

# LÓGICA DE PROGRAMAÇÃO

Professor Jefferson Chaves  
[jefferson.chaves@ifc-araquari.edu.br](mailto:jefferson.chaves@ifc-araquari.edu.br)



Técnico Integrado em  
**INFORMÁTICA**

- Conhecer e entender as estruturas de fluxo:
  - Sequencial;
  - Seleção ou tomada de decisão;
  - Repetição;
- Aplicações de cada uma;



# **LÓGICA DE PROGRAMAÇÃO | ESTRUTURA SEQUENCIAL**

Jefferson de Oliveira Chaves - IFC  
Araquari

- A característica desse tipo de algoritmo:
  - O conjunto de instruções será executado de forma linear, de cima para baixo;
  - Executa todas as linhas do algoritmo;
  - Não há qualquer desvio no código;



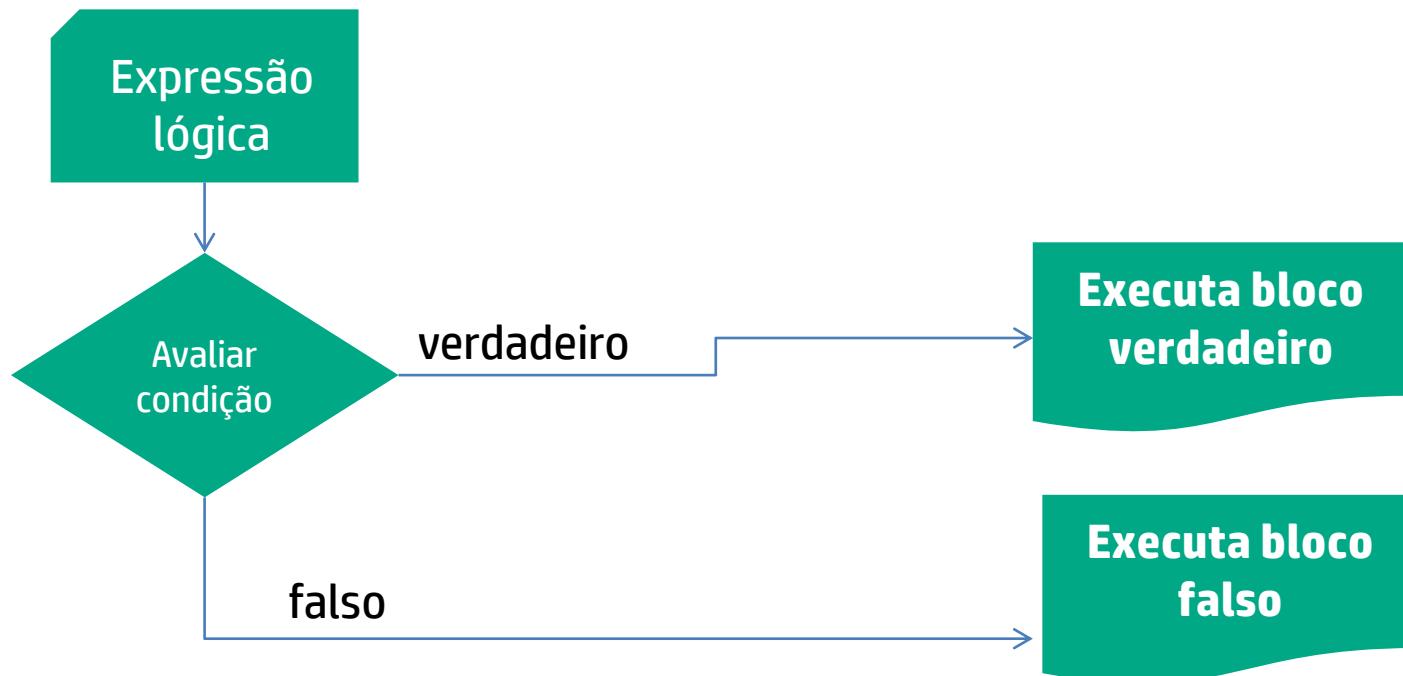
Construa um algoritmo para calcular as raízes de uma equação do 2º grau ( $ax^2 + bx + c$ ), sendo que os valores  $a$ ,  $b$  e  $c$  deverão ser fornecidos pelo usuário.



