

ALGORITMOS E PROGRAMAÇÃO

Me. Ricardo Zanni Mendes da Silveira

INICIAR

introdução

Introdução

Nesta unidade serão abordados os princípios básicos das formas de compreensão e de construção de algoritmos para o aprendizado dos tipos de algoritmos, desde a linguagem natural, passando pelo fluxograma até chegar no pseudocódigo, em que serão descritos detalhes dos comandos de entrada e saída, os tipos de processamentos, os tipos de dados, variáveis e constantes, bem como suas formas de serem apresentadas ao pseudocódigo e à linguagem de programação. Serão estudados, também, as expressões lógicas e aritméticas, os operadores matemáticos e lógicos, a construção da tabela verdade e a estrutura de um algoritmo, que irão proporcionar ao aluno o aprendizado de conceitos da lógica de programação na resolução de problemas.

Introdução à Lógica de Programação

É muito comum associarmos a palavra lógica apenas à matemática, mas esse termo normalmente está relacionado à coerência e à racionalidade (FORBELLONE, 2005). A lógica pode ser relacionada com a expressão “correção do pensamento”, pois uma de suas responsabilidades é determinar quais operações são válidas e quais não são (FORBELLONE, 2005).

De acordo com Forbellone (2005), a lógica também pode ser interpretada como a “ciência das formas do pensamento”, uma vez que a forma mais complexa do pensamento é o raciocínio, portanto, a lógica estuda a correção do raciocínio.

A lógica de programação utiliza-se do uso correto das leis do pensamento e dos processos de raciocínio na programação de computadores, com o objetivo de racionalizar e desenvolver técnicas que auxiliem na produção de soluções logicamente válidas e coerentes e que resolvam com eficiência os problemas que se deseja programar (FORBELLONE, 2005).

Segundo Forbellone (2005), o principal objetivo do estudo da lógica de programação é a construção de algoritmos válidos e compreensivos.

Conceitos Iniciais

Os seres humanos sempre estiveram determinados em desenvolver máquinas que os ajudassem com suas tarefas cotidianas, com o objetivo de economizar tempo e minimizar as dificuldades. No meio dessas máquinas, o computador vem se destacando por ser um dispositivo flexível, veloz e seguro (ASCENCIO; CAMPOS, 2012).

Atualmente, o computador é uma ferramenta essencial em diversos setores da sociedade. São apresentados nos mais diversos formatos, desde o computador de mesa, *desktops*, portáteis, *notebooks* e *tablets*, até os supercomputadores que realizam cálculos exorbitantes (GUEDES, 2014). Mas, sozinho, o computador não tem nenhuma iniciativa e precisa receber instruções claras e detalhadas de como e em quais ocasiões ele deve realizar o processamento dos dados (ASCENCIO; CAMPOS, 2012; GUEDES, 2014). Portanto, é necessário programá-lo para que ele possa executar as tarefas e solucionar problemas (GUEDES, 2014).

De acordo com Guedes (2014), para que o computador possa executar tarefas, é necessário efetuar uma programação com uma sequência bem definida de instruções, conhecida como **algoritmo**.

Um algoritmo é classificado como uma sequência de etapas que devem ser realizadas para alcançar um determinado objetivo (GUEDES, 2014). Segundo Ascencio e Campos (2012), um algoritmo é a descrição de uma sequência de passos que deve ser seguida para efetuar uma tarefa e é caracterizado por ser uma sequência finita de instruções bem definidas, cuja execução, em tempo finito, resolve um problema computacional.

É muito comum em nosso dia a dia nos depararmos com diversos exemplos de algoritmos. Um exemplo é a receita de bolo, em que está descrito um conjunto de ingredientes essenciais, com suas respectivas quantidades e uma sequência de passos que devem ser executados para que o resultado final seja alcançado com sucesso (GUEDES, 2014).

De acordo com Guedes (2014), cada tarefa que realizamos no nosso dia a dia é um algoritmo, que atua como uma receita, obedecendo às etapas necessárias para chegar ao resultado final.

Podemos observar no exemplo a seguir a execução de um algoritmo comumente utilizado em nosso dia a dia (ASCENCIO; CAMPOS, 2012).

ALGORITMO – Fazer um sanduíche

Passo 1: Pegar o pão.

Passo 2: Cortar o pão ao meio.

Passo 3: Pegar a maionese.

Passo 4: Passar a maionese no pão.

Passo 5: Pegar e cortar a alface e o tomate.

Passo 6: Colocar a alface e o tomate no pão.

Passo 7: Pegar o hambúrguer.

Passo 8: Fritar o hambúrguer.

