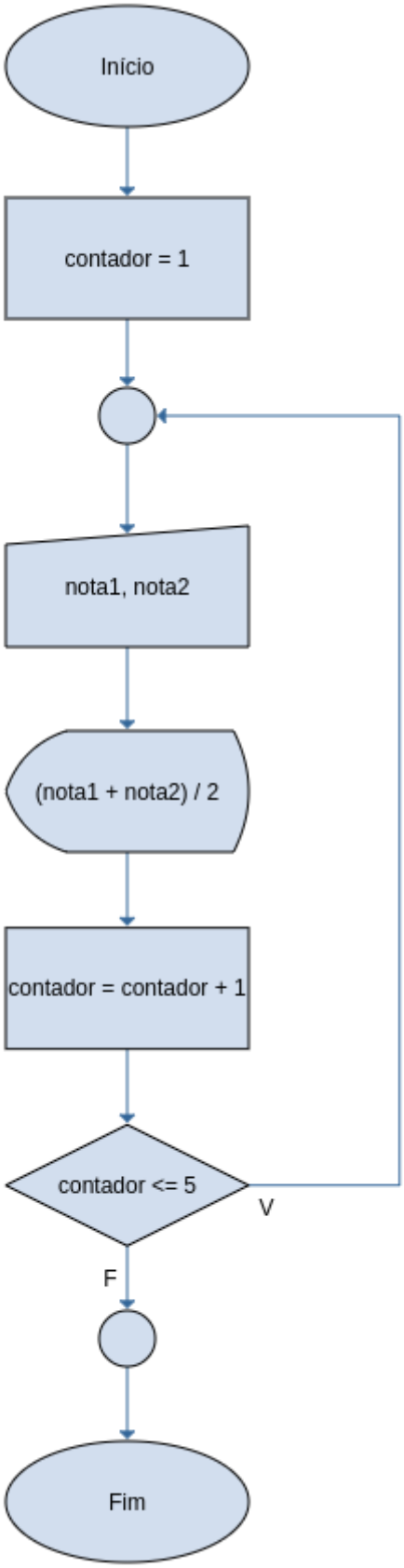
2.3 (A estrutura do/while) A Listagem 2.3.1 mostra o programa de exemplo implementado utilizando-se uma estrutura do tipo **do/while**.

Listagem 2.3.1

```
var: nota1, nota2: real;
var: contador: inteiro;
Início
   contador = 1;
   Faça
      Ler nota1;
      Ler nota2;
      Escrever (nota1 + nota2) / 2;
      contador = contador + 1
    Enquanto contador <= 5
Fim
```

A Figura 2.3.1 mostra a implementação do algoritmo como um fluxograma.

Figura 2.3.1



3 Implementação em Java

Como mencionado, a linguagem Java possui as três estruturas de repetição descritas. É importante observar que tudo o que pode ser feito com uma delas pode ser feito com qualquer outra, eventualmente ajustando alguns poucos detalhes. A seguir, vemos o problema de exemplo implementado utilizando cada uma delas.

3.1 A Listagem 3.1.1 mostra o programa de exemplo implementado em Java, utilizando a estrutura **while**.

Listagem 3.1.1

```
import javax.swing.JOptionPane;
public class ExemploWhile {
    public static void main(String[] args) {
        double nota1, nota2;
        int contador = 1;
        while (contador <= 5) {
            nota1 =
Double.parseDouble(JOptionPane.showInputDialog("Digite a nota 1"));
            nota2 =
Double.parseDouble(JOptionPane.showInputDialog("Digite a nota 2"));
            JOptionPane.showMessageDialog(null, "Média: " + (nota1 +
nota2) / 2);
            contador = contador + 1;
        }
    }
}
```

3.2 A Listagem 3.1.2 mostra como implementar o programa exemplo usando uma estrutura **for**.

Listagem 3.1.2

```
public class ExemploFor {
    public static void main(String[] args) {
        double nota1, nota2;
        for (int contador = 1; contador <= 5; contador = contador +
1) {
            nota1 =
Double.parseDouble(JOptionPane.showInputDialog("Digite a nota 1"));
            nota2 =
Double.parseDouble(JOptionPane.showInputDialog("Digite a nota 2"));
            JOptionPane.showMessageDialog(null, "Média: " + (nota1 +
nota2) / 2);
            contador = contador + 1;
        }
    }
}
```

3.3 A estrutura **do/while** é ilustrada na Listagem 3.3.1.

Listagem 3.3.1

```
import javax.swing.JOptionPane;
public class ExemploDoWhile {
    public static void main(String[] args) {
        double nota1, nota2;
        int contador = 1;
        do {
            nota1 =
Double.parseDouble(JOptionPane.showInputDialog("Digite a nota
1"));
            nota2 =
Double.parseDouble(JOptionPane.showInputDialog("Digite a nota
2"));
            JOptionPane.showMessageDialog(null, "Média: " +
((nota1 + nota2) / 2));
            contador = contador + 1;
        }while (contador <= 5);
    }
}
```

Exercícios

1. Escreva um programa que exibe os primeiros 100 números naturais. Faça versões usando for, while e do/while.

2. Reimplemente o exercício 1, fazendo com que os valores sejam exibidos em ordem reversa.

3. Escreva um programa que calcula o fatorial de um valor natural digitado pelo usuário.

4. Escreva um programa que verifica se um número natural digitado pelo usuário é primo.

Referências

DEITEL, P. e DEITEL, H. **Java Como Programar**. 8ª Edição. São Paulo, SP: Pearson, 2010.

LOPES, A. e GARCIA, G. **Introdução à Programação – 500 Algoritmos Resolvidos**. 1ª Edição. São Paulo, SP: Elsevier, 2002.