

# Criando Algoritmos Computacionais

# Criando nosso primeiro código

- Para criar o nosso primeiro algoritmo no VisuAlg, precisamos escrever as instruções do nosso código entre (as palavras) **inicio** e **fimalgoritmo**
  - Essas palavras denotam, respectivamente, o início e o final de um algoritmo.
- O código será executado de forma sequencial (linha por linha, de cima para baixo), como apresentado na imagem abaixo.

inicio

<instrução #1>

<instrução #2>

...

<instrução #n>




fimalgoritmo

# Criando nosso primeiro código

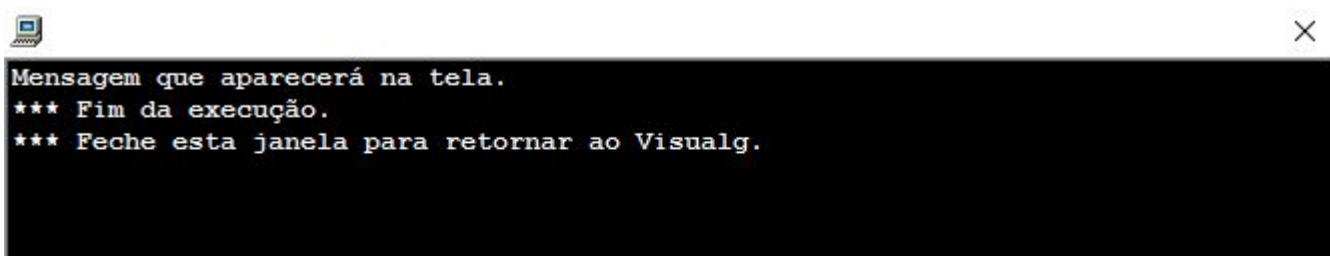
- Para testarmos se está tudo ok, vamos escrever um exemplo bem simples para fazer o VisuAlg imprimir um texto na tela.
- Para isso, usaremos um comando chamado **escreva**, que serve para escrever algo na tela do computador.
- Abaixo da palavra **inicio**, usamos o comando **escreva** e dentro dos () (parênteses) colocamos, entre aspas duplas, a mensagem que será impressa no Terminal

```
var  
  
inicio  
  escreva("Mensagem que aparecerá na tela.")  
fimalgoritmo
```



- Para executar o código, podemos ir no menu Algoritmo ➡ Executar e clicar no botão  ou, simplesmente, apertar a tecla F9 do teclado.

# Criando nosso primeiro código

- Feito isso, se o código estiver corretamente escrito, deve aparecer uma janela com a mensagem que passamos no comando **escreva**.



```
Mensagem que aparecerá na tela.  
*** Fim da execução.  
*** Feche esta janela para retornar ao Visualg.
```

- Para salvar o algoritmo, podemos ir no menu Arquivo → Salvar, clicar no botão  ou apertar as teclas Ctrl+S do teclado; escolha um nome para o arquivo e confirme.
- Para abrir um algoritmo salvo, podemos ir no menu Arquivo → Abrir, clicar no botão  ou apertar as teclas Ctrl+A do teclado; escolha o diretório onde arquivo foi salvo.

# Código Completo

## VisuAlg

```
algoritmo "mensagem"  
  
var  
  
inicio  
// escreve uma mensagem no terminal  
escreva("Mensagem que aparecerá na tela.")  
fimalgoritmo
```

## Portugol Studio

```
programa  
{  
    funcao inicio()  
    {  
        // escreve uma mensagem no terminal  
        escreva("Mensagem que aparecerá na tela.")  
    }  
}
```

# Criando nosso primeiro código

- O comando **escreva** serve para escrever algo em uma linha, caso coloque dois escreva, um abaixo do outro o comando vai imprimir as mensagens dos dois escreva, mas na mesma linha.

```
inicio  
escreva("Algo em uma linha")  
escreva("Algo em outra linha")  
finalgoritmo
```

- Para escrever duas mensagens, uma abaixo da outra, no VisuAlg, utilizamos o comando **escreval** que serve para que assim que for imprimida a mensagem “pular” uma linha
  - Assim, ao usar novamente o escreva ou escreval, o próximo texto vai ser impresso em uma nova linha.

```
inicio  
escreval("Imprimindo algo e quebrando uma linha")  
escreva("Imprimindo algo em outra linha")  
finalgoritmo
```

# Código Completo

## VisuAlg

**algoritmo** "escreve duas mensagens"

**var**

**inicio**

// escreve uma mensagem no terminal

escreval("Imprimindo algo e quebrando uma linha")

escreva("Imprimindo algo em outra linha")

**fimalgoritmo**

## Portugol Studio

**programa**

{

**funcao** inicio()

  {

    // escreve uma mensagem no terminal

    escreva("Imprimindo algo e quebrando uma linha\n")

    escreva("Imprimindo algo em outra linha")

  }

}

# Exemplos

1. Imprimindo números na tela:

```
inicio  
escreva(100)  
fimalgoritmo
```

2. Imprimindo o resultado de operações matemáticas na tela:

```
inicio  
escreva(2 + 2)  
fimalgoritmo
```

3. Imprimindo palavras e números na tela:

```
inicio  
escreva("2 vezes 2 é ", 2 * 2)  
fimalgoritmo
```



# Exercício

1. Crie um código que imprime a mensagem “Olá Mundo” na tela;
2. Crie um código que imprime a soma de três números;
3. Crie um código que imprime a mensagem “Quanto é  $2 + 2$ ?” e na linha seguinte imprima o resultado dessa operação matemática.

