

Técnico em Desenvolvimento de Sistemas



Docente: Rafael Sacramento

Sobre o Professor



- Formado em Bacharel em Sistemas de Informação, pós-Graduado em Análise de Sistemas com Ênfase em Governança e Docência da Tecnologia.
- Ex militar das Forças Armadas.
- Tenho 39 anos
- Atuo por mais de 14 anos como professor.



Histórico do Senac

O Senac – Foi criado em 10 de janeiro de 1946 através do decreto-lei 8.621. É uma entidade privada com fins públicos que recebe contribuição compulsória das empresas do comércio e de atividades assemelhadas. A nível nacional é administrado pela Confederação Nacional do Comércio.



Curso:

Carga Horária: 1.200 horas

Dividido em 12 Unidades Curriculares - UC;

Analisar requisitos e funcionalidades da aplicação – 108h;

Auxiliar na Gestão de Projetos de Tecnologia da Informação - 60h;

Desenvolver algoritmos - 108h;

Analisar programação estruturada e orientada a objetos- 48h;

Desenvolver aplicações desktop - 140h;

Criar e manter Banco de Dados - 108h;

Desenvolver aplicações web - 140h;

Desenvolver aplicações mobile - 140h;

Realizar operações de atualização e manutenção em aplicações desenvolvidas - 96h;

Realizar testes nas aplicações desenvolvidas - 108h;

Realizar operações de suporte junto ao usuário - 84h;

Projeto Integrador Desenvolvedor de aplicações - 60h;

[illegible]

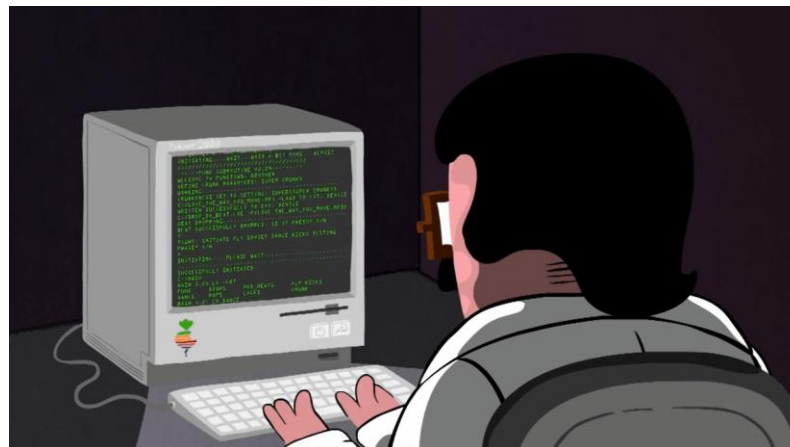
O que iremos abordar nessa UC3

- **Introdução Lógica de Programação;**
- **O que é variável?;**
- **Quais são os tipos das variáveis?;**
- **Introdução ao VisualG;**
- **Condicionais simples e composto;**
- **Estrutura de múltipla escolha;**
- **Laços de repetição: enquanto e para;**
- **Exemplos práticos utilizando laços;**
- **Vetores e Matrizes;**
- **Funções; e**
- **Exercícios.**

O que é Lógica de Programação?

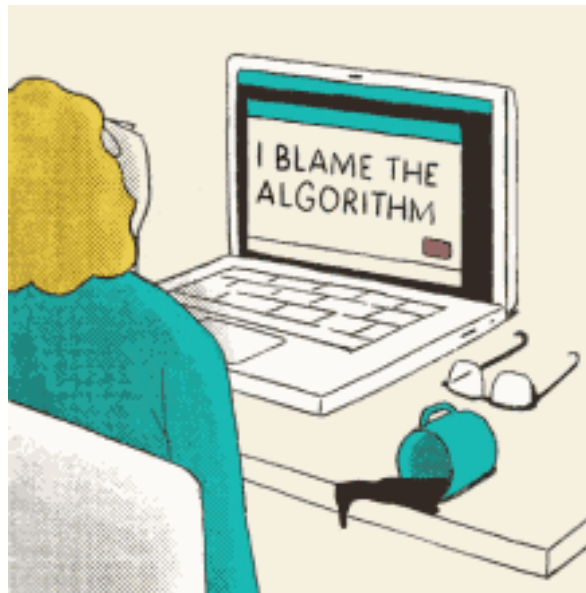


É o conjunto de regras e técnicas que os programadores utilizam para **projetar e desenvolver programas de computador**. É a **habilidade de pensar de forma lógica e estruturada**, decompondo **um problema complexo em etapas mais simples**.



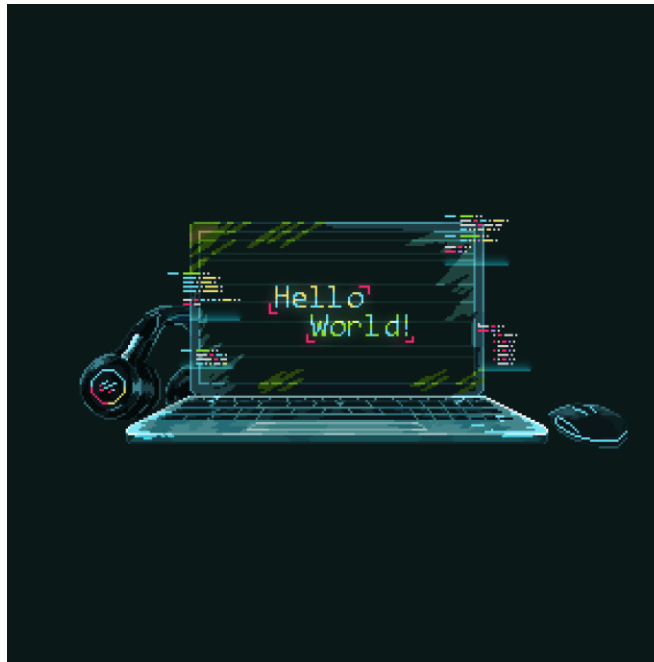
O que é um algoritmo?

Nada mais é do que uma receita que **mostra passo a passo os procedimentos necessários para a resolução de uma tarefa**. Em termos mais técnicos, um algoritmo é uma **sequência lógica, finita e definida de instruções que devem ser seguidas para resolver um problema ou executar uma tarefa**.



O que é variável?

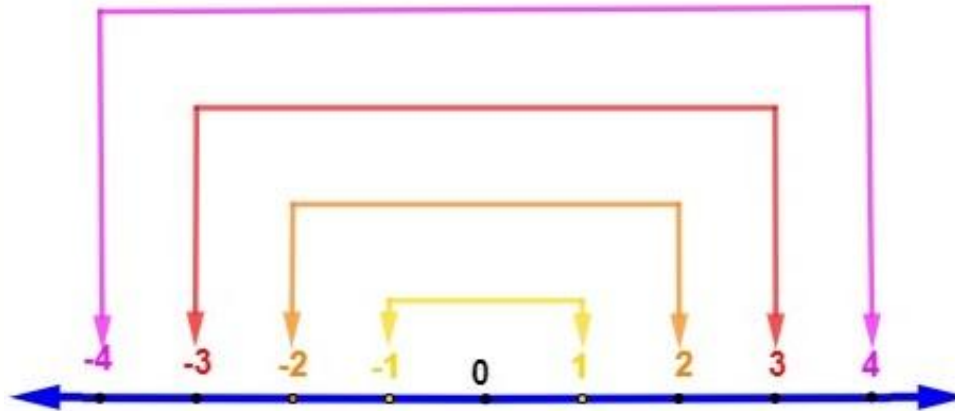
É um nome que definimos para armazenar dados de **forma simples**. O valor de uma variável pode ser alterado no andamento do algoritmo, por isso o nome de variável.



Quais o tipos de variáveis?

Int:

Variável número do tipo inteiro, positivo ou negativo.



float:

Variável **numérica** do tipo **decimal**

5

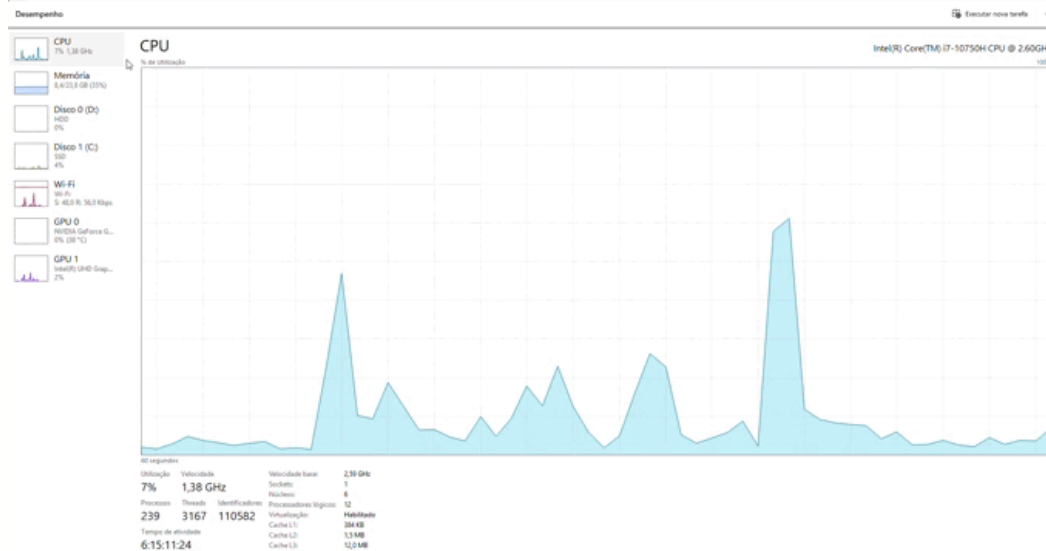
Double:

Variável **numérica do tipo decimal de precisão dupla**.

5

Qual e a diferença entre float e double?

Doubles usam mais memória do que floats. Se for necessária alta precisão nos cálculos, é **melhor usar double**. Se a precisão não for crítica e você **quiser economizar memória** e recursos computacionais, **poderá usar float**.



char:

Variável que **representa** um **caractere** do **tipo texto**.



