LÓGICA PARA REDES DE COMPUTADORES

CAPÍTULO 1 - COMO SÃO CONSTRUÍDOS OS ALGORITMOS?

Janaide Nogueira de Sousa Ximenes / Fernando Skackauskas Dias

INICIAR

Introdução

Quando os computadores surgiram, eles só podiam ser programados em linguagens de baixo nível, ou seja, em linguagens com características da arquitetura do computador. Isso ocorria por conta da pouca maturidade da ciência da computação e das limitações de *hardware* dos computadores na época em que eles surgiram. Com o avanço da computação, as aplicações foram se tornando cada vez mais complexas e as linguagens de programação passaram a utilizar instruções abstratas, chamadas de linguagem de alto nível.

Os algoritmos não foram idealizados para satisfazer às necessidades somente da computação, eles representam uma ferramenta de raciocínio lógico que pode ser aplicada em diversas tarefas, até mesmo rotineiras, sendo a computação um dos objetos de aplicação.

A lógica de programação, por sua vez, pode ser considerada como uma ciência com características matemáticas que trata das regras para a criação de

programas de computadores, pelas quais a máquina e os humanos conversam. Para estabelecer essa linguagem, precisamos entender: como construir um algoritmo? Existe alguma metodologia para a criação dos algoritmos? Como é possível aplicar a programação para solucionar problemas?

Neste capítulo serão apresentados os tipos de algoritmos, como os fluxogramas e pseudocódigo, e os operadores matemáticos e lógicos, bastante utilizados na construção de algoritmos. Vamos partir dessa abordagem introdutória para descobrir como construir um algoritmo e utilizar estruturas de seleção para solucionar problemas do dia a dia.

Leia com atenção e aproveite o conteúdo. Bons estudos!

1.1 Princípios Básicos de Programação

Vivemos em uma sociedade cercada por soluções tecnológicas, por exemplo, semáforos, diversos aplicativos, *sites*, casas com automação, aplicações de servidores de rede, *e-mails*, entre outros. Essas soluções, tão presentes em nosso dia a dia, são possíveis por meio da programação de computadores. Estas sequências de comandos são essenciais na atualidade.

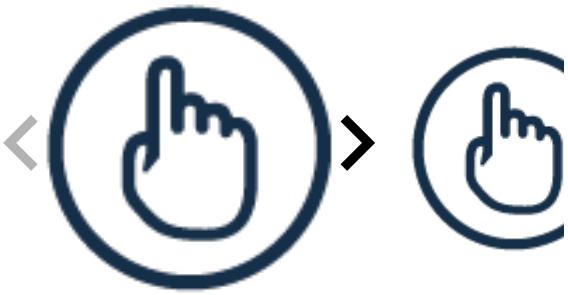
VOCÊ QUER VER?

Filmes como “Estrelas além do tempo” (SCHROEDER, 2016) recordam a corrida espacial entre Estados Unidos e Rússia. O filme conta a história de três mulheres negras na década de 1960 que superaram o preconceito racial e o preconceito contra as mulheres. O filme apresenta o inicio da utilização dos computadores e como era realizada sua programação.

A definição de Holanda (2010), aponta um algoritmo como uma sequência de raciocínios ou operações que oferece a solução de determinados problemas. Na maior parte dos casos, utilizamos a programação para solucionar

problemas do nosso dia a dia, mas, no entanto, podemos solucionar problemas de ordem administrativa, de cálculo de rotas de pacotes na rede, localização, identificação de intrusos na rede, entre outros. Começando sempre de uma necessidade real de automatizar ou solucionar um problema.

Nisso, algumas questões são levantadas, clique na interação a seguir para ver algumas delas.



Como devo iniciar meus algoritmos?

Vamos responder a essas e outras perguntas sobre programação a partir de agora.

1.1.1 Conceitos Iniciais

Para que possamos construir programas em uma determinada linguagem faz-se necessário a definição dos símbolos a serem utilizados e a forma como eles devem ser combinados, a fim de que seja obtido algum resultado. Além dos símbolos, a lógica é essencial para o desenvolvimento de aplicações.

Para Senne (2006), a lógica organiza de maneira metódica o pensar. O que se tem é que a lógica é a essência dos algoritmos e dos códigos já processáveis. A lógica determina o comportamento esperado do programa e as saídas corretas. Por isso, é fundamental conhecer com profundidade a essência daquilo que se tornará um algoritmo e, posteriormente, o programa.

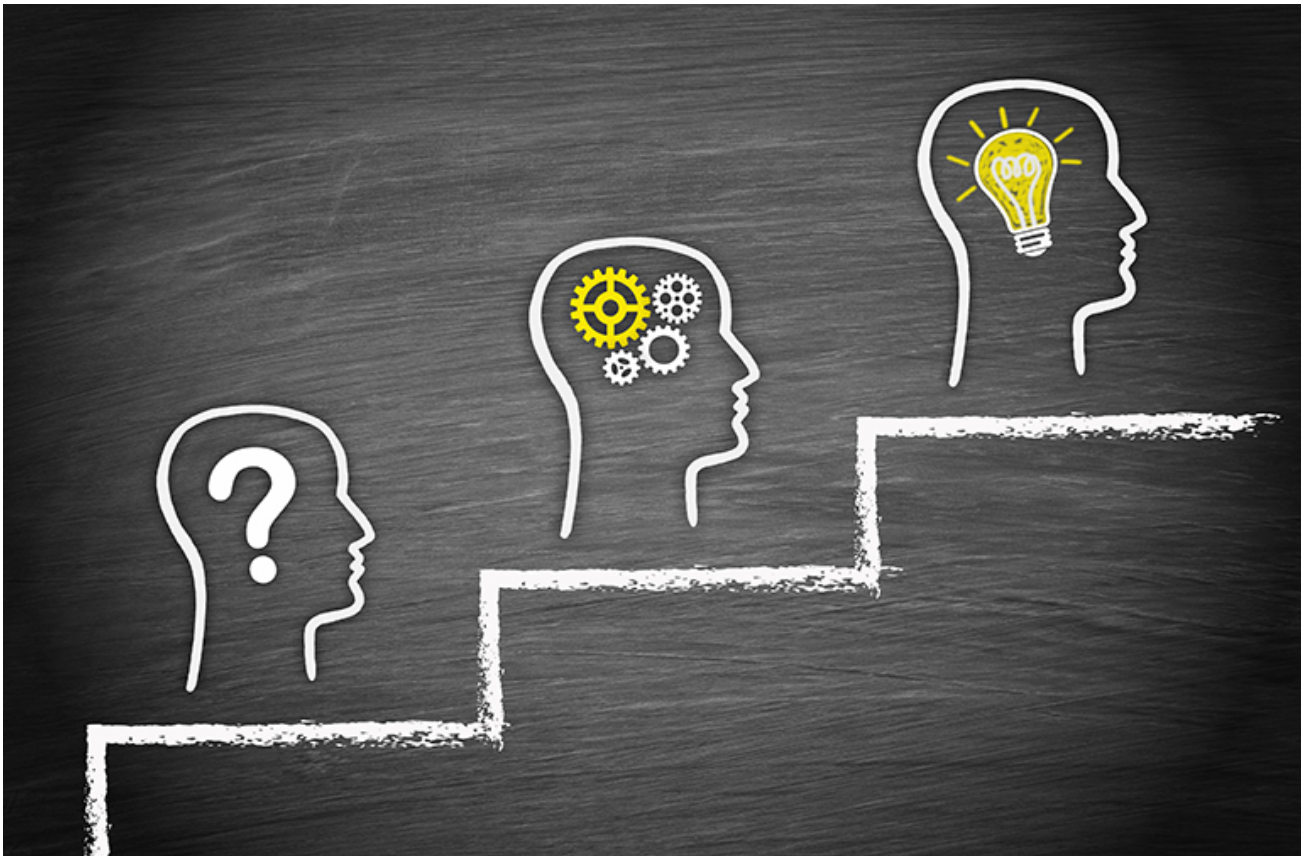


Figura 1 - O algoritmo segue uma sequência lógica de passos ou comandos. Fonte: docstockmedia, Shutterstock, 2018.

