

Técnicas de Desenvolvimento de Algoritmos



- ✓ Algoritmos?
- ✓ Interpretador/Compilador
- ✓ Tipos de Dados
- ✓ Variáveis
- ✓ Entrada e saída
- ✓ Funções matemáticas

Algoritmo?

A yellow rectangular box with a black border. In the top left corner are two orange code symbols: a less-than sign and a greater-than sign. In the bottom right corner are two orange code symbols: a less-than sign followed by a slash and a greater-than sign. The text inside the box is in bold black letters.

**FIRST, SOLVE
THE PROBLEM.**

**THEN, WRITE
THE CODE.**

— John Johnson

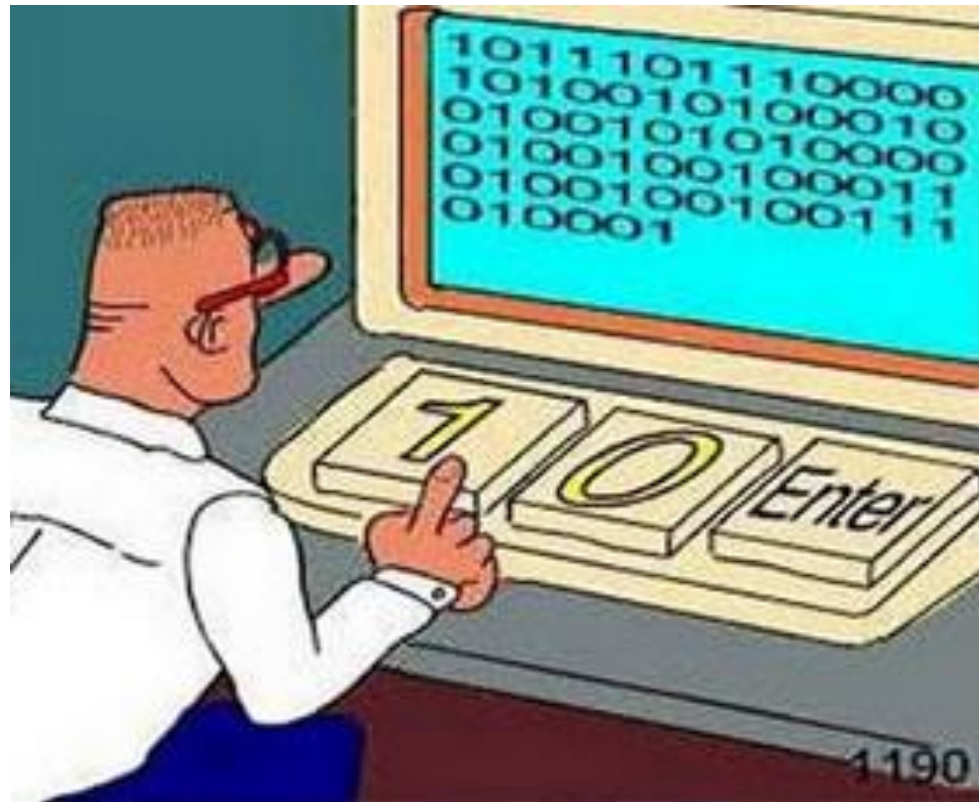
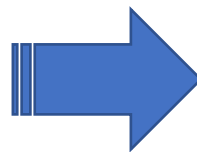
O que é um Programa de Computador?

São instruções escritas em uma linguagem que permite a comunicação entre o programador e o computador (0's e 1's) → Linguagem de programação

```
000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000001000001000
0000000101011000111100011000000011
11111000110101001111001111001111
00011010000110001111010101111000
11100000111111000011111101010111
11100000111111000011111101010111
```

Código de máquina

Um arquivo contendo instruções em linguagem de máquina é chamado de **executável**.



Linguagens de programação

- ✓ Uma linguagem natural ou idioma (português, inglês, espanhol etc.) estabelece um vocabulário ou dicionário válido e um grupo de regras sintáticas e semânticas para que as pessoas se expressem adequadamente nessa linguagem e se comuniquem.
- ✓ De forma semelhante, uma linguagem de programação estabelece um conjunto de regras sintáticas e semânticas, para escrever códigos válidos, a fim de que os computadores possam executar uma determinada função.



Como a máquina entende os códigos?

Para que o computador “entenda” um programa é necessário um meio de tradução entre a linguagem de alto nível utilizada no programa e a linguagem de máquina.

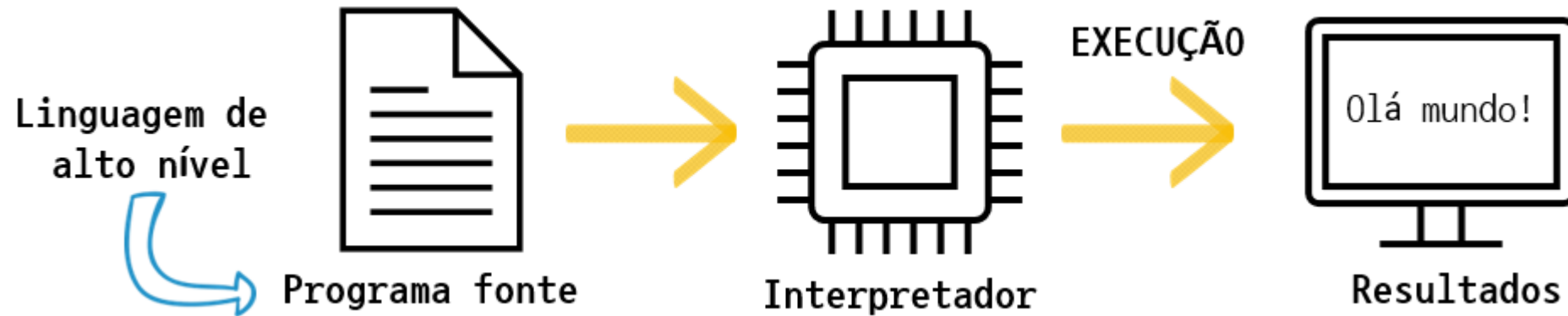
Para essa tarefa temos basicamente dois métodos:

- ✓ Compilador
- ✓ Interpretador



Interpretador

- ✓ Traduz e faz a checagem da sintaxe e envia para execução, instrução por instrução.
- ✓ Precisa estar presente todas as vezes que vamos executar o programa e o processo acima é repetido.

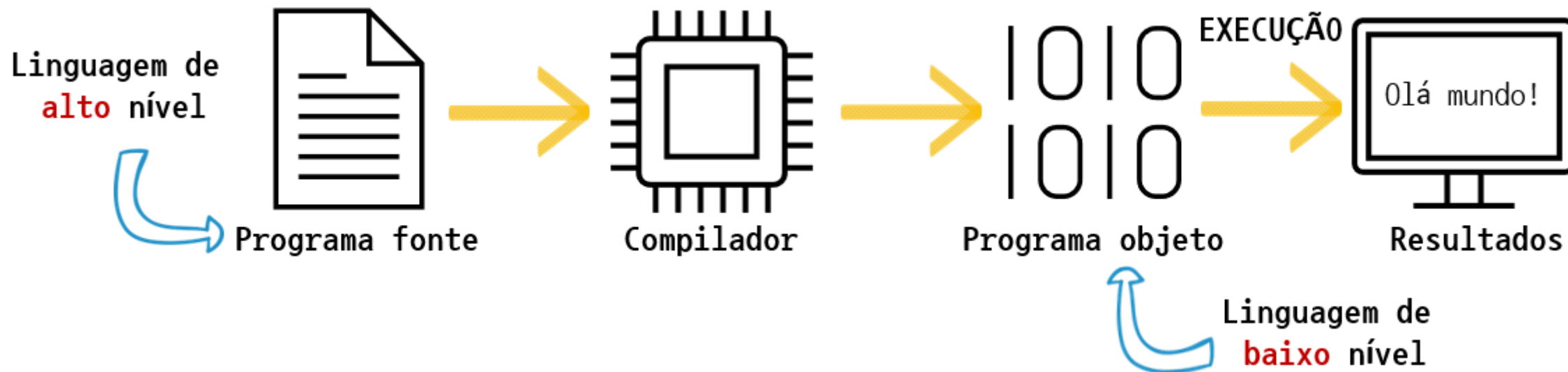


Vantagem: consome menos memória

Desvantagem: execução mais lenta

Compilador

Traduz o programa escrito em uma linguagem de programação para um programa equivalente escrito em linguagem de máquina (programa-objeto).