<tá no
código/>

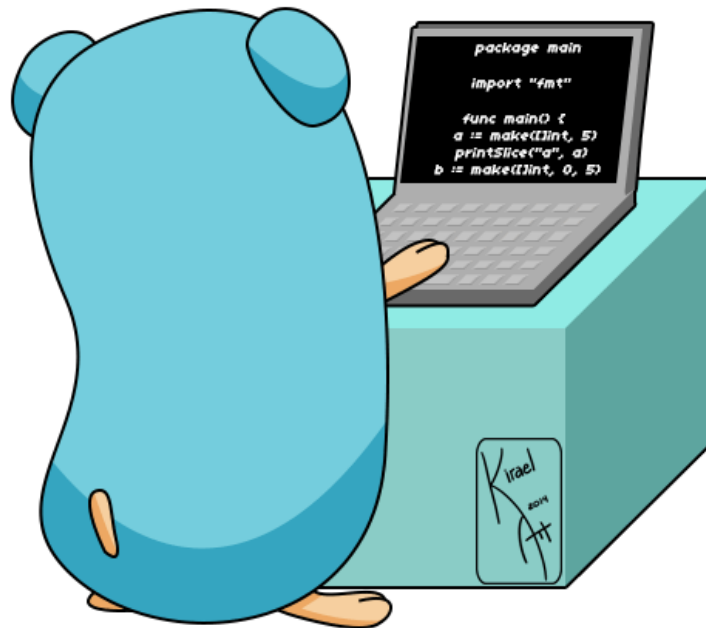
String:

Variável que representa **um conjunto de caracteres** do tipo **texto**.



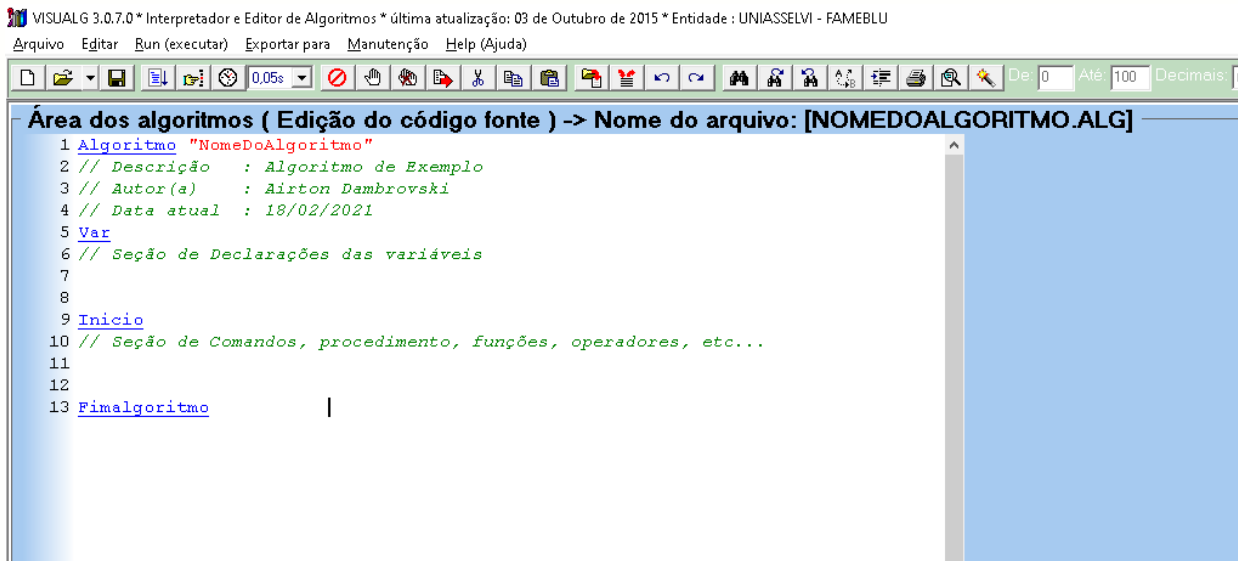
Qual e a diferença entre char e string?

Uma **String** é uma **sequência de caracteres** enquanto um **char** **representa um único caractere**.



O que é VisuAlg?

É um **programa** que **interpreta e executa algoritmos** como um "programa" normal de computador. Baseado em uma linguagem parecida com o "**Portugol**". Esta **ferramenta** permite aos alunos **iniciantes em programação** o exercício dos **seus conhecimentos** num ambiente próximo da realidade.



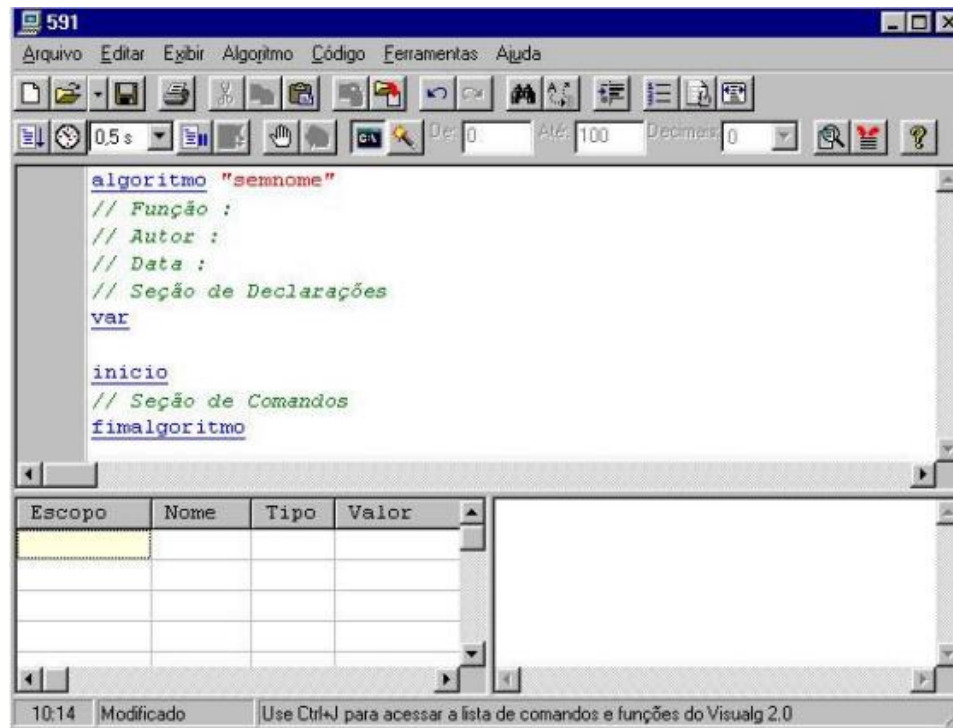
```
1 Algoritmo "NomeDoAlgoritmo"
2 // Descrição : Algoritmo de Exemplo
3 // Autor(a) : Ailton Dambrowski
4 // Data atual : 18/02/2021
5 Var
6 // Seção de Declarações das variáveis
7
8
9 Inicio
10 // Seção de Comandos, procedimento, funções, operadores, etc...
11
12
13 Fimalgoritmo
```

O VisuAlg foi criado pela Apoio Informática, **ele lê e interpreta em uma linguagem próxima do português** estruturado(Portugol) como um programa normal de computador.



Qual a linguagem Portugal?

A linguagem interpretada é bem simples: é uma versão portuguesa dos **pseudocódigos largamente utilizados nos livros de introdução à programação.**



Variável no VisuAlg

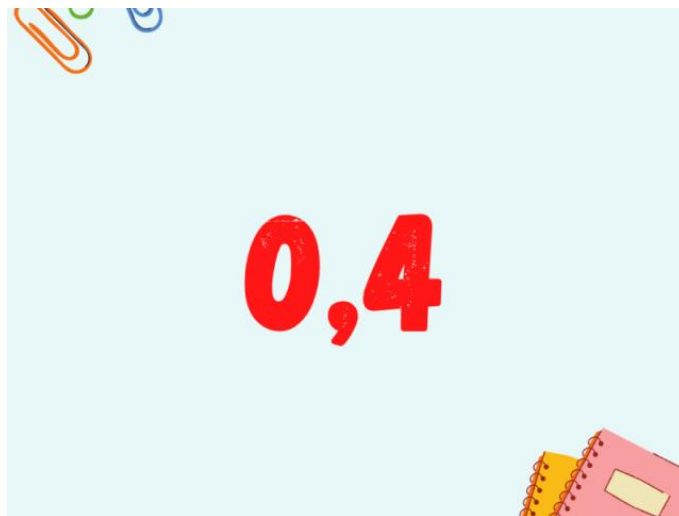
Inteiro:

Variáveis numéricas do tipo inteiro, ou seja, **sem casas decimais.**

6 5 3 1 8 7 2 4

Real:

Variáveis numéricas do tipo real, ou seja, com casas decimais.



Caractere:

Variáveis do tipo string, ou **seja, cadeia de caracteres.**

LOADING



Logico:

Variáveis do tipo booleano, ou seja, com valor VERDADEIRO ou FALSO.

