

```
1
2
3  Algoritmo 'Lógica de' {
4
5      [ Programação 1 ]
6
7
8
9
10
11
12 }
13
14
```

Unidade 03

Prof. Daniel Caixeta



```
1  'Sumário' {
2
3      </01> Linguagem de programação
4          < 1.1. Do que precisamos? 1.2. Linguagem de programação. >
5
6      </02> Ambiente de Desenvolvimento
7          < 2.1. Conceitos [...]. 2.2. Da compilação à máquina virtual.
8          2.3. Portugol/VisualG. >
9
10     </03> Estrutura sequencial
11         < 3.1. Introdução. 3.2. Expressões aritméticas. 3.3. Operadores arit-
12         méticos. 3.4. Variáveis e tipos de dados. 3.5. As 3 operações básicas.
13         3.6. Funções matemáticas. >
14
```

```
1
2
3  </04> Expressões comparativa e lógicas [...]
```

```
4      < 4.1. Expressões de comparação. 4.2. Expressões lógicas.
```

```
5      4.3. Operadores lógicos. >
```

```
6  Referência
```

```
7
8  }
9
10
11
12
13
14
```

01 {

[Linguagem de Programação]

< Regras léxicas e sintáticas para se escrever
um programa >

}

1.1. Do que precisamos?

- 1
- 2
- 3 `</1>` De uma linguagem de programação com regras léxicas
- 4 e sintáticas.
- 5
- 6 `</2>` Uma IDE: *software* para editar e testar o programa.
- 7
- 8 `</3>` Um *software* que transforma código-fonte em código-
- 9 objeto (Compilador).
- 10
- 11 `</4>` Um *software* gerador de código ou máquina virtual
- 12 que permite que o programa seja executado.
- 13
- 14

1.2. Linguagem de programação

- É um conjunto de regras léxicas (ortografia) e sintáticas (gramática) utilizadas para escrever códigos de programação.

LÉXICA

- Trata-se da correção das palavras "isoladas" (ortografia).

Exemplo em português

cachorro

caxorro



Linguagem de programação

main

maim



SINTÁTICA

- Diz respeito à correção das sentenças (gramática).

Exemplo em português

O cachorro está com fome.

A cachorro está com fome.

Linguagem de programação

x = y - 5;

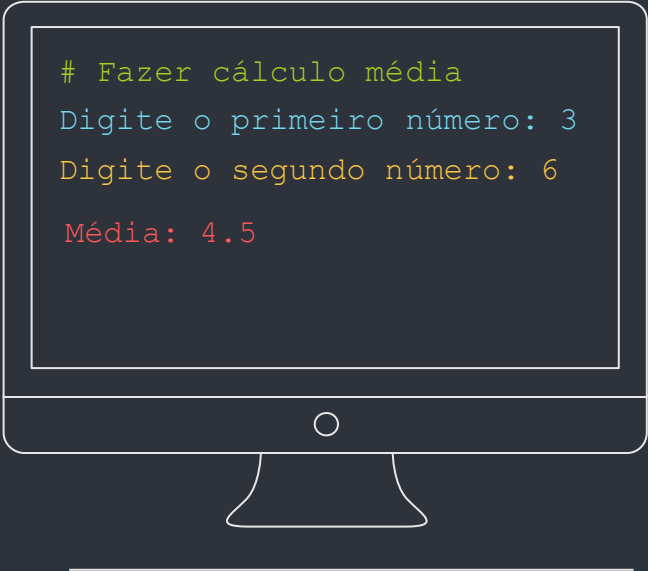
x = -y 5;

- Portanto, é importante ressaltar que essas regras são elementos fundamentais presentes em todas as linguagens (idiomas), inclusive nas linguagens de programação.

Exemplos: C, C++, C#, Java, Python, PHP, JavaScript, Rust, Ruby, etc.

