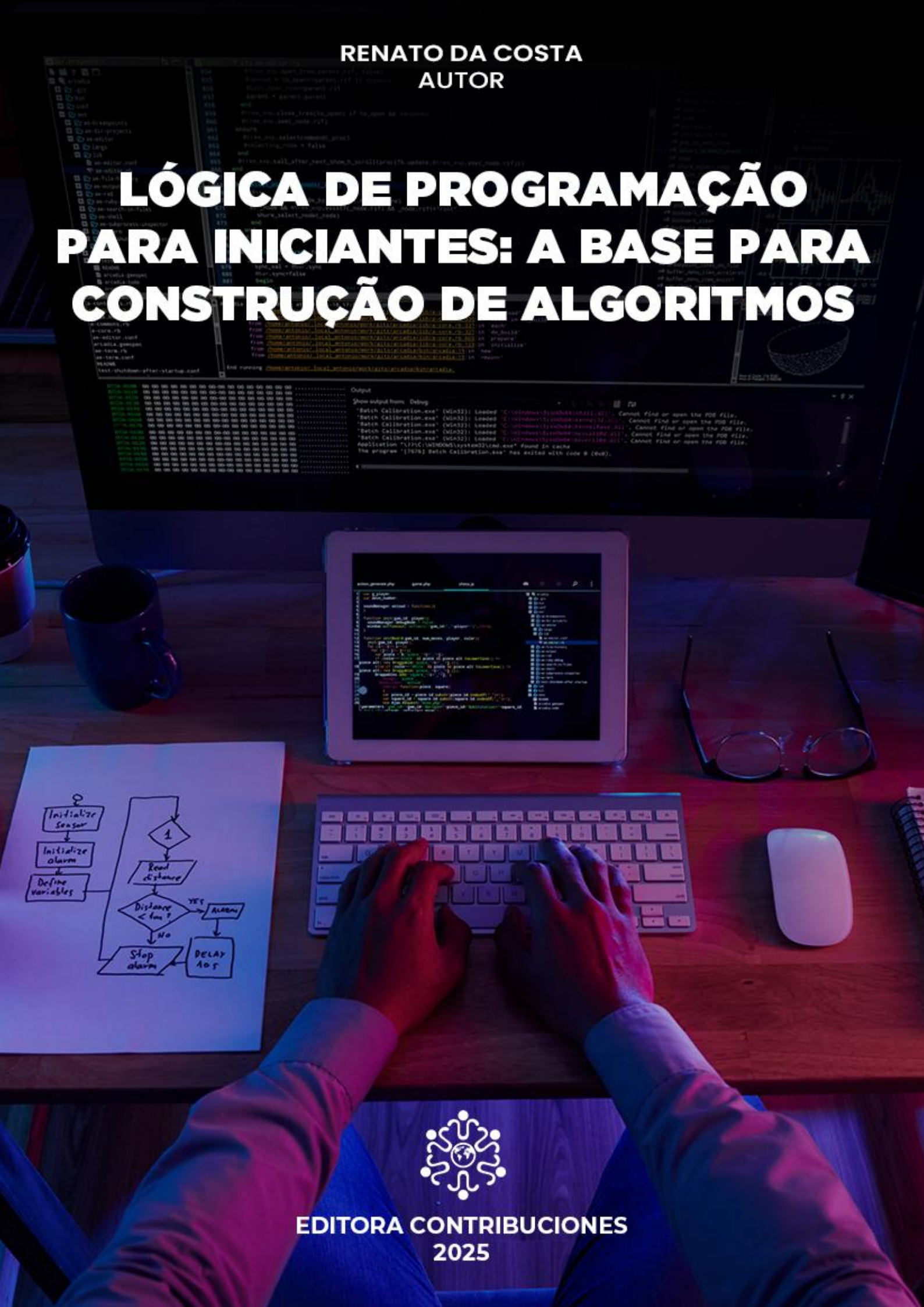
RENATO DA COSTA
AUTOR

LÓGICA DE PROGRAMAÇÃO PARA INICIANTES: A BASE PARA CONSTRUÇÃO DE ALGORITMOS



EDITORA CONTRIBUCIONES
2025

Renato da Costa

Lógica de programação para iniciantes: a base para construção de algoritmos

Editora Contemporânea
2025

Editor Chefe
João Paulo Perbiche
Conselho Editorial
Adilson Ferraz
Isabel Martins
George Fernandes da Silva
Lucas T. Galindo Filho
José Alan
José João Neves Barbosa Vicente

Site
www.revistacontemporanea.com
E-mail
ebooks@revistacontemporanea.com

Dados Internacionais de Catalogação na Publicação (CIP)

Costa, Renato dos Santos da Lógica de programação para iniciantes [livro eletrônico]: a base para construção de algoritmos / Renato dos Santos da Costa. -- Curitiba, PR: Editora Contemporânea, 2025.

PDF.
Bibliografia.
ISBN: 978-65-83227-14-0
DOI: 10.56083/edcont.978-65-83227-14-0

1. Algoritmos de computadores 2. Ciência da computação 3. Linguagem de programação para computadores 4. Programação (Computadores) Estudo e ensino.
I. Costa, Renato dos Santos da. II. Título

25-261110 CDD-005.1

EDITORA CONTEMPORÂNEA

ANO 2025

APRESENTAÇÃO

Meu primeiro contato com a computação ocorreu no final da década de 1980, quando adquiri um MSX, um computador pessoal que, conectado a uma televisão e a um gravador de fita cassete, permitiu que eu escrevesse minhas primeiras linhas de código. Naquele momento, sem imaginar aonde esse interesse me levaria, começava minha jornada no mundo da programação.

No início da década de 1990, a escassez de professores na área de computação me proporcionou a oportunidade, ainda na juventude, de dar minhas primeiras aulas. Curiosamente, a disciplina que me abriu as portas foi justamente lógica de programação, e foi nesse contexto que escrevi as primeiras páginas deste livro. A partir daí, segui um longo caminho, ensinando em diversas instituições e contribuindo para a formação de centenas de alunos ao longo de quase três décadas dedicadas à educação.

Filho de pais que não tiveram a oportunidade de cursar o ensino superior, alcançar minha primeira graduação em tecnologia em processamento de dados foi uma grande conquista. Desde então, nunca deixei de buscar conhecimento e, com dedicação, concluí graduações, especializações, mestrados e, agora, tenho a felicidade de estar no doutorado.

Este livro, que esteve por muito tempo como um projeto adormecido, finalmente se tornou realidade. Ele reflete não apenas a minha experiência como professor, mas também o desejo de tornar a lógica de programação acessível a todos que desejam aprender e se aprimorar nessa área fundamental da computação.

Nenhuma trajetória é construída sozinho, e este livro é o resultado de uma caminhada repleta de apoio, aprendizado e companheirismo. Quero expressar minha sincera gratidão a todos os alunos, professores e colegas que, ao longo dos anos, estiveram ao meu lado, trocando conhecimentos, desafios e experiências. Cada sala de aula, cada dúvida respondida e cada ideia compartilhada contribuíram para que este projeto se concretizasse.

Agradeço especialmente à minha família, meu maior alicerce, que sempre esteve presente em cada conquista e desafio. À minha esposa Caroline, por seu apoio incondicional e compreensão ao longo dessa jornada,

e à minha amada filha Isabella, que me inspira diariamente a ser um exemplo de dedicação e perseverança.

Que este livro possa servir como guia e inspiração para aqueles que desejam ingressar no fascinante mundo da programação. A todos que, de alguma forma, fizeram parte dessa história, o meu muito obrigado.

Autor

SOBRE O AUTOR

O professor Renato da Costa é Doutorando em Administração pela Unigranrio, é Mestre em Novas Tecnologias Digitais na Educação pela UniCarioca e Mestre em Educação pela Universidade de Jaén (Espanha), MBA em Gerenciamento de Projetos pela FGV e MBA em Data Science pela Infnet/RJ, pós-graduado em Tecnologia da Informação e Comunicação e pós-graduado em Docência do Ensino Superior sendo ambas as especializações pela Ucam, possui Bacharelado em Administração, Licenciatura Plena em Informática, Matemática e Pedagogia além da graduação em Tecnologia em Processamento de Dados.

Ministra aulas para concursos públicos desde 1996, tendo sido aprovado em diversos concursos nas esferas estaduais e federais. Atualmente é servidor federal, professor de computação do IFRJ, tendo sido aprovado em 1º lugar em concurso de provas e títulos para o quadro efetivo.

É autor do livro "Informática pra Concursos", atualmente na 4ª edição, pela editora Impetus, citado como referência bibliográfica por diversas bancas. Possui certificações Cisco (CCST) e Microsoft (AZ-900).

