



Lógica de Programação

☰	Repositório no GitHub	https://github.com/marlonprado04/FACULDADE_analise_e_desenvolvimento_de_sistemas/tree/main/02_L
☰	Link da página no Notion (para README)	Lógica de Programação
↗	Plano de estudos DB	Faculdade: Análise e Desenvolvimento de Sistemas (AdS)
⌄	Status	Concluído
📅	Data	@11/01/2024

Navegação

Navegação (saber sem limites) DB

Aa	Name	#	Ordem
	Cursos e Conteúdos	1	
	Planos de estudo	2	
	Instituições	3	
	Tópicos		

▼ Índice

[Navegação](#)

[Índice](#)

[Introdução a Algoritmos](#)

[Introdução a Algoritmos](#)

[Algoritmos](#)

[Exemplo e Atividade Prática 1](#)

[Texto: Introdução a Algoritmos](#)

[Tipos de Dados e Instruções Primitivas](#)

[Tipos de Dados e Instruções Primitivas](#)

[Exemplos de Tipos de Dados e Instruções Primitivas](#)

[Exemplo e Atividade Prática 2](#)

[Texto: Tipos de Dados e Instruções Primitivas](#)

[Variáveis e Constantes](#)

[Variáveis e Constantes](#)

[Operadores](#)

[Exemplo e Atividade Prática 3](#)

[Texto: Variáveis e Constantes](#)

[Entrada, Processamento e Saída](#)

[Introdução, Conceitos e Entrada de Dados](#)

[Processamento](#)

[Exemplo e Atividade Prática 4](#)

[Texto: Entrada, Processamento e Saída de Dados](#)

[Estruturas de Decisão](#)

[Comandos de decisão - parte 1](#)

[Comandos de decisão - parte 2](#)

[Exemplo e Atividade Prática 5](#)

[Exemplo de Solução 5](#)

[Texto: Estruturas de Decisão](#)

[Estruturas de Repetição](#)

[Estruturas de Repetição - parte 1](#)

[Estruturas de Repetição - parte 2](#)

[Exemplo e Atividade Prática 6](#)

[Solução de Exemplo da Atividade 6](#)

[Texto: Estruturas de repetição](#)

[Vetor](#)

[Vetor/Array - parte 1](#)

[Vetor/Array - parte 2](#)

[Exemplo e Atividade Prática 7](#)

[Solução de Exemplo da Atividade 7](#)

[Texto: Vetor/Array](#)

[Matriz](#)

[Matriz - parte 1](#)

[Matriz - parte 1](#)

[Exemplo e Atividade Prática 8](#)

[Solução de Exemplo da Atividade 8](#)

[Texto: Matriz](#)

[Procedimentos sem Parâmetros](#)

[Modularização Procedimento](#)

[Procedimentos sem parâmetros](#)

[Exemplo e Atividade Prática 9](#)

[Solução de Exemplo da Atividade 9](#)

[Texto: Procedimentos sem Parâmetros](#)

[Procedimentos com Parâmetros](#)

[Procedimentos com e sem Parâmetros](#)

[Procedimentos com Parâmetros](#)

[Exemplo e Atividade Prática 10](#)

[Solução de Exemplo da Atividade 10](#)

[Texto: Procedimentos com Parâmetros](#)

[Funções sem parâmetros](#)

[Modularização Função](#)

[Funções sem parâmetros](#)

[Exemplo e Atividade Prática 11](#)

[Solução de Exemplo da Atividade 11](#)

[Texto: Funções sem parâmetros](#)

[Funções com parâmetros](#)

[Funções com e sem parâmetros](#)

[Funções com parâmetros](#)

[Exemplo e Atividade Prática 12](#)

[Solução de Exemplo da Atividade 12](#)

[Texto: Funções com parâmetros](#)

[Visão Geral da Linguagem Java](#)

[Ambiente Java e IDE](#)

[Meu primeiro programa Java](#)

[Exemplo e Atividade Prática 13](#)

[Solução de Exemplo da Atividade 13](#)

[Texto: Visão Geral da Linguagem Java](#)

[Introdução à Linguagem Java](#)

[Trabalhando com dados no Java](#)

[Fluxo de dados no Java](#)

[Exemplo e Atividade Prática 14](#)

[Solução de Exemplo da Atividade 14](#)

[Atividade prática](#)

[Texto: Introdução à Linguagem Java](#)

[Estrutura de Controle no Java](#)

[Estrutura de Decisão no Java](#)

[Estruturas de Repetição no Java](#)
[Exemplo e Atividade Prática 15](#)
[Solução de Exemplo da Atividade 15](#)
[Texto: Estrutura de Controle no Java](#)
[Módulos e matrizes em Java](#)
[Matrizes em Java](#)
[Modularização no Java](#)
[Exemplo e Atividade Prática 16](#)
[Solução de Exemplo da Atividade 16](#)
[Texto: Módulos e matrizes em Java](#)
[Materiais de apoio e referências](#)

Introdução a Algoritmos

▼ Introdução a Algoritmos

Como desenvolver o raciocínio lógico?

Para desenvolver o raciocínio lógico é necessário **resolver problemas**.

Como estudar algoritmo?

- Primeiro pensar no problema
- Pensar em como resolver o problema

Para estudar um algoritmo os passos necessários são:

1. Conhecer as regras
2. Entender o problema proposto
3. Indicar o que deve ser feito e em que ordem
4. Executar a sequência de passos e ver se isso resolve o problema

Exemplo:

Exemplo

Problema: Atravessar uma rua

Regras: Passar de uma calçada para outra sem ser atropelado

Qual sequência de ações resolve o problema?

1. Olhar para a esquerda
2. Olhar para a direita
3. Se estiver vindo carro – espere
4. Repita passo 1 e 2
5. Senão estiver vindo carro - atravesse

Uma solução para um problema pode variar, pois nem sempre os mesmos passos resolverão o problema

▼ Algoritmos

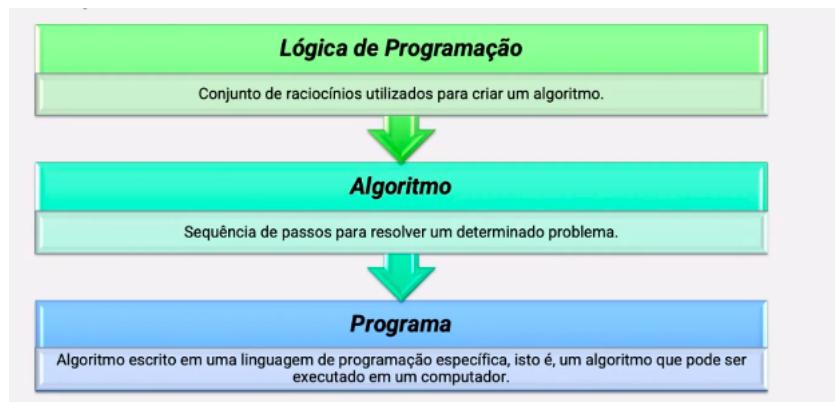
O que é um algoritmo?

É uma sequência de instruções que resolve determinado problema

Para criar um algoritmo precisamos desenvolver a lógica, após a lógica de programação, depois um algoritmo e por fim fazer um programa.

O que são cada passo?

- **Lógica** é a ciência dos princípios formais do raciocínio
- **Lógica de programação** é o conjunto de raciocínios para criar um algoritmo
- **Algoritmo** é a sequência de passos para resolver um problema
- **Programa** é escrito em uma linguagem de programação específica



Formas de representar algoritmos

Para representar um algoritmo podemos usar algumas formas, podendo ser:

- **Descrição narrativa** → linguagem natural, pouco usada pois é aberta à interpretações

Formas de Representação de Algoritmos

Descrição Narrativa

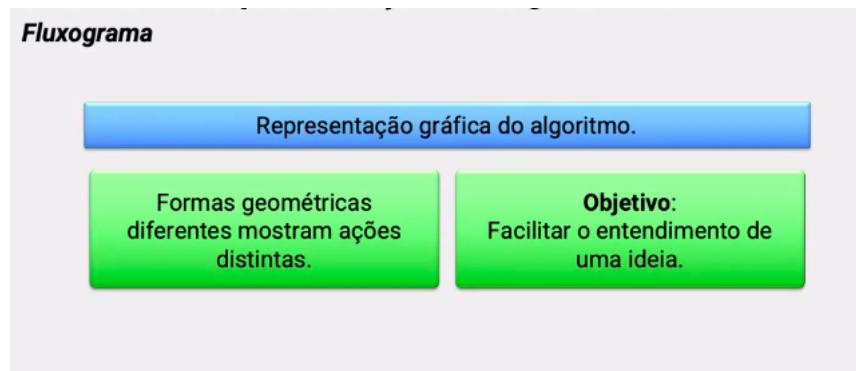
Forma de linguagem natural

Exemplo: Atravessar uma rua.

Principal desvantagem:

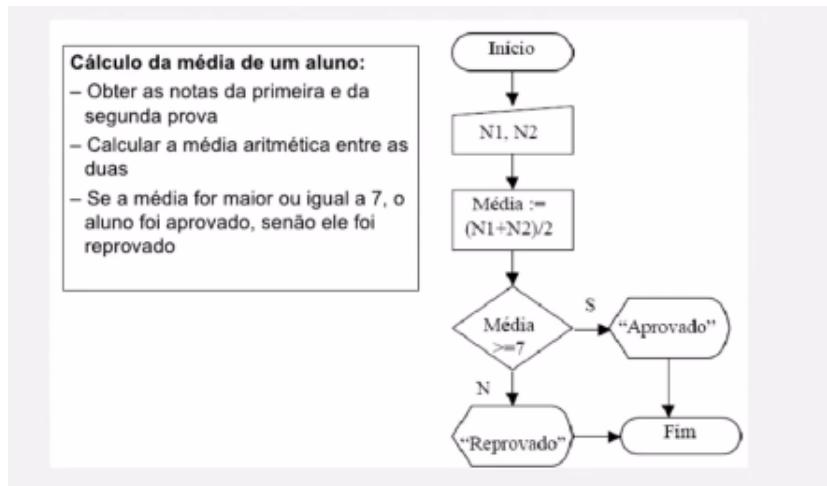
- Pouco usada na prática porque dá margem para interpretações erradas.
- Exemplo: “afrouxar ligeiramente as porcas” no algoritmo da troca de pneus. A expressão está sujeita a interpretações diferentes por pessoas distintas. O “afrouxar ligeiramente” não fica explícito em qual sentido e nem a intensidade necessária.

- **Fluxograma** → representação gráfica, formas geométricas representam diferentes ações, ajuda a entender a ideia



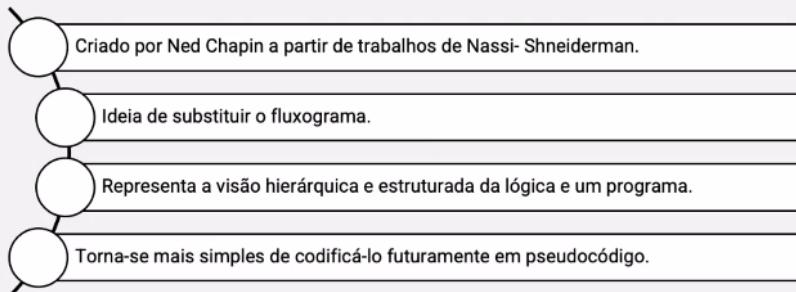
Principais Símbolos do Fluxograma

	= Início e final do fluxograma
	= Operação de entrada de dados
	= Operação de saída de dados em impressora
	= Operação de saída de dados em vídeo
	= Operações de atribuição e chamada ou retorno de subalgoritmo
	= Decisão



- **Diagrama de Chapin** → criado por Ned Chapin, criado para substituir o fluxograma, representa a visão hierárquica e estruturada de lógica de um programa

Diagrama de Chapin



- **Pseudocódigo (portugol ou português estruturado)** → intermediário entre o entendimento humano e linguagem de programação, facilmente traduzido para linguagem de programação

Pseudocódigo



▼ Exemplo e Atividade Prática 1

Estrutura do VisualG, programa usado para as atividades de exemplo:

Exemplo Prático 1 - Estrutura

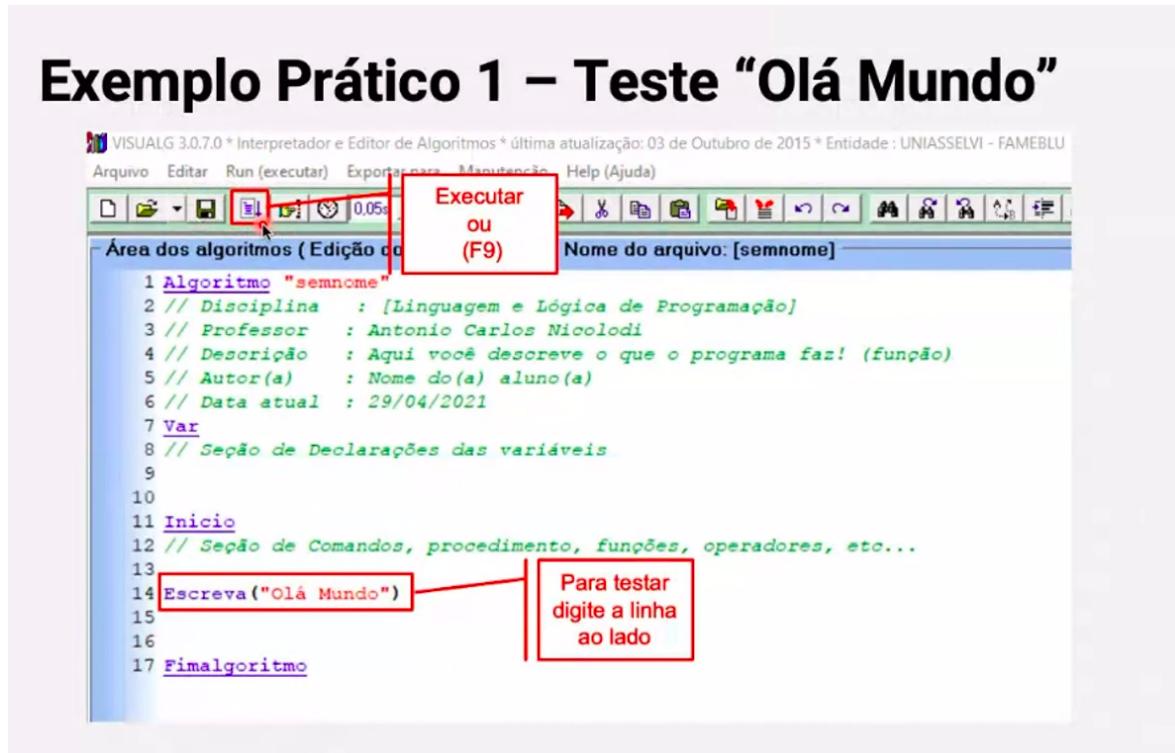
```
Area dos algoritmos ( Edição do código )
  1 Algoritmo "semnome"
  2 // disciplina : [Linguagem e Lógica de Programação]
  3 // Professor : Antonio Carlos Nicolodi
  4 // Descrição : Aqui você descreve o que o programa faz! (função)
  5 // Autor(a) : Nome do(a) aluno(a)
  6 // Data atual : 29/04/2021
  7 Var
  8 // Seção de Declarações das variáveis
  9
 10
 11 Início
 12 // Seção de Comandos, procedimento, funções, operadores, etc...
 13
 14
 15 Fimalgoritmo
```

Início e nome do algoritmo
do arquivo: [semnome]

Declarações de Variáveis

Passos do algoritmo que serão executados.