Exemplo de um programa:

- Suponha um programa que pede ao usuário dois números e depois imprime na tela a média aritmética entre eles:



```
# Fazer cálculo média
Digite o primeiro número: 3
Digite o segundo número: 6
Média: 4.5
```


Solução em linguagem C

```
1  #include <stdio.h>
2
3
4  int main(){
5      double x, y, media;
6
7      printf("Digite o primeiro número: ");
8      scanf("%lf", &x);
9      printf("Digite o segundo número: ");
10     scanf("%lf", &y);
11     media = (x + y) / 2.0;
12     printf("Média = %.2f\n", media);
13     return 0;
14 }
```

Solução em linguagem C++

```
#include <iostream>

using namespace std;

int main(){
    double x, y, media;

    cout << "Digite o primeiro número: ";
    cin >> x;
    cout << "Digite o segundo número: ";
    cin << y;
    media = (x + y) / 2.0;
    cout << "Média = ", << media << endl;
    return 0;
}
```

Solução em linguagem C#

```
1 using System;
2
3 namespace programa {
4     class Program {
5         static void Main(string[] args) {
6             double x, y, media;
7
8             Console.Write("Digite o primeiro número: ");
9             x = double.Parse(Console.ReadLine());
10            Console.Write("Digite o segundo número: ");
11            y = double.Parse(Console.ReadLine());
12            media = (x + y) / 2.0;
13            Console.WriteLine("Média = " + media);
14        }
15    }
16 }
```

Solução em linguagem Java

```
import java.util.Scanner

public class Main {

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        double x, y, media;

        System.out.print("Digite o primeiro número: ");
        x = sc.nextDouble();
        System.out.print("Digite o segundo número: ");
        y = sc.nextDouble();
        media = (x + y) / 2.0;
        System.out.println("Média = " + media);
        sc.close();
    }
}
```

02 {

[Ambiente de Desenvolvimento]

< Ambientes, IDE e Frameworks >

}

2.1. Conceitos [...]

IDE – Ambiente Integrado de Desenvolvimento

- É um *software* que edita código-fonte, faz a automação de compilação local e possui um *debugger*. Todos unidos em uma única interface de usuário gráfica (GUI).
- Exemplos: Code Blocks, Eclipse, NetBeans, Visual Studio Code, etc.

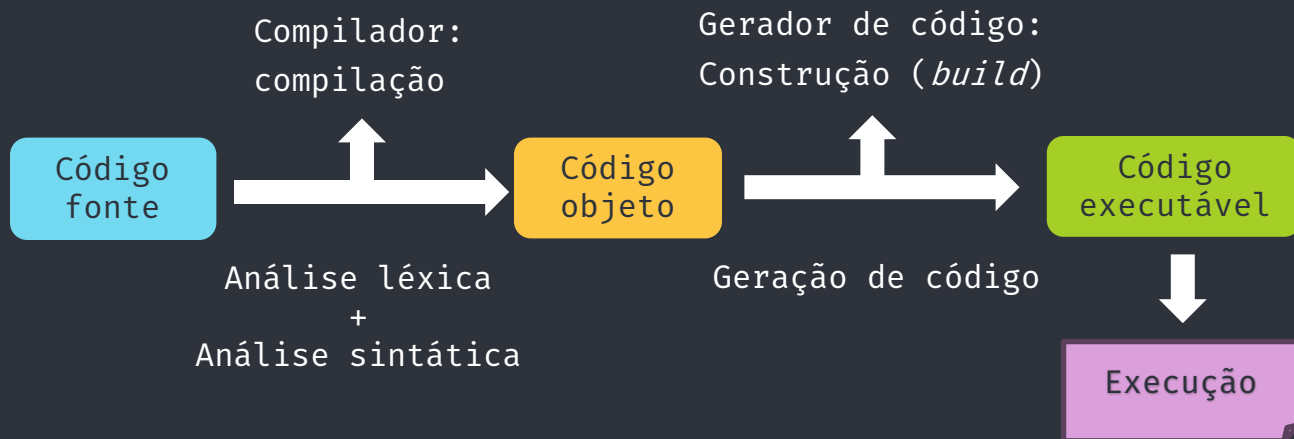
Framework

- É uma estrutura que oferece ferramentas e convenções para o desenvolvimento de *software*, permitindo aos programadores focarem na lógica específica do aplicativo, em vez de detalhes técnicos.
- Exemplos: Django e Flask (Python), React e Angular (JavaScript), Bootstrap (Design web responsivo), etc.