Lattes: http://lattes.cnpq.br/6315288152256525
E-mail para contato: renatodacosta@gmail.com

SUMÁRIO

CAPÍTULO 01 1
INTRODUÇÃO À LÓGICA DE PROGRAMAÇÃO
1.1 ALGORITMO 1
1.2 PROGRAMA 1
DOI: 10.56083/edcont.978-65-83227-14-0_1

CAPÍTULO 02 3
INTRODUÇÃO AO PORTUGOL E FUNDAMENTOS DE PROGRAMAÇÃO
2.1 ALGORITMOS EM “PORTUGOL” 3
2.2 TÉCNICAS DE PROGRAMAÇÃO 3
DOI: 10.56083/edcont.978-65-83227-14-0_2

CAPÍTULO 03 4
MATEMÁTICA NA INFORMÁTICA 4
3.1 OPERADORES MATEMÁTICOS 4
3.1.1 Operadores aritméticos 4
3.1.2 Operadores de comparação ou relacionais (relação de ordem) 4
3.2 REESCRITA DE EXPRESSÕES EM NOTAÇÃO COMPUTACIONAL 5
3.3 AGRUPAMENTO DE EXPRESSÕES MATEMÁTICAS 5
3.4 OPERADORES ARITMÉTICOS DE DIVISÃO INTEIRA (MOD e DIV) 6
3.5 FUNÇÕES 6
3.6 PRINCIPAIS FUNÇÕES NATIVAS 7
3.7 LISTA GERAL DE PRIORIDADES DAS OPERAÇÕES 8
DOI: 10.56083/edcont.978-65-83227-14-0_3

CAPÍTULO 04 9
ESTRUTURAS LÓGICAS E OPERADORES
4.1 EXPRESSÕES LÓGICAS 9
4.2 OPERADORES LÓGICOS 9
4.3 TABELA VERDADE 10
DOI: 10.56083/edcont.978-65-83227-14-0_4

CAPÍTULO 05 11
ESTRUTURAS LÓGICAS E OPERADORES
5.1 VARIÁVEIS 11
5.1.2 Variáveis de entrada e saída 11
5.1.3 Identificadores 12
5.1.4 Sinal de atribuição (recebe ou implica) 12
5.2 CONSTANTES 13
5.3 SINAL DE IGUALDADE 13
5.4 TIPOS DE DADOS 13
5.4.1 Tipos primitivos de dados 14
DOI: 10.56083/edcont.978-65-83227-14-0_5

CAPÍTULO 06 15
COMANDOS BÁSICOS E ESTRUTURAS DE ALGORITMOS
6.1 COMANDOS BÁSICOS DE ENTRADA E SAÍDA(INPUT/OUTPUT) 15
6.2 FLUXO DE UM ALGORITMO 15
6.3 CORPO GERAL DE UM ALGORITMO 16
6.4 ESTRUTURAS SEQUÊNCIAIS 17
6.4.1 Construindo o primeiro algoritmo 17
6.4.2 Construindo outros algoritmos de exemplo 19
6.5 {LINHAS DE COMENTÁRIO} 21
6.6 ASPAS ‘SIMPLES’ 21
DOI: 10.56083/edcont.978-65-83227-14-0_6

CAPÍTULO 07 22
ESTRUTURAS CONDICIONAIS
7.1 ESTRUTURA “SE” 22
7.2 NINHOS DE SE 24
7.3 ESTRUTURA “CONFORME” 27
DOI: 10.56083/edcont.978-65-83227-14-0_7

CAPÍTULO 08 29
ESTRUTURAS CONDICIONAIS
8.1 ESTRUTURA DE REPETIÇÃO DETERMINADA - “PARA” 29
8.2 ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO INICIAL – “ENQUANTO” 31
8.3 ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO FINAL – “REPITA ATÉ” 33
8.4 LOOP E CUIDADOS 35
DOI: 10.56083/edcont.978-65-83227-14-0_8

CAPÍTULO 09 36
INTRODUÇÃO À ESTRUTURAS DE DADOS
9.1 TIPOS PRIMITIVOS DE DADOS 36
9.2 TIPOS DE DADOS ESTRUTURADOS/COMPOSTOS 36
9.3 TIPOS DE DADOS DINÂMICOS 36
9.4 VETOR (ARRAY) 37
DOI: 10.56083/edcont.978-65-83227-14-0_9

CAPÍTULO 10 40
INTRODUÇÃO À ESTRUTURAS DE DADOS
DOI: 10.56083/edcont.978-65-83227-14-0_10

REFERÊNCIAS 41

ANEXO 1 42
QUESTÕES DE CONCURSOS DA BANCA CESGRANRIO 42
GABARITO 58

CAPÍTULO 01
INTRODUÇÃO À LÓGICA DE PROGRAMAÇÃO

1.1 ALGORITMO

Um Algoritmo é uma sequência de instruções ordenadas de forma lógica para a resolução de uma determinada tarefa ou problema.

> “Um algoritmo é a descrição de um padrão de comportamento, expressado em termos de um repertório bem definido e finito de ações ‘primitivas’, das quais damos por certo que elas podem ser executadas.” (GUIMARÃES; LAGES,1994, p. 4)

Para Shane (2022, p. 31) o uso de um algoritmo para uma tarefa específica poupa o problema e a despesa de um ser humano, especialmente quando a tarefa é de alto volume e repetitiva.

> “Afinal, o desempenho individual de um ser humano pode variar ao longo do dia, dependendo do quão recentemente eles comeram ou quanto dormiram, e os preconceitos ou o humor de cada pessoa pode ter um efeito enorme... Os algoritmos evitam inconsistências humanas – a partir de um conjunto de dados, eles retornarão praticamente um resultado invariável, não importa se é de manhã, meio-dia ou happy hour.” (SHANE, 2022, p. 35-36).

1.2 PROGRAMA

Um programa nada mais é do que a aplicação prática de um algoritmo utilizando uma linguagem de programação. Essas linguagens, especialmente as de alto nível, permitem que o desenvolvimento de software seja feito de maneira mais intuitiva, utilizando uma sintaxe semelhante à forma como pensamos e nos comunicamos. Para que um computador consiga interpretar e executar esses programas, o código escrito é traduzido para linguagem de máquina, tornando-se compreensível para o hardware.

A programação oferece um vasto campo de possibilidades, possibilitando a criação de diferentes tipos de software, desde jogos e editores de texto até

sistemas operacionais e ferramentas complexas. Cada linguagem possui suas particularidades, incluindo regras de sintaxe, comandos específicos e funcionalidades próprias, sendo escolhida conforme a necessidade do projeto e do programador. Exemplos de linguagens de programação: Pascal, C, Java, Python dentre outras.

2.1 ALGORITMOS EM “PORTUGOL”

Ao longo deste estudo, utilizaremos uma pseudolinguagem chamada Portugol, também conhecida como “Português Estruturado”, para desenvolver nossos algoritmos. O nome Portugol surge da junção de "Português" com "Algol", uma linguagem de programação criada no final da década de 1950.

É importante destacar que a sintaxe dos comandos em Portugol não é rigidamente padronizada, podendo apresentar pequenas variações conforme a convenção adotada. No entanto, o aspecto essencial na programação não é a sintaxe em si, mas sim a lógica por trás dos algoritmos, ou seja, a maneira como o raciocínio é estruturado para resolver problemas computacionais.

2.2 TÉCNICAS DE PROGRAMAÇÃO

Programação sequencial: o programa é descrito através de várias linhas, executadas uma após a outra.

Programação estruturada: dispõe da possibilidade de dividir o programa em subprogramas (procedimentos ou funções) diminuindo o tempo de programação para tarefas repetitivas, facilitando a manutenção do mesmo e minimizando os erros.

Programação orientada a eventos: o programa na verdade é dividido em várias partes agregadas a objetos utilizados pelo ambiente da linguagem e esses trechos do programa são despertados por ações determinadas como um clique em um botão ou uma fatia de tempo.

3.1 OPERADORES MATEMÁTICOS

Como a maioria dos programas possui algum tipo de processo matemático, iremos começar a estudar seus operadores, que podem ser aritméticos, de comparação ou lógicos.

3.1.1 OPERADORES ARITMÉTICOS

+ ➔ Adição
- ➔ Subtração
* ➔ Multiplicação
/ ➔ Divisão
^ ou ** ➔ Exponenciação ex. 2^3 = 2 ^ 3 ou 2 ** 3

Qual o resultado da expressão abaixo?
=2+2*2^2

Lembre-se que a prioridade dentre os operadores descritos anteriormente é a mesma da matemática, primeira a exponenciação seguido da multiplicação e divisão, e por último a soma e subtração. Logo o resultado encontrado na expressão é 10.

3.1.2 OPERADORES DE COMPARAÇÃO OU RELACIONAIS (RELAÇÃO DE ORDEM)

> ➔ Maior que
< ➔ Menor que
>= ➔ Maior ou Igual

<= ➔ Menor ou Igual

= ➔ Igual

<> ➔ Diferente

3.2 REESCRITA DE EXPRESSÕES EM NOTAÇÃO COMPUTACIONAL

Para a construção de algoritmos, todas as expressões aritméticas devem ser reescritas em notação linear ou notação computacional, conforme ilustrado na Figura 1.

É importante também ressalvar o uso dos operadores correspondentes da aritmética tradicional para a computacional.

