**Introdução a Algoritmos**

**Algoritmo**

Sequência de instruções que resolve determinado problema.

**Lógica de Programação**

Conjunto de raciocínios utilizados para criar um algoritmo.

**Lógica**

Sequência de passos para resolver um determinado problema.

Ciência dos princípios formais do raciocínio. A lógica é muito importante no desenvolvimento de algoritmos. Presente em diversas áreas da computação.

**Programa**

Algoritmo escrito em uma linguagem de programação específica, isto é, um algoritmo que pode ser executado em um computador.

***Formas de Representação de Algoritmos***

**Descrição Narrativa**

Forma de linguagem natural.

Desvantagem: Pouco usada na prática porque dá margem para interpretações erradas.

**Fluxograma**

Representação gráfica do algoritmo. Formas geométricas diferentes mostram ações distintas.

Objetivo: Facilitar o entendimento de uma ideia.

**Diagrama de Chapin**

Criado por Ned Chapin a partir de trabalhos de Nassi Shneiderman. Ideia de substituir o fluxograma. Representa a visão hierárquica e estruturada da lógica e um programa. Torna-se mais simples de codifica-lo futuramente em pseudocódigo.

**Pseudocódigo**

Também conhecido como português estruturado ou portugol. Ponto intermediário entre o entendimento humano e de uma linguagem de programação. Pode ser facilmente traduzido para uma linguagem de programação.

**Tipos de Dados e Instruções Primitivas**

**Dado**

Matéria prima da informação. De forma isolada não traz conhecimento.

Exemplo: 30

**Informação**

Dados tratados e analisados traz um significado.

Exemplo: idade 30 anos.

**Instrução**

Conjunto de palavras-chaves (vocabulário) de uma linguagem de programação para o processamento de dados.

Exemplo: no Visualg o comando “Escreva” corresponde a uma ação do algoritmo.

**Dados na Computação**

No contexto da computação, dados são tratados (processados) por um computador, para gerar informações ou resolver problemas. Os dados manipulados por um algoritmo podem possuir natureza distinta. O tipo de um dado define o conjunto de valores ao qual o dado pertence.

**Dados Numéricos – Inteiros**

Tipo de dados numéricos positivos e negativos (exceto números fracionários).

Exemplo: 33 / 0 / -128

**Dados Numéricos – Reais**

Tipo de dados numéricos positivos, negativos e fracionários.

Exemplo: 2 / 23.7 / 0 / -128 / -26.8 / 5 / 5.0 / Se o número vem seguido de um “.” (ponto) significa que ele é real.

**Dados Literais**

Sequência contendo letras, números e símbolos especiais. Uma sequência de Caracteres deve sempre ser indicada entre aspas duplas “”.

Exemplo: “Casa”, “123”, “num\_apartamento”, “Esta é uma frase formada por caracteres, “”.

Conhecidos como: String e Caractere.

**Dados Lógicos**

Dados com valores verdadeiro ou falso. Também chamado de Tipo Booleano.

Exemplo: Se <condição> Verdadeiro Senão Falso.

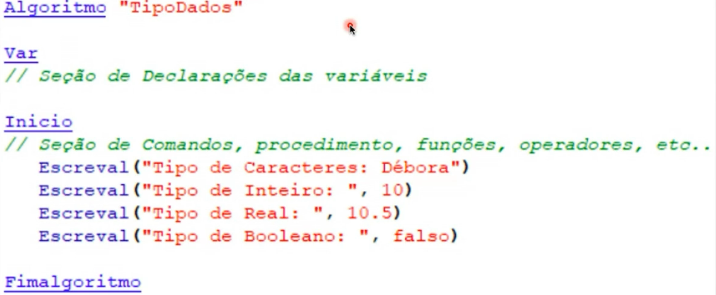
**Em síntese**

A árvore abaixo resume a classificação dos dados com relação aos tipos de dados apresentados.



**Exemplo de Tipos de Dados**

Literal (Caractere), Numérico (Inteiro e Real), e Booleano.



# **Escreval** serve para dar um espaçamento entre strings.

**Variáveis e Constantes**

**Variáveis**

“Gavetinha”. Uma variável armazena um valor de determinado tipo que pode variar ao longo da execução do programa.

