Universidade Cruzeiro do Sul

# Tec. de Desenvolvimento de Algoritmos

- REVISÃO MÉTODOS
- REVISÃO OPERADORES RELACIONAIS
- REVISÃO OPERADORES LÓGICOS
- ESTRUTURAS DE DECISÃO
- ESTRUTURAS DE DECISÃO ANINHADAS

Professores: Alcides/Ana/Cristiane/Ledón/Marco

2020

# Na aula passada...

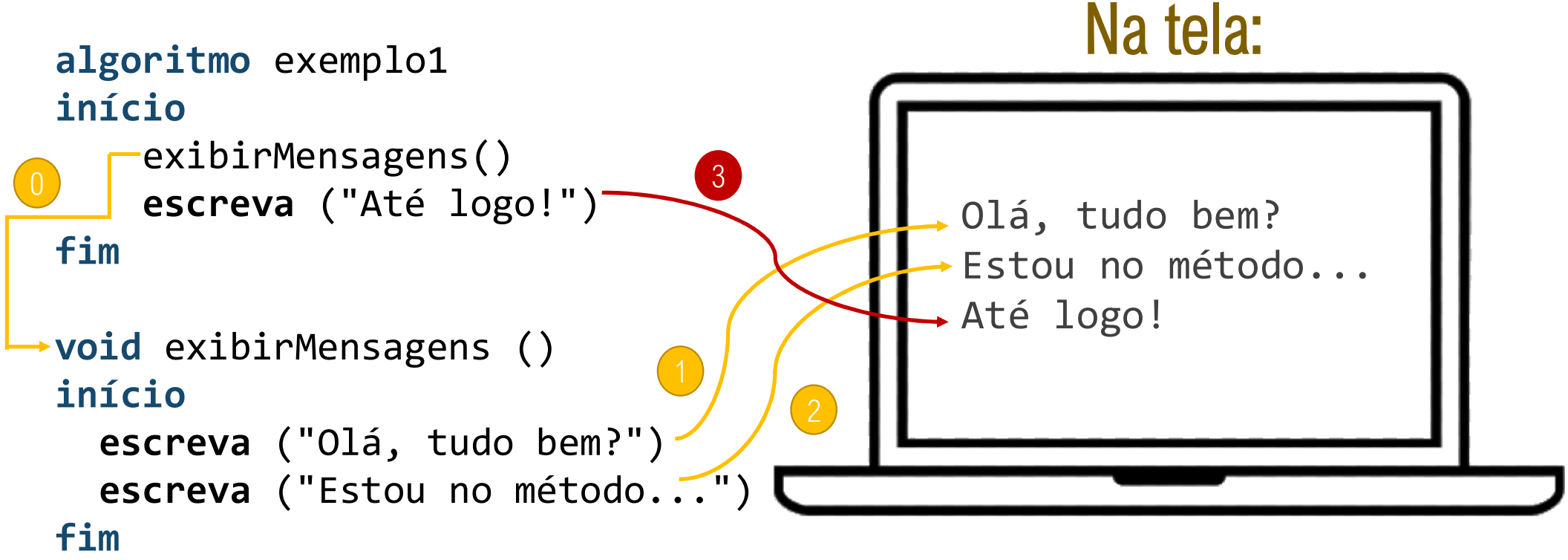
## Métodos

- Um algoritmo pode ser simplificado quando dividido em várias sub-rotinas (métodos). Os métodos podem ser classificados em: **procedimentos** (sem retorno de valor) e **funções** (com retorno de valor).
- Quando um método é chamado por um algoritmo, ele é executado e ao seu término o controle de processamento retorna automaticamente para a primeira linha de instrução após a linha que efetuou a chamada do método.

# Exemplo – utilizando um método

```
algoritmo exemplo1
início
    exibirMensagens()
    escreva ("Até logo!")
fim

void exibirMensagens ()
início
  escreva ("Olá, tudo bem?")
  escreva ("Estou no método...")
fim
```




Este método é um procedimento, porque não retorna um valor.

# Exemplo

**1-** Construir um algoritmo que calcule e visualize na tela os valores de seno, cosseno e tangente de um ângulo fornecido em graus pelo usuário. Deve existir um método com retorno que converta o ângulo de graus para radianos antes de calcular o seno, cosseno e tangente.

Utilize um método que calcule e visualize estes três resultados.