Exemplo:

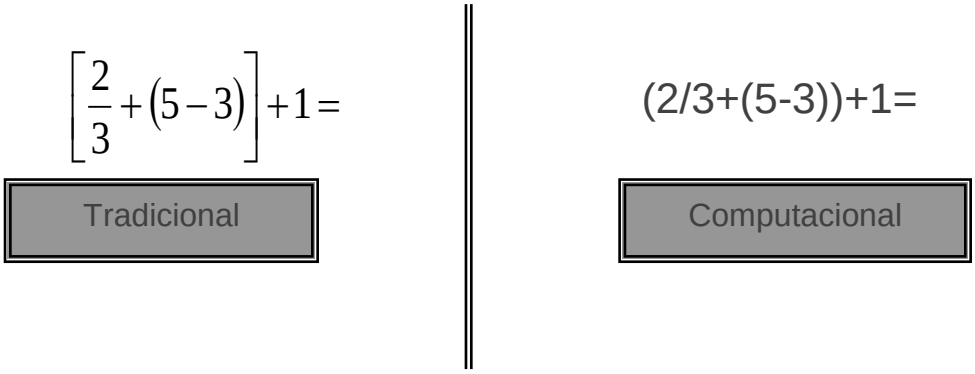


Figura 1 – Fonte do Autor

3.3 AGRUPAMENTO DE EXPRESSÕES MATEMÁTICAS

O agrupamento de expressões matemáticas consiste na divisão de uma expressão em partes menores, facilitando sua compreensão e determinando a ordem correta das operações. Esse processo permite que cálculos complexos sejam organizados de maneira clara, garantindo a correta interpretação da prioridade das operações.

Nas expressões computacionais, utilizamos exclusivamente parênteses "( )" para definir essa hierarquia, ao contrário da matemática tradicional, que pode empregar também colchetes "[ ]" e chaves "{ }". Além disso, na lógica de programação, é possível aninhar parênteses, ou seja, inserir expressões dentro de outras para indicar diferentes níveis de prioridade na avaliação dos cálculos.

Exemplos de agrupamentos:

(2+2)/2
=2

Primeiro resolve-se o que está em parênteses.

2+2/2
=3

3.4 OPERADORES ARITMÉTICOS DE DIVISÃO INTEIRA (MOD E DIV)

MOD ➔ Retorna o resto da divisão entre 2 números **inteiros.**

DIV ➔ Retorna o valor inteiro que resulta da divisão entre 2 números **inteiros.**

Exemplo:

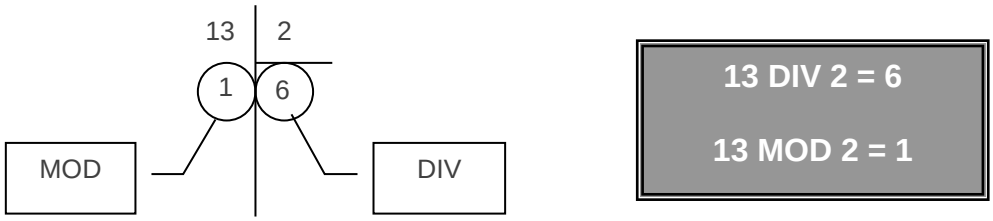


Figura 2 – Fonte do Autor

Observação: Como foi bem destacado acima não poderemos ter uma expressão tipo: 2,8 mod 2, pois 2,8 não é um número inteiro.

Observe:

8 * 3 + 7 mod 2 + 6 * 9
Calculando: 24 + 1 + 54 = 79

Observação: A prioridade dos operadores especiais é igual à da multiplicação ou divisão.

3.5 FUNÇÕES

Uma função é um instrumento (sub algoritmo) que tem como objetivo retornar um valor ou uma informação.

A chamada de uma função é feita através da citação do seu nome seguido opcionalmente de seu argumento inicial entre parênteses.

As funções podem ser predefinidas pela linguagem ou criadas pelo programador de acordo com o seu interesse.

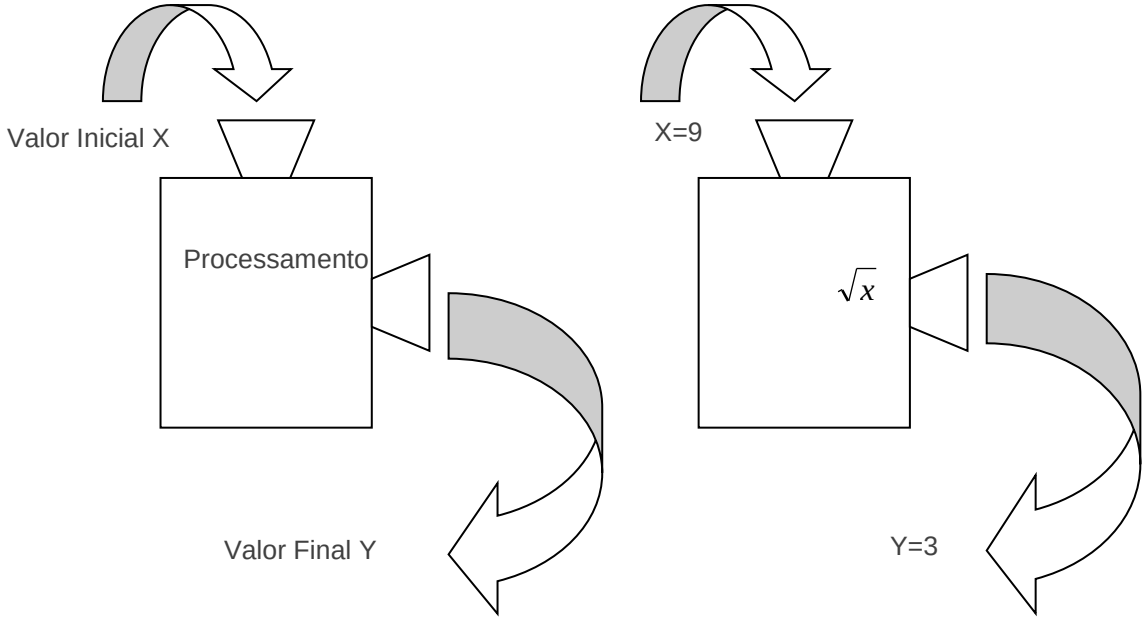


Figura 3 – Fonte do Autor

3.6 PRINCIPAIS FUNÇÕES NATIVAS

ABS( ) - VALOR ABSOLUTO |x|

SQRT( ) - RAIZ QUADRADA $\sqrt{x}$ (Square Root)

SQR( ) - ELEVA AO QUADRADO x^2

As funções acima são as mais comuns e mais importantes para nosso desenvolvimento da lógica de programação, entretanto, cada linguagem possui suas próprias funções com as devidas traduções. As funções podem ser de variadas categorias, como: aritméticas, temporais, lógicas, de texto, de aprendizado de máquina etc.

3.7 LISTA GERAL DE PRIORIDADES DAS OPERAÇÕES

1. PARÊNTESES E FUNÇÕES
2. EXPONENCIAL
3. MOD, DIV, MULTIPLICAÇÃO E DIVISÃO
4. ADIÇÃO E SUBTRAÇÃO
5. OPERADORES RELACIONAIS
6. NÃO
7. E
8. OU

4.1 EXPRESSÕES LÓGICAS

As expressões compostas de relações baseadas em uma proposição sempre resultam em um valor lógico do tipo Verdadeiro ou Falso.

Exemplos:

2+5>4	3<>3
= Verdadeiro	= Falso

4.2 OPERADORES LÓGICOS

Atuam sobre expressões lógicas retornando resultados do tipo Falso ou Verdadeiro, observe o Quadro 1.

E	RETORNA VERDADEIRO, SE TODOS OS PARÂMETROS DA EXPRESSÃO FOREM VERDADEIROS. BASTA QUE UM PARÂMETRO SEJA FALSO PARA RETORNAR FALSO.
OU	BASTA QUE UM PARÂMETRO SEJA VERDADEIRO PARA RETORNAR VERDADEIRO. RETORNA FALSO, SE TODOS OS PARÂMETROS DA EXPRESSÃO FOREM FALSOS.
NÃO	INVERTE O ESTADO DE UM PARÂMETRO FORNECIDO, DE VERDADEIRO PASSA PARA FALSO E VICE-VERSA.

Quadro 1 – Fonte do Autor

Prioridades dos operadores Lógicos:

NÃO - Negação

E - Conjunção

OU - Disjunção

4.3 TABELA VERDADE

Supondo A e B como expressões lógicas vamos verificar os estados de cada linha da tabela verdade apresentada no Quadro 2 a seguir:

A	B	A E B	A OU B	NÃO (A)
V	V	V	V	F
V	F	F	V	F
F	V	F	V	V
F	F	F	F	V

Quadro 2 – Fonte do Autor

5.1 VARIÁVEIS

Variáveis são endereços de memória destinados a armazenar informações temporariamente (durante a execução do algoritmo). Embora uma variável possa assumir diferentes valores, ela só pode armazenar um único valor a cada instante.