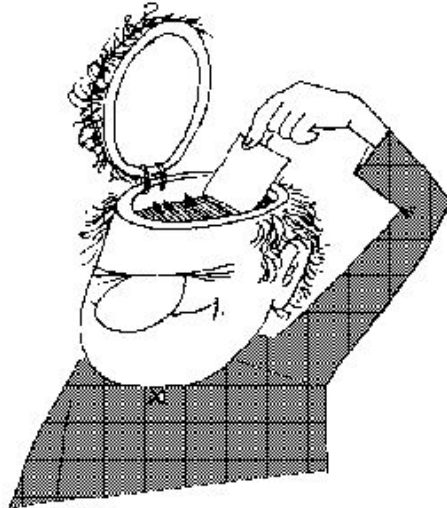
# Aprenda Mais...

- GuiTech: VisuAlg - Download e Instalação  
<<https://www.youtube.com/watch?v=dPV7fUuXEoA>>
- Portugol Studio: Portugol Studio Vídeo 01 - Introdução  
<<https://www.youtube.com/watch?v=K02TnB3lGnQ>>
- adrianoh2: Iniciando com Visualg  
<<https://www.youtube.com/watch?v=sU17rbKEPUA>>
- Tuto Studio: Visualg Aula 1 - Introdução  
<<https://www.youtube.com/watch?v=6-leAMCi8M>>

# Variáveis

# Variáveis

- “Variável” é um dos conceitos essenciais de programação!  
*“Variáveis são espaços situados na memória do computador onde podemos guardar valores ou expressões.”*
- Ou seja, se quisermos que o computador memorize um número para que possamos usá-lo poste alguma operação, “guardamos” esse memória utilizando uma variável.
- Toda variável vai precisar ter a seguinte estrutura:
  - Possuir um identificador (ou seja, um nome);
  - Possuir um tipo de dado;
  - Possuir um valor;



# Variáveis

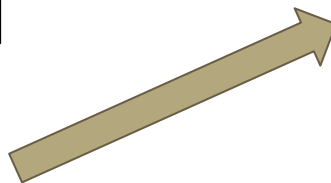
- Para compreender melhor o que são variáveis, vamos fazer uma analogia com objetos do nosso cotidiano:
- Vamos supor que temos um **armário**, no qual podemos guardar diversos objetos.
- E o nosso objetivo é guardar uma **bola** nesse armário.



# Variáveis

- Só que somos muito organizados em relação ao que guardamos no nosso armário
  - Assim, antes de colocarmos um objeto no nosso armário, colamos uma etiqueta na porta dizendo o **identificador** (o nome) desse objeto que está guardado naquela gaveta e de que **tipo** ele é.
- No caso da bola, colocaremos o **identificador** BOLA e dizemos que a bola é do **tipo** Brinquedo.

BOLA : Brinquedo



# Variáveis

- Caso quiséssemos colocar um sapato no armário, poderíamos colar uma etiqueta como:

SAPATO : Calçado



# Variáveis

- E se quiséssemos guardar o número 2:

NUMERO: Inteiro

2





# Variáveis

- Nessa analogia, pode-se entender o armário como a **memória do computador**, onde podemos “guardar” valores para serem usados posteriormente, mas esses valores necessitam ser identificados por um nome e por um tipo.
  - As **variáveis** são os espaços no armário.
  - As etiquetas são o **identificador** (ou nome) que usamos para nos referir a cada espaço vazio (i.e., cada variável)
  - O **tipo** corresponde a que objetos (valores) podem ser colocados naquele espaço.
  - O **valor** corresponde ao que está guardado naquele espaço do armário.



# Aprenda mais

- O que é variável e constante?  
<<http://www.dicasdeprogramacao.com.br/o-que-e-variavel-e-constante/>>
- Curso Introdução a Programação: O que é variável  
<[https://www.youtube.com/watch?v=8tKl\\_yppKmc](https://www.youtube.com/watch?v=8tKl_yppKmc)>
- Khan Academy em Português: O que é uma variável?  
<<https://www.youtube.com/watch?v=-ZMCNZXmzZk>>
- Hora do Código <<http://silentteacher.toxicode.fr/hourofcode>>

# Identificadores

- Os Identificadores são nomes que damos as variáveis. Eles servem, como o próprio nome diz, para identificar uma variável. Isto é importante pois as variáveis são algo que manipulamos frequentemente no nosso código e precisamos do identificador delas para isso.
- No VisuAlg, para nomearmos uma variável, ou melhor, para escolhermos o identificador para uma variável, devemos respeitar algumas regras:
  1. O nome deve começar com uma **letra**;
  2. Os próximos caracteres do nome podem ser **letras** ou **números**;
  3. **Não** pode usar **símbolos**, exceto **\_** (underline ou sublinhado)\*;
  4. **Não** pode ter espaços em branco;
  5. **Não** pode ter letras **acentuadas**;
  6. **Não** pode ser uma **palavra reservada**\*\*;

\* Geralmente o **\_** nos identificadores são usados para representar um espaço em branco, caso o identificador da variável tenham mais de uma palavra Ex: `nota_do_aluno`.

\*\* Veremos logo adiante o que são palavras reservadas.



# Palavras reservadas

- As linguagens de programação especificam algumas palavras reservadas, tais palavras são definidas na linguagem para que o programador possa usá-las no futuro.
- Geralmente elas são comandos que executam determinada ação como: se; senão; para; enquanto.\*
- Consequentemente, elas podem também definir uma estrutura de código com vimos na estrutura do editor de código do VisuAlg. As palavras **algoritmo**, **var**, **inicio** e **fimalgoritmo** são palavras reservadas do VisuAlg.
- As vezes é fácil identificar as palavras reservadas pois ALGUMAS delas têm cores diferentes.