Representação no Visualg: *nome\_da\_variável: tipo*

Exemplo no Visualg: *idade: inteiro*

**Identificador da Variável**

Nome único criado pelo programador. Não pode ser uma palavra reservada. Não pode conter espaços em branco. Pode conter apenas letras, dígitos e sublinha. Deve começar com uma letra (por padrão utilizam-se letras minúsculas). Deve permitir a identificação do valor que representa.

Exemplo: idade, nome, etc.

**Constantes**

Tudo aquilo que é fixo ou estável. Dado cujo valor se manterá inalterado toda vez que o programa for utilizado.

Exemplo: PI <- 3.14

**Operadores**

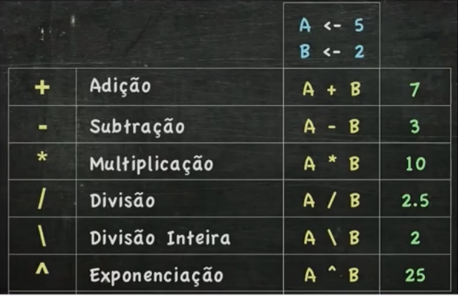
Variáveis e Constantes podem ser utilizadas para cálculos numéricos. Operadores serão utilizados em conjunto com as variáveis e constantes. Operadores são utilizados para executar cálculos numéricos e relacionar expressões.

São eles: Aritméticos, Relacionais, Lógicos

**Operadores Aritméticos**

Unários: inversão de valor

Binários: atuam em operações de exponenciação, multiplicação, divisão, adição e subtração.



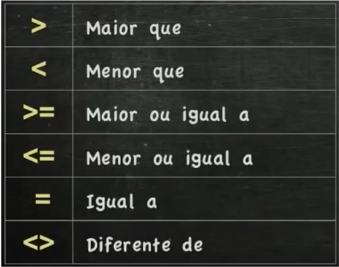
**Operadores Relacionais**

Utilizados para realizar expressões.

*Expressões*: Combinação de variáveis, constantes e operadores que, quando avaliada, resulta em um valor.

*Expressão aritmética*: resulta em um número (inteiro ou real).

*Expressão lógica*: resulta em VERDADEIRO ou FALSO.



**Operadores Lógicos**

São utilizados em expressões lógicas E, OU e NÃO. Sempre resultam em VERDADEIRO ou FALSO.



**Entrada, Processamento e Saída**

**Sistema**

Sistema é um conjunto de partes coordenadas, que concorrem para a realização de um conjunto de objetivos. Sistema é um conjunto de elementos interdependentes, ou um todo organizado, ou partes que interagem formando um todo unitário e complexo.

Sistema é um conjunto de componentes e processos que visam transformar determinadas entradas em saídas.

As Atividades de um Sistema: Entrada / Processamento / Saída

# Um sistema pode ser dividido em subsistemas. Cada subsistema tem sua funcionalidade, sendo requisitado quando necessário. Um sistema pode ser uma união de subsistemas, formando um sistema completo.

Um programa tem como objetivo resolver um problema. Para isso, geralmente é necessário a entrada de dados. Esses dados são processados. Como resposta, solução do problema em questão, ocorre ou resulta em uma saída de dados.

Passos para resolução de problemas:

Entendimento do Problema;

Criação de uma Sequência de operações para solução do problema;

Execução desta sequência;

Verificação da adequação da solução;

**Processamento**

Execução:

dos passos para resolver determinado problema;

de cada linha descrita no algoritmo;

de instruções;

