





# Pseudocódigo

Também chamado de “***Portugol***”, é a lógica do seu programa escrito na sua linguagem nativa, em nosso caso, português.

**Portugol não é uma linguagem de programação**, e sim uma maneira de representar a sua lógica de programação antes de passar para uma linguagem de programação.

Usaremos o **VisualG** para criar nossos pseudocódigos.

[VISUALG 3.0 download | SourceForge.net](#)

```

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 : 04/10/2023
7 Var
8 // Seção de Declarações das variáveis
9 |
10
11 Inicio
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13
14
15 Fimalgoritmo

```



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

```
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      : 04/10/2023  
7 Var  
8 // Seção de Declarações das variáveis  
9 |  
10  
11 Inicio  
12 // Seção de Comandos, procedimento, funções, operadores, etc...  
13  
14  
15 Fimalgoritmo
```

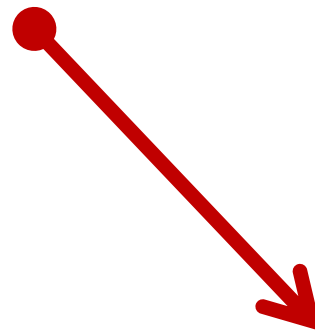
Para estrutura padrão automaticamente:

**! + CTRL + ESPAÇO**

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

```
1 algoritmo "semnome"  
2 var  
3 |  
4 inicio  
5  
6 fimalgoritmo
```

## Escreva (“Qualquer coisa”)

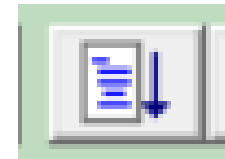


Tudo que estiver  
entre aspas é  
considerado texto,  
uma mensagem.

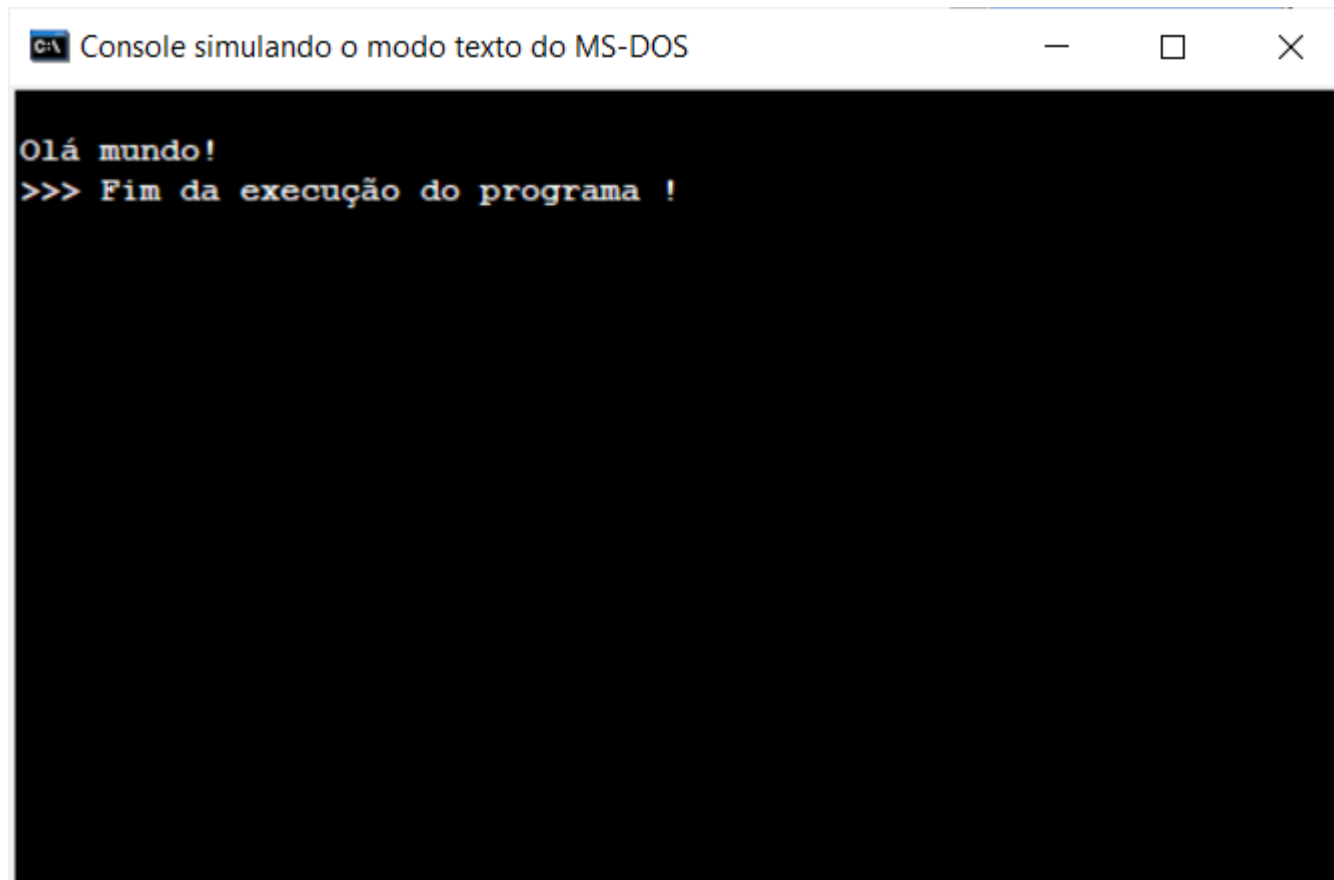
```
1 algoritmo "primeiro"  
2 var  
3  
4 inicio  
5     Escreva("Olá mundo!")  
6 fimalgoritmo
```