Atividade prática de exemplo:



Código desenvolvido:

```
Algoritmo "exemplo_01"
// Descrição : Exemplo 1
// Autor(a) : Marlon
// Data atual : 02/05/2024
Var
// Seção de Declarações das variáveis

Início
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreva("Olá Mundo!")

Fimalgoritmo
```

▼ Texto: Introdução a Algoritmos

▼ PDF

[01_introducao_a_algoritmos.pdf](#)

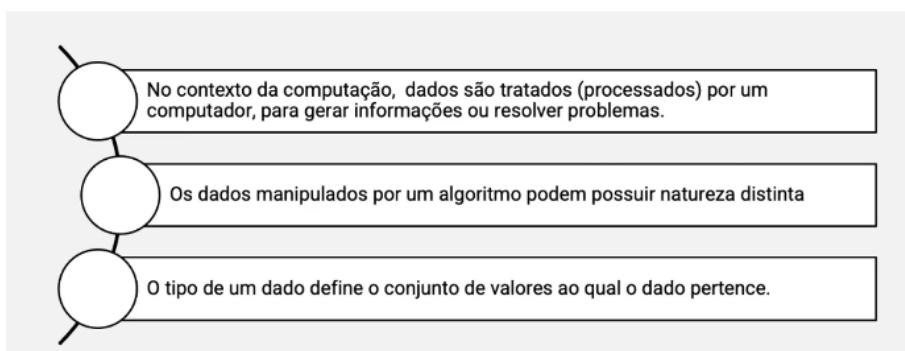
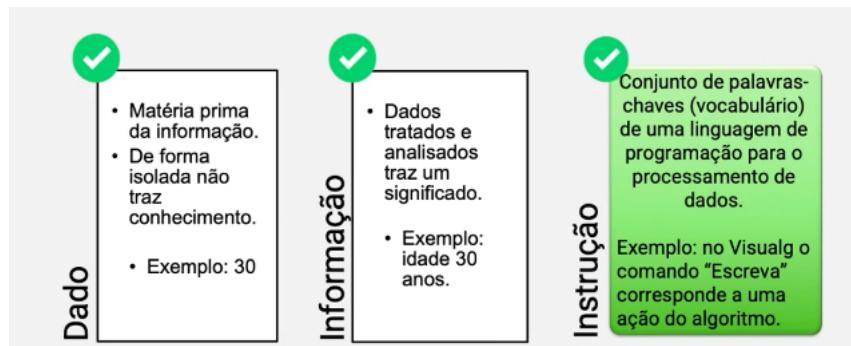
Tipos de Dados e Instruções Primitivas

▼ Tipos de Dados e Instruções Primitivas

Dado é a matéria prima da informação e não traz conhecimento

Informação é o significado trazido por um **conjunto de dados tratados e organizados**

Instrução é o **conjunto de palavras chave** usado para o processamento de dados



Tipos de dados

Na programação existem os seguintes tipos de dados:

- **Dados numéricos inteiros (int / integer)** → números sem casas decimais
- **Dados numéricos reais (double / float)** → números inteiros e com casas decimais
- **Dados literais (alfanuméricicos)** → sequências contendo letras, números e símbolos especiais
- **Dados lógicos (booleanos)** → possuem valores de verdadeiro ou falso

▼ Exemplos de Tipos de Dados e Instruções Primitivas

Exemplos de dados **numéricos**

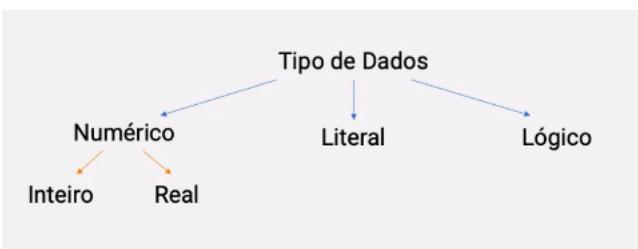
Vamos testar nosso conhecimento?

- Classificação dos dados quanto ao tipo:
 - 24 - número inteiro positivo
 - 0 - número inteiro
 - -12 - número inteiro negativo
 - 24.01 - número real positivo com duas casas decimais
 - 144. - número real positivo com zero casas decimais
 - -13.3 - número real negativo com uma casa decimal
 - 0.0 - número real com uma casa decimal
 - 0. - número real com zero casas decimais

Exemplos de dados **alfanuméricos**

- Classificação dos dados quanto ao tipo:
 - "QUAL ?" - literal de comprimento 6
 - " " - literal de comprimento 1
 - "qUaL ?!\$" - literal de comprimento 8
 - "AbCdefGHi" - literal de comprimento 9
 - "1-2+3=" - literal de comprimento 6
 - "0" - literal de comprimento 1
 - .V. - valor lógico verdadeiro
 - .F. - valor lógico falso

Os tipos de dados podem ser resumidos na árvore abaixo:



▼ Exemplo e Atividade Prática 2

Exemplo de **pseudocódigo**

Literal (Caracterer), Numérico (Inteiro e Real) e Booleano

```
Algoritmo "TipoDados"
  Var
    // Seção de Declarações das variáveis
  Início
    // Seção de Comandos, procedimento, funções, operadores, etc..
    Escreval("Tipo de Caracteres: Débora")
    Escreval("Tipo de Inteiro: ", 10)
    Escreval("Tipo de Real: ", 10.5)
    Escreval("Tipo de Booleano: ", falso)
  Fimalgoritmo
```

▼ Texto: Tipos de Dados e Instruções Primitivas

▼ PDF

[02_tipos_de_dados_e_instrucoes_primitivas.pdf](#)

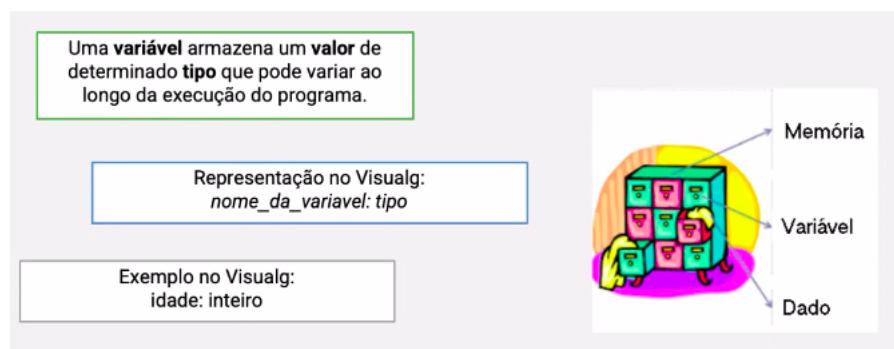
Variáveis e Constantes

▼ Variáveis e Constantes

Variáveis

Variáveis são como **gavetas**, pois assim como gavetas elas **podem armazenar valores de um determinado tipo que podem e este tipo pode variar** durante a execução da operação.

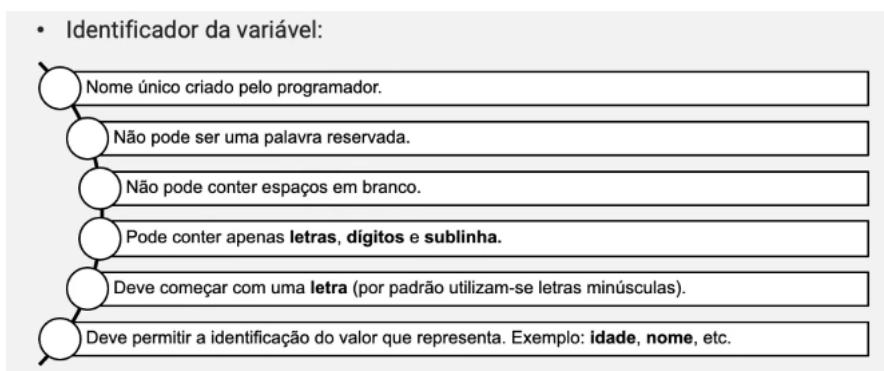
No VisualG declaramos uma variável informando o `nome:tipo`.



Para criar variáveis é necessário que elas possuam:

- **Identificador:**
 - **Nome único** criado pelo programador
 - **Não ser** uma palavra reservada
 - Não ter espaços
 - Conter apenas **letras, dígitos e sublinha**

- Deve **começar com letra** (convenção usar minúscula) → case sensitive
- Deve **identificar o valor** que representa → ex: idade, nome, etc



Variáveis válidas devem respeitar as regras anteriores, conforme abaixo:



Variáveis inválidas:



Motivos da invalidez:

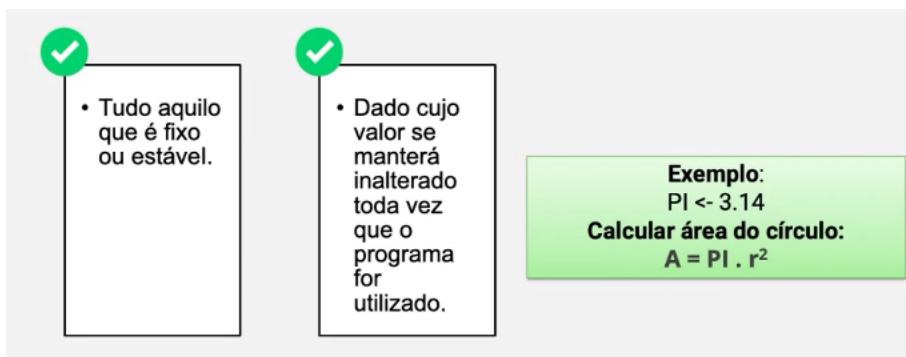
- **1dia**: começa com dígitos
- **salário do empregado** → possui espaço e acentuação
- **ddd/telefone** → possui "/" que é um caractere inválido
- **3prova1** → começa com dígito ao invés de letra

Constantes

Constantes são tudo o que é **fixo** ou **estável**.

São **dados que se mantém inalterados** durante a execução do programa.

Abaixo um exemplo:



Operadores

Variáveis e constantes são usadas para cálculos numéricos.

- Os operadores são **usados para realizar a operação de cálculos numéricos** junto às variáveis e constantes.
- Operadores também são usados para **relacionar expressões**.

-
- The diagram shows a list of three points enclosed in a light gray box:
 - **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

▼ Operadores

Os operadores são usados para **fazer cálculos numéricos** e **relacionar expressões**.

Operadores são divididos em 3 grupos:

- **Aritméticos**
- **Relacionais**
- **Lógicos**

- **Variáveis e Constante** podem ser utilizadas para cálculos numéricos
- **Operadores** são utilizados para executar cálculos numéricos e relacionar expressões. São eles:

Aritméticos

Relacionais

Lógicos

Operadores Aritméticos

São divididos em 2 grupos:

- **Unários**: servem para inverter valores
- **Binários**: realizam operações de exponenciação, multiplicação, divisão, adição e subtração



Tabela com operadores aritméticos:

		$A \leftarrow 5$	
		$B \leftarrow 2$	
+	Adição	$A + B$	7
-	Subtração	$A - B$	3
*	Multiplicação	$A * B$	10
/	Divisão	A / B	2.5
\	Divisão Inteira	$A \backslash B$	2
^	Exponenciação	$A ^ B$	25

No exemplo da tabela são declaradas 2 variáveis e seus valores, sendo que A recebe 5 e B recebe 2.

Operadores Relacionais

São usados para **relacionar expressões**.

As expressões podem ser:

- **Expressões aritméticas** → resultam em um valor inteiro ou real
- **Expressões lógica** → resultam em verdadeiro ou falso

Operadores Relacionais

- Utilizados para relacionar **expressões**.
- **Expressões:**
 - Combinação de variáveis, constantes e operadores que, quando avaliada, resulta em um valor.
- **Expressão aritmética:**
 - resulta em um numero (inteiro ou real).
- **Expressão lógica:**
 - resulta em VERDADEIRO ou FALSO.

Abaixo a tabela de operadores relacionais:

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

Fonte: Gustavo Guarabata

Operadores Lógicos

- São usados em expressões lógicas de **E**, **OU** e **NÃO**.
- Sempre resultam em **VERDADEIRO** ou **FALSO**

Operadores Lógicos

- São utilizados em expressões lógicas **E**, **OU** e **NÃO**.
- Sempre resultam em **VERDADEIRO** ou **FALSO**.
- Serão abordadas mais detalhadamente na introdução de **estruturas condicionais**.

Exemplo prático:

- Exemplo:
- A = 10
- B = 5
- Problema: A e B **obrigatoriamente** precisam ser números positivos:
 

Outro exemplo seria se A e B armazenassem VERDADEIRO ou FALSO.

Abaixo uma tabela que representa isso:

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

Na primeira linha:

- A é VERDADEIRO e B é FALSO, ao usar operador E, como ambos são VARDADEIRO, isso resulta em VERDADEIRO.
- Os demais seguem essa lógica, onde o **operador E precisa que ambos sejam verdadeiros**

▼ Exemplo e Atividade Prática 3

No exemplo abaixo foi criado um algoritmo para exemplificar as operações e variáveis.

Exemplo

```

Algoritmo "VariaveiseOperadores"

Var
// Seção de Declarações das variáveis
nomeAlg: caracter
resultadoSoma:inteiro
resultadoSubtracao:inteiro
resultadoDivisao:real
resultadoMultiplicacao:real
resultadoExp1: real
resultadoExp2: real

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
// Sinal de atribuição "<-", quando uma variável recebe um valor
    nomeAlg <- "Aprendendo Variáveis e Operadores"
    resultadoSoma <- 2+2
    resultadoSubtracao <- 10-2
    resultadoMultiplicacao <- 2*2.3
    resultadoDivisao <- 10/5
    resultadoExp1 <- 2+2+10-2+2*2.3+10/5
    resultadoExp2 <- (2+2+(10-2+2)*2.3+10/5))

    Escreval("Nome do Algoritmo: ", nomeAlg)
    Escreval("Resultado da Soma: ", resultadoSoma)
    Escreval("Resultado da Subtração: ", resultadoSubtracao)
    Escreval("Resultado da Multiplicação: ", resultadoMultiplicacao)
    Escreval("Resultado da Divisão: ", resultadoDivisao)
    Escreval("Resultado da Expressão 1: ", resultadoExp1)
    Escreval("Resultado da Expressão 2: ", resultadoExp2)

Fimalgoritmo

```

Este código de exemplo resulta na tela abaixo:

```

Nome do Algoritmo: Aprendendo Variáveis e Operadores
Resultado da Soma: 4
Resultado da Subtração: 8
Resultado da Multiplicação: 4.6
Resultado da Divisão: 2
Resultado da Expressão 1: 18.6
Resultado da Expressão 2: 29
>>> Fim da execução do programa !

```

Abaixo o exemplo do código feito manualmente:

```

Algoritmo "ExemplosDeVariavelEOperacoes"
//
//
// Descrição : Exemplifica o uso de operações e variáveis
// Autor(a)   : Marlon Prado
// Data atual : 17/02/2024

```

```

Var

varCaractere: caractere
varResultadoSoma: inteiro
varResultadoSubtracao: inteiro
varResultadoDivisao: real
varResultadoMultiplicacao: real
varResultadoExp1: real
varResultadoExp2: real

Inicio

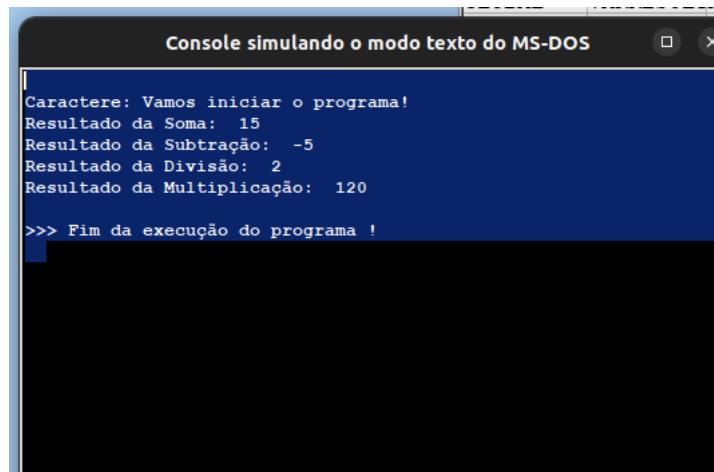
varCaractere <- "Vamos iniciar o programa!"
varResultadoSoma <- 1+2+3+4+5
varResultadoSubtracao <- 5-4-3-2-1
varResultadoDivisao <- (50/5)/(50/10)
varResultadoMultiplicacao <- 1*2*3*4*5

Escreval("Caractere: ", varCaractere)
Escreval("Resultado da Soma: ", varResultadoSoma)
Escreval("Resultado da Subtração: ", varResultadoSubtracao)
Escreval("Resultado da Divisão: ", varResultadoDivisao)
Escreval("Resultado da Multiplicação: ", varResultadoMultiplicacao)

Fimalgoritmo

```

E o resultado foi:



▼ Texto: Variáveis e Constantes

▼ PDF

[03_variaveis_e_constants.pdf](#)

Entrada, Processamento e Saída

▼ Introdução, Conceitos e Entrada de Dados

Introdução

O que é um sistema?