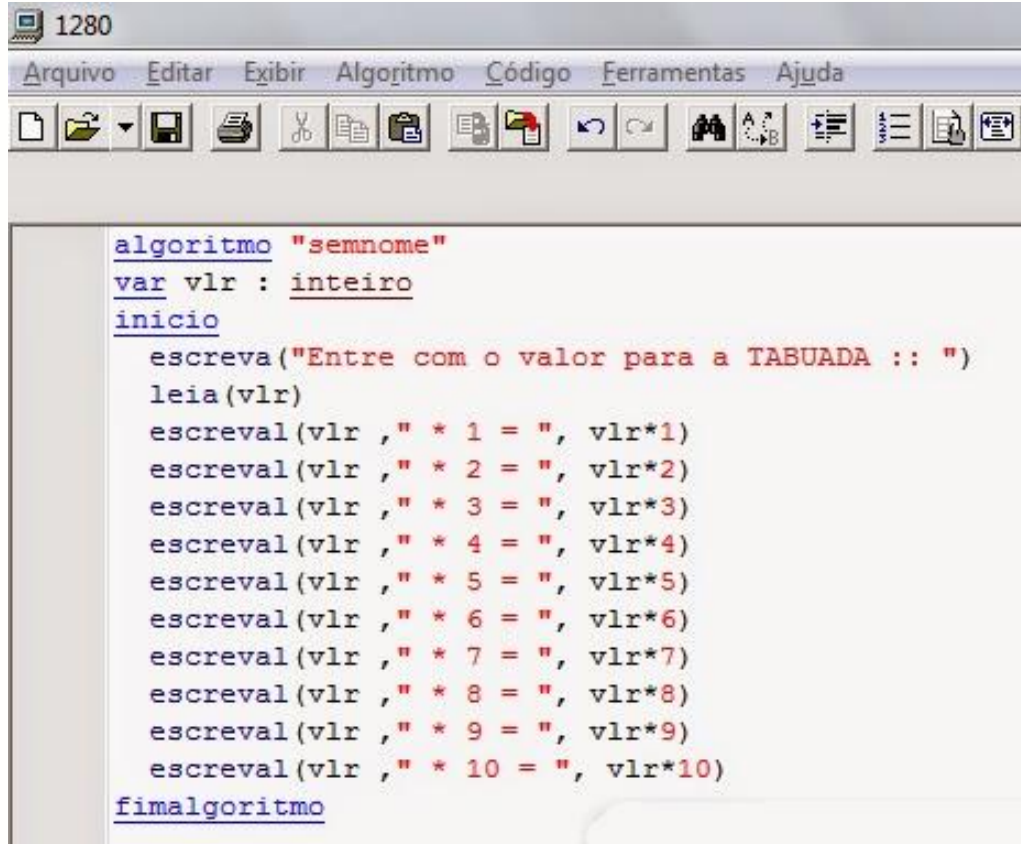


VisuAlg

Exemplos	19/05/2024 18:53	Pasta de arquivos	
help	19/05/2024 18:53	Pasta de arquivos	
skins	19/05/2024 18:53	Pasta de arquivos	
dicas	20/06/2008 07:59	Parâmetros de co...	11 KB
help	13/07/2015 18:13	Arquivo de Ajuda ...	691 KB
LEIAME	12/09/2015 00:23	Documento de Te...	1 KB
LEIA-ME	12/09/2015 00:23	Documento de Te...	1 KB
listas	27/05/2024 18:42	Documento de Te...	5 KB
Menu do Visualg autalizado	20/07/2015 02:47	Documento do A...	1.236 KB
README	12/09/2015 00:23	Documento de Te...	1 KB
RELAÇÃO DOS COMANDOS DO VISUAL...	04/10/2015 01:59	Documento de Te...	7 KB
TESTE.alg	16/02/2019 16:12	Arquivo ALG	1 KB
VISUALG	26/01/2017 16:51	Parâmetros de co...	1 KB
VISUALG30	13/07/2015 19:13	Arquivo de Ajuda ...	691 KB
visualg30	19 22:45	Aplicativo	2.110 KB
VISUALG30	27/05/2024 18:56	Parâmetros de co...	2 KB



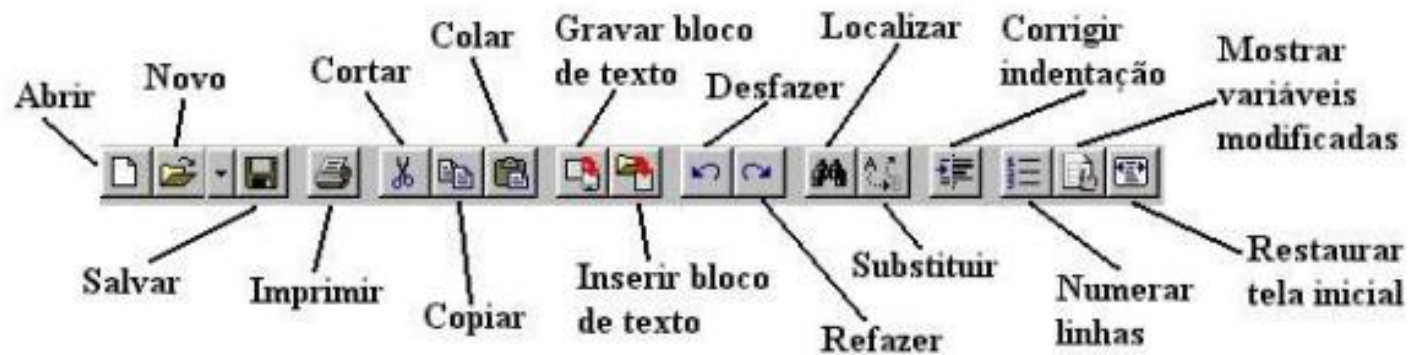
A Tela Principal do VisuAlg



The image shows the main interface of the VisuAlg software. At the top, there is a menu bar with the following options: Arquivo, Editar, Exibir, Algoritmo, Código, Ferramentas, and Ajuda. Below the menu bar is a toolbar containing various icons for file operations (like opening, saving, printing), editing (like undo, redo, cut, copy, paste), and algorithm execution (like running, stepping through, and debugging). The main area is a code editor displaying the following algorithm:

```
algoritmo "semnome"
var vlr : inteiro
inicio
    escreva("Entre com o valor para a TABUADA :: ")
    leia(vlr)
    escreval(vlr, " * 1 = ", vlr*1)
    escreval(vlr, " * 2 = ", vlr*2)
    escreval(vlr, " * 3 = ", vlr*3)
    escreval(vlr, " * 4 = ", vlr*4)
    escreval(vlr, " * 5 = ", vlr*5)
    escreval(vlr, " * 6 = ", vlr*6)
    escreval(vlr, " * 7 = ", vlr*7)
    escreval(vlr, " * 8 = ", vlr*8)
    escreval(vlr, " * 9 = ", vlr*9)
    escreval(vlr, " * 10 = ", vlr*10)
finalgoritmo
```

A Barra de Tarefas



VISUALG 3.0.7.0 * Interpretador e Editor de Algoritmos * última atualização: 03 de Outubro de 2015 * Entidade : UNIASSELVI - FAMEBLU

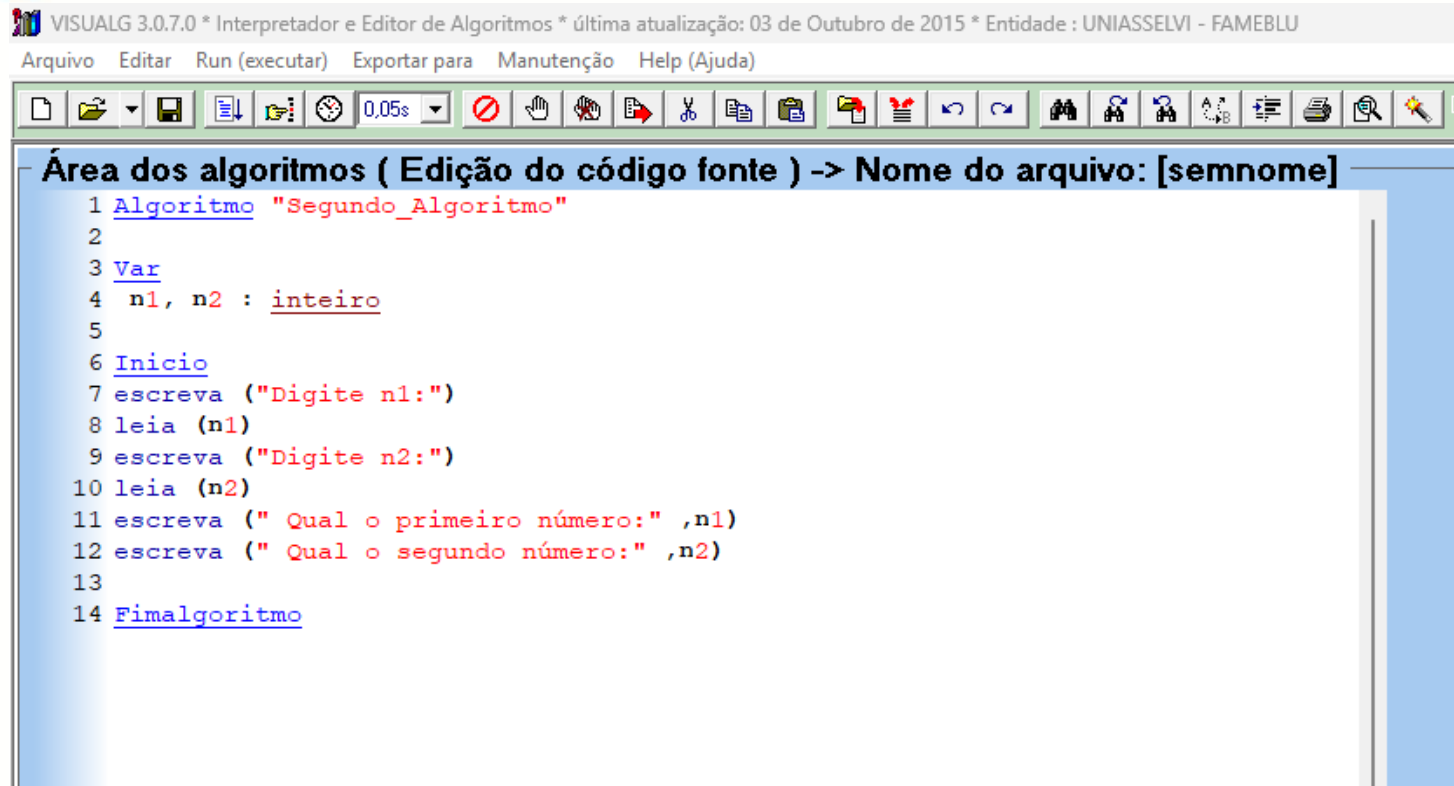
Arquivo Editar Run (executar) Exportar para Manutenção Help (Ajuda)

0,05s

Área dos algoritmos (Edição do código fonte) -> Nome do arquivo: [semnome]

```
1 Algoritmo "Primeiro_Algoritmo"
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 : 02/06/2024
7 Var
8 // Seção de Declarações das variáveis
9
10 Inicio
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12 escreva("Olá Mundo!")
13
14 Fimalgoritmo
```

Escreva: Ele imprimir ou mostra na tela.



leia: Lendo do teclado, funciona como atribuição de valor, só pelo teclado.