- 1) Menu **Run (executar)** – Rodar o algoritmo
- 2) Atalho **F9**
- 3) Ícone na barra de ferramentas



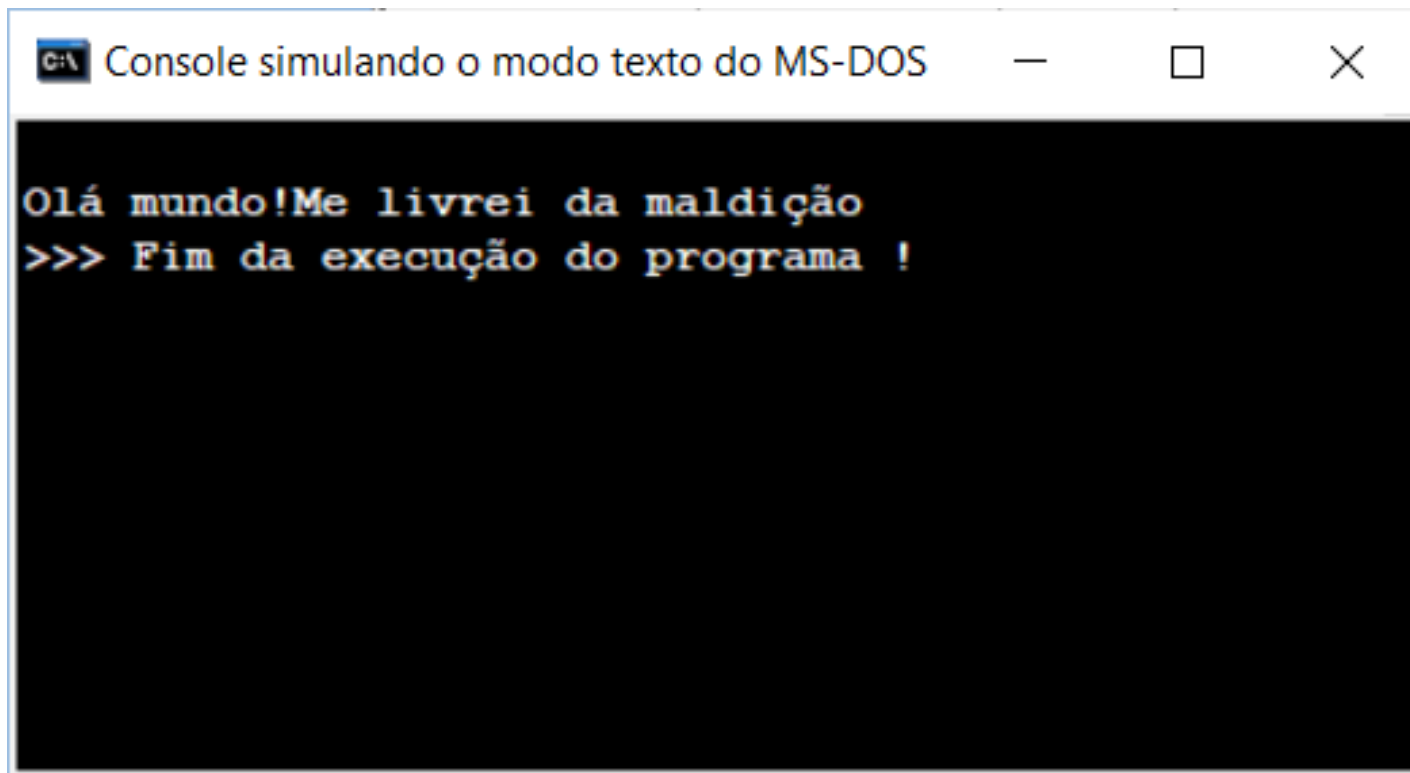
# Executar Programa

A screenshot of a console window titled "C:\ Console simulando o modo texto do MS-DOS". The window has a black background and white text. It displays two lines of output: "Olá mundo!" followed by a blank line, and then ">>>> Fim da execução do programa !". The window includes standard Windows-style window controls (minimize, maximize, close) in the top right corner.

```
C:\ Console simulando o modo texto do MS-DOS

Olá mundo!
>>> Fim da execução do programa !
```

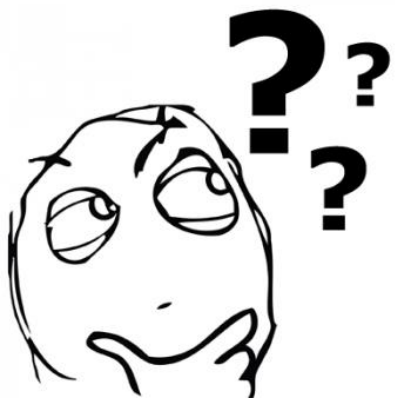
```
1 algoritmo "primeiro"  
2 var  
3  
4 inicio  
5     Escreva("Olá mundo!")  
6     Escreva("Me livrei da maldição")  
7 fimalgoritmo
```

A screenshot of a DOS console window titled "C:\ Console simulando o modo texto do MS-DOS". The window has a black background and displays two lines of text in a monospaced font: "Olá mundo!Me livre da maldição" and ">>> Fim da execução do programa !". The text is not wrapped, with the second line starting immediately after the first line ends.

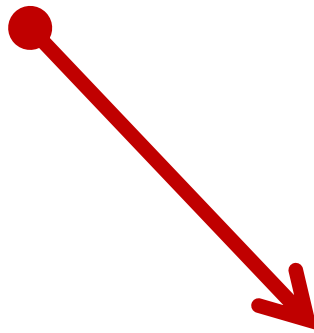
```
C:\ Console simulando o modo texto do MS-DOS

Olá mundo!Me livre da maldição
>>> Fim da execução do programa !
```

