

# PROGRAMAÇÃO DE SOLUÇÕES COMPUTACIONAIS

Prof. Ricardo Ribeiro Assink

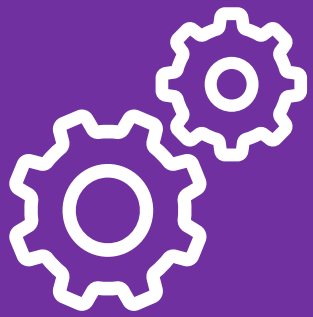




## ESTRUTURAS DE REPETIÇÃO: Para.. Faça - FOR

Juntamente com as estruturas de seleção, as estruturas de repetição são de crucial importância para a programação do algoritmo.

As estruturas de repetição nos possibilitam executar o mesmo trecho de código várias vezes seguidas, enquanto um dado critério não é satisfeito.



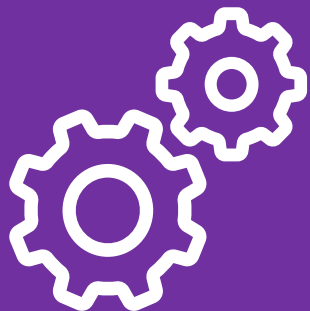
# ESTRUTURAS DE REPETIÇÃO: Para.. Faça - FOR

```
for( < VI > ; < CP >; < RC > ){  
    <comando 1>;  
    <comando 2>;  
}
```

VI Valor inicial do contador

CP Condição de parada

RC Regra do contador



# (JAVA) ESTRUTURAS DE REPETIÇÃO: Para.. Faça - FOR

```
import javax.swing.JOptionPane;  
public class Exemplofor {  
    public static void main(String[] args) {
```

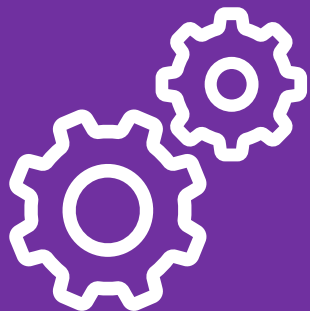
```
        for ( int i = 0; i < 5; i++ ){  
            JOptionPane.showMessageDialog(null,"Valor de i: " + i);  
        }
```

```
    }  
}
```

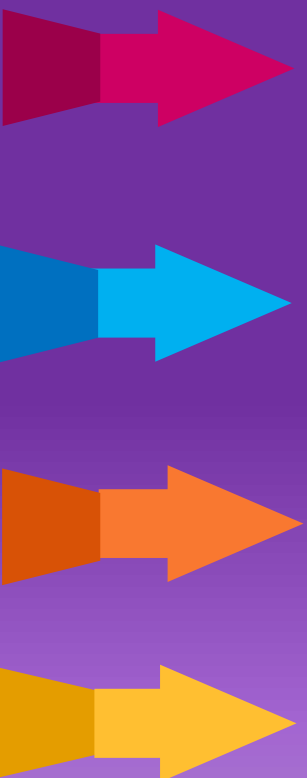


# ESTRUTURAS DE REPETIÇÃO: Enquanto.. Faça: WHILE

Para repetir um conjunto de instruções enquanto uma certa condição for satisfeita

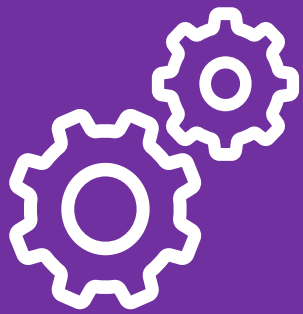


# ESTRUTURAS DE REPETIÇÃO de Repetição: Enquanto.. Faça - WHILE

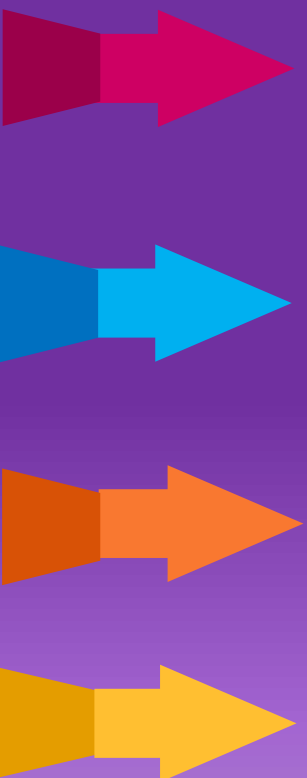


```
while(< CP > ){  
    <comando 1>;  
    <comando 2>;  
}
```

