ALGORITMOS E PROGRAMAÇÃO

sagah

Introdução de algoritmos e raciocínio lógico

Renata Junges Padilha

OBJETIVOS DE APRENDIZAGEM

- Definir o conceito de algoritmos e lógica de programação.
- Diferenciar os componentes básicos de funcionamento de um computador na execução de algoritmos.
- Aplicar as etapas de construção de um algoritmo na solução de problemas.

Introdução

Compreender o que são e como funcionam os algoritmos é essencial na área de informática. A partir da análise e da criação de algoritmos, é possível fazer um estudo sistematizado sobre um determinado problema que pode ser resolvido via computação.

O pontapé inicial na caminhada da programação é o estudo sobre algoritmos, pois é a partir dele que é extraída e compreendida a problematização. Tal problematização deve ser desenvolvida em forma de programa, facilitando o trabalho do programador.

De acordo com Berg e Figueiró (2006), é possível definir os algoritmos como um meio de obter a solução de qualquer problema ao seguir uma ordem lógica e compreensível para o computador, considerando o raciocínio lógico. Existem diversas formas de representar algoritmos, como descrição narrativa, fluxogramas e pseudocódigo (linguagem algorítmica).

Neste capítulo, você vai ver o que são algoritmos e lógica de programação. Além disso, vai estudar como os algoritmos podem ser representados e qual é a relação do computador com o desenvolvimento dos algoritmos. Também vai conhecer exemplos que demonstram como os algoritmos podem ser criados e interpretados.

Algoritmos e lógica de programação

De acordo com Manzano e Oliveira (2016), o termo algoritmo é utilizado tanto para a computação quanto para a matemática. Na matemática, ele serve para representar processos de cálculo ou resolução de problemas. Na área da computação, o termo teve origem no ano 830 d.C., quando foi usado na publicação de um livro de Al-Khwarizmi sobre álgebra.

Os algoritmos podem ser aplicados em diversos segmentos, incluindo para problemas do dia a dia, como a compra de um determinado livro. Diferentes passos devem ser tomados para se chegar a um ponto final, que é efetivar a compra do livro. O exemplo a seguir ilustra o algoritmo desse cenário da compra de um livro. É importante lembrar que um algoritmo tem um estado inicial, executa algumas ações, tem um resultado e apresenta um estado final (EDELWEISS; LIVI, 2014, p. 6).

1. Entrar na livraria.
2. Verificar se o livro está disponível. Para isso, é necessário conhecer o título e o autor do livro e ter disponibilidade financeira para a compra. Caso a compra venha a ser efetuada, deve-se:
   a) esperar que a compra seja registrada no caixa;
   b) esperar que seja feito o pacote;
   c) levar o livro até o balcão;
   d) levar o livro comprado;
   e) pagar o valor correspondente;
3. Sair da livraria.

Na área da computação, os algoritmos podem ser definidos como uma sequência de ações executadas em uma ordem preestabelecida, tendo como objetivo demonstrar o passo a passo para se chegar a uma determinada solução de uma problematização (EDELWEISS; LIVI, 2014).

Atrelada ao conhecimento de algoritmos, a lógica de programação é um ponto muito importante. A palavra lógica está relacionada a coerência e racionalidade, e na informática isso não é diferente. A lógica de programação é

uma área bem ampla, que capacita o programador ao entendimento de todas as esferas da programação. De acordo com Manzano e Oliveira (2016), a lógica de programação objetiva ações que devem ser tomadas para se desenvolver e chegar a um resultado final. Algumas das principais ações que podem e/ou devem ser observadas na construção de um programa são a facilidade de depuração, a verificação de possíveis falhas, a facilidade de alterações e a reutilização de códigos.

Saber interpretar ou criar algoritmos eficientes é o pontapé inicial para desenvolver as ações citadas. As principais formas de representar os algoritmos são por meio de descrição narrativa, fluxogramas e pseudocódigo. Na **descrição narrativa**, a representação é simples, por meio de linguagem natural (utilizada no cotidiano). Geralmente, a descrição narrativa é pouco utilizada para representar algoritmos, pois abre brechas para a ambiguidade. O exemplo de algoritmo que vimos, da compra do livro, é uma forma de descrição narrativa.