\* Estudaremos cada um desses comandos durante o curso..

# Palavras reservadas

- Abaixo segue uma tabela com todas as palavras reservadas do VisuAlg:

aleatorio	caracter	e	fimalgoritmo	grauprad	maiusc	passo	randi
abs	caso	eco	fimenquanto	inicio	mensagem	pausa	repita
algoritmo	compr	enquanto	fimescolha	int	minusc	pi	se
arccos	copia	entao	fimfuncao	interrompa	nao	pos	sen
arcsen	cos	escolha	fimpara	leia	numerico	procedimento	senao
arctan	cotan	escreva	fimprocedimento	literal	numpcarac	quad	timer
arquivo	cronometro	exp	fimrepita	log	ou	radpgrau	tan
asc	debug	faca	fimse	logico	outrocaso	raizq	verdadeiro
ate	declare	falso	função	logn	para	rand	xou

# Exemplos de Identificadores

- Abaixo segue uma tabela com exemplos de identificadores válidos e inválidos para nomear-se uma variável.

numeral	válido
numero1	válido
primeiro_número	válido
numero_2	válido
soma	válido

1numero	inválido
1ºnumero	inválido
primeiro número	inválido
algoritmo	invalido
variável	inválido

# Exercício

1. Marque com (V) os identificadores válidos ou (I) os identificadores inválidos:

☐ Ano

☐ salário

☐ 2/1

☐ #media

☐ nota1

☐ numero\_rg

☐ R\$

☐ verdadeiro

☐ telefone\_numero1

# Tipos de Dados

- Toda variável precisa ter um tipo definido para ela, o qual restringe os valores que a variável pode assumir. Se eu digo que a variável NUMERO é do tipo inteiro, ela não pode assumir, por exemplo, o valor "1,55", pois esse número não é inteiro.
- O tipo da variável é definido, em Portugol, no momento de sua declaração.





# Tipos de Dados

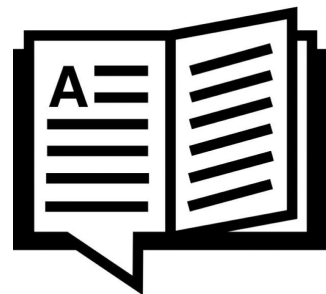
- Para representar tipos de dados simples no VisuAlg e no Portugol Studio nós temos:

Tipos de Dados Simples			
Tipo		Definição	Valor Inicial*
caractere (ou <i>cadeia</i> no Portugol Studio)		Usado para representar textos.	"" (caracteres vazios)
Tipos Numéricos	inteiro	Usado para representar números inteiros.	0 (zero)
	real	Usado para representar números reais.	
logico		Usado para representar verdadeiro ou falso.	FALSO

\* Veremos o que são valores iniciais futuramente.

# Cadeia de Caracteres

- Como o próprio nome diz, é o tipo que define que nossa variável tem como valor uma cadeia de caracteres, ou seja um texto. Sempre definimos o seu valor entre "" (aspas duplas).
- Algumas linguagens de programação chamam esse tipo de dado de string (que em inglês quer dizer justamente cadeia).
- Usamos o tipo **caractere** geralmente em variáveis que representam um texto ou uma palavra:
  - nome\_pessoal
  - endereco
  - mensagem\_boas\_vindas



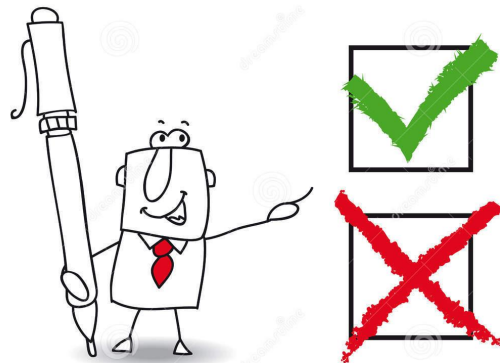
# Tipos Numéricos

- Os tipos numéricos definem que nossa variável pode receber valores numéricos que podem ser números inteiros (usando o tipo inteiro) ou números reais (usando o tipo real).
- O tipo **inteiro** é usado geralmente em variáveis que recebem apenas números exatos ou seja não separados por vírgula:
  - ano
  - idade
  - numero\_de\_itens
- O tipo **real** é usado geralmente em variáveis que assumem valores com precisão decimal:
  - altura
  - peso\_medio
  - valor\_em\_reais