# **LÓGICA DE PROGRAMAÇÃO | ESTRUTURA DE SELEÇÃO**

Jefferson de Oliveira Chaves - IFC  
Araquari

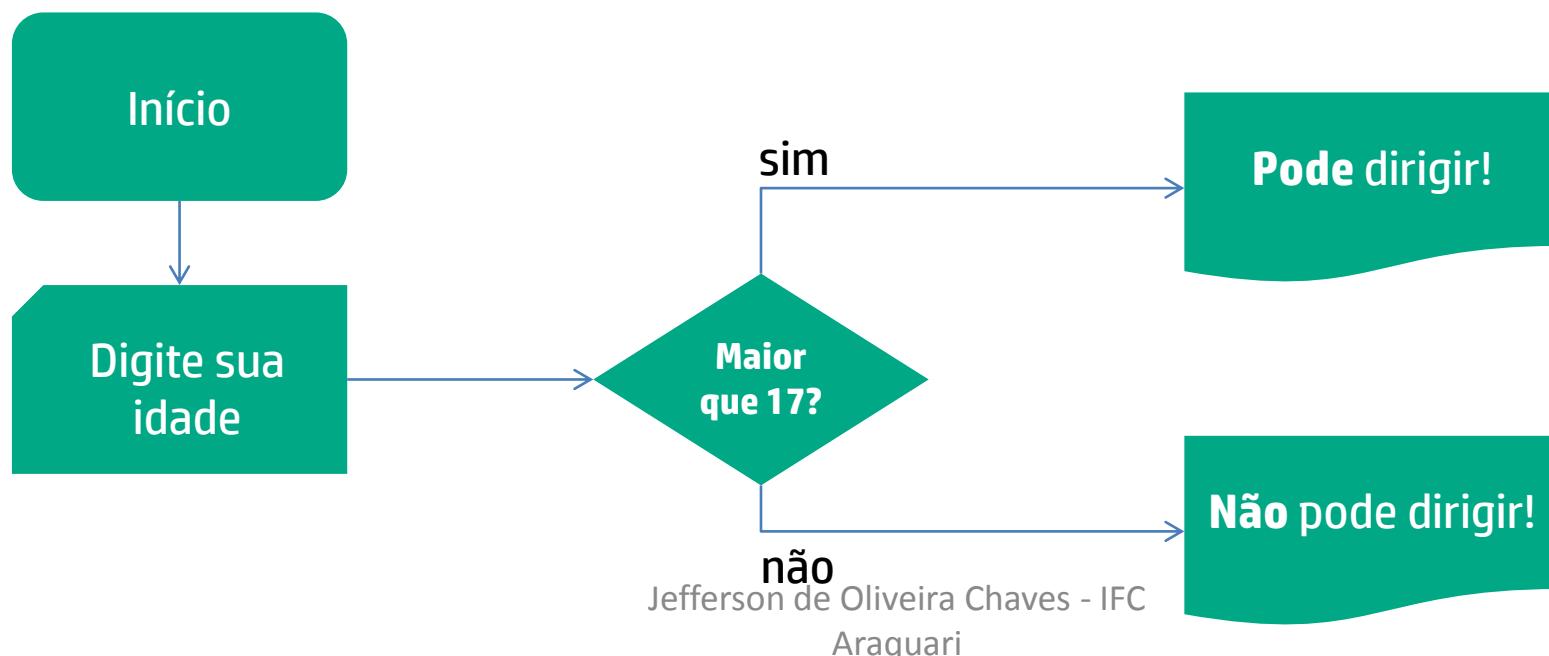
- Estruturas de seleção (ou decisão) permite ao algoritmo **executar uma ação ao invés de outra, baseada em uma condição lógica**;



# LÓGICA DE PROGRAMAÇÃO

# ESTRUTURA DE SELEÇÃO

- Por meio dos fluxos de controle que diremos aos nossos programa que ele deve realizar uma ação ao invés de outra.