Comando é leia (nomevariavel)

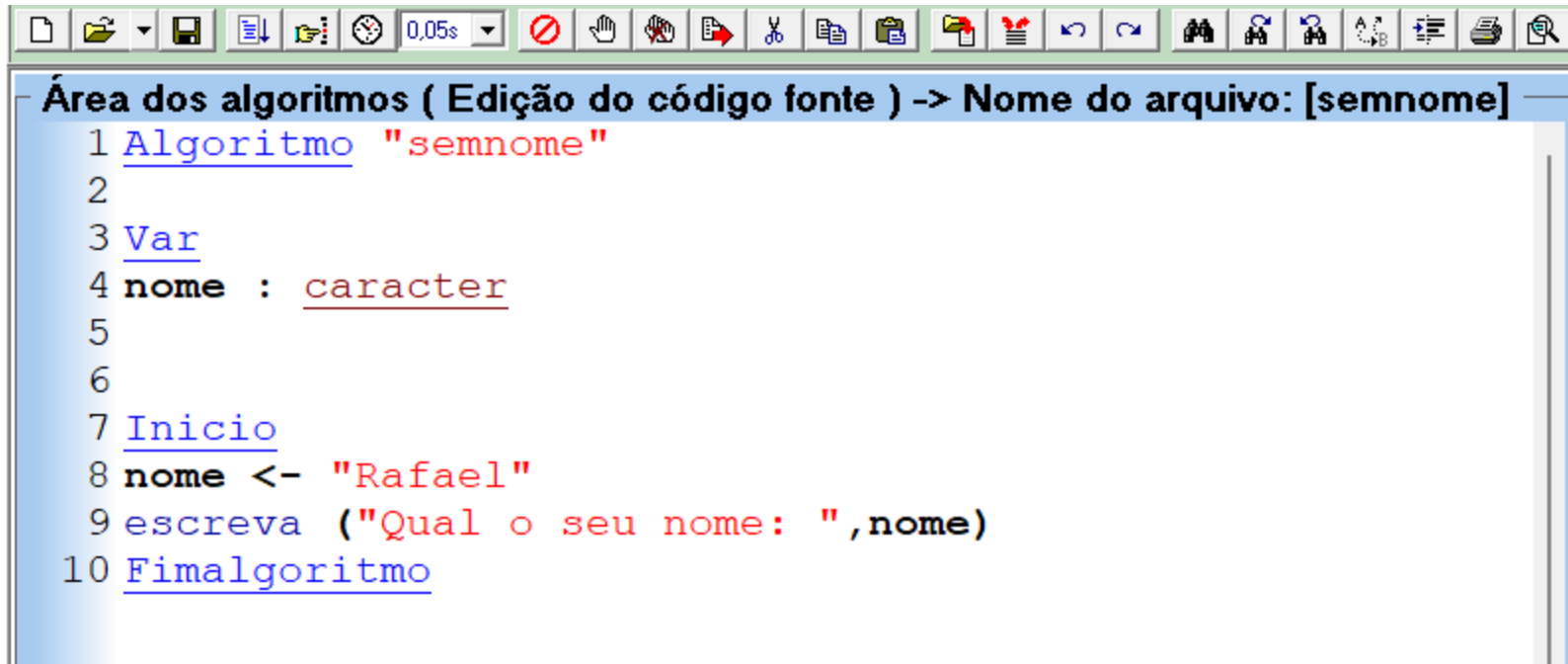
Concatenação no VisuAlg

```
escreva ("Qual o seu nome: ", nome)
```

Concatenar: É a junção de uma variável com texto **ou texto com variável.**



Atribuir um valor a variável

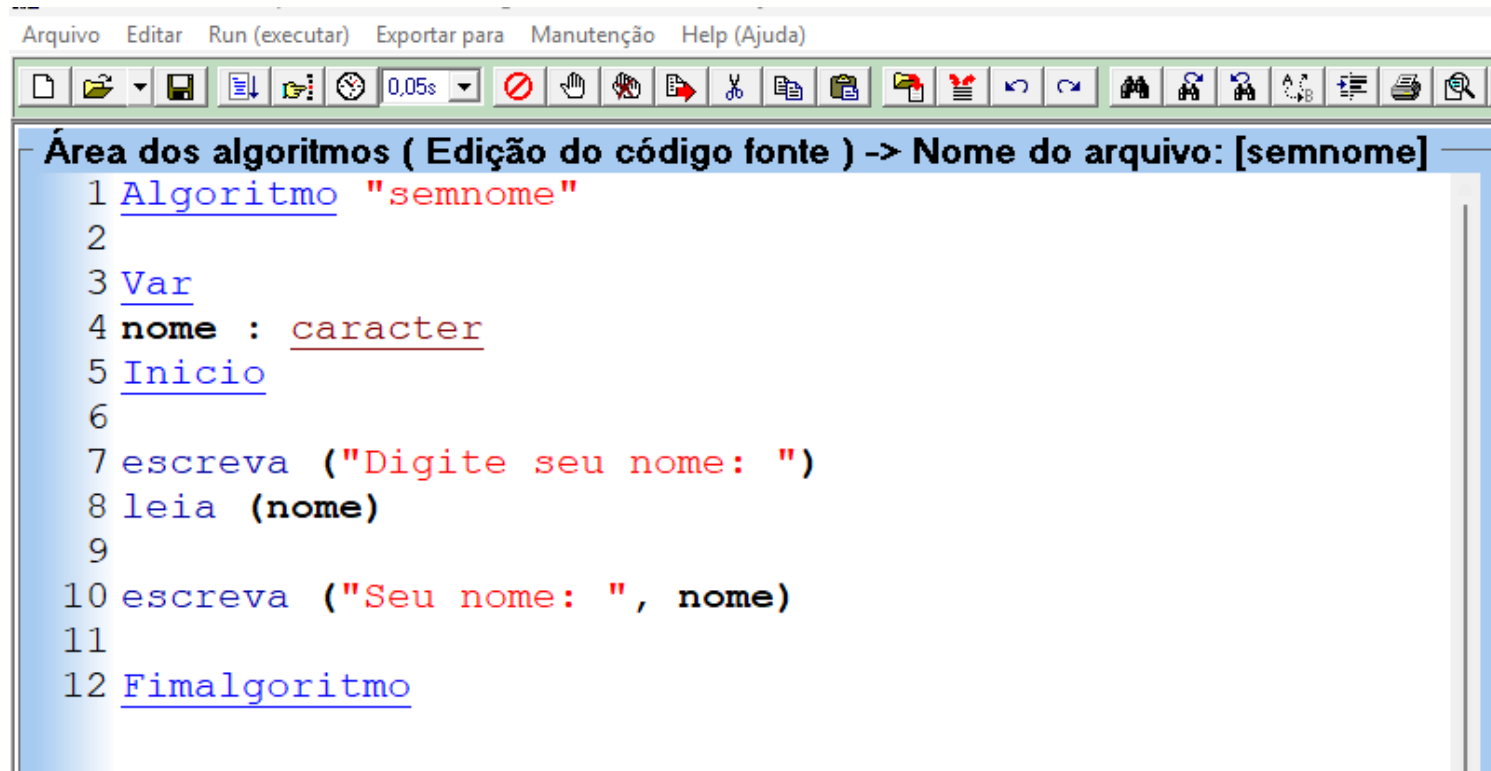


The screenshot shows a code editor window with a toolbar at the top. The title bar reads "Área dos algoritmos (Edição do código fonte) -> Nome do arquivo: [semnome]". The code is as follows:

```
1 Algoritmo "semnome"
2
3 Var
4 nome : caracter
5
6
7 Inicio
8 nome <- "Rafael"
9 escreva ("Qual o seu nome: ", nome)
10 Fimalgoritmo
```

A atribuição de valores a variáveis é feita com o operador **<-**. Do seu lado esquerdo fica a variável à qual está sendo atribuído o valor, e à sua direita pode-se colocar qualquer expressão (constantes, variáveis, expressões numéricas), **desde que seu resultado tenha tipo igual ao da variável.**

Inserir um valor a variável



The screenshot shows a software interface with a menu bar (Arquivo, Editar, Run (executar), Exportar para, Manutenção, Help (Ajuda)) and a toolbar. The main window is titled 'Área dos algoritmos (Edição do código fonte) -> Nome do arquivo: [semnome]'. It contains the following code:

```
1 Algoritmo "semnome"  
2  
3 Var  
4 nome : caracter  
5 Inicio  
6  
7 escreva ("Digite seu nome: ")  
8 leia (nome)  
9  
10 escreva ("Seu nome: ", nome)  
11  
12 Fimalgoritmo
```

Nesse exemplo usando o comando leia o usuário vai colocar o nome dele e automaticamente vai imprimir na tela conforme sugere.

