

NÃO PODE FALTAR

▲  
Imprimir

# CONCEITOS DE ALGORITMOS E PROGRAMAÇÃO

Marcio Aparecido Artero

## INTRODUÇÃO AOS ALGORITMOS

Você estudará os conceitos introdutórios relacionados a algoritmos e linguagens de programação.

Ver anotações



Fonte: Shutterstock.

**Deseja ouvir este material?**

Áudio disponível no material digital.

## CONVITE AO ESTUDO

Iniciamos aqui a primeira unidade do livro *Algoritmos e Programação Estruturada*. Aproveite ao máximo o conteúdo que nele será desenvolvido, o qual proporcionará a você a oportunidade de ser um excelente profissional.

É certo que a tecnologia é fascinante e com ela aprimoramos técnicas e elevamos o nível de conhecimento para solucionar os mais diversos tipos de problemas. Nada melhor que conhecer e compreender o que são os algoritmos, as linguagens de programação e a estrutura de um programa de computador e, assim, caminhar para a execução das mais diversas tarefas computacionais.

Nesta unidade, você terá o prazer de conhecer uma empresa de tecnologia de informação cujo foco principal é o desenvolvimento de softwares para instituições de ensino. Com a grande demanda de negócios no setor educacional, a empresa criou um projeto para contratação de estagiários para trabalhar com programação e atender à demanda de mercado. Você, um profissional exemplar na área de tecnologia da informação, foi incumbido de treinar os estagiários.

A empresa não exigiu experiência para os candidatos, e por esse motivo você deverá apresentar as definições e aplicações dos algoritmos, as inovações e os diferentes paradigmas para a área de programação, além dos componentes e das estruturas utilizadas na linguagem de programação C.

Após esse treinamento inicial, o estagiário orientado por você terá a oportunidade de saber reconhecer os conceitos e a estrutura dos algoritmos e das linguagens de programação.

Para fazer valer a proposta desta unidade, na primeira seção você terá a oportunidade de estudar os conceitos e a introdução aos algoritmos e das linguagens de programação, assim como os componentes de um programa de computador. Na segunda seção, serão apresentados a você os componentes e elementos da linguagem de programação C, e você terá a oportunidade de

aprender a respeito de variáveis, um conceito fundamental para a prática de programação. Na terceira seção, serão apresentadas as operações e expressões para um programa de computador.

## PRATICAR PARA APRENDER

Caro aluno, algoritmo é uma sequência finita de passos que podem levar à criação e execução de determinada tarefa com a intenção de resolver um problema (FORBELLONE; EBERSPÄCHER, 2005). Assim, é necessário entender as definições de um algoritmo, suas aplicações e seus tipos antes de avançar para os próximos níveis deste material.

A fim de colocarmos os conhecimentos a serem aprendidos em prática, vamos analisar a seguinte situação-problema: você inicia agora o seu trabalho em uma empresa responsável por criar um software de locação de filmes on-line. Você foi incumbido de criar uma funcionalidade para o software, que consiste em detectar se um filme pode ou não ser locado pelo cliente, com base em sua idade e na classificação indicativa do filme.

Antes de partir para a ação e implementar a funcionalidade em uma linguagem de programação, construa um algoritmo que receba como entradas a idade do cliente e a classificação indicativa dos filmes que ele pretende locar. Logo após, o programa deve processar essas informações e mostrar na tela do computador um dos possíveis resultados: “Este filme não é indicado para sua faixa etária” ou “Este filme é indicado para sua faixa etária”. O algoritmo deverá ser elaborado nas formas descritas a seguir:

- Linguagem natural.
- Diagrama de blocos (fluxograma).
- Pseudocódigo.

---

## CONCEITO-CHAVE

o

Ver anotações

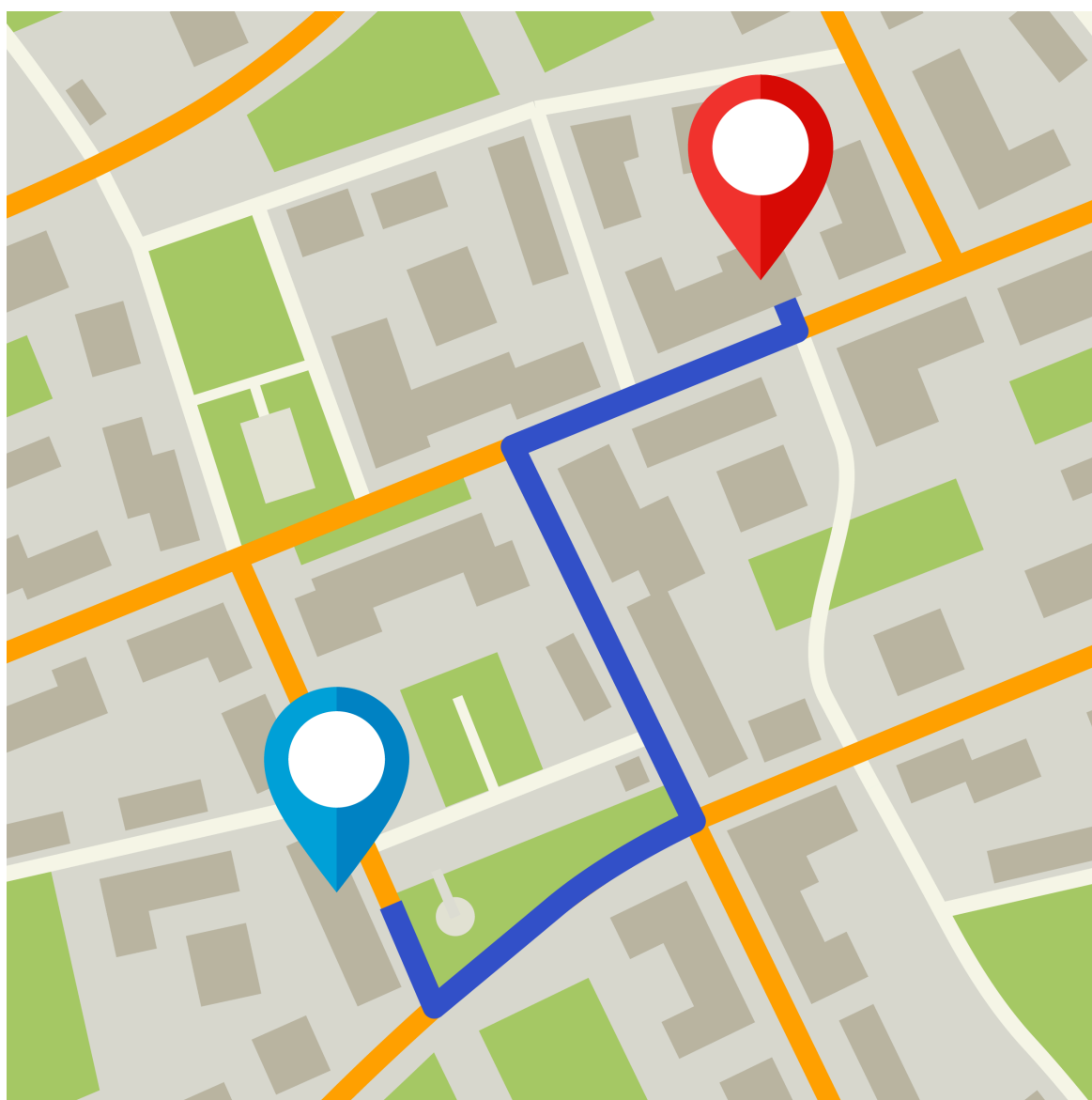
Você já deve ter se deparado muitas vezes com situações em que se deve realizar uma escolha e tomar uma decisão para resolver determinado problema. Por exemplo, ao ir ao mercado e encontrar duas marcas de um mesmo produto, qual produto você escolhe? O produto com o menor preço? O produto que você já está habituado a comprar? Ou um produto mais caro, importado? São caminhos distintos e, dependendo da escolha, você terá um impacto no orçamento.