# Resposta: sen, cos, tan, utilizando procedimento

```
algoritmo Calculo_de_seno_cosseno_tangente
início
   real  ang
   escreva ("Digite o valor do ângulo em graus: ")
   leia (ang)
   visualiza (convertGToR(ang))
fim
```

```
void visualiza (real  angulo)
inicio
 escreva ("Seno: " + sen(angulo) )
 escreva ("Cosseno: " + cos(angulo) )
 escreva ("Tangente: " + tan(angulo) )
fim
```

```
real convertGToR(real  angulo)
inicio
   retorne angulo*Pi/180
fim
```

# Operadores Relacionais (relembrado)

| Operador | Significado | Exemplo | Resultado |
|---|---|---|---|
| == | igual a | 5 == 5 | Verdadeiro |
| | | 5 == 8 | Falso |
| != | diferente de | 5 != 8 | Verdadeiro |
| | | 5 != 5 | Falso |
| > | maior que | 8 > 5 | Verdadeiro |
| | | 5 > 8 | falso |
| < | menor que | 5 < 8 | verdadeiro |
| | | 8 < 5 | falso |
| >= | maior ou igual | 8 >= 5 | verdadeiro |
| | | 5 >= 8 | falso |
| <= | menor ou igual | 5 <= 8 | verdadeiro |
| | | 8 <= 5 | falso |

- Comparações só podem ser feitas entre objetos de mesma natureza, isto é, variáveis do mesmo tipo de dado.
- O resultado de uma comparação será sempre um valor lógico.

# Lógicos (relembrado)

| Operadores | Java | Python |
|---|---|---|
| **ou** | \|\| | or |
| **e** | && | and |
| **não** | ! | not |

| A | B | A \|\| B | A && B | ! A |
|---|---|---|---|---|
| **F** | F | F | F | V |
| **F** | V | V | F | V |
| **V** | F | V | F | F |
| **V** | V | V | V | F |

Observe que:

- ou → basta que um dos seus valores seja V para que o resultado seja V.
- e → é necessário que todos os valores sejam V para que o resultado seja V.

# Operadores Lógicos

**Exemplo:** Considere A = 10, B = 5, C = 7, D = 3, F = 5

a) A > C → Verdadeiro
b) A < B → Falso
c) A == D → Falso
d) A != C → Verdadeiro
e) B <= F → Verdadeiro
f) A >= D → Verdadeiro
g) A > B e C > D → Verdadeiro
h) C > B ou A == D → Verdadeiro
i) C == B ou A == D → Falso
j) !(A == D) → Verdadeiro
k) !(B == F) → Falso

# Prioridade entre os operadores

| Operadores | Prioridade |
|---|---|
| Lógicos | 4º |
| Relacionais | 3º |
| Aritméticos | 2º |
| Parênteses | 1º |

# Estrutura Sequêncial

Nesta estrutura os comandos de um algoritmo são executados numa sequência pré-estabelecida. Cada comando é executado somente após o término do comando anterior.

```
algoritmo exibeIdade
inicio
  inteiro idade
  escreva(“Digite a sua idade”)
  leia(idade)
  escreva(“A idade é: “ + idade)
fim
```

# Estruturas de decisão

Utilizadas para controlar o fluxo de execução dos programas, possibilitando que caminhos alternativos sejam seguidos de acordo com resultado de uma condição ou mais condições.

