ALGORITMOS SEQUENCIAIS

E-Book

Vamos retomar os blocos básicos de programação. Esses blocos contêm três elementos: entrada, processamento e saída e, claro, têm as variáveis que armazenarão as informações que serão processadas durante sua execução.

Sempre podemos utilizar um exemplo para esclarecer cada uma das etapas. Considere o problema a seguir.

> Dadas as dimensões de um retângulo, calcular e mostrar a sua área.

Para construir um algoritmo que resolve este problema, podemos começar pelas **variáveis** necessárias.

```
valores reais altura, base, area.
```

Agora precisamos das informações que são necessárias para o funcionamento do algoritmo: **as entradas**. Nesse caso serão fornecidas pelo usuário.

```
ler (base, altura).
```

O **processamento** é o cálculo da área pedida.

```
área = base * altura
```

Finalmente, a **saída.** Vamos exibir o resultado para o usuário.

```
escrever (área)
```

Exemplo

Veja esse exemplo:

O Sr. João das Cores precisa saber a quantidade de latas de tinta necessárias para pintar um tanque cilíndrico de combustíveis, pois o cliente pediu um orçamento para o serviço. Ele fez uma pesquisa de mercado e o melhor preço que encontrou foi R$ 50,00 para uma lata que cobre 3 metros quadrados.

Ele pediu para o cliente passar o raio da base do tanque (r) e a sua altura (h) (--> as entradas); as contas que ele fez foram (-->processamento) area = 2.pi.r2 + 2.pi.r.h, latas = area/3, custo = latas * 50; depois ele levou o orçamento para o cliente e mostrou (-->saída).

Os algoritmos foram representados até agora como pseudocódigo, como jogo, mas eles poderiam ser escritos em qualquer linguagem de programação.

Veja um exemplo completo escrito na Linguagem Java. Vale a pena uma boa leitura das páginas 36 a 43.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/39590

DEITEL, P.; DEITEL, H. Java: como programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017.

Estruturas de Controle

Para construir uma estrutura de controle, precisamos de expressões lógicas, cujos valores podem ser *verdadeiro* o u *falso*. Essas expressões são formadas por operadores lógicos e relacionais.

Na linguagem Java, esses operadores são escritos conforme mostra a Tabela 2.14, na página 44.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/39590

DEITEL, P.; DEITEL, H. Java: como programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017.

Estruturas de Seleção

Uma estrutura de seleção permite ou não a execução de um bloco de ações, de acordo com uma expressão lógica construída com operadores aritméticos, relacionais e lógicos.

Tais estruturas podem ser simples ou compostas. Vamos estudar cada uma delas!

Seleção Simples

É utilizada quando precisamos testar uma única condição. O bloco a seguir mostra como é essa estrutura.

```
se (expressão lógica) {
     instruções para valor verdadeiro
}
senão {
     instruções para valor falso
}
```

