**Palavras do autor**

Diante dessa demanda por profissionais capazes de solucionar problemas, na primeira unidade desse livro, você terá a oportunidade de **conhecer e compreender os fundamentos de algoritmos e das linguagens de programação**. Na segunda unidade, **você aprenderá as estruturas de decisão e repetição.** O conteúdo da terceira unidade abrirá para você uma gama de possibilidades, pois você **conhecerá o que são, quais os tipos e para que servem as funções e recursividade.** Na última unidade, você aprenderá técnicas que permitirão a você **organizar e otimizar seu programa**, por meio das **estruturas de dados com listas, filas e pilhas.**

Seção 1

**Conceitos de algoritmos e programação**

Caro aluno, vale saber que um algoritmo é uma sequência finita de passos que podem levar à criação e execução de uma determinada tarefa com a intenção de resolver uma problemática (FORBELLONE; EBERSPÄCHER, 2005).

Berg e Figueiró(1998) descrevem algoritmos como uma sequência ordenada de passos que deve ser seguida para atingir um objetivo.

O algoritmo deverá ser elaborado nas formas descritas a seguir:

- Linguagem natural.

Segundo Santos (2001), a linguagem natural é uma forma de comuni­cação entre as pessoas de diversas línguas, ela pode ser falada, escrita, gesti­culada entre outras formas de comunicação.

- Diagrama de blocos (fluxograma).

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 especificas a serem executadas pelo computador.

São padronizados pela ANSI;

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

1. Estar atento aos níveis.
2. O diagrama de blocos (fluxograma) deve começar de cima para baixo e da esquerda para direita.
3. Ficar atento para não cruzar as linhas, principalmente as linhas de fluxos de dados.

- Pseudocódigo.

**Pseudocódigo**

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

importante estar atento para algumas regras básicas quando utilizar pseudocódigos:

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

Após os estudos de algoritmos e as suas formas de construções, Manzano, Franco e Villar (2015) colocam em destaque os paradigmas de programação, que são caracterizados pelos paradigmas de programação estruturada, em que o algoritmo éconstruído como sequência linear de funções ou módulo. O outro paradigma é orientado a objetos ou simplesmente programação orientada a objetos, na qual o programador abstrai um programa como uma coleção de objetos que interagem entre si.

De posse dos resultados, os algoritmos podem contribuir de forma signi­ficativa na elaboração de códigos de programação.

Nesse sentido, os algoritmos nortearão você a descobrir qual o melhor percurso para solucionar um problema computacional.

Para melhor entendimento dos algoritmos, é necessário dividi-los em três partes: entrada, processamento e saída.

1. **Entrada:** dados de entrada do algoritmo, no caso do algoritmo para cozimento do arroz, seriam os insumos (ingredientes) necessários para o preparo do arroz.
2. **Processamento:** são os procedimentos necessários para chegar ao resultado final (o cozimento do arroz).
3. **Saída:** resultado que o algoritmo quer chegar após o processamento dos dados de entrada, finalização do arroz (momento que será servido).

O que são variáveis e atribuições: As variáveis, como o próprio nome sugere, consistem em algo que pode sofrer variações,

O que são atribuição (¬), que tem a função de indicar valores para as variá­veis, ou seja, atribuir informação para variável.

valor1 <- 8

nome <- márcio

Pois bem, as variáveis são locais reservados na memória para armazenamento dos dados, cada uma possui um nome próprio para sua identificação.

Agora você vai conhecer como funcionam as constantes. Vamos lá! O próprio nome já sugere, constante é algo que não se altera. Para Schildt (2005), as constantes são consideradas modificadores de tipo de acesso, ou seja, não podem ser alteradas, elas podem ser representadas pelo comando “cons t”.

As constantes também podem ser caracterizadas por qualquer tipo de dados básicos, por exemplo: as constantes do tipo texto são envolvidas por aspas simples (‘) ou aspas duplas, as aspas simples representam um único caractere, por exemplo ‘a’ ou ‘100’ e as aspas duplas caracterizam um conjunto de caracteres, por exemplo “A conversão da temperatura de graus centígrados para graus Fahrenheit”.

**Conceitos de linguagem de programação**

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, essas utilizadas por todos no dia a dia, a linguagem de programação é a comunicação de ideias entre o computador e as pessoas.

“paradigma significa modelo, padrão. No contexto da programação de computadores, um paradigma é um jeito, uma maneira, um estilo de se programar”.

