CENTRO DE EDUCAÇÃO PROFISSIONAL JOSÉ BUSS – CEDUP
Curso Técnico em Informática
Disciplina: Lógica de Programação
Professor (a): Geovane Pereira da Silva

MATERIAL DE APOIO DA DISCIPLINA LÓGICA DE PROGRAMAÇÃO

Rio Fortuna
2026

1. INTRODUÇÃO

Este material de apoio foi organizado para ser utilizado como roteiro base de estudos da disciplina de Lógica de Programação no Curso Técnico em Informática. Não utilize ele como única fonte de estudos. Pesquise também em outros livros e materiais confiáveis para formar seu conhecimento.

Habilidades Interpretar a lógica computacional. Estruturar e desenvolver pseudocódigo, algoritmos e fluxogramas. Desenvolver raciocínio lógico para a criação de programas básicos. Compreender a estrutura lógica de um aplicativo para utilizar em uma linguagem.
Objetos de conhecimentos ● Álgebra booleana; ● Tabela verdade; P ● Proposição e conectivos lógicos (e, ou, se…então, se e somente se, negação); ● Sequência lógica; ● Representação de algoritmos (narrativa, fluxograma e pseudocódigo), organogramas e representações gráficas; tipos de dados; variáveis e constantes; tipos de operadores (aritméticos, relacionais, literais e lógicos); ● Estruturas de controle (estrutura de decisão e estrutura de repetição); ● Pseudocódigos; ● Termos técnicos e vocabulário básico de desenvolvimento; ● Ferramentas para elaboração de algoritmos; ● Teste de mesa; ● Estruturas de dados homogêneas (vetores e matrizes); ● Modularização, identificação e comentários de código.

2. Conceito de Lógica de Programação

Sem dúvida, o computador é uma das maiores invenções do homem e tem se mostrado uma ferramenta versátil, rápida e segura para a manipulação de informações. Para muitos, essa invenção é responsável pela intensificação da mecanização e descobertas científicas na vida moderna. Esta afirmação dá um caráter autônomo ao computador, como se o mesmo fizesse tudo sozinho.

Entretanto cabe esclarecer que o computador não é criativo e nem inteligente, na verdade, apenas reproduz o que lhe é ordenado por meio de seus programas de computador, os quais são construídos para resolver algum problema específico e a solução adotada é sempre uma solução lógica. Podemos entender essa “solução lógica” como uma espécie de receita de bolo a ser adotada para a solução do problema. A Lógica de Programação é então o passo inicial para a construção de um programa de computador

Para usar a lógica, é necessário ter domínio sobre o pensamento, bem como saber pensar, ou seja, possuir a “Arte de pensar”. Alguns definem o raciocínio lógico como um conjunto de estudos que visa determinar os processos intelectuais que são as

condições gerais do conhecimento verdadeiro. Isso é válido para a tradição filosófica clássica aristotélico-tomista, também conhecida por Lógica Formal.

Em contraposição a esse conceito de Lógica Formal, surgiu um outro – o de Lógica Material – para designar o estudo do raciocínio no que ele depende quanto ao seu conteúdo ou matéria. Esta distinção entre lógica formal e lógica material permite-nos agora perceber porque, tendo em vista a sua forma, o raciocínio é correto ou incorreto (válido ou invalido). Mas se atendermos à sua matéria, a conclusão pode ser verdadeira ou falsa. Exemplo:

Nenhum homem sabe dançar;
O dançarino é homem;
O dançarino não sabe dançar.

Este raciocínio é formalmente correto, uma vez que a conclusão está corretamente deduzida. Mas a conclusão é falsa, uma vez que é falsa a primeira proposição (“Nenhum homem sabe dançar”). Estamos perante um raciocínio que tem validade formal, mas não tem validade material. Logo temos que concluir que é falso.

Desde a sua criação a lógica tem registrado enormes aperfeiçoamentos, sobretudo, a partir de meados do século XIX. É costume dividir a sua história em três períodos: período Clássico, período Moderno e período Contemporâneo.
A Lógica Matemática ou Simbólica (surgida no Período Moderno), é usada para expressar o conteúdo do pensamento simbólico e defende a ideia de que o raciocínio é visto como cálculo matemático. A Lógica Matemática exerceu uma influência decisiva em muitos domínios, principalmente na Eletrônica, Cibernética, Informática, Inteligência Artificial, dentre outros.
Mesmo com essa multiplicidade de conceitos, a lógica pode ser vista como uma ciência que procura encontrar as leis em relação às quais o nosso pensamento deve obedecer para que possa ser considerado válido.
No contexto da informática, a Lógica de Programação é a técnica de encadear pensamentos para atingir determinado objetivo previamente definido, ou seja, é a técnica que nos permite expressar o que deve ser feito e em que ordem para que a solução seja alcançada.

Sempre que pensamos estamos exercitando o uso da lógica. Toda vez que falamos também estamos fazendo uso da lógica uma vez que a fala é apenas uma representação do que pensamos. Quantas vezes em nosso cotidiano, dissemos afirmações do tipo: “Isso é lógico!”, “... Não tem lógica alguma.”, “Não vejo lógica nisso.”. Saber o que é lógico, ou saber identificar uma estrutura lógica em um contexto linguístico, é algo que nos é transmitido por meio da nossa educação.
Além dessa lógica linguística, aplicamos outros tipos de raciocínio lógico em nosso dia-a-dia. Um bom exemplo seria porque não colocamos nossa mão em uma superfície quente. Isso parece lógico, não é? Na verdade, isso acontece porque nosso cérebro processa sentenças lógicas como:

A pele humana não suporta altas temperaturas (ou algo mais simples como: "queimei minha pele no último contato com uma superfície quente");
A minha mão é coberta de pele;
Logo, a minha mão não suporta altas temperaturas.

Isso deixa claro que nós pensamos de forma lógica o tempo todo. No entanto, temos uma grande dificuldade em formalizar este raciocínio lógico, pois não somos acostumados a formalizar nosso pensamento

O uso da lógica é um fator a ser considerado por todos, principalmente pelos profissionais da área de Tecnologia da Informação (programadores, analistas de sistemas), pois seu dia-a-dia dentro das organizações é solucionar problemas e atingir os objetivos apresentados por seus usuários com eficiência e eficácia, utilizando recursos computacionais. Vale ressaltar que ninguém ensina ninguém a pensar, pois todas as pessoas normais possuem esse "dom". O objetivo deste material é mostrar como desenvolver a aperfeiçoar melhor essa técnica para dizer ao computador o que deve ser feito para atingir a solução de um determinado problema

Algoritmos: aplicabilidade da lógica no desenvolvimento de programas

A construção de algoritmos é o primeiro passo para o desenvolvimento de programas de computador. É uma das tarefas mais complexas da programação de computadores, mas também uma das mais desafiadoras e empolgantes.

O termo algoritmo também é utilizado em outras áreas como: engenharia, administração, entre outras. Algumas definições de algoritmo:

- **Uma sequência de passos que visa atingir um objetivo definido.**
- **Um procedimento passo a passo para a solução de um problema.**
- **Uma sequência detalhada de ações a serem executadas para realizar alguma tarefa**

Ingredientes:
4 xícaras (chá) de farinha de trigo.
2 xícaras (chá) de açúcar cristal.
2 xícaras (chá) de achocolatado.
2 colheres (sopa) de fermento em pó.
1 pitada de sal.
3 ovos.
2 xícaras (chá) de água morna.
1 xícara (chá) de óleo.
Óleo para untar.
Farinha de trigo para polvilhar.

Modo de preparo:
Numa vasilha, misture 4 xícaras (chá) de farinha de trigo, 2 xícaras (chá) de açúcar cristal, 2 xícaras (chá) de achocolatado, 2 colheres (sopa) de fermento em pó e 1 pitada de sal. Junte 3 ovos, 2 xícaras (chá) de água morna e 1 xícara (chá)

de óleo. Misture bem. Unte uma forma retangular de 25 cm x 37 cm com óleo e polvilhe farinha de trigo e despeje a massa. Asse em temperatura média (de 170°C a 180°C) por 30 minutos.

A receita tem todas as características de um algoritmo. Ela tem uma sequência detalhada de passos, descrita no modo de preparo. Apresenta a tarefa a ser realizada, que no caso é o bolo de chocolate. Além disso, podemos identificar na receita entradas (no caso, os ingredientes) e uma saída, que é o próprio bolo.

Em programas de computador os algoritmos possuem elementos básicos que são necessários:

O algoritmo não é a solução do problema, mas uma forma de solucioná-lo. Assim, para um mesmo problema, podemos criar diferentes algoritmos usando diferentes abordagens. Em outras palavras, podemos usar diferentes sequências de instruções para resolver o mesmo problema. Em alguns casos, até mesmo diferentes instruções.
A construção de um algoritmo deve observar todos os passos necessários à execução da atividade e evitar que passos desnecessários sejam executados ou que passos interdependentes sejam executados fora de ordem.
Durante a construção de um algoritmo são realizadas constantes revisões a fim de identificar novas situações ou exceções a serem tratadas. Quando temos um problema e precisamos construir um algoritmo para resolvê-lo, devemos passar pelas seguintes etapas:

a) definir o problema;
b) realizar um estudo da situação atual e verificar qual(is) a(s) forma(s) de resolver o problema;
c) terminada a fase de estudo, descrever o algoritmo que deverá, a princípio, resolver o problema;
d) analisar junto aos usuários se o problema será resolvido. Se a solução não foi encontrada, ou surgirem exceções a serem tratadas, deverá ser retomado para a fase de estudo para descobrir onde está a falha.