Todo Algoritmo ou programa deve possuir uma ou mais variáveis, pois deverá obter algum tipo de entrada do ambiente em que atua e produzir algum tipo de saída correspondente.

Por exemplo, imagine que é preciso calcular o dobro da sua idade, a maneira de organizar este propósito em um pseudocódigo seria:

resposta=idade*2.

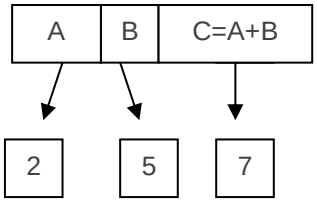
Neste pequeno exemplo temos 2 variáveis: a idade, de entrada, que vai variar para cada leitor e a resposta, de saída, que vai variar de acordo com a idade.

5.1.2 VARIÁVEIS DE ENTRADA E SAÍDA

Variáveis de Entrada armazenam informações obtidas por um meio externo, através de algum dispositivo de entrada, normalmente pelos próprios usuários, sensores ou unidades de disco.

Variáveis de Saída armazenam dados processados, sejam dados intermediários do processamento ou finais.

Exemplo:



De acordo com a figura acima A e B são Variáveis de Entrada e C é uma Variável de Saída.

5.1.3 IDENTIFICADORES

São os nomes significativos dados a variáveis, constantes e programas.

Regras para construção de Identificadores:

- Não podem ter nomes de palavras reservadas (comandos da linguagem);
- Devem possuir como 1º caractere uma letra ou Underscore (_);
- Ter como demais caracteres letras, números ou Underscore;
- Ter no máximo 127 caracteres;
- Não possuir espaços em branco;
- A escolha de letras maiúsculas ou minúsculas é indiferente.

Exemplos de identificadores válidos no Quadro 3:

NOME	TELEFONE	IDADE_FILHO
NOTA1	SALÁRIO	PI
UMNOMEMUITOCOMPRIDOEDIFICILDELER UM_NOME_MUITO_COMPRIDO_E_FACIL_DE_LER		

Quadro 3 – Fonte do Autor

5.1.4 SINAL DE ATRIBUIÇÃO (RECEBE OU IMPLICA)

Uma Variável nunca é eternamente igual a um valor, seu conteúdo pode ser alterado a qualquer momento. Portanto para atribuir valores a variáveis devemos usar o sinal de **“:=” ou** “←”.

Exemplos:

A ← 2;
B ← 3; (lê-se da seguinte forma: B recebe 3)
C :=A + B;

5.2 CONSTANTES

Assim como as variáveis, as constantes são endereços de memória destinados a armazenar informações, entretanto elas são fixas, inalteráveis durante a execução do programa.

Exemplo:

PI = 3.1416

5.3 SINAL DE IGUALDADE

As constantes são eternamente iguais a determinados valores, portanto, quando construímos um algoritmo, usamos o sinal de “=” ou “==” para identificá-las.

Exemplos:

PI **=** 3.1416;
Empresa = ‘Colégio de Informática L.T.D.A.’
V = Verdadeiro

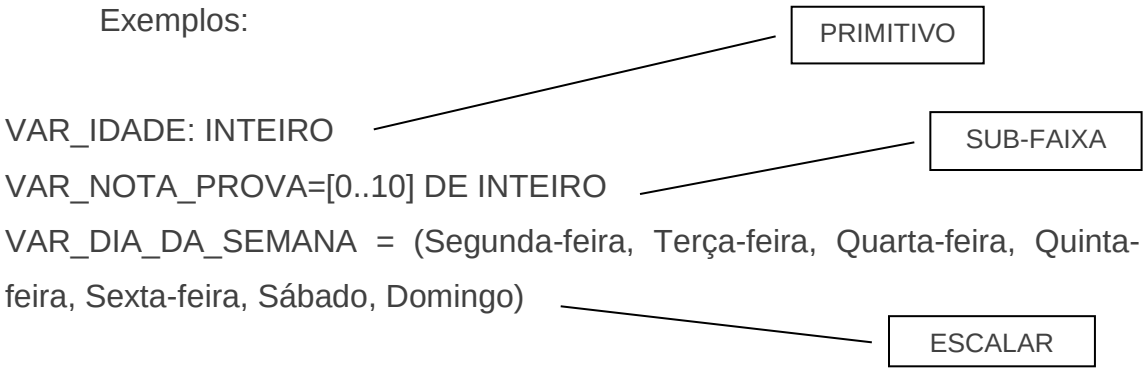
5.4 TIPOS DE DADOS

Todas as Variáveis devem assumir um determinado tipo de informação.

O tipo de dado pode ser:

- Primitivo ➔ Pré-definido pela linguagem;
- Sub-Faixa ➔ É uma parte de um tipo já existente;
- Escalar ➔ Definidos pelo programador.

Exemplos:



VAR_IDADE: INTEIRO

VAR_NOTA_PROVA=[0..10] DE INTEIRO

VAR_DIA_DA_SEMANA = (Segunda-feira, Terça-feira, Quarta-feira, Quinta-feira, Sexta-feira, Sábado, Domingo)

5.4.1 TIPOS PRIMITIVOS DE DADOS

Os tipos primitivos de dados compreendem: Inteiro, Real, Caractere e Lógico, conforme o Quadro 4 a seguir.

INTEIRO	ADMITE SOMENTE NÚMEROS INTEIROS. GERALMENTE É UTILIZADO PARA REPRESENTAR UMA CONTAGEM (QUANTIDADE).
REAL	ADMITE NÚMEROS REAIS (COM OU SEM CASAS DECIMAIS). GERALMENTE É UTILIZADO PARA REPRESENTAR UMA MEDIÇÃO.
CARACTERE	ADMITE CARACTERES ALFANUMÉRICOS. OS NÚMEROS QUANDO DECLARADOS COMO CARACTERES TORNAM SE REPRESENTATIVOS E PERDEM A ATRIBUIÇÃO DE VALOR.
LÓGICO	ADMITE SOMENTE VALORES LÓGICOS DITOS COMO BOOLEANOS (VERDADEIRO/FALSO).

Quadro 4 – Fonte do Autor

6.1 COMANDOS BÁSICOS DE ENTRADA E SAÍDA(INPUT/OUTPUT)

- **LER/LEIA** ➔ Comando de entrada que permite a leitura de Variáveis de Entrada. Alguns autores tratam esse comando como **RECEBA**, o resultado é o mesmo.
- **ESCREVER/ESCREVA** ➔ Comando de saída que exibe uma informação na tela do monitor. Alguns autores tratam esse comando com EXIBA.
- **IMPRIMIR/IMPRIMA** ➔ Comando de saída que envia uma informação para a impressora.

  Exemplos:

Imagine que queremos obter um número do usuário e guardar em uma variável chamada NUM.

Ler (NUM);

Agora queremos pegar esse valor e calcular o dobro dele e guardar esse valor na variável DOBRO.

DOBRO ← NUM * 2;

Para exibir o resultado seria:

Escrever (DOBRO)

6.2 FLUXO DE UM ALGORITMO

Todo Algoritmo é composto por um processo de fluxo basicamente composto por entrada, processamento e saída, conforme a Figura 3.

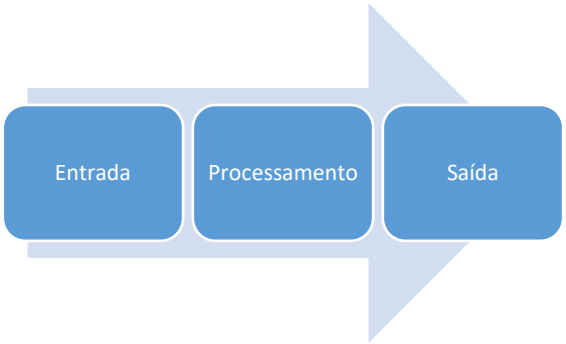


Figura 3 – Fonte do Autor

6.3 CORPO GERAL DE UM ALGORITMO

Iremos aprender o corpo geral de um algoritmo passo a passo, entendendo cada trecho do pseudocódigo.

Todo algoritmo precisa ter a primeira linha como uma identificação (nome) de acordo com o objetivo proposto:

ALGORITMO <<identificador>>;

Em seguida podemos declarar as constantes, que são sempre opcionais.

CONST

Declaramos o nome da constante e seu valor:

<<identificador>> = <<dado>>;

A declaração de variáveis é praticamente obrigatória, pois um programa sem variáveis só poderia existir para saída de informações, o que é incomum sem que haja uma entrada.

VAR

Devemos dar um nome a variável e definir o seu tipo (inicialmente iremos aprender usando apenas tipos primitivos de dados). Podemos colocar uma

variável em cada linha ou declarar muitas em uma mesma linha separando as por vírgulas, desde que elas sejam de mesmo tipo.

<<identificador1>> : <<tipo>>;
<<identificador1>> : <<tipo>>;

Finalmente iremos colocar a palavra reservada que determina o início do algoritmo, ela irá agrupar vários comandos.

