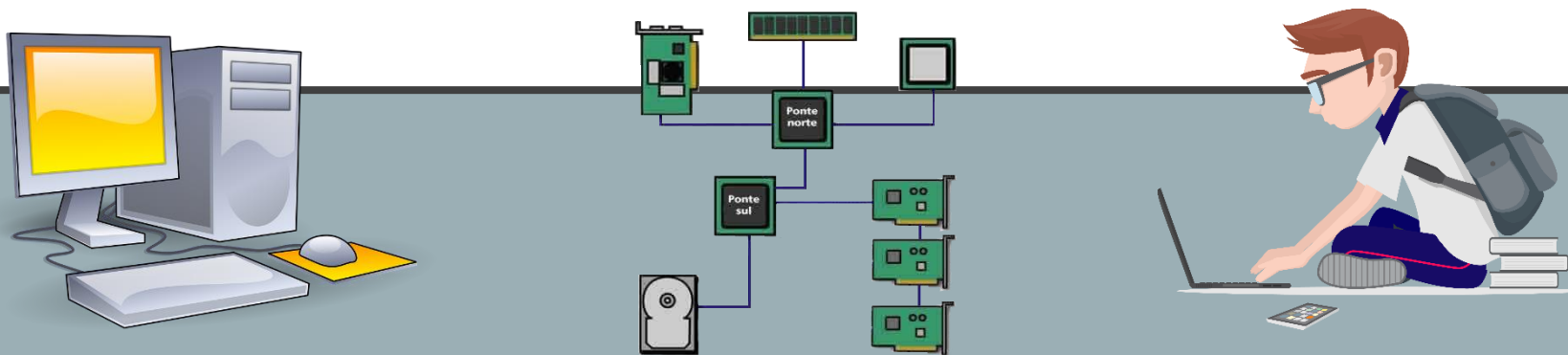


DESENVOLVIMENTO DE SISTEMAS

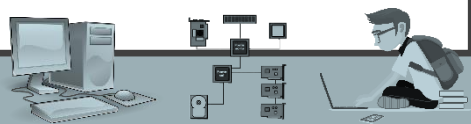


Prof. Daniel Tadeu Petinice



daniel.petinice@sp.senai.br

LÓGICA DE PROGRAMAÇÃO



Plano de Aula

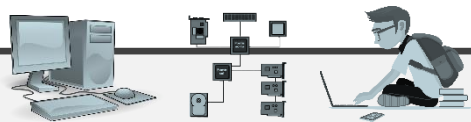
- Conteúdo:**
- Comandos sobre a Linguagem Portugol;
 - Estrutura de Controle Comando de decisão;
 - Laço de Repetição Comando de repetição;

Início:

As informações deste conteúdo visam compreender conceitos de Lógica de Programação.



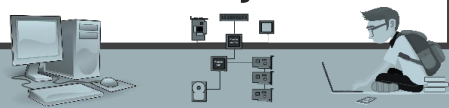
LÓGICA DE PROGRAMAÇÃO



Objetivo

Proporcionar capacidades básicas e socioemocionais que permitem desenvolver algoritmos, por meio de lógica de programação e versionamento, para resolução de problemas.

LÓGICA DE PROGRAMAÇÃO



Pseudocódigo Portugol



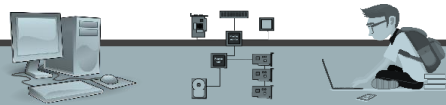
Portugol
WebStudio

Pseudocódigo em lógica de programação é uma maneira de descrever os passos de um algoritmo usando uma mistura de linguagem natural e uma estrutura que se assemelha a uma linguagem de programação, mas de forma mais simplificada e menos rígida.

Ele não é executável por um computador, mas serve como um meio-termo entre a descrição verbal de um algoritmo e o código fonte real escrito em uma linguagem de programação específica.

O Portugol é uma representação que se assemelha bastante com a linguagem C, porém é escrito em português. A ideia é facilitar a construção e a leitura dos algoritmos usando uma linguagem mais fácil aos alunos.

LÓGICA DE PROGRAMAÇÃO



Pseudocódigo Portugol

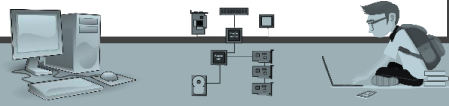


Vamos acessar o site:

<https://dgadelha.github.io/Portugol-Webstudio/>



LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol

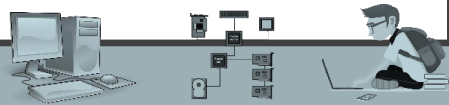


Portugol
WebStudio

Exemplo:

```
1  programa {  
2      funcao inicio() {  
3  
4      }  
5  }
```

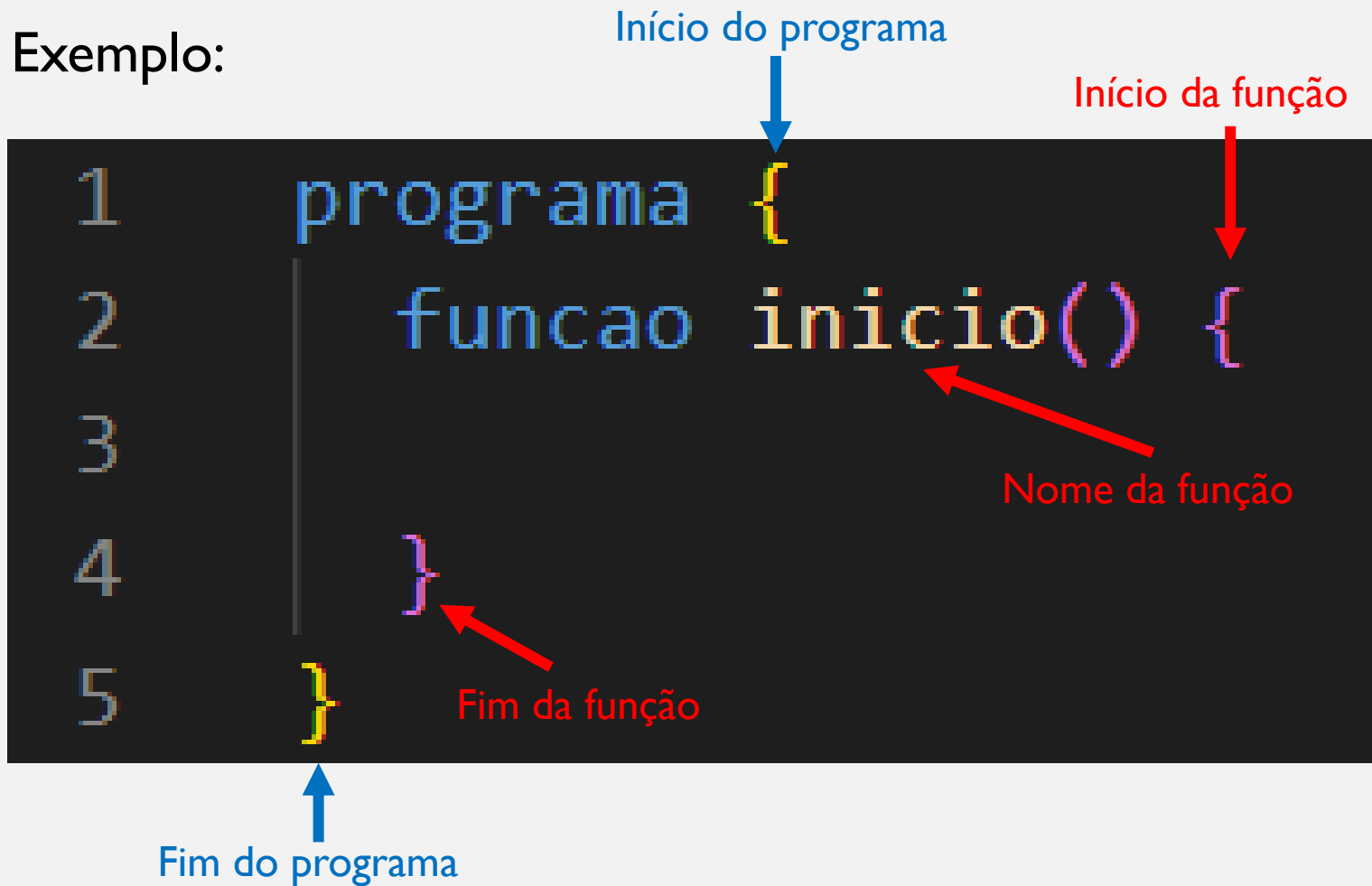
LÓGICA DE PROGRAMAÇÃO



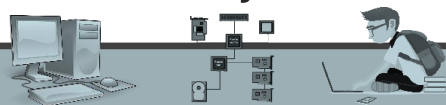
Comandos sobre Portugol



Exemplo:



LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol



Portugol
WebStudio

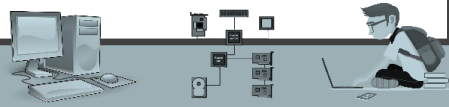
Contexto:

Vamos criar um programa que solicitará ao usuário a inserção de **dois números**.

Esses números serão **armazenados** em diferentes variáveis pré-definidas pelo programador.

Após a inserção será calculado a operação e exibido ao usuário o resultado.

LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol

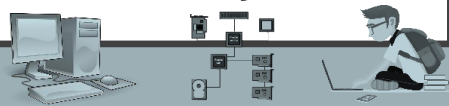


Portugol
WebStudio

Exemplo:

```
1  programa {
2      funcao inicio() {
3          inteiro numero_1
4          inteiro numero_2
5          inteiro resultado
6
7          escreva("Digite o primeiro numero: ")
8          leia(numero_1)
9
10         escreva("Digite o segundo numero: ")
11         leia(numero_2)
12
13         resultado = numero_1 + numero_2
14
15         escreva("A soma dos números é: ", resultado)
16     }
17 }
```

LÓGICA DE PROGRAMAÇÃO



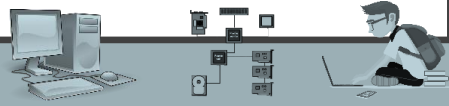
Comandos sobre Portugol



Quais são os tipos de dados que o computador pode armazenar?