Um algoritmo deve possuir:
1. Que se tenha um número finito de passos.
2. Que cada passo esteja precisamente definido, sem possíveis ambiguidades.
3. Que existam variáveis e entradas de dados bem definidos.
4. Que existam uma ou mais saídas.
5. Que exista uma condição de fim sempre atingida para quaisquer entradas e

num tempo finito

A fim de entender como um algoritmo é construído, vamos analisar a construção de um algoritmo para o seguinte problema clássico, por ser um problema cotidiano e que não exige conhecimentos específicos: “trocar uma lâmpada queimada”.

1. Pegue uma escada;
2. Posicione-a embaixo da lâmpada;
3. Busque uma lâmpada nova;
4. Suba na escada;
5. Retire a lâmpada velha;
6. Coloque a lâmpada nova.

Se examinarmos esse algoritmo, veremos que ele permite solucionar o pro blema da troca de uma lâmpada queimada. Entretanto, se considerarmos as situações nas quais o algoritmo poderá ser aplicado, perceberemos que o mesmo não irá tratar a situação em que a lâmpada não esteja queimada. Na verdade, mesmo que a lâmpada esteja funcionando, esse algoritmo irá trocá-la por uma nova.
Para solucionar esse problema, é necessário alterar o algoritmo de modo que a lâmpada seja testada antes de efetuar a troca. Para isso, basta ligar o interruptor e verificar se a lâmpada está funcionando ou não. Intuitivamente, faríamos a seguinte alteração no algoritmo:

1. Pegue uma escada;
2. Posicione-a embaixo da lâmpada;
3. Busque uma lâmpada nova;
4. Ligue o interruptor;
5. Se a lâmpada não acender, então:
   a) suba na escada;
   b) retire a lâmpada velha;
   c) Coloque a lâmpada nova.
6. Ligue o interruptor;
7. Se a lâmpada não acender, então:
   a) pegue uma escada;
   b) posicione-a embaixo da lâmpada;
   c) busque uma lâmpada nova;
   d) suba na escada;
   e) retire a lâmpada velha;
   f) Coloque a lâmpada nova.

Apesar de parecer completo, ou seja, permitindo solucionar o problema da troca da lâmpada queimada, este algoritmo não leva em consideração a possibilidade de a nova lâmpada não funcionar. Pois, nesse caso, seria necessário repetir os passos “e” e “f”. Daí surge a seguinte questão: pode ser que a outra lâmpada também não

funcione, sendo necessário repetir mais uma vez esta sequência de passos. Então, quando devemos parar de repetir?
Para toda repetição em um algoritmo devemos estabelecer uma condição de parada, ou seja, um limite para a quantidade de vezes em que os passos deverão ser repetidos. Caso não seja estabelecido esta condição de parada ocorre o que chamamos de laço infinito - mais conhecido por sua designação em inglês loop infinito - no qual os passos se repetem indefinidamente.
No caso do nosso algoritmo, uma condição de parada adequada seria repetir enquanto a lâmpada não acender. Assim o algoritmo ficaria da seguinte forma:

7. Ligue o interruptor;
8. Se a lâmpada não acender, então:
   a) pegue uma escada;
   b) posicione-a embaixo da lâmpada;
   c) busque uma lâmpada nova;
   d) suba na escada;
   e) retire a lâmpada velha;
   f) coloque a lâmpada nova;
   g) enquanto a lâmpada não acender:
      – retire a lâmpada;
      – coloque outra lâmpada.

Atividades

1)Escreva um algoritmo para o seguinte cenário:
Você está em uma sala vai enviar uma mensagem de email para um contato. Na sala há um computador desktop e este está desligado.

2)Escreva um algoritmo para:
Você precisa enviar uma mensagem para um
cliente e possui apenas um dispositivo móvel (smartphone)
novo sem carga de bateria e sem aplicativos de
mensagens instalado.

3. Tipos de Dados

Todo o trabalho realizado por um computador é baseado na manipulação das informações contidas em sua memória. Estas informações podem ser classificadas em dois tipos:

- As instruções, usadas para determinar o funcionamento da máquina e a maneira como os dados devem ser tratados. As instruções são específicas para cada arquitetura de computador, pois estão diretamente relacionadas às características do hardware particular como, por exemplo, o conjunto de instruções de cada processador;
- Os dados propriamente ditos, que correspondem às informações que deverão ser processadas pelo computador.

Por isso, é necessário que haja formas de se trabalhar com diferentes tipos de dados em um programa.

3.1. Tipos Inteiros

São caracterizados como tipos inteiros, os dados numéricos positivos ou negativos, excluindo-se destes qualquer número fracionário. Como exemplo deste tipo de dado, tem-se os valores: 35, 0, -56, 1024 entre outros.

3.2. Tipos Reais

São caracterizados como tipos reais, aqueles que possuem parte decimal ou são números fracionários, e podem ser positivos, negativos ou zero.

Exemplos de dados do tipo real são 3.2 (real positivo), 0.00 (zero real) e -19.76 (real negativo).

3.3. Tipos Caracteres

São caracterizados como tipos caracteres, as sequências contendo letras, números e símbolos especiais. Uma sequência de caracteres deve ser indicada entre aspas (“”). Este tipo de dado também é conhecido como alfanumérico, string, literal ou cadeia. Como exemplo deste tipo de dado, tem-se os valores: “Programação”, “Rua Alfa, 52 Apto 1”, “Fone 574-9988”, “04387-030”, “ ”, “7” entre outros.

3.4. Tipos Lógicos

São caracterizados como tipos lógicos os dados com valor verdadeiro e falso, sendo que este tipo de dado poderá representar apenas um dos dois valores **Verdadeiro ou Falso.** Ele é chamado por alguns de tipo booleano, devido à contribuição do filósofo e matemático inglês George Boole na área da lógica matemática.

4. Operadores

Operadores são elementos funcionais que atuam sobre termos (também chamados de operandos) e produzem um determinado resultado.

Os operadores são, na prática, instruções especiais pelas quais incrementamos, decrementamos, comparamos e avaliamos dados dentro de um programa de computador. Podemos classificar os operadores em três classes:

- Operadores aritméticos;
- Operadores relacionais;
- Operadores lógicos.

4.1. Operadores Aritméticos

São utilizados para obter resultados numéricos. Além da adição, subtração, multiplicação e divisão.

Operador	Significado
+	Adição
-	Subtração
*	Multiplicação

/	Divisão

4.2. Operadores Relacionais

Operadores Relacionais (Comparativos): são utilizados para comparar dados em um programa. Os valores a serem comparados podem estar armazenados em constantes, variáveis, valores numéricos ou literais. São operadores binários que devolvem os valores lógicos: verdadeiro e falso. Os operadores relacionais são:

SÍMBOLO	SIGNIFICADO
=	Igual a
<>	Diferente de
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

Estes operadores são somente usados quando se deseja efetuar compara ções. Comparações só podem ser feitas entre objetos de mesma natureza, isto é, variáveis do mesmo tipo de dado.

O resultado de uma comparação é sempre um valor lógico. Por exemplo, digamos que a variável inteira escolha contenha o valor 7. A primeira das expressões a seguir fornece um valor falso, e a segunda um valor verdadeiro:

escolha <= 5

escolha > 5

4.3. Operadores Lógicos

Operadores Lógicos: Os operadores lógicos – também conhecidos como operadores booleanos – servem para combinar resultados de expressões relacionais, devolvendo como resultado final os valores: verdadeiro ou falso.

Esse tipo de operador é amplamente usado na composição de expressões lógicas que são muito utilizadas nas estruturas de decisão e repetição em um programa. Os operadores lógicos são:

E (do inglês **AND**) - uma expressão desse tipo é verdadeira se todas as con dições forem verdadeiras;

```
dia
hora
SE  dia=”segunda-feira” E hora>18:30
entao escreva  “temos aula”
SENAO
escreva “nao temos aula”
```

OU (do inglês **OR**) - uma expressão desse tipo é verdadeira se pelo menos uma das condições forem verdadeiras;

nome
telefone
email
endereço

SE nome <> “” E (telefone<>”” OU email<>””) E endereço<>””
escreva “Liberado”

SENAO
escreva “ Ha informacoes faltando”

NÃO (do inglês **NOT**) - uma expressão desse tipo inverte o valor da expressão ou condição, se verdadeira inverte para falsa e vice-versa.

4.4. Tabela Verdade

Os operadores lógicos e relacionais são elementos de fundamental importância na elaboração de um programa. Em todos os programas são utilizadas expressões relacionais e lógicas para a tomada de decisões e consequente desvio do fluxo do programa. O resultado de uma operação lógica vai depender dos valores dos termos submetidos. A seguir é apresentada uma tabela com os tipos de dados resultantes de cada operador.
Tabela verdade é um instrumento lógico que contém todos os valores lógicos de uma proposição composta. A construção de uma tabela verdade para uma proposição composta envolve os valores lógicos das proposições simples que a compõem e as operações lógicas entre essas proposições.