Para executarmos os comandos do algoritmo faz-se necessário uma IDE (*Integrated Development Environment*, em português, Ambiente Integrado de Desenvolvimento). A maioria das IDE's existentes suportam mais de uma linguagem de programação. Em algumas linguagens, a ação de executar os comandos do algoritmo também pode ser chamada de compilação. Mas, em outras linguagens é necessário que os comandos sejam traduzidos e depois compilados.

Em linguagens mais antigas, como o Delphi, era necessário terminar cada comando com o símbolo de ponto-e-vírgula. Essa nomenclatura tem-se alterado significativamente nas novas linguagens, como a Python, com sintaxes e características próprias.

1.1.2 Como construir algoritmos

Para criar um algoritmo precisamos seguir uma cadeia lógica. Senne (2006) determina os passos a serem seguidos na construção de um algoritmo. Clique

na interação a seguir para conhecê-los.

Analisar o problema.

Detectar as entradas de dados.

Definir que transformações devem ser feitas pelo algoritmo(processamento).

Identificar as saídas (resultado).

Elaborar o algoritmo com o diagrama de blocos(fluxogramas).

Vamos fazer isso na prática? A seguir, daremos os passos para criar um algoritmo simples para fazer um bolo.

- Bater duas claras em neve.
- Adicionar duas gemas.
- Adicionar uma xícara de açúcar.
- Adicionar duas colheres de manteiga.
- Adicionar duas xícaras de leite de coco.
- Adicionar farinha e fermento.
- Misturar tudo.

- Colocar numa forma.
- Levar ao forno em fogo brando.

Agora, vamos criar um algoritmo para lançamento de nota para um aluno. Primeiro, devemos analisar o nosso problema de lançamento da nota e identificar as entradas, no caso teremos como entrada: nome do aluno, nota e disciplina.

Como processamento, teremos a condição de acessar o sistema, pesquisar o aluno, verificar a condição (aprovado ou reprovado) e concluir o lançamento. Acompanhe a seguir.

Acessar sistema.

Inserir usuário e senha.

Pesquisar nome do aluno.

Pesquisar disciplina.

Verificar condição do aluno.

Se nota do aluno for superior ou igual a 7.0: lançar situação do aluno como aprovado.

Se nota do aluno for inferior a 7.0: lançar situação do aluno como reprovado.

Confirmar lançamento.

Encerrar programa.

Essa sequência lógica é bem simples e compreender os conceitos e definições fundamentais para o desenvolvimento de algoritmo é muito importante para que possamos desenvolver lógicas de algoritmos bem estruturadas e livre de erros. A seguir, vamos estudar os tipos de algoritmos existentes.

1.1.3 Tipo de algoritmos: linguagem natural, fluxograma e pseudocódigo

Os algoritmos podem ser representados de diversas formas, como fluxograma, linguagem natural e até mesmo em pseudocódigo, por exemplo.

Ao utilizar a linguagem natural como uma forma de representação, estamos informando apenas a sequência de ações na linguagem que utilizamos, como o algoritmo que criamos anteriormente para fazer um bolo e um outro para trocar uma lâmpada, por exemplo. Esse tipo de representação pode gerar ambiguidades e imprecisões, como, por exemplo: qual a quantidade de farinha que deve ser adicionada?

Para visualizar melhor o fluxo lógico de um algoritmo, geralmente utilizamos um esquema gráfico que chamamos de fluxograma ou diagrama de blocos. Eles são utilizados para descrever a lógica da solução, sem considerar os detalhes da linguagem de programação ou da interface que será utilizada.

VOCÊ O CONHECE?

Augusta Ada Byron (1815-1852), mais conhecida como Ada Lovelace, escreveu o que é considerado o primeiro algoritmo da história. Sua história é interessante: ela é filha do poeta Lord Byron e sua mãe a incentivou a estudar matemática, para que a filha não fosse influenciada pela “poesia insana” do pai. Em meio a seus estudos, foi convidada pelo matemático britânico Charles Babbage, que precisava de ajuda em sua máquina analítica e acabou desenvolvendo o que, anos depois, foi considerada a primeira programação de um computador (SCHWARTZ, 2006). Depois disso, Ada inventou inúmeras técnicas de programação.

Com o uso de figuras geométricas, os fluxogramas simbolizam estruturas lógicas de sequência, de repetição condicional e o fluxo de sequência

representado por setas direcionais, o que facilita a visualização da solução descrita (SOUZA *et al*., 2000).

Quando utilizamos esse modelo de maneira estruturada, não podemos alterar o fluxo de forma indiscriminada, o que evita os efeitos indesejáveis nas soluções tradicionais em fluxogramas. Existem dois tipos de fluxograma: o diagrama de blocos e o diagrama de Chapin.

Primeiro, vamos observar as formas utilizadas nos fluxogramas em geral, na figura a seguir.

Figura 2 - Representação de uma sequência de operação utilizando um fluxograma. Fonte: MANZANO, 2004, p. 14.

A figura acima representa um modelo de execução de passos sequencial a serem executados por um computador. O símbolo que inicia e finaliza o processamento é chamado de “terminal”. Vale ressaltar que, a utilização deste símbolo é obrigatória no diagrama. Já o retângulo é conhecido como “processamento”, no qual representa de forma genérica qualquer tipo de operação.

É interessante ressaltar que os algoritmos são a semente de um programa que será escrito em uma determinada linguagem. As linguagens de programação

podem ser classificadas em lógica ou imperativa. Segundo Sebesta (2000, p. 734):

> uma linguagem de programação lógica é um exemplo de uma baseada em regras. Em uma linguagem imperativa, um algoritmo é especificado em muitos detalhes, e a ordem de execução específica das instruções ou sentenças deve ser incluída. Em uma linguagem baseada em regras, entretanto, estas são especificadas sem uma ordem em particular, e o sistema de implementação da linguagem deve escolher uma ordem na qual elas são usadas para produzir os resultados desejados.

Portanto, é fundamental para um bom desenvolvedor ter domínio das regras de criação de algoritmo, como também das linguagens de programação e suas classificações.

VOCÊ QUER LER?