Além disso, para executar uma ação, muitas vezes é necessário seguir uma sequência de passos para chegar a um resultado. Em uma receita de um prato, por exemplo, você tem uma sequência de passos a seguir para obter o resultado e, caso você mude algo, pode chegar a um objetivo diferente. No contexto computacional funciona de forma similar: ao elaborar um programa, você deve antes elaborar o algoritmo, que nada mais é do que uma sequência de passos necessária para alcançar o objetivo de um programa.

Quando você utiliza algum aplicativo de navegação por GPS (os mais conhecidos são Google Maps e Waze), você insere o destino final e o aplicativo faz a escolha da melhor rota, de acordo com menor congestionamento, menor caminho ou menor tempo de viagem, e você pode configurar qual das opções você deseja – por trás disso há um algoritmo com todas essas alternativas.

A Figura 1.1 ilustra um exemplo de aplicação desse tipo de abordagem. Imagine que você está no ponto azul e quer chegar ao vermelho: você terá algumas alternativas de rota, e o programa fará a escolha conforme os critérios estabelecidos por você.

Figura 1.1 | Exemplo de aplicação de algoritmos



Fonte: Shutterstock.

A partir de agora você vai desmistificar como funcionam os algoritmos e quais são as suas aplicações dentro da programação. Para isso, você conhecerá conceitos, aplicações e tipos de algoritmos.

## CONCEITOS, APLICAÇÕES E TIPOS DE ALGORITMOS

Berg e Figueiró (1998) descrevem algoritmos como uma sequência ordenada de passos que deve ser seguida para atingir um objetivo. Nesse sentido, os algoritmos nortearão você a descobrir qual o **melhor percurso para solucionar um problema computacional**. A elaboração de algoritmos é um passo importante para o desenvolvimento de um programa de computador (ou software), pois, com base na construção de algoritmos para a resolução de um problema, é possível traduzir o algoritmo para alguma linguagem de programação.

Conforme mencionado, para qualquer tarefa a ser executada no dia a dia podemos desenvolver um algoritmo. Como exemplo, tomemos a sequência de passos para o cozimento de arroz, que pode ser a seguinte:

1. Acender o fogo; 2. Refogar os temperos; 3. Colocar o arroz na panela;
4. Colocar a água; 5. Abaixar o fogo; 6. Esperar o ponto; 7. Desligar o fogo;
8. Servir o arroz.

Podemos, ainda, criar um algoritmo um pouco mais detalhado para preparar o cozimento do arroz:

1. Comprar o arroz; 2. Analisar a qualidade;
3. Realizar a pré-seleção para o cozimento; 4. Preparar os temperos;
5. Pegar a panela; 6. Acender o fogo;
7. Colocar os temperos na panela para refogar; 8. Adicionar o arroz;
9. Colocar a água na medida considerada ideal para a quantidade de arroz colocada;
10. Baixar o fogo; 11. Fechar a panela com a tampa; 12. Aguardar o ponto;
13. Desligar o fogo; 14. Servir o arroz.

Observe que não existe uma única forma de elaborar um algoritmo, porém, existe uma sequência lógica para a execução da tarefa. O passo 8, “Adicionar o arroz”, só pode ser realizado após pegar a panela (passo 5). Todavia, podem-se criar outras formas e sequências para alcançar o mesmo objetivo.

Para melhor entendimento dos algoritmos é necessário dividi-los em três partes:

- **Entrada:** dados de entrada do algoritmo. No caso do algoritmo para cozimento do arroz, seriam os insumos (ingredientes) necessários para o preparo do arroz.
- **Processamento:** são os procedimentos necessários para chegar ao resultado final (o cozimento do arroz).

o

Ver anotações



- **Saída:** resultado ao qual o algoritmo quer chegar após o processamento dos dados de entrada (arroz pronto para ser servido).