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

**Programação imperativa**: considerada 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 e a manipulação de exceções são seus componentes de programação.

Exemplo de programação imperativa: COBOL, Fortran, C, Ada e Perl.

**Programação orientada a objeto:** também conhecida na computação como (POO), como o próprio nome sugere, é considerado uma coleção de objetos que se inter-relacionam, facilitando assim a programação. São exemplos de POO: vSmalltalk, C++, Java e C#.

**Programação funcional**: caraterizada por possuir atuação matemática, cada uma com um espaço de entrada (domínio) e resultado (faixa). Exemplos de programação funcional: Lisp, Scheme, Haskell e ML.

**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 programação lógica o Prolog.

Tucker (2010, p. 24), “[a] sintaxe de uma linguagem de programação éuma descrição precisa de todos os seus programas gramaticalmente corretos”.

**Componentes de um programa de computador**

Segundo Damas (2016), um programa é uma sequência de instruções de tal forma que permita resolver um determinado problema. Um programa pode ser desenvolvido em módulos distintos e/ou em subprogramas.

Lembrando que para criar um programa de computador será necessário seguir as regras, segundo Aguilar (2011):

1. Definir e analisar o problema a ser solucionado.
2. Criar um algoritmo ou um diagrama de fluxo.
3. Realizar o pseudocódigo.

Para executar um programa é importante estar atento à sequência que os dados percorrem:

1. Entrada de dados: realiza as coletas de dados.
2. Processamento: os dados são transformados em informação.
3. Saída: onde todas as informações geradas pelo processamento de dados são apresentadas em um periférico.
4. Segundo Aguilar (2011), as sintaxes (instrução) deverão ser escritas e armazenadas na memória do computador na mesma ordem que se espera serem executadas, ou seja, **elas podem ser lineares** (executadas sequencial­mente) **e não lineares** (executadas de forma a serem redirecionadas, isto é, uma instrução de bifurcação).

Muito bem, a partir de agora você irá conhecer as sequências de instru­ções para criação de um programa de computador. Vamos lá!

No contexto geral ficaria assim:

1. Início do programa.
2. Definição das variáveis e de possíveis atribuições.
3. Instrução de leitura dos dados.
4. Instrução do formato de escrita.
5. Demais instruções e funções.
6. Fim do programa.

Em algoritmos, a forma de escrever (maiúsculas e minúsculas) não acarretará em erros, porém, em linguagem de programação C é preciso diferenciar as palavras em letras maiúsculas e minúsculas.

operadores aritméticos

1. Operadores binários

+ soma

- subtração

\* multiplicação

/ divisão

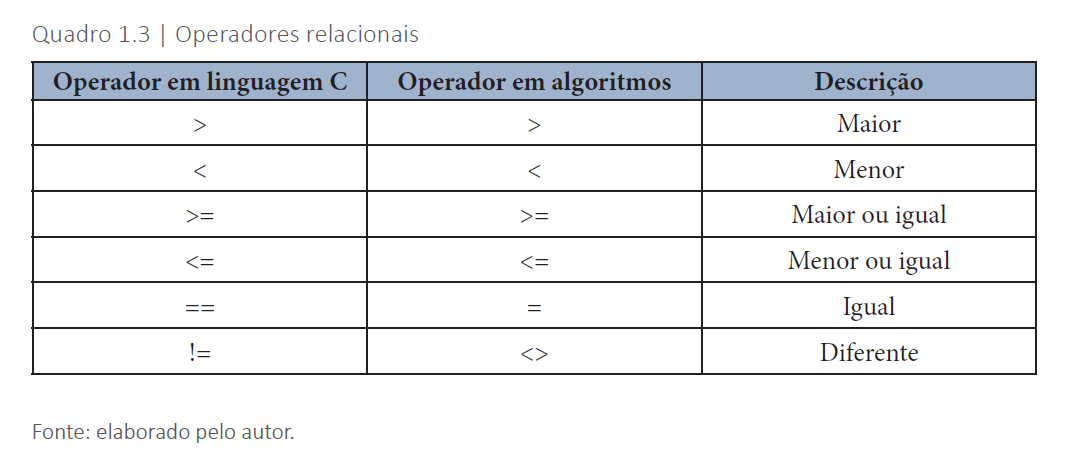
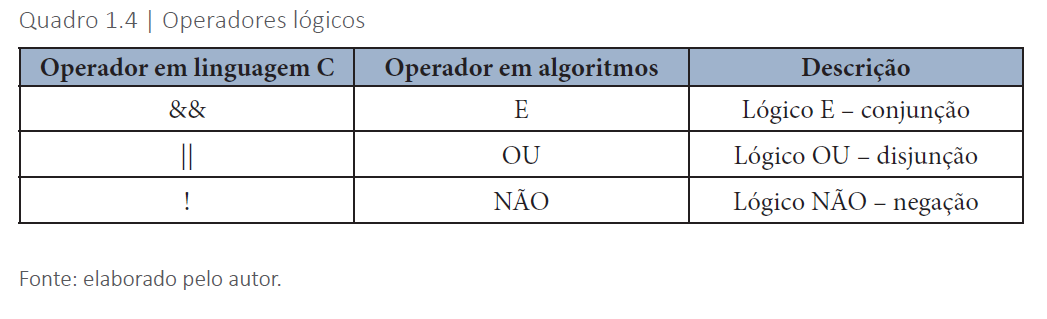
% resto de divisão