ÍNICIO

Aqui podemos escrever os comandos de entrada e saída de dados, as fórmulas e os demais procedimentos.

<<comando1>>;
<<comandoN>>

Após o bloco de comandos iremos fechar o algoritmo com a respectiva palavra reservada.

FIM.

6.4 ESTRUTURAS SEQUÊNCIAIS

Como pode ser percebido no tópico anterior, todo programa possui uma estrutura sequencial determinada por um ÍNICIO e FIM.

6.4.1 CONSTRUINDO O PRIMEIRO ALGORITMO

Segue um algoritmo que vai receber dois números inteiros digitados pelo usuário e calcular a soma.

```
ALGORITMO SOMA;
VAR
    NUMERO1, NUMERO2, SOMA: INTEIRO;
INÍCIO
    LER (NUMERO1);
    LER (NÚMERO2);
    SOMA ← NUMERO1+NUMERO2;
    ESCREVER (SOMA)
FIM.
```

Observe que o algoritmo acima demonstra bem o fluxo definido anteriormente. Primeiro é feito a entrada de dados (leitura de variáveis), depois o processamento (cálculo da soma) e em seguida a saída de dados (exibição da soma obtida no processamento).

Agora se quiséssemos criar um programa baseado nesse algoritmo precisaríamos apenas estudar quais palavras reservadas da linguagem desejada exercem as funções desejadas pelo algoritmo. Observe os exemplos:

Transcrito na linguagem Pascal:

```
PROGRAM EXEMPLO:
VAR
    NUMERO1, NUMERO2, SOMA: INTEGER;
BEGIN
    READ (NUMERO1);
    READ (NÚMERO2);
    SOMA ← NUMERO1+NUMERO2;
    WRITE (SOMA)
END.
```

6.4.2 CONSTRUINDO OUTROS ALGORITMOS DE EXEMPLO

Segundo exemplo:

Segue um Algoritmo que lê o nome e as 4 notas bimestrais de um aluno. Em seguida o Algoritmo calcula e escreve a média obtida.

```
ALGORITMO MEDIA_FINAL;
VAR
NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: REAL;
NOME : CARACTERE [35] {podemos ou não definir o tamanho de caracteres que uma variável desse tipo pode assumir - STRING}
INÍCIO
        LER (NOME);
        LER (NOTA1, NOTA2, NOTA3, NOTA4);
        MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;
        ESCREVER (NOME);
        ESCREVER (MEDIA);
FIM.
```

Observe que sempre é mantido o fluxo (envolvido pelos retângulos) de entrada, de processamento e de saída.

Agora vamos exemplificar um algoritmo que utilize uma constante:

Façamos um Algoritmo que leia o raio de uma circunferência e calcule sua área. Sabendo que a área da circunferência é igual ao valor de π multiplicado pelo quadrado do raio, sendo π =3,1416.

```
ALGORITMO AREA_CIRCUNFERENCIA;
CONST
        PI = 3.1416;
VAR
        RAIO, AREA:REAL;
INÍCIO
        LER (RAIO);
        AREA ← PI * RAIO**2;
        ESCREVER (AREA)
FIM.
```

Para concluirmos nossos algoritmos sequenciais vamos fazer um quem tenha uma variável auxiliar.

Iremos criar um algoritmo que leia 2 números inteiros A e B, troque seu conteúdo e os exiba.

```
ALGORITMO TROCA_TUDO;
VAR
        A,B, AUXILIAR: INTEIRO;
INÍCIO
        LER(A);
        LER (B);
        AUX←A;
        A←B;
        B←A;
        ESCREVER (A,B)
FIM.
```

6.5 {LINHAS DE COMENTÁRIO}

Podemos inserir em um Algoritmo comentários para aumentar a compreensão do pseudocódigo, para isso bastam que os comentários fiquem entre Chaves “{}”.

Exemplo:

LER (RAIO); **{ENTRADA}**

6.6 ASPAS ‘SIMPLES’

Quando queremos exibir uma mensagem para a tela ou impressora ela deve estar contida entre aspas simples, caso contrário, o computador irá exibir o conteúdo de uma variável ou identificar a mensagem como Variável Indefinida.

Exemplo:

```
AREA := 180
ESCREVER (‘AREA OBTIDA =’,  AREA)          {COMANDO DE SAÍDA}
```

AREA OBTIDA = 180 {RESULTADO GERADO NA TELA}

CAPÍTULO 07
ESTRUTURAS CONDICIONAIS

7.1 ESTRUTURA “SE”

Executa uma sequência de comandos de acordo com o resultado de um teste.

A estrutura de decisão SE pode ser Simples ou Composta, baseada em um resultado lógico, a partir daí uma alternativa será executada.

SIMPLES:

SE <<CONDIÇÃO>> **ENTÃO**

<<COMANDO1>>;

<<COMANDON>>

São executados se a condição for VERDADEIRA!!!

FIM-SE

COMPOSTA:

SE <<CONDIÇÃO>> **ENTÃO**

<<COMANDO1>>;

<<COMANDON>>

São executados se a condição for VERDADEIRA!!!

SENÃO

<<COMANDO1>>**;**

<<COMANDON>>**;**

São executados se a condição for FALSA!!!

FIM-SE

Vamos a mais alguns exemplos:

Demonstraremos com um pseudocódigo que lê 2 números e escreve o maior.

```
ALGORITMO ACHA_MAIOR;
VAR A, B: INTEIRO;
INÍCIO
    LER (A, B);
    SE A>B ENTÃO
    ESCREVER (A)
    SENÃO
    ESCREVER (B)
    FIM-SE
FIM.
```

A seguir, apresentamos um algoritmo que solicita o nome e as quatro notas bimestrais de um aluno. Em seguida, o algoritmo calcula a média dessas notas e exibe o resultado, indicando se o aluno foi aprovado ou reprovado com base no critério estabelecido.

Considere a média para aprovação sendo maior ou igual a 6.

```
ALGORITMO MEDIA_FINAL;
VAR
NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: REAL;
NOME : CARACTERE [35]
INÍCIO
    LER (NOME);
```

LER (NOTA1, NOTA2, NOTA3, NOTA4);

MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;

SE MEDIA>=6 ENTÃO

ESCREVER (‘APROVADO’)

SENÃO

ESCREVER (‘REPROVADO’)**;**

ESCREVER (NOME, MEDIA)

FIM-SE

FIM.

7.2 NINHOS DE SE

Usados para tomadas de decisões com mais de 2 opções.

Forma Geral:

SE <<CONDIÇÃO>> **ENTÃO**

<<COMANDO1>>**;**

<<COMANDON>>

SENÃO

SE <<CONDIÇÃO>> **ENTÃO**

<<COMANDO1>>

SENÃO

<<COMANDO1>>

FIM-SE

FIM-SE

Exemplos:

Segue um Algoritmo que lê 3 números e escreve o maior.

```
ALGORITMO ACHA_MAIOR;
VAR A, B, C : INTEIRO;
INÍCIO
        LER (A, B, C);
        SE (A>B) E (A>C)ENTÃO
                ESCREVER (A)
        SENÃO
                SE (B>A) E (B>C)ENTÃO
                        ESCREVER (B)
                SENÃO
                        ESCREVER (C)
                FIM-SE
        FIM-SE
FIM.
```

Segue outro exemplo bastante comum para fins de aprendizagem:

Algoritmo que lê os valores referentes aos lados de um triângulo e classifica ele dentre: equilátero, isósceles ou escaleno.

```
ALGORITMO TRIÂNGULO;
VAR
        L1, L2, L3;REAL;
INÍCIO
        ESCREVER (´DIGITE O PRIMEIRO LADO´);
        LER (L1);
        ESCREVER (´DIGITE O SEGUNDO LADO´);
        LER (L2);
        ESCREVER (´DIGITE O TERCEIRO LADO´);
```

```
        LER (L3);
        SE L1=L2 E L2=L3 ENTÃO
                ESCREVER (´TRIÂNGULO EQUILÁTERO´);
        SENÃO
                SE L1=L2 OU L1=L3 OU L2=L3 ENTÃO
                        ESCREVER (´TRIÂNGULO ISÓSCELES´)
                SENÃO
                        ESCREVER (´TRIÂNGULO ESCALENO´)
                FIM-SE
        FIM-SE
FIM.
```

Observe a lógica do algoritmo:

Se L1=L2 e L1=L3, os 3 lados são iguais, logo o triângulo é equilátero.

Se L1=L2 ou L1=L3 ou L2=L3, lembrando que essa decisão só vai acontecer se o triângulo não for escaleno. Caso qualquer uma dessas opções seja verdadeira o triângulo é isósceles.

Logo se o triângulo não for equilátero e nem isósceles, obviamente ele somente poderá ser escaleno.