Os fluxogramas são utilizados na programação independentemente do nível de complexidade do algoritmo. No artigo “Fluxogramas, diagrama de blocos e de Chapin no desenvolvimento de algoritmos” (TEXERA; FRANCINE; MUNIZ, 2013), você pode conhecer todos os símbolos que podemos utilizar para representar graficamente um algoritmo. Acesse em: <https://www.devmedia.com.br/fluxogramas-diagrama-de-blocos-e-de-chapin-no-desenvolvimento-de-algoritmos/28550 (https://www.devmedia.com.br/fluxogramas-diagrama-de-blocos-e-de-chapin-no-desenvolvimento-de-algoritmos/28550)>.

O diagrama de Chapin tem uma representação específica, que compreende uma identação de blocos. Na figura a seguir vemos uma demonstração do diagrama de Chapin.

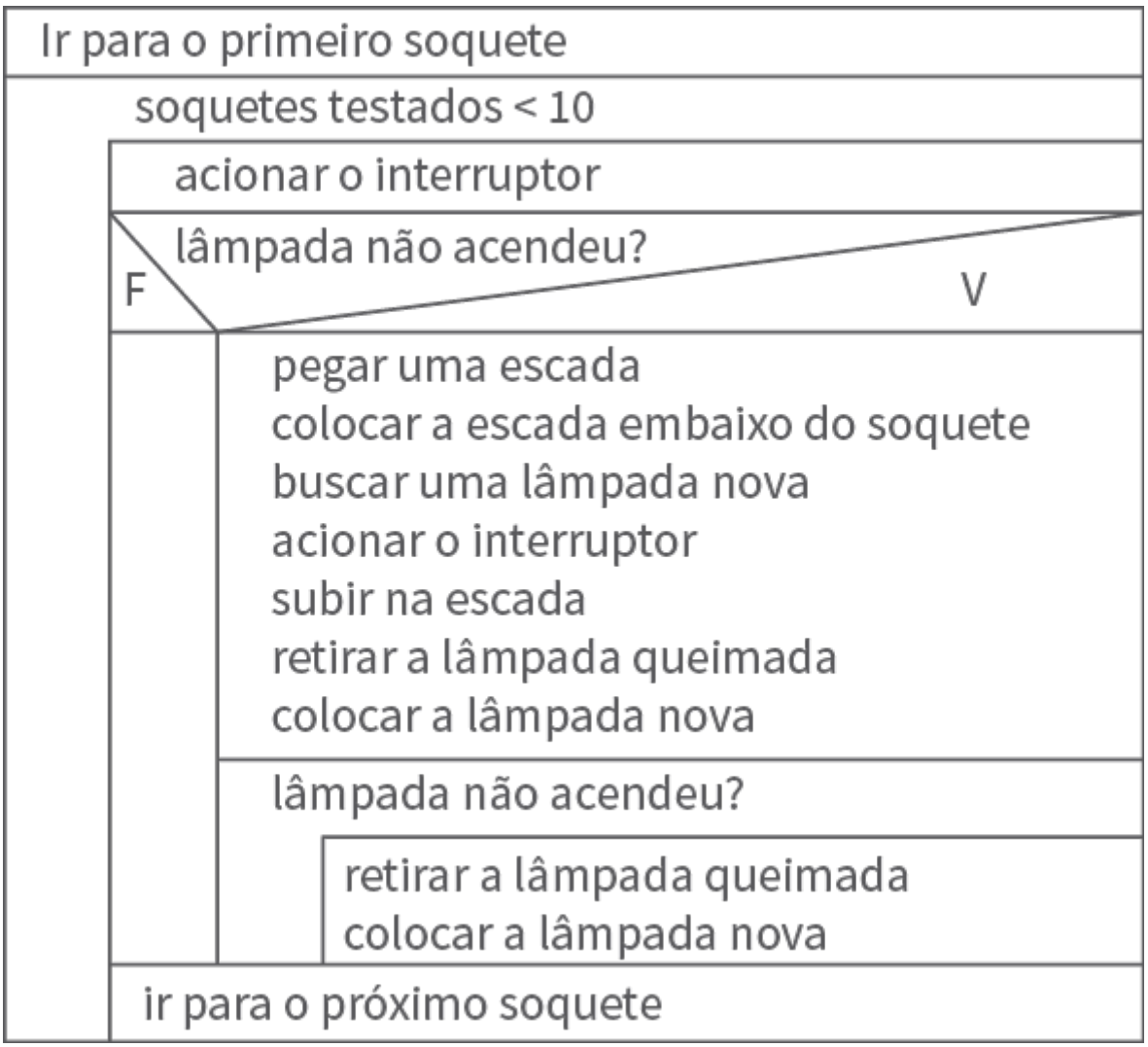


Figura 3 - Representação de uma sequência de operações utilizando um diagrama de Chapin. Fonte: FORBELLONE, 2005, p. 11.

Assim, percebemos que os algoritmos podem ser classificados conforme sua função, sendo natural, pseudocódigo ou em forma de fluxo. Vimos, também, a importância da representação gráfica de um algoritmo para um bom desenvolvimento de um código. A seguir será continuada a exploração do assunto sobre os comandos de entrada e saída.

1.1.4 Comandos de entrada e saída

Os dados a serem processados em qualquer tipo de sistema são categorizados como sendo de entrada e de saída, sendo necessário armazená-los em alguma variável. A entrada geralmente é realizada pelo teclado, mas existem outras formas, como, por exemplo, um programa pode enviar os dados para outro programa.

Figura 4 - Ao informar os dados de entrada, eles são salvos automaticamente na memória. Fonte: aboikis, Shutterstock, 2018.

Em pseudocódigo, o comando de entrada de dados é o **LEIA**. Vamos ver um exemplo: se o algoritmo necessitar do nome e da idade do usuário, considerando que já declaramos a variável, devemos solicitar da seguinte maneira:

Leia(idade);

Leia(nome);

Já os comandos de saída de dados são utilizados para fornecer a uma unidade de saída, o resultado do processamento e mensagens ao usuário ou a outro sistema. Os resultados do processamento podem ser informados por meio de conteúdos de constantes, variáveis e de resultados de expressões lógicas e aritméticas. Podemos ver uma demonstração de uma declaração de variáveis em um algoritmo, a seguir.

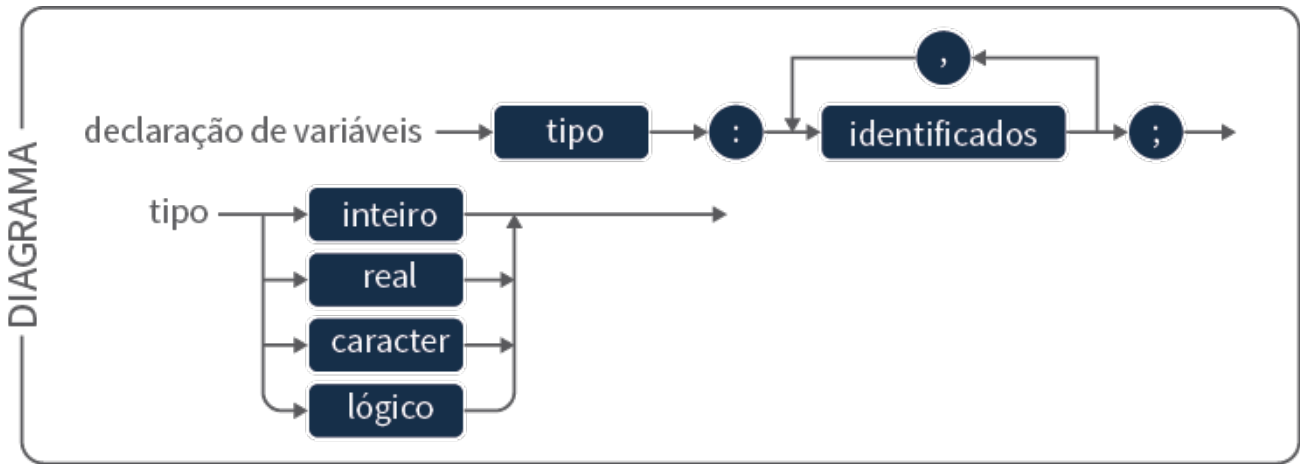


Figura 5 - Representação de uma declaração de variáveis. Fonte: FORBELLONE, 2005, p. 17.