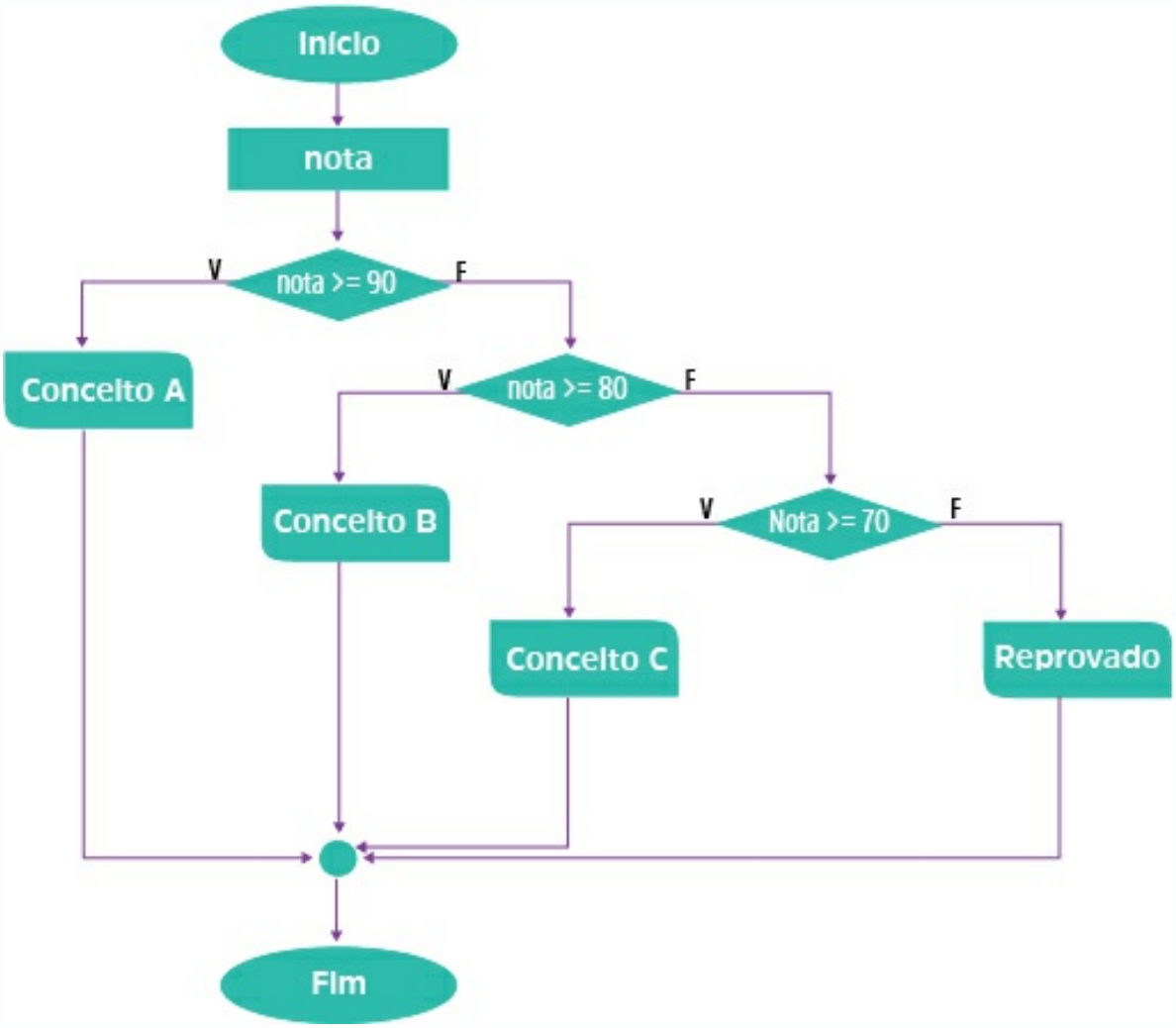
Seleção Composta

Essa estrutura é utilizada quando existem várias condições que precisam ser avaliadas para execução dos blocos de programação. De modo a facilitar o entendimento, vamos ver um exemplo.

> Ler a nota de um aluno e atribuir-lhe um conceito conforme a regra:
>
> null

A Figura 2 mostra um fluxograma para a solução desse problema. A composição utilizada é do tipo **encadeada**, que funciona adequadamente para várias possibilidades para uma única condição.

FIGURA 1 - Exemplo seleção composta encadeada



Fonte: Elaboração do autor, 2020.

A listagem a seguir mostra o pseudocódigo para a solução proposta.