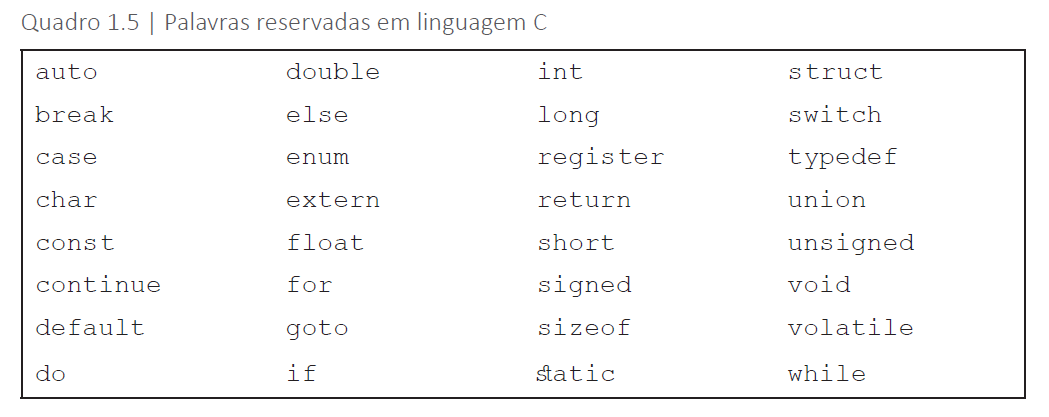
1. Operador unário

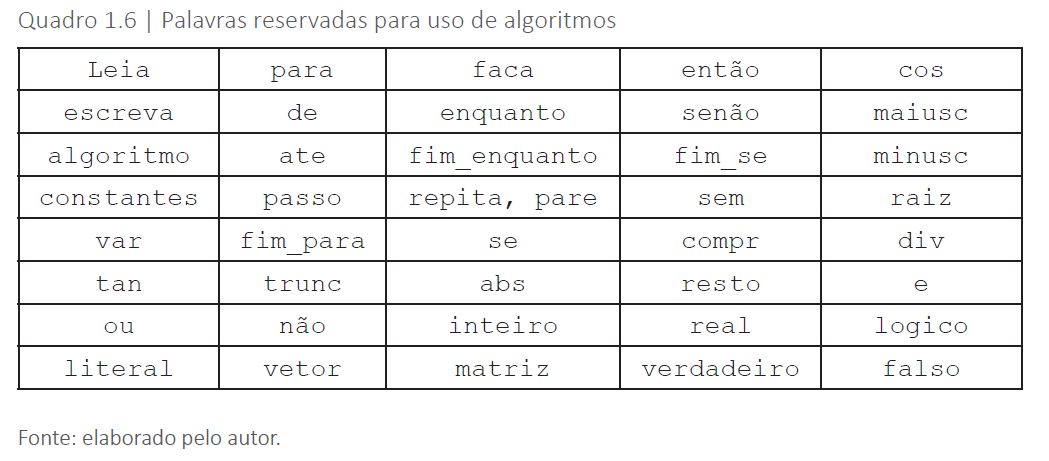
a= -5;

1. a= -a;

Para um programa retornar ao sistema operacional, segundo Manzano, Franco e Villar (2015), será necessário utilizar a instrução retorna zero “return 0”,

1. 
2. 
3. Operador Lógico Não é Unário
4. Operadores Lógicos E e OU são binários;





**Códigos de formatação para função printf()**