**E para quebrar linha?**

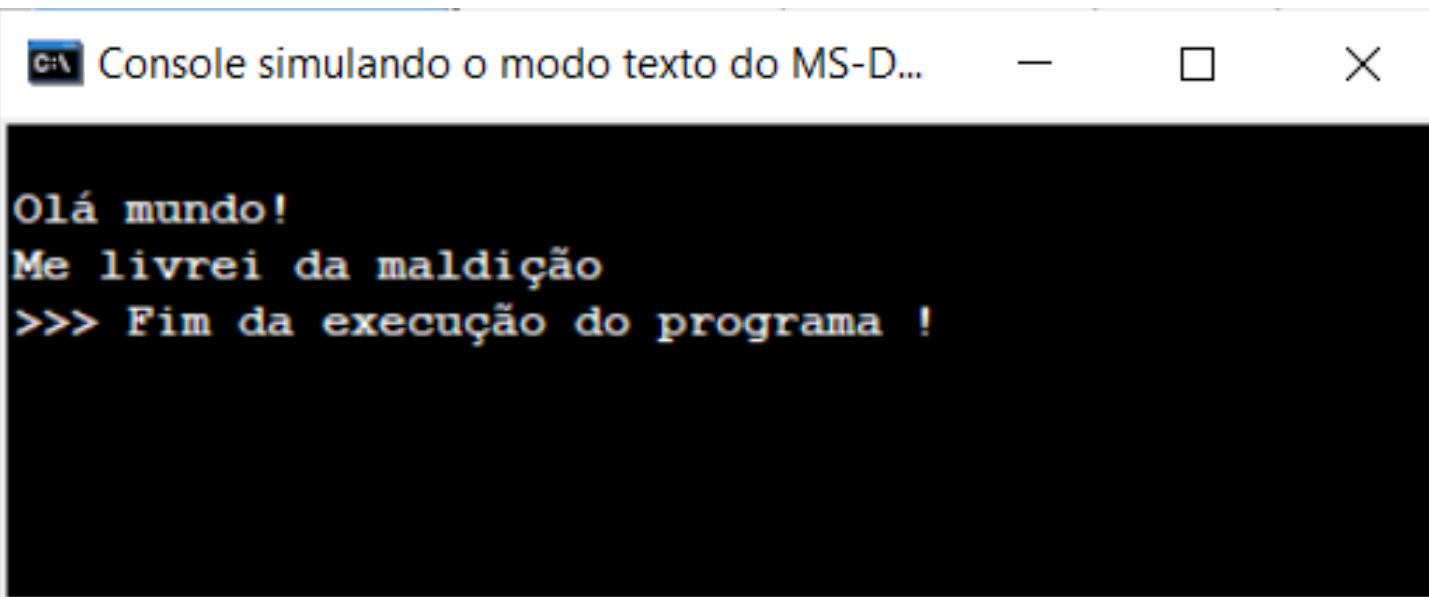


## Escreval ("Qualquer coisa")



Escreve e pula  
a linha

```
1 algoritmo "primeiro"  
2 var  
3  
4 inicio  
5     Escreval("Olá mundo!")  
6     Escreva("Me livrei da maldição")  
7 fimalgoritmo
```



C:\ Console simulando o modo texto do MS-D... — □ ×

```
Olá mundo!  
Me livrei da maldição  
>>> Fim da execução do programa !
```





São espaços para guardar dados (valores) que podem ser manipulados nos comandos de códigos.