- 1
- 2 • As funcionalidade de uma IDE, assim como de um *Framework*
- 3 são:
- 4 </1> Edição de códigos-fontes (indentação, autocompletar, des-
- 5 taque de palavras, etc.).
- 6 </2> Depuração e testes (*Debugger*).
- 7
- 8 </3> Construção do produto final (*build*).
- 9 </4> Sugestões de modelos (*templates*).
- 10 </5> Auxiliar em diversas tarefas do projeto de *software*.
- 11
- 12 Etc.
- 13
- 14

2.2. Da compilação à máquina virtual

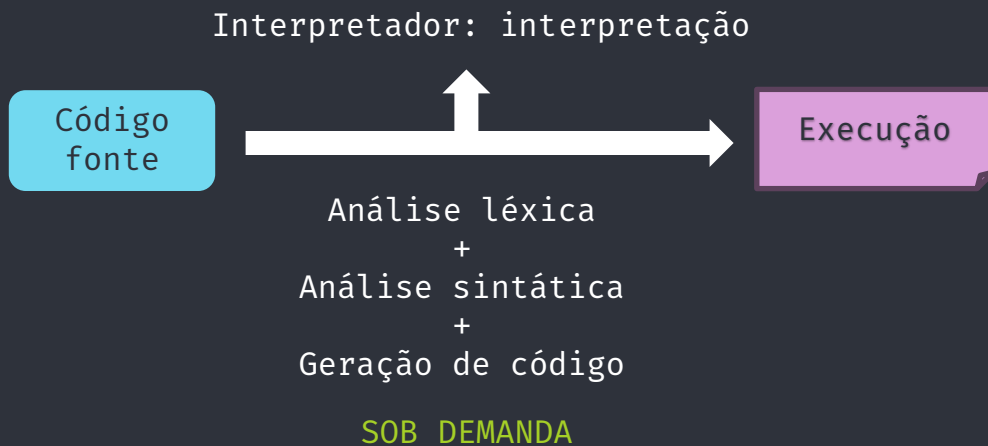
1 COMPILAÇÃO¹



Exemplos: C, C++, Fortran, etc.

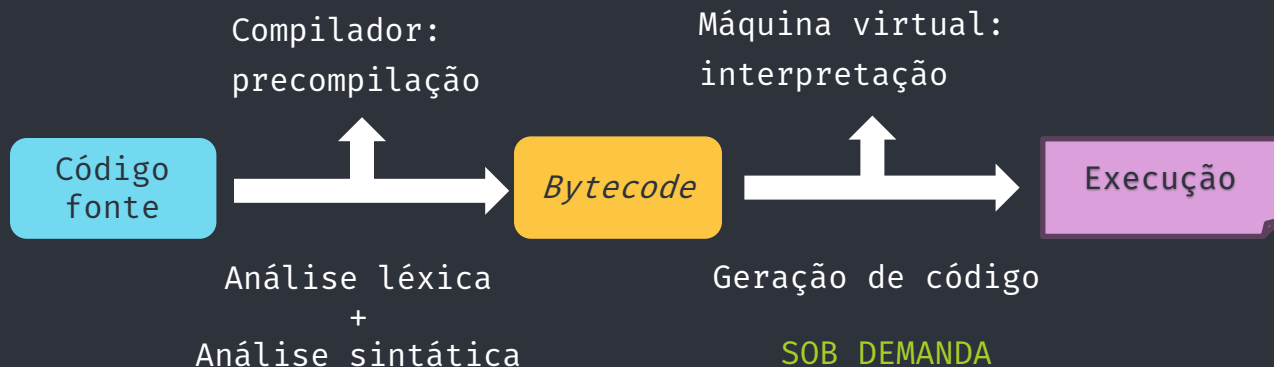
1. Compilação é o processo de tradução de um programa (código-fonte) para uma linguagem de máquina (código-objeto), para que suas instruções sejam executadas pelo processador (código executável).