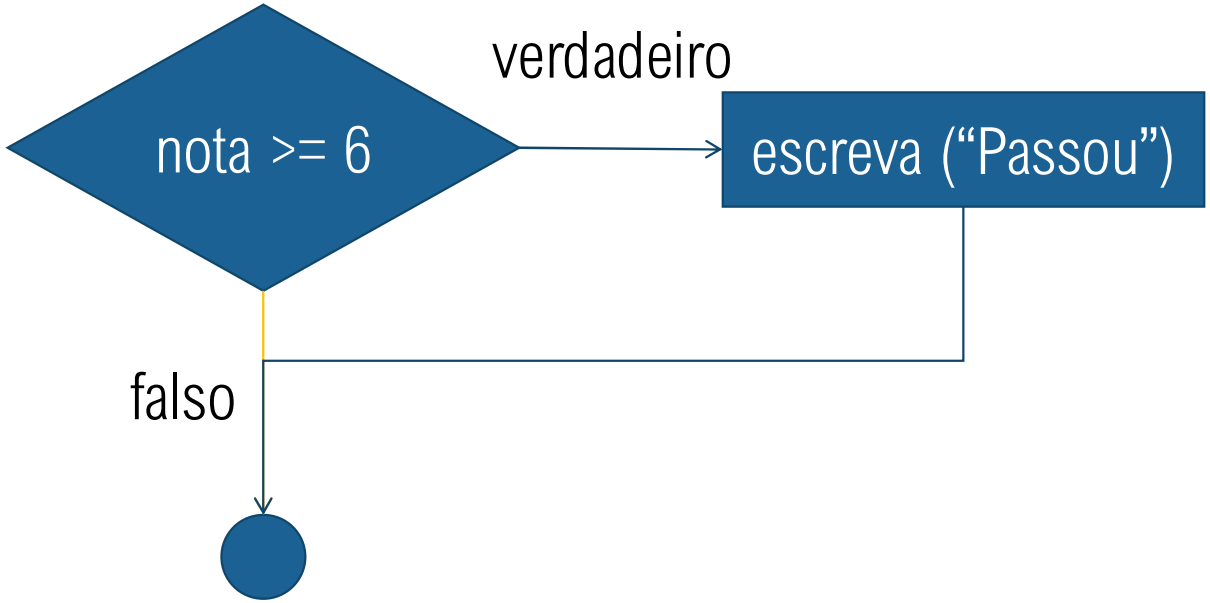
As estruturas de decisão existentes são:

- a estrutura **se (if)**
- a estrutura **se-senão (if-else)**
- a estrutura **escolha-caso (switch-case)**

# Estrutura de Decisão - SE

Define ações para a condição verdadeira:

```
se (nota >= 6)
    escreva(“Passou”)
```



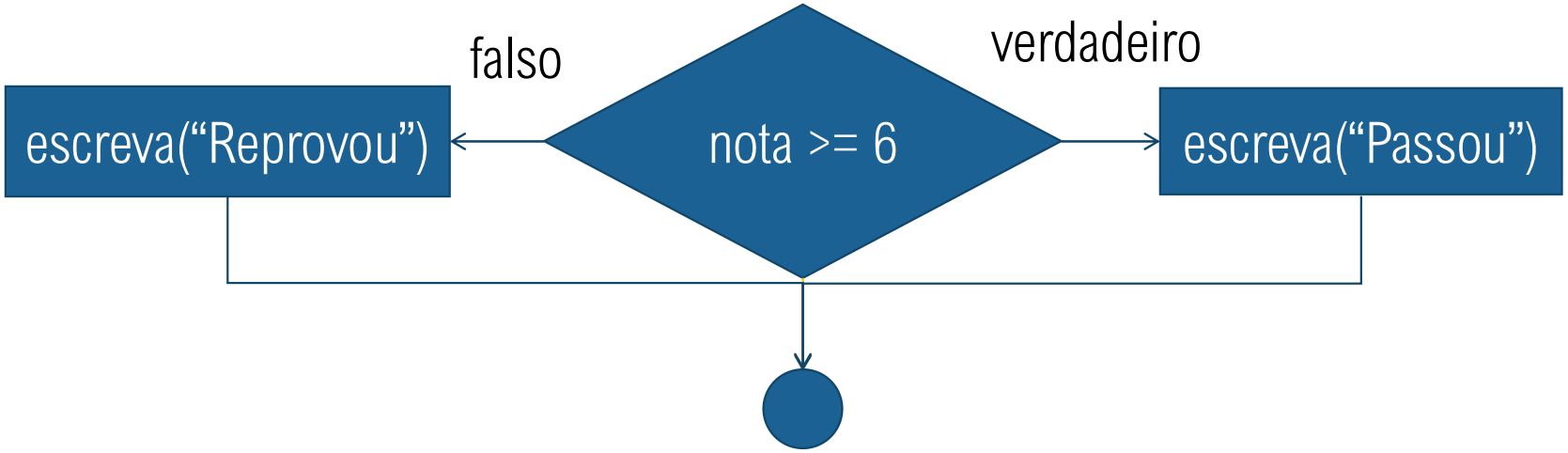
# Observações importantes

- A diretiva **SE** deve ter suas expressões contidas entre parênteses.
- O único argumento válido para um **SE** é uma expressão lógica ou variável booleana (condição).
- Preste atenção nos sinais de comparação (==) dentro de um **SE**, pois eles podem ser confundidos com o operador de atribuição (=).
- As chaves não são obrigatórias para blocos **SE** que têm apenas uma instrução, mas tome cuidado com erros de endentação.