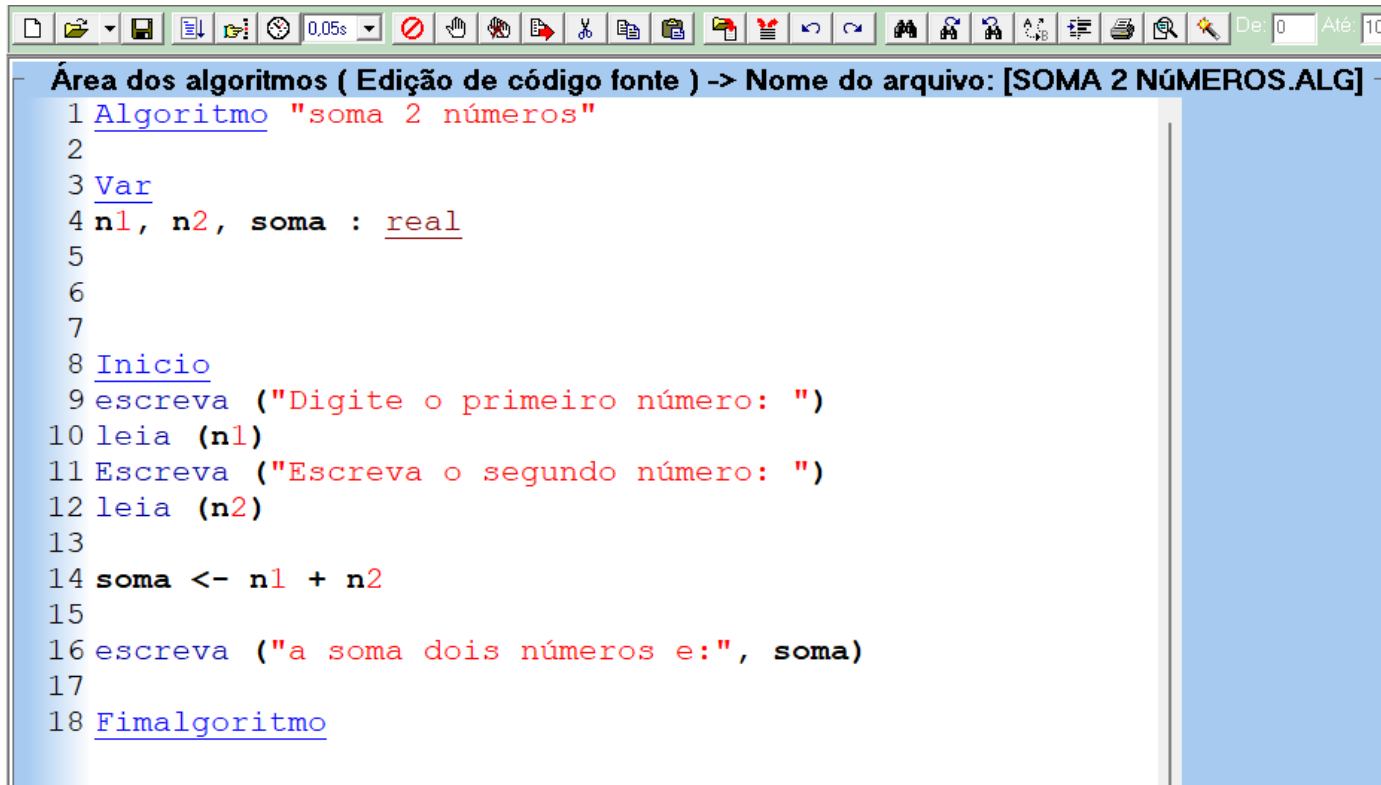
Inserir um valor a variável

Área dos algoritmos (Edição do código fonte) -> Nome do arquivo: [DIGITE SEU

```
1 Algoritmo "Digite seu nome"
2
3 Var
4 nome, sobrenome, nomecompleto : caracter
5 Inicio
6
7 escreva ("Digite seu nome: ")
8 leia (nome)
9
10 escreva ("Digite seu sobrenome: ")
11 leia (sobrenome)
12
13 nomecompleto <- nome + sobrenome;
14
15 escreva ("Seu nome: ", nomecompleto)
16
17 Fimalgoritmo
```

Nesse exemplo **Declaramos três variáveis** e na terceira variável vai receber o nome + sobrenome e imprimir o nome completo.

Soma



```
1 Algoritmo "soma 2 números"
2
3 Var
4 n1, n2, soma : real
5
6
7
8 Inicio
9 escreva ("Digite o primeiro número: ")
10 leia (n1)
11 Escreva ("Escreva o segundo número: ")
12 leia (n2)
13
14 soma <- n1 + n2
15
16 escreva ("a soma dois números e:", soma)
17
18 Fimalgoritmo
```

Nesse exemplo **Declaramos três variáveis do tipo real** e na terceira variável vai receber o $n1 + n2$ e imprimir a soma dos números digitados pelo usuário.

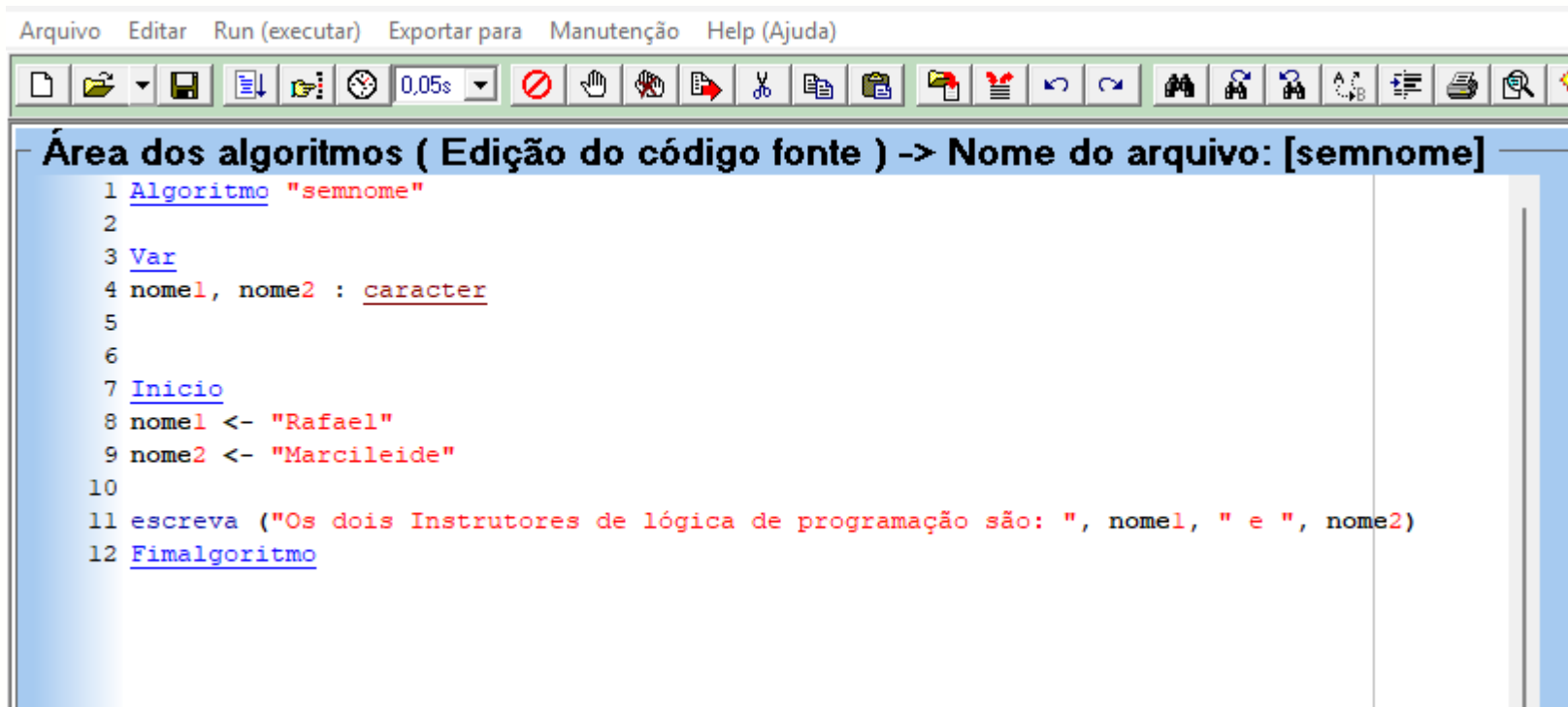
Média

Área dos algoritmos (Edição de código fonte) -> Nome do arquivo: [média.ALG]

```
1 Algoritmo "semnome"
2
3 Var
4 n1, n2, n3 : real
5
6 Inicio
7 escreva ("digite o primeiro número: ")
8 leia (n1)
9 escreva ("digite o segundo numero: ")
10 leia (n2)
11 escreva ("digite o terceiro número: ")
12 leia (n3)
13 escreva ("digite o quarto numero: ")
14 escreva (" O resultado é:", (n1+n2+n3)/3)
15
16 Fimalgoritmo
```

Nesse exemplo **Declaramos três variáveis do tipo real**, concatenei na impressão a soma dos 3 números digitados pelo usuário e foi dividido por 3.

Algoritmo que recebendo 2 caracteres



```
1 Algoritmo "semnome"
2
3 Var
4 nome1, nome2 : caracter
5
6
7 Inicio
8 nome1 <- "Rafael"
9 nome2 <- "Marcileide"
10
11 escreva ("Os dois Instrutores de lógica de programação são: ", nome1, " e ", nome2)
12 Fimalgoritmo
```

Nesse exemplo **Declaramos duas variáveis do tipo caracter,** demos um valor para cada variável e concatenei na impressão os valores atribuídos.

Operadores Aritméticos

Operador	Operação	Operandos	Resultado
+	Adição	Inteiro, Real	Inteiro, Real
-	Subtração	Inteiro, Real	Inteiro, Real
*	Multiplicação	Inteiro, Real	Inteiro, Real
/	Divisão	Inteiro, Real	Real
DIV	Divisão inteira	Inteiro	Inteiro
MOD	Resto da Divisão	Inteiro	Inteiro

Exemplos:

- 7 **DIV** 2 = **3**
- 7 **MOD** 2 = **1**

Div

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

Ex: $7 \text{ div } 2 \rightarrow 3$

Mod

Retorna **o resto da divisão inteira entre 2 números inteiros.**

Ex: $7 \text{ mod } 2 \rightarrow 1$

Obs: O nome mod é uma convenção amplamente aceita em computação para a operação de módulo, que encontra o resto da divisão de um número por outro. No Visualg, como em muitas outras linguagens, mod é utilizado para esta operação matemática fundamental.

Operadores Relacionais

= : igual

<>: diferente

> : maior que

< : menor que

>= : maior ou igual que

<= : menor ou igual que

Operadores Lógicos

E (and): Verdadeiro se ambas as expressões forem verdadeiras.

OU (or): Verdadeiro se pelo menos uma das expressões for verdadeira.

NÃO (not): Inverte o valor da expressão.