Vantagens:

- ✓ Velocidade de execução
- ✓ Oculta o código fonte

Desvantagem:

- ✓ A cada alteração no programa fonte é necessário gerar novamente o programa-objeto

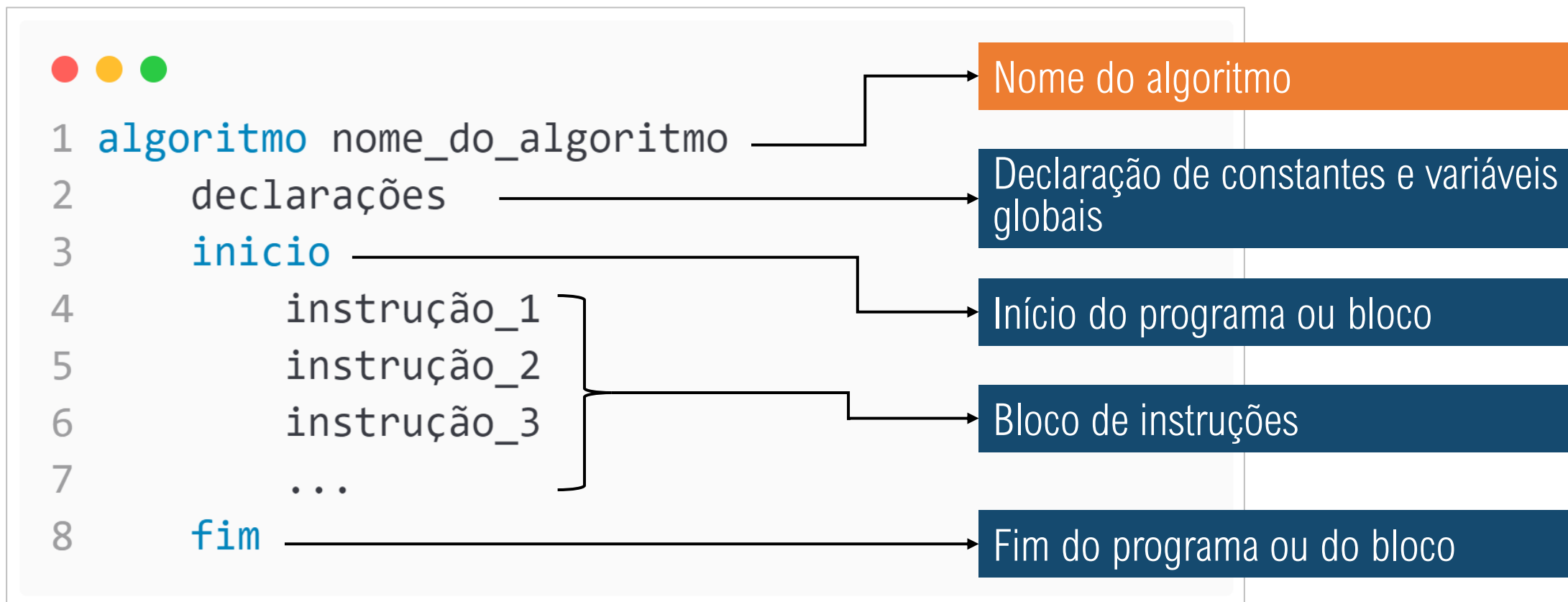
Pseudocódigo

- ✓ O pseudocódigo é a descrição do que o algoritmo faz passo a passo em um português estruturado.
- ✓ É o mais fácil de fazer e pode ser feito mentalmente e em qualquer lugar em que o estudante estiver.
- ✓ Trata-se de uma forma excelente para iniciar o aprendizado de lógica e é extremamente acessível.

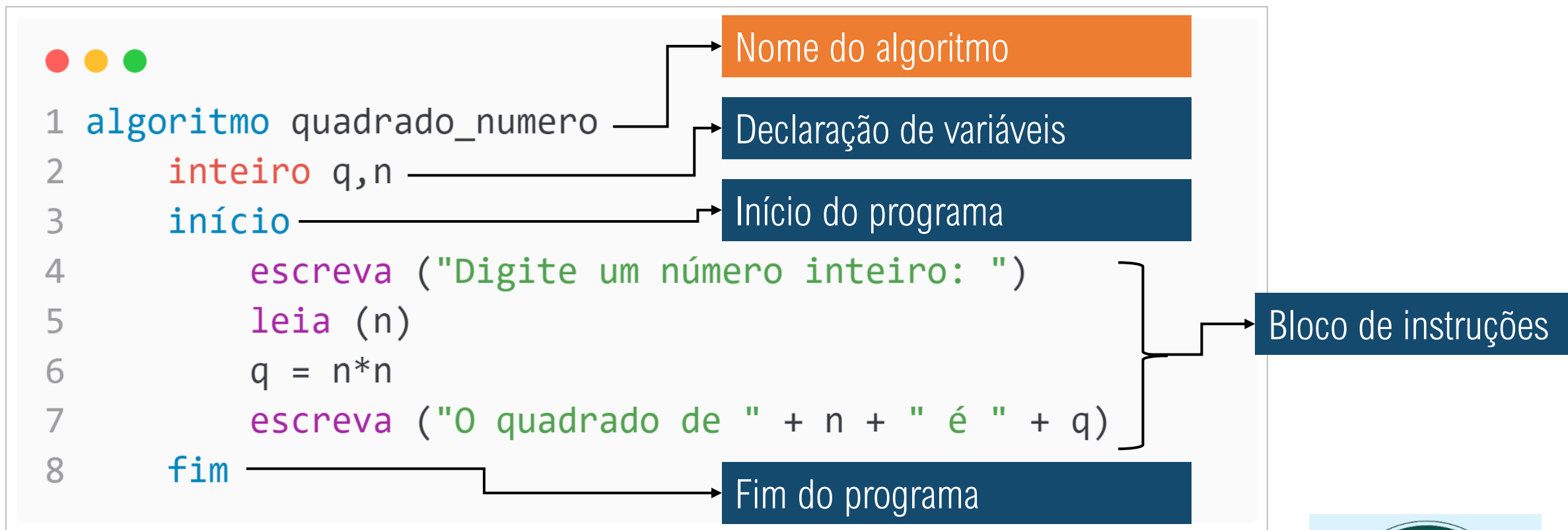


Notação utilizada na aula... PSEUDOCÓDIGO

Nesta disciplina, adotaremos a seguinte estrutura para o pseudocódigo:



Exemplos de algoritmo (pseudocódigo)



Algoritmo que calcula e exibe o quadrado de um número digitado pelo usuário.