# Tipos Lógicos

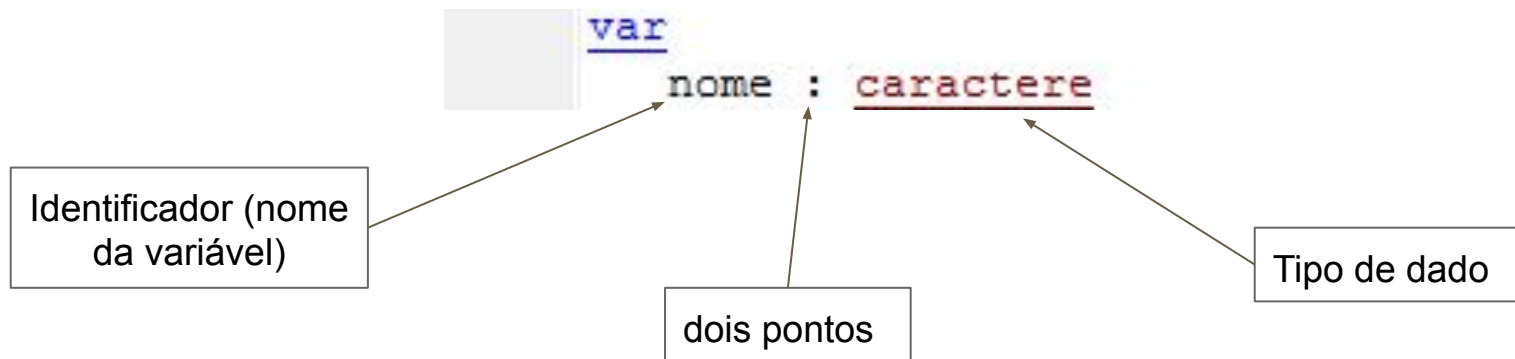
- Os tipos lógicos definem que nossas variáveis devem receber valores como **Verdadeiro** ou **Falso**, que são muito utilizados na programação para trabalhar com **condições\***.
- Em outras linguagens os tipos lógicos são chamados de booleanos (ou boolean, em inglês) que podem ser representados por valores: TRUE ou FALSE; 0 ou 1; "" (texto vazio) ou "!"@#" (texto com algo dentro). Mas no português trabalhamos apenas com Verdadeiro e Falso.
- O tipo lógico geralmente é usado cujo o valor representa uma pergunta de sim ou não (é verdadeiro ou é falso):
  - ensino\_fundamental\_completo
  - tem\_diabetes
  - esta\_cadastrado



\* Veremos o que são condições futuramente em Estruturas Condicionais

# Declarando variáveis no VisuAlg

- Abaixo mostramos a estrutura necessária para declaração de uma variável no VisuAlg, nós as declaramos logo abaixo da palavra var e antes do inicio do algoritmo colocando o seu identificador seguido de : (dois pontos) e o seu tipo de dado:



# Declarando variáveis no VisuAlg

- Abaixo, seguem alguns exemplos de declarações de variáveis:

```
var  
  nome : caractere    // Nome da pessoa  
  idade : inteiro      // Idade da pessoa  
  altura : real        // Altura da pessoa  
  aposentado : logico // É aposentado?  
inicio
```

- É possível declarar várias variáveis do mesmo tipo em uma única linha apenas separando-as por , (vírgula).

```
var  
  nome, sobrenome: caractere  
  peso, altura : real
```



# Exercício

1. Associe os tipos de dado às variáveis que mais façam sentido, segundo a nomenclatura de seus identificadores.

- a) nome ( )
- b) telefone ( )
- c) idade ( )
- d) possui\_deficiencia ( )
- e) saldo\_bacario ( )
- f) nacionalidade ( )
- g) peso ( )

(A) caractere

(B) inteiro

(C) real

(D) logico

# Aprenda mais...

- Node Studio Treinamentos: Lógica de Programação - Aula 03 - Tipos de dados, Variáveis, Constantes e Instruções  
<<https://www.youtube.com/watch?v=-ny7Kqm0V68>>
- RBtech: Lógica de programação - Aula 04 - Variáveis e constantes  
<[https://www.youtube.com/watch?v=vp4jgXA\\_BB0](https://www.youtube.com/watch?v=vp4jgXA_BB0)>
- Computação Depressão: Curso Lógica de Programação - Aula 05 - O que são variáveis? <<https://www.youtube.com/watch?v=JEHv8eF3til>>
- Bóson Treinamentos: 04 - Lógica de Programação - Variáveis e Tipos de Dados <<https://www.youtube.com/watch?v=IN2XgPTLewg>>



# Valores

- Após **declarar** uma variável (identificar e definir o tipo), precisamos atribuir um valor para ela (um valor que condiga com o seu tipo).
- Por exemplo, ao declararmos a variável *meu\_nome* do tipo *caractere*, podemos atribuir “Josevaldo da Silva” como o valor dela.
- Da mesma forma que, quando declaramos a variável *saldo\_bancario* do tipo *real*, podemos atribuir 2.453 como o valor dela.
- Como também, quando declaramos *possui\_cartao\_fidelidade* do tipo *logico*, podemos atribuir *VERDADEIRO* como valor dela.

```
meu_nome <- "Josevaldo da Silva"  
saldo_bancario <- 2.453  
possui_cartao_fidelidade <- VERDADEIRO
```

O símbolo <- (seta) é utilizado para atribuir um valor a uma variável. Você pode “lê-lo” no código com o sentido de “recebe”.  
ex: meu\_nome “recebe” o valor “Josevaldo da Silva”

# Valores Iniciais

- Quando declaramos uma variável sem atribuir qualquer valor para ela, ela assume seu valor padrão ou valor inicial.
  - Para **caractere**, o valor inicial é de uma cadeia de caracteres vazia. O que seria o mesmo de declarar a variável e atribuir o valor "" (aspas sem nenhum texto dentro).
  - Para os valores numéricos (**inteiro** e **real**), o valor inicial é 0 (zero). Qualquer operação com uma variável numérica sem atribuirmos um valor para ela, é mesmo que realizar essa operação com o número zero.
  - Para o tipo **lógico**, o valor inicial é FALSO. Ou seja qualquer variável lógica inicia com valor FALSO até mudarmos esse valor para VERDADEIRO.

# Atribuindo valores a uma variável no VisuAlg

- Para atribuirmos um valor a uma variável em um algoritmo, utilizamos operador de atribuição. O operador de atribuição do VisuAlg é representado por uma seta <- (“menor que” seguido de hífen) apontando para a esquerda.
- Como todas as operações do programa, exceto as declarações de variáveis, sempre atribuímos valores às variáveis entre o [inicio](#) e o [finalgoritmo](#).

```
inicio  
nome <- "Leonardo"
```

- No exemplo, pegamos a variável nome, usamos o operador de atribuição <- e demos o valor “Leonardo”, já que ela é do tipo caractere.