CONDIÇÕES

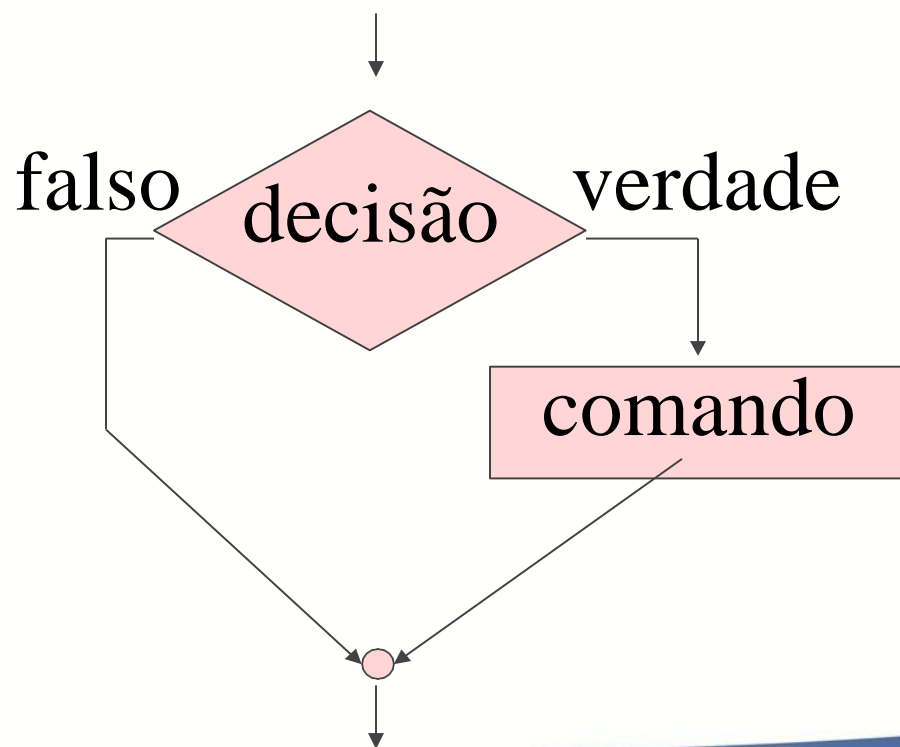
Um **comando condicional** é aquele que permite decidir se um determinado bloco de comandos **deve ou não ser executado**, a partir do resultado de uma expressão **relacional** ou **lógica**.



Estrutura Condicional simples (se)

Formas de Representação no Algoritmo

Fluxograma



Algoritmo

se (condição)

então <comando>

fim-se;

EXEMPLO (condicional): Ler um números inteiros e dizer que é maior que cinco.

Algoritmo

Var

numero: inteiro

Inicio

escreva(" Digite um número inteiro: ")

leia(numero)

se numero > 5 entao

 escreva("O número é maior que cinco.")

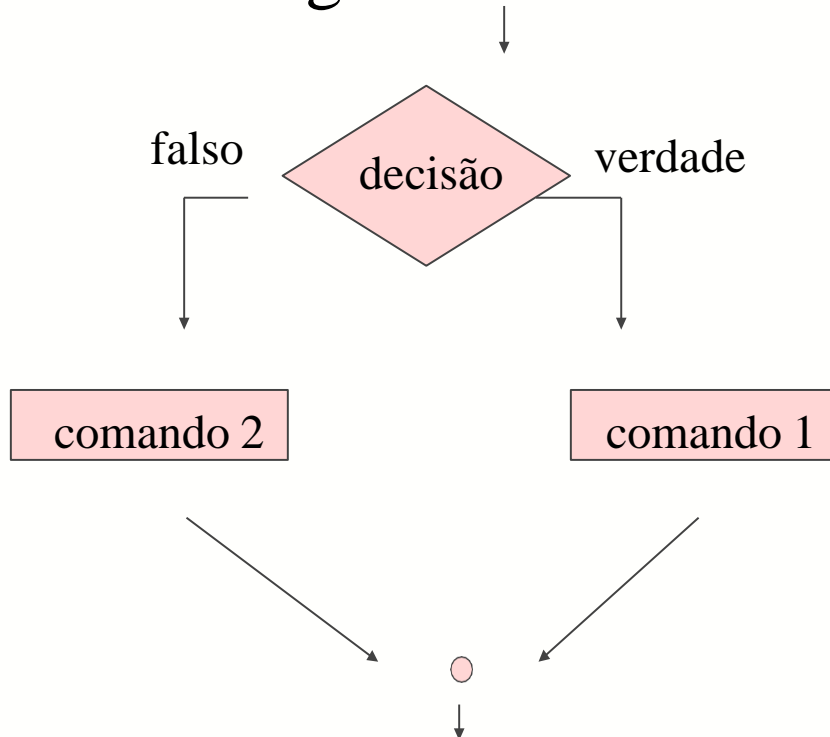
fimse

Fimalgoritmo

Estrutura Condicional composta (Se/Senao)

Formas de Representação no Algoritmo

Fluxograma



Algoritmo

se (condição)
então <comando1>
senão <comando2>

fim-se

EXEMPLO (condicional): Ler um números inteiros e dizer que é maior que cinco. (Usando o **SENÃO**)

Algoritmo

Var

numero: inteiro

Inicio

escreva(" Digite um número inteiro: ")

leia(numero)

se (numero > 5) entao

 escreva("O número é maior que cinco.")

senao

 escreval("O número é menor que cinco.")

fimse

Fimalgoritmo

Estrutura de repetição ENQUANTO

Vamos para estruturas de repetição, também conhecidas como LOOP. Tratar de forma especial a **estrutura de repetição ENQUANTO (em inglês, WHILE)**. Seu funcionamento é tão simples quanto a estrutura de decisão **SE-ENTÃO**. A diferença é que os passos dentro deste bloco, são repetidos enquanto a expressão booleana (VERDADEIRO ou FALSO) resultar VERDADEIRO.



NA PRÁTICA!

Vejamos um exemplo de algoritmo utilizando a ferramenta VisuAlg. Vamos implementar um algoritmo para **somar valores até o usuário digitar o valor 0**. Ou seja, vamos somar todos os valores que o usuário digitar, porém quando ele digitar 0 o **"loop"** acaba, a cada iteração do loop vamos apresentar o resultado atual da soma.

Abra o VisualG e digite o exemplo a seguir:

var

valorDigitado : REAL

soma : REAL

inicio

soma < 0

ESCREVA ("Digite um valor para a soma: ")

LEIA (valorDigitado)

ENQUANTO valorDigitado <> 0 **FACA**

soma <- soma + valorDigitado

ESCREVAL ("Total: ", soma)

ESCREVA ("Digite um valor para a soma: ")

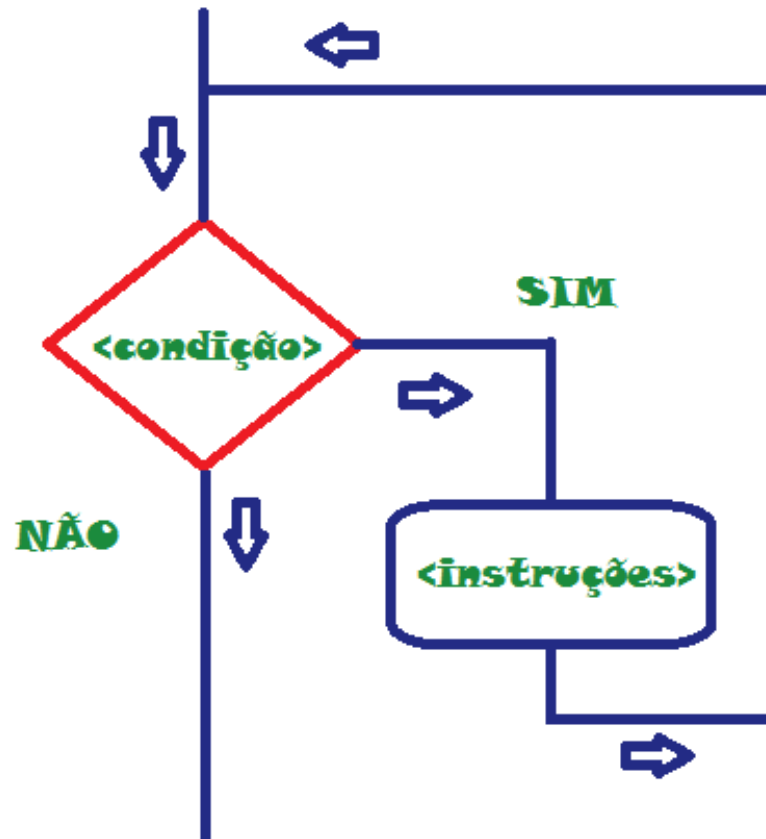
LEIA (valorDigitado)

FIMENQUANTO

ESCREVAL ("Resultado: ", soma)

fimalgoritmo

Fluxograma



Olhando o fluxograma, podemos observar se a condição for **sim** ele vai seguir as instruções e assim **entraria em um loop** e iria ser interrompido quando necessário.

Dúvidas?

Então vamos para os exercícios.

Estrutura de repetição PARA

O comando **PARA** permite construir estruturas de loop para casos onde se conhece de antemão o número de repetições que devem ser realizadas (ou seja, número finito de laços). **Por exemplo, quando sabemos de antemão que o laço deve se repetir 10 vezes; portanto, neste caso podemos usar a estrutura para.**



Diferença do ENQUANTO e o PARA

PARA - Para uma **quantidade de repetições já definidas**, ou mesmo quando você precisar contar as repetições.

ENQUANTO - Quando **seu código tiver a quantidade de repetições indefinidas** e dependendo de uma ou mais condições para parar o laço.