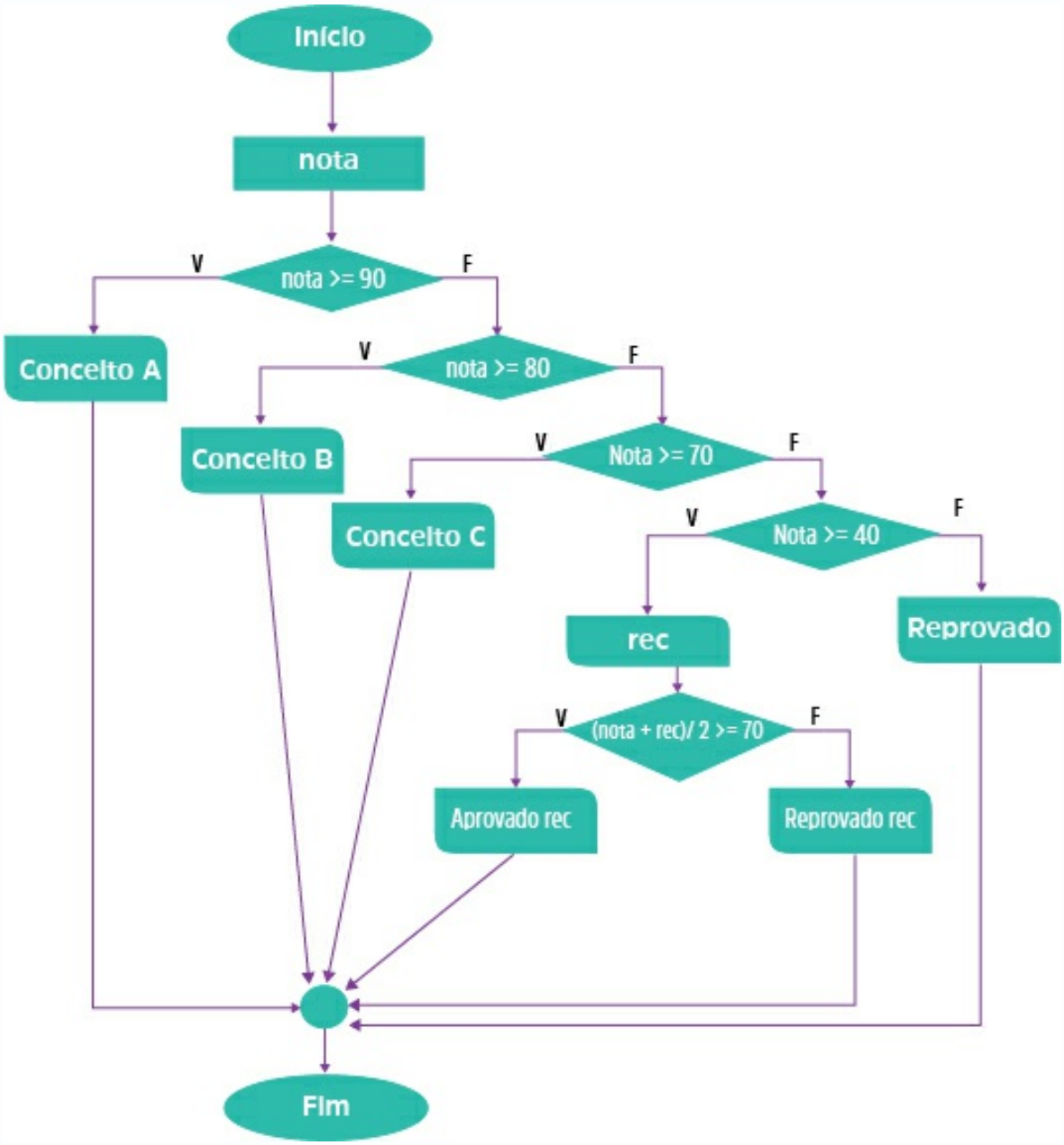
```
valor real nota;
início
     ler nota;
     se nota maior ou igual a 90
               escrever “Conceito A”
    senão se nota maior ou igual a 80
               escrever “Conceito B”
    senão se nota maior ou igual a 70
               escrever “Conceito C”
    senão
               escrever “Reprovado”
    fim se
fim
```

Também é possível compor as condições de forma **aninhada.** Ela é adequada para quando existirem condições que dependem umas das outras. Veja o exemplo anterior modificado.

> Ler a nota de um aluno e atribuir-lhe um conceito conforme a regra anterior, agora com a possibilidade de fazer uma avaliação extra, desde que sua nota seja no mínimo 40.

O fluxograma de uma possível solução é apresentado na Figura 3.

FIGURA 2 - Exemplo seleção composta aninhada



Fonte: Elaboração do autor, 2020.