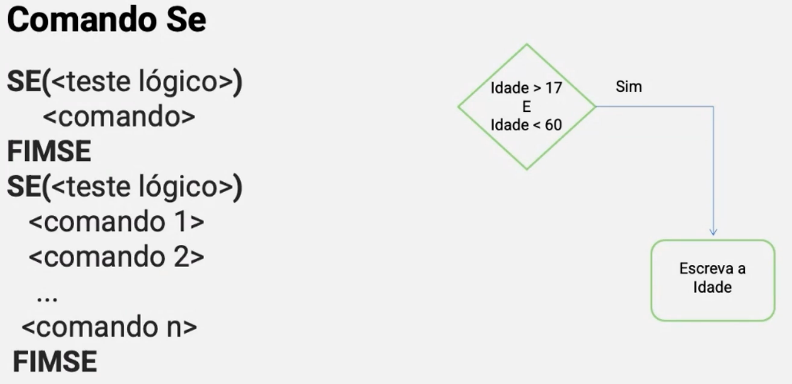
**Saída de Dados**

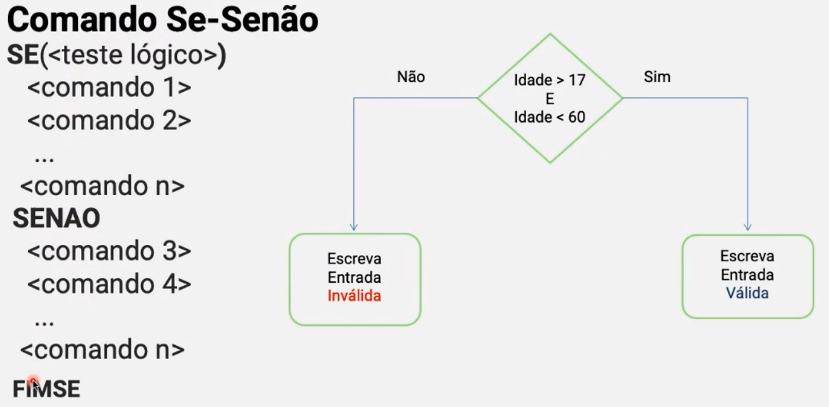
É tão simples que eu não vou nem escrever nada.

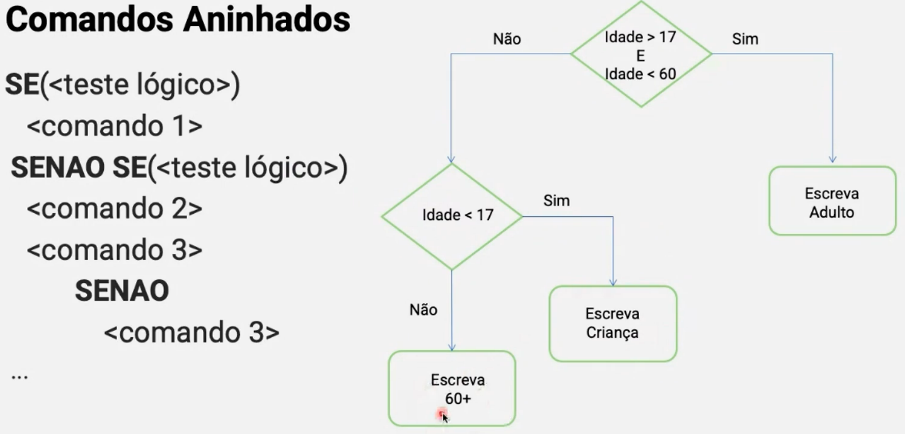
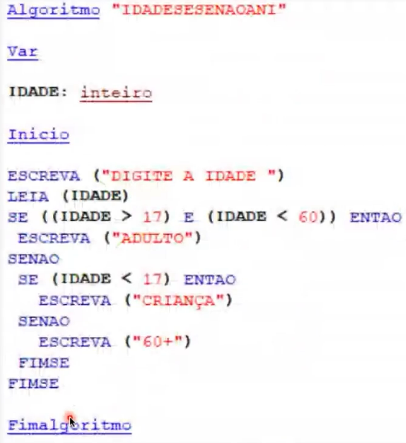
**Estruturas de Decisão**

**Estruturas de Controle**

Comandos de controle são a essência de qualquer linguagem. Gerenciam o fluxo da execução do programa.