Os **fluxogramas** são a forma gráfica de representar os algoritmos, por meio da utilização de formas geométricas com significados diferentes para cada parte do fluxo. É possível demonstrar a entrada, as ações e a saída do algoritmo de uma forma bem dinâmica. É possível dizer que os fluxogramas são a segunda forma mais utilizada para representar algoritmos (atrás apenas do pseudocódigo), pois são mais precisos que a descrição narrativa, ainda que não se preocupem com muitos detalhes computacionais (OLIVEIRA, 2004). A Figura 1 ilustra um fluxograma cujo objetivo é representar o algoritmo de um programa que recebe três notas de alunos e tem como saída a soma e a média aritmética.

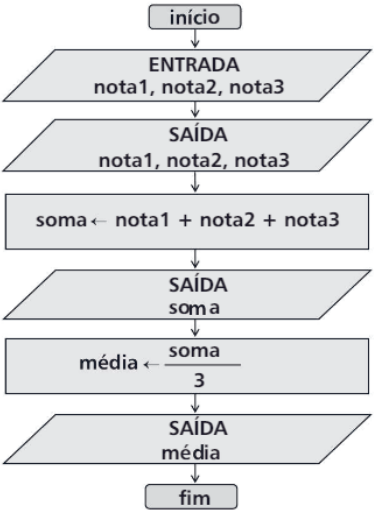

Figura 1. Fluxograma de algoritmo lendo notas e obtendo diferentes saídas.
Fonte: Edelweiss e Livi (2014, p. 66).

De acordo com Edelweiss e Livi (2014), um **pseudocódigo** expressa os algoritmos de uma forma similar a uma linguagem de programação, mas sem entrar em detalhes, como informações sobre os tipos de dados. Além disso, as operações são expressas em uma linguagem natural, facilitando o entendimento. O exemplo a seguir mostra um pseudocódigo que tem o mesmo objetivo do fluxograma da Figura 1.

```
Algoritmo media1
   {INFORMA A SOMA E A MÉDIA DAS 3 NOTAS DE UM ALUNO}
   Entradas: nota1, nota2, nota3 (real)
   Saídas: soma, média (real)
   início
     ler (nota1, nota2, nota3) {ENTRADA DAS 3 NOTAS}
     escrever (nota1, nota2, nota3) {INFORMA AS NOTAS LIDAS}
     soma ← nota1 + nota2 + nota3 {CALCULA A SOMA}
     escrever (soma) {INFORMA SOMA}
     média ← soma / 3 {CALCULA A MÉDIA}
     escrever (média) {INFORMA MÉDIA CALCULADA}
  fim
```

O algoritmo tratado tem as informações sobre as três notas dos alunos. Inicialmente, há a entrada (leitura), e depois as três notas são impressas na tela, mostrando para o usuário o que foi lido (digitado). O próximo passo é realizar a soma das notas e mostrá-las. Por fim, a média é calculada e mostrada para o usuário. Diferentemente do fluxograma da Figura 1, o pseudocódigo do exemplo traz informações de tipos de dados e declaração das variáveis, além de comentários para facilitar o entendimento.

Saiba mais

As variáveis são um determinado espaço de memória que foi utilizado para guardar valores. Tais valores podem variar ao longo da execução de um programa.

Com base nos exemplos e conceitos apresentados, é possível notar como os algoritmos e a lógica de programação são importantes para a área da computação. Na próxima seção, vamos estudar a relação dos componentes básicos de um computador com o desenvolvimento de algoritmos.

Relação entre computador e algoritmos

Como visto, os algoritmos são uma sequência de ações definidas para se chegar a um determinado resultado. Também vimos que os algoritmos podem ser executados por um computador. É importante entender essa relação entre computador e algoritmo, pois ela permite conhecer como os algoritmos podem ser executados.

A partir da tradução de um algoritmo para uma linguagem que o computador entenda, é gerado o programa que pode ser executado pelo computador. O componente responsável por essa execução e pela execução de todas as instruções na ordem correta é a CPU (*central processing unit*, ou unidade central de processamento) (EDELWEISS; LIVI, 2014).

A CPU é composta pela unidade aritmética e lógica e pela unidade de controle. A unidade de controle é responsável por executar instruções de controle do computador, direcionando instruções. Já a unidade aritmética e lógica é o componente mais importante da CPU, pois executa o processamento de operações matemáticas e lógicas (MANZANO; OLIVEIRA, 2016).