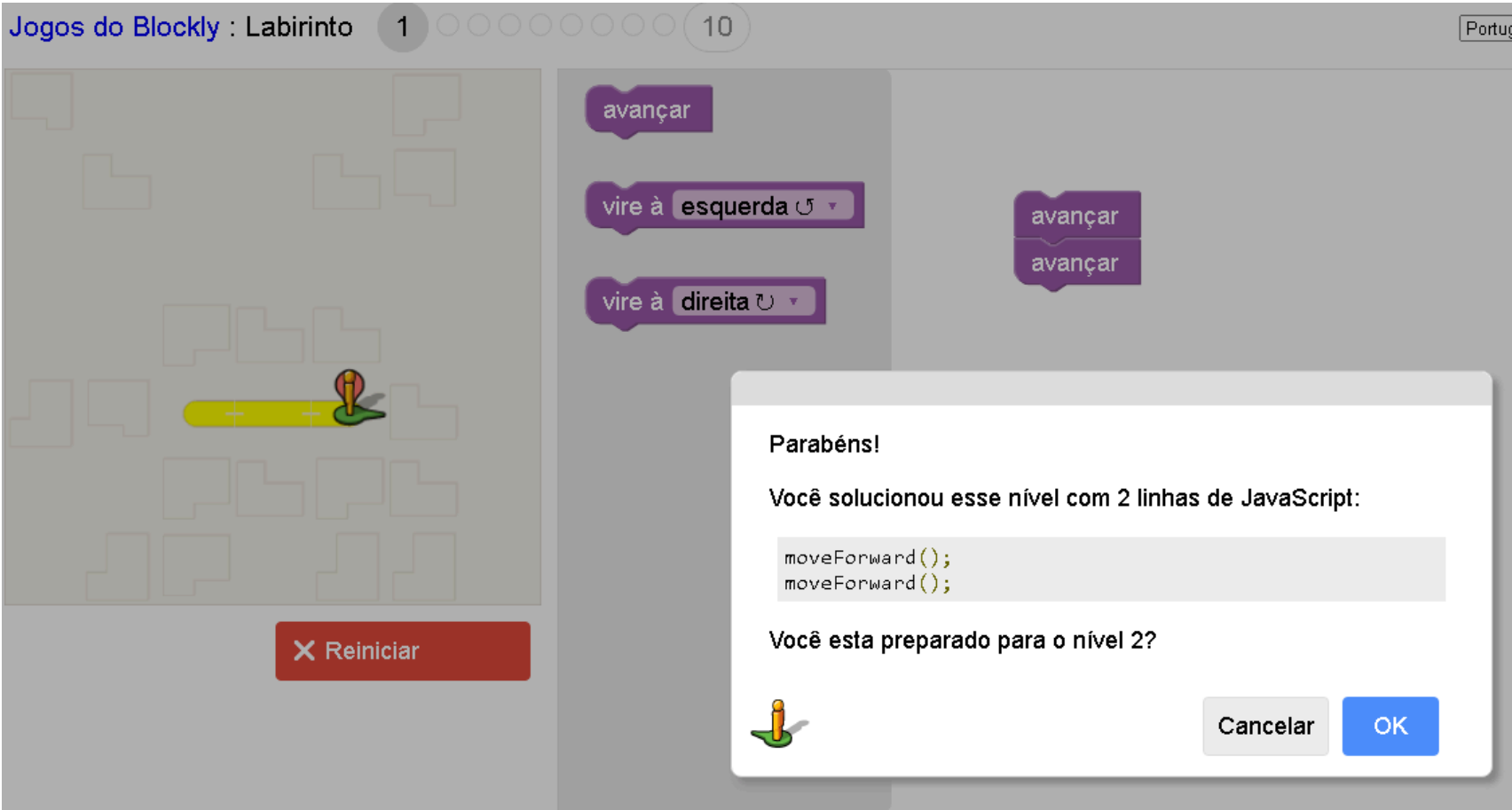
A seguir, você vai entender o funcionamento dos algoritmos usando a linguagem natural, os diagramas de blocos (em algumas literaturas são conhecidos como fluxograma) e os pseudocódigos.

EXEMPLIFICANDO

Blockly Games é um software gratuito, composto por um conjunto de jogos educacionais com enfoque no ensino de programação (BLOCKLY GAMES, 2020).

A Figura 1.2 ilustra um exemplo de jogo no qual o aluno precisa levar o avatar do Google Maps ao seu destino. Para isso, ele deve usar o conjunto de comandos disponíveis na plataforma. Neste exemplo, bastou utilizar dois comandos “Avançar” para que objetivo fosse atingido. Ao resolver o problema, o software informa a quantidade de linhas de códigos em outra linguagem, chamada Javascript. Todavia, para o treino de lógica é muito interessante.

Figura 1.2 | Jogo Labirinto Blockly para treino de lógica



Fonte: captura de tela do jogo Labirinto Blockly elaborada pelo autor.



Agora é com você . Veja se consegue completar todos os desafios do jogo “Labirinto ”.

Para visualizar o objeto, acesse seu material digital.

o

Ver anotações

## LINGUAGEM NATURAL

Segundo Santos (2001), a linguagem natural é uma forma de comunicação entre as pessoas de diversas línguas, que pode ser falada, escrita ou gesticulada, entre outras formas de comunicação. A linguagem natural é uma grande contribuição para o desenvolvimento de uma aplicação computacional, pois pode direcionar de forma simples e eficiente as descrições dos problemas e suas soluções.

O algoritmo para cozimento de arroz, apresentado anteriormente, é um exemplo de uso da linguagem natural. Para reforçar esse conceito, podemos considerar o cadastro das notas dos alunos de um curso. O problema é o seguinte: o usuário (possivelmente o professor) deverá entrar com dois valores que representam as notas de um aluno em cada bimestre, e o computador retornará o resultado da média desses valores (média das notas). Então, se a média for maior ou igual a 6.0 (seis), o aluno está aprovado; caso contrário, está reprovado.

Para realizar a solução desse problema, podemos fazer uso da seguinte estrutura:

*Início.*

Entrar com a nota do aluno no primeiro bimestre.

Entrar com a nota do aluno no segundo bimestre.

Realizar a soma das duas notas e dividir o resultado por dois (isso corresponde à média do aluno).

Armazenar o valor da média encontrada.

Mostrar na tela o valor da média.

Se a média for maior ou igual a seis, mostrar na tela que o aluno está



aprovado.

Senão, mostrar na tela que o aluno está reprovado.

*Fim.*

Observe que a linguagem natural é muito próxima da nossa linguagem.

Antes de iniciar a explicação acerca do diagrama de blocos e pseudocódigo, vamos entender sucintamente o que são variáveis e atribuições, para que você, aluno, tenha condições de interpretar e avançar nos seus estudos de algoritmos.

As **variáveis**, como o próprio nome sugere, consistem em algo que pode sofrer variações. Elas estão relacionadas à identificação de uma informação, por exemplo, o nome de um aluno, suas notas bimestrais, entre outras. A **atribuição**, representada em pseudocódigo por meio do símbolo  $\leftarrow$ , tem a função de indicar valores para as variáveis, ou seja, atribuir informação para variável. Por exemplo:

idade  $\leftarrow$  8

nome\_aluno  $\leftarrow$  "márcio"

nome\_professor  $\leftarrow$  "josé"

nota\_aluno  $\leftarrow$  9.5

O exemplo apresentado indica que o número "8" está sendo atribuído como valor (informação) para a variável "idade". Analogamente, o texto "márcio" está atribuído como valor para a variável "nome\_aluno" e o valor real "9.5" está sendo atribuído como valor para a variável "nota\_aluno".

Dando sequência ao seu estudo de algoritmos, veja agora o funcionamento dos diagramas de blocos, que também podem ser chamados de fluxogramas.