- Existem três tipos de estruturas de seleção:
  - Simples;
  - Composta;
  - Encadeada;

# INSTRUÇÃO CONDICIONAL SE

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



- Seleção simples:
  - **Se** a <condição> for verdadeira então o bloco verdade será executado;
  - **Caso contrário**, nada acontece;

bloco de  
instruções  
que serão  
executada  
s caso a  
condição  
seja  
verdadeira

condição

if (condição) {

instrução 1;  
instrução 2;  
⋮  
instrução n;

}

Simples

## Exemplo #1

```
<?php  
//exemplo seleção simples  
$idade = 21;  
  
if ($idade >= 18) {  
    echo "Maior de idade";  
}  
}
```

usuário digitou 19. instrução  
do bloco será executada

## Exemplo #1

```
<?php  
    echo "digite sua idade: ";  
    $idade = fgets(STDIN);  
  
    if ($idade >= 18) {  
        echo "Maior de idade";  
    }  
}
```

digite sua idade: 19  
Maior de idade



# INSTRUÇÃO CONDICIONAL SE ou SENÃO

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



- **Seleção composta:**
  - Deve ser usada em algoritmos em que duas **alternativas** dependam da mesma decisão: uma **verdadeira** e a outra **falsa**;

Composto

**bloco verdadeiro**  
instruções que  
serão executadas  
caso a condição  
seja verdadeira

**bloco falso**  
instruções que  
serão executadas  
caso a condição  
seja falsa

```
if (condição) {  
    instrução v1;  
    instrução v2;  
    instrução vn;  
}  
  
else {  
    instrução f1;  
    instrução f2;  
    instrução fn;  
}
```

## Exemplo #2

```
<?php
//exemplo seleção composta
$idade = 15; //quinze aninhos

if ($idade >= 18) {
    echo "Maior de idade";
}

else {
    echo "menor de idade";
}
```

```
<?php  
echo "digite sua idade: ";  
$idade = fgets(STDIN);  
  
if ($idade >= 18) {  
    echo "Maior de idade";  
}  
  
else {  
    echo "menor de idade";  
}
```

## Exemplo #2

usuário digitou 15.  
instrução do bloco  
será executada

```
digite sua idade: 15  
Menor de idade
```



- **Seleção encadeada:**

- Quando uma condição puder ter mais de uma condição verdadeira possível, usamos a **seleção encadeada**;

Composto

**bloco verdadeiro**  
instruções que  
serão executadas  
caso a condição  
seja verdadeira

**bloco falso**  
instruções que  
serão executadas  
caso a condição  
seja falsa

```
if (condição) {  
    instrução v1;  
    instrução v2;  
    instrução vn;  
}  
  
elseif(condicao) {  
    instrução v1;  
    instrução v2;  
    instrução vn;  
}  
else {  
    instrução f1;  
}
```