A unidade de entrada está relacionada aos dados inseridos no computador (dados de entrada). Esses dados podem ser armazenados em memória primária ou secundária. Alguns exemplos de componentes de entrada são os periféricos teclado e *mouse*. Já a unidade de saída representa os

dados que foram processados e retornados como resultados (MANZANO; OLIVEIRA, 2016).

Por meio das unidades de entrada e saída, é possível realizar a comunicação do computador com o usuário durante a execução de um programa, pois é a partir da entrada que são inseridos, por exemplo, os valores iniciais de uma soma de números. Como saída, é mostrado o resultado desse cálculo.

Na Figura 2, é possível visualizar o esquema de processamento de um computador. De forma geral, o processamento que um computador faz diz respeito a executar ações e/ou tarefas, como comparações, operações aritméticas, etc. Para isso acontecer, é necessário ter os dados de entrada, que são processados para, em seguida, mostrar resultados na saída. O fluxo de processamento ocorre a partir da leitura dos dados na unidade de entrada. Tais dados são processados na CPU (composta por unidades que realizam a execução de ações/atividades), juntamente com a memória principal, realizando o retorno do processamento na unidade de saída (EDELWEISS; LIVI, 2014).

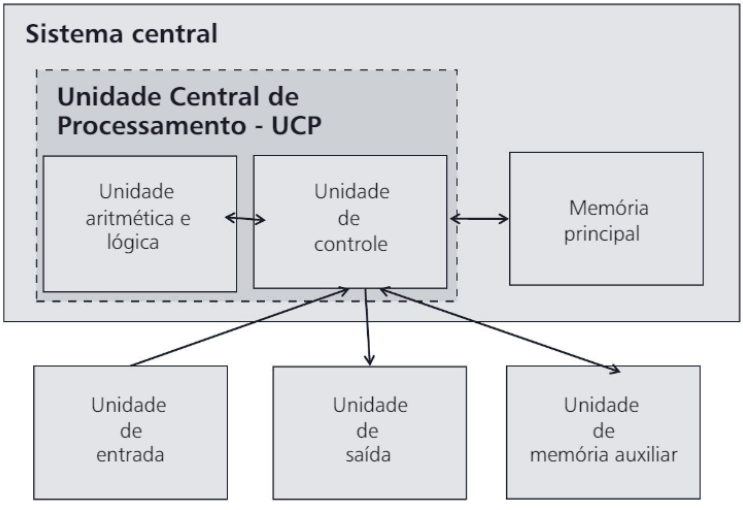

Figura 2. Esquema simplificado de um computador.
Fonte: Edelweiss e Livi (2014, p. 11).

Compreender o fluxo de processamento realizado pelo computador ajuda a entender como os programas são lidos e como funciona a relação entre computador e algoritmos. Deve-se ter em mente que o computador executa programas feitos em diferentes linguagens de programação, mas que têm

o mesmo esquema (entrada, processamento e saída). É por isso que os algoritmos são tão importantes. Afinal, independentemente da linguagem de programação utilizada, a forma como eles são criados ajuda os programadores a entender como o programa deve se comportar. Vale lembrar que os algoritmos têm dados, informações e fluxo, que podem ser observados no processamento de um computador.

Na próxima seção, vamos ver as principais fases que constituem a criação de um algoritmo. Para isso, conheceremos alguns exemplos de algoritmos construídos a partir de problematizações.

Etapas de construção de um algoritmo

Até aqui, estudamos os conceitos de algoritmo e lógica de programação, vimos alguns exemplos de representação de algoritmos e identificamos a relação importante entre o computador e os algoritmos. Nesta seção, vamos ver as etapas que constituem o esquema ideal para a construção de um algoritmo, considerando alguns exemplos de problematizações.

Pereira (2018) aborda a importância das etapas na construção de um programa. Ter conhecimento sobre a evolução das etapas é um ponto crucial no desenvolvimento de um programa de qualidade. É importante saber analisar a problematização, construir o algoritmo, realizar a implementação, testar e manter o programa em bom funcionamento.

A Figura 3 ilustra um esquema que define as etapas na construção de um programa bem analisado. De acordo com Edelweiss e Livi (2014), as etapas para a construção de um algoritmo são bem dinâmicas, e seguir esses passos faz com que o programa seja de qualidade e atenda às necessidades. As etapas podem ser definidas como a seguir (EDELWEISS; LIVI, 2014).