É um conjunto de partes coordenadas que concorrem para realizar um conjunto de objetivos

É um conjunto de
elementos interdependentes que juntos **formam um todo unitário e complexo**

É um conjunto de componentes e processos que
transformam entradas em saídas

- *O que é Sistema?*
- Observe algumas definições de sistema a seguir:
 - “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”.

Como funcionam as atividades de um sistema?

Um sistema opera sempre com uma
entrada → processamento → saída.

Onde as
entradas são todos os dados que serão **processados**.
O
processamento é a **operação realizada nos dados**.
A
saída são os **dados processados**.



Um sistema pode ser dividido em subsistemas

Cada subsistema tem sua funcionalidade e pode ser chamado somente quando necessário

Um sistema pode ser a união de subsistemas que compõem um sistema complexo.

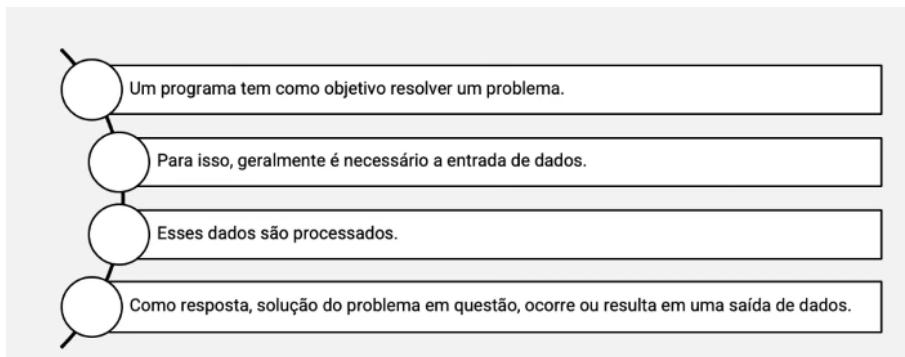
- Um sistema pode ser dividido em subsistemas;
- Cada subsistema tem sua funcionalidade, sendo requisitado quando necessário;
- Um sistema pode ser uma união de subsistemas, formando um sistema completo.

Abaixo um diagrama de exemplo:



Nesse contexto a faculdade pode ser vista como um sistema que recebe estudantes, professores e administradores e ao fim entrega estudantes, serviços e comunidade.

Passos para resolução do problema



Para resolver um problema precisamos respeitar certos passos que auxiliam nessa resolução, sendo eles:

1. **Entendimento do problema** → qual o problema? O que soluciona esse problema?
2. **Criar sequência de etapas para resolução** → fluxograma, algoritmo
3. **Execução da sequência** → executar o fluxograma ou código desenvolvido
4. **Verificação da solução** → verificar se o problema foi resolvido

- Passos para resolução de problemas:
 1. Entendimento do Problema;
 2. Criação de uma sequência de operações para solução do problema;
 3. Execução desta sequência;
 4. Verificação da adequação da solução.
- O computador desempenha apenas uma parte deste processo (3º passo).

Se repararmos, o **computador realiza apenas uma etapa (3)** desse processo.

Ou seja, quanto mais trabalharmos as demais etapas (1 e 2), mais facilitada será a etapa 3 e por consequência a etapa 4 será atendida.

| Devemos nos preocupar em entender o problema e definir muito bem a solução

Entrada de Dados

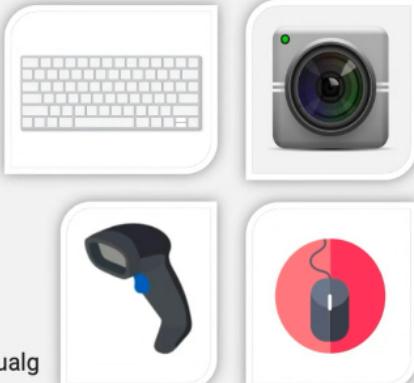
A entrada de dados pode ocorrer através de diversos periféricos, como:

- Teclado
- Câmera
- Leitor Ótico

- Mouse
- etc

Entrada de Dados

- Pode ocorrer através de:
 - Teclado;
 - Câmera;
 - Leitor Ótico;
 - Mouse, etc.
- Exemplo: instrução *Leia* – Portugol Visualg



Fonte: imagens gratuitas de vecteezy: <https://www.vecteezy.com/>

Por exemplo, no portugol usamos a instrução **Leia** que aguarda a digitação de um usuário.

▼ Processamento

O processamento faz parte da execução, que é composta por:

- Os passos necessários para resolver o problema
- Cada linha de código

Saída de Dados

A saída de dados pode ocorrer através de diversos periféricos, por exemplo, no portugol usamos a palavra **Escreva** para que o sistema escreva na tela a mensagem solicitada.

- Execução:
 - dos passos para resolver determinado problema.
 - de cada linha descrita no algoritmo.
 - de instruções.

PROCESSAMENTO

Alguns exemplos de periféricos de saída de dados:

- **Monitor** → saem imagens, palavras, vídeos
- **Impressora** → saem impressões
- **Caixa de som** → saem sons gerados

- **Projetor** → saem imagens projetadas

- Pode ocorrer por diversos periféricos;
- Exemplo em algoritmo: instrução *Escreva ou Escreval* – Portugol Visualg;
- Vamos conhecer os periféricos que correspondem a saída de dados:

Existem periféricos que atual como entrada e saída ao mesmo tempo?

Sim!

Alguns periféricos que são saída e entrada:

- **Monitores touch screen** → você pode tocar na tela e ver as imagens
- **SSD / HD (hard disk)** → pode gravar e ler informações
- **Gravador de DVD** → você pode gravar e ler as informações
- **Smartphone** → pode usar câmera, teclado, touch e a tela para entrada e saída (além do processamento)

▼ Exemplo e Atividade Prática 4

Abaixo um exemplo de algoritmo com entradas e saídas de dados.

No trecho abaixo ocorre a entrada de dados:

```

Exemplo
Algoritmo "ExEntradaProcessamentoSaida"
Var
// Série de Declarações das variáveis
    num1:real
    num2:real
    resultado: real
Início
// Série de Comandos, procedimento, funções, operadores, etc...
    Escreval("Digite o primeiro número: ")
    Leia(num1)
    Escreval("Digite o segundo número: ")
    Leia(num2)

    resultado <- num1*num2

    Escreval("Os números são: ")
    Escreval("Primeiro número digitado: ", num1)
    Escreval("Segundo número digitado: ", num2)
    Escreval("Resultado: ", resultado)

Fimalgoritmo

```

No trecho abaixo ocorre o processamento:

Processamento dos Dados

```
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreval("Digite o primeiro número: ")
Leia(num1)
Escreval("Digite o segundo número: ")
Leia(num2)

resultado <- num1*num2

Escreval("Os números são: ")
Escreval("Primeiro número digitado: ", num1)
Escreval("Segundo número digitado: ", num2)
Escreval("Resultado: ", resultado)
```

No trecho abaixo ocorre a saída de dados:

Saída dos Dados

```
Início
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreval("Digite o primeiro número: ")
Leia(num1)
Escreval("Digite o segundo número: ")
Leia(num2)

resultado <- num1*num2

Escreval("Os números são: ")
Escreval("Primeiro número digitado: ", num1)
Escreval("Segundo número digitado: ", num2)
Escreval("Resultado: ", resultado)

Fim algoritmo
```

▼ Texto: Entrada, Processamento e Saída de Dados

▼ PDF

[04_entrada_processamento_e_saida_de_dados.pdf](#)

Estruturas de Decisão

▼ Comandos de decisão - parte 1

- Comandos de controle gerenciam o fluxo do programa

Uma das principais estruturas é o **se** e o **senão**

Existem dois tipos de comandos se, o composto e o aninhado, sendo eles:

Composto

Exemplo de código:

```

Algoritmo "IDADESE"
Var
  IDADE: inteiro
Inicio
  ESCREVA ("DIGITE A IDADE: ")
  LEIA (IDADE)
  SE ((IDADE > 17) E (IDADE < 60)) ENTAO
    ESCREVA (IDADE)
  FIMSE

Fimalgoritmo

```

Estrutura lógica

Comando Se-Senão

SE(<teste lógico>)

<comando 1>

<comando 2>

...

<comando n>

SENAO

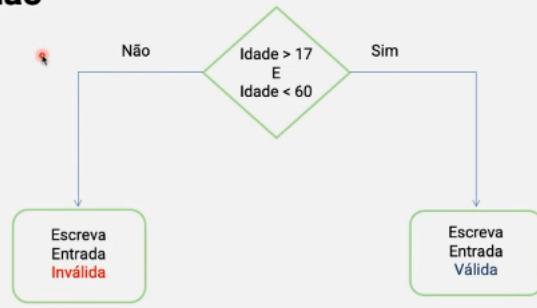
<comando 3>

<comando 4>

...

<comando n>

FIMSE



Aninhado

É usado quando temos que tirar mais de uma decisão uma dentro de outra.

Diagrama de exemplo:

Comandos Aninhados

SE(<teste lógico>)

<comando 1>

SENAO SE(<teste lógico>)

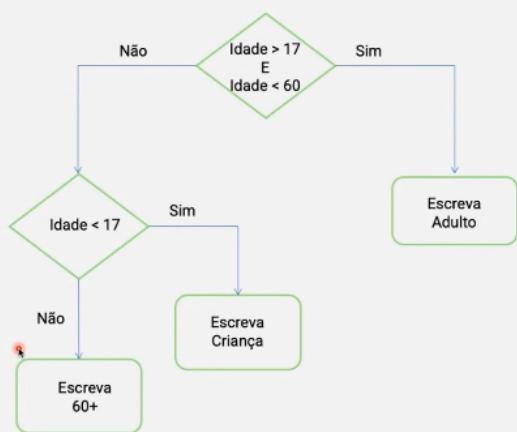
<comando 2>

<comando 3>

SENAO

<comando 3>

...



Exemplo em código:

```
Algoritmo "IDADESESENAOANI"
Var
  IDADE: inteiro
Inicio
  ESCREVA ("DIGITE A IDADE ")
  LEIA (IDADE)
  SE ((IDADE > 17) E (IDADE < 60)) ENTAC
    ESCREVA ("ADULTO")
  SENAO
    SE (IDADE < 17) ENTAO
      ESCREVA ("CRIANÇA")
    SENAO
      ESCREVA ("60+")
    FIMSE
  FIMSE
Fimalgoritmo
```

▼ Comandos de decisão - parte 2

Estruturas de **seleção múltipla** determinam o resultado a partir de uma seleção.

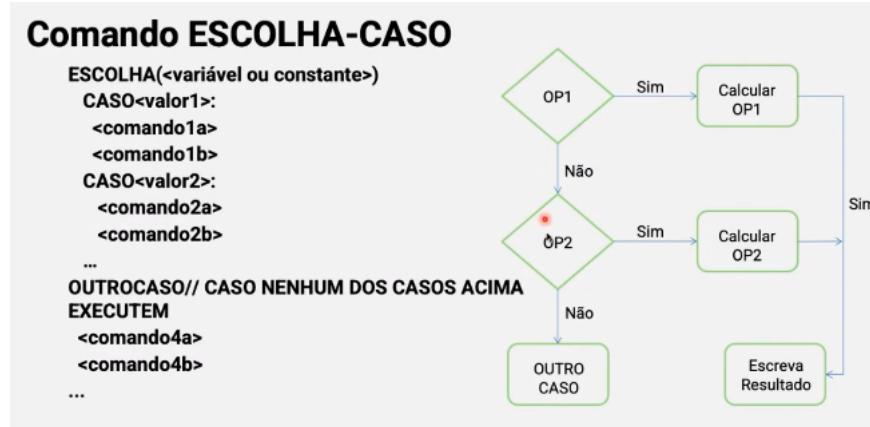
Um exemplo é o **escolha caso**.

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.
- Estrutura de Seleção

Escolha-caso

Diagrama de exemplo:



Exemplo de código:

```
Algoritmo "ESCOLHA"
Var
    // Seqüência de Declarações das variáveis
    OP: Inteiro
    Num, Resultado: Real;
Início
    // Seqüência de Comandos, procedimento, funções, operadores, etc...
    Escreval ("Opções:")
    Escreval ("1 - Calcular o dobro do número")
    Escreval ("2 - Calcular o triplo do número")
    Escreval ("Escolha uma opção: ")
    Leia (OP)
    Escreva ("Digite o número: ")
    Leia (Num)
    Escolha (OP)
        Caso 1
            Resultado <- Num*2
        Caso 2
            Resultado <- Num*3
    FimEscolha
    Escreval (Resultado)
Fimalgoritmo
```

▼ Exemplo e Atividade Prática 5

Vamos fazer um código que verifica qual o maior número dentre 2 números informados.

Exemplo de código:

```
Algoritmo "NUMMAIOR"
Var
    num1, num2: Real
Início
    Escreva ("Escreva um número: ")
    Leia (num1)
    Escreva ("Escreva outro número: ")
    Leia (num2)
    Se (num1 > num2) Então
        Escreva ("O primeiro número é o maior")
    FimSe
Fimalgoritmo
```

Resultado:

```
Escreva um número:  
2  
Escreva outro número: 1  
O primeiro número é o maior  
>>> Fim da execução do programa !
```

Enunciado do exercício:

Atividade Prática 5

Tomada de Decisão – parte 1 e 2 – SE e ESCOLHA/CASO

- A. Escreva um algoritmo em pseudocódigo que solicite ao usuário dois números e verifique se o primeiro número digitado é maior que o segundo número digitado, se verdadeira escreva a frase "O primeiro número é maior", senão escreva a frase "O segundo número é maior".
- B. Execute o código Exemplo B e mostre o resultado de uma escolha.

▼ Exemplo de Solução 5

Enunciado:

Atividade Prática 5

Tomada de Decisão – parte 1 e 2 – SE e ESCOLHA/CASO

- A. Escreva um algoritmo em pseudocódigo que solicite ao usuário dois números e verifique se o primeiro número digitado é maior que o segundo número digitado, se verdadeira escreva a frase "O primeiro número é maior", senão escreva a frase "O segundo número é maior".
- B. Execute o código Exemplo B e mostre o resultado de uma escolha.