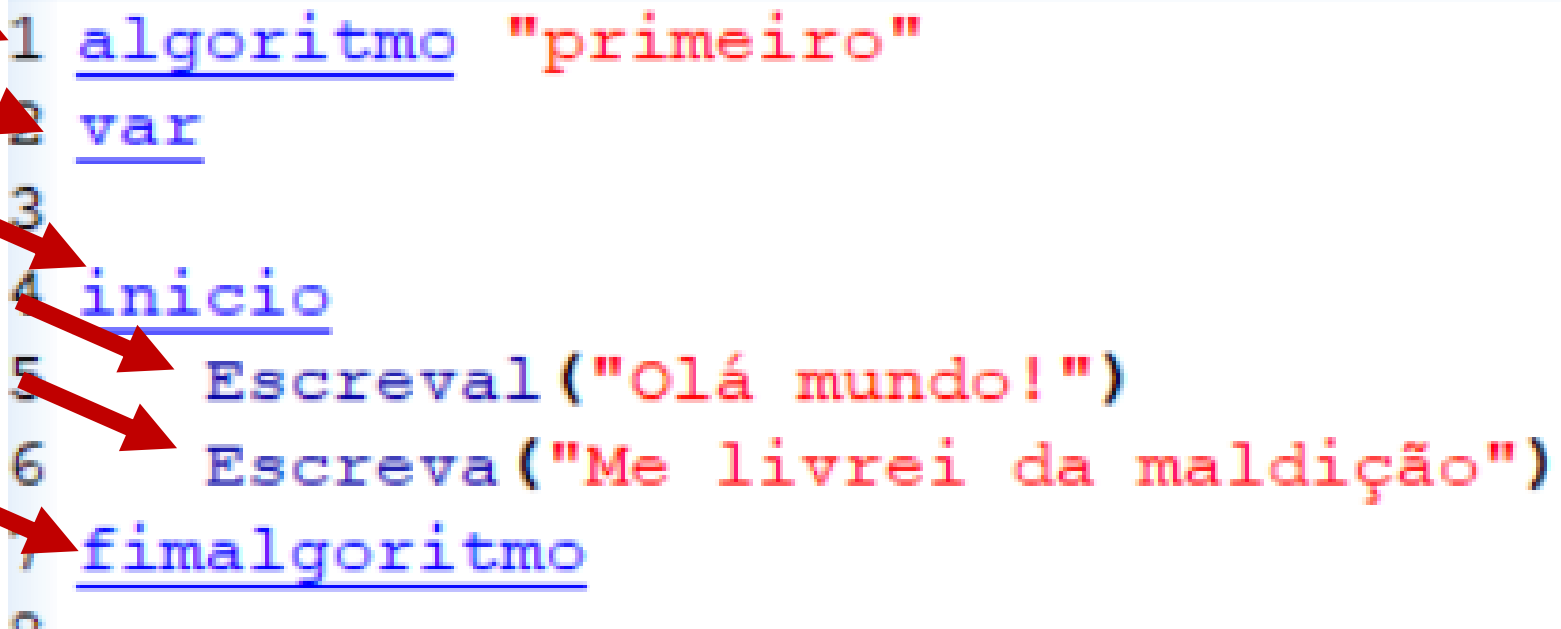
**var**

identificador: tipo

**Identificador:** identificam determinadas variáveis. Existem 6 regras para nomeação de identificadores no VisualG.

- 1) Deve começar com uma letra;**
- 2) Os próximos caracteres podem ser letras ou números;**
- 3) Não pode utilizar nenhum símbolo, exceto *\_*(underline);**
- 4) Não pode conter espaços em branco;**
- 5) Não pode ter letras com acento;**
- 6) Não pode ser uma palavra reservada**

**Palavras reservadas:** no VisualG aparecem sublinhadas ou em um tom de azul:



```
1 algoritmo "primeiro"  
2 var  
3  
4 inicio  
5     Escreval("Olá mundo!")  
6     Escreva("Me livrei da maldição")  
7 fimalgoritmo  
8
```

The diagram illustrates reserved keywords in VisualG code. Red arrows point to the following keywords: algoritmo, var, inicio, and fimalgoritmo. These keywords are underlined and colored blue in the original image. The other lines of code, including the string literals and the `Escreval` and `Escreva` statements, are not reserved keywords and are shown in red.

**Exemplos:**

Nota1



Média



Salário Bruto



9dade



**Existem 4 tipos:** inteiro, real, caractere e lógico.

**Inteiro:** guarda apenas números inteiros.

Exemplos: 1, 3, -5, 198, 0