OPERADOR	TERMO 1	TERMO 2	RESULTADO
E	Verdadeiro	Verdadeiro	Verdadeiro
	Verdadeiro	Falso	Falso
	Falso	Verdadeiro	Falso
	Falso	Falso	Falso
OU	Verdadeiro	Verdadeiro	Verdadeiro
	Verdadeiro	Falso	Verdadeiro
	Falso	Verdadeiro	Verdadeiro
	Falso	Falso	Falso
NÃO	Verdadeiro	-	Falso
	Falso	-	Verdadeiro

Principais conectivos da tabela verdade
Os conectivos (ou operadores) lógicos são símbolos ou palavras associados a operações que conectam uma proposição simples com outra proposição simples para produzir uma proposição composta.

Há cinco principais conectivos, cujos operação, símbolo e significado estão indicados no quadro abaixo.

Operação	Símbolo	Significado
Negação	~	não
Conjunção	^	e
Disjunção	v	ou
Condicional	→	se... então
Bicondicional	↔	se e somente se

	Modo de ler
~p	não p
p ^ q	p e q
p □ q	p ou q
p → q	se p então q
p ↔ q	p se e somente se q

- **Tabela verdade da negação**

Dada uma proposição simples *p*, o valor lógico da proposição ~ *p* é o contrário do valor lógico de *p*. Assim, se *p* é verdadeira, ~ *p* é falsa; e se *p* é falsa, ~ *p* é verdadeira.

p	~ p
V	F
F	V

- **Tabela verdade da conjunção**

Dadas as proposições *p* e *q*, o valor lógico da proposição *p* ^ *q* é verdadeiro apenas quando ambas as proposições são verdadeiras.

p	q	p ^ q
V	V	V
V	F	F
F	V	F
F	F	F

- **Tabela verdade da disjunção**

Dadas as proposições p e q, o valor lógico da proposição $p \vee q$ é verdadeiro quando, pelo menos, uma das proposições é verdadeira.

p	q	p ∨ q
V	V	V
V	F	V
F	V	V
F	F	F

- **Tabela verdade da condicional**

Dadas as proposições p e q, o valor lógico da proposição $p \rightarrow q$ é falso quando p é verdadeiro e q é falso e é verdadeiro nos demais casos.

p	q	p → q
V	V	V
V	F	F
F	V	V
F	F	V

- **Tabela verdade da bicondicional**

Dadas as proposições p e q, o valor lógico da proposição $p \leftrightarrow q$ é verdadeiro apenas quando ambas as proposições são verdadeiras ou ambas são falsas.

p	q	p ↔ q
V	V	V
V	F	F
F	V	F
F	F	V

Exemplo: Construa a tabela verdade da proposição ~ (*p* ∧ *q*).

Vamos utilizar uma tabela verdade com quatro colunas: uma para a proposição *p*, uma para a proposição *q*, uma para a proposição *p* ∧ *q*, e a última para a proposição final, que é ~ (*p* ∧ *q*).

p	q	p ∧ q	~ (p ∧ q)

Por fim, a quarta coluna é a negação de cada valor lógico da terceira coluna.

p	q	p ∧ q	~ (p ∧ q)
V	V	V	F
V	F	F	V
F	V	F	V
F	F	F	V

4.5. Exercícios Tabela Verdade

1 - Construa a tabela verdade da proposição ~ (p ∧ ~ q).
2 - Construa a tabela verdade da proposição ~ p ∨ q → ~ q.

3- Alice estava estudando para o concurso público da prefeitura municipal de Itabira/MG e se deparou com uma questão que solicitava a montagem da tabela verdade de ~(pV~q). Para solucionar ela montou o seguinte quadro:
O número de V encontrado na coluna p v ~q multiplicado pelo número de F encontrado na última coluna é:

p	**q**	**~q**	**p v ~q**	**~(pv~q)**
F	V			
V	V			
V	F			
F	V			

A.1
B.6
C.9
D.4

5. Formas de representação de Algoritmos

Com o passar do tempo e de estudos dos algoritmos, foram desenvolvidas inúmeras formas de se representar um algoritmo de modo a facilitar o seu entendimento e, mais tarde, a sua tradução para uma linguagem de programação específica. Entre as formas de representação de algoritmos mais conhecidas, podemos citar:

- Descrição Narrativa.
- Fluxograma.
- Diagrama de Chapin
- Pseudocódigo.

5.1. Formas de representação de Algoritmos: Descrição Narrativa

Nessa forma de representação, os algoritmos são expressos diretamente em linguagem natural.

- Usar somente um verbo por frase;
- Imaginar que você está desenvolvendo um algoritmo para pessoas que não trabalham com informática;
- Usar frases curtas e simples;
- Ser objetivo;
- Procurar usar palavras que não tenham sentido dúbio.

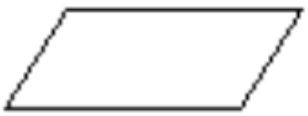
Exemplos:
Troca de um pneu furado:

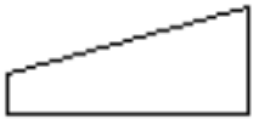
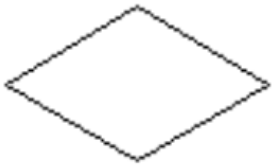
1. Afrouxar ligeiramente as porcas.
2. Suspender o carro.
3. Retirar as porcas e o pneu.
4. Colocar o pneu reserva.
5. Apertar as porcas.
6. Abaixar o carro.
7. Dar o aperto final nas porcas.

Cálculo da média de um aluno:
1. Obter as notas da primeira e da segunda prova.
2. Calcular a média aritmética entre as duas notas.
3. Se a média for maior ou igual a 7, o aluno foi aprovado, senão ele foi reprovado

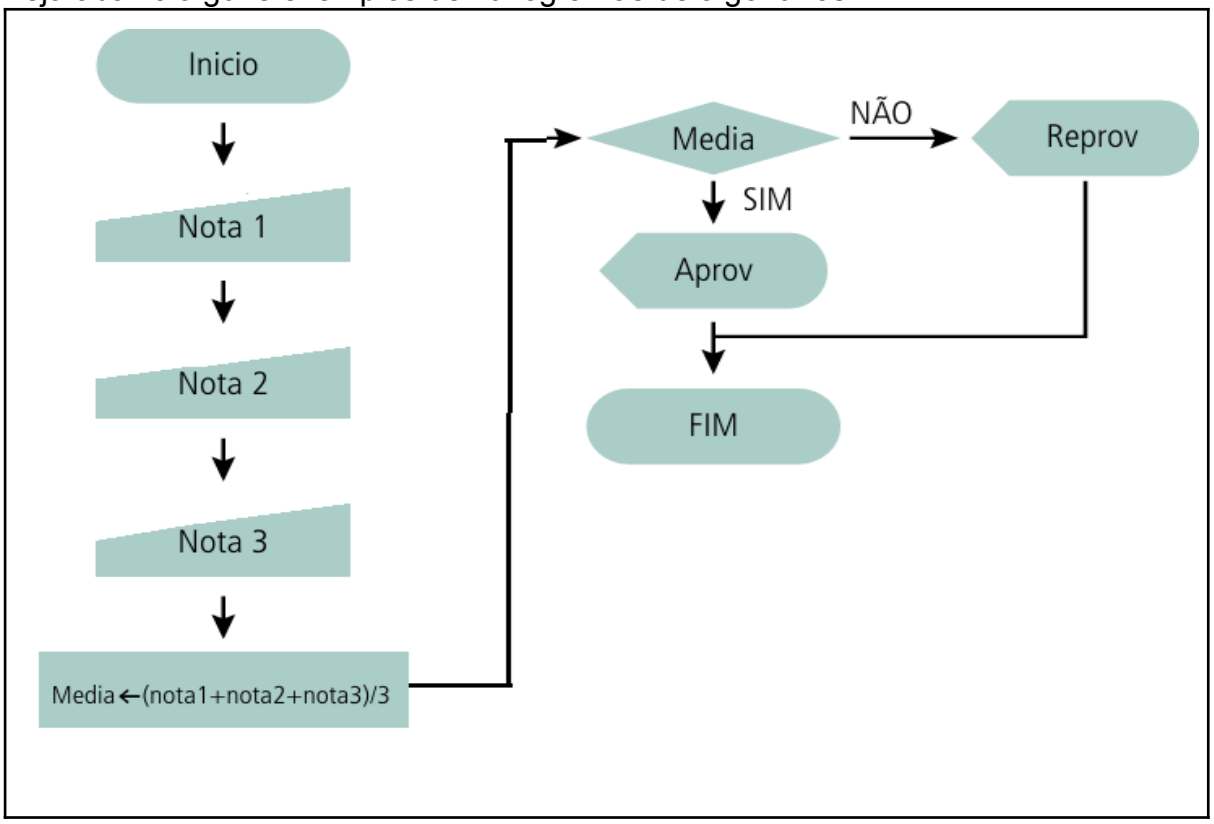
5.2. Formas de representação de Algoritmos: Fluxograma