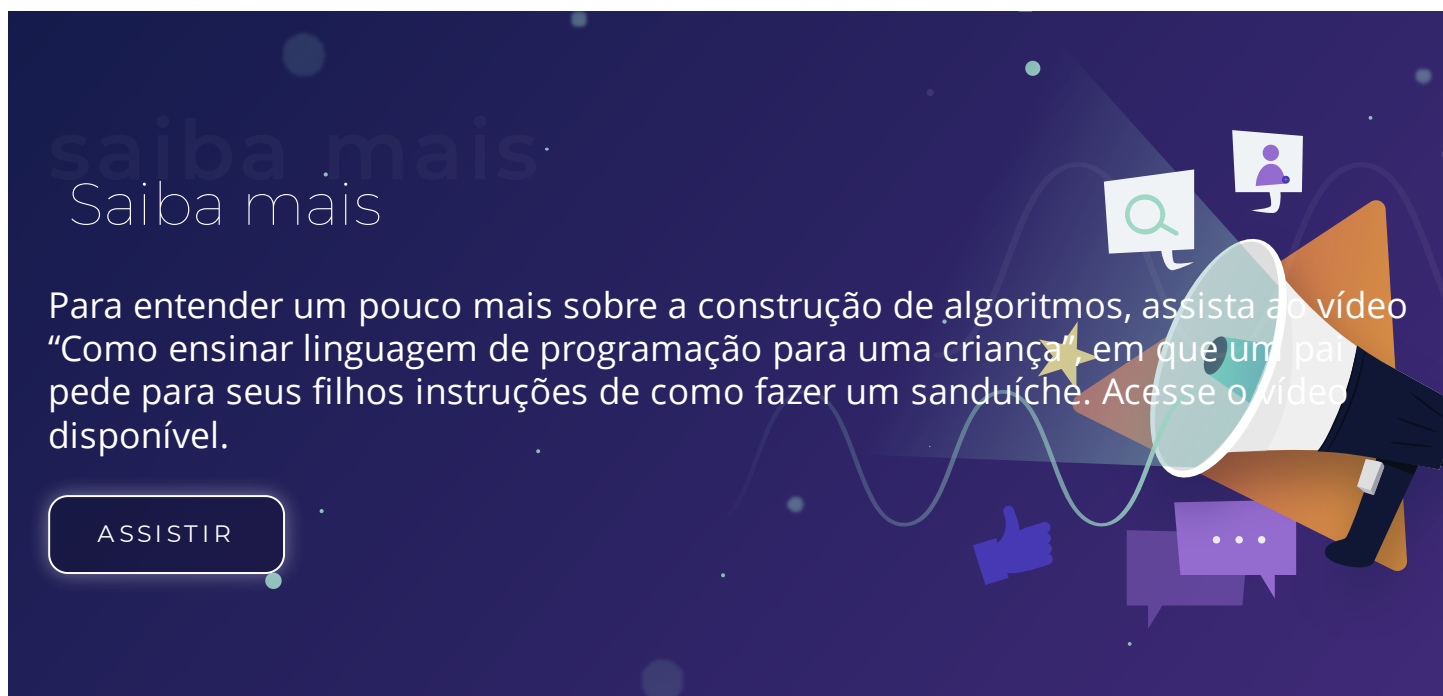
Passo 9: Colocar o hambúrguer no pão.

reflita
Reflita

É fundamental entender que todo algoritmo é finito!



Os algoritmos podem ser executados de maneiras diferentes, assim como os problemas podem ter diversas soluções, mas sempre alcançando o mesmo resultado. Sendo assim, podem existir diversos algoritmos para solucionar o mesmo problema (ASCENCIO; CAMPOS, 2012; GUEDES, 2014).



saiba mais

Saiba mais

Para entender um pouco mais sobre a construção de algoritmos, assista ao vídeo "Como ensinar linguagem de programação para uma criança" em que um pai pede para seus filhos instruções de como fazer um sanduíche. Acesse o vídeo disponível.

ASSISTIR

Como Construir Algoritmos

Um algoritmo tem por finalidade fazer a representação mais fiel do raciocínio envolvido na Lógica de Programação, desse modo, nos permite absorver uma sequência de detalhes computacionais (FORBELLONE, 2005).

De acordo com Forbellone (2005), a construção de algoritmos é importante, pois, uma vez compreendida uma solução algorítmica para um determinado problema, esta pode ser traduzida para qualquer linguagem de programação.

Segundo Ascencio e Campos (2012), para a construção de qualquer tipo de algoritmo, é preciso seguir os passos descritos a seguir:

- a. compreender completamente o problema a ser resolvido e destacar os pontos mais importantes e os objetos que o compõem;
- b. definir os dados de entrada, isto é, quais dados serão fornecidos e quais objetos fazem parte do cenário do problema;
- c. definir o processamento. Quais operações serão efetuadas e quais serão as restrições para essas operações. A responsabilidade de transformar os dados de entrada em dados de saída e de verificar quais objetos são responsáveis pela atividade é atribuição do processamento;
- d. definir os dados de saída;
- e. construir o algoritmo utilizando um dos tipos mais comuns de algoritmos;
- f. testar o algoritmo por meio de simulações.

Na prática, qualquer pessoa amparada na própria experiência é capaz de resolver problemas, entretanto, um programa de computador não tem conhecimento prévio e não adquire experiência própria na solução de problemas, o que ocasiona que devemos determinar em detalhes todas as ações que ele deve executar, prevendo as dificuldades e as maneiras de superá-las para que se consiga a solução do problema. Todo esse processo é realizado pelos programadores, também conhecidos como construtores de algoritmos (FORBELLONE, 2005).

Tipos de Algoritmos: Linguagem Natural, Fluxograma e Pseudocódigo

Um algoritmo é uma linha de raciocínio que pode ser descrita de diversas maneiras, de forma textual ou gráfica, e cada uma dessas técnicas tem suas vantagens e desvantagens, mas que permitem um nível elevado de clareza em relação ao fluxo de execução (FORBELLONE, 2005).

De acordo com Ascencio e Campos (2012), os três tipos de algoritmos mais utilizados são: linguagem natural, fluxograma e pseudocódigo.

Linguagem Natural

Linguagem natural ou descrição narrativa consiste em analisar o problema e escrever utilizando uma linguagem natural, por exemplo, a língua portuguesa, os passos para sua resolução (ASCENCIO; CAMPOS, 2012).

A vantagem de utilizar a descrição narrativa é que não há necessidade de aprender nenhum conceito novo, e sua desvantagem é que a linguagem natural pode ser interpretada de várias maneiras, o que pode dificultar a transcrição do algoritmo para uma linguagem de programação (GUEDES, 2014).

Exemplo de algoritmo em linguagem natural para a multiplicação de dois números:

Passo 1: Receber os dois números que serão multiplicados.

Passo 2: Multiplicar os dois números.

Passo 3: Mostrar o resultado da multiplicação dos dois números.

Fluxograma

O algoritmo baseado em fluxograma consiste em interpretar o enunciado do problema e escrever os passos a serem seguidos para a solução do problema utilizando símbolos gráficos predefinidos (ASCENCIO; CAMPOS, 2012).