# Mudando o valor de uma variável no VisuAlg

- Os valores das variáveis são mutáveis (daí o nome “variável”), ou seja podemos mudar seus valores a qualquer momento e quantas vezes quisermos, mesmo depois que o programa esteja em execução.
- Por exemplo, uma variável que representa o número de gols em jogo de futebol: a cada gol marcado essa variável teria que mudar seu valor.
- No VisuAlg, o código a seguir ilustra isso:

```
var  
  numero_gols : inteiro  
inicio  
  numero_gols <- 1  
  numero_gols <- 2  
fimalgoritmo
```

- No exemplo, atribuímos o valor 1 para a variável *numero\_gols* e depois mudamos o seu valor para 2.

# Exemplos

- Abaixo, seguem alguns exemplos de declaração de variáveis e atribuição dos seus respectivos valores.

```
var
  nome : caractere
  idade : inteiro
  peso, altura : real
  graduado : logico
inicio
  nome <- "Leonardo"
  idade <- 29
  peso <- 1.5
  graduado <- Verdadeiro
fimalgoritmo
```

# Valores devem corresponder aos tipos

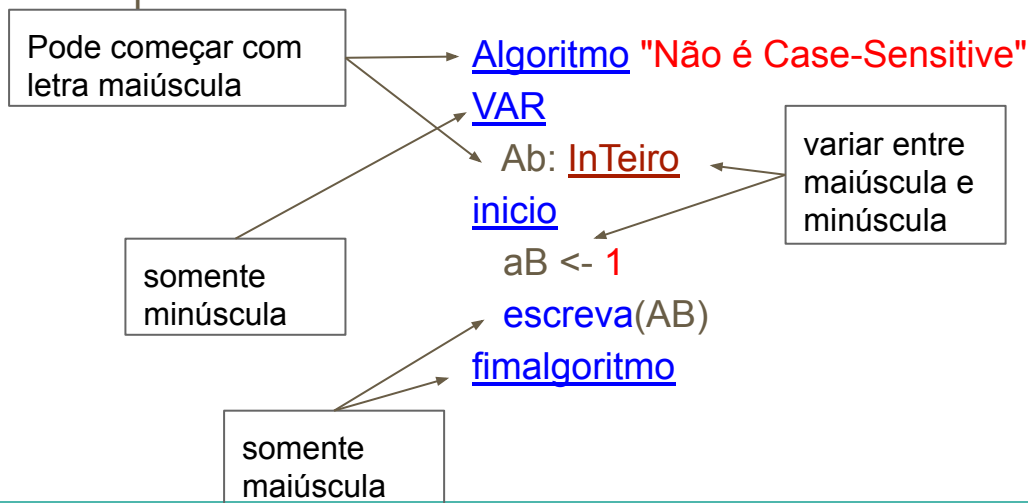
- É importante salientar que **só devemos atribuir às variáveis valores que correspondam ao tipo de dado**. Assim, o seguinte comando ocasionaria um erro em “tempo de execução”:

```
var  
    saldo : real  
inicio  
    saldo <- "Insuficiente"  
fimalgoritmo
```

- No caso, o erro foi tentarmos atribuir um valor de texto em uma variável do tipo real.

# VisuAlg não é Case-Sensitive

- Tantos os comandos como as variáveis no VisuAlg não são case-sensitive, ou seja, tanto faz se você escrever algum comando, palavra-chave, nome de variável, nome de rotina em caixa alta ou caixa baixa. O VisuAlg vai “compreender” o que você escreveu.
- Logo, o código a seguir compila sem problemas:
- Também é possível declarar uma variável ou rotina com o identificador variado com letras minúsculas e maiúsculas e atribuir ou fazer operações com esse mesmo identificador com caixa alta, caixa baixa ou variando.



# Aprenda Mais...

- Lógica com VisuAlg 3.0 - Tipos de Dados, disponível em:  
<<https://www.youtube.com/watch?v=mENTOK6lB2s>>
- Atribuição (computação) - Wikipédia, disponível em:  
<[https://pt.wikipedia.org/wiki/Atribui%C3%A7%C3%A3o\\_\(computa%C3%A7%C3%A3o\)](https://pt.wikipedia.org/wiki/Atribui%C3%A7%C3%A3o_(computa%C3%A7%C3%A3o))>
- Lógica de Programação Para Iniciantes (Definir, Imprimir e Ler Variáveis) - VisuALG #01  
<[https://www.youtube.com/watch?v=rbu7-Uy9\\_Eo](https://www.youtube.com/watch?v=rbu7-Uy9_Eo)>



# Constantes

- Conceitualmente, na programação, constantes são variáveis cujo seu valor é fixo (não pode ser mudado).
- Elas assumem um valor no início do programa e não é possível mudá-lo. Por exemplo a velocidade da luz que é 300.000 Km/s, esse valor não se altera, então poderíamos colocar em uma constante.

```
var  
    velo_da_luz: inteiro  
inicio  
    // 300.000 Km/s  
    velo_da_luz <- 300000  
fimalgoritmo
```

- Existem muitas “constantes” famosas, como o valor de PI, por exemplo.
- Algumas linguagens de programação definem regras específicas para declarar uma constante e assim diferenciar bem elas das variáveis comuns.
- Entretanto, infelizmente, no caso do VisuAlg, não existe sintaxe para declarar uma constante.