Em pseudocódigo utilizamos o comando **ESCREVA** para fornecer os dados de saída, como por exemplo:

- **Escreva(idade);: Este comando irá retornar o valor contido na variável idade;**
- **Escreva(“Valor inválido!”);: Este comando irá retornar a mensagem “Valor inválido!” ao usuário;**
- **Escreva(“O resultado é: ”, resultado);: Considerando que a variável resultado possui o valor 5. Este comando irá retornar a seguinte mensagem: “O resultado é: 5”.**

VOCÊ QUER LER?

É possível utilizar comandos de entrada e saída de dados em todas as linguagens de programação. O livro “C++ Como Programar” (DEITEL, 2006) apresenta a sintaxe de utilização dos comandos de entrada e saída de dados na linguagem C++. Acesse o livro na biblioteca virtual.

Quando utilizamos o comando ESCREVA para mostrar o valor contido na variável, não podemos utilizar aspas, porém se quisermos enviar uma

mensagem ao usuário, necessitamos das aspas. A utilização das aspas duplas no comando ESCREVA implica na impressão literal da mensagem.

1.2 Introdução à lógica de programação

Para começar, precisamos entender que o computador não entende a linguagem que utilizamos e sim um conjunto de oito "*bits*". Este conjunto é conhecido como "*Byte*". Então, ao digitarmos a letra "a", o computador irá compreender: "01100001", ou se digitarmos o número "3", teremos: "00000011". Quando se desenvolve qualquer algoritmo, é necessário a declaração de variáveis, constantes e expressões que completem o código. Por exemplo, para cálculos matemáticos, são necessárias as variáveis para armazenar cada valor, sendo que existem tipos distintos e com prioridades também distintas.

Você já deve ter se perguntado o que são estas constantes e estas variáveis e para que as utilizamos. Vamos entender isso melhor a seguir.

1.2.1 Tipos de processamento: variáveis; constantes; expressões aritméticas e lógicas

No processo computacional é possível manipular dois tipos de dados contidos na memória. O primeiro são as instruções que gerenciam a forma como os dados são manipulados. O outro tipo são os próprios dados, que devem ser manipulados pelo computador. Os dados podem ser de diversos tipos, cada um dos tipos é representado e processado de maneira diferente.

No momento da definição de um determinado tipo de dado, é reservado um espaço que será alocado, além da classificação dos dados, conforme seu conteúdo. Portanto, variável é a alocação de um espaço de memória para armazenar um tipo de dado. As variáveis são nomeadas para que possam ser referenciadas e alteradas, se houver a necessidade. Já as constantes são valores que não são alterados durante as instâncias, ou seja, ao longo do tempo. De maneira geral, as constantes e variáveis possuem três tipos:

Existem algumas regras na hora de criar uma variável, como, por exemplo, a primeira letra deve ser um caractere alfabético, nenhuma palavra reservada poderá ser utilizada para nomear uma variável, pode conter números desde que inicie com letras, algumas linguagens são *CASE SENSITIVE*, ou seja, diferenciam letras maiúsculas de minúsculas (nome é diferente de NOME).

As expressões aritméticas sempre retornam um valor real ou inteiro. É importante ressaltar que, nas expressões aritméticas, existem regras a serem seguidas: a solução segue da esquerda para a direita, se houver prioridade igual. Sendo a prioridade maior os parênteses mais internos, logo após e com prioridades iguais: a divisão, a multiplicação e o mod (representa o resto de uma divisão por inteiro), e depois, as adições e subtrações, com a mesma prioridade. Por exemplo:

$$3 * 2 - \frac{4}{2} + \sqrt{(2 * 8)} \ / 2$$

$$3 * 2 - \frac{4}{2} + \sqrt{16} \ / 2$$

$$6 - \frac{4}{2} + 4 \ / 2$$

$$6 - 2 + 4 \ / 2$$

$$6 - 2 + 2$$

$$6$$

Temos também as expressões lógicas, ou expressões booleanas, que possuem operadores lógicos ou relacionais e os operandos são do tipo lógico. Por exemplo, considerando a variável A com o valor VERDADEIRO e a variável B com o valor FALSO e o operador lógico AND, temos como resultado da expressão FALSO, pois utilizando o AND só teremos o resultado VERDADEIRO se os dois valores fossem VERDADEIROS.

1.2.2 Operadores matemáticos, funções matemáticas

Ao elaborarmos um algoritmo simples ou um bem mais complexo, necessitamos quase sempre das noções matemáticas. Seja para calcular a média dos alunos, a idade do usuário ou até mesmo o valor total de uma

compra. Os operadores aritméticos, nos auxiliam nestas tarefas, então vamos conhecer os mais utilizados a seguir. Clique para ver.

Soma	na computação e na matemática é representada pelo sinal +: 5+5: na expressão, temos a soma dos valores como resultado 10; X+Y: nessa expressão temos a soma do valor de duas variáveis.
Subtração	na computação e na matemática é representada pelo sinal -: 5-5: na expressão, temos a subtração dos valores como resultado 0; X-Y: nessa expressão temos a subtração do valor de duas variáveis.
Multiplicação	representada na matemática por x ou . e na computação por *: 5*3: na expressão, temos o produto dos valores como resultado 15; X*Y: nessa expressão temos o produto do valor de duas variáveis.
MOD	representada na computação por %, este operador retorna o resto da divisão: 5%2: na expressão, temos o valor 1 como resultado, pois é o resto da divisão; X%Y: nessa expressão temos o resto da divisão do valor de duas variáveis.

Divisão	representada na matemática e na computação por /: 6/3: na expressão, temos a divisão dos valores como resultado 2; X/Y: nessa expressão temos a divisão do valor de duas variáveis.

Além dos operadores citados, as funções matemáticas também auxiliam na hora de programar, como por exemplo:

- **** ou pot(x,y) :** retorna a potência de um número, por exemplo:

2**3 = 2^3 = 8;

Pot(2,3) = 2^3 =8;

- **// ou rad(x):** retorna a radiciação de um número qualquer, por exemplo:

16 // 2 = 4 ($\sqrt{16} = 4$)

Rad(4) = 2 ($\sqrt{4} = 2$)

- **DIV:** retorna o quociente inteiro da divisão, seja:

15 DIV 7 Resulta 2.

- **INT(x):** retorna a parte inteira de um número fracionário, por exemplo:

INT(34.56) Resulta 34.