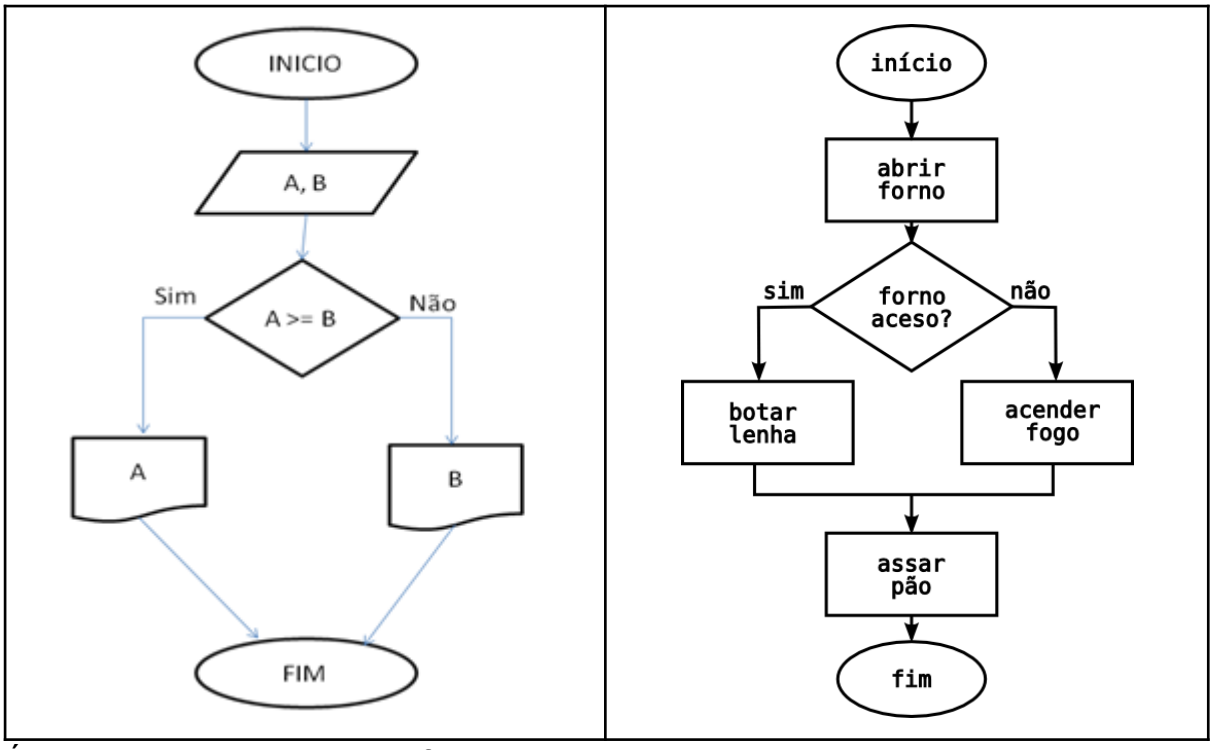
Se utiliza de figuras geométricas padronizadas para ilustrar os passos a serem seguidos para a resolução de problemas. E o ideal é que um algoritmo seja entendido da mesma forma por diferentes pessoas que o utilizarem. Para que um fluxograma seja interpretado corretamente, é necessário entender sua sintaxe e sua semântica.

Símbolo	Significado
	**TERMINAL** Indicador de Início e Fim do fluxograma
	**FLUXO** Indicado o sentido do fluxo de dados. Usado para conectar os demais componentes.
	**PROCESSAMENTO** Processar dados Exemplo: Fazer cálculo de valores
	**ENTRADA DE DADOS** De arquivos ou banco de dados

	**ENTRADA DE DADOS** Digitados pelo usuário Ex: Solicitar que o usuário digite o nome
	**ESTRUTURA DE DECISÃO** Utilizados para processos de decisão.
 	**SAÍDA DE DADOS** Impressão ou Tela por exemplo.

Veja abaixo alguns exemplos de fluxogramas de algoritmos.





É importante observar qual formato as empresas trabalham, quais os padrões de notação cada empresa adota.

Fatorial
Média Aritmética
Média Ponderada

Exercícios: Fazer algoritmos apresentando-os em fluxograma.

1. Crie um algoritmo que leia um número diferente de zero e diga se este número é positivo ou negativo.
2. Crie um algoritmo que receba 3 números e informe qual o maior entre eles.
3. Faça um algoritmo que leia a idade de uma pessoa expressa em anos, meses e dias e mostre-a expressa em dias.

Leve em consideração o ano com 365 dias e o mês com 30. (Ex: 3 anos, 2 meses e 15 dias = 1170 dias.)

4. Um usuário deseja um algoritmo onde possa escolher que tipo de média deseja calcular a partir de 3 notas. Faça um algoritmo que leia as notas, a opção escolhida pelo usuário e calcule a média.

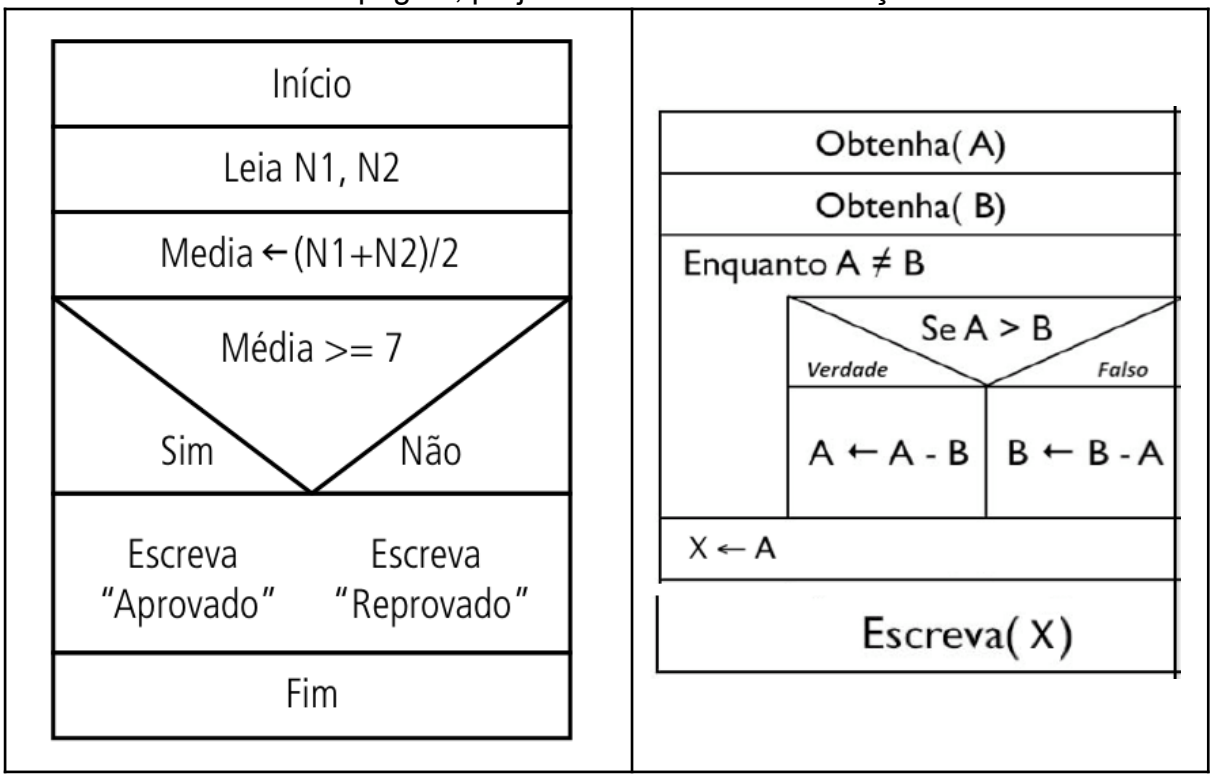
1 -aritmética
2 -ponderada (pesos: 3,3,4)

5. Escrever um algoritmo que lê um valor inteiro, calcula o fatorial desse número e mostra o resultado.

6. Crie um algoritmo que recebe 2 números e multiplica o num1 pelo num2 através de somas repetidas. (ex: 2 e 3 = 2 + 2 + 2)
7. Escreva um algoritmo para ler o número total de eleitores de um município, o número de votos brancos, nulos e válidos. Calcular e escrever o percentual que cada um representa em relação ao total de eleitores
8. Escrever um algoritmo que lê: - a percentagem do IPI a ser acrescido no valor das peças - o código da peça 1, valor unitário da peça 1, quantidade de peças 1 - o código da peça 2, valor unitário da peça 2, quantidade de peças 2 O algoritmo deve calcular o valor total a ser pago e apresentar o resultado. Fórmula : (valor1*quant1 + valor2*quant2)*(IPI/100 + 1)

5.3. Diagrama de Chapin

O diagrama de Chapin foi criado por Ned Chapin a partir de trabalhos de Nassi-Shneiderman, os quais resolveram substituir o fluxograma tradicional por um diagrama que apresenta uma visão hierárquica e estruturada da lógica do programa. A grande vantagem de usar este tipo de diagrama é a representação das estruturas que tem um ponto de entrada e um ponto de saída e são compostas pelas estruturas básicas de controle de sequência, seleção e repetição. Embora o diagrama de Chapin ofereça uma representação muito clara do algoritmo, à medida que os algoritmos vão se tornando mais complexos, fica difícil realizar os desenhos necessários numa única página, prejudicando a sua visualização.



5.4. Pseudocódigo

Esta forma de representação de algoritmos, também conhecida como português estruturado ou portugol, é bastante rica em detalhes e, por assemelhar- -se bastante à forma em que os programas são escritos, tem muita aceitação, sendo portanto a forma de representação de algoritmos mais utilizada.

O pseudocódigo é um código de escrita em que se utilizam termos convencionais para indicar as instruções do programa. Esses termos são geralmente uma mistura de palavras da nossa linguagem natural com palavras e notações típicas das linguagens de programação.