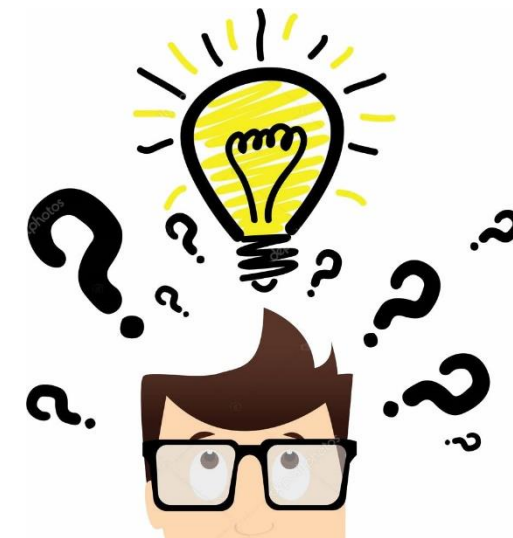


Exemplos de algoritmo (pseudocódigo)



```
1  algoritmo média
2      real: media, n1, n2
3      início
4          escreva("Digite o primeiro número: ")
5          leia(n1)
6          escreva("Digite o segundo número: ")
7          leia(n2)
8          media = (n1 + n2) / 2
9          escreva("A média dos números é: " + media)
10     fim
11
```

Algoritmo que calcula e exibe a média entre dois números digitados pelo usuário.

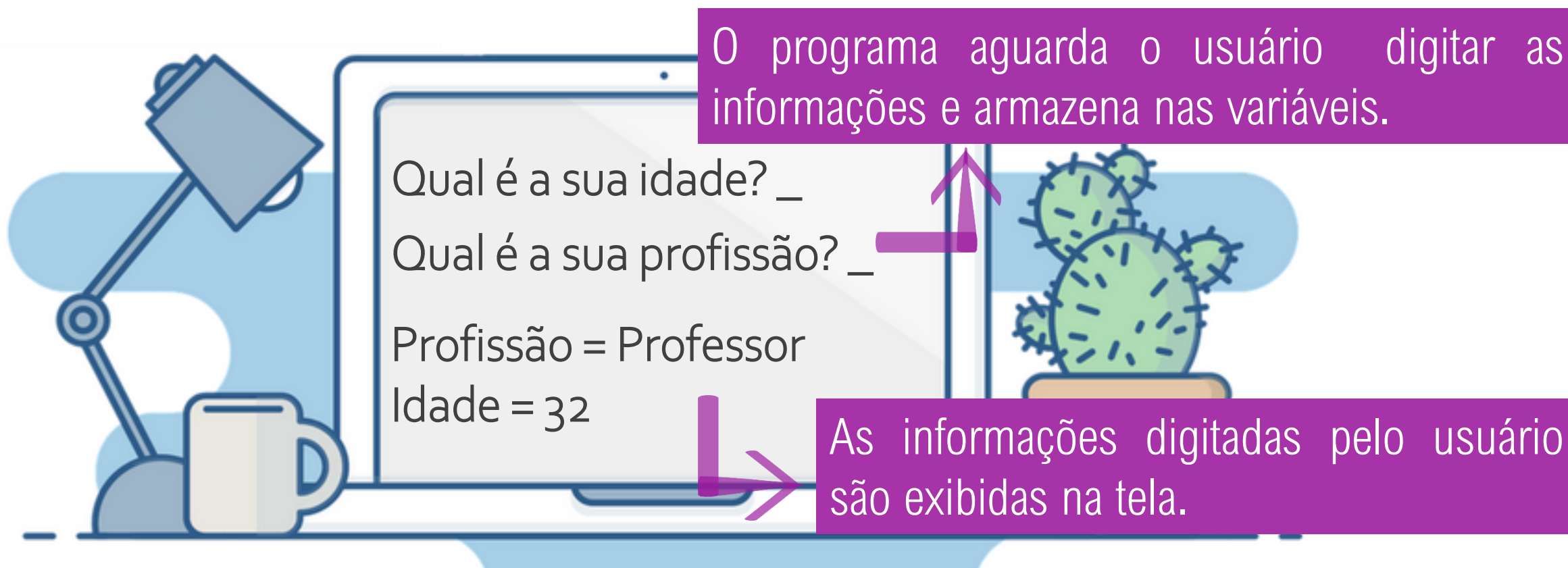


Exemplos de aplicação

1- Crie um algoritmo que solicite ao usuário a sua idade e armazene essa informação em uma variável, solicite também a profissão e armazene em outra variável. Após obter os dados, apresente em uma única mensagem os valores digitados (variáveis).

Exemplos de aplicação

1- Crie um algoritmo que solicite ao usuário a sua idade e armazene essa informação em uma variável, solicite também a profissão e armazene em outra variável. Após obter os dados, apresente em uma única mensagem os valores digitados (variáveis).



Exemplos de aplicação

// Exemplo 1: idade e profissão do usuário em Pseudocódigo

```
algoritmo Exemplo1
    inteiro idade
    literal profissao
    início
        escreva("Digite a sua idade: ")
        leia(idade)
        escreva("Digite a sua profissão: ")
        leia(profissao)
        escreva("Profissão: " + profissao)
        escreva("Idade: " + idade)
    fim
```



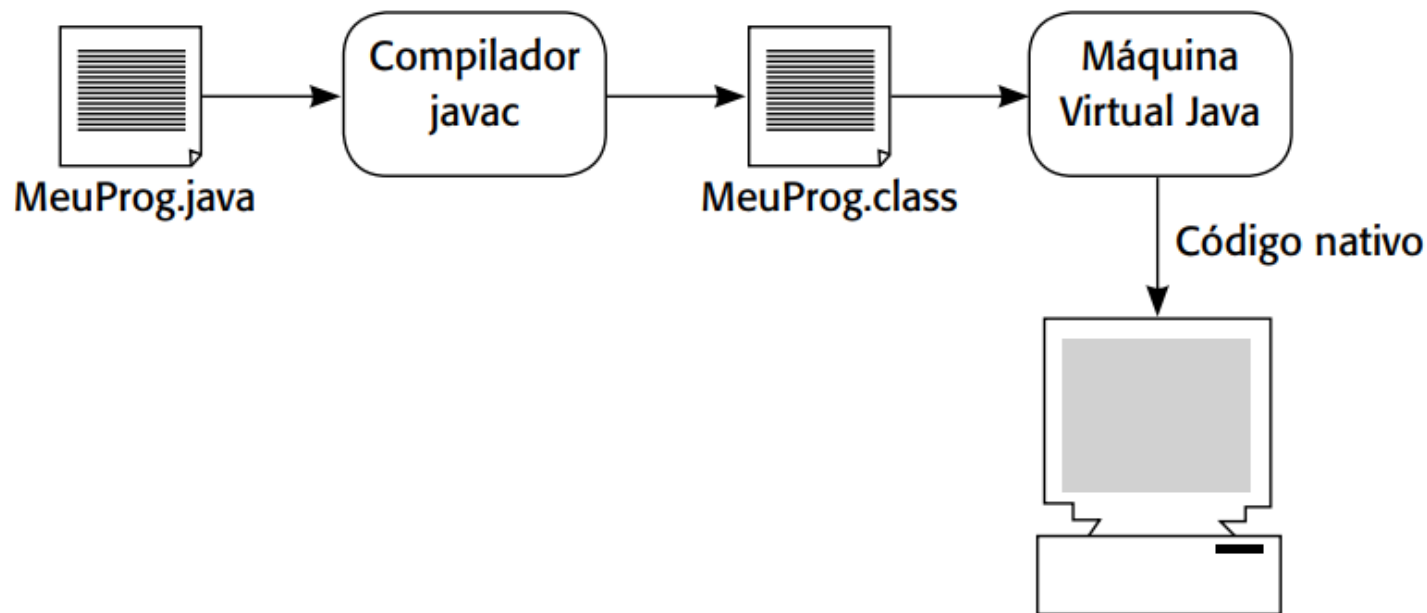
Linguagem Java

- ✓ Nesta disciplina utilizaremos a linguagem Java para implementação de alguns algoritmos.
- ✓ É uma linguagem de programação orientada a objetos desenvolvida na década de 90 por uma equipe de programadores chefiada por James Gosling, na empresa Sun Microsystems.
- ✓ Diferente das linguagens de programação modernas, que são compiladas para código nativo, a linguagem Java é compilada para um bytecode que é interpretado por uma máquina virtual (Java Virtual Machine, mais conhecida pela sua abreviação JVM).