A listagem do pseudocódigo correspondente é dada a seguir. Note que o exemplo apresenta tanto o encadeamento quanto aninhamento.

```
valores reais nota, rec;

início
```

```
     ler nota;
     se nota maior ou igual a 90
               escrever “Conceito A”
     senão se nota maior ou igual a 80
               escrever “Conceito B”
     senão se nota maior ou igual a 70
               escrever “Conceito C”
     senão
               se null
                    ler rec
                    se null
                         escrever “Aprovado Rec”
                    senão
                         escrever "Reprovado"
                    fim se
               senão
                    escrever "Reprovado"
               fim se
     fim se
fim
```

Você pode ver a implementação em Java dessas estruturas, com comentários e dicas importantes nas páginas de 81 a 88 da bibliografia sugerida.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/39590

DEITEL, P; DEITEL, H. Java: como programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017.

Além do comando *se - senão* disponível em várias linguagens de programação, existem outros comandos que permitem a seleção de blocos.

Estrutura de seleção composta do tipo *avalie - caso*

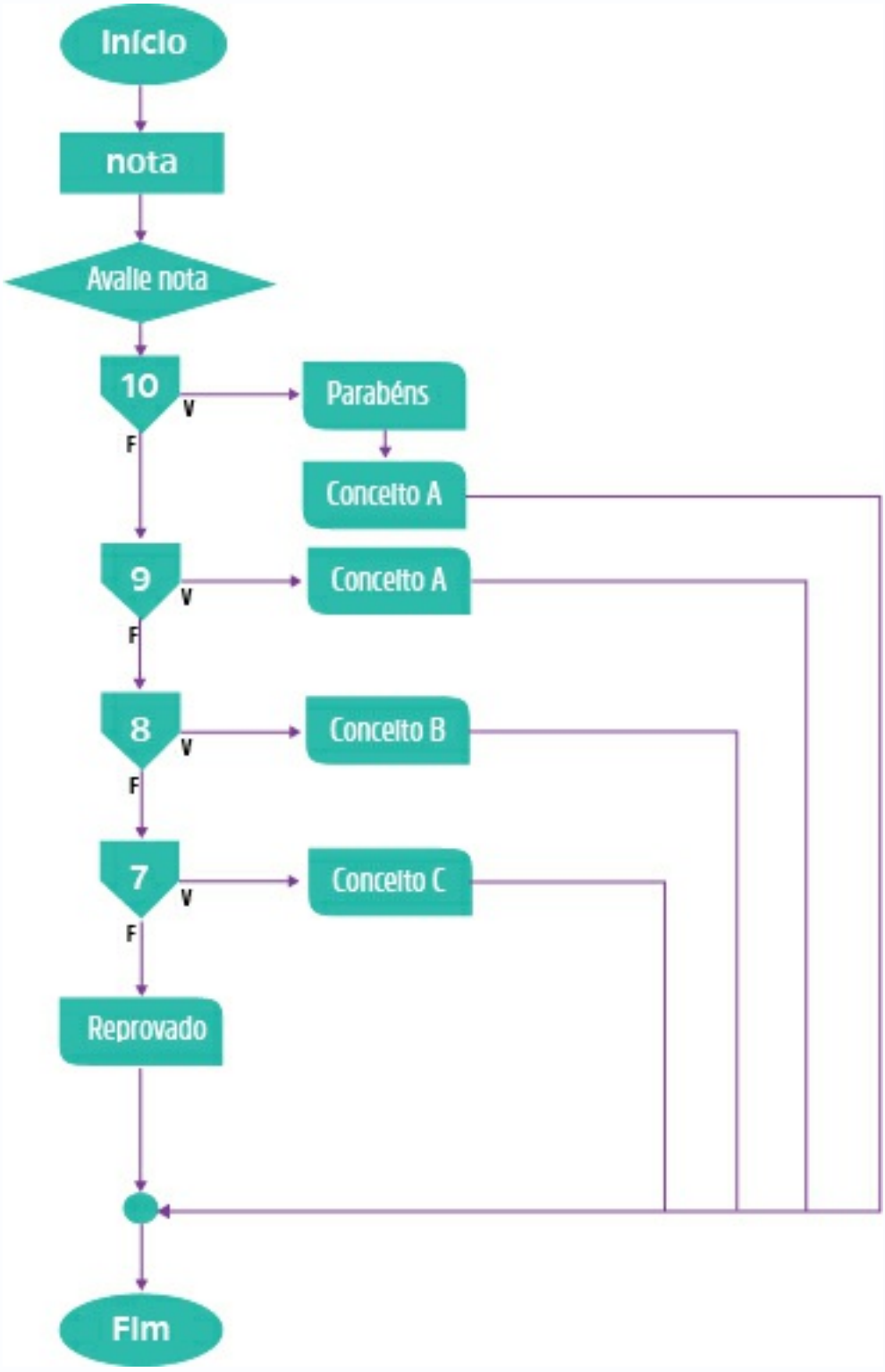
Esse tipo de estrutura permite a realização de diversas ações com base nos possíveis valores de uma expressão. A diferença é que esses valores não são lógicos e sim um valor contável, como um inteiro ou um caractere. Algumas linguagens já implementam outros tipos, como veremos na bibliografia sugerida.