A utilização de pseudocódigo permite ao programador expressar as suas ideias sem ter de se preocupar com a sintaxe da linguagem de programação. Para isso, são utilizadas **primitivas (comandos genéricos)** que podem ser facilmente traduzidos para uma linguagem de programação específica. As primitivas possuem a mesma função dos símbolos em um fluxograma, ou seja, descrever as ações a serem executadas e garantir a correta interpretação do algoritmo.

Dica Importante: Alguns comandos funcionam como nas linguagens de programação, onde necessitam ser "iniciados e finalizados".

Exemplo:

Se …… Fim SE

Para…... Fim Para

If(sentença)**{**

….

}

PseudoCódigo

Estrutura do Algoritmo

A estrutura que empregaremos para a construção de nossos pseudocódigos será a seguinte:

algoritmo "nome" //Tem como objetivo identificar o algoritmo, deve-se utilizar um nome o mais significativo possível, para facilitar a identificação

var // Seção de Declarações - Neste ponto são informadas quais variáveis, e seus respectivos tipos, serão utilizadas no algoritmo

inicio // Seção de Comandos - Aqui será escrita a seqüência de // comandos que deve ser executada para solucionar o problema em questão

fimalgoritmo //marca o final do algoritmo

Observemos agora um pseudocódigo que recebe um valor inteiro, fornecido pelo usuário, e o retorna no monitor.

algoritmo "exemplo 1"

var

x: inteiro
y: inteiro
início
leia (x)
escreva (x) f
fimalgoritmo

ALGORITMO - primitiva usada para nomear o algoritmo representado

INÍCIO (do inglês **START**)- esta primitiva é usada para identificar o ponto inicial do algoritmo;

FIM (do inglês **END**) - identifica o ponto final do algoritmo;

LEIA , (do inglês **INPUT**) - primitiva usada para a leitura de dados do utilizador. Equivalente à entrada de dados manual do fluxograma;

ESCREVA , (do inglês **OUTPUT**) – primitiva usada para a apresentação de dados ao utilizador. Equivalente ao símbolo de exibição do fluxograma; +- - atribuição do resultado da expressão à variável indicada;

SE ENTÃO (do inglês **IF ... THEN**)

SENÃO (do inglês **ELSE**)

FIM SE (do inglês **END IF**) - primitiva de controlo de fluxo equivalente à decisão dos fluxogramas;

ESCOLHA (<variável» **(**do inglês **SWITCH)**
CASO <valor 1>: <instruções a executar>
CASO <valor 2>: <instruções a executar>
CASO <valor N>: <instruções a executar>

FIM ESCOLHA - primitiva de controlo de fluxo, usado para representar uma estrutura de controle de seleção múltipla;

ENQUANTO FAÇA (do inglês **WHILE**)) <instruções a executar
enquanto a condição for verdadeira>

FIM ENQUANTO (do inglês **END WHILE**) - primitiva de controlo de fluxo, sem equivalência direta em fluxogramas, que implementa um ciclo executado enquanto a condição referida seja verdadeira;

REPITA (do inglês **REPEAT**) <instruções a executar enquanto a condição for verdadeira>

ATÉ <condição> (do inglês **UNTIL**) - primitiva de controle de fluxo, sem

equivalência direta em fluxogramas, que implementa um ciclo executado até que a condição referida seja verdadeira;

PARA <condição inicial> **ATÉ** <condição final> **FAÇA** [**PASSO** <incremento>] (do inglês **FOR** ... **TO** ... [**STEP** ...] **DO**)
<instruções a executar enquanto a condição final for falsa>

FIM PARA (do inglês **END FOR**) - primitiva de controlo de fluxo, que executa as instruções nela contidas enquanto a condição final for falsa. O incremento pode ser omitido desde que seja unitário positivo;

VAR <nome da variável> [,]: <tipo da variável> (neste caso é usado uma redução do termo VARIÁVEL - do inglês **VARIABLE**) – primitiva utilizada na definição de variáveis. Os tipos de variáveis são discutidos em uma aula à frente;

CONST <nome da constante> **=** <valor da constante> [,] (neste caso é usado uma redução do termo CONSTANTE - do inglês **CONSTANT** - primitiva utilizada na definição de constantes. As constantes são discutidas à frente;

ESTRUTURA <nome da variável>: <tipo da variável> [,] (do inglês **STRUCT**) [<nome da variável>: <tipo da variável>]

FIM ESTRUTURA (do inglês **END STRUCT**) - primitiva utilizada na definição de estruturas de variáveis. Os tipos de variáveis são definidos à frente;

FIM FUNCAO (do inglês **END FUNCTION**) - primitiva utilizada na definição de funções. Por parâmetros entende-se uma lista dentro da função. Em certas situações, como veremos mais à frente, os parâmetros podem ser utilizados para a função exportar valores;

18/03/2024
Operações Matemáticas

- Calcular a média
- Calcular Desconto
- Calcular Acréscimo
- Calcular Markup
- Calcular Margem de Lucro
- Calcular Fatorial

Exemplo de pseudocódigo

```
Algoritmo Media
        VAR Nl, N2, N3, Media: real
INÍCIO
        Escreva “Digite Nota 1:”
        LEIA Nl
        Escreva “Digite Nota 2:”
        LEIA N2
        Escreva “Digite Nota 3:”
        LEIA N3
                Media (NI + N2 + N3) / 3
                SE Media >= 7 ENTAO
                        ESCREVA” Aprovado”
                SENAO
                        ESCREVA “Reprovado”
                FIM SE
FIM
```

Uma grande vantagem da utilização de pseudocódigo é o fato de permitir ao programador expressar as suas ideias sem ter de se preocupar com a sintaxe da Linguagem de Programação. Por outro lado, esse tipo de representação é o que mais se aproxima da codificação final em uma Linguagem de Programação e, por isso, é a mais utilizada na prática.

Cláusula Enquanto Faça:

1. Algoritmo que imprime os números de 1 a 10:

```
 inteiro numero = 1
enquanto (numero <= 10) faça
   Escreva (numero)
   numero = numero + 1
fim enquanto
```

2. Algoritmo que calcula a soma dos números de 1 a 100:

```
inteiro soma = 0
inteiro numero = 1
enquanto (numero <= 100) faça
   soma = soma + numero
   numero = numero + 1
fim enquanto
Escreva "A soma dos números de 1 a 100 é: " + soma
```

3. Algoritmo que calcula o fatorial de um número:

```
inteiro numero = 5
inteiro fatorial = 1
```

```
inteiro contador = 1
enquanto (contador <= numero) faça
   fatorial = fatorial * contador
   contador = contador + 1
fim enquanto
Escreva ("O fatorial de " + numero + " é: " + fatorial)
```

Exemplo de Clausula Escolha

```
algoritmo "CalculadoraBasicaComSE"
var
  numero1 : REAL
  numero2 : REAL
  operacao : CARACTERE
  resultado : REAL
inicio
    ESCREVA ("Digite o primeiro número: ")
    LEIA (numero1)
    ESCREVA ("Digite a operação: ")
    LEIA (operacao)
    ESCREVA ("Digite o segundo número: ")
    LEIA (numero2)
        ESCOLHA operacao
                CASO "+"
                resultado := numero1 + numero2
                CASO "-"
                resultado := numero1 - numero2
                CASO "*"
                resultado := numero1 * numero2
                CASO "/"
                resultado := numero1 / numero2
        FIMESCOLHA
   ESCREVA ("Resultado: ", resultado)
fim
```

5.5. Atividades

1 - Faça um algoritmo que leia os valores de A, B, C e em seguida imprima na tela a soma entre A e B e mostre se a soma é menor que C.

```
Algoritmo Exercício_1
real A = 0.00;
real B = 0.00;
real C = 0.00;
real soma = 0.00;
        Inicio
                #Passo 1
```

```
        Escreva “Digite o valor de A:”
        Leia A
        Escreva “Digite o valor de B:”
        Leia B
        Escreva “Digite o valor de C:”
        Leia C

        # Passo 2
        soma <- A + B
        Escreva soma
        # Passo 3
                SE soma < C ENTAO
                Escreva "A soma de A e B é menor que C."
                SENAO
                Escreva "A soma de A e B é maior ou igual a C."
                FIMSE
    Fim
```

2 - Faça um algoritmo que leia dois valores inteiros A e B, se os valores de A e B forem iguais, deverá somar os dois valores, Caso contrário deverá multiplicar A por B. Ao final de qualquer um dos cálculos deve-se atribuir o resultado a uma variável C e imprimir seu valor na tela.

4 - Faça um algoritmo que receba um número inteiro e imprima na tela o seu antecessor e o seu sucessor.

5 - Faça um algoritmo que leia o valor do salário mínimo e o valor do salário de um usuário, calcule quantos salários mínimos o usuário ganha e imprima na tela o resultado.

6 - Faça um algoritmo que leia um valor qualquer e imprima na tela com um reajuste de +5%.

7 - Faça um algoritmo que leia dois valores booleanos (lógicos) e determine se ambos são VERDADEIRO ou FALSO.

8 - Faça um algoritmo que leia três valores inteiros diferentes e imprima na tela os valores em ordem decrescente.

9 - Faça um algoritmo que calcule o IMC (Índice de Massa Corporal) de uma pessoa, leia o seu peso e sua altura e imprima na tela sua condição de acordo com a tabela abaixo:
Fórmula do IMC = peso / (altura) ²
Tabela Condições IMC

Abaixo de 18,5 | Abaixo do peso

Entre 18,6 e 24,9 | Peso ideal (parabéns)
Entre 25,0 e 29,9 | Levemente acima do peso
Entre 30,0 e 34,9 | Obesidade grau I
Entre 35,0 e 39,9 | Obesidade grau II (severa)
Maior ou igual a 40 | Obesidade grau III (mórbida)

10 - Faça um algoritmo que leia três notas obtidas por um aluno, e imprima na tela a média das notas.