Execução de programas em Java

- ✓ Os programas são escritos em arquivos texto com a extensão .java
- ✓ Ao serem compilados com o compilador javac, são gerados os arquivos .class
- ✓ Um arquivo .class é constituído por bytecodes, código interpretado pela Máquina Virtual Java (Java Virtual Machine).



Estrutura de uma classe executável em Java

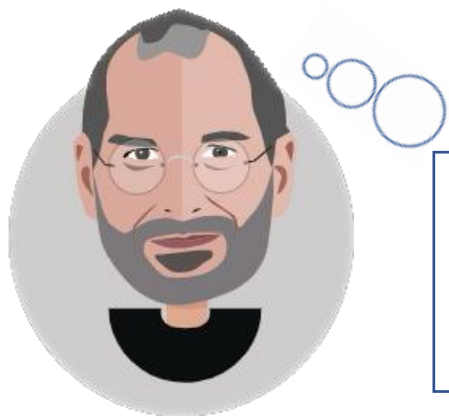
class é a palavra reservada que marca o início da declaração de uma classe.

O nome da classe deve ter o mesmo nome do arquivo

```
public class [nome] {  
    public static void main(String[] args) {  
        ...  
    }  
}
```

Bloco de instruções

Método **main**: onde inicia a execução



se
Liga
Aí

Todos os programa desenvolvidos em Java tem essa estrutura.



Estrutura de uma classe executável em Java

Nome do arquivo


Os comentários são ignorados pelo compilador

A classe e o arquivo têm mesmo nome

No comentário de uma única linha usa-se o delimitador //

Inicia-se com o delimitador /* e termina com */

```
1 package com.mycompany.exemplo;
2 /**
3  * Programa: Exibe mensagem na tela
4  * @author marco
5  */
6 public class Exemplo {
7     //O método main inicia a execução do aplicativo Java
8     public static void main(String[] args) {
9         System.out.println("Olá Mundo");
10    } //fim do método main
11 } //fim da classe Exemplo
/* Esse é um
comentário
com várias linhas*/
```



Estrutura de uma classe executável em Java

nome do
arquivo

A classe e o arquivo têm
mesmo nome

Os comentários
são ignorados
pelo compilador

OlaMundo.java

```
1 public class OlaMundo {
2     /*
3      * Programa que exibe uma mensagem na tela
4      * @autor: marco
5      */
6     public static void main(String[] args) {
7         //O método main inicia a execução do aplicativo Java
8         System.out.println("Olá mundo!");
9     } //fim do método main
10 } //fim da classe OlaMundo
```

Comentários de várias linha
usa os delimitadores `/* */`

No comentário de uma
única linha usa-se o
delimitador `//`



Material complementar



Tipos de Dados

Quando criamos um algoritmo, devemos detalhar:

- ✔ os **DADOS** (números binários, isto é, sequências de 0s e 1s, armazenados na memória, correspondem à porção das informações a serem processadas) que serão processados e
- ✔ as **INSTRUÇÕES** (ou comandos, comandam o funcionamento da máquina e determinam como devem ser manipulados os dados) que vão operar sobre esses.

O objetivo é classificar os dados de acordo com o tipo de informação contida neles. A classificação apresentada não se aplica a nenhuma linguagem de programação específica.



Tipos de Dados

- ✓ **inteiro**: informações que não possuem componente decimal ou fracionário, podendo ser positivo ou negativo.
- ✓ **real**: informações que podem possuir componentes decimais ou fracionários, podem ser positivos ou negativos. A simples existência do ponto decimal diferencia um dado numérico do tipo inteiro de um do tipo real.
- ✓ **literal ou caracteres**: é constituído por uma sequência de caracteres contendo letras, dígitos e/ou símbolos especiais. São representados nos algoritmos pela coleção de caracteres, delimitada pelas aspas (“texto”) ou aspas simples para um caracter (‘p’).
- ✓ **lógico**: informação que podem assumir apenas dois possíveis valores: verdadeiro ou falso, sim/não, 1/0, true/false.

Tipos de Dados

Classificação	Tipos	Exemplo de utilização
numérico	inteiro	idade, ano, quantidade de filhos
	real (separador de casas decimais é o ponto)	salário, peso, altura
texto	literal (representa 1 caractere, aspas simples ou sequência de caracteres, aspas duplas)	opção, primeira letra do nome, operação matemática, nome, cargo, endereço
lógico	logico (verdadeiro ou falso)	formado, solteiro

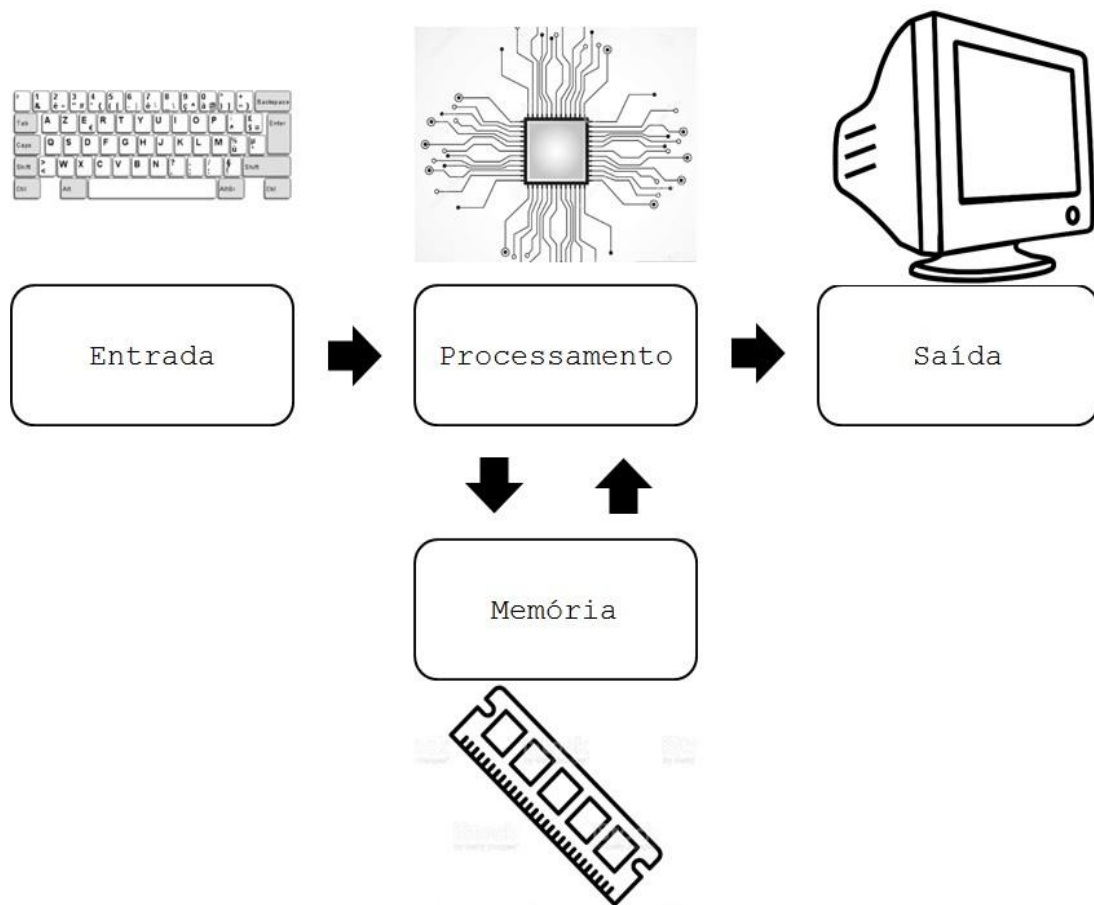
Tipos de dados primitivos em Java

Classificação	Tipo	Descrição
Lógico	boolean	Pode possuir valores true (verdadeiro) ou false (falso)
Inteiro	byte	Abrange de -128 a 127 (8 bits)
	short	Abrange de -32768 a 32767 (16 bits)
	int	Abrange de -2.147.483.648 a 2.147.483.647 (32 bits)
	long	Abrange de -2^{63} a $2^{63} - 1$ (64 bits)
Ponto flutuante	float	Abrange de $-3.4028E+38$ a $3.4028E+38$ (32 bits) com precisão simples
	double	Abrange de $-1.7976E+308$ a $1.7976E+308$ (64 bits) com precisão dupla
Caractere	char	Pode armazenar um caractere Unicode (16 bits) ou um inteiro entre 0 e 65535

É uma boa prática utilizar o tipo **double** para ponto flutuante, a menos que você tenha uma razão específica para usar **float** (como restrições de memória).