De acordo com Guedes (2014), uma das vantagens em se utilizar símbolos gráficos é que estes são mais simples de compreender em relação aos textos, mas é necessário aprender a simbologia, que não permite detalhes mais precisos, o que dificulta a transcrição do algoritmo para uma linguagem de programação e problemas complexos resultam em um fluxograma muito amplo, o que torna difícil a visualização.

Exemplo de algoritmo em fluxograma para a multiplicação de dois números:

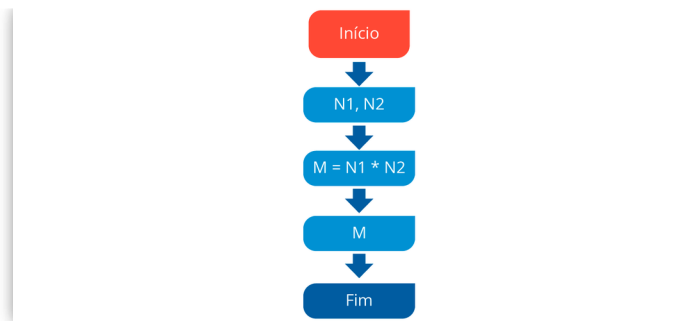


Figura 1.1 – Algoritmo representado em fluxograma
Fonte: Adaptada de Ascencio e Campos (2012, p. 4).

A figura a seguir ilustra o conjunto de símbolos utilizados em fluxogramas.

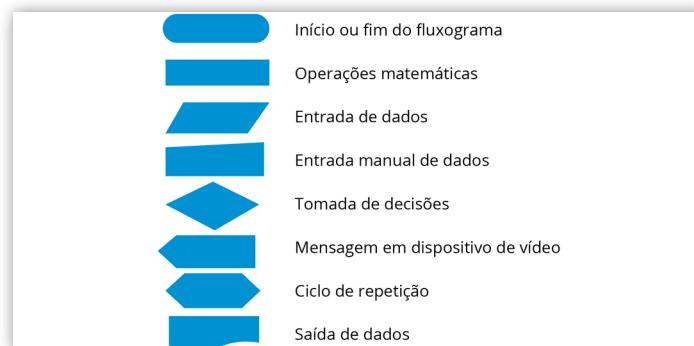


Figura 1.2 – Simbologia utilizada em fluxogramas
Fonte: Adaptada de Guedes (2014, p. 7).

Pseudocódigo

Também conhecido como português, o pseudocódigo consiste em interpretar o enunciado do problema e escrever os passos a serem seguidos para sua resolução por meio de regras predefinidas (ASCENCIO; CAMPOS, 2012).

No pseudocódigo, a transcrição do algoritmo para qualquer linguagem de programação é quase imediata, mas é necessário aprender as regras do padrão do pseudocódigo utilizado. Na prática, é o mesmo que desenvolver o programa aplicativo para depois reproduzir para uma linguagem de programação (ASCENCIO; CAMPOS, 2012).

O pseudocódigo é amplamente utilizado pela maioria dos programadores para desenvolver seus algoritmos. É mais comum utilizar o pseudocódigo por ser o tipo de algoritmo mais próximo de uma linguagem usual (GUEDES, 2014).

Exemplo de algoritmo em pseudocódigo para a multiplicação de dois números:

Algoritmo

Declare N1, N2, M Numérico

Escreva "Digite dois números"

Leia N1, N2

$M \leftarrow N1 * N2$

Escreva "Multiplicação = ", M

Fim_Algoritmo.

O mesmo exemplo em programação C:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int n1, n2, M;
```

```
    printf("insira o primeiro numero\n");
```

```
    scanf("%d\n",&n1);
```

```
    printf("insira o segundo numero\n");
```

```
    scanf("%d\n",&n2);
```

```
    M=n2*n1 ;
```

```
    printf("O resultado é %d\n",M);
```

```
}
```

Comandos de Entrada e Saída

Os algoritmos necessitam ser abastecidos com dados provenientes do meio externo para que consigam realizar as operações e os cálculos que são fundamentais para almejar o resultado necessário. Sendo assim, são utilizados os comandos de entrada e saída (FORBELLONE, 2005).

O comando de entrada em algoritmos é utilizado para receber os dados digitados pelo usuário e que serão armazenados em variáveis (ASCENCIO; CAMPOS, 2012), ou seja, a finalidade desse comando é atribuir o dado a ser fornecido para uma variável identificada (FORBELLONE, 2005). No pseudocódigo, esse comando é representado pela palavra LEIA (ASCENCIO; CAMPOS, 2012).

Exemplo:

LEIA X

Nesse caso, um valor digitado pelo usuário será armazenado na variável X. Em C será como o exemplo a seguir:

```
scanf("%d",&X);
```

Quando o algoritmo mostra os dados que calculou, como uma forma de apresentar a resposta ao problema que solucionou, utiliza-se o comando de saída ESCREVA, cuja finalidade é exibir o conteúdo da variável identificada (FORBELLONE, 2005). Esse comando é utilizado para mostrar os dados na tela (ASCENCIO; CAMPOS, 2012).

Exemplo:

ESCREVA X

Exibe o valor armazenado na variável X. Em linguagem C é feito como o exemplo a seguir:

```
printf("insira o primeiro numero\n");
```

atividade

Atividade

Esse tipo de algoritmo consiste em interpretar o enunciado do problema e escrever os passos a serem seguidos para sua resolução por meio de regras predefinidas. Assinale a alternativa que mais se adequa com as características desse tipo de algoritmo.

ASCENCIO, A. F. G.; CAMPOS, E. A. V. **Fundamentos da Programação de Computadores**: algoritmos, Pascal, C/C++ (padrão ANSI) e Java. 3. ed. São Paulo: Pearson Education do Brasil, 2012.

- ☐ a) Linguagem de programação.
 - ☐ b) Linguagem natural.
 - ☐ c) Fluxograma.
 - ☐ d) Pseudocódigo.
 - ☐ e) Lógica de programação.
-

Variáveis, Operadores e Constantes

Quando o assunto é programação, esbarramos em dois tipos de valores: os **variáveis**, que se alteram conforme determinadas condições e que exigem que o algoritmo esteja preparado para atender essas alterações, e os valores **constantes**, que não se alteram, permanecem sempre os mesmos, não importando a condição. Também são utilizados as operações e os operadores, que caracterizam as expressões de cálculo, condição, atribuição e comparação (GUEDES, 2014).

Tipos de Processamento: Variáveis, Constantes, Expressões Aritméticas e Lógicas

Variáveis

Nos algoritmos, são utilizadas as variáveis para representarem valores conhecidos e desconhecidos. Esses valores são utilizados na resolução de um problema e podem ser alterados de acordo com a condição. Sendo

assim, é possível dizer que as variáveis armazenam dados de uma forma temporária que serão utilizados durante o processamento do programa para a resolução do problema em questão (GUEDES, 2014).

Um exemplo do nosso cotidiano que podemos utilizar para entender uma variável, é fazermos uma analogia de uma variável no estacionamento da faculdade. Os alunos, ao chegarem no estacionamento para guardar seu veículo, não possuem uma vaga fixa, ou seja, onde estiver vago eles irão estacionar. Esse é um exemplo de uma variável, pois nem todos os dias eles irão estacionar na mesma vaga. Em termos computacionais na programação, poderíamos declarar uma variável da seguinte forma:

caractere : vaga; // nesse caso caractere é o tipo da variável, ou seja, ela só poderá armazenar textos.

Quando um dado tem a possibilidade de ser alterado durante a execução do algoritmo ele é considerado uma variável (FORBELLONE, 2005). Uma variável representa uma posição de memória e tem nome e tipo e seu conteúdo pode variar ao longo do tempo de execução do programa, e, embora uma variável possa assumir valores diferentes, ela pode armazenar somente um valor de cada vez (ASCENCIO; CAMPOS, 2012).

Constantes

Quando um dado não sofre nenhuma alteração no decorrer da execução do programa, ou seja, quando seu valor se mantém constante do início até o fim da execução do programa, esse dado é considerado uma constante (FORBELLONE, 2005).

Vamos fazer uma analogia igual, igual a utilizada em uma variável, como exemplo de uma constante no estacionamento da faculdade. A diretoria, ao chegar no estacionamento para guardar seu veículo, possui uma vaga fixa, ou seja, seu veículo sempre será estacionado na mesma vaga. Esse é um exemplo de uma constante, pois, nesse caso, todos os

dias eles irá estacionar na mesma vaga. Em termos computacionais na programação, poderíamos declarar uma constante da seguinte forma:

caractere : vaga = diretoria; // nesse caso caractere é o tipo da constante, ou seja, ela só poderá armazenar textos.

Neste exemplo a constante recebe uma vaga que pertence à diretoria, poderíamos ainda fazer da seguinte forma vaga =36 (neste caso a vaga 36 pertence a um determinado diretor.

Expressões Aritméticas e Lógicas

Quando os operadores são aritméticos e os operandos são constantes ou variáveis do tipo numérico, temos, então, uma expressão aritmética (FORBELLONE, 2005). E quando temos um conjunto de símbolos que representam as operações básicas da matemática, temos os operadores aritméticos (FORBELLONE, 2005).

As expressões lógicas são aquelas cujos operadores são lógicos ou relacionais e os operandos são relações, variáveis ou constantes do tipo lógico (FORBELLONE, 2005).

Operadores: Matemáticos, Funções Matemáticas

Os operadores aritméticos são utilizados para a realização de cálculos matemáticos e são compostos por um conjunto de símbolos utilizados nas operações básicas da matemática (GUEDES, 2014).

Operador	Função	Exemplos
+	Adição	$1 + 2$, $A + B$
-	Subtração	$6 - 3$, $A - B$
*	Multiplicação	$2 * 4$, $A * B$
/	Divisão	$6/3$, A/B

Tabela 1.1 – Operadores aritméticos

Fonte: Adaptada de Forbellone (2005, p. 19).

Alguns operadores aritméticos utilizados na realização de cálculos podem ser também representados por funções matemáticas, como a potenciação, a radiciação, o resto da divisão e o quociente da divisão (GUEDES, 2014).

Operador	Função	Significado	Exemplos
pot(x,y)	Potenciação	x elevado a y	Pot(2,4)
rad(x)	Radiciação	Raiz quadrada de x	rad(25)

Tabela 1.2 – Potenciação e radiciação.

Fonte: Adaptada de Forbellone (2005, p. 19).

Operador	Função	Exemplos
mod	Resto da divisão	9 mod 4 resulta em 1 27 mod 5 resulta em 2
div	Quociente da divisão	9 div 4 resulta em 2 27 div 5 resulta em 5

Tabela 1.3 – Operador de resto e quociente da divisão inteira

Fonte: Adaptada de Forbellone (2005, p. 19).

De acordo com Forbellone (2005), para a resolução das expressões aritméticas, as operações devem obedecer a uma hierarquia entre elas.