## Exemplo #3

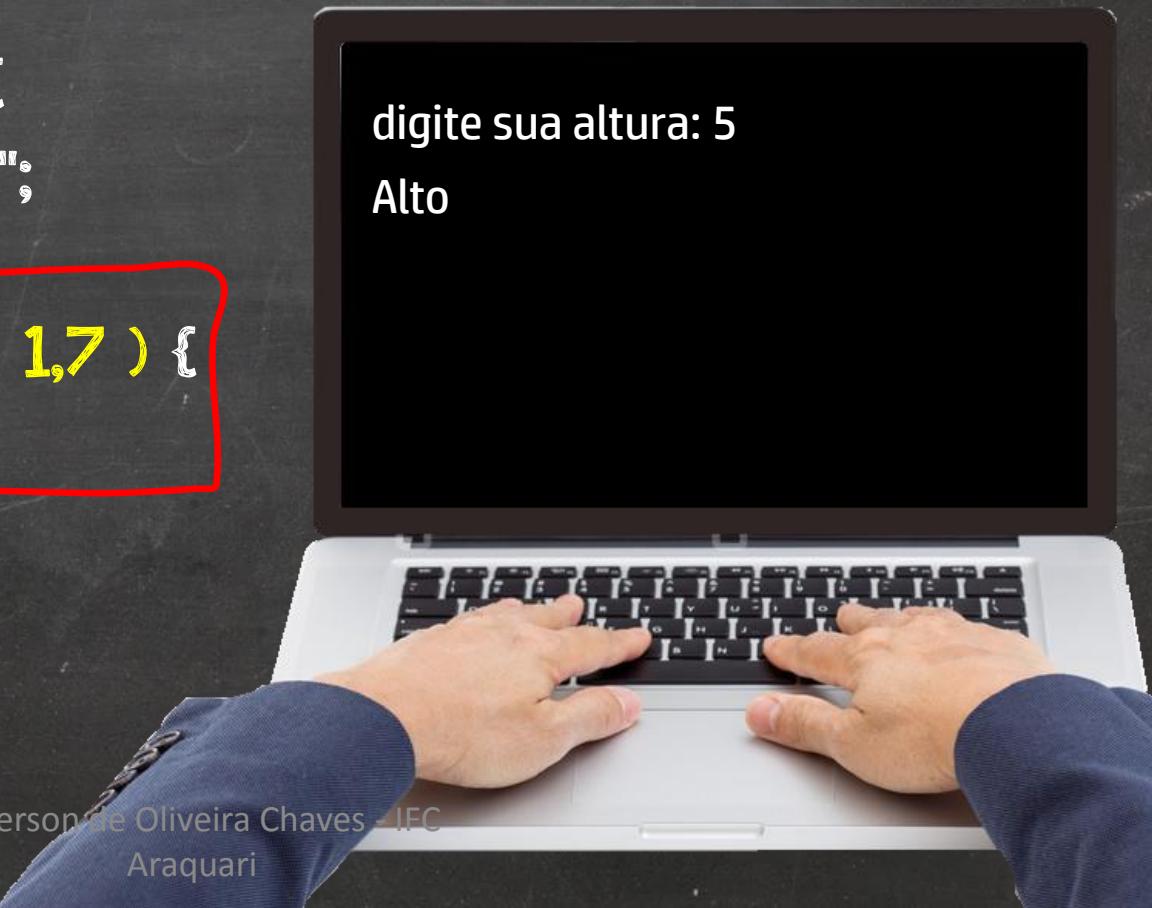
```
<?php
```

```
echo "digite sua altura: ";
$altura= fgets(STDIN);
```

```
if ($altura > 2,0 ) {
    echo "Gigante";
}

} elseif ($altura >= 1,7 ) {
    echo "Alto";

} else {
    echo "Baixo";
}
```



```
digite sua altura: 5
Alto
```

- Operador ternário:

- O operador ternário, permite de forma resumida expressar uma condição **se** ou **senão**;
- Veja a sintaxe no exemplo baixo:

(condição) ? <condição verd.> : <condição falsa>;