Processamento de dados

Modelo abstrato



Basicamente, um programa manipula dados que são, em geral, armazenados em variáveis localizadas na memória RAM.

- ✓ A variável modela a unidade de memória;
- ✓ Possui um identificador;
- ✓ Associa um endereço físico de memória;
- ✓ Tem um tamanho em bytes.

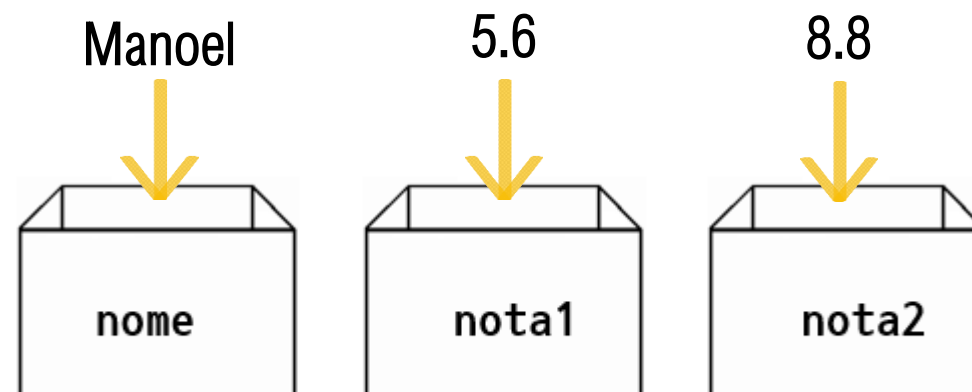
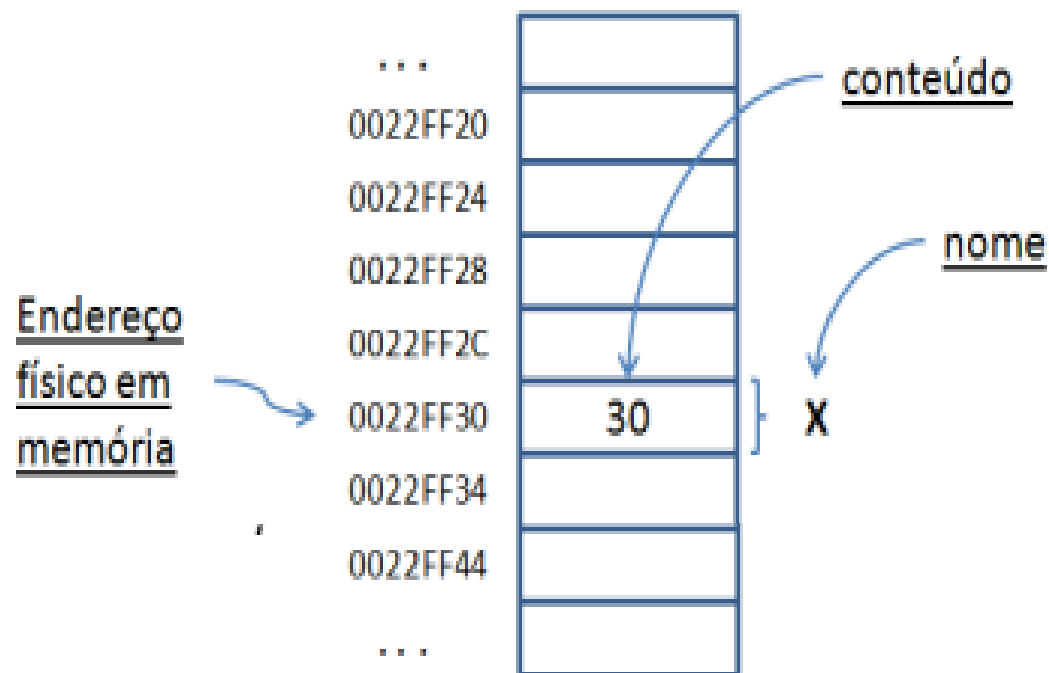
Variáveis

- ✓ Quando desejamos guardar temporariamente uma informação no computador, seja uma frase, um nome, dados numéricos ou até mesmo o resultado de um cálculo, fazemos isto na memória.
- ✓ Mas, para isto precisamos **identificar** em que parte dela estamos guardando para depois poder recuperar a informação.
- ✓ Quando fazemos esta identificação, damos um nome para esse espaço na memória onde a informação será armazenada.



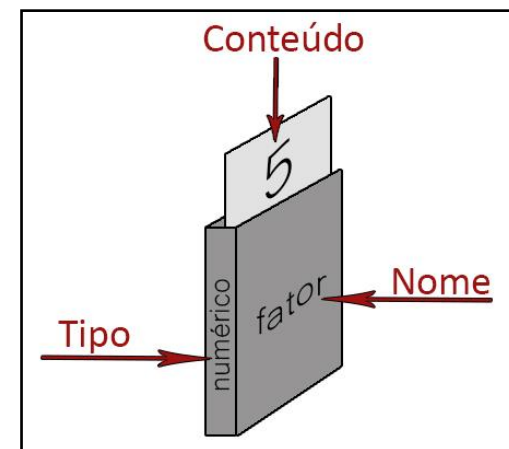
Variáveis

- ✔ Portanto, podemos definir variável como o espaço de memória que pode receber um valor e sofrer alteração no decorrer do algoritmo/tempo.
- ✔ Toda variável tem um nome único que a identifica (**identificador**), um valor e o tipo correspondente à informação a ela atribuída.



Variáveis

- ✓ Nos algoritmos, cada variável corresponde a uma posição de memória.
- ✓ Embora uma variável possa assumir diferentes valores, ela só pode armazenar um valor a cada instante.
- ✓ Uma variável possui três atributos:
 - um nome (ou **identificador**),
 - um tipo de dado e
 - a informação por ela guardada.
- ✓ Cada linguagem de programação estabelece suas próprias regras de formação de nomes de variáveis.



Identificador (nome)

É através dele que você irá se referir a variável no código para lhe atribuir algum dado ou para recuperar um dado que foi armazenado.

O identificador de uma variável deve ser representativo do seu conteúdo e seguir as seguintes regras:



- ✓ Não pode começar com números, apenas com letras.
- ✓ Não pode conter espaços em branco.
- ✓ Não pode conter caracteres especiais (#, ?, !, @, +, -, ...).
- ✓ Não pode ser palavra reservada.

Exemplos:

Válidos	Inválidos
qtde_filhos	meu nome
idade	1tentativa
nota1	Real
Nome_Completo	ficha#2

Declaração de Variáveis

- ✓ Todas as variáveis utilizadas devem ser definidas antes de serem utilizadas. Isto se faz necessário para permitir que o compilador reserve um espaço na memória para as mesmas.
- ✓ Para indicar o tipo de uma ou mais variáveis é feita a declaração de variáveis. A partir do momento da declaração das variáveis, é feita uma associação do nome escolhido, com a respectiva posição de memória.