7.3 ESTRUTURA “CONFORME”

A estrutura de condição CONFORME equivale a um ninho de SE, usada quando dispomos de mais de 2 opções para uma decisão. Seu modo é mais fácil de ser compreendido:

Forma Geral:

CONFORME

CASO <<CONDIÇÃO1>>

<<COMANDO1>>;

CASO <<CONDIÇÃON>>

<<COMANDO1>>;

OUTROS CASOS

<<COMANDO1>>;

FIM CONFORME

Iremos usar o mesmo exemplo do utilizado anterior para demonstrar a estrutura CONFORME, perceba como o algoritmo fica menor e de melhor compreensão.

Segue um Algoritmo que lê 3 números e escreve o maior.

```
ALGORITMO ACHA_MAIOR;
VAR A, B, C : INTEIRO;
INÍCIO
    LER (A, B, C);
    CONFORME
    CASO (A>B) E (A>C)
            ESCREVER (A);
```

```
        CASO (B>A) E (B>C)
ESCREVER (B);
        OUTROS CASOS
ESCREVER (C);
        FIM CONFORME
FIM.
```

Podemos ter inúmeras opções (CASOS) em uma estrutura conforme. A Condição OUTROS CASOS seria equivalente a um SENÃO, ou seja, se nenhuma das opções for verdadeira ela é executada. Logo podemos concluir que a opção OUTROS CASOS, pode ou não existir nessa estrutura (não há obrigatoriedade).

Caso haja várias condições que possam ser verdadeiras dentro da estrutura, somente a **primeira** será executada, desprezando todas as demais posteriores.

8.1 ESTRUTURA DE REPETIÇÃO DETERMINADA - “PARA”

Quando uma sequência de comandos deve ser executada repetidas vezes, tem-se uma estrutura de repetição. A estrutura de repetição, assim como a de decisão, envolve sempre a avaliação de uma condição. Na estrutura de repetição PARA, o algoritmo apresenta previamente a quantidade de repetições desejadas.

Forma Geral:

PARA <<VARIAVEL INTEIRA>> **:=** <<VALOR INICIAL>> **ATÉ** <<VALOR FINAL>> **FAÇA**

<<COMANDO1>>;

FIM PARA

A repetição por padrão determina o passo do valor inicial até o valor final como sendo 1. Determinadas linguagens possuem passo –1 ou permitem que o programador defina o passo.

Segue abaixo um algoritmo que vai exibir 5 vezes o nome “RENATO” de uma maneira primitiva:

```
ALGORITMO REPETE_1;
INÍCIO
        ESCREVER (´RENATO´);
        ESCREVER (´RENATO ´);
        ESCREVER (´RENATO ´);
        ESCREVER (´RENATO ´);
        ESCREVER (´RENATO ´);
FIM.
```

Que algoritmo grande para algo tão simples, não? Ainda bem que no Word existe Copiar e Colar caso contrário não colocaria esse exemplo... Imagine ainda se fosse para exibir 100 vezes! Ufa... Vamos ver do modo mais simples agora:

```
ALGORITMO REPETE2;
VAR I:INTEIRO;
INÍCIO
        PARA I=1 ATÉ 5 FAÇA
                ESCREVER (´RENATO´)
        FIM PARA
FIM.
```

VARIÁVEL IMPLEMENTADA DE 1 EM 1

Desse modo se quiséssemos repetir 100 vezes, bastava substituir o valor final da repetição de 5 por 100.

Segue um algoritmo que escreve os 100 primeiros números pares.

```
PROGRAMA PARES;
VAR I, PAR: INTEGER;
INÍCIO
        PAR:=0;
        PARA I:=1 ATÉ 100 FACA
                ESCREVER (PAR);
                PAR := PAR+2
        FIM PARA
FIM.
```

Segue um algoritmo, quase que padrão, exigido como exercício nos cursos de graduação: criar um algoritmo que leia um número N e calcule seu fatorial:

```
ALGORITMO FATORIAL;
VAR
        FAT, N, I :INTEIRO;
INÍCIO
        FAT := 1;
        LER (N);
        PARA I DE 1 ATÉ N FAÇA
        FATORIAL = FATORIAL * I
        FIM PARA
FIM
```

8.2 ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO INICIAL – "ENQUANTO"

É usada para repetir N vezes uma ou mais instruções. Tendo como vantagem o fato de não ser necessário o conhecimento prévio do número de repetições.

VALIDAÇÃO INICIAL

ENQUANTO <<CONDIÇÃO>> **FAÇA**

        <<COMANDO1>>;

FIM ENQUANTO

Segue um algoritmo que calcule a soma dos salários dos funcionários de uma empresa. O programa termina quando o usuário digitar um salário menor que 0.

```
ALGORITMO SOMA_SALÁRIOS;
VAR SOMA, SALÁRIO: REAL;
INÍCIO
        SOMA:=0;
        SALÁRIO:=0;
        ENQUANTO SALÁRIO>=0 FAÇA
                LER (SALÁRIO);
                SOMA:=SOMA+SALÁRIO
        FIM ENQUANTO
        ESCREVER (SOMA)
FIM.
```

Se o primeiro valor testado for falso, a repetição terminará sem que suas instruções sejam executadas, pois o teste precede os comandos. Logo o número de repetições varia de 0 a N vezes.

A estrutura de repetição ENQUANTO pode substituir a estrutura PARA, mas a recíproca nem sempre é verdadeira, lembrando que cada situação deve ditar a estrutura ideal.

Abaixo segue um exemplo de cálculo de fatorial utilizando a estrutura ENQUANTO.

```
ALGORITMO FATORIAL;
VAR
        FAT, N, I :INTEIRO;
INÍCIO
        FAT := 1;
        LER (N);
```

```
        I=0;
ENQUANTO I<=N FAÇA
        FATORIAL = FATORIAL * I
        I:=I+1    {CONTADOR}
        FIM ENQUANTO
FIM
```

TODAS AS VARIÁVEIS QUE ACUMULAM VALORES DEVEM RECEBER UM VALOR INICIAL.

REPITA ENQUANTO FOR VERDADEIRO!

8.3 ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO FINAL – "REPITA ATÉ"

Assim como a estrutura ENQUANTO É usada para repetir N vezes uma ou mais instruções.

Sua principal característica é ter a validação final, fazendo com que a repetição seja executada ao menos uma vez.

Forma Geral;

```
REPITA
        <<COMANDO1>>;
        <<COMANDON>>
ATÉ <<CONDIÇÃO>>
```

VALIDAÇÃO FINAL

Segue um algoritmo que calcule a soma dos salários dos funcionários de uma empresa. O programa termina quando o usuário digitar um salário menor que 0.

```
ALGORITMO SOMA_SALÁRIOS;
VAR
SOMA, SALÁRIO: REAL;
INÍCIO
        SOMA:=O;
        REPITA
                LER (SALÁRIO);
                SOMA:=SOMA+SALÁRIO
        ATÉ SALÁRIO<0;
        ESCREVER (SOMA)
FIM.
```

Segue um outro exemplo para análise, um algoritmo que escreve os 100 primeiros números pares.

```
ALGORITMO PARES_2;
VAR I, PAR, CONTADOR: INTEIRO;
INÍCIO
        CONTADOR := 0;
PAR := 0;
        REPITA
                ESCREVER (PAR);
                PAR := PAR+2;
                CONTADOR := CONTADOR+1;
        ATÉ CONTADOR=100
FIM.
```

REPITA ATÉ SER FALSO!

8.4 LOOP E CUIDADOS

Devemos ter atenção ao trabalharmos com estruturas de repetição indeterminada, para não cairmos no erro de Loop, ou seja, criarmos um programa com uma rotina que se torne eterna, pois o processamento irá causar erro e travamento da máquina.

Trecho de algoritmo com exemplo de Loop Eterno:

```
ENQUANTO 10>2 FAÇA
        ESCREVER (´BRASIL´)
FIM ENQUANTO
```

Como 10 sempre será uma constante maior do que 2, a condição sempre será verdadeira e o programa irá repetir a palavra BRASIL infinitamente caracterizando um comportamento inadequado que poderá levar ao travamento da máquina, sendo considerado um bug (erro).

CAPÍTULO 09
INTRODUÇÃO À ESTRUTURAS DE DADOS

A Estrutura de Dados é uma técnica de programação que possibilita armazenar dados na memória ou em disco objetivando atingir o menor consumo de espaço possível no menor tempo.

9.1 TIPOS PRIMITIVOS DE DADOS

Conforme apresentado na seção 5.4.1, os tipos primitivos de dados são nativos da linguagem e a partir deles podem ser definidos novos tipos.

Ex: Inteiro, Real, Caractere e Lógico.

9.2 TIPOS DE DADOS ESTRUTURADOS/COMPOSTOS

Os tipos de dados estruturados ou compostos são formados a partir de tipos primitivos já existentes e possuem um tamanho fixo, definido no momento da sua declaração. Durante a execução do programa, sua estrutura não sofre alterações no escopo ou na quantidade de elementos armazenados.

- Vetor (Array Unidimensional): Conjunto de elementos do mesmo tipo, acessíveis por um índice.
- Matriz (Array Multidimensional): Estrutura tabular com múltiplas dimensões.
- Registro (Struct ou Record): Agrupa diferentes tipos de dados sob um único identificador.