**CP** Condição de parada



# (JAVA) ESTRUTURAS DE REPETIÇÃO: Enquanto.. Faça - WHILE

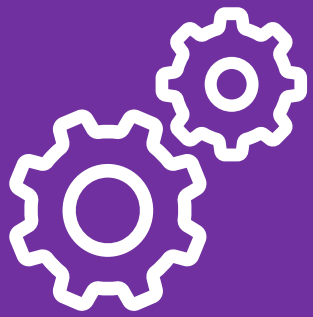


```
import javax.swing.JOptionPane;
public class Exemplowhile {
    public static void main(String[] args) {
        int i = 0;
        while( i < 5){
            JOptionPane.showMessageDialog(null,"Valor de i: " + i);
            i++;
        }
    }
}
```

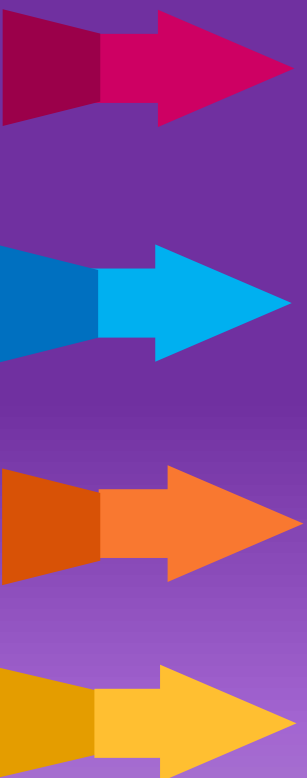
## ESTRUTURAS DE REPETIÇÃO: Repita.. Até: DO - WHILE

Para repetir um conjunto de instruções enquanto uma certa condição for satisfeita, este conjunto é executado pelo menos 1 vez e esta é a diferença para a estrutura Enquanto.. Faça(while) que pode não executar nenhuma vez caso a condição não seja satisfeita.





# ESTRUTURAS DE REPETIÇÃO: Repita.. Até: DO - WHILE

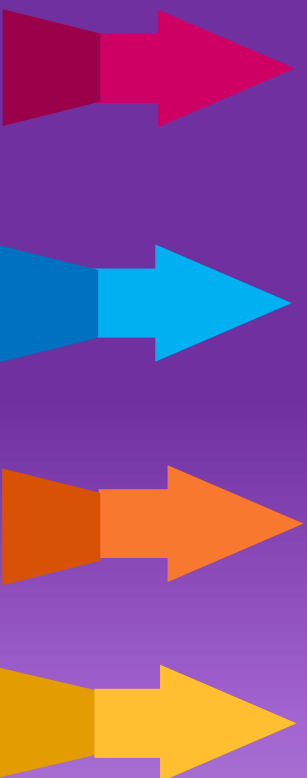


```
do{  
    <comando 1>;  
    <comando 2>;  
} while(CP);
```

CP Condição de parada



# (JAVA) ESTRUTURAS DE REPETIÇÃO: Repita.. Até: DO - WHILE



```
import javax.swing.JOptionPane;
public class Exemplowhile {
    public static void main(String[] args) {
        int i = 0;
        do{
            JOptionPane.showMessageDialog(null,"Valor de i: " + i);
            i++;
        }while(i < 5);
    }
}
```

# VARIÁVEL INDEXADA:

Uma variável indexada corresponde a uma sequência de posições de memória, a qual daremos único Nome, sendo que cada uma destas pode ser acessada através do que conhecemos por índice.

O índice corresponde a um valor numérico (exceto Real).