## DIAGRAMA DE BLOCOS (FLUXOGRAMA)

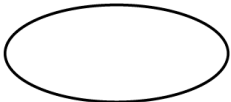


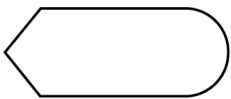
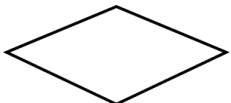

0

Ver anotações

Segundo Manzano, Franco e Villar (2015), podemos caracterizar diagrama de blocos como um conjunto de símbolos gráficos, em que cada um desses símbolos representa ações específicas a serem executadas pelo computador. Vale lembrar que o diagrama de blocos determina a linha de raciocínio utilizada pelo desenvolvedor para resolver problemas. Ao escrever um diagrama de blocos, o desenvolvedor deve estar ciente de que os símbolos utilizados estejam em harmonia e sejam de fácil entendimento. Para que os diagramas de blocos tenham certa coerência, os seus símbolos foram padronizados pela ANSI (*American National Standards Institute* ou, em português, Instituto Americano de Padronização). O Quadro 1.1 mostra os principais símbolos utilizados para se descrever um algoritmo.

Ver anotações

Quadro 1.1 | Descrição e significados dos principais símbolos no diagrama de blocos

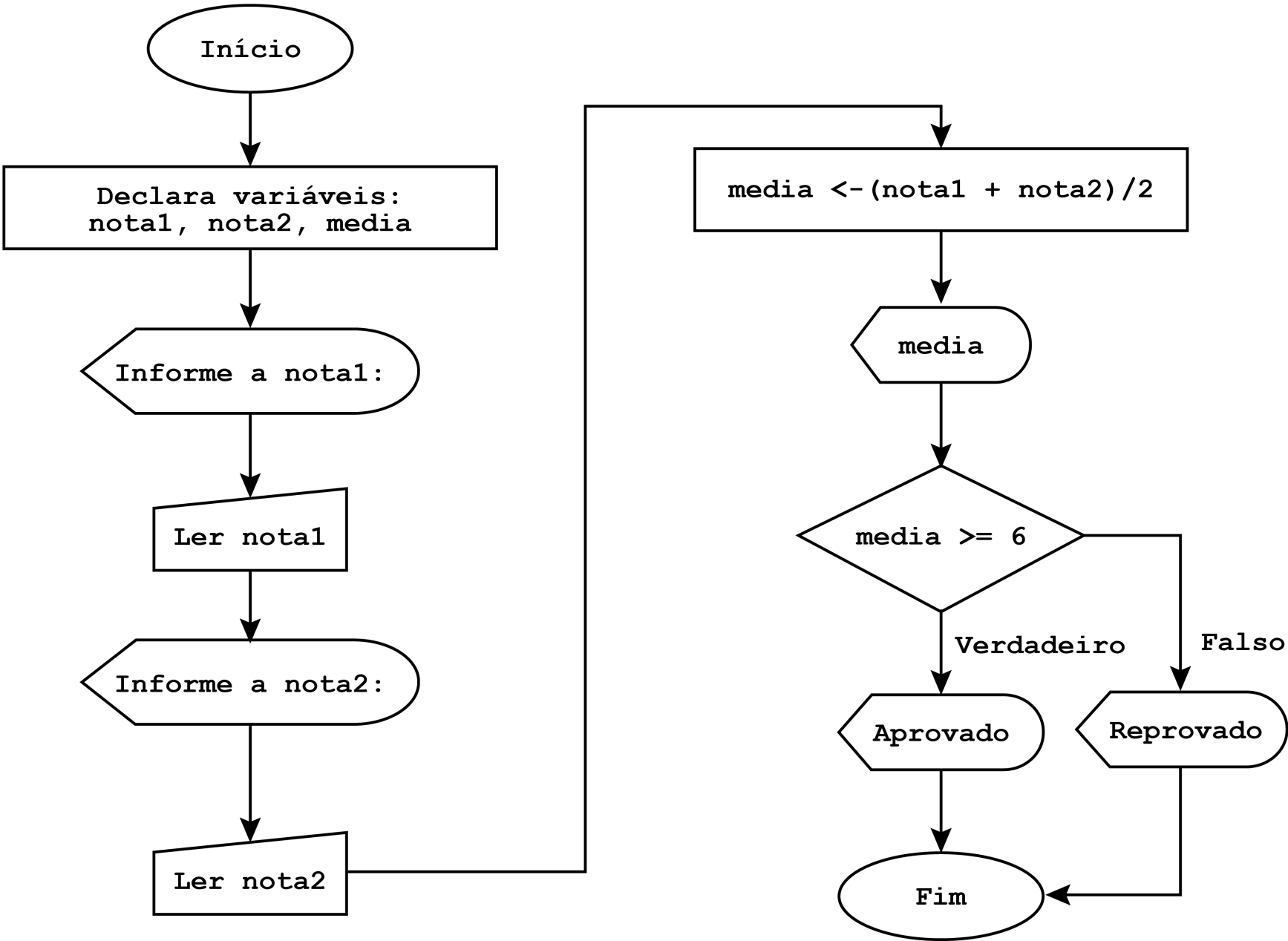
Símbolo	Significado	Descrição
	Terminal	Representa o início ou o fim de um fluxo lógico.
	Entrada/Saída de dados	Determina a entrada manual de dados por meio de um teclado ou a saída de dados.
	Processo	Representa operações/ações do algoritmo.
	Exibição	Mostra o resultado de uma ação, geralmente por meio da tela do computador.
	Condição	Mostra direções que o algoritmo deve seguir, conforme o resultado de uma condição. São os desvios condicionais.
	Fluxo	Sentido (direção) dos passos do algoritmo.

Fonte: adaptado de Manzano. Franco e Villar (2015).

Ao utilizar os símbolos do diagrama de blocos com suas respectivas instruções, você vai aprendendo e desenvolvendo cada vez mais a sua lógica em relação à resolução de problemas. A Figura 1.3 ilustra o diagrama com o algoritmo descrito em linguagem natural, que calcula a média de notas e a situação de um aluno, representado por meio de um diagrama de blocos.

Ver anotações

Figura 1.3 | Diagrama de blocos (fluxograma)



Fonte: elaborada pelo autor.

**ASSIMILE**

Algumas dicas para construir um diagrama de blocos (fluxograma) são as seguintes:

- Estar atento à sequência das instruções.
- Certificar-se de que o diagrama de blocos (fluxograma) comece de cima para baixo e da esquerda para direita.

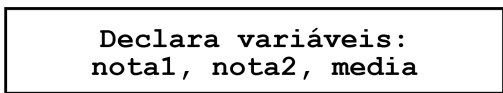
- Ficar atento para não cruzar as linhas dos fluxos.

Vejamos, então, as representações de cada passo do diagrama:

O símbolo terminal deu início ao algoritmo.



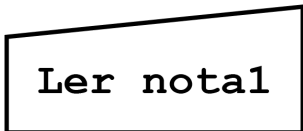
O símbolo de processamento definiu as variáveis.



O símbolo exibição mostra na tela o que o usuário deve fazer. Nesse caso, ele deve informar o valor da primeira nota do aluno, isto é, a nota do primeiro bimestre.