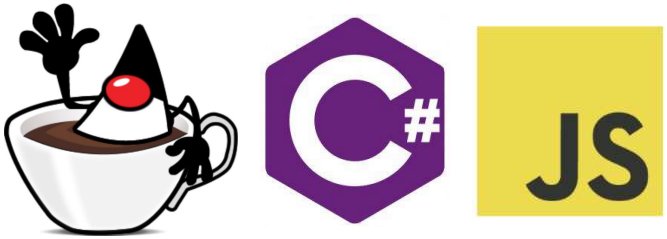
# Estrutura de Decisão - Se…Senão

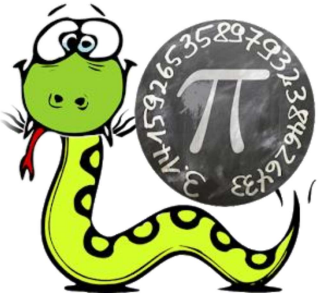
Define ações para a condição verdadeira e também para a falsa:

```
se (nota >= 6)
    escreva(“Passou”)
senão
    escreva (“Reprovou”)
```



# Tomada de decisão do tipo se-então-senão

```
No pseudocódigo

se (condição) então
    grupo_1_de_comandos
senão
    grupo_2_de_comandos
```

```
//Em Java,
JavaScript, C#

if (condição) {
   grupo_1_de_comandos
}
else {
   grupo_2_de_comandos
}
```

```
# Em Python

if  condição:
   grupo_1_de_comandos
else:
   grupo_2_de_comandos
```

# Estrutura de decisão Simples – se

É chamada de estrutura de decisão única ⇒ seleciona uma única ação ou um grupo de ações.

**Pseudocódigo**

```
se(condição)
{
   instruções
}
```

Executa as **<instruções>** se a condição for verdadeira

**condição** é uma expressão lógica e, portanto, deve resultar no valor lógico F ou V.

**Java/JavaScript/C#**

```
if(condição)
{
   instruções
}
```

**Python**

```
if condição:
   instruções
```

# Exemplos

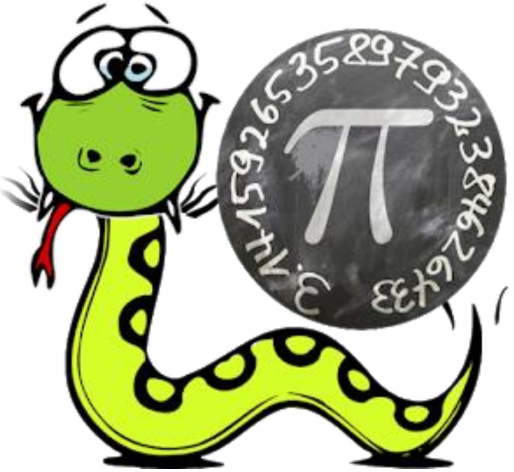
2- Escreva um algoritmo que solicite um número inteiro ao usuário e mostre-o caso o mesmo seja par.

```
algoritmo par
inicio
    inteiro num
    escreva (“Digite um número inteiro”)
    leia (num)
    se (num%2==0) {
        escreva (“O número: “ + num + “ é par”)
    }
fim
```

# Pausa para programação

```
# Exemplo2: programa que solicita um número
# inteiro ao usuário e mostre-o caso
# o mesmo seja par.

num = int(input("Entre com um número inteiro"))
if num % 2 == 0:
    print("O número: " , num , " é par.")
```



# Pausa para programação

```java
#Exemplo2: programa Java que verifica se o número é par.
import java.util.Scanner;
class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Digite um número inteiro");
      int x = sc.nextInt();
      if(x % 2 == 0){
        System.out.println("O número " + x + " é par");
      }
   }
}
```

# Pausa para programação

```
#Exemplo2: programa C# que verifica se o número é par.
using System;
public class Main {
     public static void main() {
          int x;
          Console.Write ("Digite um número inteiro");
          x = int.Parse(Console.ReadLine());
          if(x % 2 == 0){
             Console.Write ("O número " + x + " é par");
          }
      }
}
```