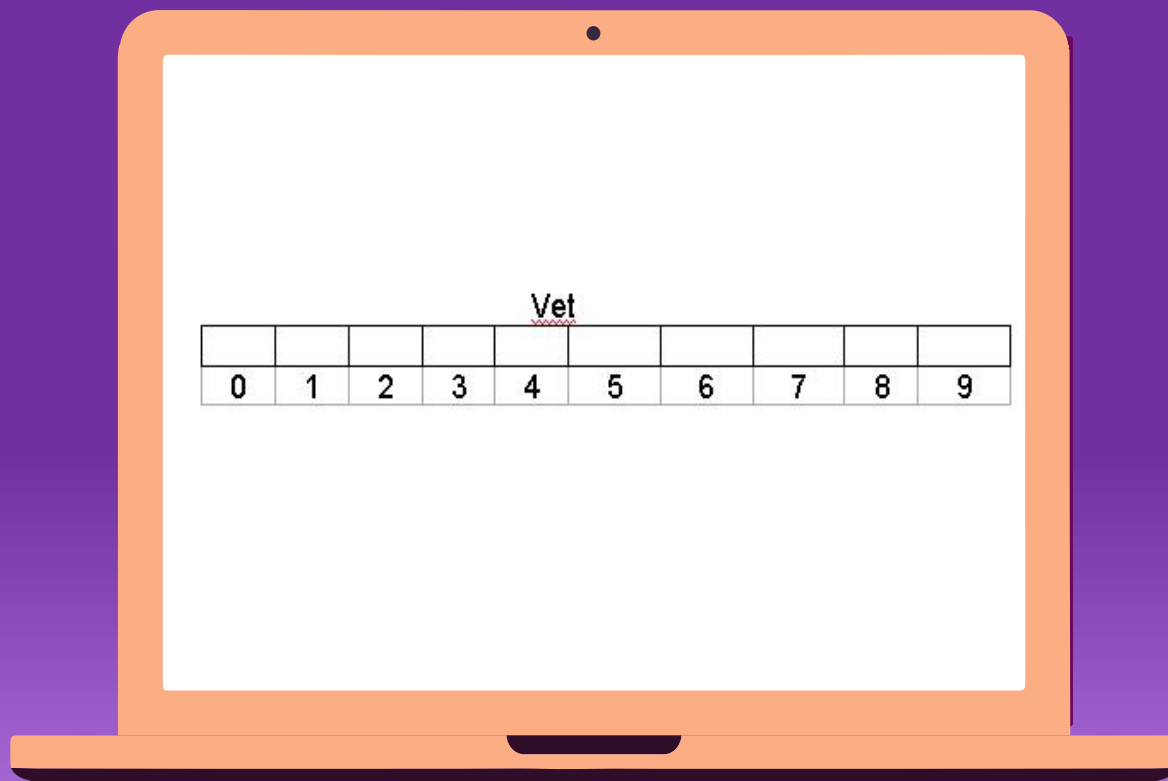
Cada uma das posições de memória de uma variável indexada pode receber valores no decorrer do algoritmo como se fosse uma variável comum, a única diferença reside na Sintaxe de utilização desta variável.

## TIPOS

**Variáveis  
Indexadas  
Unidimensionais  
(Vetor)**

**Variáveis  
Indexadas  
Bidimensionais  
(Matriz)**

# VARIÁVEL INDEXADA UNIDIMENSIONAL (VETOR) :



## EXEMPLO:

Definir uma variável indexada como sendo do tipo REAL, sendo que a mesma deverá corresponder a 10 posições de memória.

```
public class Exemplo {  
    public static void main(String args[ ]) {  
        double vet[ ] = new double[10];  
        <comandos>;  
    }  
}
```



Descrição de imagem:

Mostra quadro com os números de 0 a 9 e lacunas representando espaços na memória.  
1 – 0 1 2 3 4 5 6 7 8 9

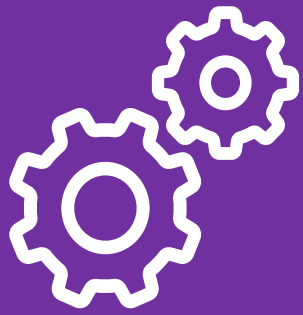
# VARIÁVEL INDEXADA UNIDIMENSIONAL (VETOR)