● ● ● pseudocódigo

```
1 inteiro idade, num1
2 real nota1, media
3 literal nome
4 logico aprovado
```

● ● ● Java

```
1 int idade, num1;
2 double nota1, media;
3 String nome;
4 boolean aprovado;
```

Inicialização de variáveis

Existem várias maneiras de atribuir valores a variáveis:

- ✓ Dizendo no algoritmo qual o valor a variável deve assumir:



```
1 algoritmo compras
2     real preco
3     início
4         preco = 12.99
5     fim
```



Neste exemplo, a variável preco recebe o valor 12.99



Inicialização de variáveis

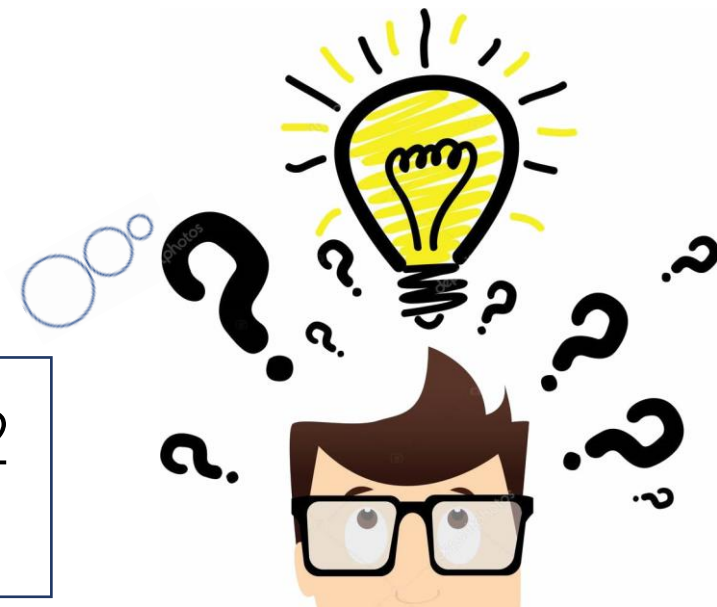
Existem várias maneiras de atribuir valores a variáveis:

- ✓ Definindo que uma variável assume o valor de uma outra variável:

```
1 algoritmo inteiros
2     inteiro n1, n2
3     início
4         n1 = 10
5         n2 = n1
6     fim
```



Qual o valor está armazenado na variável n2 após a execução deste algoritmo?????



Inicialização de variáveis

Existem várias maneiras de atribuir valores a variáveis:

- ✔ Atribuindo a uma variável o resultado de uma expressão:

```
1 algoritmo atribuicao
2     real a, b, c
3     início
4         a = 12.05
5         b = 5.20
6         c = a * b
7     fim
```



- ✔ Usuário digitando o valor utilizando um comando de entrada, como veremos a seguir.

Inicialização de variáveis

Existem várias maneiras de atribuir valores a variáveis:

- ✓ Atribuindo a uma variável o resultado de uma expressão:

```
1 public class Atribuicao {  
2     public static void main(String[] args) {  
3         double a, b, c;  
4         a = 12.05;  
5         b = 5.20;  
6         c = a * b;  
7     }  
8 }
```



- ✓ Usuário digitando o valor utilizando um comando de entrada, como veremos a seguir.

Comandos de Entrada e Saída (Input/Output)

✓ Comando de saída em Pseudocódigo:

escreva cuja finalidade é exibir uma mensagem, que pode ser um texto, o conteúdo de uma variável, ou ambos juntos!

Exemplo:

```
algoritmo soma_simples
```

```
inteiro num
```

```
inicio
```

Comando de
saída

```
← escreva("Digite um número")
```

```
leia(num)
```

```
← escreva("O número digitado foi " + num)
```

```
fim
```

Somente
mensagem

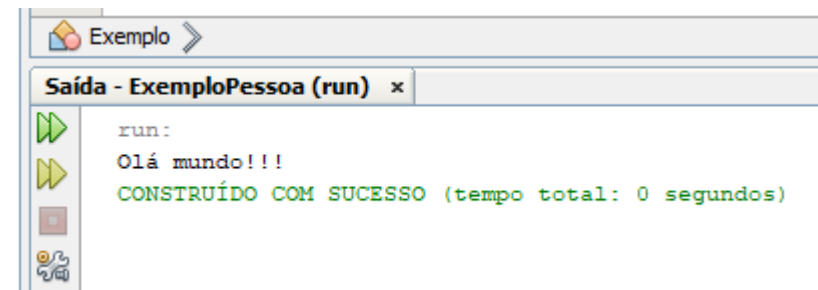
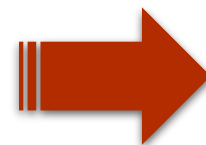
Mensagem +
variável



Comandos de saída em Java

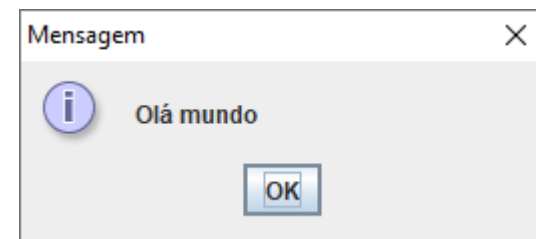
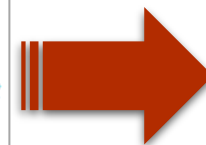
Modo texto:

```
public class Exemplo {  
    public static void main(String[] args) {  
        System.out.println("Olá Mundo!!!");  
    }  
}
```



Modo gráfico:

```
public class Exemplo {  
    public static void main(String[] args) {  
        JOptionPane.showMessageDialog(null, "Olá Mundo!!!");  
    }  
}
```



O modo gráfico requer o pacote:

```
import javax.swing.JOptionPane;
```



Exemplos de aplicação

2- Faça um programa em Java que exiba a mensagem “Olá mundo!” no modo texto e no modo gráfico.

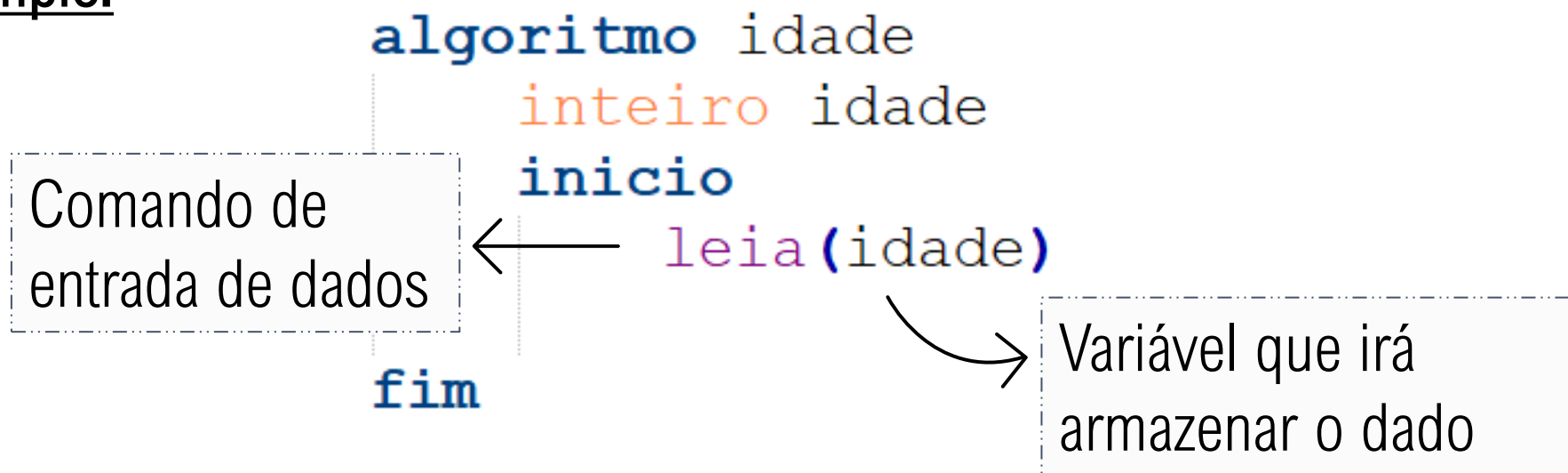
Comandos de Entrada e Saída (Input/Output)

Os algoritmos precisam ser “alimentados” com dados provenientes do meio externo para efetuarem as operações e cálculos e é necessário também mostrar os resultados.