Código de resposta da atividade A:

```
Algoritmo "VerificaNúmero"  
// Disciplina : [Linguagem e Lógica de Programação]  
// Professor  : Antonio Carlos Nicolodi
```

```
Var
```

```

// Seção de Declarações das variáveis
num1, num2: Real

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreval("Digite o primeiro número: ")
Leia(num1)
Escreval("Digite o segundo número: ")
Leia(num2)
Se(num1 > num2) Então
    Escreva("Número 1 é maior que o número 2!")
Senao
    Escreva("Número 2 é maior que número 1!")
FimSe

Fimalgoritmo

```

▼ Texto: Estruturas de Decisão

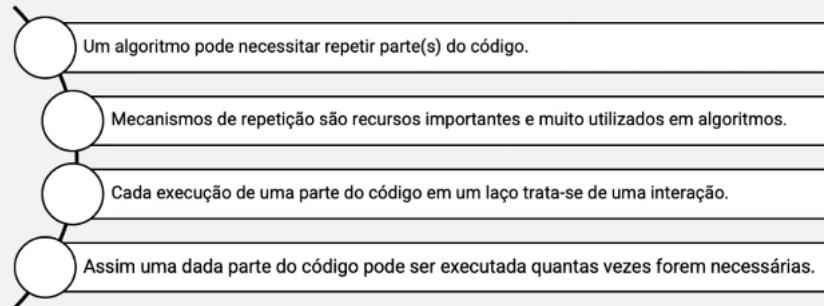
▼ PDF

[05_comandos_de_decisao.pdf](#)

Estruturas de Repetição

▼ Estruturas de Repetição - parte 1

Estruturas de Repetição (ou Laços)



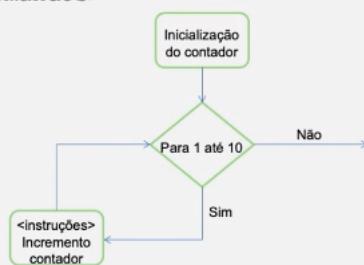
Estrutura PARA

- Estrutura **PARA** consiste 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>] faça  
    <sequência-de-comandos>  
fimpara
```

Comando PARA

```
para <variável> de <valor-inicial> ate <valor-limite> faça  
    <sequência-de-comandos>  
fimpara
```



Exemplo

```
Algoritmo "PARA1A10"  
  
Var  
// Seção de Declarações das variáveis  
    i:inteiro  
  
Inicio  
// Seção de Comandos, procedimento, funções, operadores, etc...  
    PARA i DE 1 ATÉ 10 FAÇA  
        ESCREVA (i)  
    FIMPARA  
  
Fimalgoritmo
```

Exemplo PARA - Aninhados

```
Algoritmo "PARA1A10Aninhado"
Var
// Seção de Declarações das variáveis
i,j:inteiro
•
Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
PARA i DE 1 ATÉ 10 FACA
    ESCREVAL ("Valor de I: ", i)
    PARA j DE 1 ATÉ 3 FACA
        ESCREVAL ("J: ", j)
    FIMPARA
FIMPARA

Finalgoritmo
```

▼ Estruturas de Repetição - parte 2

Estrutura de Repetição - ENQUANTO



A 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**.

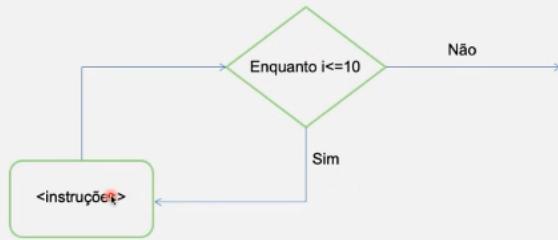


Não se conhece a priori o número de repetições que serão executadas no corpo do laço.



Comando ENQUANTO

ENQUANTO <expressão booleana> **FACA**
<sequência-de-comandos>
FIMENQUANTO



Exemplo

```
Algoritmo "ENQUANTO1A10"
Var
// Seção de Declarações das variáveis
i:inteiro
Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
i<-1
ENQUANTO i<=10 FAÇA
    ESCREVAL ("Valor de I: ", i)
    i<-i+1
FIMENQUANTO
Fimalgoritmo
```

Estrutura de Repetição - REPITA...ATÉ

- Semelhante a estrutura ENQUANTO;
- Permite repetir um trecho de código enquanto uma determinada condição for verdadeira;
- Diferença:

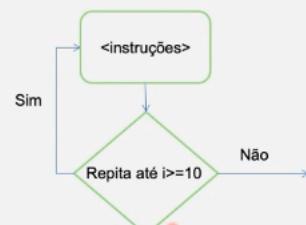
- Semelhante a estrutura ENQUANTO
- Permite repetir um trecho de código enquanto uma determinada condição for verdadeira.
- Diferença:

O bloco de comandos associado a estrutura de repetição é executado obrigatoriamente pelo menos uma vez.

Isso acontece porque a avaliação da condição é feita no final da estrutura de repetição REPITA...ATÉ.

Estrutura de Repetição - REPITA...ATÉ

REPITA
 <sequência-de-comandos>
ATE <expressão-lógica>



Exemplo

```
Algoritmo "REPITALA10"
Var
// Seção de Declarações das variáveis
i:inteiro

Início
// Seção de Comandos, procedimento, funções, operadores, etc...
i<-1
REPITA
    ESCREVAL ("Valor de I: ", i)
    i<-i+1
ATE i>=10

Fimalgoritmo
```

▼ Exemplo e Atividade Prática 6

Exemplo

```
Algoritmo "ENQUANTO1A5SOMA"
Var
// Seção de Declarações das variáveis
i,soma,num:inteiro

Início
// Seção de Comandos, procedimento, funções, operadores, etc...
i<-1
ENQUANTO i<=5 FAÇA
    ESCREVAL ("Digite um número do tipo inteiro para a SOMA: ")
    LEIA (num)
    soma<-soma+num
    i<-i+1
FIMENQUANTO
    ESCREVAL ("Resultado da SOMA: ", soma)
Fimalgoritmo
```

Atividade Prática 6

Tomada de Decisão – parte 1 e 2 – PARA e
ENQUANTO/REPITA...ATE

- A. Escreva um algoritmo em pseudocódigo que solicite ao usuário 5 números e calcule a multiplicação destes números digitados pelo usuário (utilize a estrutura de repetição PARA)
- B. Escreva um algoritmo em pseudocódigo que solicite ao usuário 5 números e calcule a multiplicação destes números digitados pelo usuário (utilize a estrutura de repetição REPITA...ATE)

▼ Solução de Exemplo da Atividade 6

Enunciado

- A. Escreva um algoritmo em pseudocódigo que solicite ao usuário 5 números e calcule a multiplicação destes números digitados pelo usuário (utilize a estrutura de repetição PARA)
- B. Escreva um algoritmo em pseudocódigo que solicite ao usuário 5 números e calcule a multiplicação destes números digitados pelo usuário (utilize a estrutura de repetição REPITA...ATE)

ATIVIDADE A

Solução atividade A em código:

```
Algoritmo "MultiplicarCincoNumeros"

Var
    // Seção de Declarações das variáveis
    numero, resultado: Inteiro
    contador: Inteiro

Inicio
    // Inicializa a variável resultado com 1
    resultado <- 1

    // Seção de Comandos, procedimento, funções, operadores, etc...
    PARA contador DE 1 ATÉ 5 FAÇA
        Escreva("Digite um número: ")
        Leia(numero)
        resultado <- resultado * numero

        SE (contador = 5) ENTAO
            Escreva("O resultado é: ", resultado)
        FIMSE
    FIMPARA
Fimalgoritmo
```

Solução atividade A print:

```

1 Algoritmo "MultiplicarCincoNumeros"
2
3 Var
4   // Seção de Declarações das variáveis
5   numero, resultado: Inteiro
6   contador: Inteiro
7
8 Inicio
9   // Inicializa a variável resultado com 1
10  resultado <- 1
11
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13 PARA contador DE 1 ATÉ 5 FAÇA
14   Escreva("Digite um número: ")
15   Leia(numero)
16   resultado <- resultado * numero
17
18   SE (contador = 5) ENTAO
19     Escreva("O resultado é: ", resultado)
20   FIMSE
21 FIMPARA
22 Fimalgoritmo
23

```

Console simulando o modo texto do MS-DOS

```

Digite um número: 1
Digite um número: 2
Digite um número: 3
Digite um número: 4
Digite um número: 5
O resultado é: 120
>>> Fim da execução do programa !

```

ATIVIDADE B

Resposta em código

```

Algoritmo "MultiplicarCincoNumeros"

Var
// Seção de Declarações das variáveis
numero, resultado: Inteiro
contador: Inteiro

Inicio
// Inicia contador com 0
contador <- 0
resultado <- 1

REPITA
// Pede numero ao usuario e repete...
Escreva("Digite um número: ")
Leia(numero)
// Multiplica o valor digitado pelo usuário pelo numero
resultado <- resultado * numero

contador <- contador + 1

SE (contador = 5) ENTAO
Escreva("O resultado é: ", resultado)

```

```
FIMSE
```

```
ATE contador = 5
```

```
Fimalgoritmo
```

Resultado em print:

```
1 Algoritmo "MultiplicarCincoNumeros"
2
3 Var
4 // Seção de Declarações das variáveis
5 numero, resultado: Inteiro
6 contador: Inteiro
7
8 Inicio
9 // Inicia contador com 0
10 contador <- 0
11 resultado <- 1
12
13 REPITA
14 // Pede numero ao usuário e repete...
15 Escreva("Digite um número: ")
16 Leia(numero)
17 // Multiplica o valor digitado pelo usuário pelo numero
18 resultado <- resultado * numero
19
20 contador <- contador + 1
21
22 SE (contador = 5) ENTAO
23   Escreva("O resultado é: ", resultado)
24 FIMSE
25
26 ATE contador = 5
27
28 Fimalgoritmo
```

```
Console simulando o modo texto do MS-DOS
```

```
Digite um número: 1
Digite um número: 2
Digite um número: 3
Digite um número: 4
Digite um número: 5
O resultado é: 120
>>> Fim da execução do programa !
```

▼ Texto: Estruturas de repetição

▼ PDF

[06_estruturas_de_repeticao.pdf](#)

Vetor

▼ Vetor/Array - parte 1

Vetor também é conhecido como **Estrutura Homogênea Unidimensional**.

Pode armazenar **diversas variáveis do mesmo tipo** onde cada item do vetor **pode ser acessado através de um índice**.

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.

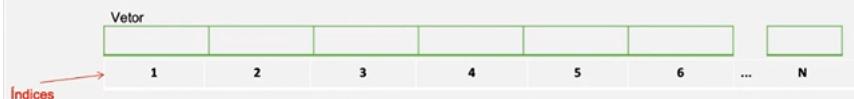
Vamos imaginar...

Imagine o seguinte problema:

Você precisa criar um algoritmo que lê o **nome de uma lista de alunos de uma turma**.

Vetor é uma variável que **permite armazenar várias variáveis do mesmo tipo**, permitindo dinamismo para situações como a mencionada.

- Uma variável que armazena várias variáveis do mesmo tipo.



Por exemplo:

Vetor

- No problema apresentado anteriormente, é possível utilizar um vetor de 50 posições para armazenar os nomes dos 50 alunos.

Vetor: Nome							
Indices	1	2	3	4	5	6	...
	João	Anna	Maria	Pedro	Gustavo	José	Barros

- Qual o nome de índice ou posição 5? **Gustavo**

▼ Vetor/Array - parte 2

Como declarar um vetor no algoritmo?

Vetor

- Declaração**

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

- Exemplo**

nomes: **vetor** [1..5] de caractere

Como usar um vetor?

Para usar um vetor precisamos usar uma estrutura de repetição

Vetor

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

```
PARA <variável> DE <valor-inicial> ATÉ <valor-limite> [passo <incremento>] FAÇA  
    <sequência-de-comandos para acessar um vetor>  
FIMPARA
```

Por que usar um vetor?

Para evitar declarar muitas variáveis e armazenar muitos dados do mesmo tipo separadamente

Por que utilizar o vetor?

```
ESCREVA("Digite o nome do aluno(a) número 1 de 5: ")
LEIA(nome1)
ESCREVA("Digite o nome do aluno(a) número 2 de 5: ")
LEIA(nome2)
ESCREVA("Digite o nome do aluno(a) número 3 de 5: ")
LEIA(nome3)
ESCREVA("Digite o nome do aluno(a) número 4 de 5: ")
LEIA(nome4)
ESCREVA("Digite o nome do aluno(a) número 5 de 5: ")
LEIA(nome5)
```

Exemplo de uso...

Comando VETOR - Exemplo

```
algoritmo "VETORNOME"
var
    nomes: vetor [1..5] de caractere
    contadorLoop1: inteiro
inicio
    //Leitura dos nomes de cada aluno
    PARA contadorLoop1 DE 1 ATÉ 5 FAÇA
        ESCREVA("Digite o nome do aluno(a) número ", contadorLoop1, " de 5: ")
        LEIA(nomes[contadorLoop1])
    FIMPARA
    //APRESENTAÇÃO DOS RESULTADOS
    PARA contadorLoop1 DE 1 ATÉ 5 FAÇA
        ESCREVAL("Nome do aluno(a) ", nomes[contadorLoop1])
    FIMPARA
fimalgoritmo
```

▼ Exemplo e Atividade Prática 7

- Escreva um algoritmo em portugol que solicite ao usuário 5 nomes e 5 notas e calcule a média das notas da turma (armazenar as notas em um vetor).

▼ Solução de Exemplo da Atividade 7

Resposta

```
Algoritmo "VETORMEDIA5NOTAS"
// Disciplina : [Linguagem e Lógica de Programação]
// Autor(a)   : Marlon Prado
// Data atual  : 06/06/2024
Var
    // Seção de Declarações das variáveis
    notas: vetor [1..5] de real
    contador: inteiro
    media: real
    soma: real
```

```

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...

soma <- 0

para contador de 1 ate 5 faca
    Escreva("Digite a nota de nº", contador, " de 5 do aluno: ")
    Leia(notas[contador])
    soma <- soma + notas[contador]
fimpara

para contador de 1 ate 5 faca
    Escreval("A ", contador, "a nota foi: ", notas[contador])
fimpara

media <- soma / 5

Escreval("A média foi de: ", media)

Fimalgoritmo

```

▼ Texto: Vetor/Array

▼ PDF

[07_vetor_e_array.pdf](#)

Matriz

▼ Matriz - parte 1

Matrizes são estruturas homogêneas, assim como vetores. Mas a diferença é que são multidimensionais.

Ou seja, um vetor de vetores.

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.



Matrizes são representadas por MxN, onde:

- M → dimensão horizontal
- N → dimensão vertical

Matriz

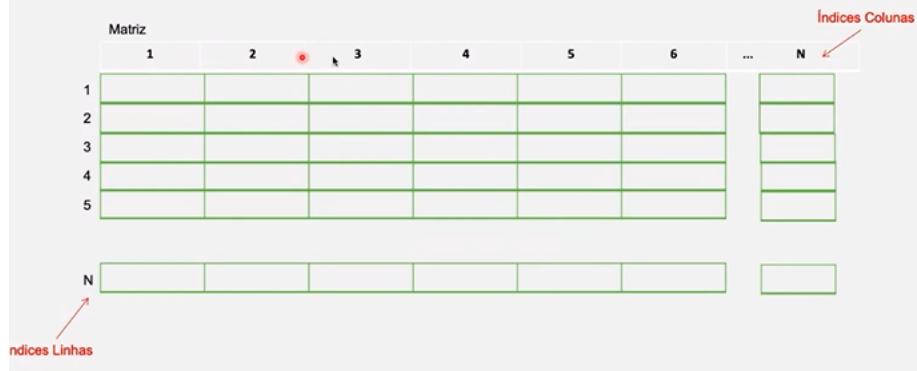
- As matrizes são, comumente referenciadas através de suas dimensões (quantidade de linhas e colunas).
- A notação comum é: $M \times N$, onde
 - M é a dimensão horizontal (quantidade de linhas).
 - N é dimensão vertical (quantidade de colunas).



Vetores: a quantidade de linhas é sempre 1!

Como são representadas:

Estrutura de uma Matriz



Imagine o problema abaixo...

Imagine o seguinte problema:

Você precisa criar um algoritmo que lê e armazena as notas parciais, calcula e armazena a média e informa o resultado.

Desta forma a solução seria:

Estrutura de uma Matriz - Notas

Matriz			
	1	2	3
1	9	7,5	8
2	5	5	8
3	6	8	8,5
4	10	9,5	10
5	0	7	5

Índices Linhas

Índices Colunas

Qual é o valor da variável de linha 50 e coluna 4 [50,4]?

▼ Matriz - parte 1

Para declarar matrizes é similar a declarar vetores.

• 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

Como matrizes são compostas de linhas e colunas, para acessar e trabalhar com elas devemos usar 2 estruturas de repetição:

Matriz

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

```
PARA <variável> DE <valor-inicial> ATÉ <valor-final> [passo <incremento>] FACA  
    PARA <variável> DE <valor-inicial> ATÉ <valor-final> [passo <incremento>] FACA  
        <sequência-de-comandos para acessar um vetor>  
    FIMPARA  
FIMPARA
```

Passar pelas LINHAS

Passar pelas COLUNAS

Exemplo de uso de matriz:

Exemplo:

```

PARA contador i DE 1 ATE 50 FACA
    ESCREVA("Aluno(a) número ", i)
    PARA contador j DE 1 ATE 4 FACA
        ESCREVA("Digite a nota: ", j)
        LEIA(notas[i , j])
    FIMPARA
FIMPARA

```

Comando Matriz- Exemplo

```

Algoritmo "exemploMatriz"
    Var
        // Seção de Declarações das variáveis
        numeros: vetor [1..3,1..2] de inteiro
        i:inteiro

    Inicio
        // Seção de Comandos, procedimento, funções, operadores, etc...
        // Laço para percorrer as linhas
        PARA i DE 1 ATE 3 FACA
            ESCREVA("Digite o valor para a posicao ", i, ", 1: ")
            LEIA(numeros[i, 1])
            ESCREVA("Digite o valor para a posicao ", i, ", 2: ")
            LEIA(numeros[i, 2])
        FIMPARA
Fimalgoritmo

```

Nesse exemplo não tem dois comandos PARA pois fixamos o valor do índice da coluna [1] e [2].

▼ Exemplo e Atividade Prática 8

```

Algoritmo "exemploMatriz"
    Var
        // Seção de Declarações das variáveis
        numeros: vetor [1..3,1..2] de inteiro
        i,j:inteiro

    Inicio
        // Seção de Comandos, procedimento, funções, operadores, etc...
        // Laço para percorrer as linhas
        PARA i DE 1 ATE 3 FACA
            // Laço para percorrer as colunas
            PARA j DE 1 ATE 2 FACA
                ESCREVA("Digite o valor para a linha ", i, " e coluna ", j, ": ")
                LEIA(numeros[i, j])
            FIMPARA
        FIMPARA
Fimalgoritmo

```

Utilizando dois laços – PARA i linha e PARA j coluna.

Matriz

- Vamos ver como fica este resultado armazenado na matriz:

```

Digite o valor para a linha 1 e coluna 1: 5
Digite o valor para a linha 1 e coluna 2: 6
Digite o valor para a linha 2 e coluna 1: 3
Digite o valor para a linha 2 e coluna 2: 8
Digite o valor para a linha 3 e coluna 1: 6
Digite o valor para a linha 3 e coluna 2: 5

>>> Fim da execução do programa !

```

		(índice)
		1 2
	1	5 6
	2	3 8
	3	6 5

Atividade prática:

Atividade Prática 8

Matriz

- Com base no exemplo anterior, desenvolva um algoritmo que some cada valor de uma posição da matriz com 2. Mostre o resultado na tela.
 - Dica1: soma=soma + 2
 - Dica2: Use mais dois laços para mostrar na tela o resultado.

▼ Solução de Exemplo da Atividade 8

Minha resposta

```
Algoritmo "ATIVIDADE_8_MATRIZ"
// Disciplina : [Linguagem e Lógica de Programação]
// Autor(a)   : Marlon Prado
// Data atual  : 07/06/2024
Var
// Seção de Declarações das variáveis

matriz: vetor[1..2,1..2] de inteiro
i, j: inteiro

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
    i <- 0

    para i de 1 ate 2 faca
        para j de 1 ate 2 faca
            Escreva("Digite o valor da linha ", i, " e coluna ", j, ": ")
            Leia(matriz[i,j])
        fimpara
    fimpara

    para i de 1 ate 2 faca
        Escreval("")
        para j de 1 ate 2 faca
            Escreva(matriz[i,j] + 2)
        fimpara
    fimpara

Fimalgoritmo
```

Resposta do professor:

Exemplo

```

Algoritmo "exemploMatriz3"
  Var
    // Seção de Declarações das variáveis
    numeros: vetor [1..3,1..2] de inteiro
    i,j:inteiro
    soma:inteiro
  
```

Laços para adicionar valores na matriz → Variáveis

```

  Início
    // Seção de Comandos, procedimento, funções, operadores, etc...
    soma <-1
    // Laço para percorrer as linhas
    PARA i DE 1 ATÉ 3 FAÇA
      // Laço para percorrer as colunas
      PARA j DE 1 ATÉ 2 FAÇA
        soma<-soma+2
        numeros[i, j]<- soma
      FIMPARA
    FIMPARA
    // Laço para percorrer as linhas
    PARA i DE 1 ATÉ 3 FAÇA
      // Laço para percorrer as colunas
      PARA j DE 1 ATÉ 2 FAÇA
        ESCREVAL("O valor para a linha ", i, " e coluna ", j,": ")
        ESCREVAL(numeros[i, j])
      FIMPARA
    FIMPARA
  
```

Laços para mostrar valores armazenados na matriz →

```

  Fimalgoritmo

```

Resposta

(índice)

```

O valor para a linha 1 e coluna 1:
3
O valor para a linha 1 e coluna 2:
5
O valor para a linha 2 e coluna 1:
7
O valor para a linha 2 e coluna 2:
9
O valor para a linha 3 e coluna 1:
11
O valor para a linha 3 e coluna 2:
13
>>> Fim da execução do programa !

```

	1	2
1	3	5
2	7	9
3	11	13

▼ Texto: Matriz

▼ PDF

[08_matriz.pdf](#)

Procedimentos sem Parâmetros

▼ Modularização Procedimento

Modularização é a separação de responsabilidades, tornando problemas grandes em problemas menores mais alcançáveis.

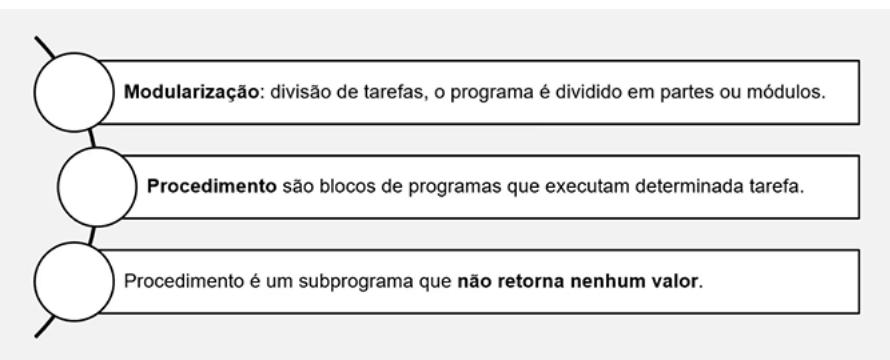
Existem 2 tipos de modularização:

- procedimento ou função
- com e 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
- Exemplo visita

- **Modularização** são divisões de **tarefas**, divide o **programa em partes ou módulos**.
- **Procedimentos** são blocos de programas que **executam alguma tarefa e NÃO retornam valor** nenhum.



▼ Procedimentos sem parâmetros

Procedimentos sem Parâmetros

• Estrutura

```
PROCEDIMENTO <nome-de-procedimento>
VAR
    // Seção de Declarações Variáveis do Procedimento

    INICIO
        // Seção de Comandos

    FIMPROCEDIMENTO
```

Exemplo de estrutura

```

• Exemplo
PROCEDIMENTO ExemploProcedimento
VAR
// 

INICIO
ESCREVA ("Sou como a estrutura sequencial")
FIMPROCEDIMENTO

```

Exemplo real de algoritmo

Procedimento - Exemplo

Algoritmo nome e declarações de variáveis	<code>algoritmo "ExemploProcedimento" var</code>
Procedimento Início e Fim	<code>procedimento mostreNaTela var inicio ESCREVA ("Meu primeiro procedimento") fimprocedimento</code>
Algoritmo e Chamada do Procedimento	<code>inicio ESCREVAL ("Mensagem do procedimento: ") mostreNaTela fimalgoritmo</code>

▼ Exemplo e Atividade Prática 9

Procedimento Início e Fim	<code>Algoritmo "ProcedimentoExemplo" procedimento soma var aux: inteiro inicio aux <- n + m res <- aux fimprocedimento</code>
Algoritmo e Chamada do Procedimento	<code>var res, n, m: inteiro Início n <- 4 m <- -9 soma escreva(res) Fimalgoritmo</code>

Enunciado atividade 9:

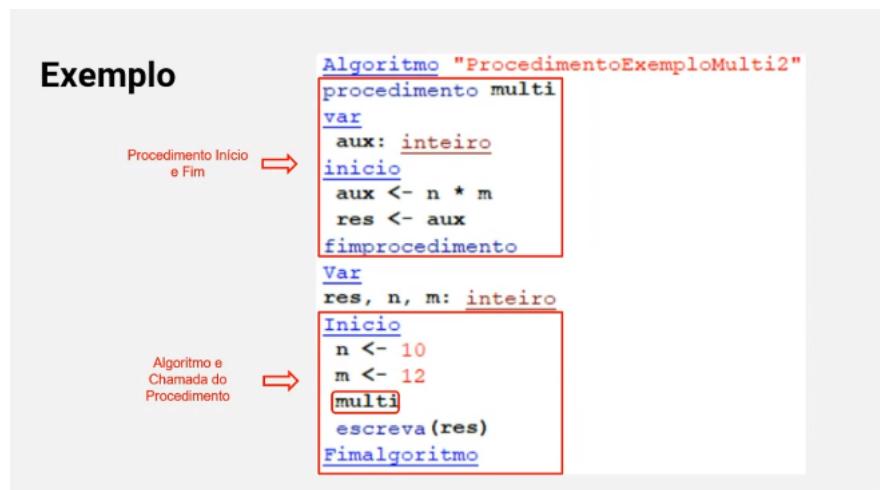
Atividade Prática 9

Procedimentos

- Com base no exemplo anterior, desenvolva um algoritmo que:
 - tenha um procedimento que receba dois números e calcula o seu produto (o resultado da multiplicação dos dois números)
 - o procedimento mostra o resultado do produto na tela
 - faça a chamada adequada do procedimento

▼ Solução de Exemplo da Atividade 9

Exemplo de resultado:



Minha resposta:

```
Algoritmo "ATIVIDADE_9_PROCEDIMENTO"
// Disciplina : [Linguagem e Lógica de Programação]
// Autor(a)   : Marlon Prado
// Data atual  : 07/06/2024
procedimento multiplicar

var
    produto: real

    inicio
        produto <- val1 * val2
        Escreva("O resultado de ", val1, " x", val2, " é: ", produto)
    fimprocedimento

    Var
```

```

val1: real
val2: real

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...

Escreva("Digite o primeiro valor: ")
Leia(val1)
Escreva("Digite o segundo valor: ")
Leia(val2)

multiplicar

Fimalgoritmo

```

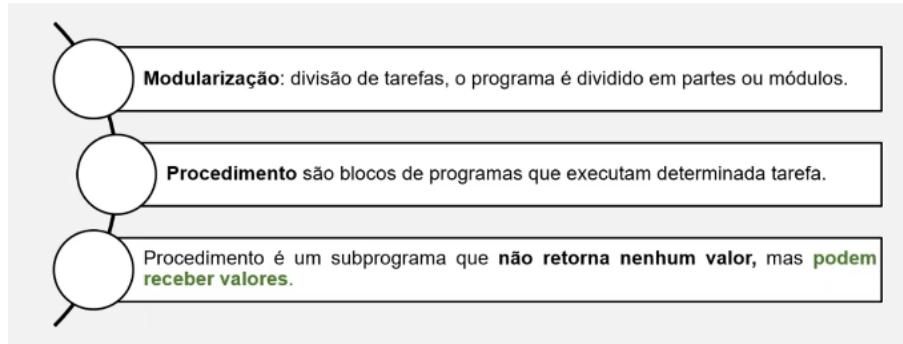
▼ Texto: Procedimentos sem Parâmetros

▼ PDF

[09_procedimentos_sem_parametros.pdf](#)

Procedimentos com Parâmetros

▼ Procedimentos com e sem Parâmetros



▼ Procedimentos com Parâmetros

- **Estrutura**
- ```

PROCEDIMENTO <nome-de-procedimento> [(<sequência-de-declarações-de-parâmetros>)]
// Seção de Declarações Internas

INICIO
// Seção de Comandos

FIMPROCEDIMENTO
* a <sequência-de-declarações-de-parâmetros> é uma sequência de [var] <sequência-de-parâmetros>: <tipo-de-dado> separadas por ponto e vírgula.

```

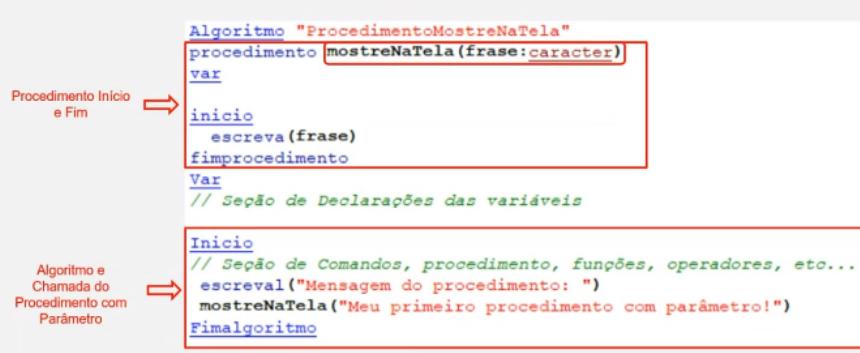
• Exemplo

```
PROCEDIMENTOS ProcComParametro(frase:caracter)
VAR
```

```
INICIO
```

```
 ESCREVA (frase)
```

```
FIMPROCEDIMENTO
```



▼ Exemplo e Atividade Prática 10

**Exemplo**

Inicio do algoritmo

Procedimento Início e Fim

Algoritmo e Chamada do Procedimento com Parâmetro

Algoritmo "ProcedimentoExemploParametro"

procedimento soma(x,y:inteiro)

var

aux: inteiro

inicio

aux <- x + y

res <- aux

fimprocedimento

Var

// Seção de Declarações das variáveis

res, n, m: inteiro

Variáveis Globais

Início

// Seção de Comandos, procedimento, funções, operadores, etc...

n <- 4

m <- -9

soma(n,m)

escreva(res)

Fimalgoritmo

▼ Solução de Exemplo da Atividade 10

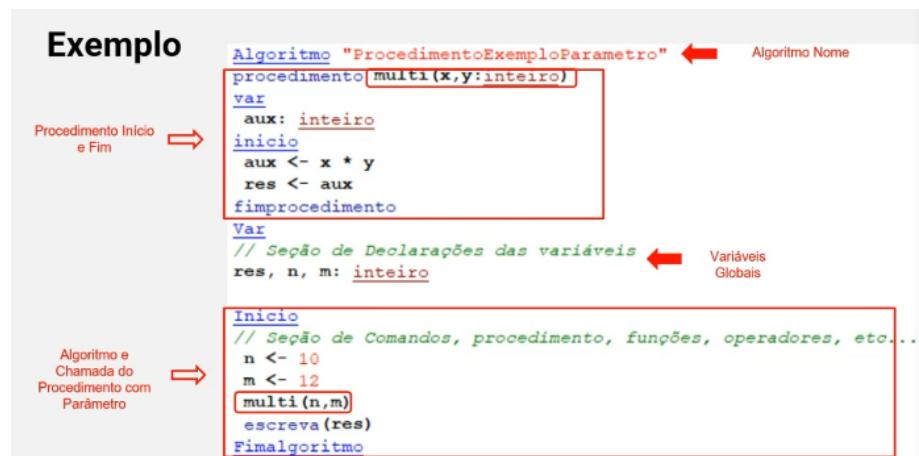
Enunciado:

**Atividade Prática 10**

Procedimentos

- Com base no exemplo anterior, desenvolva um algoritmo que:
  - tenha um procedimento que receba dois números por parâmetro e calcula o seu produto (o resultado da multiplicação dos dois números)
  - o procedimento mostra o resultado do produto na tela
  - faça a chamada adequada do procedimento

Resposta da professora:



Minha resposta:

```

Algoritmo "ATIVIDADE_10_PROCEDIMENTO_COM_PARAMETROS"
// Disciplina : [Linguagem e Lógica de Programação]
// Autor(a) : Marlon Prado
// Data atual : 10/06/2024
procedimento multiplicar(n1,n2:real)

var

produto: real

inicio

produto <- n1 * n2
Escreva("O resultado de ", n1, " x", n2, " é: ", produto)

fimprocedimento

Var

val1: real
val2: real

Início
// Seção de Comandos, procedimento, funções, operadores, etc...

Escreva("Digite o primeiro valor: ")
Leia(val1)
Escreva("Digite o segundo valor: ")
Leia(val2)

multiplicar(val1, val2)

Fimalgoritmo

```

#### ▼ Texto: Procedimentos com Parâmetros

▼ PDF

## Funções sem parâmetros

### ▼ Modularização Função

#### 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
- Exemplo visita

#### 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:** divisão de tarefas, divide-se o programa em partes ou módulos.
- **Funções:** são blocos de programas que executam determinada tarefa.
- **Diferença**

**Procedimento** é um subprograma que  
não retorna nenhum valor.

**Função** é um subprograma que retorna  
valores.

### ▼ Funções sem parâmetros

## Funções sem Parâmetros

### • Estrutura

```
FUNCAO <nome-de-função> : <tipo-de-dado>
VAR
 // Seção de Declarações Internas

INICIO
 // Seção de Comandos
 RETORNE <valor>
FIMFUNCAO
```

## Funções

### • Exemplo

```
FUNCAO olaFunction : caracter
VAR
 frase : caracter
INICIO
 frase <- "Exemplo Function!"
 RETORNE frase
FIMFUNCAO
```

## Funções - Exemplo

Algoritmo nome → Algoritmo "FuncaoMostreNaTela"  
funcao mostreNaTela:caracter  
var  
 frase:caracter  
início  
 frase <- "Minha primeira função!"  
 retorno frase  
fimfuncao

Função com retorno → // Seção de Declarações das variáveis

Variáveis Globais →

Algoritmo chamando a função e recebendo o retorno → Inicio  
// Seção de Comandos, procedimento, funções, operadores, etc...
 escreval("Mensagem da função: ")
 escreval(mostreNaTela)
Fimalgoritmo

### ▼ Exemplo e Atividade Prática 11

## Exemplo

Função com retorno

```
Algoritmo "FuncaoExemplo"
funcao soma: inteiro
var aux: inteiro
inicio
// n, m e res são variáveis globais
aux <- n + m
retorne aux
fimfuncao
```

Algoritmo chamando a função  
e recebendo o retorno

```
var
n,m:inteiro
res:inteiro
inicio
n <- 4
m <- -9
res <- soma
escreva(res)
Finalgoritmo
```

## Atividade Prática 11

Funções

- Com base no exemplo anterior, desenvolva um algoritmo que:
  - Tenha uma função que receba dois números e calcule o seu produto (o resultado da multiplicação de dois números)
  - A função deve retornar o resultado do produto
  - Faça a chamada adequada da função
  - Mostre o resultado na tela.

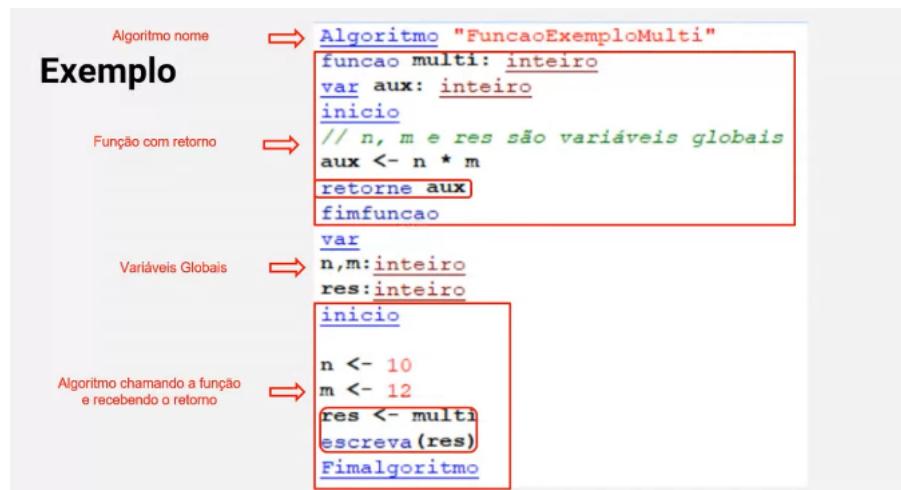
### ▼ Solução de Exemplo da Atividade 11

## Atividade Prática 11

Funções

- Com base no exemplo anterior, desenvolva um algoritmo que:
  - Tenha uma função que receba dois números e calcule o seu produto (o resultado da multiplicação de dois números)
  - A função deve retornar o resultado do produto
  - Faça a chamada adequada da função
  - Mostre o resultado na tela.

Exemplo:



### ▼ Texto: Funções sem parâmetros

▼ PDF

[11\\_funcoes\\_sem\\_parametros.pdf](#)

## Funções com parâmetros

### ▼ Funções com e sem parâmetros

### Função com e sem Parâmetros

- O que são argumentos passados por parâmetro
- Quando usar parâmetro
- Exemplo Visita

## Funções



**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** e **podem receber valores**.

## Funções

- **Modularização:** divisão de tarefas, divide-se o programa em partes ou módulos.
- **Funções** são blocos de programas que executam determinada tarefa.

### ▼ Funções com parâmetros

#### 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
```

## Funções

- Exemplo

```
FUNCAO olaFunction(texto:caracter):caracter
VAR

INICIO
 frase <- texto
RETORNE frase
FIMFUNCAO
```

## Função - Exemplo

Algoritmo nome → Algoritmo "FuncaoMostreNaTelaParam"

Função que recebe e retorna valor →

```
funcao mostreNaTela(texto:caracter):caracter
var
 frase:caracter
inicio
 frase <- texto
 retorno frase
fimfuncao

Var
// Seção de Declarações das variáveis
 res:caracter
```

Variáveis Globais ←

Chamada e retorno da função →

```
Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
 escreval("Mensagem da função: ")
 res <- mostreNaTela("Minha primeira função com parâmetro!")
 escreva(res)
Fimalgoritmo
```

## ▼ Exemplo e Atividade Prática 12

### Exemplo

Algoritmo nome → Algoritmo "FuncaoSomaParam"

Função que recebe e retorna valor →

```
funcao soma (x,y: inteiro): inteiro
```

```
inicio
 retorno x + y
fimfuncao
```

Variáveis Globais →

```
var
```

```
 n,m,res:inteiro
```

Chamada e retorno da função →

```
inicio
```

```
 n <- 4
```

```
 m <- -9
```

```
 res <- soma(n,m)
```

```
 escreva(res)
```

```
Fimalgoritmo
```

## Atividade Prática 12

Funções

- Com base no exemplo anterior, desenvolva um algoritmo que:
  - Tenha uma função que recebe dois números por parâmetro e calcula e retorne o seu produto (o resultado da multiplicação dos dois números).
  - A função deve retornar o produto
  - Faça a chamada adequada da função
  - Mostre o resultado na tela

---

### ▼ Solução de Exemplo da Atividade 12

## Atividade Prática 12

Funções

- Com base no exemplo anterior, desenvolva um algoritmo que:
  - Tenha uma função que recebe dois números por parâmetro e calcula e retorne o seu produto (o resultado da multiplicação dos dois números).
  - A função deve retornar o produto
  - Faça a chamada adequada da função
  - Mostre o resultado na tela

---

### Exemplo

Algoritmo nome

→ Algoritmo "FuncaoMultiParam"

Função que recebe  
e retorna valor

```
funcao multi (x,y: inteiro): inteiro
 inicio
 retorno x * y
 fimfuncao
```

Variáveis Globais

→ var

Chamada e retorno  
da função

→ n,m,res:inteiro

```
inicio
 n <- 10
 m <- 12
 res <- multi (n,m)
 escreva (res)
fimalgoritmo
```



---

### ▼ Texto: Funções com parâmetros

▼ PDF

## Visão Geral da Linguagem Java

### ▼ Ambiente Java e IDE

#### A linguagem Java



Linguagem de programação que tem sua própria estrutura, regras de sintaxe e paradigma de programação.

Deriva da linguagem C, portanto suas regras de sintaxe assemelham-se às regras de C.

**Exemplo:** os blocos de códigos são modularizados em métodos e delimitados por chaves ({ e }) e variáveis são declaradas antes que sejam usadas.

#### O que é preciso para programar em Java?

- Será necessário instalar:
  - O **JDK** (*Kit de desenvolvimento Java*)  
<https://www.oracle.com/java/technologies/javase-downloads.html>.
  - O **Bloco de Notas**
  - O **CMD**

## O compilador Java

Programa Java  
(Código-fonte em arquivos .java)

O compilador transforma os arquivos .java em arquivos .class

Bytecode é um conjunto de instruções que são executadas em uma Java virtual Machine (JVM).

## Java Virtual Machine - JVM

No tempo de execução, a JVM lê e interpreta arquivos .class e executa as instruções do programa.

A JVM interpreta o bytecode.

A JVM é o núcleo do princípio "gravação única, execução em qualquer local" da linguagem Java.

## Java Development Kit - JDK

- **Contém:**

- compilador e de outras ferramentas.
- biblioteca de classe completa de utilitários que o ajuda a realizar tarefas de desenvolvimento de aplicativo mais comuns.

## IDE

- Eclipse
- IntelliJ
- NetBeans
- JDeveloper
- MyEclipse
- BlueJ



Fonte: www.edureka.co

### ▼ Meu primeiro programa Java

#### Exemplo

Ambiente de Programação

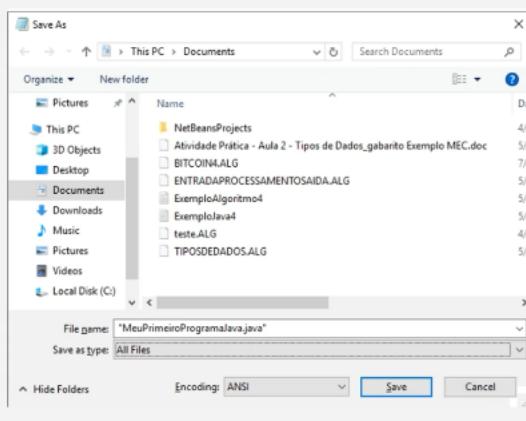
```
MeuPrimeiroProgramalava - Notepad
File Edit Format View Help
//salvar como MeuPrimeiroProgramaJava.java

//nome da classe
class MeuPrimeiroProgramaJava
{
 //módulo principal com a entrada pela linha de comando
 public static void main (String entrada[])
 {
 //declaração de variáveis
 int inteiro = 47;
 char caracter = 'F';
 double real = 1.65;
 String frase = "Lucy Mari ";
 boolean VF=true;

 if (VF == true)
 {
 System.out.println("Eu sou a " + frase + " tenho " + inteiro + " anos e tenho " + real + "m de altura");
 }

 System.exit(0);
 }
}
```

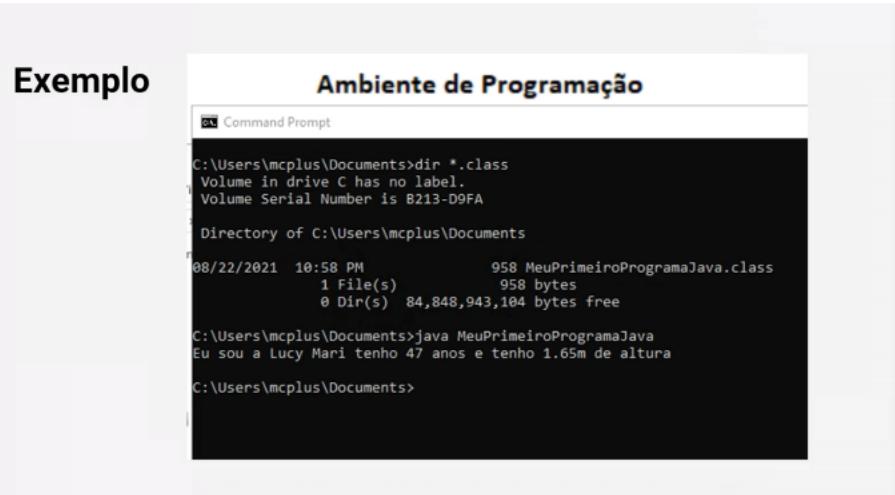
#### Exemplo



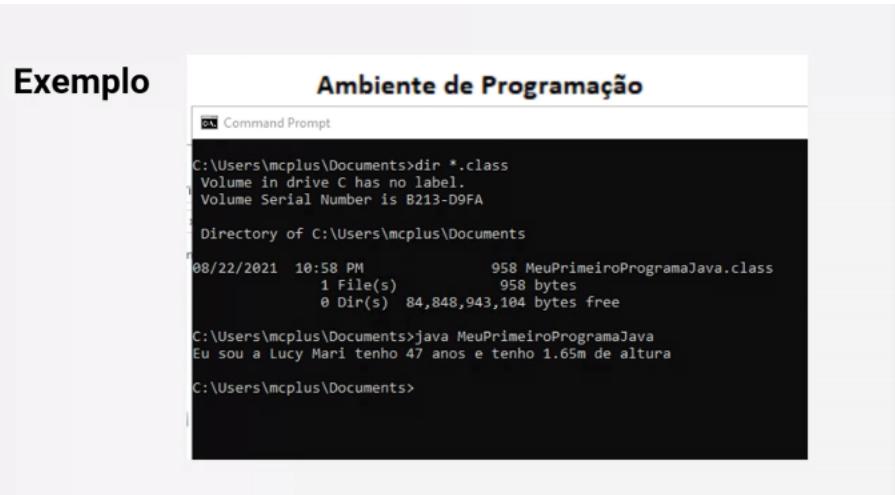
## Exemplo



## Exemplo



## Exemplo



▼ Exemplo e Atividade Prática 13

## Exemplo

```
Ambiente de Programação
MeuSegundoProgramaJava - Notepad
File Edit Format View Help
//salvar como MeuPrimeiroProgramaJava.java

//nome da classe
class MeuSegundoProgramaJava
{
 //módulo principal com a entrada pela linha de comando
 public static void main (String entrada[])
 {
 //declaração de variáveis
 int n1, n2, soma;

 //entrada de dados
 n1 = Integer.parseInt(entrada[0]);
 n2 = Integer.parseInt(entrada[1]);
 //processamento
 soma = n1 + n2;
 //saída de resultados
 System.out.println(n1 + " + " + n2 + " = " + soma);
 System.exit(0);
 }
}
```

## Exemplo

```
Ambiente de Programação
Command Prompt
C:\Users\mcplus\Documents>javac MeuSegundoProgramaJava.java
C:\Users\mcplus\Documents>java MeuSegundoProgramaJava 10 12
10 + 12 = 22
C:\Users\mcplus\Documents>
```

## Atividade Prática 13

- Com base no exemplo anterior, desenvolva um programa em Java que escreva na tela:
  - Seu nome completo em String
  - Seu curso em String
  - Sua idade em int
  - Seu gênero em char
  - Seu peso em double

### ▼ Solução de Exemplo da Atividade 13

## Exemplo

```
//nome do programa
//salvar como MeuPrimeiroExercicioJava.java

//nome da classe
class MeuPrimeiroExercicioJava
{
 //módulo principal com a entrada pela linha de
 comando
 public static void main (String entrada[])
 {
 //declaração de variáveis
 int idade = 47;
 char genero = 'F';
 double peso = 50.5;
 String nome = "Lucy Mari ";
 String curso = "Ciéncia da Computação ";

 System.out.println("Eu sou a " + nome + "tenho "
 + idade + " anos e tenho " + peso + "kg de
 peso e curso " + curso);

 System.exit(0);
 }
}
```

## Exemplo

### Ambiente de Programação

```
cmd Command Prompt
C:\Users\mcplus\Documents>javac MeuPrimeiroExercicioJava.java
C:\Users\mcplus\Documents>java MeuPrimeiroExercicioJava
Eu sou a Lucy Mari tenho 47 anos e tenho 50.5kg de peso e curso Ciéncia da Computa?o
C:\Users\mcplus\Documents>
```

## ▼ Texto: Visão Geral da Linguagem Java

### ▼ PDF

[13\\_visao\\_geral\\_da\\_linguagem\\_java.pdf](#)

## Introdução à Linguagem Java

### ▼ Trabalhando com dados no Java

## Exemplo

- Desenvolvimento de um programa Java que:
  - Declara variáveis: inteira, real e caracter;
  - Recebe essas informações pela linha de comando;
  - Calcula a adição dos dois números;
  - Mostra essas informações pela linha de comando.

## Exemplo

```
1 //salvar como Programa01.java
2
3 class Programa01
4 {
5 public static void main (String entrada[])
6 {
7 //declaração de variáveis
8 int NumInt;
9 double NumReal, soma;
10 char Caracter;
11
12 //entrada de dados
13 NumInt = Integer.parseInt(entrada[0]);
14 NumReal = Double.parseDouble(entrada[1]);
15 Caracter = (entrada[2]).charAt(0);
16 //processamento
17 soma = (double)NumInt + NumReal;
18 //saída de resultados
19 System.out.println((double)NumInt + " + " +
20 NumReal + " = " + soma + " sinal " + Caracter);
21
22 System.exit(0);
23 }
}
```

## ▼ Fluxo de dados no Java

## Exemplo

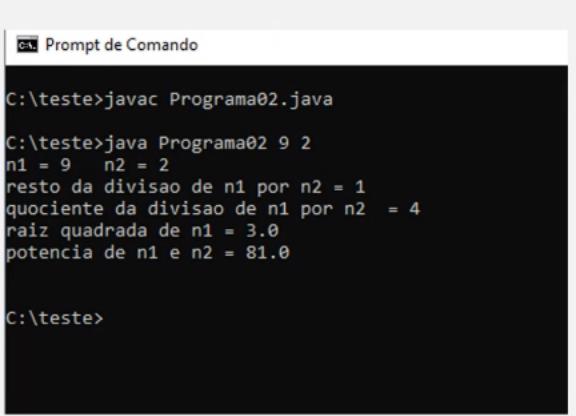
- Desenvolvimento de um programa Java que:
  - Declara variáveis: inteira, real e String
  - Recebe dois números pela linha de comando
  - Calcula o resto da divisão dos dois números, o quociente da divisão dos dois números, a raiz quadrada do primeiro número e a potência do primeiro para o segundo número
  - Mostra essas informações pela linha de comando

**Exemplo**

```

1 //salvar como Programa02.java
2 class Programa02
3 {
4 public static void main (String entrada[])
5 {
6 int n1, n2;
7 int mod, div;
8 double raiz, pot;
9 String msg = "";
10
11 //entrada de dados
12 n1 = Integer.parseInt(entrada[0]);
13 n2 = Integer.parseInt(entrada[1]);
14 //processamento
15 mod = n1 % n2;
16 div = (int)n1 / (int)n2;
17 raiz = Math.sqrt(n1);
18 pot = Math.pow(n1, n2);
19 //saída de resultados
20 msg = "n1 = " + n1 + " n2 = " + n2 + "\n";
21 msg = msg + "resto da divisão de n1 por n2 = " + mod + "\n";
22 msg = msg + "quociente da divisão de n1 por n2 = " + div + "\n";
23 msg = msg + "raiz quadrada de n1 = " + raiz + "\n";
24 msg = msg + "potência de n1 e n2 = " + pot + "\n";
25 System.out.println(msg);
26 System.exit(0);
27 }
28 }
```

**Exemplo**



The terminal window shows the following command and its execution:

```
C:\teste>javac Programa02.java
C:\teste>java Programa02 9 2
n1 = 9 n2 = 2
resto da divisão de n1 por n2 = 1
quociente da divisão de n1 por n2 = 4
raiz quadrada de n1 = 3.0
potência de n1 e n2 = 81.0
```

▼ Exemplo e Atividade Prática 14

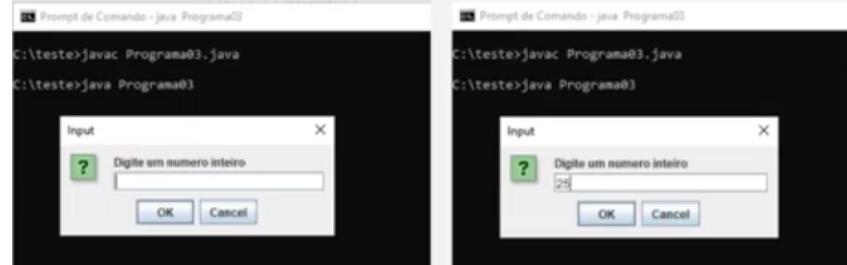
**Exemplo**

- Desenvolvimento de um programa Java que:
  - Declara variáveis: inteira, real e String
  - Recebe dois números inteiros usando interface com usuário (swing)
  - Calcula a resto da divisão dos dois números, a raiz quadrada do primeiro número e do segundo número
  - Mostra essas informações usando interface com usuário (swing)

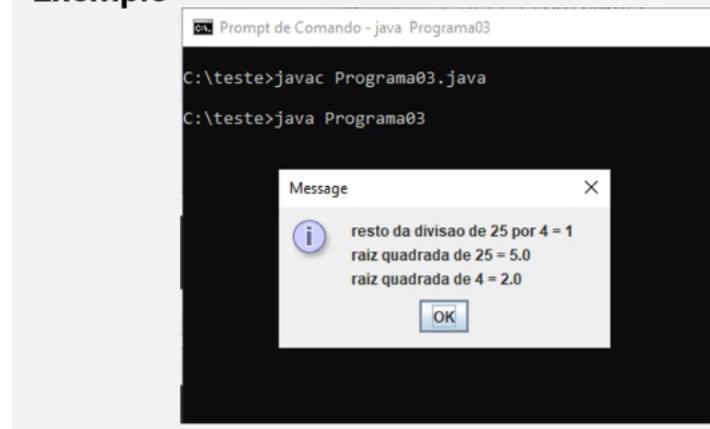
## Exemplo

```
1 //salvar como Programa03.java
2 import javax.swing.*;
3 |
4 class Programa03
5 {
6 public static void main (String entrada[])
7 {
8 //declaração de variáveis
9 int n1, n2, mod;
10 double raiz1, raiz2;
11 String msg="";
12 //entrada de dados
13 n1 = Integer.parseInt(JOptionPane.showInputDialog("Digite um numero inteiro"));
14 n2 = Integer.parseInt(JOptionPane.showInputDialog("Digite outro numero inteiro"));
15 //processamento
16 mod = n1 % n2;
17 raiz1 = Math.sqrt(n1);
18 raiz2 = Math.sqrt(n2);
19
20 //saída de resultados
21 msg = msg + "resto da divisão de " + n1 + " por " + n2 + " = " + mod + "\n";
22 msg = msg + "raiz quadrada de " + n1 + " = " + raiz1 + "\n";
23 msg = msg + "raiz quadrada de " + n2 + " = " + raiz2 + "\n";
24 JOptionPane.showMessageDialog(null, msg);
25
26 System.exit(0);
27 }
28}
```

## Exemplo



## Exemplo



## Atividade Prática 14

- Com base no exemplo anterior, desenvolva de um programa Java que:
  - Declara variáveis: inteira, real e String
  - Recebe dois números inteiros usando interface com usuário (swing)
  - Calcula a quociente da divisão dos dois números, a potência do primeiro número pelo segundo número
  - Mostra essas informações usando interface com usuário (swing)

### ▼ Solução de Exemplo da Atividade 14

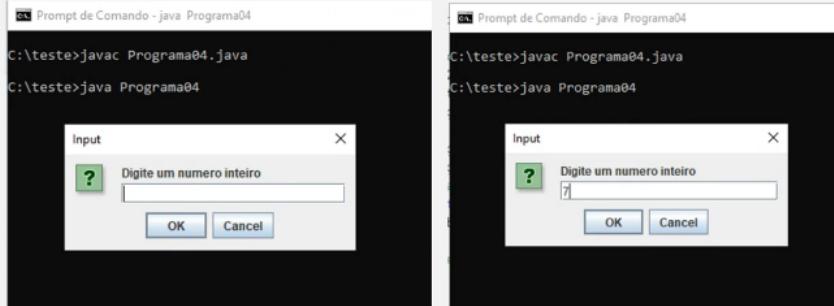
## Atividade Prática 14

- **Com base no exemplo anterior, desenvolva de um programa Java que:**
  - Declara variáveis: inteira, real e String
  - Recebe dois números inteiros usando interface com usuário (swing)
  - Calcula a quociente da divisão dos dois números, a potência do primeiro número pelo segundo número
  - Mostra essas informações usando interface com usuário (swing)

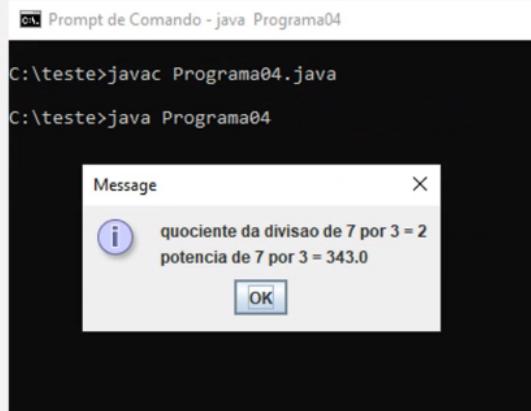
### Exemplo

```
1 //salvar como Programa04.java
2 import javax.swing.*;
3
4 class Programa04
5 {
6 public static void main (String entrada[])
7 {
8 //declaração de variáveis
9 int n1, n2, div;
10 double pot;
11 String msg="";
12 //entrada de dados
13 n1 = Integer.parseInt(JOptionPane.showInputDialog("Digite um numero inteiro"));
14 n2 = Integer.parseInt(JOptionPane.showInputDialog("Digite outro numero inteiro"));
15 //processamento
16 div = (int)n1 / (int)n2;
17 pot = Math.pow(n1,n2);
18 //saída de resultados
19 msg = msg + "quociente da divisao de " + n1 + " por " + n2 + " = " + div + "\n";
20 msg = msg + "potencia de " + n1 + " por " + n2 + " = " + pot + "\n";
21 JOptionPane.showMessageDialog(null, msg);
22
23 System.exit(0);
24 }
25 }
26 }
```

## Exemplo



## Exemplo



### ▼ Atividade prática

```
import javax.swing.*;

public class Main {

 public static void main(String[] argumentos) {
 // Declarando variáveis
 int num1, num2;
 double quociente, potencia;

 // Recebendo os dois números inteiros usando interface com usuário
 num1 = Integer.parseInt(JOptionPane.showInputDialog("Digite o primeiro número inteiro"));
 num2 = Integer.parseInt(JOptionPane.showInputDialog("Digite o segundo número inteiro"));

 // Calculando o quociente da divisão dos dois números
 quociente = num1 % num2;

 // Calculando a potência do primeiro número pelo segundo número
 potencia = Math.pow(num1, num2);

 // Mostrando as informações usando interface com usuário
 String message = "Quociente da divisão de " + num1 + " por " + num2 + " é: " + quociente;
 JOptionPane.showMessageDialog(null, message);
 }
}
```

```
 }
}
```

## ▼ Texto: Introdução à Linguagem Java

▼ PDF

[14\\_introducao\\_a\\_linguagem\\_java.pdf](#)

## Estrutura de Controle no Java

### ▼ Estrutura de Decisão no Java

#### Exemplo

- Desenvolvimento de um programa Java que:
  - Declara variáveis: inteira, caracter e String
  - Recebe uma opção e um número inteiro
  - Calcula se o número é par ou ímpar, positivo ou não positivo e apresenta apenas a opção selecionada
  - Mostra essas informações

#### Exemplo

```
1 //salvar como ProgDecisao.java
2 import javax.swing.*;
3 |
4 class ProgDecisao
5 {
6 public static void main (String entrada[])
7 {
8 int num;
9 char op=0;
10 String msg="", msgEntr="Digite 1 para par/impar\nDigite 2 para positivo/negativo";
11 //entrada de dados
12 num = Integer.parseInt(JOptionPane.showInputDialog("Digite um numero inteiro"));
13 op = (JOptionPane.showInputDialog(msgEntr)).charAt(0);
14 //processamento
15
16 switch(op)
17 {
18 //saida de resultados
19 if (op == '1' || op == '2')
20 {
21 JOptionPane.showMessageDialog(null, msg);
22 }
23 System.exit(0);
24 }
25 }
26}
```

### Exemplo

```

16
17 switch(op)
18 {
19 case '+':
20 {
21 if(num % 2 == 0)
22 {
23 msg = msg + num + " eh par.\n";
24 }
25 else
26 {
27 msg = msg + num + " eh impar.\n";
28 }
29 break;
30 }
31 case '-':
32 {
33 if(num > 0)
34 {
35 msg = msg + num + " eh positivo.\n";
36 }
37 else
38 {
39 msg = msg + num + " eh nao positivo.\n";
40 }
41 break;
42 }
43 default: JOptionPane.showMessageDialog(null,"Opcao invalida, calculos nao realizados.");
44 }

```

### Exemplo

## ▼ Estruturas de Repetição no Java

### Exemplo

- Desenvolvimento de um programa Java que:
  - Declara variáveis: int, char e String
  - Recebe um número para calcular a tabuada desse número e uma opção por qual tipo de repetição calcular a tabuada
  - Calcula a tabuada pelas repetições for, while e do/while
  - Mostra a tabuada calculada pela opção escolhida

## Exemplo

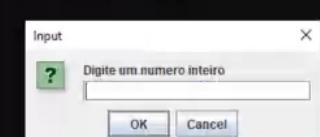
```
1 //salvar como ProgRepeticao.java
2 import javax.swing.*;
3
4 class ProgRepeticao
5 {
6 public static void main (String entrada[])
7 {
8 int Tabuada;
9 char op=0;
10 String msg="", msgEntr="Digite 1 repeticao
11 for\nDigite 2 para repeticao while\nDigite 3 para
12 repeticao do/while";
13 //entrada de dados
14 Tabuada = Integer.parseInt(JOptionPane.
15 showInputDialog("Digite um numero inteiro"));
16 op = (JOptionPane.showInputDialog(msgEntr)).charAt(0);
17 //processamento
18
19 switch(op)
20 {
21 //saida de resultados
22 if (op >='1' && op <='3')
23 {
24 JOptionPane.showMessageDialog(null, msg);
25 }
26 System.exit(0);
27 }
28 }
29}
30
```

## Exemplo

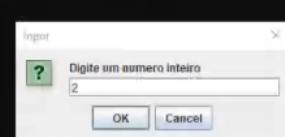
```
16 switch(op)
17 {
18 case '1':
19 {
20 msg = msg + "Tabuada do " + Tabuada + " pelo for: \n\n";
21 for(int i=1; i<=10; i+=1)
22 {
23 msg = msg + Tabuada + " x " + i + " = " + Tabuada*i + "\n";
24 }
25 break;
26 }
27 case '2':
28 {
29 msg = msg + "Tabuada do " + Tabuada + " pelo while: \n\n";
30 int i = 1;
31 while(i<=10)
32 {
33 msg = msg + Tabuada + " x " + i + " = " + Tabuada*i + "\n";
34 i+=1;
35 }
36 break;
37 }
38 case '3':
39 {
40 msg = msg + "Tabuada do " + Tabuada + " pelo do/while: \n\n";
41 int i = 1;
42 do
43 {
44 msg = msg + Tabuada + " x " + i + " = " + Tabuada*i + "\n";
45 i+=1;
46 } while (i<=10);
47 break;
48 }
49 default: JOptionPane.showMessageDialog(null,"Opcao invalida, calculos nao realizados");
50 }
```

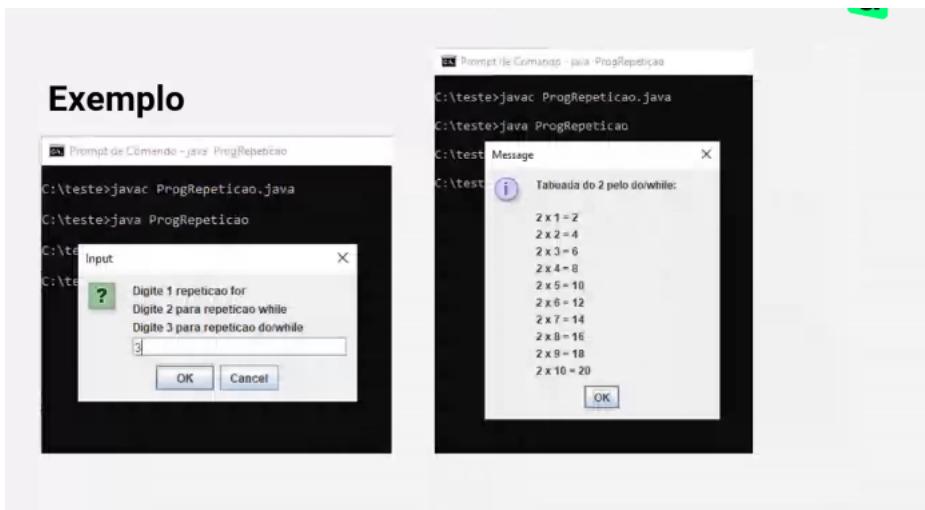
## Exemplo

C:\Prompt de Comando - java ProgRepeticao  
C:\teste>javac ProgRepeticao.java  
C:\teste>java ProgRepeticao

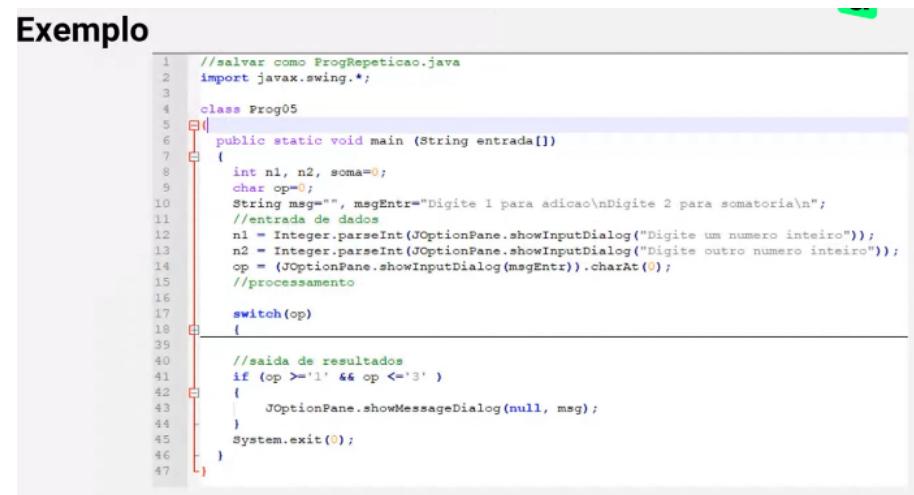
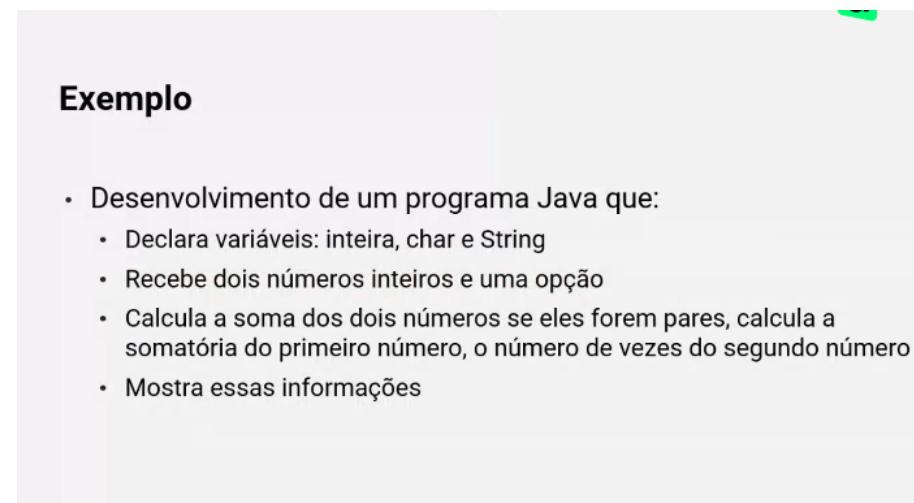


C:\Prompt de Comando - java ProgRepeticao  
C:\teste>javac ProgRepeticao  
C:\teste>java ProgRepeticao





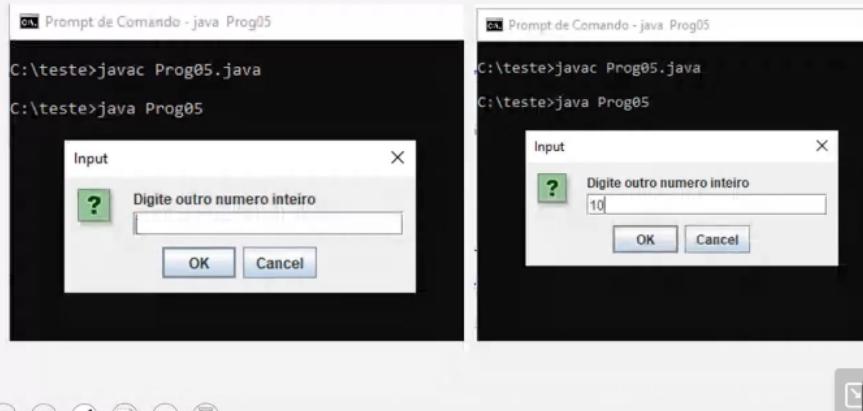
## ▼ Exemplo e Atividade Prática 15



## Exemplo

```
17 switch(op)
18 {
19 case '1':
20 {
21 if (n1%2==0 && n2%2==0)
22 {
23 soma = n1 + n2;
24 msg = msg + "Soma de " + n1 + " por " + n2 + " = " + soma + "\n\n";
25 }
26 break;
27 }
28 case '2':
29 {
30 for(int i=1; i<=n2; i=i+1)
31 {
32 soma = soma + n1;
33 }
34 msg = msg + "Somatoria de " + n1 + ", " + n2 + " vezes eh: " + soma + "\n\n";
35 break;
36 }
37 default: JOptionPane.showMessageDialog(null,"Opção invalida, cálculos não realizados");
38 }
```

## Exemplo



### ▼ Solução de Exemplo da Atividade 15

## Atividade Prática 15

- Desenvolvimento de um programa Java que:
  - Declara variáveis: int, char e String
  - Recebe dois números inteiros e uma opção
  - Calcula o produto dos dois números se eles forem positivos, calcula a produtória do primeiro número, o número de vezes do segundo
  - Mostra essas informações

## Exemplo

```
1 //salvar como Prog06.java
2 import javax.swing.*;
3
4 class Prog06
5 {
6 public static void main (String entrada[])
7 {
8 int n1, n2, p=1;
9 char op=0;
10 String msg="", msgEntr="Digite 1 para produto\nDigite 2 para produtoria\n";
11 //entrada de dados
12 n1 = Integer.parseInt(JOptionPane.showInputDialog("Digite um numero inteiro"));
13 n2 = Integer.parseInt(JOptionPane.showInputDialog("Digite outro numero inteiro"));
14 op = (JOptionPane.showInputDialog(msgEntr)).charAt(0);
15 //processamento
16
17 switch(op)
18 {
19
20 //saída de resultados
21 if (op >='1' && op <='3')
22 {
23 JOptionPane.showMessageDialog(null, msg);
24 }
25 System.exit(0);
26 }
27 }
28
29
30
31
32
33
34
35
36
37
38 }
```

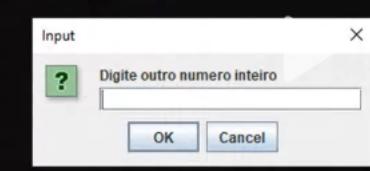
## Exemplo

```
17
18 switch(op)
19 {
20 case '1':
21 {
22 if (n1>0 && n1<0)
23 {
24 p = n1 * n2;
25 msg = msg + "Produto de " + n1 + " por " + n2 + " = " + p + "\n\n";
26 }
27 break;
28 case '2':
29 {
30 for(int i=1; i<=n2; i=i+1)
31 {
32 p = p * n1;
33 }
34 msg = msg + "Produtoria de " + n1 + ", " + n2 + " vezes eh: " + p + "\n\n";
35 break;
36 }
37 default: JOptionPane.showMessageDialog(null,"Opcão invalida, calculos nao realizados");
38 }
```

## Exemplo

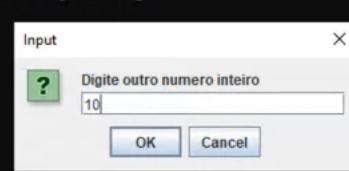
C:\teste>javac Prog06.java

C:\teste>java Prog06



C:\teste>javac Prog06.java

C:\teste>java Prog06



## ▼ Texto: Estrutura de Controle no Java

### ▼ PDF

[15\\_java\\_iii\\_estrutura\\_de\\_controle\\_no\\_java.pdf](#)

## Módulos e matrizes em Java

### ▼ Matrizes em Java

#### Exemplo

- Desenvolvimento de um programa Java que:
  - Declara variáveis: uma matriz unidimensional (vetor) de 5 posições e uma matriz 2x3 inteiros
  - Coloca ou recebe informações nas matrizes
  - Mostra essas informações

#### Exemplo

```
1 //salvar como ProgMatriz.java
2 import javax.swing.*;
3
4 class ProgMatriz
5 {
6 public static void main (String entrada[])
7 {
8 int vetor[] = {2, 4, 6, 8, 10};
9 int matriz[][] = new int [3][3];
10 String msg = "vetor = ";
11
12 //vetor =
13 for (int i = 0 ; i < vetor.length ; i++)
14 {
15 msg = msg + vetor[i] + " ";
16 }
17 JOptionPane.showMessageDialog(null, msg);
18
19 msg = "Matriz = \n";
20 for (int i = 0 ; i < matriz.length ; i++)
21 {
22 for (int j = 0 ; j < matriz[i].length ; j++)
23 {
24 matriz[i][j] = Integer.parseInt(JOptionPane.showInputDialog("Digite um inteiro para posicao " + i + " e " + j));
25 msg = msg + matriz[i][j] + " ";
26 }
27 msg = msg + "\n";
28 }
29
30 JOptionPane.showMessageDialog(null, msg);
31 System.exit(0);
32 }
}
```

#### Exemplo

C:\ Prompt de Comando - java ProgMatriz

C:\teste>javac ProgMatriz.java

C:\teste>java ProgMatriz



## ▼ Modularização no Java

### Exemplo

- Desenvolvimento de um programa Java que:
  - Declara variáveis: inteiros e reais
  - Recebe dois números inteiros
  - Calcula a soma, o produto, a diferença e o resultado da divisão de dois inteiros usando modularização
  - Mostra essas informações

### Exemplo

```
1 //salvar como ProgMod.java
2 import javax.swing.*;
3
4 class ProgMod
5 {
6 public static void soma ()
7 {
8 public static void subtracao (int x, int y)
9 {
10 public static int produto ()
11 {
12 public static double divisao (int x, int y)
13 {
14 public static void main (String entrada[])
15 {
16 int n1, n2, s;
17 double r;
18
19 soma ();
20 n1 = Integer.parseInt(JOptionPane.showInputDialog("Digite um numero"));
21 n2 = Integer.parseInt(JOptionPane.showInputDialog("Digite outro numero"));
22 subtracao(n1,n2);
23 s = produto();
24 JOptionPane.showMessageDialog(null, "O produto eh "+ s);
25 r = divisao(n1,n2);
26 JOptionPane.showMessageDialog(null, "A divisao eh "+ r);
27 System.exit(0);
28 }
29 }
30 }
31 }
32 }
33
34
35
36
37
38
39
40 }
```

### Exemplo

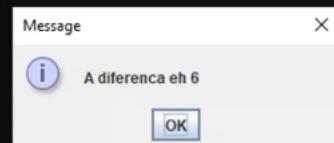
```
6 public static void soma ()
7 {
8 int n1, n2;
9 n1 = Integer.parseInt(JOptionPane.showInputDialog("Digite um numero"));
10 n2 = Integer.parseInt(JOptionPane.showInputDialog("Digite outro numero"));
11
12 JOptionPane.showMessageDialog(null, "A soma eh "+ (n1+n2));
13 }
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40 }
```

## Exemplo

```
C:\teste>javac ProgMod.java
C:\teste>java ProgMod
```



```
C:\teste>javac ProgMod.java
C:\teste>java ProgMod
```



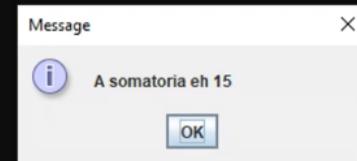
## ▼ Exemplo e Atividade Prática 16

### Exemplo

```
1 //salvar como Prog07.java
2 class Prog07
3 {
4 public static void soma ()
5 {
6 int s=0, vet[] = {1, 2, 3, 4, 5};
7
8 for (int i = 0; i<vet.length; i++)
9 {
10 s = s + vet[i];
11 }
12
13 JOptionPane.showMessageDialog(null, "A somatoria eh "+ s);
14 }
15
16 public static int produto ()
17 {
18 int p=1, vet[] = {1, 2, 3, 4, 5};
19
20 for (int i = 0; i<vet.length; i++)
21 {
22 p = p * vet[i];
23 }
24
25 return p;
26 }
27
28 public static void main (String entrada[])
29 {
30 int r;
31
32 soma ();
33 r = produto ();
34 JOptionPane.showMessageDialog(null, "A produtoria eh "+ r);
35 System.exit(0);
36 }
37 }
```

## Exemplo

```
C:\teste>javac Prog07.java
C:\teste>java Prog07
```



## Atividade Prática 16

- Desenvolvimento de um programa Java que:
  - Declara variáveis: vetor de inteiros
  - Recebe números inteiros num vetor de 5 posições
  - Calcula a somatória e a produtória desses números com um procedimento e uma função, ambos com parâmetros
  - Mostra essas informações usando

### ▼ Solução de Exemplo da Atividade 16

## Atividade Prática 16

- Desenvolvimento de um programa Java que:
  - Declara variáveis: vetor de inteiros
  - Recebe números inteiros num vetor de 5 posições
  - Calcula a somatória e a produtória desses números com um procedimento e uma função, ambos com parâmetros
  - Mostra essas informações usando

## Exemplo

```
1 //salvar como Prog06.java
2 import javax.swing.*;
3 class Prog06
4 {
5 public static void soma (int vet[])
6 {
7 int s=0;
8 for (int i = 0; i<vet.length; i++)
9 {
10 s = s + vet[i];
11 }
12 JOptionPane.showMessageDialog(null, "A somatoria eh "+ s);
13 }
14
15 public static int produto (int vet[])
16 {
17 int p=1;
18
19 for (int i = 0; i<vet.length; i++)
20 {
21 p = p * vet[i];
22 }
23
24 return p;
25 }
26
27
28 public static void main (String entrada[])
29 {
30 int s=0, vetor[] = {2, 4, 6, 8, 10};
31 int r;
32
33 soma(vetor);
34 r = produto(vetor);
35 JOptionPane.showMessageDialog(null, "A produtoria eh "+ r);
36 System.exit(0);
37 }
}
```

## Exemplo

The screenshot shows a Windows Command Prompt window titled "Prompt de Comando - java Prog08". It contains the following text:  
C:\teste>javac Prog08.java  
C:\teste>java Prog08  
A message dialog box titled "Message" is displayed, containing the text "A somatoria eh 30" and an "OK" button.

### ▼ Texto: Módulos e matrizes em Java

▼ PDF

[16\\_modulos\\_e\\_matrizes.pdf](#)

### ▼ Materiais de apoio e referências

▼ Ementa da disciplina

[Ementa - Lógica de Programação.pdf](#)