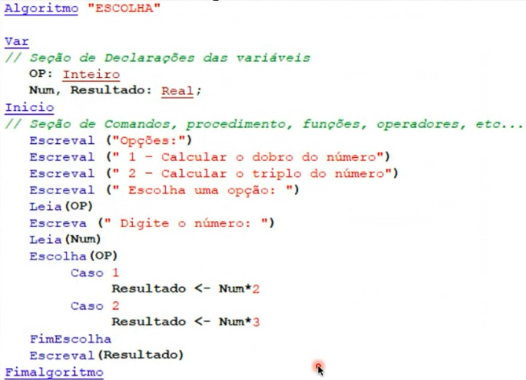
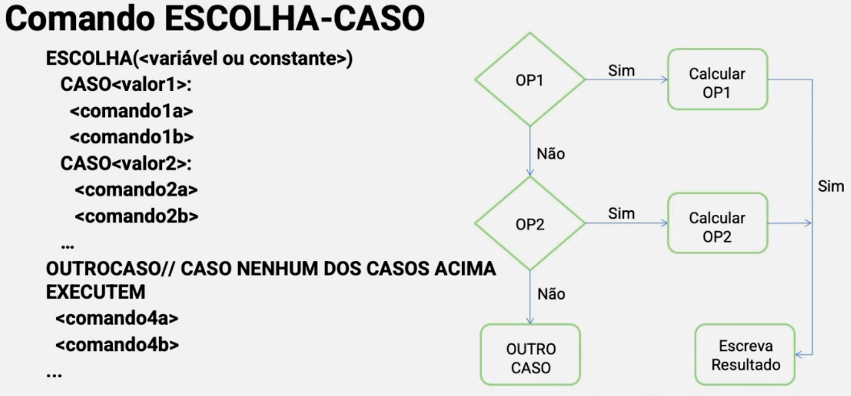






**Estrutura de Seleção Múltipla**

Determinam qual a ação a ser tomada com base no resultado de uma seleção. Permitem selecionar entre ações alternativas dependendo de critérios desenvolvidos no decorrer da execução do programa.



**Estruturas de Repetição**

**Estruturas de Repetição (ou Laços)**

Um algoritmo que necessita repetir parte(s) do código. Mecanismos de repetição são recursos importantes e muito utilizados em algoritmos. Cada execução de uma parte do código em um laço trata-se de uma interação. Assim uma dada parte do código pode ser executada quantas vezes forem necessárias.

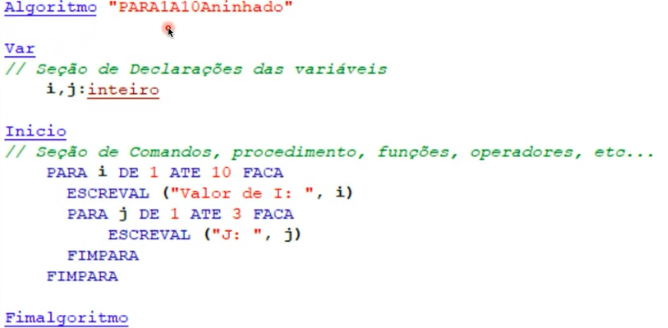
**Estrutura PARA**

Consistem em empregar uma variável, geralmente um contador, para controlar as repetições a serem executadas. Forma geral da estrutura PARA:

para <variável> de <valor-inicial> ate <valor-limite> [passo <incremento>] faca <sequência-de-comandos>

fimpara

**Exemplo PARA – Aninhados**



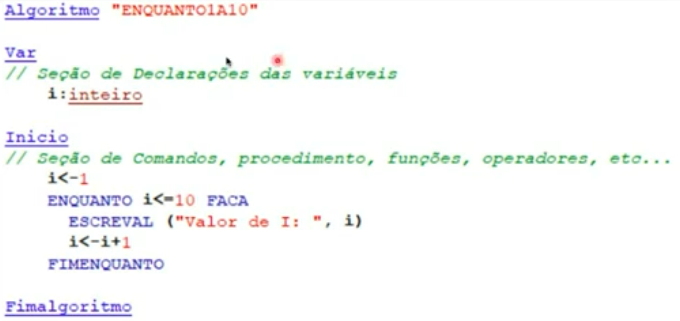
**Estrutura de Repetição – Enquanto**

Permite repetir um trecho de código enquanto uma determinada condição for verdadeira. Em geral, uma estrutura ENQUANTO fornece um código mais simples e fácil de ser entendido do que a estrutura PARA. Não se conhece a priori o número de repetições que serão executadas no corpo do laço.