Prioridade	Operadores
1 ^a	parênteses mais internos
2 ^a	pot rad
3 ^a	* / div mod
4 ^a	+ -

Tabela 1.4 – Precedência entre os operadores aritméticos

Fonte: Adaptada de Forbellone (2005, p. 20).

No caso de operadores de mesma prioridade, deve-se resolver da esquerda para a direita, conforme a sequência existente na expressão aritmética (FORBELLONE, 2005).

a. $4 + 2 + 6/2$
 $4 + 2 + 3$

$$b. 4 - 2 * 6/3 - \text{pot}(2,3)$$

$$4 - 2 * 6/3 - 8$$

$$4 - 12/3 - 8$$

$$4 - 4 - 8$$

$$- 8$$

$$c. \text{pot}(4,2) - 6/3 + \text{rad}(1 + 4 * 6) / 5$$

$$\text{pot}(4,2) - 6/3 + \text{rad}(1 + 24) / 5$$

$$\text{pot}(4,2) - 6/3 + \text{rad}(25) / 5$$

$$16 - 6/3 + 1$$

$$16 - 2 + 1$$

$$15$$

As expressões lógicas são aquelas cujos operadores são lógicos ou relacionais e os operandos são relações, variáveis ou constantes do tipo lógico (FORBELLONE, 2005).

Os operadores relacionais são utilizados para realizar comparações entre dois valores do mesmo tipo primitivo, em que esses valores são representados por constantes, variáveis ou expressões aritméticas, e nas construções de equações, é comum a utilização de operadores relacionais, em que o resultado de uma relação é sempre um valor lógico (FORBELLONE, 2005).

Operador	Função	Exemplos
==	Igual a	2 == 2, A == B
>	Maior que	8 > 7, A > B
<	Menor que	2 < 3, A < B
>=	Maior ou igual a	4 >= 2, A >= B
<=	Menor ou igual a	2 <= 4, A <= B
<>	Diferente de	4 <> 5, A <> B

Tabela 1.5 – Operadores relacionais

Fonte: Adaptada de Forbellone (2005, p. 21).

Exemplos:

a. $3 * 4 = 24/2$

$12 = 12$

V

b. $15 \text{ mod } 4 > 9 \text{ mod } 4$

$3 > 1$

V

c. $2 * 5 \text{ div } 3 \leq \text{pot}(2,3) / 2$

$10 \text{ div } 3 \leq 8 / 2$

$3 \leq 4$

V

Tabela Verdade (Operadores Lógicos)

Os operadores lógicos são utilizados para associar expressões que estabelecem uma comparação entre valores (GUEDES, 2014) para a formação de novas proposições lógicas compostas a partir de outras proposições lógicas mais simples (FORBELLONE, 2005).

Operador	Função
não	negação
e	conjunção
ou	disjunção

Tabela 1.6 – Operadores lógicos

Fonte: Adaptada de Forbellone (2005, p. 22).

A tabela verdade é o conjunto de todas as possibilidades combinatórias entre os valores de diversas variáveis ou expressões lógicas em função do operador lógico utilizado que se encontram em duas possíveis situações, verdadeiro ou falso (FORBELLONE, 2005; GUEDES, 2014), sendo uma ferramenta muito utilizada para facilitar a análise da combinação de expressões e variáveis (GUEDES, 2014).

As tabelas a seguir apresentam exemplos de tabela verdade.

A	não A
F	V
V	F

Tabela 1.7 – Operação de negação

Fonte: Adaptada de Forbellone (2005, p. 23).

A	B	A e B
F	F	F
F	V	F
V	F	F
V	V	V

Tabela 1.8 – Operação de conjunção

Fonte: Adaptada de Forbellone (2005, p. 23).

A	B	A ou B
F	F	F
F	V	V
V	F	V
V	V	V

Tabela 1.9 – Operação de disjunção

Fonte: Adaptada de Forbellone (2005, p. 23).

De acordo com Forbellone (2005), a seguir constam alguns exemplos de aplicações com tabela verdade:

- a. Se chover **e** relampejar, eu fico em casa.
Quando eu fico em casa?

Nesse exemplo, a proposição somente será verificada quando os termos chover e relampejar forem simultaneamente verdadeiros.

b. Se chover **ou** relampejar eu fico em casa.

Quando eu fico em casa?

Com o operador lógico ou, as possibilidades de “eu fico em casa” são maiores, pois, de acordo com a tabela verdade, a proposição se tornará verdadeira nos casos: somente chovendo, somente relampejando e chovendo e relampejando.

c. $2 < 5$ e $15/3 = 5$

V e $5 = 5$

V = V

V

d. $2 < 5$ ou $15/3 = 5$

V ou V

V

Prioridade	Operadores
1 ^a	não
2 ^a	e
3 ^a	ou

Tabela 1.10 – Operação de conjunção

Fonte: Adaptada de Forbellone (2005, p. 24).

Prioridade	Operadores
1ª	parênteses mais internos
2ª	operadores aritméticos
3ª	operadores relacionais
4ª	operadores lógicos

Tabela 1.11 – Precedência entre todos os operadores

Fonte: Adaptada de Forbellone (2005, p. 24).

Exemplo:

a. não (4 <> 8/2) ou V e 3 – 6 > 6 – 3 ou V
 não (4 <> 4 ou V e – 3 > 3 ou V)
 não (F ou V e F ou V)
 não (F ou F ou V)
 não (F ou V)
 não (V)
 F

Estrutura Sequencial

De acordo com Forbellone (2005), uma estrutura sequencial de um algoritmo está relacionada com o fato de que o conjunto de ações primitivas será executado em uma sequência linear de cima para baixo e da esquerda para a direita, ou seja, da mesma maneira em que foi escrita.

As ações serão seguidas de um ponto e vírgula (;), que tem por finalidade separar uma ação de outra e auxiliar na organização sequencial das ações, pois, ao encontrar um ponto e vírgula, se deve executar o próximo comando na sequência (FORBELLONE, 2005).

Segundo Forbellone (2005), o modelo apresentado a seguir representa um modelo básico de estrutura sequencial para se escrever um algoritmo, em que identifica-se o bloco, coloca-se início e fim, e dentro são iniciados a declaração das variáveis, e, depois, o corpo do algoritmo.

Algoritmo Modelo básico

1. ***início*** // identificação do início do bloco correspondente ao algoritmo
- 2.
3. *// declaração de variáveis*
- 4.
5. *// corpo do algoritmo*
6. Ação 1;
7. Ação 2;
8. Ação 3;
9. .
10. .
11. .
12. Ação n;
13. ***Fim.*** // fim do algoritmo

Modelo básico em C

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Oi Fundamentos da Programação");
```

```
    return 0;
```

```
}
```