# Constantes no Portugol Studio

- Diferente do VisuAlg, o Portugol Studio define um padrão para se declarar uma constante.
- Isso é feito com a palavra reservada **const**, seguido de seu tipo e identificador. Além disso, as constantes necessitam que coloquemos algum valor inicial no momento em que a declaramos.

```
funcao inicio()  
{  
    // constante velocidade da Luz = 300.000 km/s  
    const inteiro vel_da_luz = 300000  
}
```

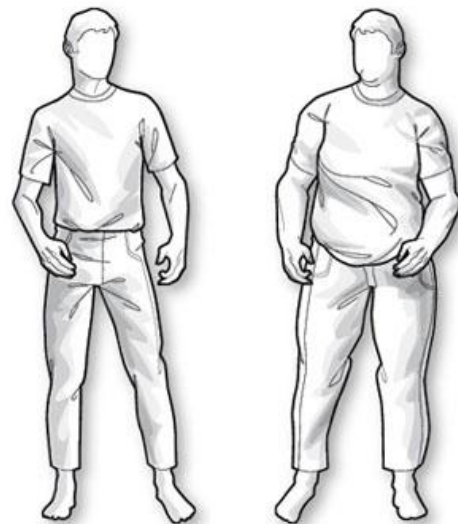
# Aprenda mais...

- Constantes  
<[http://www.sj.ifsc.edu.br/~mello/livros/portugol/manual-portugol/tipo\\_dados\\_constantes.html](http://www.sj.ifsc.edu.br/~mello/livros/portugol/manual-portugol/tipo_dados_constantes.html)>
- Constante (programação)  
<[https://pt.wikipedia.org/wiki/Constante\\_\(programa%C3%A7%C3%A3o\)](https://pt.wikipedia.org/wiki/Constante_(programa%C3%A7%C3%A3o))>

# Entrada e Saída (E/S) de Dados

# Entrada e Saída de Dados

- Geralmente, os programas recebem dados a partir de uma fonte (como o usuário do programa) para fazer algum processamento e gerar uma saída.
- E/S de dados consiste em comandos que o permitem a interação com o usuário, através de dispositivos de dispositivos de entrada (como o teclado do computador) e saída (como o monitor de um computador).
- Um exemplo é um programa para calcular o IMC (índice de massa corporal), onde o usuário informaria como **entradas** o peso e altura, e o programa gerará uma **saída** e exibirá o seu índice de massa corporal.



# Comandos de Saída

- No VisuAlg, existem dois comando para saída padrão, os quais já vimos antes. São eles: o **escreva** e o **escreval**, como já sabemos tanto o escreva como o escreval imprimem na tela do computador uma mensagem, além disso, esses comandos podem imprimir também valores de variáveis e resultados de expressões.
- Só lembrando que diferente do escreva, o escreval imprime e depois “pula” uma linha, ou seja, se tentarmos escrever na tela algo **depois** de já ter escrito com um escreval, esta nova mensagem será impressa em uma nova linha, abaixo da anterior.

# Imprimindo valores de variáveis

- Agora, vamos usar o comando de saída escreva para imprimir os valores de nossas variáveis e para isso é simples, só colocar a variável dentro dos parênteses:

```
var  
    saldo: real  
inicio  
    valor <- 12367.13  
    escreva(valor)  
fimalgoritmo
```

- Isso vai imprimir o valor **12367.13** na tela, ou seja o valor que guardamos na variável **saldo**.
- Também é possível usar uma , (vírgula) para separar textos das variáveis e de expressões dentro do comando.

```
inicio  
    saldo <- 12367.13  
    escreva("Seu saldo atual é: ", saldo)  
fimalgoritmo
```

# Código Completo

## VisuAlg

**algoritmo** "Saldo da Conta Bancária"

**var**

saldo: real

**inicio**

// atribui o valor a variável saldo

saldo <- 12367.13

// imprime uma mensagem com o valor da variável

escreva("Seu saldo atual é: ", saldo)

**fimalgoritmo**

## Portugol Studio

**programa**

{

**funcao** inicio()

  {

    // atribui o valor a variável saldo

**real** saldo

    // imprime uma mensagem com o valor da variável

    saldo = 12367.13

**escreva**("Seu saldo atual é: ", saldo)

  }

}



# Exercício

1. Crie uma variável do tipo caractere chamada **nome** e atribua seu nome para ela.
2. Crie algumas variáveis referentes às características de um carro. Ex. marca, cor, tem\_quatro\_portas, etc. Atribua valores que condizem com os identificadores dessas variáveis.
3. Crie um código onde que tem a variável boas\_vindas atribua o texto “Olá Mundo” a essa variável e depois imprima essa mensagem na tela com o comando escreva.

# Comandos de Entrada

- No VisuAlg, existe um único comando da entrada padrão que é o **leia**, que serve para pedir valores digitados pelo usuário e atribuí-los a variáveis.
  - O comando tem a seguinte sintaxe **leia**(variavel\_que\_recebera\_o\_valor);
- Ao chamar o comando **leia**, o programa vai esperar um valor ser digitado pelo usuário
  - Quando em execução, o programa não executará os próximos comandos até que o usuário digite algum valor e tecele Enter.

```
var  
    saldo : real
```

```
inicio  
    leia(saldo)  
    escreva("Seu saldo é", saldo)  
fimalgoritmo
```

respondente ao tipo da variável.