- **FRAC(x):** retorna a parte fracionária de um número qualquer, por exemplo:

FRAC(34.87) Resulta 87.

- **ABS(x):** retorna o valor absoluto de um número, seja:

ABS(-34) Resulta 34.

Considerando que já conhecemos tanto os operadores quanto as funções matemáticas, vamos calcular:

$(2**3 + 4**2) + ABS(INT\,(-15/2))$

$(8 + 16) + ABS(INT - 7.5))$

$24 + ABS(-7)$

$24 + 7$ = 31

Nesse caso, a prioridade para o computador será os parênteses mais internos, logo após as funções matemáticas e por fim os operadores matemáticos. É muito importante saber a ordem de prioridade considerada pelo computador, pois ela deve ser levada em consideração na hora de elaborar os algoritmos.

1.2.3 Tabela verdade (Operadores Lógicos)

A tabela verdade nos permite demonstrar, verificar ou testar a veracidade de qualquer argumento (ALENCAR FILHO, 2002). O valor obtido de uma relação é sempre lógico.

Portanto, a tabela verdade é uma ferramenta de origem na matemática muito utilizada no raciocínio lógico. O objetivo é verificar a validade lógica de uma proposição composta que é um argumento formado por duas ou mais proposições simples.

Para que esta relação exista, faz-se necessário os operadores relacionais, que podem ser:

Operadores	Representação
Igual	= =
Igual	<> ou !=
Igual	>
Igual	<
Maior ou igual	>=
Igual	<=

Tabela 1 - Todos os operadores lógicos possuem representação gráfica e podem ser utilizados na criação de algoritmos. Fonte: Elaborada pelos autores, 2018.

Em nosso dia a dia fazemos o uso de expressões como: ou, não, e, se... então, e equivalentes. Estas expressões que utilizamos auxiliam na comunicação do nosso pensamento e são chamadas de operadores lógicos. Elas possibilitam a formação de proposições compostas. Como por exemplo as apresentadas na tabela a seguir. Clique para ver.

Q	Ana canta, ou João estuda matemática.
S	Se Ana canta, então João estuda matemática.
P	Ana canta e João estuda matemática.

Segundo Alencar Filho (2002), a lógica matemática adota alguns princípios como regras básicas do pensamento.

- **Princípio da não contradição**: a proposição não pode ser falsa e verdadeira ao mesmo tempo.
- **Princípio do terceiro excluído**: a proposição pode assumir somente dois valores: falso ou verdadeiro, sendo somente possíveis estas condições, sendo, por isso, uma lógica bivalente.

Considerando estes princípios, é possível determinar o valor lógico de uma proposição composta, consultando a tabela verdade. Esta tabela contém todas as possíveis atribuições dos valores lógicos existentes. Agora, considere uma proposição composta cujas proposições simples são p e q e cujos os valores lógicos são os seguintes:

p	q
V	V
V	F
F	V
F	F

Tabela 2 - Os valores lógicos são alternados entre V(Verdadeiro) e F(Falso). Fonte: Elaborada pelos autores, 2018.

Para realizar as operações relacionais, é necessária a utilização dos operadores lógicos que listamos a seguir.

- **Negação**: representado pelo símbolo (~), consideramos a negação de q, a proposição “Não q”, cujo valor é falso(F) quando q é verdadeira. Então a negação pode ser considerada como o oposto do valor lógico da proposição. Por exemplo:

  p: João é dançarino
  ~p: João não é dançarino

  Considerando os valores lógicos, temos: p= V e q = F:
  (I) p e ~ q = V e ~F= V e V= V
  (II) ~p e ~q= ~V e ~F= F e V= F

P	~P
V	F
F	V

Tabela 3 - Tabela verdade da negação da proposição P. Fonte: Elaborada pelos autores, 2018.

- **Conjunção**: Representada pelo(^), podemos chamar de conjunção de proposições. Por exemplo:

  p: João é cantor(V)
  q: Maria comeu maçã(F)
  p^q: V^F = F

- **Disjunção**: é representada pelo ($\bigvee$), podemos considerar que a disjunção de duas proposições, cujo valor lógico é Verdadeira, quando, ao menos uma das proposições, é verdadeira. Por exemplo:

  p: João é cantor(V)
  q: Maria comeu maçã(F)
  p$\bigvee$q: V $\bigvee$ F = V

- **Disjunção Exclusiva**: é representada pelo($\bigvee$), em nossa linguagem o "ou" possui dois sentidos:

  p: Maria é casada ou solteira
  q: João é cantor ou professor

Na proposição p, apenas uma das proposições pode ser verdadeira, "Maria é casada", "Maria é solteira", já que Maria não pode ser casada e solteira. Na proposição q João pode ser cantor e professor. Sendo assim, podemos dizer que a proposição q, o "ou é inclusivo" e na proposição p, o "ou é exclusivo".

P	Q	P V Q	P ⩒ Q	P ⩔ Q
V	V	V	V	F
V	F	F	V	V
F	V	F	V	V
F	F	F	F	F

Tabela 4 - Tabela verdade para os operadores de conjunção, disjunção e disjunção exclusiva. Fonte: Elaborada pelos autores, 2018.

A quantidade de linhas da tabela verdade de uma proposição composta com X proposições simples corresponde a 2^x linhas.

1.2.4 Estrutura sequencial

Ao criarmos algoritmos, são utilizados os fundamentos de bloco lógico, entrada e saída de dados, constantes atribuições, expressões lógicas e aritméticas, bem como também os comandos que, juntamente com as declarações, constituem o conjunto de ações que se espera como resultado do algoritmo. É necessário haver uma relação coerente e lógica entre as definições dos componentes, processos e o fluxo de execução do algoritmo.

VOCÊ SABIA?

A diferença entre o algoritmo descritivo e os algoritmos de sistemas é a linguagem utilizada. O algoritmo descritivo é escrito de uma forma racional compreendida sem o conhecimento de qualquer linguagem. Para o algoritmo de sistemas, é necessário usar uma linguagem de programação que será interpretada pelo computador. Esta última torna o programa executável para o usuário.

A programação estruturada é sequencial, na qual cada instrução é executada logo após a anterior ser executada, como uma espécie de efeito em cascata (FORBELLONE, 2005).

A estrutura de um algoritmo é, basicamente, o INICIO, comando que inicia a execução, a declaração das variáveis que serão necessárias para a solução do problema, logo após, temos os comandos de entrada, saída e processamento e, então, temos o fim, que encerra a execução do algoritmo.

```
INICIO
|       // DECLARAÇÃO DE VARIÁVEIS
|       // COMANDO A;
|       // COMANDO B;
|       // COMANDO C;
|       .....;
|       // COMANDO N;
FIM.
```

Agora que você conhece a estrutura de um algoritmo, vamos elaborar um algoritmo para a seguinte situação: João, é um professor muito dedicado e precisa calcular a média aritmética de seu aluno. Este aluno possui 03 notas do tipo real.

```
INICIO
|       real: nota1, nota2, nota3, media;//Declaração de Variáveis
|       //Processamento
|       escreva(“Digite a primeira nota:”);
|       leia(nota1);
|       escreva(“Digite a segunda nota:”);
|       leia(nota2);
|       escreva(“Digite a terceira nota:”);
|       leia(nota3);
```

```
|       media = (nota1+nota2+nota3)/3;
|       //Saída do algoritmo
|       escreva(“A média é: ”, media);
FIM.
```

Agora, vamos criar um algoritmo para receber do usuário dois valores inteiros e calcular o produto dos mesmos. Precisaremos de duas variáveis do tipo inteiro, além dos comandos para ler o valor digitado, calcular o produto e informar o resultado ao usuário.

```
INICIO
|       inteiro: num1, num2, resultado;//Declaração de Variáveis
|        //Processamento
|       escreva(“Digite o primeiro valor:”);
|       leia(num1);
|       escreva(“Digite o segundo valor:”);
|       leia(num2);
|       resultado = num1*num2;;
|       //Saída do algoritmo
|       escreva(“ O produto é: ”, resultado);
FIM.
```

É fundamental ter sólido conhecimento do problema a ser solucionado, definir as variáveis, bem com os comandos para assim, desenvolver o algoritmo. Outro aspecto importante é saber definir a ordem correta dos comandos, assim o algoritmo terá bons resultados.

1.3 Construção condicional

Em nosso dia a dia, quase sempre utilizamos uma condição para realizar alguma atividade, não é verdade? Por exemplo, “irei ao cinema, se não estiver chovendo” ou “comprarei a casa, se estiver em boas condições”. Forbellone (2005, p. 68) estabelece que “uma estrutura de seleção ou condicional permite a escolha de um grupo de ações (bloco de ações) a ser executado quando determinadas condições, representadas por expressões lógicas ou relacionais, são ou não satisfeitas”. Ao criar um algoritmo para selecionar, por exemplo, a melhor rota para a rede, você deve elencar condições para selecionar a melhor rota, para as tabelas de roteamento. Temos abaixo, uma figura de uma expressão lógica em forma de algoritmo.

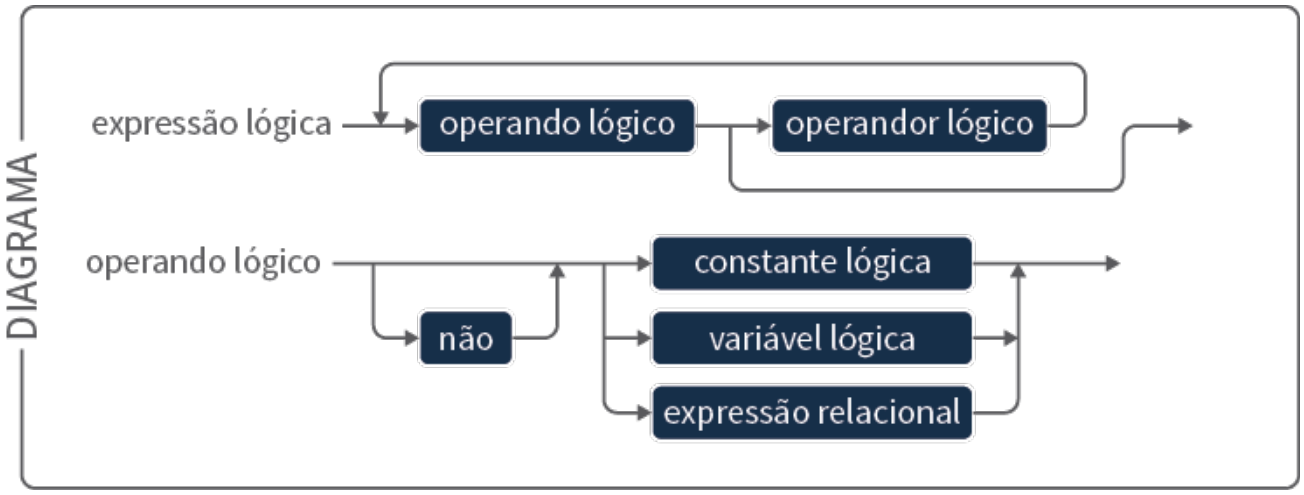


Figura 6 - Representação de expressão lógica em forma de algoritmo. Fonte: FORBELLONE, 2005, p. 21.

Ao estudar os conteúdos apresentados a seguir você poderá compreender melhor a construção de algoritmos condicionais.

1.3.1 Sintaxe da estrutura sequencial: seleção simples e seleção composta

As estruturas de seleção são utilizadas para testar alguma condição e realizar ações de acordo. Estas estruturas podem ser de vários tipos e, neste tópico, abordaremos a seleção simples e a composta. No caso da seleção simples,

temos a estrutura composta por: Se/Então. Para Almeida (2008), uma estrutura de decisão simples é utilizada quando se faz necessário testar uma condição antes de executar um comando. A sintaxe será:

```
INICIO
|       //Declaração de Variáveis
|       //Comandos;
|      Se então
|        //Comandos;
|        //Comandos;
|        //Comandos;
|         FimSe;
|         //Saída do algoritmo;
FIM.
```

A <condição> que será testada deve ser uma expressão lógica, em que o resultado será falso ou verdadeiro. Caso o valor seja VERDADEIRO, os comandos após o então serão executados. Caso o valor seja FALSO, nenhum comando da estrutura de seleção será executado, apenas os comandos que estiverem após o FimSe. Diante disso, vamos analisar o algoritmo a seguir.

```
INICIO
|        inteiro: num;//Declaração de Variáveis
|         //Processamento
|        escreva(“Digite um valor:”);
|        leia(num);
|        Se(num>10) então
|       Escreva(“Valor maior que 10”);
```

```
|       FimSe;
FIM.
```

Este algoritmo utilizou a estrutura condicional simples: Se/Então, para definir se o número inteiro digitado pelo usuário é maior que 10. Caso o valor digitado seja maior que 10, será impresso para o usuário uma mensagem, porém, se o valor for igual ou menor que 10, não será realizada nenhuma ação. A seleção composta possui mais uma opção, como no modelo:

```
INICIO
|       inteiro: num;//Declaração de Variáveis
|        //Processamento
|       escreva(“Digite um valor:”);
|       leia(num);
|       Se(num%2==0) então
|            Escreva(“Valor par”);
|       senão
|             escreva(“Valor ímpar”);
|       FimSe;
FIM.
```

O algoritmo acima possui a função de receber um número inteiro do usuário e verificar se o valor é par ou ímpar. Por exemplo, se o usuário informar o valor 5, o algoritmo irá verificar a condição, que no caso será falsa e executará o comando escreva do SENÃO e retornará a mensagem “Valor ímpar”. Será utilizado o operador % para o cálculo do resto da divisão.