**Real:** guarda números fracionários. Exemplos: 0.5, 5.0, 9.8, -77.3, 3.1415 (representado com pontos)



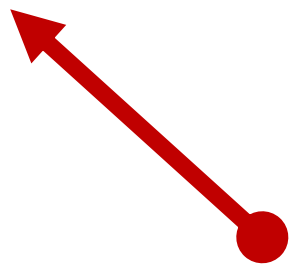
**Caractere:** tudo o que é colocado entre aspas (textos). Exemplos: “Válter”, “Algoritmo”, “123”. Dentro das aspas podemos colocar acento, palavras reservadas, etc... sem ferir as regras.

**Lógico:** guarda apenas dois valores: verdadeiro e falso.

**var**

**msg: caractere**

**msg <- "Olá, mundo!"**



Atribuição: a variável msg  
irá receber a mensagem  
"Olá, mundo"!

```
1 algoritmo "primeiro"  
2 var  
3     msg: caractere  
4 inicio  
5     msg <- "Olá, mundo!"  
6     Escreval(msg)  
7 fimalgoritmo
```

**Escreva ("msg")**

Escreve a palavra msg

**Escreva (msg)**

Escreve o conteúdo da  
variável msg

**Escreva ("mensagem", msg)**

Separando por vírgulas,  
podemos escrever tanto  
palavras quanto variáveis

```
1 algoritmo "primeiro"  
2 var  
3     msg: caractere  
4 inicio  
5     msg <- "Olá, mundo!"  
6     Escreval("Mensagem", msg)  
7 fimalgoritmo
```

```
1 algoritmo "primeiro"  
2 var  
3     msg: caractere  
4 inicio  
5     msg <- "Olá, mundo!"  
6     Escreval("Mensagem: ", msg)  
7 fimalgoritmo
```