PARA



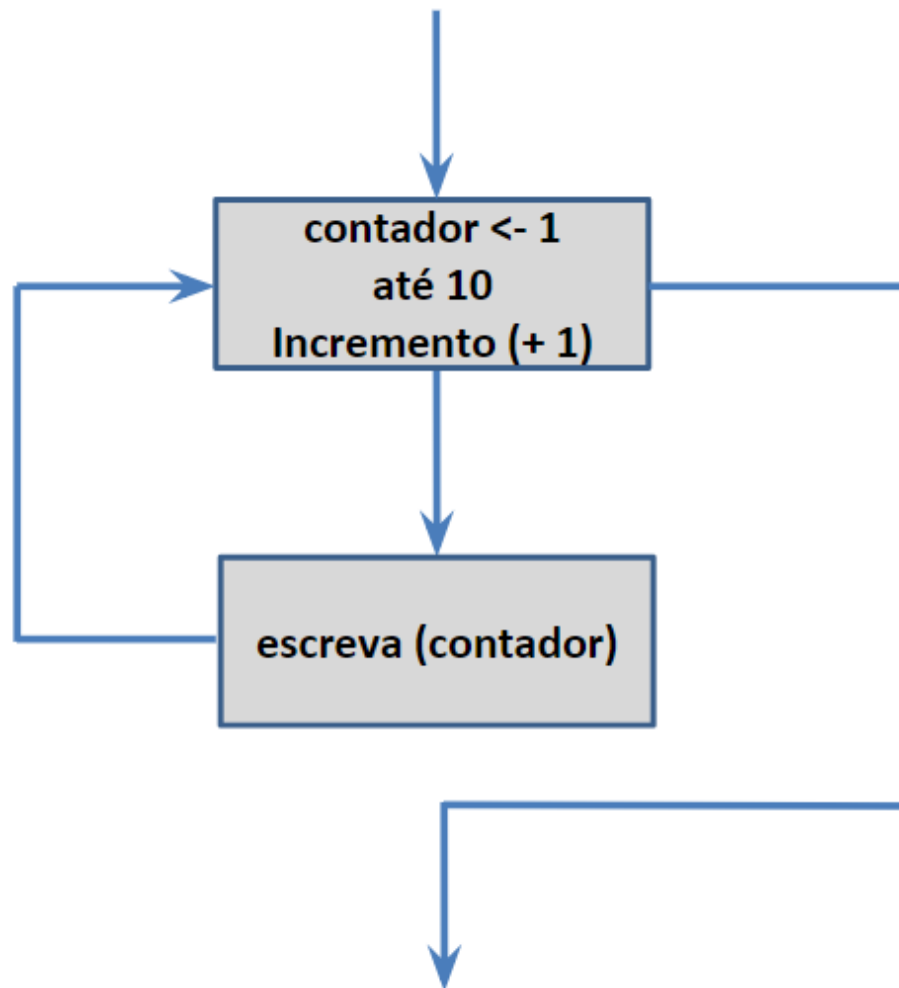
ENQUANTO

Exemplo 1

Imprimir na tela os números de 1 a 10, desta vez usando a estrutura de repetição “para”.

```
var
  contador: inteiro
inicio
  para contador de 1 ate 10 faca
    escreva(contador, " ")
  fimpara
finalgoritmo
```

O fluxograma a seguir mostra o funcionamento deste código:



Contagem até 10 c/ passo usando o para e faça:

x: inteiro

para x de 1 ate 10 passo 1 faça

 escreval(x)

fimpara

Exemplo 2

Imprimir na tela os **números de 10 a 50 de 5 em 5**, desta vez usando a estrutura de repetição “para”.

```
var
    contador: inteiro
inicio
    para contador de 10 ate 50 passo 5 faca
        escreva(contador, " ")
    fimpara
fimalgoritmo
```

Exemplo 3

Imprimir na tela os **números ímpares de 1 a 20 com condição**, desta vez usando a estrutura de repetição “para”.

var

numeros: inteiro

inicio

para numeros de **1** **ate** **20** faca

se numeros **mod 2 <> 0** entao

 escreva(numeros, " ")

fimse

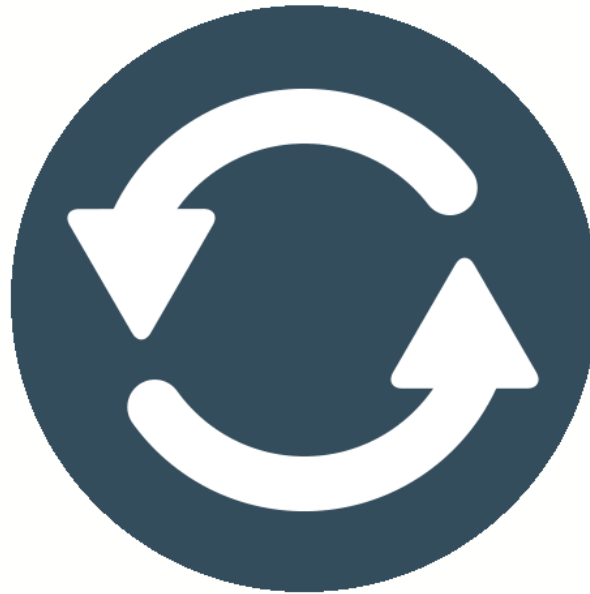
fimpara

fimalgoritmo

Estrutura de repetição REPITA

Trata-se de uma estrutura que efetua um teste lógico **no final do loop**, em vez de no início, como a estrutura enquanto.

Com o REPITA ATÉ o conjunto de instruções é **executado enquanto a condição testada retornar Falso**.



Sintaxe

repita

Instruções executadas enquanto
condição falsa
até (condição seja verdadeira)

Exemplo 1:

Imprimir na tela os números de 1 a 10, agora usando estrutura de repetição “repita...até” (código para o VisualG):

Var

num : inteiro

Inicio

num <- 1

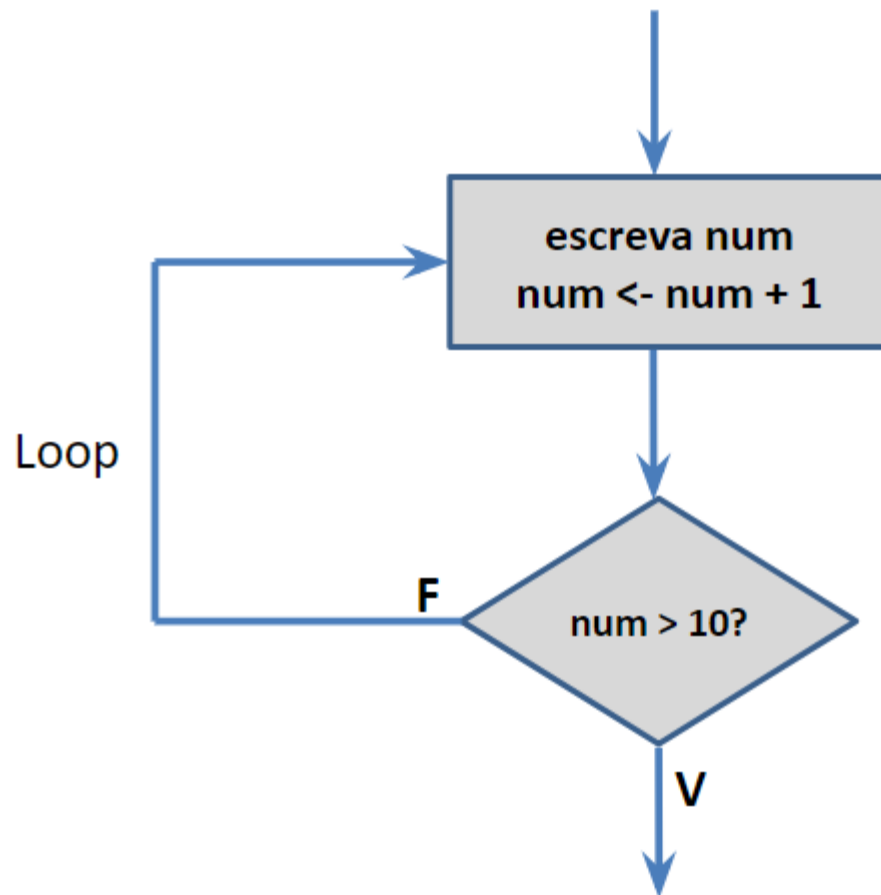
repita

escreval (num)

num <- num + 1

ate (num > 10)

Fimalgoritmo



Exemplo 2:

Imprimir na tela o calculo da média de 4 números “repita...até” (código para o VisualG):

Var

contador, numero, soma: inteiro
media: real

Inicio

contador <- 1

Repita

Escreva("Digite o ", contador, "º número: ")

Leia(numero)

soma <- soma + numero

contador <- contador + 1

Ate contador > 4

media <- soma / 4

Escreva("A média dos números digitados é: ", media)

Fimalgoritmo

Exemplo 3:

Mostre quantos números pares tem de 1 a 20 “repita...até” (código para o VisualG):

Var

contador, num: inteiro

Inicio

contador <- 1

Repita

Se (contador mod 2 = 0) Entao

num <- num + 1

Fimse

contador <- contador + 1

Ate contador > 20

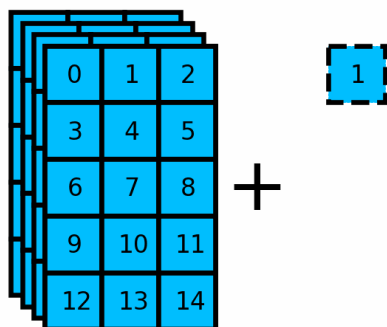
Escreva("A quantidade de números pares de 1 a 20 é: ", num)

Fimalgoritmo

Vetor (Arrays)

Um **array** é uma estrutura de dados **homogênea** que mantém uma **série de elementos de dados de mesmo tipo**. Pode-se acessar os elementos individuais armazenados no array por meio de uma **posição de índice associada, geralmente numérica**.

No geral, os **arrays possuem tamanho fixo, ou seja, número de posições definida**; em algumas linguagens de programação, existem estruturas de arrays que possuem tamanho variável. Vamos estudar aqui os arrays tradicionais, de tamanho especificado.