Outra situação problemática que podemos solucionar é quando precisamos calcular a média das notas de um aluno e verificar a sua aprovação. Sendo que, neste cálculo é necessário o uso da média aritmética e, para aprovação do

aluno, a média deve ser maior ou igual a 7.00.

```
INICIO
|       inteiro: nota1, nota2, media;//Declaração de Variáveis
|      //Processamento
|       escreva(“Digite a primeira nota:”);
|       leia(nota1);
|       escreva(“Digite a segunda nota:”);
|       leia(nota2);
|       media = (nota1+nota2)/2;
|       Se(media >= 7) então
|            Escreva(“Aluno aprovado!”);
|         senão
|            escreva(“Aluno Reprovado!”);
|       FimSe;
FIM.
```

Após o recebimento das informações fornecidas pelo usuário, o sistema deve calcular a média em função da ordem de prioridade. Nesse sentido, é fundamental que a soma seja separada da divisão por parênteses. Após o cálculo da média, será verificada a condição de verdadeiro ou falso. Se verdadeiro, serão executados os comandos após a cláusula “então”. Se for falsa, serão executados os comandos após a cláusula “senão”.

1.4 Estrutura de seleção

Agora que você já percebeu como é essencial utilizar as estruturas de seleção para resolver problemas simples do cotidiano, vamos conhecer alguns tipos de

seleção e múltipla escolha. Pense bem, se você estiver criando um algoritmo que necessite de um menu com várias opções e você vai utilizar uma das estruturas de seleção que você conheceu anteriormente, dependendo da quantidade de opções do menu, você terá um código muito extenso.

Podemos solucionar esta situação e alguns questionamentos, como: em quais situações posso utilizar as estruturas de seleção? Todas as estruturas de seleção têm a mesma função?

Vamos entender isso a seguir.

1.4.1 Seleção homogênea, seleção heterogênea e múltipla escolha

Além da seleção simples e composta, existe também a seleção encadeada, conhecida também como seleção aninhada. A seleção encadeada pode ser dividida em: seleção encadeada homogênea e seleção encadeada heterogênea. Chamamos de encadeada, por utilizarmos diversas estruturas de seleção simples e compostas em conjunto.

Na seleção homogênea ocorre quando é possível identificar um padrão, como:

- se – então – se: quando depois de cada então ocorre outro se;
- se – senão – se: quando depois de cada senão ocorre outro se;

Para exemplificar melhor, vamos observar este algoritmo:

```
INICIO
|       inteiro: nota1, nota2, media;//Declaração de Variáveis
|        //Processamento
|       escreva(“Digite a primeira nota:”);
|       leia(nota1);
|       escreva(“Digite a segunda nota:”);
|       leia(nota2);
```

```
|       media = (nota1+nota2)/2;
|        Se(media >= 7) então
|            Escreva(“Aluno aprovado!”);
|          senão Se(media<4)então
|              escreva(“Aluno em Recuperação!”);
|                    senão Se((media>=4) E (media>=0)então
|                          escreva(“Aluno Reprovado!”);
|                   FimSe;
|              FimSe;
|       FimSe;
FIM.
```

A desvantagem da utilização da estrutura homogênea é que, geralmente, são realizados testes desnecessários. Neste caso é previsível que após o SENÃO, haverá um SE e a condição, já na seleção heterogênea, não é possível definir qual será a próxima estrutura a ser utilizada. Analisando o próximo algoritmo, podemos perceber que não há uma sequência lógica, temos estruturas iniciando e terminado em todo o algoritmo.

Agora, vamos descrever o algoritmo para determinação do grau de obesidade, sendo obtido pela relação entre o peso e altura de uma pessoa. O grau de obesidade é determinado pelo índice de massa corpórea (massa=peso/altura2), na tabela abaixo.

Índice Massa Corpórea	Grau de obesidade
<26	Normal
>=26 e <30	Obeso
>=30	Obeso mórbido

Tabela 5 - Relação entre o índice de massa corpórea e grau de obesidade. Fonte: Elaborada pelos autores, 2018.

```
INICIO
|       real: peso, altura, imc;//Declaração de Variáveis
|       escreva(“Digite o peso e a altura:”);
|       leia(peso);
|       leia(altura);
|       imc = peso/(altura*altura);
|       Se imc < 26então
|            Escreva(“IMC: Normal”);
|       Senão
|         Se imc < 30 então
|            Escreva(“IMC: Obeso”);
|            senão
|            Escreva(“IMC: Obesidade Mórbida”);
|         FimSe;
|       FimSe;
FIM.
```

Caso o usuário esteja com o IMC menor que 26, o algoritmo encerra na primeira condição e não verifica as demais, porém se o IMC for maior que 30, o computador verifica todas as condições, até executar o comando do SENÃO. Na figura a seguir, temos uma demonstração em forma de algoritmo da declaração de expressões aritméticas.

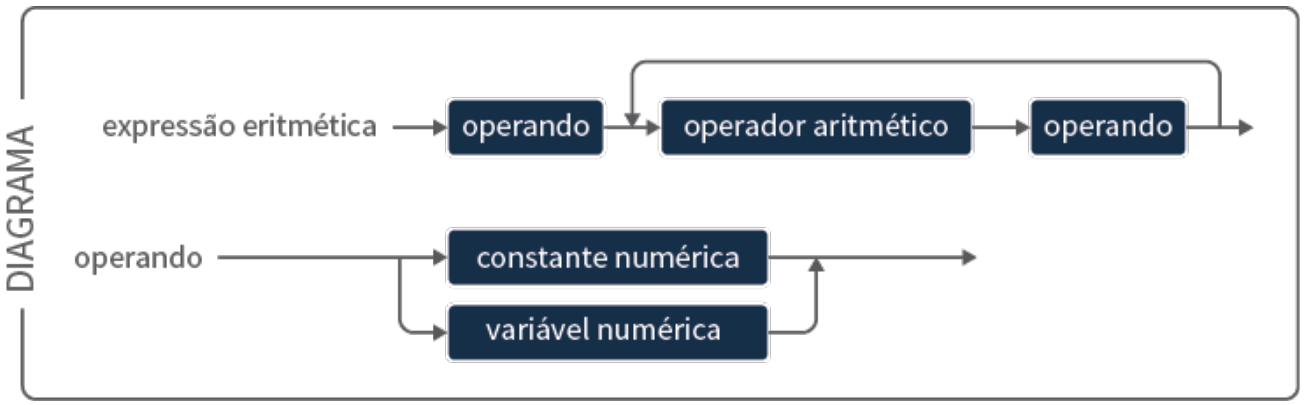


Figura 7 - Representação de expressão aritmética em forma de algoritmo. Fonte: FORBELLONE, 2005, p. 19.

Quando houver a necessidade de ser utilizado um teste de condição em uma única expressão, temos o tipo de decisão de múltipla escolha tipo “Caso/Selecione”. Nesta estrutura pode se ter mais de uma condição a ser testada e um comando diferente para cada uma destas. (ALMEIDA, 2008).

CASO

Uma churrascaria está com uma promoção e os proprietários perceberam a necessidade de investir em um sistema de controle de pedidos, frente à crescente movimentação de clientes. Foi solicitado, portanto, o desenvolvimento de um sistema de controle de pedidos, no qual haja uma integração com os sistemas já existentes na empresa.

Diante da situação, ficou decidido que o sistema de pedidos terá um menu com 15 opções de pratos e mais 20 opções para que o cliente possa montar o seu prato.