11 - Faça um algoritmo que leia quatro notas obtidas por um aluno, calcule a média das nota obtidas, imprima na tela o nome do aluno e
se o aluno foi aprovado ou reprovado. Para o aluno ser considerado aprovado sua média final deve ser maior ou igual a 7.

12 - Faça um algoritmo que leia o valor de um produto e determine o valor que deve ser pago, conforme a escolha da forma de pagamento
pelo comprador e imprima na tela o valor final do produto a ser pago. Utilize os códigos da tabela de condições de pagamento para efetuar o cálculo adequado.

Tabela de Código de Condições de Pagamento

1 - À Vista em Dinheiro ou Pix, recebe 15% de desconto
2 - À Vista no cartão de crédito, recebe 10% de desconto
3 - Parcelado no cartão em duas vezes, preço normal do produto sem juros
4 - Parcelado no cartão em três vezes ou mais, preço normal do produto mais juros de 10%

13 - Faça um algoritmo que leia o nome e a idade de uma pessoa e imprima na tela o nome da pessoa e se ela é maior ou menor de idade.

14 - Faça um algoritmo que receba um valor A e B, e troque o valor de A por B e o valor de B por A e imprima na tela os valores.

15 - Faça um algoritmo que leia o dia, o mês e o ano em que uma pessoa nasceu. imprima na tela quantos anos, meses e dias essa pessoa já viveu. Leve em consideração o ano com 365 dias e o mês com 30 dias.
(Ex: 5 anos, 2 meses e 15 dias de vida)

16 - Faça um algoritmo que leia três valores que representam os três lados de um triângulo e verifique se são válidos, determine se o triângulo é
equilátero, isósceles ou escaleno.

17 - Faça um algoritmo que leia uma temperatura em Fahrenheit e calcule a temperatura correspondente em grau Celsius. Imprima na tela as duas temperaturas.
Fórmula: C = (5 * (F-32) / 9)

18 - Francisco tem 1,50m e cresce 2 centímetros por ano, enquanto Sara tem 1,10m e cresce 3 centímetros por ano. Faça um algoritmo que calcule e imprima na tela em quantos anos serão necessários para que Sara seja maior que Francisco.

19 - Faça um algoritmo que imprima na tela a tabuada de 1 até 10.

20 - Faça um algoritmo que receba um valor inteiro e imprima na tela a sua tabuada.

21 - Faça um algoritmo que mostre um valor aleatório entre 0 e 100.

22 - Faça um algoritmo que leia dois valores inteiros A e B, imprima na tela o quociente e o resto da divisão inteira entre eles.

21 - Faça um algoritmo que efetue o cálculo do salário líquido de um professor. As informações fornecidas serão: valor da hora aula, número de aulas lecionadas no mês e percentual de desconto do INSS. Imprima na tela o salário líquido final.

22 - Faça um algoritmo que calcule a quantidade de litros de combustível gastos em uma viagem, sabendo que o carro faz 12 km com um litro. Deve-se fornecer ao usuário o tempo que será gasto na viagem a sua velocidade média, distância percorrida e a quantidade de litros utilizados para fazer a viagem.
Fórmula: distância = tempo x velocidade.
litros usados = distância / 12.

23. Escreva um algoritmo que recebe um número inteiro e verifica se ele é positivo.
24. Escreva um algoritmo que recebe dois números inteiros e verifica se o primeiro é maior que o segundo.
25. Escreva um algoritmo que recebe a idade de uma pessoa e verifica se ela é maior de idade (idade >= 18).
26. Escreva um algoritmo que recebe um número inteiro e verifica se ele é par.
27. Escreva um algoritmo que recebe a nota de um aluno e verifica se ele foi aprovado (nota >= 6.0).
28. Escreva um algoritmo que recebe um número inteiro e verifica se ele é um múltiplo de 5.
29. Escreva um algoritmo que recebe o sexo de uma pessoa ('M' para masculino e 'F' para feminino) e verifica se é masculino.
30. Escreva um algoritmo que recebe a idade de uma pessoa e verifica em qual faixa etária ela se encaixa (criança, adolescente, adulto ou idoso).

6. REFERÊNCIAS

RIZZO, Maria Luiza Alves. "Tabela verdade"; *Brasil Escola*. Disponível em: https://brasilescola.uol.com.br/matematica/tabela-verdade.htm. Acesso em 25 de fevereiro de 2024.

RESPOSTAS EXERCICIOS:

```
Algoritmo Exercício_1
real A = 0.00;
real B = 0.00;
real C = 0.00;
real soma = 0.00;

Inicio
#Passo 1
Escreva "Digite o valor de A:"
Leia A
Escreva "Digite o valor de B:"
Leia B
Escreva "Digite o valor de C:"
Leia C

# Passo 2
soma <- A + B
Escreva soma
# Passo 3
SE soma < C ENTAO
    Escreva "A soma de A e B é menor que C."
SENAO
    Escreva "A soma de A e B é maior ou igual a C."
FIMSE

Fim


Algoritmo Exercício_2
inteiro A = 0.00;
inteiro B = 0.00;
inteiro C = 0.00;

Início
Escreva "Digite o valor de A:"
Leia A
Escreva "Digite o valor de B:"
Leia B
SE A = B ENTÃO
      C <- A + B
SENAO
      C <- A * B
FIMSE
Escreva C

Fim


Algoritmo Exercício_3
int A = 0

Inicio
#Passo 1
Escreva "Digite um numero:"
Leia A
```

```
Escreva "Numero Digitado:", A
Escreva "Antecessor do Número Digitado:", A-1
Escreva "Sucessor do Numero Digitado:", A+1


Fim
```

```
Algoritmo Exercício_8

Var
real A=0
real B=0
real C=0
real D=0

Inicio
Escreva "Digite o valor A:"
Leia A
Escreva "Digite o valor B:"
Leia B
Escreva "Digite o valor C:"
Leia C

SE A < B ENTÃO
      D <- A
      A <- B
      B <- D
      D <- 0
FIMSE
SE A < C ENTÃO
      D <- A
      A <- C
      C <- D
      D <- 0
FIMSE
SE B < C ENTÃO
      D <- B
      B <- C
      C <- D
      D <- 0
FIMSE

Escreva A
Escreva B
Escreva C
FIM
```

1. Construa a tabela verdade da proposição p ~ p

p	**~p**
V	**F**
F	**V**
V	**F**
V	**F**

2. Alice estava estudando para o concurso público da prefeitura municipal de Itabira/MG e se deparou com uma questão que solicitava a montagem da tabela verdade de ~(pV~q). Para solucionar ela montou o seguinte quadro: (onde: ~ representa não, V representa ou)

p	**q**	**~q**	**p v ~q**	**~(pv~q)**
F	V	F	F	**V**
V	V	F	V	**F**
V	F	V	V	**F**
F	V	F	F	**V**

O número de V encontrado na coluna p v ~q multiplicado pelo número de F encontrado na última coluna é:

A.4 B.9 **C.2** D.6

3. Desenvolva um algoritmo que mostra a expressão: "*CEDUP - Centro de Educação Profissional José Buss - CEDUP*"

```
Algoritmo tres
início
        Escreva "CEDUP - Centro de Educação Profissional José Buss - CEDUP"
fim
```

4. Desenvolva um algoritmo que leia o nome de uma pessoa, e em seguida mostre o nome lido

```
Algoritmo quatro
var
        nome: literal
início
        Escreva "Digite um nome"
        Leia nome
        Escreva "Nome digitado:",nome
fim
```

5. Desenvolva um algoritmo que leia as variáveis A e B, e no final mostra a soma de ambas

```
Algoritmo cinco
var
        A, B,Soma: real
```

```
início
        Escreva "Digite o valor de A"
        Leia A
        Escreva "Digite o valor de B"
        Leia B
        Soma <- A+B
        Escreva "Resultado:",Soma
fim
```

6. Desenvolva um algoritmo que leia 3 notas de alunos e ao final mostre a média das notas.

```
Algoritmo seis
var
        A, B,C,Soma,Media: real
início
        Escreva "Digite a Nota A"
        Leia A
        Escreva "Digite a nota  B"
        Leia B
        Escreva "Digite a nota c"
        Leia c
        Soma <- A+B+C
        Media <- Soma/3
        Escreva "A média é:",Media
fim
```

7. Desenvolva um algoritmo que leia 2 valores reais. Mostre o triplo do primeiro, e mostre também a soma da metade do segundo valor com o triplo do primeiro.

```
Algoritmo sete
var
        A, B,Soma,Triplo,metade: real
início
        Escreva "Digite a Nota A"
        Leia A
        Escreva "Digite a nota  B"
        Leia B
        Triplo<-A*3
        metade<-B/2
        Soma<-Triplo+metade
        Escreva "Triplo de A:", Triplo
        Escreva "A soma resultante é:",Soma
Fim
```

8. Escreva um algoritmo que recebe um número inteiro e verifica se ele é positivo.

```
Algoritmo oito
var
        A: inteiro
início
        Escreva "Digite um numero:"
        Leia A
        SE A>=0 entao
                Escreva " O numero ",A,"é positivo"
        SENAO
                Escreva " O numero ",A,"é negativo"
```

```
        FIMSE
FIM
```