Segundo Ascencio e Campos (2012), as variáveis são declaradas após a palavra DECLARE e os tipos mais usuais são: NUMÉRICO (para variáveis que irão receber números), LITERAL (para as variáveis que irão receber caracteres) e LÓGICO (para as variáveis que irão receber valores verdadeiros ou falsos).

Exemplo:

DECLARE

X NUMÉRICO
Y, Z LITERAL
TESTE LÓGICO

O comando de atribuição é usado para fornecer valores ou operações para as variáveis (ASCENCIO; CAMPOS, 2012).

Exemplo:

X = 6
X = X + 2
Y = "nome"
teste = verdade

Exemplo do algoritmo para calcular a média aritmética:

Algoritmo Média Aritmética

1. **início** // começo do algoritmo
- 2.
3. Declare // declaração de variáveis
4. N1, N2, N3, N4 numérico // notas bimestrais
5. MA numérico // média anual
- 6.
7. // entrada de dados
8. Leia (N1, N2, N3, N4);
- 9.
10. // processamento

```
11.    MA ← (N1, N2, N3, N4) / 4;
12.
13.    // saída de dados
14. Escreva (MA);
15.
16.    Fim. // fim do algoritmo
```

Em linguagem C o mesmo algoritmo ficaria da forma a seguir:

```
#include<stdio.h>

#include<stdlib.h>

int main(void)
{

    float nota1, nota2,nota3, nota4, media;

    //Entrada de dados

    printf("Digite a primeira nota do aluno: ");

    scanf("%f",&nota1);

    printf("Digite a segunda nota do aluno: ");

    scanf("%f",&nota2);

    printf("Digite a terceira nota do aluno: ");

    scanf("%f",&nota3);
```

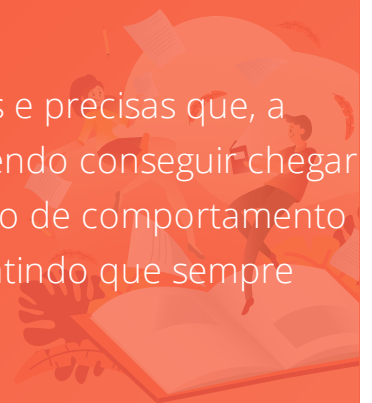
```
printf("Digite a quarta nota do aluno: ");  
  
scanf("%f",&nota4);  
  
media = (nota1 + nota2+ nota3 + nota4) / 4;  
  
printf("Media do aluno = %.1f\n",media);  
  
return 0;  
  
}
```

reflita

Reflita

“Quando é criado um algoritmo, deve-se especificar ações claras e precisas que, a partir de um estado inicial e após um determinado tempo, podendo conseguir chegar no resultado final esperado. Um algoritmo estabelece um padrão de comportamento a ser seguido, com o objetivo de solucionar um problema, garantindo que sempre que for feito do mesmo modo, o resultado final será o mesmo.”

Fonte: Forbellone (2005, p. 4).



atividade

Atividade

É um tipo de dado que representa uma posição de memória e tem nome e tipo e seu conteúdo pode variar ao longo do tempo de execução do programa e, embora também possa assumir valores diferentes, a memória somente pode armazenar um valor de cada vez.

Assinale somente a alternativa correta.

ASCENCIO, A. F. G.; CAMPOS, E. A. V. **Fundamentos da Programação de Computadores**: algoritmos, Pascal, C/C++ (padrão ANSI) e Java. 3. ed. São Paulo: Pearson Education do Brasil, 2012.

- ☐ **a)** Constante.
 - ☐ **b)** Variável.
 - ☐ **c)** Operador aritmético.
 - ☐ **d)** Operador relacional.
 - ☐ **e)** Expressão lógica.
-

Estrutura de Seleção

A estrutura de seleção permite a escolha de um grupo de ações ou bloco a ser executado quando determinadas condições, representadas por expressões lógicas ou relacionais, são ou não satisfeitas (GUEDES, 2014). Serão abordados dois tipos de estrutura de seleção: simples e composta.

Sintaxe da Estrutura Sequencial: Seleção Simples e Seleção Composta

A seleção simples é utilizada para testar certa condição antes de executar determinada ação. Se a condição for aceita, um determinado bloco de instruções deverá ser executado, e se a condição não for aceita, o fluxo da execução do algoritmo irá seguir após o fim do bloco de decisão (GUEDES, 2014), como mostra o modelo a seguir:

```
se <condição>  
    então  
        <conjunto de instruções>  
fimse
```

Em linguagem C ficaria da seguinte forma, substituindo a <condição> por $x==y$ e <conjunto de instruções> por $a = x+y$; :

```
if(x==y){  
  
    a = x+y;  
  
}
```

Exemplo:

Algoritmo Saber se o número informado é ímpar

1. declare
2. numero numérico
- 3.
4. **início**
- 5.
6. leia (numero)
- 7.
8. **se** ((numero mod 2) = 1 **então**
- 9.
10. escreva ("O número informado é ímpar")
- 11.
12. **fimse**
- 13.
14. **fim.**

Em linguagem C o programa ficaria da seguinte forma :

```
int main (void)  
  
{  
  
    int numero;
```

```
scanf("%d",&numero);

if (numero % 2 != 0)

printf ("%d\nO Número digitado é impar");

}
```

Como no exemplo anterior existe somente uma condição a ser executada, então essa estrutura de seleção é denominada **estrutura simples** (GUEDES, 2014).