# Lendo valores e atribuindo-os às variáveis

- Podemos, também, colocar uma mensagem antes do **leia** para informar para o usuário o que o programa está pedindo.

```
inicio  
    escreva("Digite seu saldo: ")  
    leia(saldo)  
    escreva("Seu saldo é", saldo)  
fimalgoritmo
```

- Também é possível usar o leia para receber vários valores de uma vez atribuindo-os às variáveis separando-as por , (vírgula) no comando.

```
var  
    peso, altura : real  
inicio  
    escreva("Digite seu peso e logo após a sua altura ")  
    leia(peso, altura)  
    escreva("Seu peso é ", peso, " e sua altura é ", altura)  
fimalgoritmo
```

# Exemplos

## 1. Lendo caracteres:

```
var  
    nome : caractere  
inicio  
    escreva("Qual é o seu nome: ")  
    leia(nome)  
    escreva("Bem vindo ", nome, "!!")  
fimalgoritmo
```

## 2. Lendo tipo lógico:

```
var  
    tem_experiencia : logico  
inicio  
    escreva("Já tem experiencia na área: ")  
    leia(tem_experiencia)  
fimalgoritmo
```

# Exercício

1. Crie um programa em que peça ao usuário o seu nome usando o comando `leia`, depois imprima uma mensagem de boas vindas, tipo: "Bem vindo <Nome do Usuário>".
2. Crie um programa que peça ao usuário seu nome, sua idade e o nome da cidade onde ele mora, depois imprima a mensagem: "Olá meu nome é <Nome do Usuário> tenho <idade> e moro em <cidade>".

# Aprenda mais...

- Aula 8 - Comandos de Entrada e Saída  
<<https://pt.slideshare.net/LuizAugustoMacdoMorais/aula-8-comandos-de-entrada-e-sada-9596065>>
- Weslei Felix - Aula 04 Entrada e Saída de Dados VisuAlg  
<<https://www.youtube.com/watch?v=lkFhxuAdxC8>>
- Bóson Treinamentos - 07 - Lógica de Programação - Comandos de Entrada e Saída de Dados <<https://youtu.be/lrSEggh6GQA>>

# Expressões Aritméticas

# Linearização de Expressões

- Para a construção de algoritmos que realizam cálculos matemáticos, todas as expressões aritméticas devem ser linearizadas, ou seja, colocadas em linhas, devendo também ser feito o mapeamento dos operadores da aritmética tradicional para os do Português Estruturado.

Tradicional	Computacional
$\left\{ \left[ \frac{2}{3} - (5 - 3) \right] + 1 \right\} . 5$	<code>((2/3 - (5 - 3)) + 1) * 5</code>



# Linearização de Expressões Aritméticas

- As tabelas seguintes mostram os operadores aritméticos disponíveis no Português Estruturado.

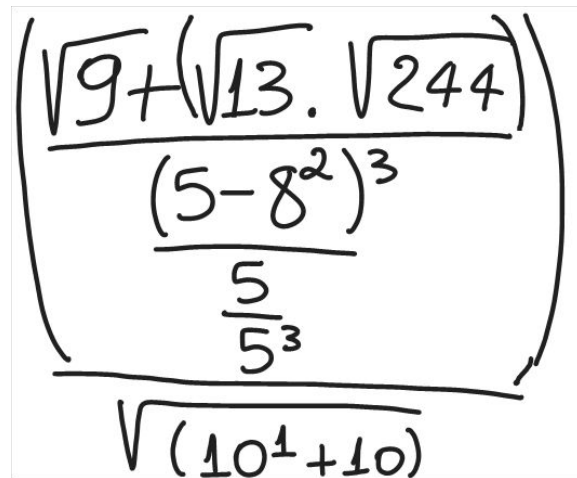
OPERADORES ARITMÉTICOS	PORTUGUÊS ESTRUTURADO
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Divisão Inteira	\
Exponenciação	^ ou Exp (<base>,<expoente>)
Módulo (resto da divisão)	%

# Ordem de Precedência

- Na programação, operações com números seguem a mesma ordem de precedência das expressões numéricas da aritmética, respeitando a seguinte ordem:

- 1º operações dentro dos parênteses indo das mais internas para as mais externas;
- 2º operações de potencia e raiz;
- 3º depois multiplicação e divisão;
- 4º adição e subtração;

- Importante ressaltar que: na programação, os níveis de importância das operações são representando apenas por () (parênteses), ou seja não usamos [] (colchetes) ou {} (chaves).



A handwritten mathematical expression enclosed in large parentheses, illustrating the order of operations. The expression is: 
$$\frac{\sqrt{9 + (\sqrt{13} \cdot \sqrt{244})}}{\frac{(5 - 8^2)^3}{\frac{5}{5^3}}}$$
 The expression is written in a way that shows the hierarchy of operations: innermost parentheses first, then exponents, then multiplication and division, and finally addition and subtraction.

# Ponto Flutuante

- Como vimos antes, os números reais na linguagem de computação são usados para representar medidas, preços e afins, pois são números que possuem uma parte inteira e uma parte fracionária, ou também podemos chamar de números de ponto flutuante.
- Diferente do que estamos acostumados, em diversas linguagens de computação, assim como no VisualG e Portugol Studio, separamos a parte inteira da parte fracionária de um número do tipo **real** com o **ponto** ao invés de vírgula. Ou seja, usamos a notação de países de cultura inglesa para representar esses números.