9. Faça um algoritmo que receba um valor inteiro e imprima na tela a sua tabuada.

```
Algoritmo nome
var
        A, contador, calculo: inteiro
início
        contador<-1
        Escreva "Digite um numero:"
        Leia A
        ENQUANTO contador<=10 ENTAO
                calculo<- A * contador
                Escreva A ," x ", contador, " = ", calculo
        FIMENQUANTO
FIM
```

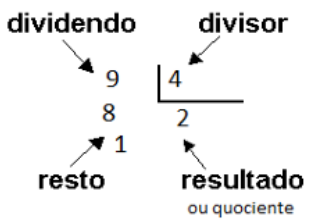
10. Faça um algoritmo que leia o nome e a idade de uma pessoa e imprima na tela o nome da pessoa e se ela é maior ou menor de idade.

```
Algoritmo dez
var
        idade: inteiro
        nome: literal
inicio
        Escreva "Digite seu nome:"
        Leia nome
        Escreva "Digite Idade"
        Leia idade

        SE idade>=18 ENTAO
                Escreva nome, " é maior de idade"
        SENAO
                Escreva nome, " é menor de idade"
        FIMSE
Fim
```

Como Verificar se um número é par ou impar

Resto da Divisão



O símbolo utilizado para a operação de resto é o símbolo de % (percentual).

```
Algoritmo Exemplo
 numero: inteiro
resto: inteiro
Inicio
```

```
        escreva "Digite um numero: "
        leia numero
        resto <- numero % 2
                Se resto = 0 Entao
                escreva numero, " é par"
                senao
                escreva numero, " é impar"
                FimSe
Fim
```

Array / Vetores

Os arrays são estruturas que servem para guardar dados, e organizá-los. Seu objetivo é ser um espaço fixo na memória do computador que armazena elementos. Esses elementos podem ser acessados por um tipo de indicação, que chamamos de índice.
Exemplos:

Faça um algoritmo que leia o nome de 5 pessoas

Algoritmo Array1

```
nTemp: literal
listaNome[5]: literal
contador: inteiro
Inicio
        contador<-0
        Enquanto contador<5 entao
                Escreva “Digite um nome:”
                Leia nTemp
                listaNome[contador]<-nTemp
                contador<-contador+1
        Fim Enquanto
Fim
```

Faça um algoritmo que leia 3 notas e mostre a media

Algoritmo Array2

```
NotaTemp: real
nMedia: real
nNotas[3]: real
cont: inteiro
        Inicio
        cont<-0
                Enquanto cont<3 Entao
                Escreva “Digite a nota:”
                Leia NotaTemp
```

```
            nNotas[cont]<-NotaTemp
            cont<-cont+1
            FimEnquanto
        cont<-0
            Enquanto cont<3 Entao
            nSoma<nSoma+nNotas[cont]
            cont<-cont+1
            FimEnquanto
        nMedia<-nSoma/3
        Escreva "A Media é:", nMedia
        Fim
```

Faça um algoritmo que leia 50 idades e mostre quantas idades são maiores que 40 anos.

Algoritmo Array3

```
TempIdade: inteiro
cont: inteiro
ListaIdades[50]: inteiro
qtd40: inteiro
        Inicio
            cont<-0
                Enquanto cont<50 então
                Escreva "Digite uma idade:"
                Leia TempIdade
                ListaIdade[cont]<-TempIdade
                cont<-cont+1
                FimEnquanto
            cont<-0
                Enquanto cont<50 então
                TempIdade<-ListaIdade[cont]
                    Se TempIdade>40 entao
                    qtd40<- qtd40+1
                    FimSe
                cont<-cont+1
                FimEnquanto
        Escreva "A quantidade idades maior que 40 anos e´:", qtd40
        Fim
```

Faça um algoritmo que leia 20 numeros reais, e em seguida mostre em ordem inversa os valores digitados.

- Mostre o somatório dos valores digitados
- Mostre a média dos valores digitados
- Mostre o Triplo do somatorio dos Valores

Algoritmo Array4

```
nTemp: real
nSomatorio:real
nMedia: real
nTriplo: real
cont: inteiro
ListaVlr[20]: real
Inicio
cont<-0
        Enquanto cont<20 entao
        Escreva “Digite um valor:”
        Leia nTemp
                ListaVlr[cont]<-nTemp
        cont<-cont+1
        FimEnquanto
Cont<-0
        Enquanto cont<20 entao
        Escreva ListaVlr[cont]
nSomatorio<-nSomatorio+ListaVlr[cont]

        cont<-cont+1
FimEnquanto

nMedia<-nSomatorio/20
nTriplo<-nSomatorio*3
Escreva nSomatorio
Escreva nMedia
Escreva nTriplo
Fim
```

Exercícios

1. Crie um algoritmo para ler 10 números inteiros e mostrar os números pares deste vetor;
2. Crie um algoritmo para ler 15 números inteiros e mostrar no final, os que forem maiores ou igual a 10;
3. Faça um algoritmo que leia 20 números e armazene em um vetor. Depois, some os 10 primeiros elementos deste vetor;

4. Faça um algoritmo para ler um vetor com 10 elementos e inverter a posição destes elementos, de tal modo que o primeiro elemento venha a ser o último depois da inversão;
5. Faça um algoritmo que leia 30 valores do tipo inteiro e armazene-os em um vetor. A seguir, o algoritmo deverá informar
    a. todos os números pares que existem no vetor;
    b. o menor e o maior valor existente no vetor;
    c. quantos dos valores do vetor são maiores que a média desses valores:
6. Faça um algoritmo que leia 10 valores numéricos inteiros em um vetor e três valores inteiros nas variáveis A B e C. Após a leitura, informe o número de vezes que os números A, B e C aparecem no vetor.
7. Dados dois vetores de tamanho N, faça uma função que diga se os mesmos possuem conteúdo igual.

8. Desenvolva um algoritmo que leia 10 nomes de pessoas e ao final mostre uma única vez todos um abaixo do outro
9. Desenvolva um algoritmo que leia 10 números reais e ao final mostre a soma de todos
10. Desenvolva um algoritmo que leia 5 nomes de cidade e mostre-as em ordem inversa a que foram digitadas.

Menus em Algoritmos

Faça um algoritmo onde o usuário escolha quais operações ele quer realizar: (Adicao, Subtracao, Multiplicacao, Divisao). E em seguida peça ao usuário dois valores a serem Calculados, o algoritmo deve mostrar o resultado para o usuário.