## LÓGICA DE PROGRAMAÇÃO

## OPERADOR TERNÁRIO

```
<?php
```

```
$valor = 10;
```

```
($valor>50) ? “maior” : “menor”;
```

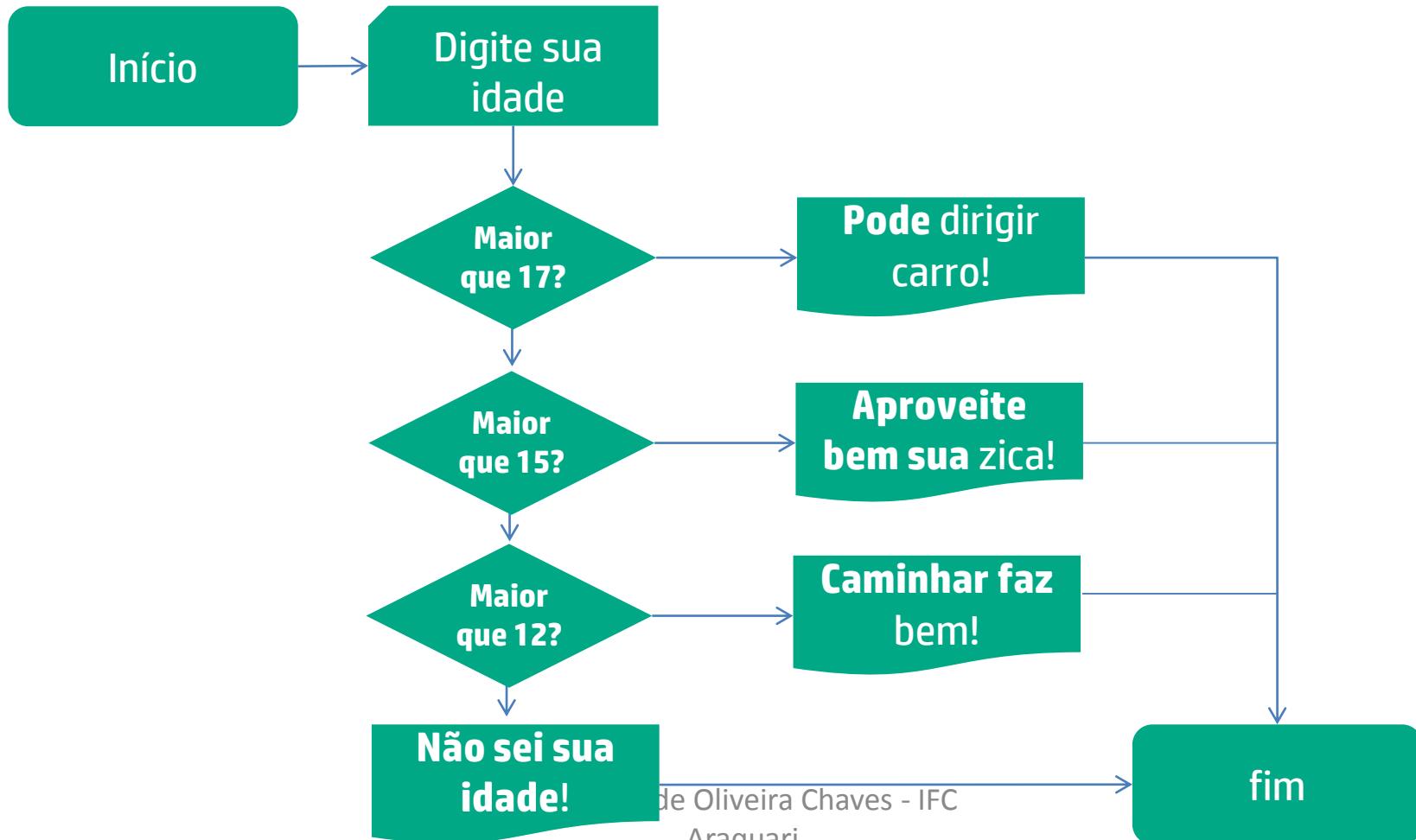
# SELEÇÃO DE MÚLTIPLA ESCOLHA - SWITCH

```
switch ( opção ) {  
    case "op"  
        //instrução  
        break;  
  
    default:  
        //instrução  
}
```



- A instrução **switch** permite que você realize operações de fluxo de controle mais complexas **sem usar** uma infinidade de if's e else's encadeados;
- O **switch** é comumente utilizado quando se deseja **comparar** um valor com uma LISTA de **valores já definidos**;

## A instrução **switch**:



```
<?php
```

```
echo "digite seu prato favorito: ";
$opcao = fgets(STDIN);
```

```
switch($opcao) {
    case 'arroz':
        echo 'Você escolheu arroz';
        break;

    case 'batata':
        echo 'Você escolheu batata';
        break;

    default:
        echo 'Enjoado hein?!';
}
```

## Exemplo #3

```
digite seu prato favorito: arroz
Você escolheu arroz
```



# LÓGICA DE PROGRAMAÇÃO | ESTRUTURAS DE REPETIÇÃO



- Pergunta do mestre ao jovem gafanhoto:
- Quando usar o **if**?
- E o **else**?
- Qual a principal diferença entre o **if** e **switch**?