COMO ESTAMOS HABITUADOS	COMO DEVEMOS USAR NA COMPUTAÇÃO
2,50	2.50

# Operações com Variáveis

- Como já sabemos, as variáveis podem guardar valores de diversos tipos, ou seja, é possível também fazer operações usando variáveis, como também é possível guardar os valores dessas operações em outras variáveis.

```
algoritmo "Operações com Variáveis"
var
    numero1: inteiro
    numero2: inteiro
    resultado_soma: inteiro
inicio
    numero1 <- 2
    numero2 <- 3
    resultado_soma <- numero1 + numero2
    escreva(resultado_soma)
fimalgoritmo
```

- No exemplo, foram criadas três variáveis do tipo **inteiro**, duas para representar os números que farão parte da operação e uma para receber o resultado que será também do tipo **inteiro**.

# Operações com Variáveis

- Quando tentamos atribuir o resultado de uma operação a uma variável, devemos ter cuidado com o possível valor resultante dessa operação. Em casos que sabemos que o valor resultante é um número decimal devemos atribuir o resultado dessas operações em uma variável do tipo real.
- No VisuAlg, as operações de divisão, potenciação e raiz resultam em um valor decimal portanto sempre devemos atribuir o valor resultante dessas operações em uma variável do tipo real.
- No caso, em uma divisão (3 dividido por 2) cujo o valor do resultado é 1,5, só podemos guardar esse valor em uma variável do tipo **real**.

```
algoritmo "Operação com Variáveis"  
  
var  
    numero1: inteiro  
    numero2: inteiro  
    resultado_divs: real  
  
inicio  
    numero1 <- 2  
    numero2 <- 3  
    resultado_divs <- numero2 / numero1  
    escreva(resultado_divs)  
fimalgoritmo
```

# Código Completo

## VisuAlg

**algoritmo** "Operações com Variáveis"

**var**

numero1: inteiro

numero2: inteiro

resultado\_soma: inteiro

**inicio**

// atribui os valores as variáveis

numero1 <- 2

numero2 <- 3

// soma os valores e atribui a soma a variável

resultado\_soma <- numero1 + numero2

// imprime uma mensagem com o valor da variável

escreva(resultado\_soma)

**fimalgoritmo**

## Portugol Studio

**programa**

{

**funcao** inicio()

  {

**inteiro** numero1

**inteiro** numero2

**inteiro** resultado\_soma

    // atribui os valores as variáveis

    numero1 = 2

    numero2 = 3

    // soma os valores e atribui a soma a variável

    resultado\_soma = numero1 + numero2

    // imprime uma mensagem com o valor da variável

    escreva(resultado\_soma)

  }

}

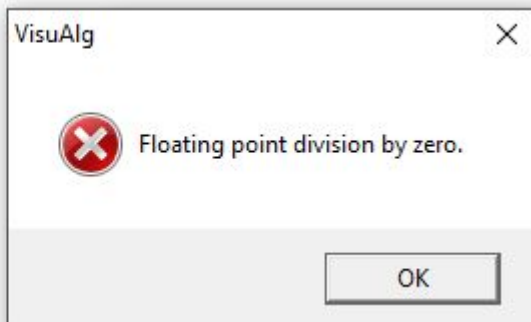
# Exemplos

- Expressão com vários operadores e subníveis:

```
inicio  
  num1 <- 2  
  num2 <- 4  
  num3 <- 6  
  resultado <- ((num2 + 8) / num1) + (num3 + 2) * 2  
  escreva(resultado)  
fimalgoritmo
```

- Divisão por zero, resulta em erro:

```
var  
  num1, num2: inteiro  
  resultado: real  
inicio  
  num1 <- 2  
  num2 <- 0  
  resultado <- num1 / num2  
  escreva(resultado)  
fimalgoritmo
```



# Exemplos

- Para potenciação, pode-se usar o comando Exp(<base>, <expoente>):

```
var
  num1, num2: inteiro
  resultado: real
inicio
  num1 <- 4
  num2 <- 2
  // Faz a operação de 4² (quatro ao quadrado)
  resultado <- Exp(num1, num2)
  escreva(resultado)
fimalgoritmo
```

- Para raiz quadrada, pode-se usar o comando Raizq(<expressão>):

```
var
  num1: inteiro
  resultado: real
inicio
  num1 <- 4
  // Faz a operação de raiz quadrada de 4
  resultado <- Raizq(num1)
  escreva(resultado)
fimalgoritmo
```



# Exercício

1. Reescreva a seguinte expressão  $\{2 \times 5 - [7 - (3 - 1)] \div 2\}$  de forma linear.
2. Reescreva a seguinte expressão  $\{2 + [(\sqrt{16} + 2^3)]\} \times 3$  de forma linear.
3. Escreva um programa para a calcular a potenciação de um número e guarde-o em uma variável, em seguida imprima a raiz quadrada dessa variável.
4. Sabendo que a formula de delta é  $b^2 - 4ac$ , crie um programa que tenha 3 variáveis do tipo inteiro:  $a = 1$ ,  $b = -3$  e  $c = -10$ . O programa deve calcular o valor de delta e imprimir na tela.
5. Escreva um programa que peça ao usuário para informar o peso e a altura com o comando **leia**, depois calcule e imprima na tela o IMC dessa pessoa sabendo que a fórmula do IMC é  $(\text{peso}/\text{altura}^2)$ .

# Aprenda Mais...

- 06 - Lógica de Programação - Operadores e Expressões Aritméticas  
<<https://www.youtube.com/watch?v=eH9Prly92BU>>
- Aula 2 - Programação em Portugol - Uso de Variáveis e Operações Matemáticas  
<[https://www.youtube.com/watch?v=ooP\\_lhkaZwc](https://www.youtube.com/watch?v=ooP_lhkaZwc)>
- Comando de Entrada e Operadores - Curso de Algoritmos #03 - Gustavo Guanabara  
<<https://www.youtube.com/watch?v=RDrfZ-7WE8c>>