Classe Java

```
package aula2705;
import java.awt.Desktop;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.ArrayList;
import java.util.Date;
import javax.swing.*;
public class inicio {
	public static void main(String[] args) {
		String funcao="";
		while(!"sair".equals(funcao)) {//enquanto
			funcao=JOptionPane.showInputDialog("Selecione a opcao:\n"
					+ "1 - Conta Corrente  da Rozita\n"
					+ "2 - Captura Peso\n"
					+ "3 - Array\n"
					+ "sair - para finalizar");
			if("1".equals(funcao)) {//se
				calculaContaBanco();
			}//fimSe
			if("2".equals(funcao)) {//se
				calculaPeso();
			}//fimSe

			if("3".equals(funcao)) {//se
				controlaArray();
			}//fimSe

		}//fimEnquanto*/

	}

	public static void controlaArray() {
		String funcao="";
		ArrayList lista=new ArrayList();
		while(!"sair".equals(funcao)) {//enquanto
			funcao=JOptionPane.showInputDialog(null,"digite nome");
			lista.add(funcao);
		}//fimEnquanto*/

		int contador=lista.size()-1;
		while(contador>=0) {
			String temp=(String)lista.get(contador);
			System.out.println(temp);
			contador=contador-1;
		}
```

```
	}

	public static void calculaContaBanco() {
		double saldo=0.00;
		String valor="";
		String operacao="";

		while(!"sair".equals(operacao.toLowerCase())) {//enquanto
		operacao=JOptionPane.showInputDialog("Saldo
Atual:"+saldo+"\nDigite:\n "
				+ "+ para Depósito\n"
				+ "- para Saque\n"
				+ "Sair para finalizar\n"
				+ " by Fulano");
		valor=JOptionPane.showInputDialog("Digite o valor de movimento:");
		if("+".equals(operacao)) {
		saldo=saldo+Double.parseDouble(valor);
		}
		if("-".equals(operacao)) {
			saldo=saldo+Double.parseDouble(valor);
		}

		}//fimEnquanto
		JOptionPane.showMessageDialog(null,"Saldo Final:"+saldo);
		}

	public static void calculaPeso() {
	double variavel=0.00;
	double pesoTotal=0.0;
	while (variavel != 9999) {
		String  recbido=JOptionPane.showInputDialog("Digite o peso do Suino
na Balança"
				+ " \n ou 9999 para Encerrar:");
		recbido=recbido.replace(",", ".");
		System.out.println("vlr. Lido:"+recbido);
		variavel= Double.parseDouble(recbido);
		if(variavel != 9999) {
			pesoTotal=pesoTotal+variavel;
		}
	}
	JOptionPane.showMessageDialog(null,"Peso Total da Carga: "+pesoTotal);
	}

	public static void abrePagina() {
		String   url=  JOptionPane.showInputDialog("Digite o endereco de um
Site:");;
		 URI link = null;
```

```
                try {
                        link = new URI(url);
                } catch (URISyntaxException e) {
                        e.printStackTrace();
                }
    try {
                        Desktop.getDesktop().browse(link);
                } catch (IOException e) {
                        e.printStackTrace();
                }

        }



        public static void calculadora() {
                int a=0;
                int b=0;
                String nomOperacao="";
                int result=0;

                a = Integer.parseInt(JOptionPane.showInputDialog("Digite valor A:"));
                b = Integer.parseInt(JOptionPane.showInputDialog("Digite valor B:"));
                nomOperacao = JOptionPane.showInputDialog("Selecione a operação:"
                                + "\n + para adicao"
                                + "\n - para subtracao "
                                + "\n * para multiplicacao "
                                + "\n / para divisao");

                if("+".equals(nomOperacao)) {
                        result=a+b;
                }
                if("-".equals(nomOperacao)) {
                        result=a-b;
                }
                if("*".equals(nomOperacao)) {
        result=a*b;
                }
                if("/".equals(nomOperacao)) {
        result=a/b;
                }

                JOptionPane.showMessageDialog(null,"Resultado: "+result);

        }



        public static void primeiraAula() {
```

```
                JOptionPane.showMessageDialog(null,"CEDUP - Técnico em Informática 2024\n"
                                + " Lógica de Programação\n Nós somos capazes, Vamos dominar o mundo!!!");

/*
*       //      JOptionPane.showMessageDialog(null,"Aula 27/05/2024");
                String  recebido = "";
                //recebido = JOptionPane.showInputDialog("Digite Alguma coisa:");

                  //JOptionPane.showMessageDialog(null,"Top!!!  Você  digitou:  " + recebido+"");

                //  JOptionPane.showMessageDialog(null,"Top!!!  Você  digitou:  " + recebido+"", "Tela1", JOptionPane.INFORMATION_MESSAGE);
        /*
                JOptionPane.showMessageDialog(null,"Top!!!  Você  digitou:  " + recebido+"", "Tela2",
          JOptionPane.ERROR_MESSAGE);

                JOptionPane.showMessageDialog(null,"Top!!!  Você  digitou:  " + recebido+"", "Tela3",
          JOptionPane.WARNING_MESSAGE);

                JOptionPane.showMessageDialog(null,"Top!!!  Você  digitou:  " + recebido+"", "Tela4",
          JOptionPane.CANCEL_OPTION);

                JOptionPane.showMessageDialog(null,"Top!!!  Você  digitou:  " + recebido+"", "tela5",
          JOptionPane.OK_CANCEL_OPTION);

                JOptionPane.showMessageDialog(null,"Top!!!  Você  digitou:  " + recebido+"", "Tela6",
          JOptionPane.NO_OPTION);

                JOptionPane.showMessageDialog(null,"Top!!!  Você  digitou:  " + recebido+"", "Tela7",
         JOptionPane.PLAIN_MESSAGE);


                */

                String  vlrRecebidoTela1=  JOptionPane.showInputDialog("Digite um Valor Inteiro(int)");
                int numeroRecebido=Integer.parseInt(vlrRecebidoTela1);

                int contador = 1;
```

```
                while (contador < numeroRecebido) {
                        System.out.print("Contador: " + contador + "-> ");
                        Date mmm = new Date();
                        JOptionPane.showMessageDialog(null,"Contador: "+ contador);

                        System.out.println(mmm);
                        contador = contador + 1;
                }


        /*      String  vlrRecebidoTela2=  JOptionPane.showInputDialog("Digite  um
Valor Real(double ex.: 0.00)");
                double numeroRecebido2=Double.parseDouble(vlrRecebidoTela2);



                /* String  url= JOptionPane.showInputDialog("Digite o endereco de um
Site:");;
                  URI link = null;
                try {
                        link = new URI(url);
                } catch (URISyntaxException e) {
                        e.printStackTrace();
                }
      try {
                        Desktop.getDesktop().browse(link);
                } catch (IOException e) {
                        e.printStackTrace();
                }

                /*
                int a = 2;
                int b = 8;
                int c = 0;
                c = a + b;
                if (c >= 10) {
                        System.out.println("Sucesso");
                } else {
                        System.out.println("Tem que estudar mais");
                }
                */


        }
        }
```

Atividades 10/06/2024 Página 40

1 - Desenvolva um algoritmo que solicite ao usuário para digitar 10 nomes. Em seguida, o algoritmo deve mostrar todos os nomes na ordem inversa ao que foram digitados.

2 - Desenvolva um algoritmo que solicite ao usuário para digitar alturas de pessoas. Ao sair deve mostrar a média das alturas digitadas.

3 - Desenvolva um algoritmo que guarde uma lista de perguntas. Em seguida questione mostre as perguntas e para que o usuário digite as respostas. (fazer um menu, onde o usuário selecione se quer cadastrar perguntas ou responder).

Mover Tela

```
public static void mostratela() {
                JFrame frame = new JFrame("Teste: Frame");
                 frame.setSize(200, 200);
                 frame.setVisible(true);
                int contax=0;
                int contay=0;

                while(contax<1000) {
                        frame.setLocation(contax, contay);
                        contax=contax+40;
                        contay=contay+50;
                        if(contax>400) {
                                //contax=0;
                                contay=0;
                        }
                        try {
                                TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }

        }
```

Atividades 01/07/2024 Página 42

1 - Desenvolva um algoritmo que solicite ao usuário que digite os dados de veiculos: Placa, cor, modelo, ano fabricação, ano modelo. Criar uma classe veículo e popular os dados dentro de um array.
(Objetivo: deixar a estrutura para salvar em banco de dados em seguida).

03/07/2024

```
public static ArrayList buscarVeiculos() {
      ArrayList vetor = new ArrayList();
      try {
            String driver = "org.postgresql.Driver";
             String senha = "postgres";
             String user = "postgres";
             String url = "jdbc:postgresql://localhost:5433/aula0307";
          String ler = "select *  from veiculos  order by id asc   "
                    ;
          Class.forName(driver);
          Connection MyConn = DriverManager.getConnection(url, user, senha);
          Statement MyStm = MyConn.createStatement();
         // PreparedStatement inserir = MyConn.prepareStatement(ler);
          ResultSet MyRSu = MyStm.executeQuery(ler);
          while (MyRSu.next()) {
             veiculo veic=new veiculo();
             veic.setId(MyRSu.getInt("id"));
            // veic.setNomedomodelo(MyRSu.getString("nom_modelo"));
                    String              temp="ID:"+MyRSu.getInt("id")+"              -              "+
MyRSu.getString("nom_modelo")+
                    "\n";
             System.out.println(temp);
             JOptionPane.showInternalMessageDialog(null, temp);
             //veiculo veic =new veiculo();
             //veic.setNumerodaplaca();
                    //veic.setNomedomodelo();
                    //veic.setCordoveiculo();
                    //veic.setAnodefabricacao();
             //vetor.add(veic);
          }
//           inserir.close();
          MyStm.close();
          MyConn.close();
      } catch (Exception e) {
        System.out.println(e);
```

```
    }
    return vetor;
  }
```

Método Gravar no banco de dados

```
GravanoBanco("UPDATE","FERRARI","4" );
//GravanoBanco("INSERIR","",0 );


public static boolean GravanoBanco(String acao, String nome, String id) {
            try {
                  String driver = "org.postgresql.Driver";
                   String senha = "postgres";
                   String user = "postgres";
                   String url = "jdbc:postgresql://localhost:5433/aula0307";
        String sql="";
                   if("INSERIR".equals(acao)) {
                         sql = " insert into veiculos(id, nom_modelo,num_placa) "
                                     + " values((select max(id)+1 from veiculos),'CARRO
'||now(),'XXX-1234');" ;

                    }
                  if("UPDATE".equals(acao)) {
                         sql  =  "  update  veiculos  set  nom_modelo='"+nome+"'  where
id="+id ;
                     }
               System.out.print("######################\n"+sql
                         +"\n############################");


                   Class.forName(driver);
                  Connection MyConn = DriverManager.getConnection(url, user, senha);
                  PreparedStatement inserirCadsp = MyConn.prepareStatement(sql);
                  inserirCadsp.executeUpdate();
                  inserirCadsp.close();
                  MyConn.close();

            } catch (Exception e) {
                  System.out.print(""+e);
                  return false;
            }
            return true;
          }
```

Estrutura Padrão de um Algoritmo

ALGORITMO Nome_do_Algoritmo

VARIAVEIS
tipo variavel1
tipo variavel2
tipo variavel3

INICIO

ESCREVA "Digite um valor"
LEIA variavel1

ESCREVA "Digite outro valor"
LEIA variavel2

variavel3 <- variavel1 * variavel2
ESCREVA "Resultado final: ", variavel3

FIM

Tipos mais comuns:

- inteiro - 1 2 3 4
- real - 1.2 2.5 3.4
- texto - oi, maria
- logico - V ou F, true ou false

1 - Faça um algoritmo que leia os valores de A, B, e armazene o resultado de umaa multiplicacao em uma C e em seguida imprima na tela a multiplicacao entre A e B

Algoritmo CalculaMultiplicacao

inteiro A
inteiro B
inteiro C

Inicio
Escreva "Digite o valor de A: "
Leia A

Escreva "Digite o valor de B: "
Leia B

C <- A * B

Escreva "O resultado da multiplicação de A por B é: ", C

Fim

Crie um Algoritmo que calcule o quadrado de um numero