## EXEMPLO 01

```
public class Atribui {  
    public static void main(String args[]) {  
        String nomes[ ] = new String[20];  
        nomes[0] = "João da Silva";  
        nomes[1] = "Ana";  
    }  
}
```

## EXEMPLO 02

```
public class VetorAtribuir {  
    public static void main ( String args[]) {  
        int X[ ] = new int[20]; // declarando e instanciando o vetor  
        X [0] = 100;  
        X [1] = X [0] + 3;  
    }  
}
```



# VARIÁVEL INDEXADA UNIDIMENSIONAL (VETOR):

## Leitura - Exemplo

```
import javax.swing.*;
public class Exemplo1 {
    public static void main(String args[]) {
        int i;
        String nomes[] = new String[20];
        for(i = 0; i < 20; i++) {
            nomes[i] = JOptionPane.showInputDialog("Digite o nome: ");
        }
    }
}
```



# VARIÁVEL INDEXADA UNIDIMENSIONAL (VETOR):

## Escrita - Exemplo

```
import javax.swing.*;
public class Exemplo3 {
    public static void main(String args[]) {
        int i;
        String nomes[ ] = new String[20];
        nomes[0] = "Unisul";
        nomes[1] = "Aluno";
        nomes[2] = "Sistema";
        for(i = 0; i < 3; i++) {
            JOptionPane.showMessageDialog(null, "o nome na posição "+i+ " é "+ nomes[i] );
        }
    }
}
```



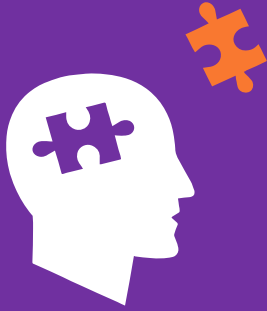
## (JAVA) EXERCÍCIO 19



Solicite ao usuário que escreva uma frase e o número de vezes que a mesma deve ser mostrada. Implemente o algoritmo usando for







## (JAVA) EXERCÍCIO 20



Escreva um algoritmo para mostrar os valores de 1 até 10 usando for

---





## (JAVA) EXERCÍCIO 21



Escreva um algoritmo que mostre todos os números pares entre 33 e 57 usando for

---





## (JAVA) EXERCÍCIO 22

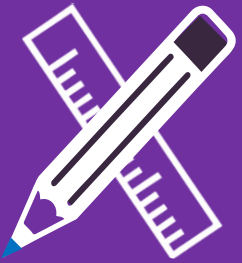


Escreva um algoritmo para calcular e mostrar a média dos números entre 1 e 1000 usando while





## (JAVA) EXERCÍCIO 23



Escreva um algoritmo usando while que solicite ao usuário um número inicial e um número final. Calcule a soma de todos os números dentro da faixa de valor informada INCLUINDO o número inicial e final





## (JAVA) EXERCÍCIO 24

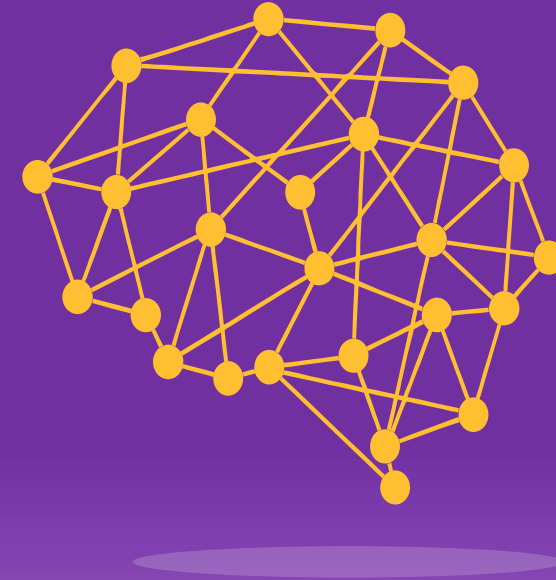


Escreva um algoritmo que mostre todos os números pares entre 13 e 23 usando do..while.



# Busca Ativa!

- 1 Procure e assista vídeos na internet que demonstrem o funcionamento de vetores.
- 2 Implemente e execute TODOS os exemplos da aula de hoje.
- 3 Crie um exemplo de uso de vetores.



**FIM**