INTERPRETAÇÃO²



Exemplos: PHP, JavaScript, Python, Ruby, etc.

2. Na linguagem interpretada, o código-fonte não é traduzido diretamente em código de máquina, apenas lê e executa o código.

ABORDAGEM HÍBRIDA³

Exemplos: Java (JVM), C# (Microsoft .NET Framework)

3. São as linguagens que implementam as duas formas de leituras, compilado e interpretado, *E.g.*, o Java tem um compilador que converte seu código para *bytecode* e na sequência isso é lido para a JVM (Java Virtual Machine) que é basicamente um interpretador, para no final ser executado pelo CPU.

VANTAGENS

COMPILAÇÃO

- Velocidade do programa.
- Auxílio do compilador antes da execução.

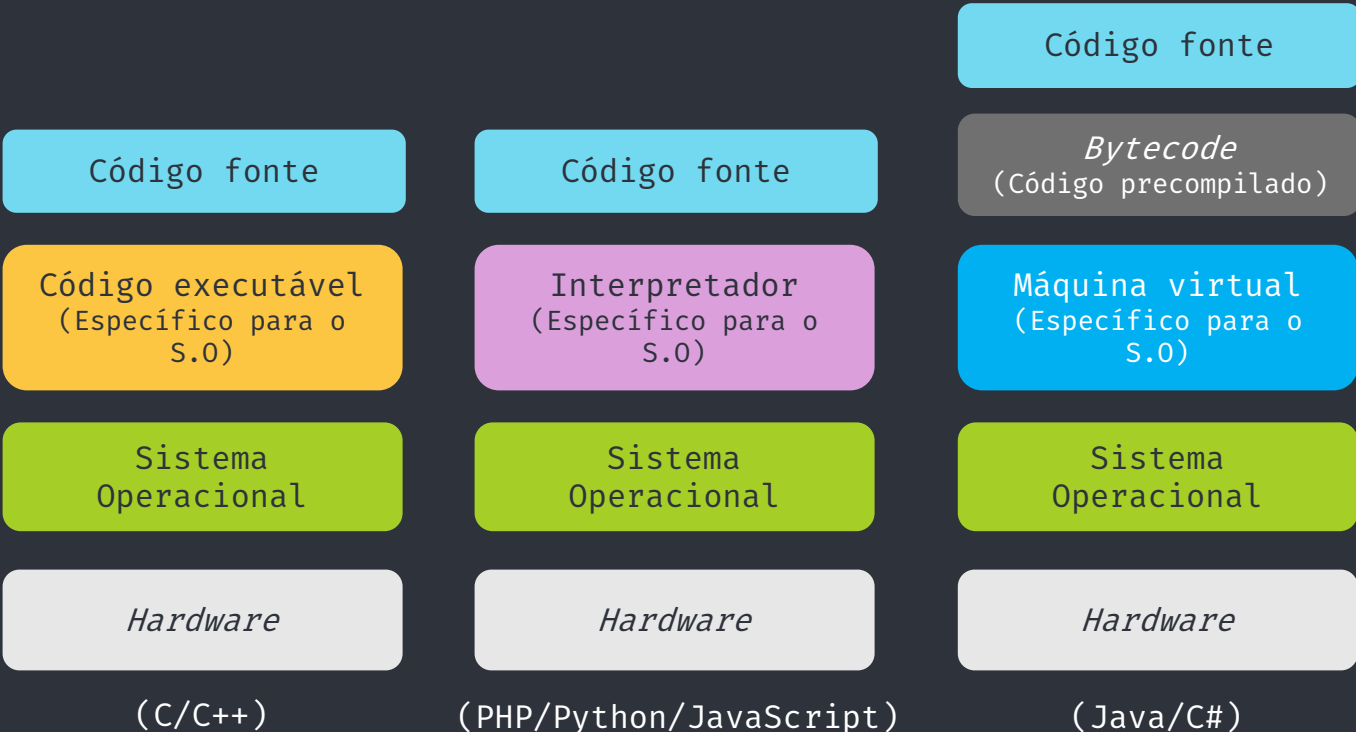
INTERPRETAÇÃO

- Flexibilidade de manutenção do aplicativo em produção.
- Expressividade da linguagem.
- Código fonte não precisa ser recompilado para rodar em plataformas diferentes.