Utiliza-se a estrutura de seleção composta quando houver casos em que duas alternativas dependam de uma mesma condição, uma de a condição ser verdade e outra de a condição ser falsa (FORBELLONE, 2005), conforme modelo a seguir:

```
se <condição>
    então
        <conjunto de instruções para o teste de condição resultar em
verdade>
    senão
        <conjunto de instruções para o teste de condição resultar em
falso>
fimse
```

Em linguagem C ficaria da seguinte forma, substituindo a <condição> por $x==y$ e <conjunto de instruções para o teste de condição resultar em verdade> por $a = x+y$; e <conjunto de instruções para o teste de condição resultar em falso> $a=x-y$:

```
if(x==y){

a = x+y;

}
```

```
else {
```

```
a= x-y;
```

```
}
```

Exemplo:

Algoritmo Cálculo do bônus salarial

1. declare
2. salario, bônus, tempo numérico
- 3.
4. **início**
- 5.
6. leia (salario)
7. leia (tempo)
- 8.
9. **se** (tempo >= 5) **então**
- 10.
11. bônus ← salario * 0,20
- 12.
13. **senão**
- 14.
15. bônus ← salario * 0,10
- 16.
17. **fimse**
- 18.
19. escreva ("O valor do bônus é", bônus)
- 20.
21. **fim.**

Em linguagem em C ficaria da seguinte forma:


```
#include<stdio.h>

#include<stdlib.h>

int main() {

    float salario;

    int tempo;

    float bonus;

    printf("Digite o valor do salario:");

    scanf("%f", &salario);

    printf("\nDigite o valor do tempo:");

    scanf("%d", &tempo);

    if (tempo >= 5) {

        bonus = (salario * 0,20);

    }

    else {

        bonus = (salario * 0,10);

    }

    printf("O valor do bônus é: %f \n", bonus);

}
```

Nota-se que existe uma condição para a resposta verdadeira e outra condição para a resposta falsa, sendo assim, caracterizado como uma

estrutura de **seleção composta** (GUEDES, 2014).

atividade

Atividade

Uma estrutura de seleção permite a escolha de um grupo de ações a ser executado quando determinadas condições são ou não satisfeitas, sendo que existem os alguns tipos de seleção. Observe o código:

```
if(x<=y){
```

```
  a = (x*y)+1;
```

```
}
```

Observando o código trata-se de uma seleção? Assinale somente a alternativa correta.

- ☐ **a)** Simples.
 - ☐ **b)** Composta.
 - ☐ **c)** Múltipla escolha.
 - ☐ **d)** Soma.
 - ☐ **e)** Igualdade.
-

Introdução à Linguagem de Programação

Segundo Guedes (2014), a linguagem de programação é constituída de um conjunto de regras e palavras agrupadas em frases que irão resultar em um determinado significado, sendo assim, essas palavras podem ser chamadas de comandos e as frases oriundas de estruturas de programação.

Um programa de computador é formado de comandos e estruturas de programação definidas pelas regras da linguagem de programação e produzidas de maneira que sejam fáceis de serem compreendidas pelos seres humanos (GUEDES, 2014).

As linguagens de programação foram criadas para solucionarem determinados tipos de problemas, sendo que algumas podem ser melhores para determinadas aplicações do que outras. Portanto, a linguagem de programação depende muito da sua adequação para a tarefa que se pretende realizar (GUEDES, 2014).

História da Linguagem de Programação

De acordo com Guedes (2014), existem diferentes linguagens de programação, que surgiram ao longo do desenvolvimento dos sistemas computacionais, e cada uma delas tem características próprias e recursos existentes da época de sua criação. Algumas permanecem ativas e foram atualizadas, recebendo novas funcionalidades e se adaptando para a exigente realidade de avanços no desempenho de hardware e exigências dos sistemas.

As primeiras linguagens de programação que surgiram eram de estrutura sequencial, não tinham interatividade e se destinavam a uma única atividade específica, devido às características dos sistemas existentes à época. Depois, com a evolução dos computadores, dos sistemas operacionais e da complexidade computacional, surgiram as linguagens estruturadas, que permitiram a criação de sistemas mais interativos, organizados e com mais funcionalidades (GUEDES, 2014).

Posteriormente, surgiram as linguagens orientadas a objetos, que promoveram uma grande transformação no modo como os sistemas são desenvolvidos e codificados, retornando grande interatividade, processamento distribuído e alta diversidade para dispositivos (GUEDES, 2014).

Tipos de Dados, Constantes e Variáveis

Os tipos de dados mais utilizados são numéricos, lógicos e literais. Os dados numéricos se dividem em dois grupos: inteiros e reais. Os dados numéricos do tipo inteiros podem ser positivos ou negativos e não podem ser fracionários (GUEDES, 2014). Exemplos: -10, 32, 129, -418. Já os dados numéricos reais podem ser positivos ou negativos e apresentam parte fracionária, e seguem a notação da língua inglesa, em que a parte decimal é separada da parte inteira por um ponto, e não por uma vírgula (GUEDES, 2014). Exemplo: 1.45, - 3.50, 225.75, 319.12.

Também conhecidos como dados booleanos, os dados lógicos assumem valores verdadeiros e falsos (GUEDES, 2014). Já os dados literais são tipos de

dados formados por um único caractere ou por uma sequência de caracteres. Esses dados podem ser caracteres do alfabeto, números, letras maiúsculas e minúsculas e caracteres especiais (GUEDES, 2014). Exemplo: "nome", "2A", "23#", '5'.

Os caracteres representados entre aspas duplas (" ") caracterizam um conjunto de caracteres e um único caractere é representado por aspas simples (' ') (GUEDES, 2014).

De acordo com Guedes (2014), os tipos de constantes, que são valores que não sofrem alterações ao longo do desenvolvimento do algoritmo ou da execução do programa, são expressos como dados constantes, como a raiz quadrada de um número, constante de Napier (e) e o π que é o Pi, no caso o seu valor nunca altera, por exemplo:: $\text{Pi} = 3.14159265359$, isso é uma constante, ou, até mesmo, um valor informado que não se altera durante a execução do algoritmo.