O símbolo de entrada manual libera o usuário para entrar com a primeira nota. Observe que há uma diferença entre “nota 1”, no símbolo anterior, e “nota1” descrita a seguir. No símbolo anterior, “nota 1” refere-se ao texto que será apresentado ao usuário, para que ele saiba o que deve ser feito. Já o símbolo que segue, “nota1”, refere-se ao nome da variável declarada anteriormente (no segundo símbolo utilizado nesse diagrama de blocos).



De forma análoga, o exemplo segue com a leitura da segunda nota do aluno.

O próximo símbolo de processamento realiza a atribuição do resultado do cálculo da média aritmética das duas notas lidas anteriormente à variável “media”.

Neste momento, você pode se perguntar se não precisa adicionar a soma das notas em uma variável antes de calcular a média. A resposta é: depende! Se você for usar o valor da soma das notas para mostrá-lo na tela ou como

0

Ver anotações

entrada de outro cálculo, além da média, então sim, vale a pena armazená-lo em uma variável. Como esse não é o nosso caso, então não foi preciso declarar mais uma variável no algoritmo.

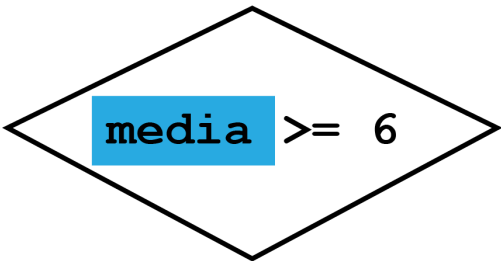
Ver anotações

```
media <- (nota1 + nota2) / 2
```

O símbolo de exibição mostra na tela o resultado da média calculada.



O símbolo de decisão define a condicional (verdadeiro ou falso) para a expressão "media >= 6".



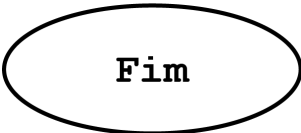
Se a condição for verdadeira, o texto impresso na tela do usuário será "Aprovado".



Se a condição for falsa, o texto impresso na tela do usuário será "Reprovado".

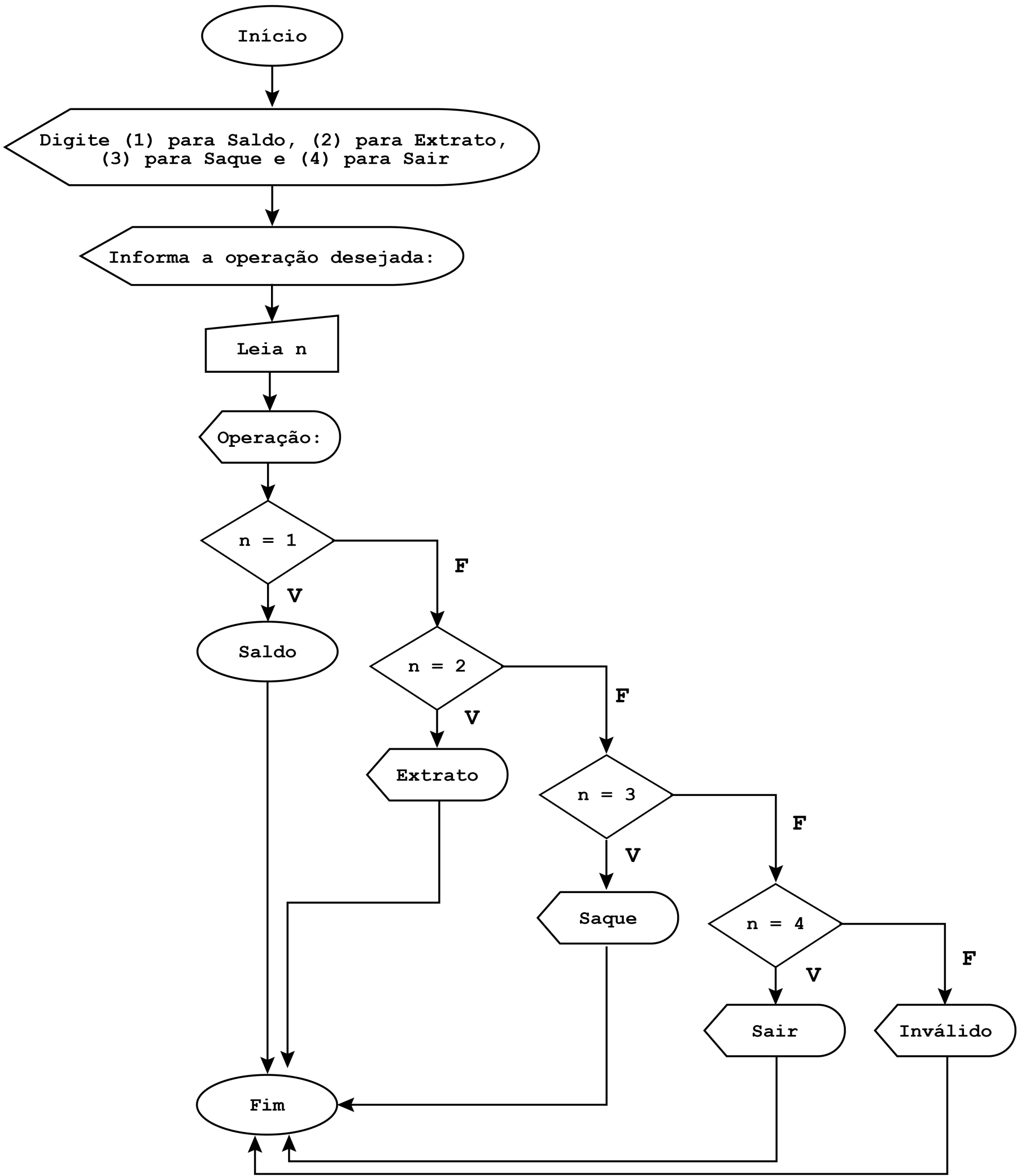


Para finalizar o algoritmo, o símbolo terminal é utilizado mais uma vez.



Outro exemplo que podemos destacar é a operação em um caixa eletrônico. Uma das primeiras atividades que o usuário deve realizar após ter se identificado é selecionar a operação a ser executada. Por exemplo: verificar saldo, emitir extrato, saque e sair. A Figura 1.4 ilustra o diagrama para realizar essa operação.

Figura 1.4 | Fluxograma de operação de caixa eletrônico



Fonte: elaborada pelo autor.