A listagem a seguir mostra uma variação do exemplo estudado.

```
inteiro nota;
início
    ler nota
    avalie nota:
        caso 10:
            escrever "Parabéns!"
            escrever "Conceito A"
            pare
        caso 9:
            escrever "Conceito A"
            pare
        caso 8:
            escrever "Conceito B"
            pare
        caso 7:
            escrever "Conceito C"
            pare
        caso contrário:
            escrever "Reprovado"
    fim avalie
fim
```

A importância da instrução **pare** deve-se ao fato da lógica de implementação da estrutura **avalie - caso** ser uma espécie de lógica em queda aplicada à sua execução: quando um **caso** é escolhido, a execução começa ali e continua até uma de duas coisas acontecer: ou uma instrução pare ser encontrada ou o **fim** da estrutura ser encontrado. A Figura 4 mostra o fluxograma para esta estrutura.

FIGURA 3 - Fluxograma da estrutura de seleção avalie - caso



Fonte: Elaboração do autor, 2020.

As páginas 130 e 131 mostram a implementação dessa estrutura em Java.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/39590

DEITEL, P.; DEITEL, H. Java Como Programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017.

O operador ternário recebe este nome porque é o único operador em grande parte das linguagens, incluindo o Java, que aceita três operandos. Sua sintaxe é dada por:

```
<expressão booleana="">? <expressão 1="">: <expressão 2="">;
            </expressão></expressão></expressão>
```

sendo:

- o primeiro operando deve ser uma *expressão booleana;*
- o segundo e o terceiro operandos podem ser qualquer *expressão* que retorne algum valor.

A construção ternária retorna *expressão 1* como saída, se o primeiro operando for avaliado como *verdadeiro*, ou *expressão 2* caso contrário.

Você Sabia?

Você pode substituir várias linhas de código de controle if-else por uma única linha de código usando o operador ternário. Isso torna seu código mais legível. No entanto, não exagere.