$\text{Pi} = 3.14159265359$, isso é uma constante."

Como exemplos de tipos de variáveis, podemos citar a cotação de moedas, como dólar e euro, o índice de inflação (GUEDES, 2014), as ações de uma empresa na bolsa de valores, o valor do consumo de quilowatt-hora de uma residência etc.

Comandos de Entrada e Saída: Estrutura Sequencial

A estrutura sequencial em uma linguagem de programação pode ser descrita como no exemplo a seguir, em que foi utilizada a estrutura sequencial da linguagem de programação Pascal (ASCENCIO; CAMPOS, 2012).

```
PROGRAM nome;  
USES nomes_das_unidades;  
VAR nomes_das_variáveis: tipo;  
BEGIN
```

bloco_de_comandos
END.

Uma observação USES nomes_das_unidades: Unidades são as bibliotecas utilizadas pela linguagem pascal para a correta execução do programa.

A declaração de variáveis em Pascal é dada da seguinte maneira:

```
VAR X: INTEGER;  
    Y, Z: REAL;  
    NOME: STRING  
    SEXO: CHAR;  
    TESTE: BOOLEAN;
```

E as declarações de constantes realizadas após a palavra CONST.

```
CONST X = 10;  
Y = 6.5;  
NOME = "ZECA";  
SEXO = 'M';  
TESTE = TRUE;
```

A atribuição de valores é feita da seguinte forma:

```
X := 5;  
Y := 3.5;  
Nome := "Arthur";  
Sexo := 'M';  
Teste := false;
```

Um comando de entrada é descrito da seguinte forma:

```
READLN(nome_da_variável);
```

Exemplo:

```
READLN(NOME);
```

Um comando de saída é descrito da seguinte forma:

```
WRITELN(nome_da_variável);
```

Exemplo:

```
WRITELN(NOME);
```



atividade

Atividade

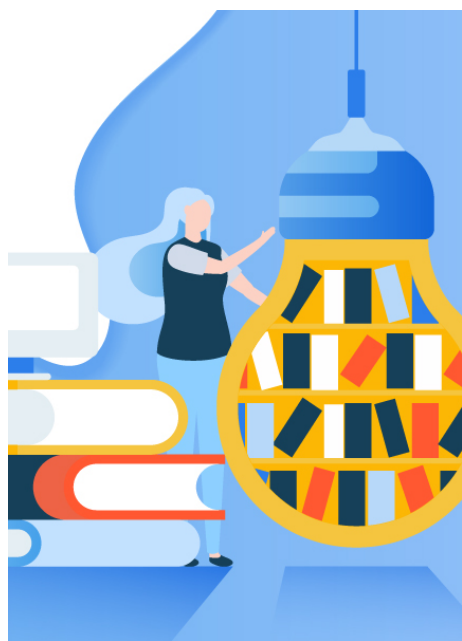
Os tipos de dados mais utilizados em uma estrutura de algoritmo são os numéricos, os lógicos e os literais. Em relação aos tipos de dados lógicos, assinale a alternativa correta.

GUEDES, S. **Lógica de Programação Algorítmica**. São Paulo: Pearson Education do Brasil, 2014.

- ☐ **a)** São do tipo inteiros e reais.
 - ☐ **b)** São formados por uma sequência de caracteres.
 - ☐ **c)** São conhecidos como dados booleanos e assumem valores verdadeiros e falsos.
 - ☐ **d)** Podem ser positivos ou negativos.
 - ☐ **e)** Assumem valores fracionários.
-

indicações

Material Complementar



LIVRO

Algoritmos e Programação: Teoria e Prática

Editora: Novatec

Autor: Marco Medina e Cristina Fertig

ISBN: 978-8575220733

Comentário: Essa é uma obra bastante didática, que demonstra como compreender algoritmos e pseudocódigo com uma linguagem muito simples. O livro detalha todos os processos da construção de algoritmos e oferece diversos exemplos em pseudocódigo e nas linguagens de programação Pascal e C.



FILME

O Jogo da Imitação

Ano: 2014

Comentário: O filme é baseado na história real de Alan Turing, matemático britânico considerado o pai da computação. Em plena Segunda Guerra Mundial, Turing é contratado pelo governo inglês para ajudar a construir uma máquina capaz de criptografar mensagens nazistas e assim contribuir para que sejam definidas novas estratégias de guerra pelos aliados com o intuito de vencer a guerra. Essa máquina ficou conhecida como máquina de Turing, e para sua construção, foram utilizados conceitos de lógica e de algoritmos.

TRAILER

conclusão

Conclusão

Nesta unidade estudamos que a lógica está diretamente relacionada com a “correção do pensamento”, e que a utilização dos processos lógicos de programação é fundamental na construção de algoritmos. Vimos, também, que um algoritmo é uma sequência de etapas bem detalhadas e que tem por finalidade encontrar a solução de um determinado problema. Também foram detalhados os conceitos de tipos de dados, variáveis, constantes e operadores, e as expressões aritméticas e lógicas que compõem a estrutura de um algoritmo. Foram abordados os conceitos de estrutura sequencial e de seleção, em que percebemos que, na estrutura sequencial, o algoritmo é executado passo a passo, do início da ação até seu fim, e que a estrutura de seleção permite que uma ação seja ou não executada, dependendo do valor da sua condição.

referências

Referências Bibliográficas

ASCENCIO, A. F. G.; CAMPOS, E. A. V. **Fundamentos da Programação de Computadores**: algoritmos, Pascal, C/C++ (padrão ANSI) e Java. 3. ed. São Paulo: Person Education do Brasil, 2012.

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

GUEDES, S. **Lógica de Programação Algorítmica**. São Paulo: Pearson Education do Brasil, 2014.

IMPRIMIR