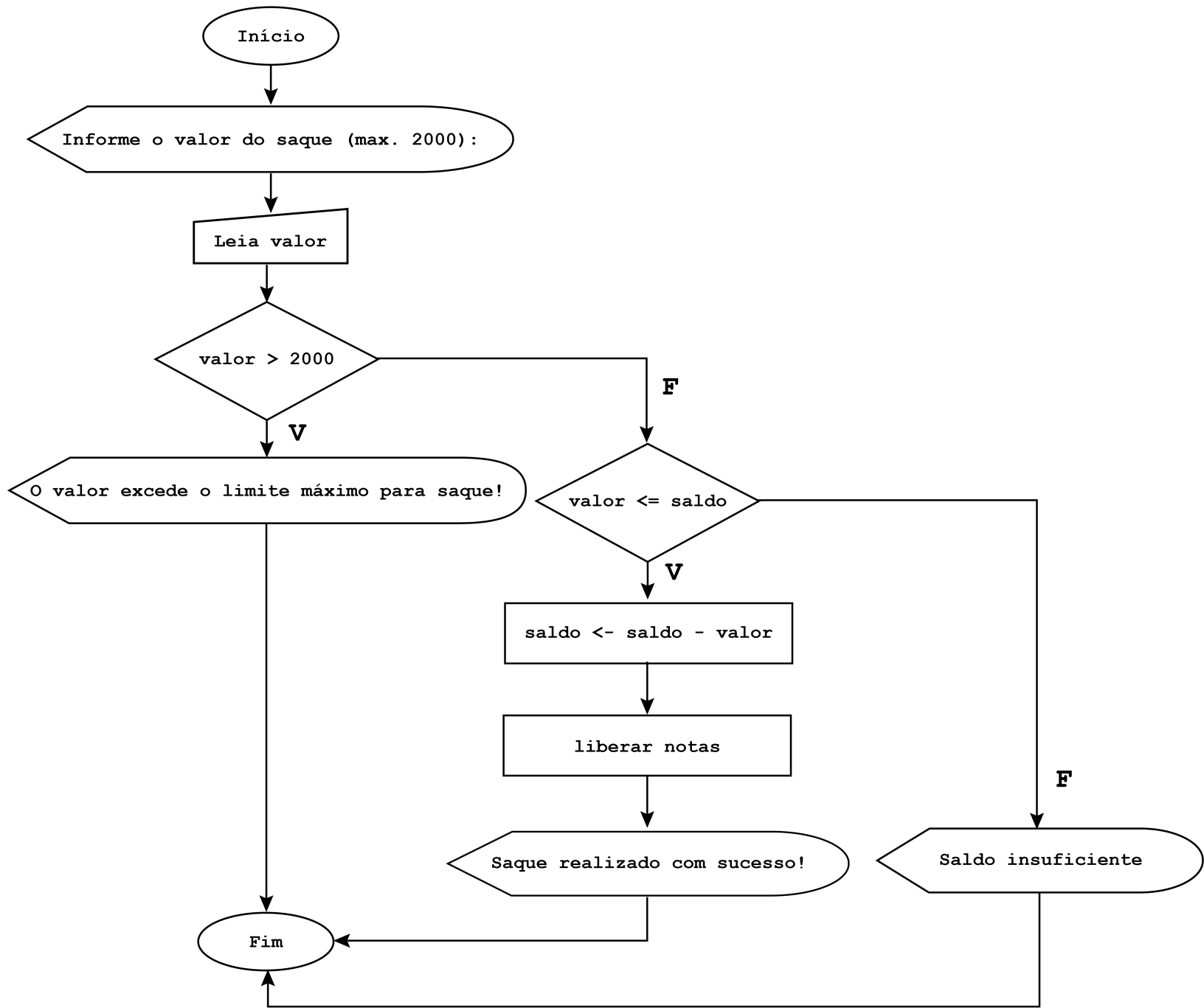
EXEMPLIFICANDO

Ainda a respeito do exemplo da Figura 1.4, para que o fluxograma não fique muito extenso, dificultando seu entendimento, você pode tratar cada operação bancária em um fluxograma à parte.



A Figura 1.5 ilustra o exemplo de fluxograma que trata da operação de saque em um sistema bancário.

Figura 1.5 | Fluxograma da operação de saque



Fonte: elaborada pelo autor.

REFLITA

Observando o fluxograma que representa a operação de saque em um sistema bancário, apresentado na Figura 1.5, como você alteraria o fluxograma para que o usuário possa escolher informar ou não um novo valor, caso o valor atualmente informado por ele seja inválido (ou seja, exceda o limite máximo ou seja maior do que o saldo atual do cliente)?

PSEUDOCÓDIGO

Ver anotações

Conforme apresentado, um algoritmo pode ser representado graficamente na forma de um fluxograma. Porém, existem outras maneiras de representar algoritmos, e uma delas é uma forma textual muito próxima da linguagem natural, denominada pseudocódigo. Segundo Aguilar (2011), o pseudocódigo corresponde à escrita de um algoritmo em palavras similares ao inglês ou ao português, com o intuito de facilitar a interpretação e o desenvolvimento de um programa. Nesse caso, para a criação de um pseudocódigo, deve-se analisar o problema a ser resolvido e escrever o algoritmo, por meio de regras predefinidas, com a sequência de passos a ser seguida para a resolução.

Ver anotações

Para a escrita do seu primeiro pseudocódigo, é necessário conhecer algumas estruturas importantes e comandos básicos. No Quadro 1.2 são descritos alguns desses comandos.

Quadro 1.2 | Comandos básicos utilizados para criar um pseudocódigo

Estruturas	Descrição
Escreva (" ")	Imprime uma mensagem na tela.
Leia ( )	Lê valores digitados no teclado.
<-	Comando de atribuição.
inicio	Inicia o algoritmo/programa.
fimalgoritmo	Finaliza o algoritmo.
var	Realiza a declaração de variáveis
algoritmo	Indica o início do algoritmo.

Fonte: elaborado pelo autor.

É importante estar atento para algumas regras básicas ao se utilizar pseudocódigos:

- Escolher um nome para identificar seu algoritmo.
- Avaliar as variáveis, dar atenção aos seus tipos e características.
- Descrever de forma clara o que será armazenado e se as variáveis destinadas a essa informação estão corretas.
- Verificar se as instruções fazem sentido e se têm uma sequência lógica.
- Avaliar o resultado e, quando pertinente, mostrá-lo na tela.
- Finalizar o algoritmo.

o  
Ver anotações

Recordando o exemplo que calcula a média das notas de aluno, observe o pseudocódigo a seguir :

```
1  Algoritmo "Calcula média de notas"
2  Var
3      nota1, nota2, media: Real
4  Início
5      Escreva("Informe a nota 1:")
6      Leia(nota1)
7      Escreva("Informe a nota 2:")
8      Leia(nota2)
9      media <- (nota1 + nota2)/2
10     Escreva(media)
11     Se (media >=6) então
12         Escreva("Aprovado")
13     senão
14         Escreva("Reprovado")
15     fimse
16 Fimalgoritmo
17
```

0  
Ver anotações

Os detalhes dos pseudocódigos são explicados linha a linha a seguir.