1. Elabora um algoritmo que leia o valor de dois números inteiros e a operação aritmética desejada; calcule então a resposta adequada;

SÍMBOLO	OPERAÇÃO
+	ADIÇÃO
-	SUBTRAÇÃO
/	DIVISÃO
*	MULTIPLICAÇÃO

1. Dados três A, B, C verificar se eles podem ser os comprimentos dos lados de um triângulo, se forem, verificar se compõem um triângulo equilátero, isósceles ou escaleno. Informar se não compõem nenhum triângulo.
  - Dados de entrada: três lados de um provável triangulo;
  - Dados de saída: Não compõem triângulo, triângulo equilátero, triângulo isósceles ou triangulo escaleno.

# LÓGICA DE PROGRAMAÇÃO

# EXERCÍCIOS

1. Construa um algoritmo que, tendo como dados de entrada o preço de um produto e seu código de origem, mostre o preço junto com sua procedência. Caso o código não seja nenhum dos especificados, ele deve ser encarado como importado.

CÓDIGO DE ORIGEM	PROCEDÊNCIA
1	SUL
2	NORTE
3	NORDESTE
4, 5 ou 6	SUDESTE
7, 8 ou 9	CENTRO-OESTE



**LÓGICA DE  
PROGRAMAÇÃO**

**ESTRUTURAS DE  
REPETIÇÃO**

# LÓGICA DE PROGRAMAÇÃO

- Por vezes precisaremos repetir uma instrução de acordo com uma condição lógica;
- Nesses casos usamos estruturas de repetição (loop):
  - While ([enquanto](#));
  - Do while ([faça, enquanto](#));
  - For ([loop para](#));
  - Foreach;



E agora? Tenho que  
mostrar os  
números na  
sequência de 0 a  
100. Estou frito!

# LÓGICA DE PROGRAMAÇÃO

<?php

echo 1;

echo 2;

echo 3;

[...]

//deve ter um meio mais inteligente

# INSTRUÇÃO de REPETIÇÃO WHILE

```
while(condição verdadeira) {  
    //instruções  
}
```



condição

Laço de  
Repetição

while (condição)

{

instrução 1;

instrução 2;

:

instrução n;

}

bloco de instruções

instruções que  
serão repetidas  
caso a condição  
seja verdadeira

## *Exemplo #1*

//O que acontece ao se executar  
//o algoritmo abaixo?

```
$numero = 0;
```

```
while ($numero < 3) {
```

```
    echo $numero;
```

```
}
```

## *Exemplo #1*

//O que acontece ao se executar  
//o algoritmo abaixo?

```
$numero = 0;  
  
while ($numero < 3) {  
    echo $numero;  
    $numero = $numero + 1;  
}
```

## Exemplo #2

```
$numero = ?;
```

```
while ($numero != 0) {  
    $numero = fgets(STDIN);  
    echo "você digitou $numero";  
}
```

# INSTRUÇÃO de REPETIÇÃO DO WHILE

```
do {  
    //instruções  
} while(condição verdadeira);
```



do

{

instrução 1;  
instrução 2;  
:  
instrução n;

} while (condição)

condição

define quando o laço  
deve continuar

Laço de  
Repetição

é realizado  
antes da  
primeira  
iteração e  
cada  
iteração

## Exemplo #2

```
do {
```

```
    $numero = fgets(STDIN);  
    echo "você digitou: $numero \n";
```

```
} while ($numero != 0);
```