ABORDAGEM HÍBRIDA

```
graph LR; A[ABORDAGEM HÍBRIDA] --> B[COMPILAÇÃO]; A --> C[INTERPRETAÇÃO];
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14



2.3. Portugol/VisualG

- 1 • O Portugol é uma linguagem de programação didática, simpli-
2 ficada, cujo objetivo é ensinar lógica de programação para
3 estudantes de língua portuguesa.
- 4 • A sua sintaxe varia um pouco, dando origem ao que denomina-
5 mos de dialetos.
- 6 • Se refere também como “Português estruturado”, “Linguagem
7 de algoritmos”, etc.
- 8 • Já o VisualG é uma IDE para editar e interpretar programas
9 em Portugol.
- 10 • Então:
 - 11 1. Portugol: linguagem (regras léxicas e sintáticas).
 - 12 2. VisualG: Ferramenta para escrever e interpretar programas.

Solução em Portugol

```
1  
2  
3 x, y, media : real  
4 escreva("Digite o primeiro número: ")  
5 leia(x)  
6 escreva("Digite o segundo número: ")  
7 leia(y)  
8 media <- (x + y) / 2.0  
9  
10  
11  
12  
13  
14
```

Aprender a lógica.



Solução em linguagem C

```
#include <stdio.h>  
  
int main(){  
    double x, y, media;  
  
    printf("Digite o primeiro número: ");  
    scanf("%lf", &x);  
    printf("Digite o segundo número: ");  
    scanf("%lf", &y);  
    media = (x + y) / 2.0;  
    printf("Média = %.2f\n", media);  
    return 0;  
}
```

Aprender a implementar a lógica em linguagem C.

Estrutura do código em Portugol

```
1  Nome do algoritmo → Algoritmo "media"
2
3  Área de declaração de variável {
4      VAR
5      x, y, media : real
6
7  Corpo do algoritmo {
8      Inicio
9      escreva("Digite o primeiro número: ")
10     leia(x)
11     escreva("Digite o segundo número: ")
12     leia(y)
13     media <- (x + y) / 2.0
14     escreva("Média = ", media:8:2)
15     Fimalgoritmo
```

03 {

[Estrutura sequencial]

< Expressões aritméticas, variáveis,
dados e funções >

}

3.1. Introdução

- Um algoritmo deve seguir uma sequência lógica e precisa para desempenhar sua função corretamente, com cada comando sendo executado em ordem, de cima para baixo, da esquerda para a direita.
- Por exemplo:

```
x <- 10
y <- 20
media <- (x + y) / 2
```

(Sequência correta)