**Linha 1** – “Calcula média de notas”: esse é o nome reservado para identificar o algoritmo.

**Linha 2** – “Var”: indica a declaração das variáveis.

**Linha 3** – são os nomes dados para as variáveis (nota1, nota2, media). Nessa linha também é definido o tipo dessas variáveis (“Real”).

**Linha 4** – inicia os procedimentos dos algoritmos (“Início”).

**Linha 5** – “Escreva”: é um comando de saída que indica o que vai sair na tela do computador. Geralmente o conteúdo do texto a ser mostrado fica entre aspas (“Informe a nota 1:”).

**Linha 6** – “Leia”: é comando de entrada, e o valor digitado é armazenado na variável (nota1).

**Linha 9** – realiza o cálculo da média e atribui o valor encontrado à variável média:  $media \leftarrow (nota1 + nota2)/2$ .

**Linha 11** – utiliza o resultado da média para criar uma condição verdadeira ou falsa: se (media  $\geq 6$ ).

**Linha 12** – se o resultado da média for maior ou igual a seis (condição verdadeira), o computador escreve na tela “Aprovado”.

**Linha 14** – se o resultado da média for menor que seis (condição falsa), o computador escreve na tela “Reprovado”.

**Linha 15** – encerra a condição (fimse).

**Linha 16** – encerra o algoritmo com a palavra “Fimalgoritmo”.

### EXEMPLIFICANDO

Para testar os pseudocódigos, você pode utilizar o Visualg. Trata-se de um software gratuito capaz de executar algoritmos escritos em pseudocódigo, utilizando o idioma português do Brasil. O download está disponível no site da ferramenta (VISUALG3, 2020b).

Veja a seguir o exemplo de algoritmo escrito em pseudocódigo e executado no Visualg:

o

Ver anotações

```
1  algoritmo "Imprime o maior"
2  var
3      num1, num2, maior: real
4  inicio
5      Escreval("Digite o primeiro número: ")
6      Leia (num1)
7      Escreval("Digite o segundo número: ")
8      Leia (num2)
9      Se (num1 >= num2) então
10         maior <- num1
11     senão
12         maior <- num2
13     fimse
14     Escreval("Maior número: ", maior)
15 Fimalgoritmo
16
```

0  
Ver anotações

## CONCEITOS DE LINGUAGEM DE PROGRAMAÇÃO

Após compreendermos os conceitos, aplicações e tipos de algoritmos, vamos entender a importância da linguagem de programação e das suas famílias, assim como as projeções profissionais que a carreira de programador pode proporcionar.

Segundo Marçula (2013, p. 170):

“

A Linguagem de Programação (LP) pode ser entendida como um conjunto de palavras (vocabulário) e um conjunto de regras gramaticais (para relacionar essas palavras) usados para instruir o sistema de computação a realizar tarefas específicas e, com isso, criar os programas. Cada linguagem tem o seu conjunto de palavras-chave e sintaxes.

Para Tucker (2010), da mesma forma que entendemos as linguagens naturais, utilizadas por todos no dia a dia, a linguagem de programação é a comunicação de ideias entre o computador e as pessoas. Ainda segundo o autor, as primeiras



linguagens de computador utilizadas foram as linguagens de máquina e a linguagem *assembly*, a partir da década de 1940. Desde então, muitas linguagens surgiram, bem como foram criados **paradigmas de linguagem de programação**.

---

De acordo com Houaiss, Franco e Villar (2001, p. 329), “paradigma significa modelo, padrão. No contexto da programação de computadores, um paradigma é um jeito, uma maneira, um estilo de se programar”. Segundo Tucker (2010), um paradigma de programação está relacionado a um padrão de soluções de problemas, os quais, por sua vez, estão relacionados a uma determinada linguagem de programação.

---

o  
Ver anotações

Segundo Tucker (2010), quatro paradigmas de programação tiveram sua evolução reconhecida nas últimas décadas:

1. **Programação imperativa:** considerado o paradigma mais antigo, pode armazenar o programa e suas variáveis juntamente, assim como a abstração procedural, as atribuições, as sequências, os laços, os comandos condicionais. Exemplo de linguagens de programação (LP) que utilizam programação imperativa: COBOL, Fortran, C, Ada e Perl.
2. **Programação orientada a objeto:** também conhecida na computação como POO, como o próprio nome sugere, é considerada uma coleção de objetos que se inter-relacionam. São exemplos de LP relacionados à POO: Smalltalk, C++, Java e C#.
3. **Programação funcional:** caracterizada por apresentar atuação matemática, cada uma com um espaço de entrada (domínio) e resultado (faixa). Exemplos de LP desse paradigma: Lisp, Scheme e Haskell.
4. **Programação lógica:** considerada uma programação declarativa, na qual um programa pode modelar uma situação-problema declarando qual resultado o programa deve obter em vez de como ele deve ser obtido. Podemos citar como exemplo de LP lógica o Prolog.

Todas as linguagens de programação para a criação de um programa apresentam uma sintaxe, que nada mais é do que a forma como o programa é escrito. De acordo com Tucker (2010, p. 24), “[a] sintaxe de uma linguagem de programação é uma descrição precisa de todos os seus programas gramaticalmente corretos”. °

Não há como negar: todas as áreas estão voltadas para a tecnologia e é por meio de diversas formas de pensamentos que os algoritmos são realizados. Por se tratar de algumas das profissões do futuro, é necessário ter em mente que as linguagens evoluem e que será preciso estar sempre atualizado, realizar certificações, estudar línguas e buscar novos caminhos na sua área de atuação. Você poderá ser um grande entusiasta em algoritmos e linguagens de programação. Ver anotações

Como já vimos os principais tipos de paradigmas e algumas linguagens de programação relacionadas a eles, vamos partir para o próximo passo e conhecer a linguagem com a qual trabalharemos ao longo deste livro, a saber, a linguagem C.

## FAÇA VALER A PENA

### Questão 1

Segundo Aguilar (2011), o pseudocódigo é considerado uma ferramenta que pode auxiliar na programação e pode ser escrito em palavras similares ao inglês ou português para facilitar a interpretação e o desenvolvimento de um programa.