- Análise do problema: tem como foco a problematização, definindo bem o problema abordado.
- Especificação dos requisitos do problema: realiza uma verificação de quais dados/informações devem ser fornecidos para a entrada e quais resultados devem ser apresentados na saída.
- Construção do algoritmo: etapa em que o algoritmo é construído, considerando a análise realizada na etapa anterior. Nessa etapa, é construído o algoritmo que caracteriza os passos que devem ser tomados para se chegar a um resultado final.
- Validação do algoritmo: verifica se o algoritmo atende à necessidade do programa. Para isso, podem ser realizados testes de mesa, atribuindo

diversos valores possíveis para ver se o algoritmo consegue realizar o processamento.

- Construção do programa: a implementação em uma linguagem de programação (como a linguagem C) é aplicada considerando a estrutura do algoritmo.
- Verificação do programa: é realizada a compilação e depuração do programa. Permite realizar testes (como o teste de mesa), inserindo valores no programa para verificar se o processamento está correto.
- Manutenção: etapa importante que tem como objetivo acompanhar o programa depois de finalizado. Esse acompanhamento diz respeito ao cuidado com eventuais erros que podem surgir, assim como a atualizações e novas funcionalidades que podem ou devem ser inseridas.

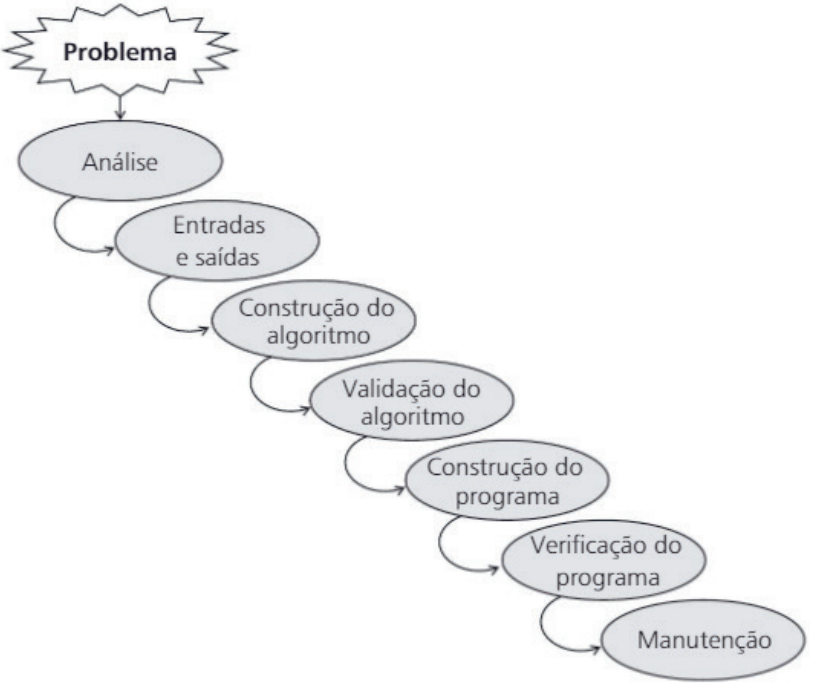

Figura 3. Etapas da construção de um programa.
Fonte: Edelweiss e Livi (2014, p. 16).

Para ilustrar as etapas de construção de um programa, pense na seguinte problematização: um professor de educação física necessita de um programa que realize o cálculo de índice de massa corporal (IMC) de seus alunos para fazer treinos específicos conforme a classificação resultante.

Considerando essa problematização, devemos seguir os passos para a construção do programa. A princípio, é realizada a análise do problema, obtendo resumidamente a definição do problema, que é informar a classificação do IMC.

Na etapa de especificação dos requisitos do problema, são identificadas as entradas e as saídas necessárias para o programa; veja a seguir.

- Entradas: peso e altura da pessoa, digitados via teclado.
- Saídas: o resultado do cálculo do IMC e a classificação correspondente, mostrados na tela.