```
media <- (x + y) / 2
x <- 10
y <- 20
```

(Sequência errada)

3.2. Expressões aritméticas

- É uma combinação de números, operadores aritméticos (adição, subtração, multiplicação, divisão), parênteses e possivelmente outras funções matemáticas, que representa uma operação ou cálculo matemático.

- Alguns exemplos:

$$2 + 3 = 5$$

$$(5 * 4) - 7 = 13$$

$$12 / (3 + 1) = 3$$

- No cálculo dessas expressões obtemos um resultado numérico específico.

3.3. Operadores aritméticos

	Operador	Significado	Precedência
1			
2	()	Parêntesis	1º
3	Raiz(Q)	Radiciação	1º
4	^ ou **	potenciação (real)	1º
5	% ou mod	Resto da divisão (inteiro)	2º
6	/	Divisão	2º
7	\	Divisão inteira	2º
8	*	Multiplicação	2º
9	+	adição	3º
10	-	subtração	3º
11			
12			
13	(Operadores aritméticos do VisualG)		
14			

Exemplos de expressões aritméticas </1> {

$$2 * 6 / 3 =$$

$$3 + 2 * 4 =$$

$$(3 + 2) * 4 =$$

$$2 * 3 ^ 4 =$$

$$60 / (3 + 2) * 4 =$$

$$60 / ((3 + 2) * 4) =$$

}

Exemplos com o operador “mod” ou % </2> {

$$12 \% 3 = 2$$

$$19 \% 5 = 4$$

$$\begin{array}{r} 14 \quad | \quad 3 \\ \hline 2 \quad 4 \end{array}$$

$$\begin{array}{r} 19 \quad | \quad 5 \\ \hline 4 \quad 3 \end{array}$$

}

3.4. Variáveis e tipos de dados

- 1
2 • Em programação, uma variável é uma porção de memória (RAM)
3 utilizada para armazenar dados durante a execução dos pro-
4 gramas.
- 5 • A declaração de variáveis geralmente depende da linguagem
6 de programação. No entanto, em muitas linguagens populares,
7 a declaração envolve especificar o tipo de dado que a vari-
8 ável irá armazenar, seguido pelo nome da variável.
- 9
10 • Aqui estão alguns exemplos de como declarar variáveis em
11 diferentes linguagens de programação:
12
13
14

Python

```
1 # Variável inteira  
2 numero = 10  
3  
4 # Variável de ponto flutuante  
5 valor = 3.14  
6  
7 # Variável de string  
8 nome = "Maria"
```

C

```
9 // Variável inteira  
10 int numero = 10;  
11  
12 // Variável de ponto flutuante  
13 float valor = 3.14;  
14  
15 // Variável de caractere  
16 char letra = "Maria";
```

Java

```
// Variável inteira  
int numero = 10;  
  
// Variável de ponto flutuante  
double valor = 3.14;  
  
// Variável de string  
String nome = "Maria";
```

JavaScript

```
// Variável inteira  
let numero = 10;  
  
// Variável de ponto flutuante  
let valor = 3.14;  
  
// Variável de string  
let nome = "Maria";
```

DECLARAÇÃO DE VARIÁVEIS (VISUALG)

Sintaxe:

<nome> : <tipo>

// Exemplos:

idade = inteiro

altura = real

logradouro : caractere

// Uma variável possui:

- ✓ Nome (ou identificador)
- ✓ Tipo de dados
- ✓ Valor
- ✓ Endereço na memória

Memória RAM

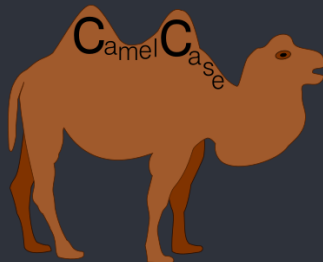


OS TIPOS BÁSICOS DE DADOS (VISUALG)

Tipo	Descrição	Valor padrão	Valores possíveis
Inteiro	Número inteiro	0	-2.147.483.648 a -2.147.483.647
Real	Número com ponto flutuante	0	-1,4024E-37 a 3,4028E+38
Caractere	Texto	""	Textos
Lógico	Valor verdade	FALSO	Falso e/ou Verdadeiro

NOMES DE VARIÁVEIS

- Não pode começar com dígito: use uma letra ou _ .
- Não pode ter espaço em branco.
- Não usar acentos ou til.
- Sugestão: use o padrão “Camel Case”.



Errado

```
5minutos : inteiro
salário : real
salário do funcionário: real
```

Certo

```
_5minutos : inteiro
salario : real
salarioDoFuncionário: real
```


3.5. As 3 operações básicas

ENTRADA DE DADOS

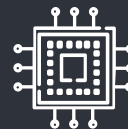
- É o processo pelo qual as informações são fornecidas a um *software* ou sistema por meio de dispositivos de entrada, como teclado, mouse, etc.



PROCESSAMENTO DE DADOS

- É quando o programa realiza cálculos, tarefas, rotinas, etc., por meio de sua execução. Ocorre por meio de um comando de ATRIBUIÇÃO (<-).