Estruturas de repetição

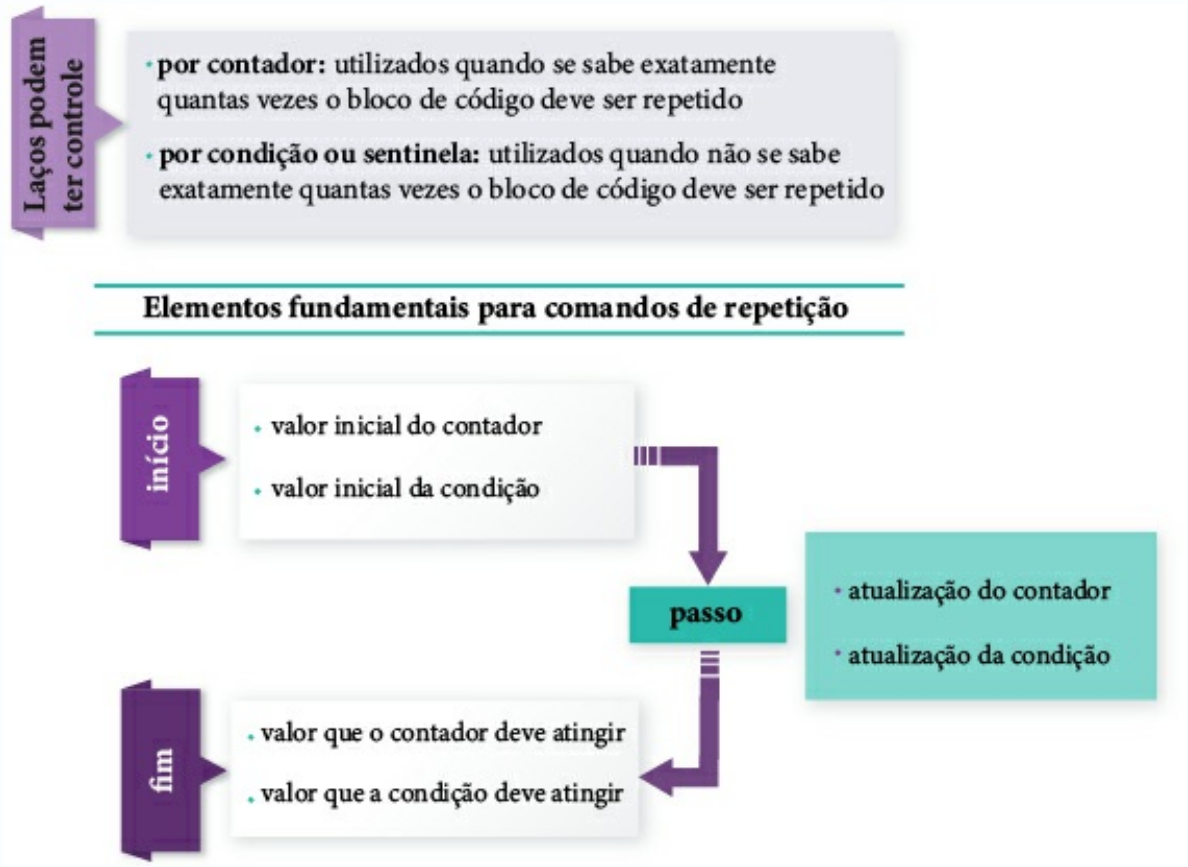
Muitas vezes nossos programas possuem blocos de código cuja execução deve ser repetida por um número determinado de vezes ou mesmo de maneira indefinida, de acordo com alguma condição.

É importante dizer que o número de repetições pode ser indeterminado, mas é necessariamente **finito**.

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 trecho do programa que deverá ser repetido vai desde a entrada de dados até o cálculo da média de cada um. O que resta saber é como isso pode ser feito, já que copiar e colar código não é uma boa prática, ainda mais que o número de alunos pode variar para cada turma.

Cada linguagem de programação oferece vários comandos de repetição, cada um com determinadas características de controle. A Figura 5 mostra um esquema que vai ajudar você a entender o funcionamento dos comandos de repetição: controle do número de vezes que o bloco de código pode ser repetido, os três elementos fundamentais e como tudo isso é integrado.

FIGURA 4 - Infográfico



Fonte: Elaboração do autor, 2020.

As expressões de controle para os comandos de repetição podem utilizar todos os operadores vistos até agora:

- aritméticos;
- relacionais;
- lógicos.

No entanto, podemos utilizar novas expressões: os **acumuladores**. Esses acumuladores podem ser utilizados como contadores dos laços contados, ou como o próprio nome já diz, para acumular valores que serão utilizados ao longo do programa.

Uma **variável acumulador** é qualquer variável que recebe um valor inicial e é incrementada nas iterações de um laço por uma expressão, ocorrendo em ambos os lados de uma atribuição.

Cada linguagem de programação permite escrever acumuladores de formas diferentes. Veja como escrever na linguagem Java nas páginas 101 a 104.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/39590

DEITEL, P.; DEITEL, H. Java Como Programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017.