✓ Comando de entrada em Pseudocódigo:

leia tem como finalidade atribuir o dado a ser fornecido à variável identificada.

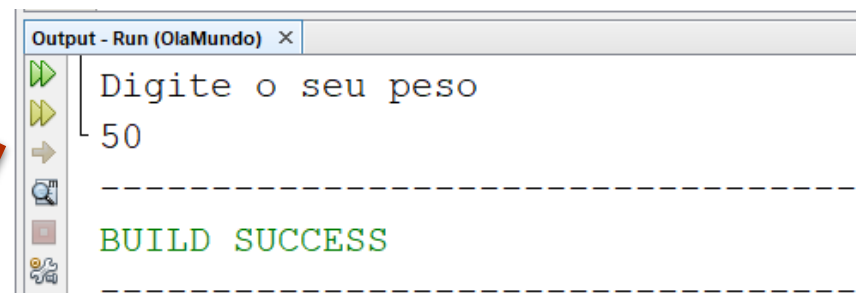
Exemplo:



Comandos de entrada em Java

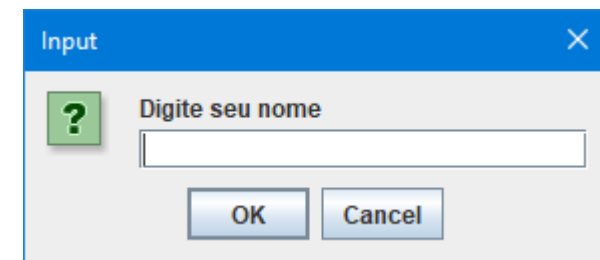
Modo texto:

```
public static void main(String[] args) {  
    Scanner leia = new Scanner(System.in);  
    System.out.println("Digite o seu peso");  
    float peso = leia.nextFloat();  
}
```



Modo gráfico:

```
public static void main(String[] args) {  
    String nome = JOptionPane.showInputDialog(null, "Digite seu nome");  
}
```



Para entrada no modo texto, importar o pacote:

```
import java.util.Scanner;
```

Para entrada no modo gráfico, importar o pacote:

```
import javax.swing.JOptionPane;
```



Comandos de entrada em Java

- ✓ Cada tipo de dado primitivo exige uma chamada do método para retornar o valor especificado na entrada de dados, sempre seguindo o formato **nextTipoDado()**.

```
Scanner leia = new Scanner(System.in);  
float peso = leia.nextFloat();  
double salario = leia.nextDouble();  
int idade = leia.nextInt();  
byte valor1 = leia.nextByte();  
long valor2 = leia.nextLong();  
boolean b1 = leia.nextBoolean();  
String nome = leia.nextLine();  
String nome = leia.next();
```

Observe o tipo de dado para String!!!



Conversões de tipos em Java

Supondo a variável x	Conversão para	y recebe o valor convertido	Resultado
int x = 10	float	float y = (float) x	10.0
float x = 10.5f	int	int y = (int) x	10
String x = "10"	int	int y = Integer.parseInt(x)	10
String x = "10.5"	double	double y = Double.parseDouble(x)	10.5
String x = "10.5"	float	float y = Float.parseFloat(x)	10.5
int x = 10 float x = 10.5	String	String y = String.valueOf(x)	"10" "10.5"



Exemplos de aplicação

3- Implemente em **Java** o algoritmo do exemplo 1 que solicita ao usuário a sua idade e profissão e armazena em duas variáveis. Após obter os dados, apresente em uma única mensagem os valores digitados (variáveis).

```
1 import java.util.Scanner;
2
3 public class Exemplo2{
4     public static void main(String[] args) {
5         int idade;
6         String profissao;
7         Scanner sc = new Scanner(System.in);
8         System.out.print("Digite a idade: ");
9         idade = sc.nextInt();
10        System.out.print("Digite a sua profissão: ");
11        profissao = sc.next();
12        System.out.println("Idade: " + idade);
13        System.out.println("Profissão: " + profissao);
14    }
15 }
```



Exemplos de aplicação

4- Crie um algoritmo que leia os valores dos lados de um retângulo e calcule/exiba o perímetro e a área do mesmo.

```
algoritmo retangulo
    real ladoA, ladoB, perim, area
    início
        escreva ("Digite o valor de um lado (em cm): ")
        leia (ladoA)
        escreva ("Digite o valor de outro lado (em cm): ")
        leia (ladoB)
        perim = 2*ladoA + 2*ladoB
        escreva ("Perímetro: " + perim + " cm ")
        area = ladoA * ladoB
        escreva ("Área do retângulo: " + area + " cm2 ")
    fim
```



Programação em Java

// Exemplo 4: área e perímetro do triângulo em Java

```
1 import javax.swing.JOptionPane;
2
3 public class Exemplo4{
4     public static void main(String[] args) {
5         double ladoA, ladoB, perimetro, area;
6         ladoA = Double.parseDouble(JOptionPane.showInputDialog(null,
7             "Digite o valor de um lado em cm: "));
8         ladoB = Double.parseDouble(JOptionPane.showInputDialog(null,
9             "Digite o valor de outro lado em cm: "));
10        perimetro = 2*ladoA + 2*ladoB;
11        JOptionPane.showMessageDialog(null, "O perímetro do triângulo é: "
12            + perimetro + "cm");
13        area = ladoA * ladoB;
14        JOptionPane.showMessageDialog(null, "A área do triângulo é: " + area + "cm²");
15    }
16 }
```



Exemplos de aplicação

5- Faça um algoritmo e um programa em Java que obtenha um número inteiro, calcule e mostre o resultado do quadrado desse número.