Exemplo: **ENQUANTO** <expressão booleana> **FACA**

<sequência de comandos>

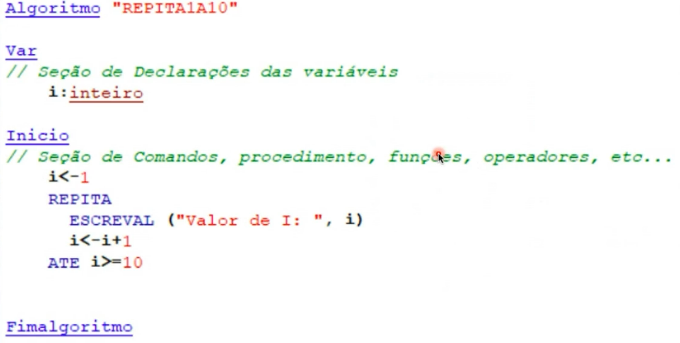
**FIMENQUANTO**

****

**Estrutura de Repetição – REPITA... ATÉ**

Semelhante a estrutura ENQUANTO. Permite repetir um trecho do código enquanto uma determinada condição for verdadeira.

*Diferença*: O bloco de comandos associados a estrutura de repetição e executado obrigatoriamente pelo menos uma vez. Isso acontece porque a avaliação da condição é feita no final da estrutura de repetição REPETA... ATÉ.



**Vetor**

**Estrutura Homogênea Unidimensional**

Estrutura de dados muito simples, também conhecida como vetor ou array. Possui apenas uma dimensão e pode armazenar diversas variáveis do mesmo tipo. Cada item (ou elemento) do vetor pode ser acessado por um índice. Exemplo: lista, dicionário e etc.

**Declaração**

VARIÁVEL: vetor [VALOR INICIAL.. VALOR FINAL] de TIPO

Exemplo:

nomes: vetor [1..5] de caracter

**Para usar o vetor é necessário usar uma estrutura de repetição:**

PARA <variável> DE <valor-inicial> ATE <valor-limite> [passo<incremento>] FACA

<sequência-de-comandos para acessar um vetor>

FIMPARA

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Matriz**

**Estrutura Homogênea Multidimensional**

Um vetor de vetores, ou seja, conjunto de variáveis do mesmo tipo. Possui várias dimensões. Vetores são, na verdade, matrizes de uma única dimensão. Cada item da matriz é acessado por um índice.

**Matriz**

As matrizes são comumente referenciadas através de suas dimensões (quantidade de linhas e colunas).

A notação comum é: MxN, onde

M é a dimensão horizontal (quantidade de linhas)

N é dimensão vertical (quantidade de colunas).

Tabela

Descrição gerada automaticamente com confiança média

**Declaração**

VARIÁVEL: vetor [VALOR INICIAL L.. VALOR FINAL L, VALOR INICIAL C.. VALOR FINAL C] de TIPO

**Exemplo**

notas: vetor [1... 50, 1... 4] de inteiro

***>>O VisualG aceita a palavra reservada “vetor” também em situações de matrizes<<.***

**Para usar o vetor é necessário usar 2 estruturas de repetição:**

PARA <variável> DE <valor-inicial> ATE <valor-final> [passo <incremento>] FACA

PARA <variável> DE <valor-inicial> ATE <valor-final> [passo <incremento>] FACA

<sequência-de-comandos para acessar um vetor>

FIMPARA

FIMPARA

**Procedimento Sem Parâmetros**

**Modularização**

Imagine um sistema inteiro desenvolvido dentro da mesma estrutura. Complexidade depende do tamanho. Dividir para conquistar. Modularização: Procedimento ou Função / Com ou sem parâmetros.

**Modularização Procedimentos**

Modularização: Divisão de tarefas, o programa é dividido em partes ou módulos.

Procedimento: São blocos de programas que executam determinada tarefa. É um subprograma que não retorna valor nenhum.