Outra característica importante das estruturas de repetição é o momento do **teste de continuidade**. Algumas fazem teste no início, outras no final, o que permite que o desenvolvedor obtenha soluções adequadas para determinados problemas computacionais.

Atenção

Não é comum, mas algumas linguagens de programação possuem comandos de repetição que realizam **testes de parada** ao invés de testes de continuidade.

Estruturas com teste no início

São estruturas que permitem repetir **nenhuma** ou **diversas** vezes um bloco de código, sempre verificando antes de cada execução se essa deve ser realizada.

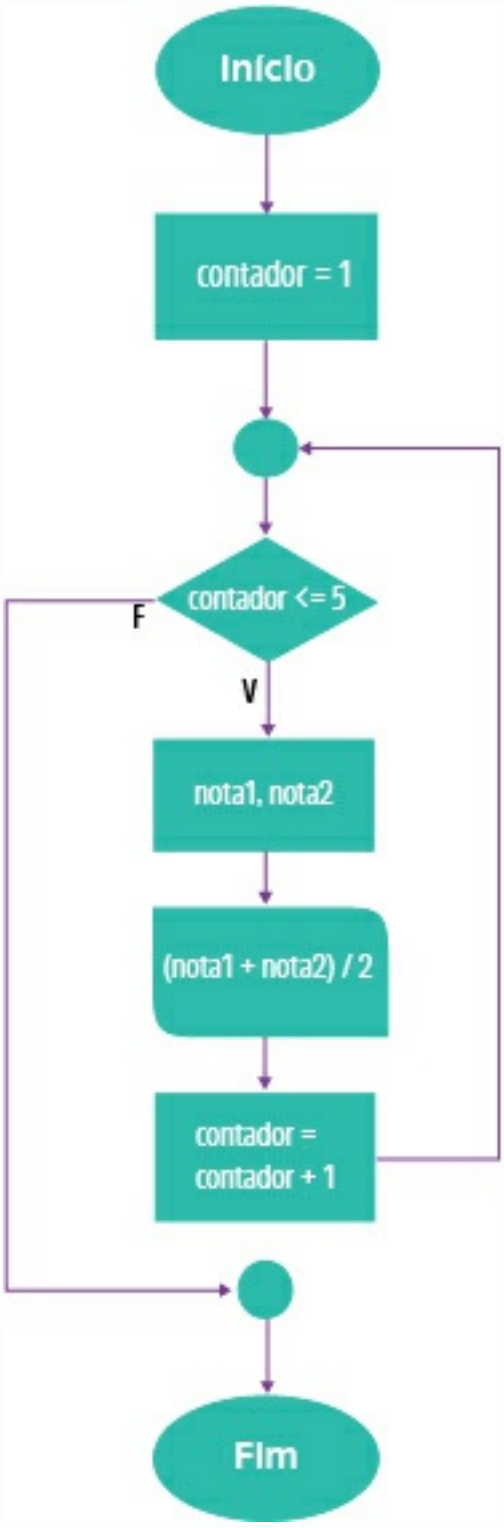
Numa estrutura com teste de **continuidade** n o **início**, um bloco de programa é repetido **enquanto** o teste resultar o valor lógico verdadeiro. Note que tal estrutura pode ser aplicada tanto para laços contados quanto para laços condicionais.

A estrutura enquanto: Vamos ilustrar seu uso por meio de um programa que permite ao professor informar a nota de 5 alunos, calcular a média de cada um e exibir. O pseudocódigo a seguir ilustra o seu uso.

```
real nota1, nota2;
início
     inteiro contador = 1;
     enquanto contador <= 5 faça
           ler nota1, nota2
           escrever (nota1 + nota2)/2
           contador = contador + 1
     fim enquanto
fim
```

Confronte o pseudocódigo com o respectivo fluxograma (Figura 6), esse último mostra a dinâmica de funcionamento da estrutura de repetição. Perceba o retorno do código para o teste no início do bloco.

FIGURA 5 - Fluxograma: retorno do código



Fonte: Elaboração do autor, 2020.