```
media <- (x + y) / 2.0
```



SAÍDA DE DADOS

- É o processo pelo qual um programa entrega informações processadas ao usuário, seja por meio de dispositivos de saída, como monitores, impressoras, etc.



UM EXEMPLO

Algoritmo "teste_saida"

Var

a : inteiro

b : real

c : caractere

d : lógico

Inicio

a ← 32

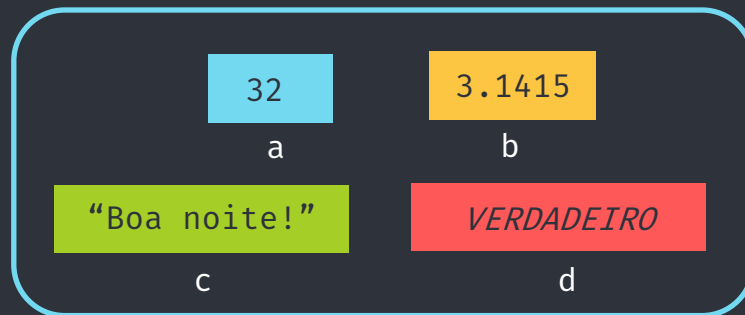
b ← 3.1415

c ← "Boa noite!"

d ← VERDADEIRO

Fimalgoritmo

Memória RAM



PROCESSAMENTO DE DADOS <- COMANDO DE ATRIBUIÇÃO

• Sintaxe:

<variável> <- <expressão>

Lê-se "recebe"

</1> Exemplo

x, y : inteiro

x <- 5

y <- 2 * x

escreval(x)

escreval(y)

</2> Exemplo

x : inteiro

y : real

x <- 5

y <- 2 / x

escreval(x)

escreval(y)

Regras:

1. A expressão é calculada.
2. O resultado da expressão é armazenado na variável.

3.6. Funções matemáticas

- 1
- 2 • Funções matemáticas em programação se referem a conjuntos
- 3 de operações predefinidas que realizam cálculos matemáticos
- 4 específicos.
- 5 • São ferramentas essenciais para manipular números e reali-
- 6 zar operações matemáticas em programas de computador.
- 7
- 8 • Essas funções são disponibilizadas por meio de bibliotecas
- 9 e podem incluir diversas operações, como trigonometria, ex-
- 10 ponenciação, raiz quadrada, arredondamento, entre outras.
- 11 • São usadas em uma variedade de finalidades, incluindo:

12
13
14

- 1
- 2 1. Cálculos simples: Adição, subtração, multiplicação e divisão
- 3 de números.
- 4 2. Operações avançadas: Cálculos trigonométricos, exponenciação,
- 5 logaritmos, raiz quadrada, etc.
- 6
- 7 3. Arredondamento e formatação: Arredondar números para um valor
- 8 em casas decimais, formatar números em formatos específicos
- 9 (*e.g.*, notação científica).
- 10 4. Geração de números randômicos: Gerar números aleatórios dentro
- 11 de um intervalo específico.
- 12 Etc. ...
- 13
- 14

1 **ALGUMAS FUNÇÕES [...]**3 **Exemplo** **Significado**4 `A <- RaizQ(x)` Variável A recebe a raiz quadrada de x.5 `A <- Exp(x, y)` Variável A recebe o resultado de x elevado a y.6 `A <- Pi` Variável A recebe o valor de Pi (π).7 `A <- Abs(x)` Variável A recebe o valor absoluto de x.

- 10
- 11 • Essas funções são projetadas para tornar a implementação de
12 cálculos matemáticos mais eficiente e precisa, eliminando a
13 necessidade de escrever código complexo para realizar operações
14 comuns.

04 {

[Expressões comparativas &
Lógicas]

< Operadores de comparação e Lógicos >

}

4.1. Expressões de comparação

- Os operadores de comparação são símbolos usados em programação para comparar valores e expressões.
- Retornam um valor booleano (V ou F).
- Esses operadores são fundamentais para controle de fluxo e tomada de decisões em algoritmos/programas.

Operador	Significado	Operador	Significado
>	Maior	<=	Menor ou igual
<	Menor	=	Igual
>=	Maior ou igual	<>	Diferente

Operadores de comparação em VisualG.

- Por exemplo:

Expressão	V	F
$5 \leq 8$		
$10 > 12$		
$8^2 = 64$		
$(2 * 4) <> (9 - 1)$		

- Os operadores de comparação são amplamente utilizados em estruturas de controle condicional, como instruções *if*, *else if* e *while*, para tomar decisões com base nas relações entre os valores das variáveis.

4.2. Expressões lógicas

- São combinações de símbolos e operadores lógicos que representam relações entre proposições ou valores lógicos.
- Essas expressões são usadas na lógica formal e na programação para representar condições, decisões e raciocínios.
- Os operadores lógicos comuns incluem **AND (E)**, **OR (OU)** e **NOT (NÃO)**, que são usados para combinar proposições ou inverter seu valor lógico.
- Por exemplo:
"Se estiver chovendo **OU** se estiver ensolarado, vou levar um guarda-chuva!"

4.3. Operadores lógicos

Operador	Significado
E (AND)	Verdadeiro se todas as condições forem verdadeiras.
OU (OR)	Verdadeiro se pelo menos uma condição for verdadeira.
NÃO (NOT)	Verdadeiro se a condição for falsa.

Operador NOT

A	S ou A'
0	1
1	0

Operador AND

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Operador OR

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

(A . B)

(A + B)

A IDEIA POR TRÁS DO OPERADOR “E” (AND)

- **TODAS** as condições devem ser **VERDADEIRAS!**
- Por exemplo:

Você pode obter uma habilitação de motorista se:

1. For aprovado no exame psicotécnico, **E**
2. For aprovado no exame de legislação, **E**
3. For aprovado no exame de legislação.

A	B	A E B
0	0	0
0	1	0
1	0	0
1	1	1

A IDEIA POR TRÁS DO OPERADOR “OU” (OR)


- Pelo menos **UMA** condições deve ser **VERDADEIRA!**
- Por exemplo:

Você pode estacionar na vaga especial se:

1. **For** idoso(a), **OU**
2. **For** uma pessoa com deficiência, **OU**
3. **For** uma gestante.

A	B	A ou B
0	0	0
0	1	1
1	0	1
1	1	1

1 EXEMPLOS DE EXPRESSÕES LÓGICAS

2 • Assumindo $X = 3$, então:3
4 1. $(X \leq 5)$ OU $(X = 9)$ Resultado:5
6 2. $(X > 1)$ OU $(X < 6)$ Resultado:7
8 3. $(X \leq 10)$ OU $(X = 6)$ OU $(X \neq 3)$ Resultado:9
10
11 4. $(X \leq 3)$ OU $(X = 3)$ E $(X \neq 3)$ Resultado:12
13 5. $(X > 2)$ E $(X = 3)$ OU $(X \leq 4)$ Resultado:14
 Expressões híbridas

A IDEIA POR TRÁS DO OPERADOR “NÃO” (NOT)

- O operador “NÃO” inverte a condição.

- Por exemplo:

Você tem direito a receber uma bolsa de estudos se você:

- NÃO

Possuir renda maior que R\$ 2.500,00



Golias
R\$: 2.850,00



V



Davi
R\$: 2.300,00



N

A	S ou A'
0	1
1	0

1

2

3 </1> ASCENCIO, Ana Fernanda G. ARAÚJO, Grazielle. S. Estruturas de Dados:
4 algoritmos, análise da complexidade e implementações em Java e C/C++. São Paulo:
Pearson Prentice Hall, 2010. (Biblioteca virtual).

5

6 </2> PUGA, Sandra. RISSETTE, Gerson. Lógica de Programação e Estrutura de Dados.
2. ed. São Paulo? Pearson Prentice Hall, 2009.

7

8

9

10

11

12

13

14

A 'Obrigado' Is {

Algoritmo & Linguagem de Programação 1

|
}