A etapa de construção do algoritmo pode ser observada no exemplo a seguir. Considerando as análises realizadas nas etapas anteriores, o algoritmo é construído com base em ações que devem ser executadas para se chegar ao objetivo do programa. É importante saber que, nessa etapa, os nomes das variáveis já podem ser informados.

```
Algoritmo – IMC
   {INFORMAR A CLASSIFICAÇÃO DO IMC DE UMA PESSOA}
   Entradas: peso, altura
   Saídas: imc, classificação
   início
     ler (peso, altura) {ENTRADA DO PESO E ALTURA DA PESSOA}
     imc ← peso / (altura * altura) {CALCULA O IMC}
     escrever (imc) {INFORMA O VALOR ADQUITIDO DO IMC}
     se imc < 18.5 então
          escrever "Abaixo do peso"
     senão
          se imc >= 18.5 e < 25 então
              escrever "Peso normal"
          senão
               se imc >= 25 e <30
                    escrever "Acima do peso"
               senão
                    se imc >= 30 e <34
                         escrever "Obeso"
```

```
                    senão "Muito obeso"
    fimse
fim
```

Depois de construído o algoritmo, a próxima etapa (validação do algoritmo) pode ser realizada por meio de testes de mesa, atribuindo diferentes dados para ver o que o algoritmo retorna do processamento. Por exemplo, podemos atribuir os seguintes valores.

- Peso: 60.
- Altura: 1.70.
- IMC = 60 / (1.70 * 1.70).
- IMC = 60 / (2.89).
- IMC = 20.76.
- IMC < 18.5 (Falso).
- IMC >= 18.5 e < 25 (Verdadeiro).
- Classificação "Peso normal".

A etapa seguinte, a de codificação do programa, é a implementação propriamente dita, em que uma linguagem de programação é utilizada. O exemplo a seguir ilustra a implementação do algoritmo IMC na linguagem de programação C. Apesar de o foco deste capítulo não ser a implementação de linguagens de programação, é importante ter uma dimensão de como o algoritmo facilita a interpretação para implementar em qualquer linguagem de programação.

```
Linguagem C – Programa IMC

#include <stdio.h>

int main() {
     int peso;
     float altura, imc;

     printf("Informe seu peso (em kgs):\n");
     scanf("%d", &peso);

     printf("\nInforme sua altura:\n");
     scanf("%f", &altura);
```

```
    imc = peso / (altura * altura);

    printf("\n\n Seu IMC = %.2f", imc);
    printf("\nClassificacao IMC");
    if (imc < 18.5)
       printf(" Abaixo do peso");
    else if ((imc >= 18.5) && (imc < 25))
            printf(" Peso normal");
         else if ((imc >= 25) && (imc < 30))
                 printf(" Acima do peso");
              else if ((imc >= 30) && (imc < 34))
                      printf(" Obeso");
          else
              printf(" Muito obeso");
}
```

A compilação e depuração (verificação do programa) é a etapa em que podemos realizar testes (como o teste de mesa já mostrado) inserindo valores no programa para verificar se o processamento está correto.

Por fim, a manutenção representa o acompanhamento do programa desenvolvido, seja para corrigir eventuais erros, seja para atribuir uma nova funcionalidade, como a de informar o nome da pessoa que está realizando a classificação do IMC.

Independentemente da abordagem realizada, todas têm um esquema que considera etapas que facilitam a construção de um programa de qualidade, tanto para os desenvolvedores quanto para os usuários. Os exemplos trazidos neste capítulo podem ser dimensionados para problemas mais complexos, respeitando as etapas de sua construção.

É essencial saber o quanto os algoritmos são importantes para facilitar a depuração dos programas por parte dos programadores e/ou da equipe de desenvolvimento. Conhecer como a relação do computador com os algoritmos funciona facilita esse entendimento. Já a construção dos algoritmos deve apresentar um conteúdo claro, preciso e não ambíguo. É importante seguir as etapas da construção de um programa para ter sucesso em seus resultados finais, considerando o propósito e a facilidade do desenvolvimento e produzindo um programa de qualidade.

Referências

BERG, A. C.; FIGUEIRÓ, J. P. *Lógica de programação*. 3. ed. rev. e atual. Canoas: Ulbra, 2006.

EDELWEISS, N.; LIVI, M. A. C. *Algoritmos e programação com exemplos em Pascal e C*. Porto Alegre: Bookman, 2014.

MANZANO, J. A. N. G.; OLIVEIRA, J. F. *Algoritmos*: lógica para desenvolvimento de programação de computadores. 28. ed. rev. e atual. São Paulo: Saraiva, 2016.

OLIVEIRA, L. A. H. G. *Introdução à informática*. Natal: Departamento de Computação e Automação, 2004. (Apostila de Introdução à informática, Curso de Engenharia Química, Universidade Federal do Rio Grande do Norte).

PEREIRA, S. L. *Algoritmos e lógica de programação em C*: uma abordagem didática. São Paulo: Saraiva, 2018.

Conteúdo:

sagah