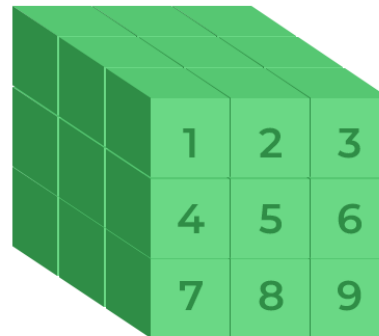
Classificação dos arrays

Os arrays são classificados de acordo com a sua dimensão de armazenamento de dados, como segue:

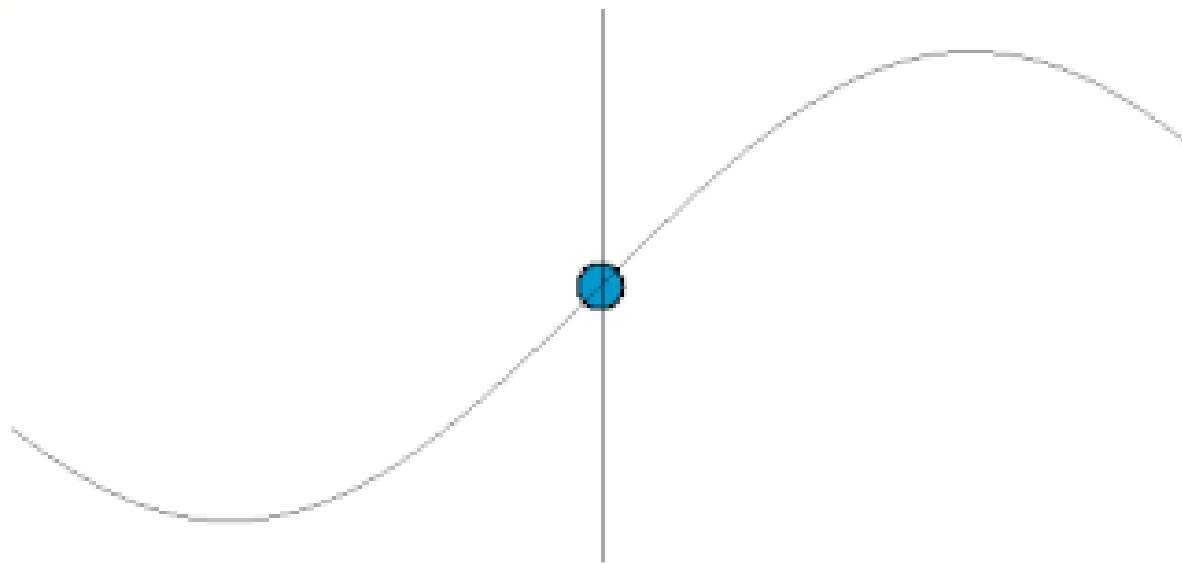
Unidimensional: **Vetor**

Bidimensional: **Matriz**

Tridimensional: **Cubo**

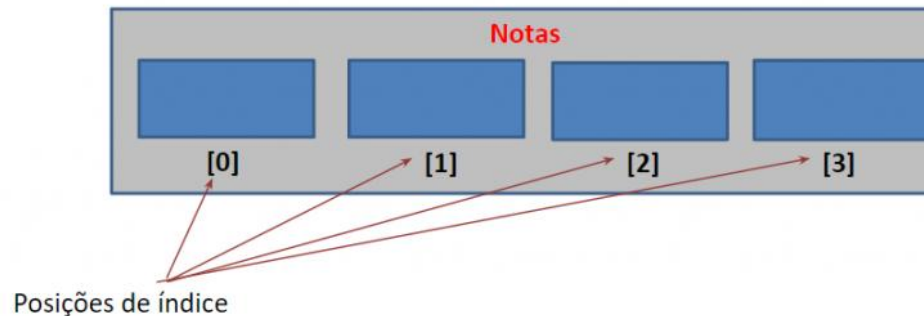


Arrays unidimensionais, também conhecidos como “Vetores”. Usaremos essa terminologia ao longo da semana. Posteriormente **vamos estudar também os arrays bidimensionais, chamados também de “matrizes”.**



Vetores

Um vetor é um array unidimensional, ou seja, de uma única dimensão; é análogo a uma linha única de dados em uma planilha ou tabela. A figura a seguir ilustra a estrutura interna de um vetor de quatro posições, que permite portanto armazenar até quatro dados, de nome **Notas**:



No geral a **contagem das posições se inicia em zero (0)**, de modo que a **primeira posição do vetor será a posição 0**, a segunda posição será 1, e assim por diante; a **última posição do vetor será a de número $n - 1$, onde n é o número total de posições disponíveis (tamanho do array)**. Assim, em um vetor de 4 posições a última posição será $4 - 1 = 3$.

As posições em um vetor são sempre indicadas pelo número da posição entre colchetes [].

Declaração de vetores

Podemos declarar um vetor em português estruturado usando a seguinte sintaxe:

nomeVetor: vetor [i..f] de Tipo_Dados

Onde:

nomeVetor é o nome escolhido para o vetor, que deve seguir as regras de nomeação de variáveis.

i = Valor da primeira posição do vetor (preferencialmente **zero**)

F = Valor da última posição do vetor (tamanho do vetor – 1)

Tipo_Dados = tipo dos dados que serão armazenados nas posições do vetor

Um array (arranjo ou vetor) é um conjunto de dados (que pode assumir os mais diversos tipos, desde do tipo primitivo, a objeto dependendo da linguagem de programação). **Arrays são utilizados para armazenar mais de um valor em uma única variável.** Isso é comparável a uma variável que pode armazenar apenas um valor.

Exemplo1:

Neste exercício iremos criar um programa que permita entrar com 4 notas de um aluno, armazená-las na memória, calcular a média aritmética dessas notas e então exibir essa média na tela.

Logo após, exibir também as notas que foram usadas no cálculo da média.

Var

NOTAS : **vetor [0..3]** de real

SOMA, MEDIA : real

i : inteiro

Inicio

SOMA <- 0

escreval ("Digite as quatro notas do aluno: ")

para i de 0 ate 3 faca

 leia (**NOTAS[i]**)

 soma <- soma + **NOTAS[i]**

fimpara

MEDIA <- soma / 4

escreval ("A média é ", **MEDIA**)

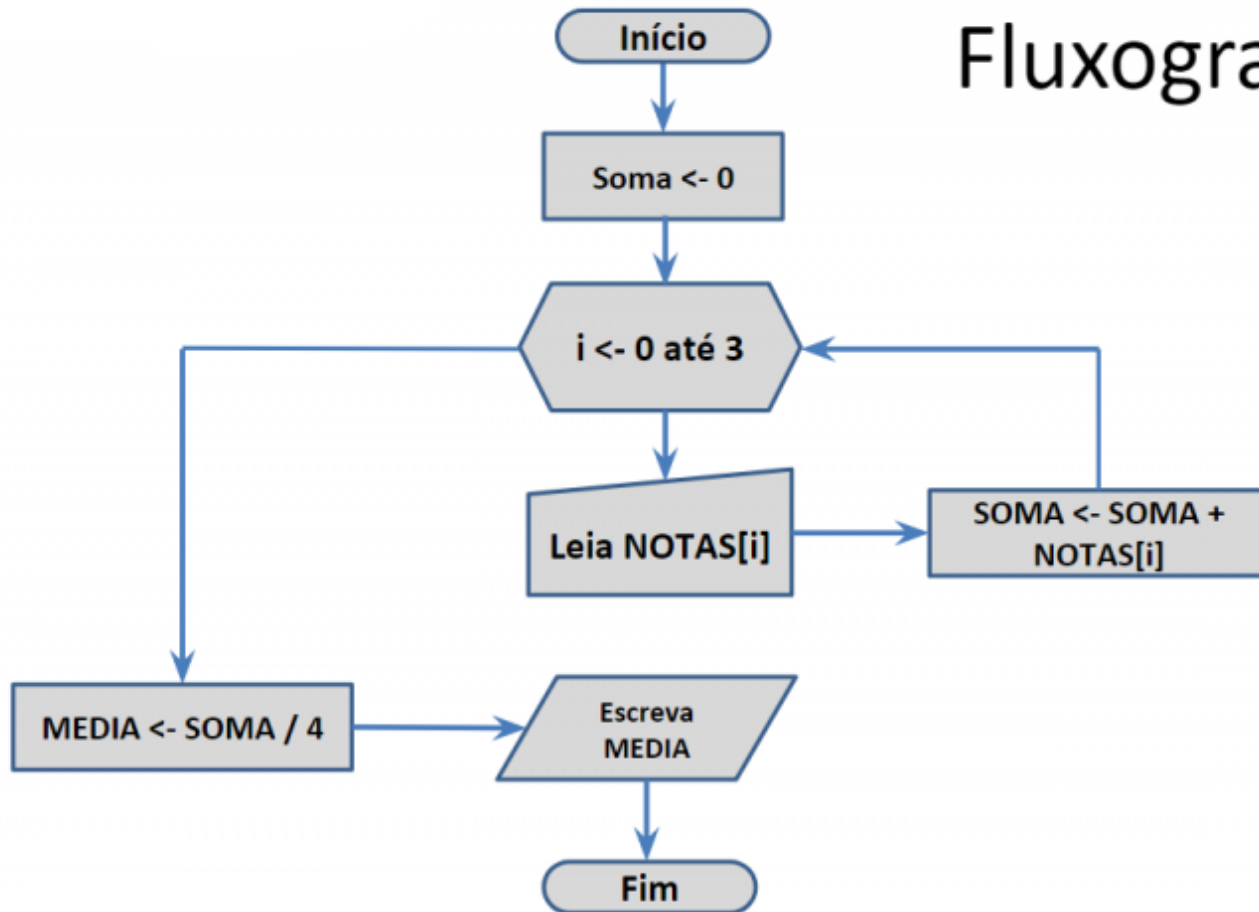
para i de 0 ate 3 faca

 escreval ("Nota ", **i** + 1, ": ", **NOTAS[i]**)

fimpara

Fimalgoritmo

Fluxograma



Um array (arranjo ou vetor) é um conjunto de dados (que pode assumir os mais diversos tipos, desde do tipo primitivo, a objeto dependendo da linguagem de programação). **Arrays são utilizados para armazenar mais de um valor em uma única variável.** Isso é comparável a uma variável que pode armazenar apenas um valor.

Exemplo2:

Neste exercício iremos criar um programa que leia 3 números inteiros e os armazene em um array. Após a leitura, some todos os elementos do array e exiba o resultado da soma:

algoritmo "Soma"

var

numeros: **vetor**[1..3] de **inteiro**

soma : **inteiro**

i: **inteiro**

inicio

soma <- 0

para i de 1 ate 3 faca

 escreva("Digite um número: ")

 leia(numeros[**i**])

soma <- **soma** + numeros[**i**]

fimpara

 escreva("A soma dos elementos é: ", **soma**)

fimalgoritmo

Agradeço atenção.

- Rafael Sacramento – rferfa@gmail.com
- Linkledin - <https://www.linkedin.com/in/rafael-do-sacramento-bomfim-9150784b/>
- Instagram - <https://www.instagram.com/rafaelrfe/>

“90% DO SUCESSO SE BASEIA EM INSISTIR”