## Importante!

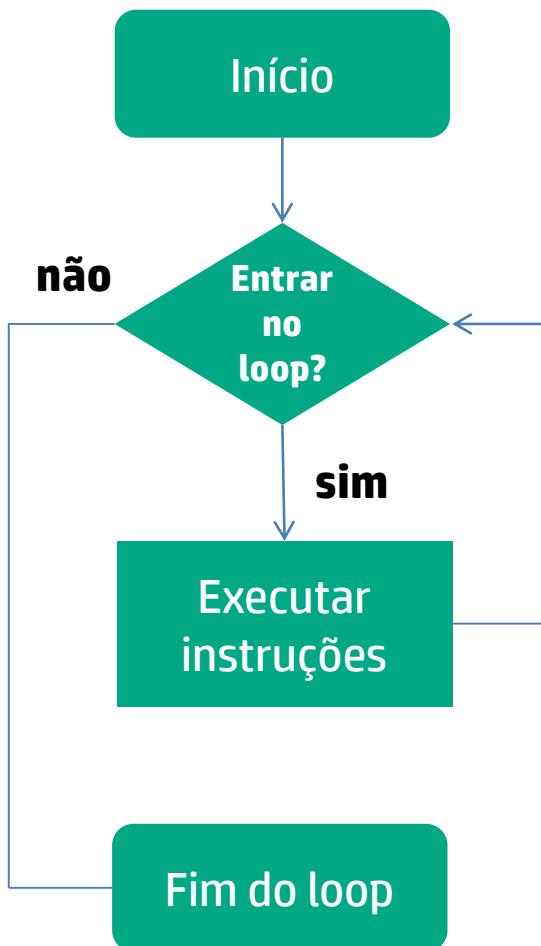
A diferença entre o **while** e o **do while** é que este é executado pelo menos uma vez.



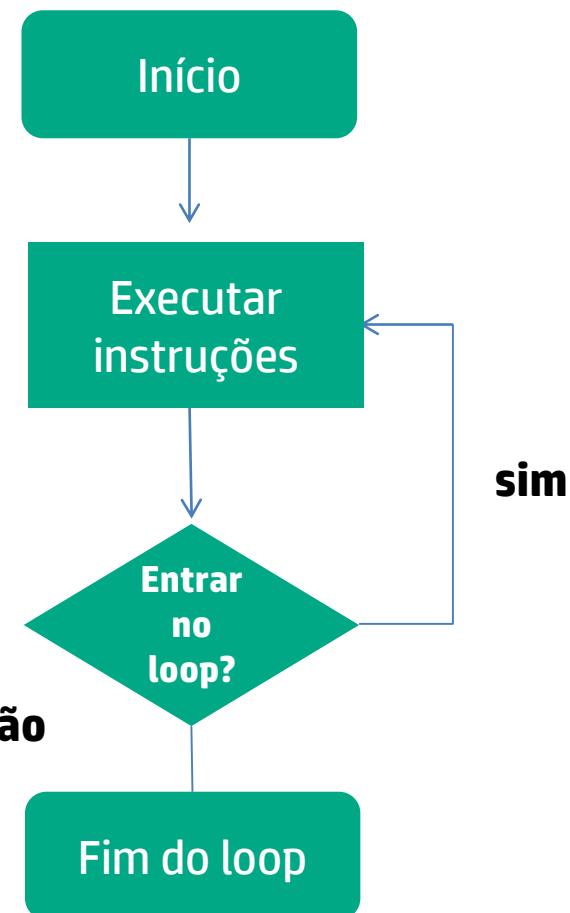
# LÓGICA DE PROGRAMAÇÃO

# ESTRUTURAS DE REPETIÇÃO

## WHILE



## DO WHILE



# INSTRUÇÃO de REPETIÇÃO

## FOR

```
for ($i; $i < 10; $i++){  
    //instruções  
}
```



Acontece  
apenas uma  
vez

define quando o  
laço deve continuar

Laço de  
Repetição

for (inicialização; condição; incremento)

{

instrução 1;  
instrução 2;  
:  
instrução n;

serão  
repetidas  
a cada  
iteração

}

acontece ao  
final de cada  
iteração, após o  
bloco de  
instruções

## *Exemplo #1*

```
for($cont =0; $cont <= 3; $cont++){  
    echo $cont;  
}
```

## Exemplo #1

```
$tabuada = trim(fgets(STDIN));
```

```
for($i =1; $i <= 10; $i++) {  
    echo "In $i * $tabuada:" . $i * $tabuada  
}
```

# CONTROLANDO LOOPS: BREAK E CONTINUE

```
for ($i; $i < 10; $i++){  
    if(true){  
        break;  
    }  
}
```



- Apesar de termos condições booleanas nos nossos laços, **em algum momento, podemos decidir parar o loop** por algum motivo especial sem que o resto do laço seja executado usando o **break**:

```
for ($i = 0; $i < 1000; $i++ ) {  
    if ($i == 777){  
        echo "Achei o número 777";  
        break;  
    }  
}
```