A estrutura para: Ela também pode ser utilizada para implementar o algoritmo de exemplo. Veja sua representação como pseudocódigo a seguir e note que ela é muito parecida com a estrutura **enquanto**. Ocorre que as três partes de controle são especificadas de uma única vez em uma, potencialmente simplificando a leitura do código.

```
real nota1, nota2;
início
     inteiro contador = 1;
     para contador de 1 até 5 faça
          ler nota1, nota2
          escrever (nota1 + nota2)/2
     fim para
fim
```

Em Java, temos esses comandos que implementam as duas estruturas para laços com teste no início. Embora ambos funcionem tanto para laços contados quanto para laços condicionais, cada um deles fica melhor num caso específico. As páginas 89 a 98 trazem detalhes do comando while (estrutura enquanto). Veja também as páginas de 121 a 128 para o comando for (para).

https://plataforma.bvirtual.com.br/Acervo/Publicacao/39590

DEITEL, P.; DEITEL, H. Java Como Programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017.

Estruturas com teste no final

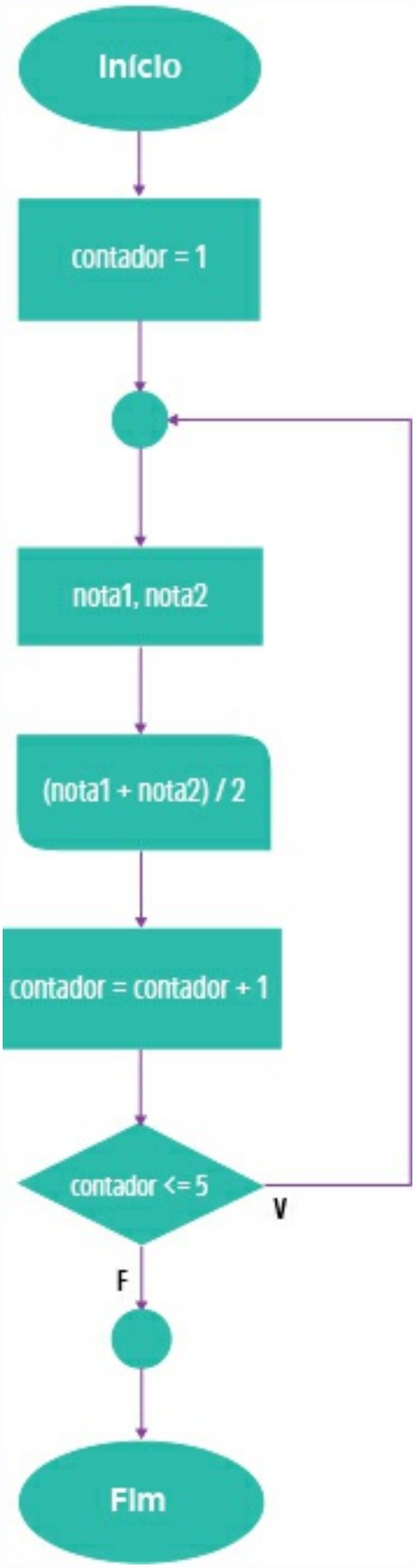
Numa estrutura com teste de **continuidade** n o **final**, um bloco de programa é repetido **enquanto** o teste resultar o valor lógico verdadeiro, depois da sua execução. Isso significa que o bloco será executado **pelo menos uma** o u **diversas vezes**. Note que esta estrutura também pode ser aplicada tanto para laços contados quanto para laços condicionais.

A estrutura faça-enquanto: Vamos ilustrar seu uso com o mesmo exemplo anterior, veja o pseudocódigo a seguir.

```
real nota1, nota2;
início
     inteiro contador = 1;
     faça
          ler nota1, nota2
          escrever (nota1 + nota2)/2
          contador = contador + 1
     enquanto null
fim
```

O fluxograma exibido na Figura 7 mostra claramente que a execução do bloco é realizada antes do teste.

FIGURA 6 - Fluxograma faça-enquanto



Fonte: Elaboração do autor, 2020.

Curioso para ver a implementação na linguagem Java? Veja as páginas 128 e 129.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/39590

DEITEL, P.; DEITEL, H. Java Como Programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017.