9.3 TIPOS DE DADOS DINÂMICOS

São aqueles que sofrem alterações durante a execução do programa.

- Ex: Ponteiros (Pilha, Fila, Lista, Árvore).

9.4 VETOR (ARRAY)

O vetor é uma estrutura de dados homogênea (de mesmo tipo) e unidimensional.

Imagine que queremos criar um algoritmo para ler 15 idades, calcular e escrever a soma, a média e depois exibir uma idade qualquer solicitada pelo usuário.

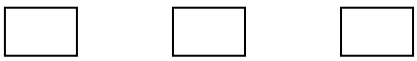
Inicialmente poderíamos pensar em usar apenas uma variável, e acumular seus valores para calcular a soma e a média, entretanto com isso a cada idade lida, a anterior seria apagada da memória não permitindo uma posterior consulta do usuário.

Podemos então criar um programa com 15 variáveis de idade, solução fácil, mas, que não é viável diante da quantidade excessiva de pseudocódigo. Suponha que houvesse uma alteração no programa, e não fossem mais 15 idades e sim 150, a dificuldade que seria para implementação.

A necessidade de utilizar um Vetor surge nessa situação, ele nada mais é que um conjunto de variáveis de mesma característica.

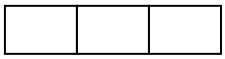
Abaixo segue a representação de três variáveis:

Idade1
Idade2
Idade3



Abaixo segue a representação de 1 vetor para substituir as variáveis:

Idade



Idade[1]
Idade[2]
Idade[3]

O vetor possuí um índice que identifica a posição do dado armazenado, esse índice é do tipo Inteiro e fica representado entre colchetes.

Podemos imaginar o vetor desta forma:

Cidadão Nº (índice do vetor)	Idade
1	18
2	23
3	30
4	15
5	20
6	23
7	18
8	19
9	20
10	65
11	23
12	56
13	22
14	24
15	9

Onde a coluna cidadão é o Índice do Vetor e a coluna Idade é o Vetor.

Forma geral para declaração do vetor:

<<NOME>> : VETOR[1..N] DE <<TIPO>>;

Segue então a resolução do algoritmo utilizado no exemplo:

```
ALGORITMO VETOR_IDADE;
VAR
        IDADE:VETOR[1..15] DE INTEIRO;
        SOMA, CIDADAO, I: INTEIRO;
        MEDIA:REAL;
INICIO
        SOMA:=0;
        PARA I:=1 ATÉ 15 FAÇA
                LER(IDADE[I])
                SOMA:=SOMA+IDADE[I]
        FIM-PARA
```

```
        ESCREVER (SOMA);
        MEDIA := SOMA/15;
        ESCREVER (MEDIA);
        ESCREVER (´DIGITE O NUMERO DO CIDADAO O QUAL DESEJA OBTER A IDADE´);
        LER (CIDADAO);
        ESCREVER (IDADE[CIDADAO])
FIM.
```

A estrutura de repetição PARA é utilizada para carregar o vetor com dados obtidos via teclado através do usuário e acumular os dados na variável SOMA, com um código muito mais enxuto do que seria a leitura de 15 variáveis referentes a cada idade, uma por uma.

CAPÍTULO 10
INTRODUÇÃO À ESTRUTURAS DE DADOS

Ao longo deste livro, exploramos os fundamentos da lógica de programação, abordando desde a construção de algoritmos até a aplicação de estruturas de controle, tipos de dados e estruturas de repetição. Utilizamos a pseudolinguagem Portugol como ferramenta de aprendizado, permitindo que os conceitos fossem assimilados sem a complexidade inicial das linguagens de programação formais. Essa base é essencial para qualquer pessoa que deseja ingressar no mundo da computação e se tornar um desenvolvedor eficiente.

No entanto, este é apenas o começo da jornada. O próximo passo para aprofundar o conhecimento em programação envolve o estudo de estruturas de dados mais complexas, estendendo o estudo de vetores e dando cadência ao estudo de matrizes, que são fundamentais para armazenar e manipular conjuntos de informações de forma eficiente. Além disso, é importante compreender funções e modularização, que permitem a organização do código de maneira mais estruturada e reutilizável.

Com a lógica bem estabelecida, recomenda-se a transição para linguagens de programação amplamente utilizadas no mercado, como Python, Java e C. O Python, por exemplo, é uma excelente opção para iniciantes, pois possui uma sintaxe simples e legível, além de ser amplamente aplicado em diversas áreas, como desenvolvimento web, ciência de dados, automação e inteligência artificial.

Por fim, vale lembrar que aprender a programar é um processo contínuo. Cada novo desafio superado fortalece o pensamento lógico e abre portas para novas oportunidades no mundo da tecnologia. Espero que este livro tenha sido um guia útil em sua trajetória e que os conhecimentos adquiridos aqui sirvam como base para desafios cada vez mais complexos.

Desejo sucesso em seus estudos e no desenvolvimento de suas habilidades na programação!

REFERÊNCIAS

KNEUSEL, Ronald. T. ***Como a Inteligência Artificial Funciona***. São Paulo: Novatec. ISBN 9788575228852

GUIMARÃES, A. M., & Lages, N. A. C. (1984). ***Algoritmos e Estruturas de Dados***. Rio de Janeiro: LTC. ISBN 8521603789.

KOFFMAN, Elliot B. (1999). ***Pascal Estruturado: Programação e Estrutura de Dados***. Rio de Janeiro: LTC Editora. ISBN 8521611749.

ANEXO 1

QUESTÕES DE CONCURSOS DA BANCA CESGRANRIO

1) Analise o algoritmo abaixo em português estruturado

```
algoritmo segredo;
variáveis
   x, y, z : inteiro;
fim-variáveis
início
   x := 15;
   y := 10;
   z := 0;

   enquanto y > 0 faça
      z := z + x;
      y := y - 1;
   fim-enquanto
   imprima(z);
fim
```

Que número seria impresso caso esse programa executasse?

a) 0

b) 10

c) 15

d) 100

e) 150

2) Algoritmo faca_contas

```
inicio
   numero = 3
   para i de 1 ate 5, faca
      leia (X)
      se X > 4 então
         numero ← numero + X
      caso contrario
         numero ← numero - X
      fim-se
   fim-para
   escreva (numero)
fim-algoritmo
```

Qual é a saída do algoritmo faca_contas para a entrada 7, 3, 5, 2, 3 ?

a) 3

b) 6

c) 7

d) 10

e) 12

3) Considere o seguinte algoritmo, descrito em pseudocódigo, que manipula um vetor de oito posições, indexadas de 1 a 8

```
INÍCIO
   INTEIROS: i, temp
   VETOR M[1..8] de INTEIROS
   M ← [10, 20, 30, 40, 50, 60, 70, 80]
   PARA-TODO i de 1 a 8
      temp ← M[i]
```

```
        M[i] ← M[9 - i]
        M[9 - i] ← temp
    FIM-PARA-TODO
FIM
```

Ao final da execução, o conteúdo do vetor M será

a) 10, 20, 30, 40, 50, 60, 70, 80

b) 40, 10, 80, 20, 70, 30, 60, 50

c) 40, 30, 20, 10, 80, 70, 60, 50

d) 50, 60, 70, 80, 10, 20, 30, 40

e) 80, 70, 60, 50, 40, 30, 20, 10

4) Beto, estudante de programação, ao resolver um problema, testou diversos algoritmos e, com um deles, achou a seguinte impressão com o resultado correto: soma = 39.

Esse resultado foi entregue ao professor. Indagado sobre com qual algoritmo tinha encontrado o resultado correto, Beto percebeu que não mais lembrava. E teve que repetir as experiências.