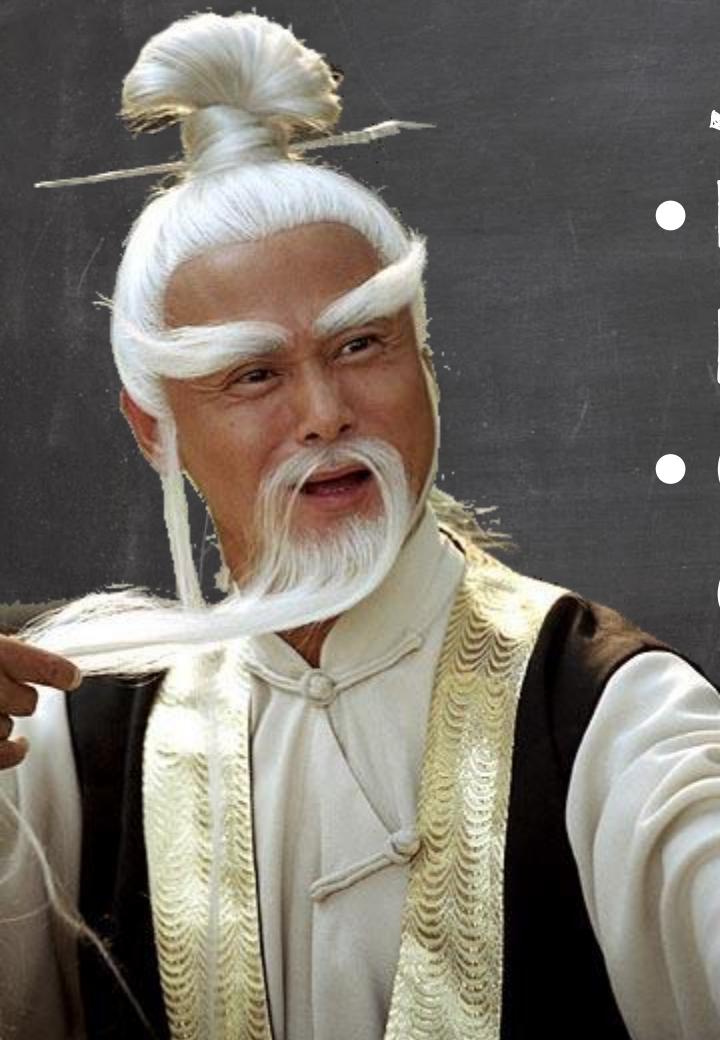
- Também é possível obrigar o loop a executar o próximo laço. Para isso usamos a palavra chave **continue**.

```
for ($i = 0; $i <= 100; $i++) {  
    if ($i > 50 && $i < 60) {  
        continue;  
    }  
    echo $i;  
}
```

## LÓGICA DE PROGRAMAÇÃO

## ESTRUTURAS DE REPETIÇÃO

- Pergunta do mestre ao jovem gafanhoto:
- Mas se temos o **while** para que o bendito **for**?
- Qual as principais diferença?



1. Faça um programa que calcule o mostre a média aritmética de N notas.
2. Numa eleição existem três candidatos. Faça um programa que peça o número total de eleitores. Peça para cada eleitor votar e ao final mostrar o número de votos de cada candidato.
3. Faça um programa que calcule o fatorial de um número inteiro fornecido pelo usuário. Ex.:  $5!=5 \cdot 4 \cdot 3 \cdot 2 \cdot 1=120$ . A saída deve ser conforme o exemplo abaixo:

1. Faça um jogo que o algoritmo sorteie um número e o usuário informa outro. Compare os valores e ao ser igual mostre a mensagem “venceu o jogo”.
2. Exiba as mensagens perdeu ao perder e ganhou ao ganhar.  
Repita **enquanto** não ganhar.

1. Aprimore seu jogo. Informe ao usuário ao errar, se o número informado foi mais alto ou mais baixo que o sorteado.
2. Informe também o número de tentativas.

1. Faça 3 versões de um algoritmo que imprima a tabuada do número 5.
2. Use as estruturas de repetição while, do while e for.