**Estrutura Procedimento Sem Parâmetro**

PROCEDIMENTO <nome\_do\_procedimento>

VAR

//

INICIO

ESCREVA(“Sou como a estrutura sequencial”)

FIMPROCEDIMENTO

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Procedimento Com Parâmetro**

**Procedimentos**

Modularização: Divisão de tarefas, o programa é dividido em partes ou módulos.

Procedimento: São blocos de programas que executam determinada tarefa.

Procedimento é um subprograma que não retorna nenhum valor, mas podem receber valor.

**Estrutura**

PROCEDIMENTO <nome\_do\_procedimento> [(<sequência-de-declarações-de-parametros>)]

//Seção de Declarações Internas

INICIO

//Seção de Comandos

FIMPROCEDIMENTO

a <sequencia-de-declarações-de-parametros> é uma sequência de [var] <sequencia-de-parametros>: <tipo-de-dado> separadas por ponto e vírgula

Texto

Descrição gerada automaticamente

**Funções Sem Parâmetros**

**Modularização Função**

Imagine um sistema inteiro desenvolvido dentro da mesma estrutura. Complexidade depende do tamanho. Dividir para conquistar. Modularização: procedimento ou função / com ou sem parâmetros.

**Modularização Procedimentos**

Modularização: Divisão de tarefas, o programa é dividido em partes ou módulos.

Função: São blocos de programas que executam determinada tarefa. Função é um subprograma que retorna algum valor.

**Funções**

Modularização “”

Função “”

Diferença: Procedimento é um subprograma que não retorna nenhum valor.

Função é um subprograma que retorna valores.

**Estrutura**

FUNCAO <nome-de-função>: <tipo-de-dado>

VAR

//Seção de Declarações Internas

INICIO

//Seção de Comandos

RETORNE <valor>

FIMFUNCAO

**Funções com Parâmetros**

**Declaração**

FUNCAO <nome-de-função> [(<sequência-de-declarações-de-função>)]: <tipo-de-dado>

VAR

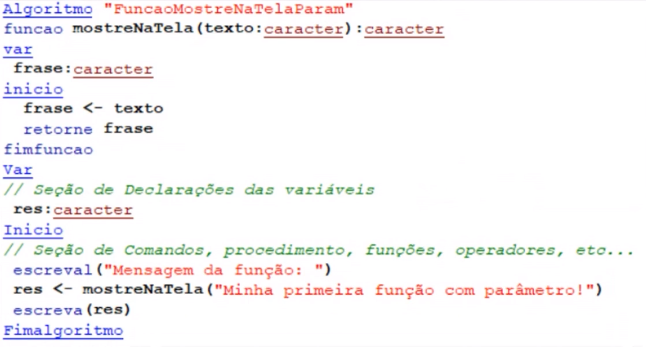
//Seção de Declarações Internas

INICIO

//Seção de Comandos

RETORNE <valor>

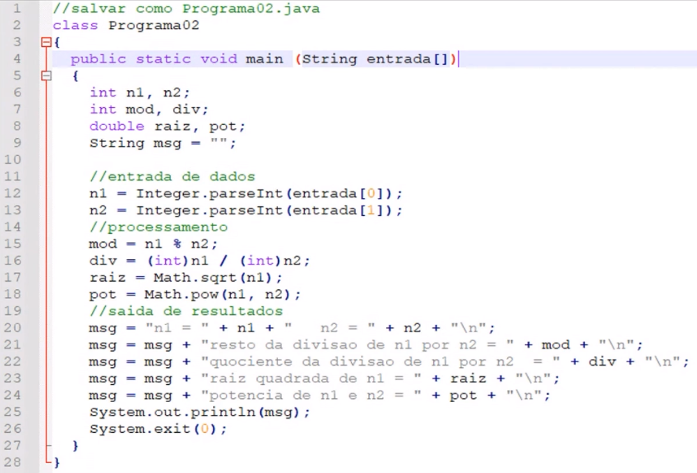
FIMFUNCAO



**Introdução à Linguagem Java**

**Fluxo de Dados no Java**

Exemplo:



double = “É usado quando precisamos armazenar números de ponto-flutuante (com parte fracionária) na faixa 4.94065645841246544e-324 até 1.79769313486231570e+308, possui precisão de 14 ou 15 dígitos significativos.”

Operadores:

