



EJA IV – Ensino Fundamental  
**Qualificação Profissional – Informática Básica**  
**Análise e Lógica da Programação**

# Algoritmo: Formas de representação e refinamentos sucessivos

Rildo Oliveira



05/03/2024

## ROTEIRO DE AULA

### **OBJETO DO CONHECIMENTO:**

Algoritmo: Formas de representação e refinamentos sucessivos

**HABILIDADE:** (EMIFFTP02) Levantar e testar hipóteses para resolver problemas do cotidiano pessoal, da escola e do trabalho, utilizando procedimentos e linguagens adequados à investigação científica.

### **OBJETIVOS:**

Apresentar as diferentes formas que podem ser representados um algoritmo, fluxograma e pseudocódigo;

Realizar exercícios que exemplifica como construir código.

**DA TEORIA À PRÁTICA:** Uso de imagens, texto e conceitos para um melhor entendimento do tema abordado.

# Pasta Compartilhada EJA



<https://github.com/rildexter/eja2024/tree/main>

# Conceito de Algoritmo

Um algoritmo é uma sequência de instruções lógicas que resolve um problema ou executa uma tarefa.

Exemplo em Portugal:

```
algoritmo exemplo_algoritmo  
  escreva("Digite um número:")  
  leia(numero)  
  se numero > 0 entao  
    escreva("O número é positivo.")  
  senao  
    escreva("O número é negativo ou zero.")  
  fimse  
fimalgoritmo
```

# Passos para Construção de um Algoritmo

Identifique o problema.

Divida o problema em etapas menores.

Escreva os passos em ordem lógica.

Revise e teste o algoritmo.

# Passos para Construção de um Algoritmo

Exemplo em Portugol:

Este algoritmo calcula a média de duas notas.

```
algoritmo calculo_media  
  escreva("Digite a primeira nota:")  
  leia(nota1)  
  escreva("Digite a segunda nota:")  
  leia(nota2)  
  media <- (nota1 + nota2) / 2  
  escreva("A média é: ", media)  
finalgoritmo
```

# Pseudocódigo

O pseudocódigo é uma descrição detalhada dos passos de um algoritmo.

Exemplo:

```
Início
  Leia um número
  Se o número for positivo
    Escreva "Número positivo"
  Senão
    Escreva "Número negativo"
  Fim Se
Fim
```

# Fluxograma

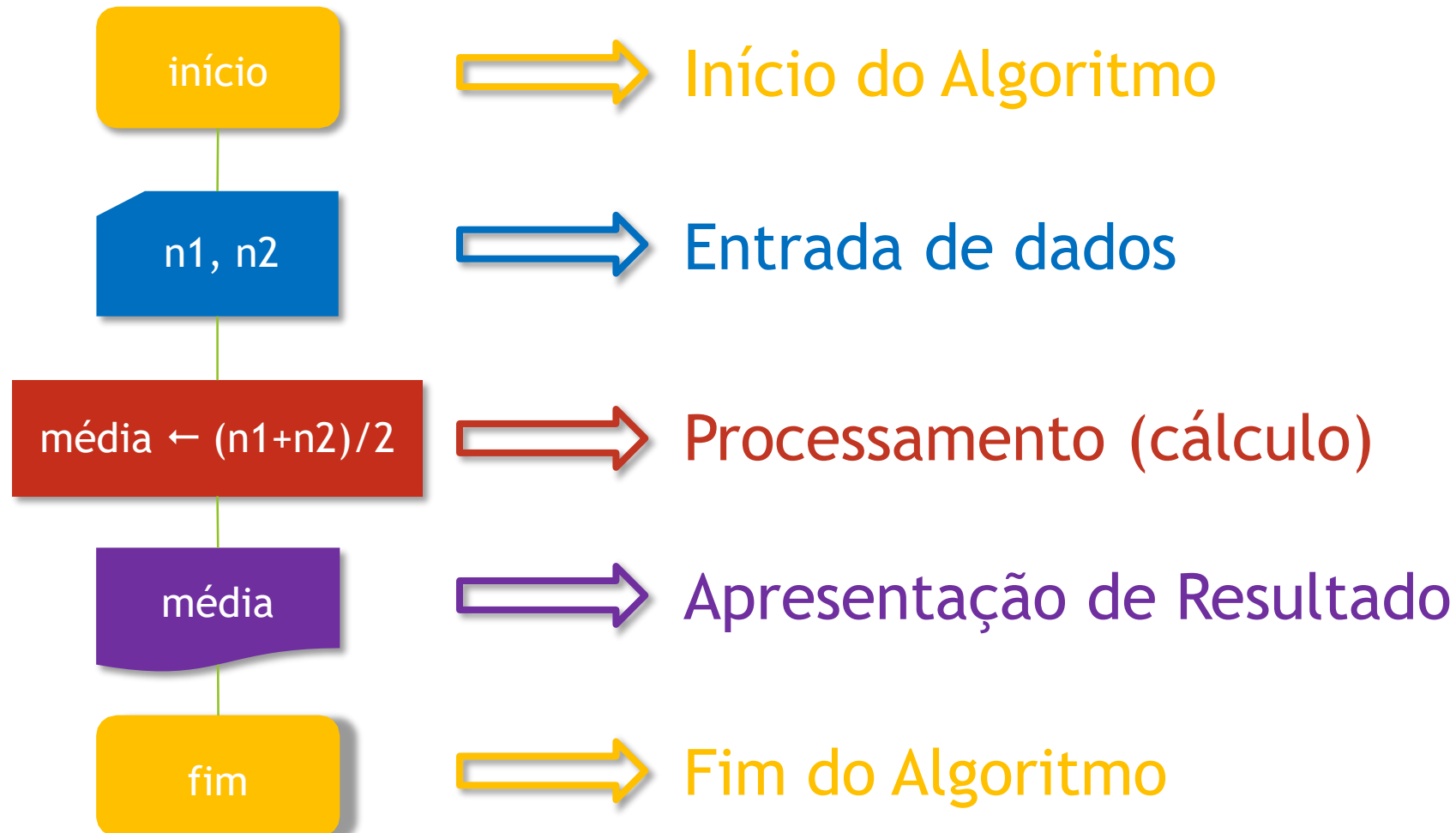
## Características

- ▶ A representação gráfica é mais concisa que a representação textual, portanto, é necessário aprender a simbologia dos fluxogramas;
- ▶ Representação gráfica por meio de símbolos geométricos, são:





## Exemplo: Algoritmo para calcular média



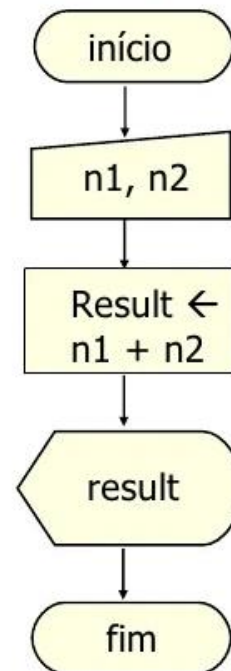
## Algoritmo - Exemplo

- × **Elaborar o algoritmo que faça a soma de dois números inteiros quaisquer:**

### Pseudocódigo

```
algoritmo somadoisnum  
  var n1, n2, result: inteiro  
  
início  
  leia n1  
  leia n2  
  result  $\leftarrow$  n1 + n2  
  escreva result  
  
fim
```

### Fluxograma



### Linguagem de Programação

```
program somadoisnum;  
  var n1, n2, result: integer;  
  
begin  
  readln(n1);  
  readln(n2);  
  result := n1 + n2;  
  writeln (result);  
  
end.
```

# Tipos de Dados

Os tipos de dados são essenciais na programação. Eles determinam como os valores são armazenados e manipulados.

Alguns tipos de dados comuns:

**Numérico Inteiro:** Variável numérica sem parte decimal.

**Numérico Real:** Variável numérica com parte decimal.

**Texto (Caractere):** Representa letras, números ou símbolos.

**Cadeia de Caracteres (String):** Conjunto de caracteres organizados consecutivamente.

**Lógico (Booleano):** Pode ser verdadeiro ou falso.

# Tipos de Dados

Exemplo em  
Portugol:

```
algoritmo tipos_de_dados
    inteiro idade
    real altura
    caractere inicial_nome
    logico aprovado
    escreva("Digite sua idade:")
    leia(idade)
    escreva("Digite sua altura:")
    leia(altura)
    escreva("Digite a inicial do seu nome:")
    leia(inicial_nome)
    escreva("Você foi aprovado? (true/false):")
    leia(aprovado)
finalgoritmo
```

# Memória em Computador

A memória do computador armazena dados e instruções.

Tipos de memória: RAM, ROM, cache, disco rígido, etc.

A memória é organizada em endereços.

Exemplo: A variável `idade` está armazenada em um endereço específico na memória.

# Variáveis e Constantes

Variáveis: Espaços de memória que armazenam valores mutáveis.

Constantes: Valores fixos que não podem ser alterados.

Exemplo:

```
algoritmo variaveis_e_constantes
    constante pi = 3.14
    inteiro idade
    escreva("Digite sua idade:")
    leia(idade)
    escreva("O valor de pi é: ", pi)
finalgoritmo
```

# Regras para Construção de Variáveis

Devem começar com uma letra ou underscore.

Podem conter letras, números e underscores.

Não podem ser palavras reservadas.

Exemplo: `nome`, `\_idade`, `nota1`.

# Declarando Variáveis

Em Portugal, declaramos variáveis com o tipo e o nome.  
Exemplo: `inteiro idade`.

```
algoritmo variaveis_e_constantes
    constante pi = 3.14
    inteiro idade
    escreva("Digite sua idade:")
    leia(idade)
    escreva("O valor de pi é: ", pi)
finalgoritmo
```



# Exemplo de Tipos de Dados Declarados em Variáveis

Declarar variáveis com diferentes tipos de dados:

```
algoritmo tipos_de_dados_exemplo
    inteiro idade
    real altura
    caractere inicial_nome
    logico aprovado

    escreva("Digite sua idade:")
    leia(idade)
    escreva("Digite sua altura (em metros):")
    leia(altura)
    escreva("Digite a inicial do seu nome:")
    leia(inicial_nome)
    escreva("Você foi aprovado? (true/false):")
    leia(aprovado)
```

```
    escreva("Idade: ", idade)
    escreva("Altura: ", altura)
    escreva("Inicial do nome: ", inicial_nome)
    escreva("Aprovado? ", aprovado)
finalgoritmo
```

# Quando Usar Variáveis e Constantes?

Use variáveis quando precisar armazenar valores que podem mudar ao longo do programa.

Use constantes para valores fixos que não devem ser alterados.

# Quando Usar Variáveis e Constantes?

Exemplo:

Neste exemplo, usamos a constante `pi` e a variável `raio` para calcular a área de um círculo.

```
algoritmo variaveis_e_constantes_exemplo
    constante pi = 3.14
    inteiro raio
    real area
    escreva("Digite o raio do círculo:")
    leia(raio)
    area <- pi * raio * raio
    escreva("A área do círculo é: ", area)
finalgoritmo
```

# Atribuição de Valores em Variáveis

Atribua valores a variáveis usando o operador de atribuição (`<-`).

Exemplo:

Aqui, atribuímos o valor `10` à variável `x`.

```
algoritmo atribuicao_valores
  inteiro x
  x <- 10
  escreva("O valor de x é: ", x)
finalgoritmo
```

# Visão Geral do VisualG vs-2

**Nome do Algoritmo**

**Declaração de variáveis**

**Bloco de comandos**

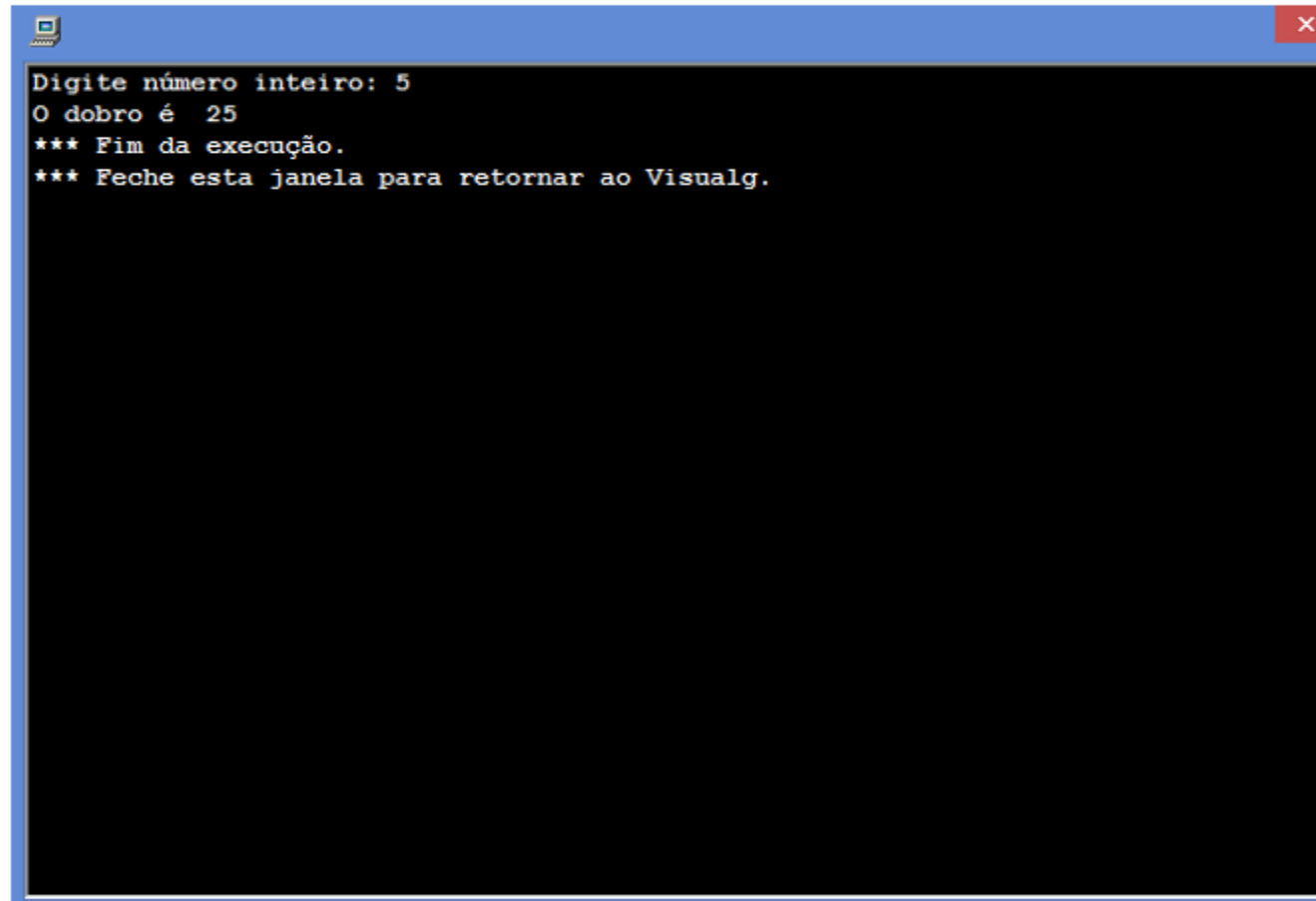
Escopo	Nome	Tipo	Valor
GLOBAL	NUMERO	I	5
GLOBAL	DOBRO	I	25

**Teste de Mesa**

Início da execução  
Digite número inteiro: 5  
O dobro é 25  
Fim da execução.

# Visão Geral do VisualG

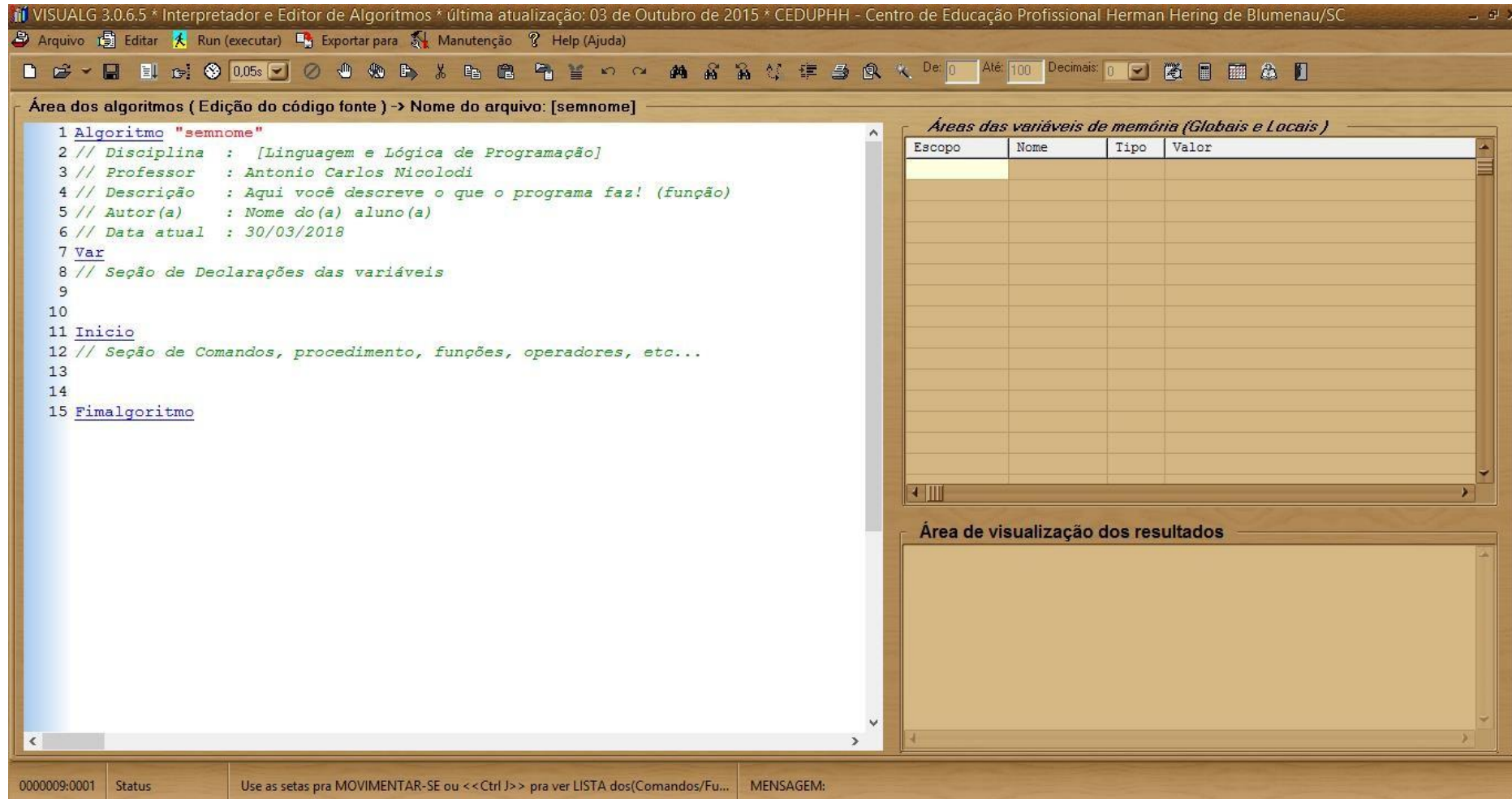
Executando >> F9



```
Digite número inteiro: 5
O dobro é 25
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Resultado da Execução

# Visão Geral do VisualG



Tela Inicial (versão 3.0.6.5)

# Visão Geral do VisualG

## Palavras Reservadas

[www.apoioinformatica.inf.br/produtos/visualg](http://www.apoioinformatica.inf.br/produtos/visualg)

aleatorio	faca	ou
algoritmo	falso	outrocaso
arquivo	fimalgoritmo	para
asc	fimenquanto	passo
ate	fimescolha	pausa
carac	fimfuncao	pos
caracpnum	fimpara	real
caractere	fimprocedimento	procedimento
caso	fimrepita	repita
compr	fimse	retorne
copia	funcao	se
cronometro	inicio	senao
debug	int	timer
e	inteiro	var
eco	interrompa	vetor
enquanto	leia	verdadeiro
entao	limpatela	xou
escolha	logico	
escreva	maiusc	
escreval	minusc	
	mod	
	nao	
	numpcarac	



## Exemplo

**var**

matricula : inteiro

nome\_funcionario : caractere

salario : real

gratificacao : logico

**inicio**

//Entrada de Dados

matricula <- 211

nome\_funcionário <- "Carlos Silva"

salario <- 950.00

gratificação <- verdadeiro //ou falso

# COMANDO DE ENTRADA

## Sintaxe

`leia (<lista de variáveis>)`

## Exemplos:


- ▶ `leia (n1)` - O valor digitado será armazenado na variável `n1`;
- ▶ `leia (k)` - Um ou vários caracteres digitados serão armazenados na variável `k` (Definida como caractere);
- ▶ No caso de utilizar `leia (<lista de variáveis>)`, será respeitada a ordem da lista de variáveis, da esquerda para direita;
- ▶ Ex.: `leia (n1, k, n2)`

# COMANDO DE ENTRADA

## Sintaxe

`escreva (<lista-de-variáveis>)`

`escreval (<lista-de-variáveis>)`



Imprime a variável na tela e o cursor vai para uma nova linha.

## Exemplos:

- ▶ `escreva (n1)` - Será mostrado na tela o conteúdo da variável `n1`.
- ▶ `escreva ("O texto digitado foi ", k)`
  - ▶ Será mostrado o texto entre " " e depois o conteúdo da variável `k`.

## Atividade

Um algoritmo foi criado em visual G.

Algoritmo "teste"

Var

a: inteiro;

Inicio

a<-2

escreva(a)

Fimalgoritmo

Qual seria o resultado em tela?

## Atividade

Um algoritmo foi criado em visual G.

Algoritmo "teste"

Var

a,b: inteiro;

Inicio

a<-3

b<-1

escreva(a)

Fimalgoritmo

Qual seria a o resultado em tela?

## Atividade

Um algoritmo foi criado em visual G.

Algoritmo "teste"

Var

a,b,c: inteiro;

Inicio

a<-3

b<-1

c<-a+b

escreva(c)

Fimalgoritmo

Qual seria a o resultado em tela?

# Exemplos

```
algoritmo "Algo_Ex01"  
  
var  
    sal_fixo, val_vendas, sal_total : real  
  
inicio  
  
    // Entrada  
    escreva("Digite Salário Fixo: ")  
    leia(sal_fixo)  
  
    escreva("Digite Valor de Vendas: ")  
    leia(val_vendas)  
  
    // Processamento  
    sal_total <- sal_fixo + (val_vendas * 0.15)  
  
    // Saída  
    escreva("Salário Total = ",sal_total)  
  
fimalgoritmo
```

# OPERADORES

+

<

=

<>





# Exemplos

Símbolo	Descrição
$\leftarrow$	Atribuição
$+$	Adição
$-$	Subtração
$*$	Multiplicação
$/$	Divisão
$a \setminus b$	Retorna o quociente da divisão inteira de a por b
$a \% b$	Retorna o resto da divisão inteira de a por b
$a ^ b$	Retorna o valor de a elevado a b

## Exemplos

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

# Exemplos

```
algoritmo "Algo_Ex02"
```

```
var
```

```
  x, y : inteiro
```

```
  gratificacao, aprovado : logico
```

```
inicio
```

```
  // Entrada
```

```
  x <- 5
```

```
  y <- 2
```

```
  gratificacao <- verdadeiro
```

```
  aprovado <- verdadeiro
```

```
  // Saída
```

```
  escreval("X = ", x)
```

```
  escreval("Y = ", y)
```

```
  escreval("Divisão X por Y é ", x / y)
```

```
  escreval("Quociente da divisão X por Y = ", x \ y)
```

```
  escreval("Resto da divisão X por Y = ", x % y)
```

```
  escreval("X elevado a Y = ", x ^ y)
```

```
  escreval("Gratificação = ", gratificacao)
```

```
  escreval("Aprovado = ", nao(aprovado))
```

```
fimalgoritmo
```

# Exemplos



```
X = 5
Y = 2
Divisão X por Y é 2.5
Quociente da divisão X por Y = 2
Resto da divisão X por Y = 1
X elevado a Y = 25
Gratificação = VERDADEIRO
Aprovado = FALSO

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

# Exemplos

algoritmo "Algo\_Ex03"

var

n1, n2, media : real  
resultado : caractere

inicio

*// Entrada*

escreva("Digite primeira Nota: ")

leia(n1)

escreva("Digite segunda Nota: ")

leia(n2)

*// Processamento*

media <- (n1+n2)/2

se (media >= 7) entao

    resultado <- "Aprovado"

senao

    result<-) = "Reprovado"

fimse

*// Saída*

escreval("Aluno está ",resultado, " com Média = ",media)

fimalgoritmo

# Exemplos



```
Digite primeira Nota: 9,25  
Digite segunda Nota: 8,25  
Aluno está Aprovado com Média = 8.75  
  
*** Fim da execução.  
*** Feche esta janela para retornar ao Visualg.
```

## REFERÊNCIAS

**K19: Lógica de Programação.** Abril. 2011. Disponível em:  
<http://k19.com.br>.

BERNARDO, Alessandro. **Algoritmos e Linguagens de Programação I.** FACITEC - FACULDADE DE CIÊNCIAS SOCIAIS E TECNOLÓGICAS - Tecnologia em Redes de Computadores.

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. **Algoritmos: teoria e prática.** 2.ed. Rio de Janeiro: Elsevier, 2002.



**ATÉ A PRÓXIMA AULA!**