Qual foi o algoritmo utilizado pelo estudante?

a)

```
inicio
    variável inteiro chave, soma
    soma <- 0
    chave <- 10
    enquanto chave < 18 faz
        soma <- soma + chave
        chave <- chave + 3
    fim enquanto
    escrever "soma =", soma
fim
```

b)

```
inicio
    variável inteiro chave, soma
    soma <- 0
    chave <- 10
    enquanto chave < 18 faz
        escrever "soma =", soma
        soma <- soma + chave
        chave <- chave + 3
    fim enquanto
fim
```

c)

```
inicio
    variável inteiro chave, soma
    soma <- 0
    chave <- 10
    escrever "soma =", soma
    enquanto chave < 18 faz
        soma <- soma + chave
        chave <- chave + 3
    fim enquanto
fim
```

d)

```
inicio
    variável inteiro chave, soma
    soma <- 0
    chave <- 10
```

```
    se chave < 18 então
        soma <- soma + chave
        chave <- chave + 3
    fim se
    escrever "soma =", soma
fim
```

e)

```
inicio
    variável inteiro chave, soma
    soma <- 0
    chave <- 10
    se chave < 18 então
        chave <- chave + 3
        soma <- soma + chave
    fim se
    escrever "soma =", soma
fim
```

5)

```
inicio
    inteiro X, Y
    Ler X
    Ler Y
    Enquanto X ≥ Y - 1 faça
        X <- X - 1
        Y <- Y + 2
```

```
   Fim Enquanto
   Escrever "saída =", Y - X
fim
```

A saída do algoritmo apresentado acima para as entradas 100 e 20, respectivamente, é

a) -5

b) -2

c) 1

d) 4

e) 7

6) A avaliação de uma disciplina é feita de tal forma que, um aluno, para ser aprovado, deverá realizar, primeiramente, 3 provas (P1, P2 e P3), a partir das quais será obtida uma média M1 (média aritmética de P1, P2 e P3). Dependendo dessas notas e dessa média, o aluno terá ou não que fazer uma quarta prova (P4). Nesse caso, a nova média (MF) será calculada pela média aritmética na qual se substitui a menor das notas P1, P2 e P3 pela P4. O critério de aprovação é o seguinte:

```
SE (P1 ≥ 4 E P2 ≥ 4 E P3 ≥ 4 E M1 ≥ 6) ENTÃO
   APROVADO
SENÃO
   SE (P4 ≥ 5 E MF ≥ 5) ENTÃO
      APROVADO
   SENÃO
      REPROVADO
```

Qual dos conjuntos de notas a seguir permite a aprovação do aluno?

a) P1 = 0, P2 = 5, P3 = 6, P4 = 5

b) P1 = 3, P2 = 5, P3 = 3, P4 = 5

c) P1 = 3, P2 = 6, P3 = 9, P4 = 3

d) P1 = 3, P2 = 7, P3 = 9, P4 = 4

e) P1 = 4, P2 = 5, P3 = 4, P4 = 5

7) Considere o seguinte algoritmo:

```
INÍCIO
   S := 0;
   Obter n do teclado;
   ENQUANTO (n ≥ 0)
         {
         S := S + n;
         SE (S ≥ 100)
                        S := 0;
         Obter n do teclado;
         }
   Imprimir S;
FIM
```

Se a sequência de números digitados pelo teclado for

20, 8, 32, 40, 35, 11, 27, 11, 32, -16,

o resultado impresso será

a) 0

b) 16

c) 32

d) 73

e) 84

8) Analise o algoritmo abaixo, onde a%b representa o resto da divisão de a por b.

```
inicio
    inteiro x, y, i, r
    ler x
    ler y
    para i de 1 até x
        se (x % i = 0) e (y % i = 0) então
            r <- i
        fim se
    próximo
    escrever r
fim
```

Qual será a resposta, caso as entradas sejam 128, para x, e 56, para y?

a) 2

b) 8

c) 56

d) 64

e) 128

9) Dado o vetor W tal que $W[i] = i \text{ para } 1 \leq i \leq 9$

Dado o algoritmo

```
algoritmo Z
    para j de 1 até 3, faça
        q = j + 1
        aux ← W[q]
        W[q] ← W[9 - q]
        W[9 - q] ← aux
    fim para
fim algoritmo Z
```

Qual será o conteúdo de vetor W depois de executado o algoritmo Z?

a) 176543289

b) 987654321

c) 543219876

d) 987651234

e) 123456789

10) Observe o fragmento de código abaixo.

```
x = 3;
y = 4;
z = 5;
if ((x - 1) > 2)
   y = y + 1;
else
   y = y - 1;
z = x + y;
for (i = 1; i < 9; i++)
   y = y + 1;
z = z + y;
```

Ao final da execução desse código, qual o valor de z?

a) 12

b) 15

c) 16

d) 17

e) 20

11) Suponha que o primeiro elemento do vetor no pseudocódigo abaixo esteja posicionado no índice 1.

```
para x ← 2 até tamanho_vetor faça
   conteudo ← vetor[x]
   y ← x - 1
   enquanto y > 0 e vetor[y] > conteudo, faça
      vetor[y + 1] ← vetor[y]
      y ← y - 1
   fim enquanto
   vetor[y + 1] ← conteudo
fim para
```

Qual o resultado final da variável "vetor" a partir da entrada {5,4,3,2,8}?

a) {2,3,4,5,8}

b) {4,3,2,1,7}

c) {5,3,2,1,7}

d) {6,5,4,3,9}

e) {8,2,3,4,5}

12) Considere o algoritmo abaixo, em pseudocódigo.

```
Algoritmo CESGRANRIO
Var
   A, R: real
   NOME: literal[32]
Início
   Leia A, NOME
   Escolha
      Caso NOME = "João"
         R ← 5 * A
```

```
      Caso NOME = "Maria"
         R ← 2 * A
      Senão
         R ← A
   Fim_escolha
   A ← 3 + A
   Escreva "Resultado: ", R
Fim
```

Suponha que os dados de entrada A e NOME sejam, respectivamente, "1" e "Maria". Qual a saída do algoritmo?

a) 1

b) 2

c) 3

d) 4

e) 5

13) Considere o algoritmo abaixo, descrito em pseudocódigo.

```
Algoritmo GRATIFICAÇÃO
Var
   SALARIO, SALARIO_GRATIF: real
   TITULO: literal[32]
Início
   Leia SALARIO, TITULO
   Escolha
      Caso TITULO = "Doutor"
         SALARIO_GRATIF ← 3 * SALARIO
      Caso TITULO = "Mestre"
```

```
        SALARIO_GRATIF ← 2 * SALARIO
    Senão
        SALARIO_GRATIF ← SALARIO
  Fim_escolha
  Escreva "Salário com a gratificação = ", SALARIO_GRATIF
Fim
```

Suponha que os dados de entrada SALÁRIO e TITULO sejam, respectivamente, "1000,00" e "Doutor". Qual o valor do salário com a gratificação, em reais, que será apresentado na saída do algoritmo?

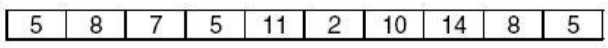
a) 1000,00
b) 2000,00
c) 3000,00
d) 5000,00
e) 6000,00

14) Seja a seguinte sub-rotina:

```
Algoritmo
  declare X, Y, AUX numérico
  declare V[1:10] numérico
  X ← 1
  Y ← 0
  AUX ← 0
  repita
    se X > 10 então
      interrompa
    fim se
    se V[X] > Y então
      Y ← V[X]
```

```
            AUX ← AUX + 1
        fim se
        X ← X + 1
    fim repita
    AUX ← AUX * Y
    escreva AUX
fim algoritmo
```

"V" é variável composta unidimensional contendo os 10 elementos numéricos abaixo.



5	8	7	5	11	2	10	14	8	5

Pode-se afirmar que o valor da variável "AUX" na linha "escreva AUX" é:

a) 21

b) 40

c) 56

d) 64

e) 70

15) Dadas as variáveis numéricas A e B, contendo os valores 2 e 6, respectivamente; a variável L, contendo o literal FALSO; e a variável lógica V, contendo o valor lógico verdadeiro, assinale a expressão lógica cujo resultado possui valor lógico falso.

a) A2 > B ou V

b) A > B ou L = "FALSO"

c) A < B e L = "LITERAL"

d) A > B e V ou L = "FALSO"

e) A - B < 2 e L ? "VERDADEIRO" e V

16)

```
Algoritmo
   declare A, B, C numérico
   leia A
   leia B
   C ← (A + B) * B
   escreva C
fim algoritmo
```

Com base no algoritmo acima, e supondo que o valor fornecido para "A" na linha "leia A" seja 3 e o valor fornecido para "B" na linha "leia B" seja 4, pode-se afirmar que o valor da variável "C" na linha "escreva C" é:

a) 24

b) 28

c) 32

d) 34

e) 43

17) Seja a seguinte sub-rotina:

```
Algoritmo
   declare A, B, C, D numérico
   leia A
   leia B
   leia C
   D ← 0
   se A < B e A < C então
      D ← A
   senão se B < C então
      D ← B
```

```
   senão
      D ← C
   fim se
   escreva D
fim algoritmo
```

Com base no algoritmo acima, e supondo que o valor fornecido para "A" na linha "leia A" seja 10, o valor fornecido para "B" na linha "leia B" seja 7 e o valor fornecido para "C" na linha "leia C" seja 4, pode-se afirmar que o valor da variável "D" na linha "escreva D" é:

a) 0
b) 4
c) 7
d) 10
e) 14

18)

```
Algoritmo
   declare A, B numérico
   leia B
   A ← 0
   repita
      B ← B + 2
      se B > 10 então
         interrompa
      fim se
      A ← A + B
   fim repita
   escreva A
fim algoritmo
```

Com base no algoritmo acima, e supondo que o valor fornecido para "B" na linha "leia B" seja 5, pode-se afirmar que o valor da variável "A" na linha "escreva A" é:

a) 7

b) 9

c) 12

d) 16

e) 21

GABARITO

1. E
2. C
3. A
4. A
5. D
6. A
7. B
8. B
9. A
10. D
11. A
12. B
13. C
14. C
15. C
16. B
17. B
18. D

Agência Brasileira ISBN
ISBN: 978-65-982396-5-7