# Comandos de Entrada

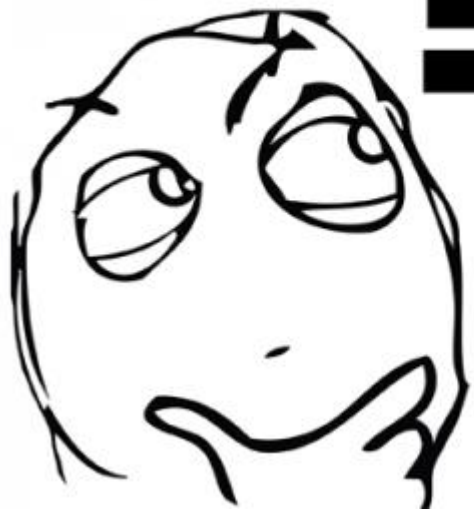


Comandos que solicitam alguma coisa ao usuário que está utilizando o sistema, promovendo interação.

```
1 Algoritmo "MeuNome"  
2 Var  
3  
4     Nome: caractere  
5  
6 Inicio  
7  
8     Nome <- "Válder"  
9     Escreva ("Muito prazer, ", Nome)  
10  
11 Fimalgoritmo
```

**Qual o problema deste programa?**

???




Ele funciona apenas para um nome.

Seria legal se o programa no início perguntasse o nome, para após escrever a mensagem.



```
1 Algoritmo "MeuNome"  
2 Var  
3  
4     Nome: caractere  
5  
6 Inicio  
7  
8     escreva ("Digite seu nome:")  
9     leia (Nome)  
10    escreva ("Muito prazer, ", Nome)  
11  
12 Fimalgoritmo
```

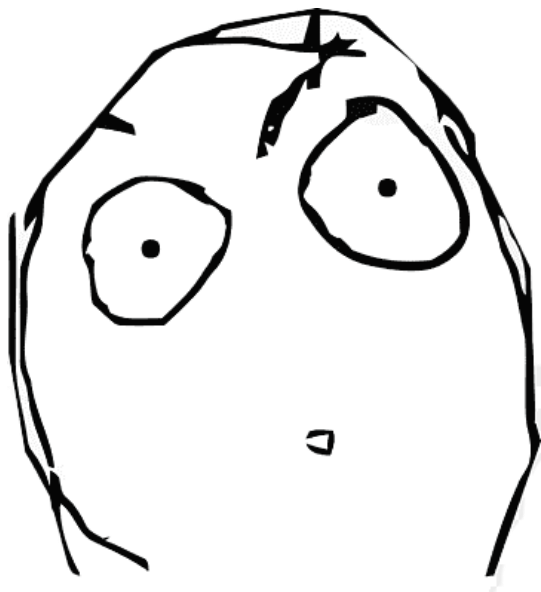
```
6 Inicio
7
8     escreva ("Digite seu nome:")
9     leia (Nome)
10    escreva ("Muito prazer, ", Nome)
11
```



Comando de entrada, funciona com qualquer tipo primitivo de valor.



**Solicitar dois números para o usuário e mostrar a soma entre eles.**



```
1 Algoritmo "Soma"  
2 Var  
3  
4     N1, N2, soma: inteiro  
5  
6 Inicio  
7  
8     escreva ("Digite um número: ")  
9     leia (N1)  
10    escreva ("Digite outro número: ")  
11    leia (N2)  
12    Soma <- N1 + N2  
13    escreva ("A soma dos dois números é ", soma)  
14  
15 Fimalgoritmo
```

```
1 Algoritmo "Soma"  
2 Var  
3  
4   N1, N2, soma: inteiro  
5  
6 Inicio  
7  
8   escreva ("Digite um número: ")  
9   leia (N1)  
10  escreva ("Digite outro número: ")  
11  leia (N2)  
12  Soma <- N1 + N2  
13  escreva ("A soma do número ", N1, " e do número ", N2, " é ", soma)  
14  
15 Fimalgoritmo
```

# Operadores Aritméticos



**Operadores aritméticos ou matemáticos** são um dos grupos de operadores suportados pela maioria das linguagens de programação, e realizam cálculos entre variáveis.

# Operadores Aritméticos

		$A \leftarrow 5$ $B \leftarrow 2$
+	Adição	$A + B$
-	Subtração	$A - B$
*	Multiplicação	$A * B$
/	Divisão	$A / B$
\	Divisão Inteira	$A \setminus B$
^	Exponenciação	$A ^ B$
%	Módulo	$A \% B$



# 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
$\backslash$	Divisão Inteira	$A \backslash B$	2
$\wedge$	Exponenciação	$A \wedge B$	25
$\%$	Módulo	$A \% B$	1