Em função dessa demanda, o analista do sistema se viu diante de um problema. Como seria possível elaborar uma seleção pré-definida (15 opções) com a opções de mais 20 opções de livre escolha pelo cliente? Portanto, o analista verificou que poderia ser utilizado uma estrutura de seleção de múltipla escolha para elaborar o menu e após cada opção será acrescentado o valor da opção selecionada ao valor total do usuário.

Ao ir a uma loja, podemos escolher uma ou mais opções dentre as disponíveis, ou seja, posso comprar um ou mais produtos na mesma compra e cupom

fiscal. A múltipla escolha pode ser aplicada em diversas situações no cotidiano. Tendo como exemplo a definição de faixas etárias de pessoas, vamos considerar que, de 0 a 2, será bebê, de 3 a 5, será criança, porém se o usuário informar outro valor, ele retornará a mensagem informando que o valor está fora da faixa etária avaliada.

```
INICIO
|       inteiro: idade;//Declaração de Variáveis
|        //Processamento
|       escreva(“Digite a idade:”);
|       leia(idade);
|       Escolha(idade):
|         Caso 0:
|       Escreva(“Bebê”);
|         Caso 1:
|              Escreva(“Bebê”);
|         Caso 2:
|              Escreva(“Bebê”);
|         Caso 3:
|              Escreva(“Criança”);
|         Caso 4:
|             Escreva(“Criança”);
|         Caso 5:
|             Escreva(“Criança”);
|         CasoContrario
|             escreva(“Valor fora do intervalo”);
|       FimEscolha;
```

FIM.

Ao utilizar o ESCOLHA, você deverá definir qual a variável será verificada o valor. No algoritmo anterior, será a variável idade. O comando CASOCONTRARIO será executado, se o valor que o usuário informar não estiver entre o intervalo de 0 a 5. A utilização do CASOCONTRARIO, não é obrigatória. Mas observe que é muito importante informar ao usuário possíveis erros de digitação ou erros no código do algoritmo.

Assim, podemos ter uma base de como se constrói um algoritmo e entender que a maneira como ele é planejado e implementado faz toda a diferença na hora de sua execução.

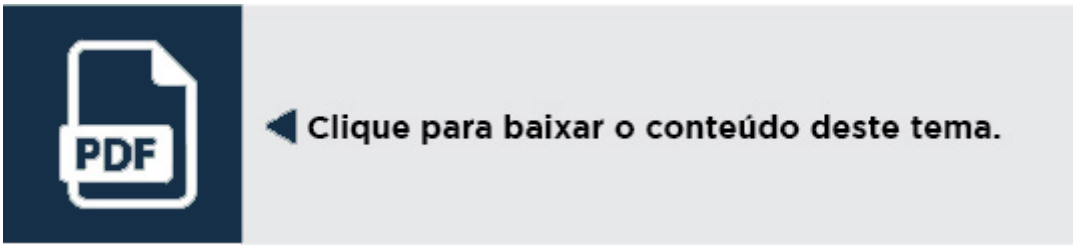
Síntese

Concluímos o capítulo. Conhecemos os princípios básicos da programação estruturada, como fluxogramas, pseudocódigo e os comandos fundamentais de entrada e saída de dados, como também as variáveis, constantes e expressões aritméticas. Percebemos como utilizar as funções matemáticas e as tabelas verdade, que servem de base fundamental para o desenvolvimento de algoritmos com alto nível de qualidade.

Neste capítulo, você teve a oportunidade de:

- identificar os princípios básicos das formas de construção de algoritmos e desenvolver algoritmos sequenciais;
- conhecer os modelos de fluxogramas, além de conhecer algumas estruturas em pseudocódigo;
- identificar os princípios teóricos e práticos referentes ao desenvolvimento de algoritmos;
- identificar, na situação problema, a possibilidade de aplicar uma estrutura de decisão.
- aplicar as estruturas de decisão para solucionar situações problemas;
- aplicar as estruturas de seleção simples, compostas, homogêneas,

heterogêneas e de múltipla escolha para solucionar diversos problemas no cotidiano;
- analisar qual estrutura de seleção é mais indicada para ser aplicada em cada situação.



Bibliografia

ALMEIDA, M. **Curso essencial de lógica de programação**. São Paulo: Universo dos Livros, 2008.

DEITEL, H. M. **C++ Como Programar**. 5. ed. São Paulo: Pearson, 2006.

DEITEL, P.; DEITEL, H. **C Como Programar**. 6. ed. São Paulo: Pearson Prentice Hall, 2011.

ALENCAR FILHO, E. **Iniciação à lógica matemática**. São Paulo: NBL Editora, 2002.

FORBELLONE, A. L. V. **Lógica de programação - A construção de algoritmos e estruturas de dados.** 3. ed. São Paulo: Prentice Hall, 2005.

HOLANDA, A. B. *et al*. **Dicionário Aurélio**. Salvador: Positivo Editora, 2010.

LOPES, A.; GARCIA, G. **Introdução à programação**: 500 algoritmos resolvidos. Rio de Janeiro: Elsevier Brasil, 2016.

MANZANO, J. A. N. G. Revisão e Discussão da Norma ISO 5807-1985 (E) Proposta para Padronização Formal da Representação Gráfica da Linha de Raciocínio Lógico Utilizada no Desenvolvimento da Programação de Computadores a ser Definida no Brasil. **Revisa eletrônica Thesis**. São Paulo: Faculdade Cantareira, ano, v. 1, p. 1-31, 2004.

SANTIAGO, R.; DAZZI, R. L. S. Ferramentas que auxiliam o desenvolvimento da lógica de programação. **XII Seminário de Computação**. Universidade Regional

de Blumenau, 5 a 8 de agosto de 2003. Disponível em: <http://www.inf.furb.br/seminco/2003/artigos/118-vf.pdf (http://www.inf.furb.br/seminco/2003/artigos/118-vf.pdf)>. Acesso em: 19/12/2018.

SCHWARTZ, J. *et al*. Mulheres na informática: quais foram as pioneiras. **Cadernos Pagu**, v. 27, n. 1, p. 255-278, 2006.

SEBESTA, R. W. **Conceitos de Linguagem de Programação**. 4. ed. Porto Alegre: Bookman, 2000.

SENNE, E. L. F. **Primeiro Curso de Programação em C–**. 3. ed. Florianópolis: Visual Books, 2006.

SCHROEDER, A. **Estrelas além do tempo**. Direção: Theodore Melfi. Produção: Theodore Melfi; Peter Chernin; Pharrel Williams. Twentieth Century Fox. Estados Unidos, 2016.

SOUZA, E. M. S. *et al*. Reavaliando o ensino de algoritmos. **I Simpósio Catarinense de Computação**, v. 3, p. 69-79, 2000.

TEXERA, G. G.; FRANCINE, P; MUNIZ, J. **Fluxogramas, diagrama de blocos e de Chapin no desenvolvimento de algoritmos**. Portal Devmedia, publicado em 2013. Artigo desenvolvido sob orientação do Prof. Juliano Schimiguel. Disponível em: <https://www.devmedia.com.br/fluxogramas-diagrama-de-blocos-e-de-chapin-no-desenvolvimento-de-algoritmos/28550 (https://www.devmedia.com.br/fluxogramas-diagrama-de-blocos-e-de-chapin-no-desenvolvimento-de-algoritmos/28550)>. Acesso em: 19/12/2018.