Analise o algoritmo a seguir que calcula o bônus de 30% para os funcionários cadastrados e complete a linha de programação que está faltando:

```
1  Algoritmo "Calcula bônus"
2  Var
3      nome: Caractere
4      _____
5
6  Inicio
7      Escreval("Informe o nome do funcionário: ")
8      Leia(nome)
9      Escreval("Informe o salário do funcionário: ")
10     Leia(salario)
11
12     _____
13     salario_total <- salario + bonus
14
15     Escreval("Salário bonificado = R$", salario_total)
16
17 Fimalgoritmo
18
```

0  
Ver anotações

Assinale a alternativa correta:

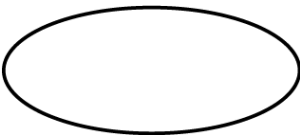

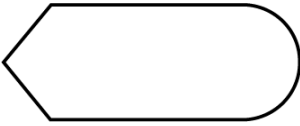
- a. salario, bonus: Real  
bonus <- salario \* (30/100).
- b. salario, bonus, salario\_total: Real  
salario <- bonus \* (30/100).
- c. salario, bonus, salario\_total: Real  
bonus <- salario \* (30/100).
- d. salario, bonus, salario\_total: Real
- e. bonus <- salario \* (30/100).

Questão 2

Segundo Manzano, Franco e Villar (2015), podemos caracterizar diagrama de blocos como um conjunto de símbolos gráficos, no qual cada um desses símbolos representa ações específicas a serem executadas pelo computador. Vale lembrar que o diagrama de blocos determina a linha de raciocínio utilizada pelo programador para resolver um problema. Analise os símbolos a seguir:

o

Ver anotações

- I.  Esse símbolo significa um terminal, o qual pode representar o início ou o fim de um fluxo lógico. Em alguns casos, define as sub-rotinas.
- II.  Esse símbolo significa exibição e é responsável por mostrar o resultado de uma ação, geralmente por meio da tela de um computador.
- III.  Esse símbolo significa entrada manual e representa a entrada de dados para o programa, geralmente por meio de um teclado.

É correto o que se afirma em:

- a. I, apenas.
- b. II, apenas.
- c. I e II, apenas.
- d. II e III, apenas.
- e. I, II e III.

Questão 3

De acordo com Houaiss, Franco e Villar (2001, p. 329), “paradigma significa modelo, padrão. No contexto da programação de computadores, um paradigma é um jeito, uma maneira, um estilo de se programar”. Para Tucker (2010), um paradigma de programação está relacionado a um padrão de soluções de problemas, os quais, por sua vez, estão relacionados a determinada linguagem de programação.

Com base no contexto apresentado, analise as afirmativas a seguir:

- I. **Programação imperativa:** considerado o paradigma mais antigo, pode armazenar o programa e suas variáveis juntamente, assim como a abstração procedural, as atribuições, as sequências, os laços, os comandos condicionais.
- II. **Programação orientada a objeto:** também conhecida na computação como POO, como o próprio nome sugere, é considerada uma programação somente por objetos, não podendo ter manipulações por código.
- III. **Programação funcional:** caraterizada por apresentar atuação matemática, cada uma com um espaço de entrada (domínio) e resultado (faixa).
- IV. **Programação lógica:** considerada uma programação declarativa, na qual um programa pode modelar uma situação-problema declarando qual resultado o programa deve obter, em vez de como ele deve ser obtido.

Ver anotações

É correto o que se afirma apenas em:

- a. I, II e III, apenas.
- b. II, III e IV, apenas.
- c. II e III, apenas.
- d. III, apenas.
- e. I, III e IV, apenas.

## REFERÊNCIAS

AGUILAR, L. J. **Fundamentos de programação:** algoritmos, estruturas de dados e objetos. 3. ed. Porto Alegre: AMGH, 2011.

BERG, A. C.; FIGUEIRÓ, J. P. **Lógica de programação.** 2 ed. Canoas: Ulbra, 1998.

BLOCKLY GAMES. Página inicial. 2020. Disponível em: <https://bit.ly/3noPmSL>. Acesso em: 28 set. 2020.

DAMAS, L. **Linguagem C**. Tradução: João Araújo Ribeiro, Orlando Bernardo Filho. 10. ed. Rio de Janeiro: LTC, 2016.

DIA DIAGRAM EDITOR. **Dia installer**. The Dia Developers, 2014. Disponível em: <http://dia-installer.de/>. Acesso em: 17 mar. 2018.

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. **Lógica de programação**: a construção de algoritmos e estruturas de dados. São Paulo: Makron, 2000.

HOUAISS, A.; FRANCO, F. M. M.; VILLAR, M. S. **Dicionário Houaiss da Língua Portuguesa**. São Paulo: Objetiva, 2001.

LOPES, A.; GARCIA, G. **Introdução à programação**. 8. reimp. Rio de Janeiro: Elsevier, 2002.

LUCIDCHART. Página inicial. Lucid Software Inc., 2020. Disponível em: <https://bit.ly/3kfaMzH>. Acesso em: 17 mar. 2018.

MANZANO, J. A. N. G. **Estudo dirigido de Linguagem C**. 17. ed. rev. São Paulo: Érica, 2013.

MANZANO, J. A. N. G.; MATOS, E.; LOURENÇO, A. E. **Algoritmos**: técnicas de programação. 2. ed. São Paulo: Érica, 2015.

MARÇULA, M. **Informática**: conceitos e aplicações. 4. ed. rev. São Paulo: Érica, 2013.

MIZRAHI, V. V. **Treinamento em linguagem C**. 2. ed. São Paulo: Pearson Prentice Hall, 2008.

PEREIRA, A. P. **O que é algoritmo?** TecMundo, 12 maio 2009. Disponível em: <https://bit.ly/35jE0Jk>. Acesso em: 19 mar. 2018.



PIVA JUNIOR, D. *et al.* **Algoritmos e programação de computadores**. Rio de Janeiro: Elsevier, 2012.

PORTUGOL WEBSTUDIO. Página inicial. 2020. Disponível em: <https://bit.ly/3lkNwSb>. Acesso em: 16 out. 2020.

SALIBA, W. L. C. **Técnica de programação**: uma abordagem estruturada. São Paulo: Makron, 1993.

SANTOS, D. **Processamento da linguagem natural**: uma apresentação através das aplicações. Organização: Ranchhod. Lisboa: Caminho, 2001.

SOFFNER, R. **Algoritmos e programação em Linguagem C**. São Paulo: Saraiva, 2013.

SZWARCFITER, J. L.; MARKENZON, L. **Estruturas de dados e seus algoritmos**. Rio de Janeiro: LTC, 1994.

TUCKER, A. B. **Linguagens de programação**: princípios e paradigmas. Porto Alegre: AMGH, 2010.

VISUALG3. Página inicial. VisualG3, 2020a. Disponível em: <http://visualg3.com.br/>. Acesso em: 10 mar. 2020.