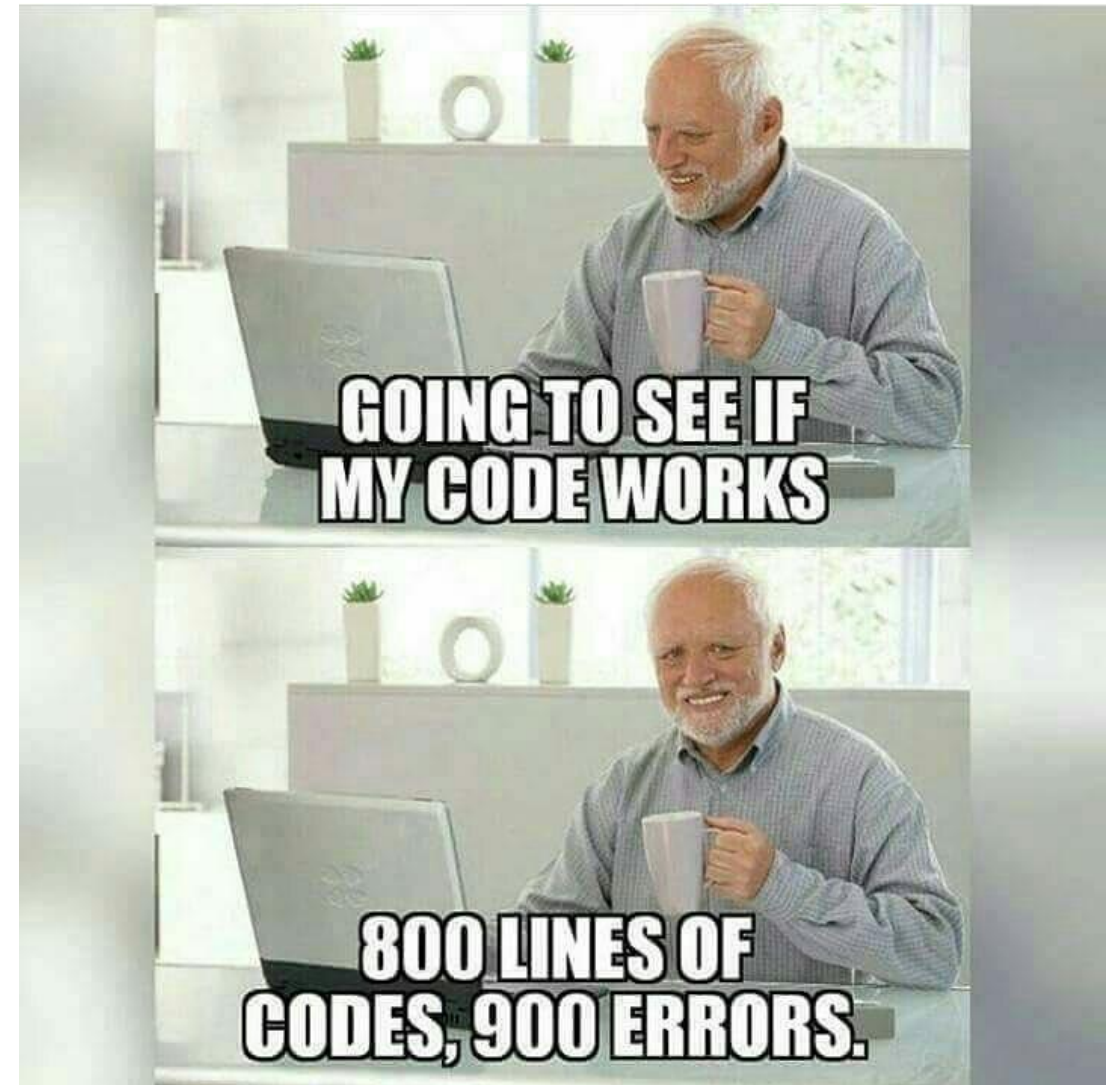
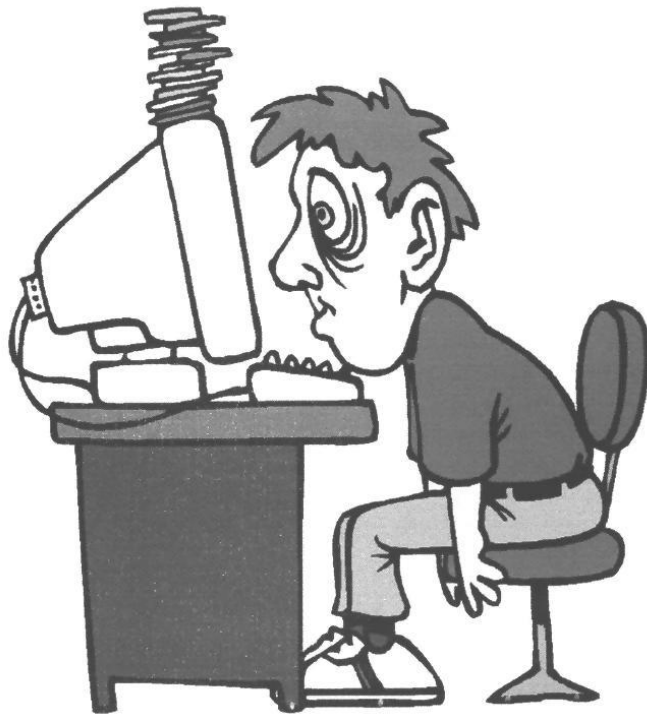
```
algoritmo Quadrado de um Número
    início
        inteiro q, n
        escreva ("Entre com o número")
        leia (n)
        q = n*n
        escreva ("O quadrado de " + n + " é " + q)
    fim
```



Alguma dúvida????



Vamos trabalhar um pouquinho?!!!



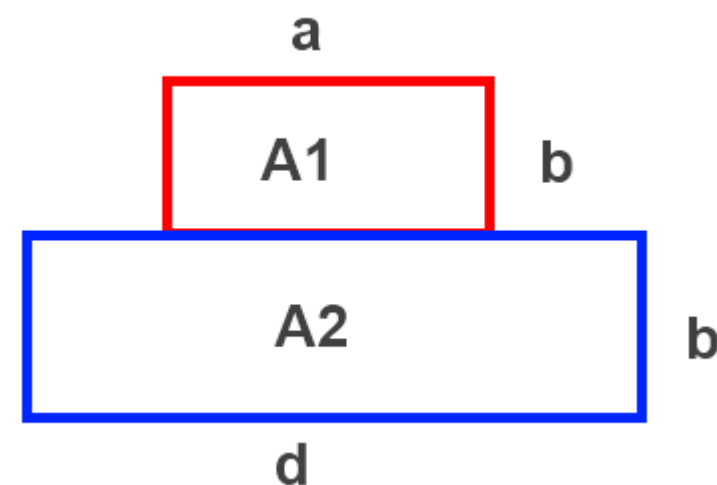
Observações sobre exercícios

- ✓ Todos os exercícios devem ser resolvidos em pseudocódigo e na linguagem Java. Os algoritmos podem ser feitos no caderno ou no Notepad++ | VSCode | Bloco de Notas. Neste último caso, os algoritmos deverão ser salvos com a extensão **.alg**.
- ✓ **Não utilizar o Word ou qualquer outra ferramenta para edição de textos. Não salvar todos os exercícios em um único arquivo!!!! Cada exercício é um arquivo diferente!**
- ✓ A entrega deverá ser feita somente pelo BlackBoard, no link correspondente a aula.



Exercícios de aplicação

- 1- Faça um algoritmo que solicite ao usuário o nome de um funcionário, a quantidade de dependentes e o cargo ocupado pelo funcionário. Mostre os dados (nome, quantidade de dependentes e cargo) digitados.
- 2- Faça um algoritmo que receba dois números inteiros, calcule e exiba a soma deles.
- 3- Elaborar um algoritmo que solicite os dados de 2 retângulos para calcular e visualizar três áreas: A_T (área total das duas figuras), A_1 e A_2 (áreas dos retângulos superior e inferior). Os únicos dados conhecidos são os valores a , b , d .



Exercícios de aplicação

4- Faça um algoritmo que leia a cotação do dólar (taxa de conversão), leia um valor em dólares e converta e mostre o valor equivalente em Reais.

5- Faça um algoritmo que leia dois valores inteiros representando, respectivamente, um valor de hora e um de minutos e informe quantos minutos se passaram desde o início do dia. **Exemplo:**

valores lidos: 13 e 15

impressão: 795 minutos

Exercícios de aplicação

6- Faça um algoritmo que leia dois números inteiros e calcule e mostre o resultado das seguintes operações aritméticas: soma, subtração e multiplicação.

7- Crie um algoritmo que obtenha um número real, calcule e mostre o valor de seu triplo.

Sugestão para implementação dos algoritmos

Para implementação dos exercícios propostos nesta aula, você pode utilizar uma ferramenta on-line de sua preferência. Seguem algumas sugestões:

- ✓ <https://www.programiz.com/java-programming/online-compiler/>
- ✓ <https://repl.it/languages/>

Outra possibilidade envolve a instalação e configuração de uma ferramenta em seu computador:

- ✓ VSCode (<https://code.visualstudio.com/download>)
- ✓ NetBeans (<https://netbeans.apache.org/download/index.html>)
- ✓ Eclipse (<https://www.eclipse.org/downloads/>)

Créditos

Esta aula foi elaborada com base no material produzido e cedido gentilmente pelos **Professores Alcides, Lédon, Ana e Cristiane, Marco Antonio.**





That's all Folks!