**Ordem de precedência** indica a ordem que os operadores serão considerados em uma mesma expressão.

<b>( )</b>	Parênteses
<b>^</b>	Exponenciação
<b>* /</b>	Multiplicação / Divisão
<b>+ -</b>	Adição / Subtração

$$3 + 2/2 =$$

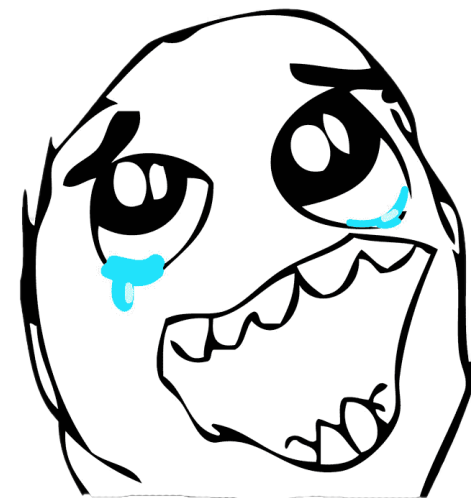
$$(3 + 2) / 2 =$$



$$3 + 2/2 = 4$$

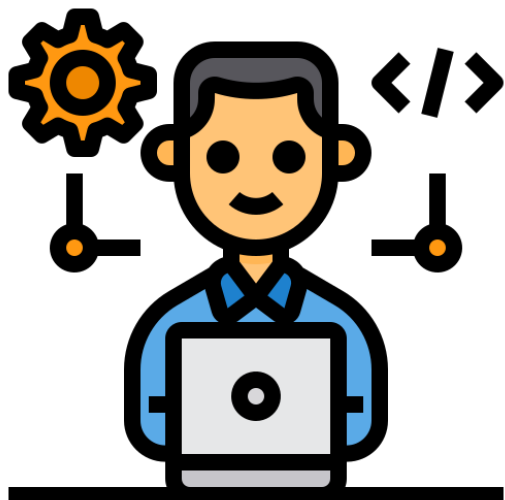
$$(3 + 2) / 2 = 2.5$$

**Solicitar dois números para o usuário e mostrar a média entre eles.**



```
1 Algoritmo "Soma"  
2 Var  
3  
4   N1, N2: inteiro  
5   media: real  
6  
7 Inicio  
8  
9   escreva ("Digite um número: ")  
10  leia (N1)  
11  escreva ("Digite outro número: ")  
12  leia (N2)  
13  media <- (N1 + N2) / 2  
14  escreva ("A média do número ", N1, " e do número ", N2, " é ", media)  
15  
16 Fimalgoritmo
```

Algumas outras funções aritmética que podemos usar no código para facilitar nossos cálculos em algum programa desenvolvido:



<b>Abs</b>	Valor Absoluto	<b>Abs(-10)</b>
<b>Exp</b>	Exponenciação	<b>Exp(3,2)</b>
<b>Int</b>	Valor Inteiro	<b>Int(3.9)</b>
<b>RaizQ</b>	Raiz Quadrada	<b>RaizQ(25)</b>
<b>Pi</b>	Retorna Pi	<b>Pi</b>
<b>Sen</b>	Seno (rad)	<b>Sen(0.523)</b>
<b>Cos</b>	Cosseno (rad)	<b>Cos(0.523)</b>
<b>Tan</b>	Tangente (rad)	<b>Tan(0.523)</b>
<b>GraupRad</b>	Graus para Rad	<b>GraupRad(30)</b>



# Outras Funções Aritméticas

<b>Abs</b>	Valor Absoluto	<b>Abs(-10)</b>	<b>10</b>
<b>Exp</b>	Exponenciação	<b>Exp(3,2)</b>	<b>9</b>
<b>Int</b>	Valor Inteiro	<b>Int(3.9)</b>	<b>3</b>
<b>RaizQ</b>	Raiz Quadrada	<b>RaizQ(25)</b>	<b>5</b>
<b>Pi</b>	Retorna Pi	<b>Pi</b>	<b>3.14..</b>
<b>Sen</b>	Seno (rad)	<b>Sen(0.523)</b>	<b>0.5</b>
<b>Cos</b>	Cosseno (rad)	<b>Cos(0.523)</b>	<b>0.86</b>
<b>Tan</b>	Tangente (rad)	<b>Tan(0.523)</b>	<b>0.57</b>
<b>GraupRad</b>	Graus para Rad	<b>GraupRad(30)</b>	<b>0.52</b>

**Crie um algoritmo chamado “conversor” que converta um ângulo dado em graus para seno.**