- Tipo **Cadeia**
- Tipo **Caracter**
- Tipo **Inteiro**
- Tipo **Real**
- Tipo **Lógico**

LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol



Quais são os tipos de dados que o computador pode armazenar?

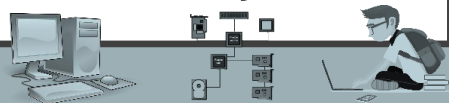
Portugol

- Tipo **Cadeia**
- Tipo **Caracter**
- Tipo **Inteiro**
- Tipo **Real**
- Tipo **Lógico**

Java

String
char
int **ou** **long**
float **ou** **double**
boolean

LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol



Portugol
WebStudio

Tipo **Cadeia**

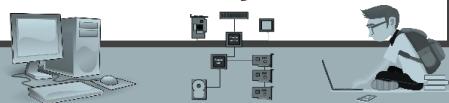
Em algumas situações precisa-se armazenar em uma variável, um texto ou **uma quantidade grande de caracteres**. Para armazenar este tipo de conteúdo, utiliza-se uma variável do tipo **cadeia**.

Cadeia é uma sequência ordenada de caracteres (símbolos) escolhidos a partir de um conjunto pré-determinado.

```
cadeia curso
```

```
curso = "Lógica de Programação"
```

LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol



Portugol
WebStudio

Tipo **Caracter**

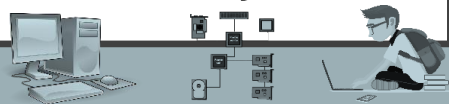
A variável do tipo caracter é aquela que contém uma informação composta de apenas **UM** **carácter** **alfanumérico** ou **especial**. Exemplos de caracteres são letras, números, pontuações e etc.

```
caracter letra
```

```
letra = 'L'
```

Observação: quando utilizamos **caracter**, a informação tem que ser inserida entre **aspas simples** e não **dupla**.

LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol



Portugol
WebStudio

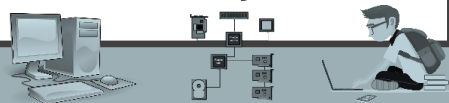
Tipo **Inteiro**

A variável do tipo inteiro armazena **somente números inteiros** sem a necessidade de colocar entre aspas a informação.

```
inteiro valor
```

```
valor = 10
```

LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol



Portugol
WebStudio

Tipo Real

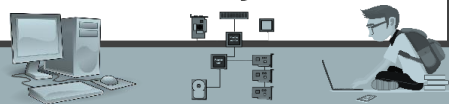
A variável tipo real refere-se a valores que podemos armazenar **números com casas decimais**.

Quando queremos trabalhar com números que não são inteiros, como 3.14 ou 0.001, usamos o tipo real.

```
real pi
```

```
pi = 3.1415926535
```

LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol



Portugol
WebStudio

Tipo Lógico

Uma variável do tipo logico é aquela que contém um tipo de dado, usado em operações lógicas, que possui somente dois valores, que são consideradas pelo Portugol como **verdadeiro** e **falso**.

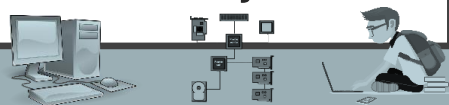
```
logico teste
```

```
teste = verdadeiro
```

```
// ou
```

```
teste = falso
```


LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol



Portugol
WebStudio

Exibir informação na tela

Comando `escreva()` é utilizado e Portugol para imprimir algo na tela.

Exemplo:

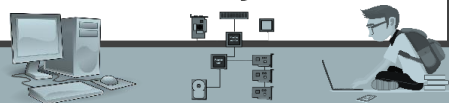
Código:

```
escreva("Curso: Lógica de Programação")
```

Exibição código:

```
Curso: Lógica de Programação  
Programa finalizado. Tempo de execução: 42 ms
```

LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol



Portugol
WebStudio

Exibir informação na tela

Caso precise fazer uma quebra de linha, podemos utilizar a expressão '\n'.

Exemplo:

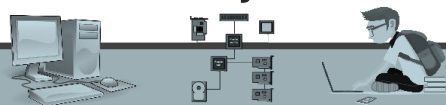
Código:

```
escreva("Curso:\nLógica de Programação")
```

Exibição código:

```
Curso:  
Lógica de Programação  
Programa finalizado. Tempo de execução: 53 ms
```

LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol



Armazenar informação em variáveis

O Comando `leia()` é utilizado quando se deseja obter informações do teclado do computador, ou seja, é um comando de entrada de dados. Esse comando aguarda um valor a ser digitado e o atribui diretamente na variável.

Exemplo:

Código:

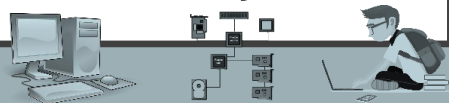
```
cadeia nome  
leia(nome)
```

Exibição código:

```
Fulano
```

```
Programa finalizado. Tempo de execução: 3175 ms
```

LÓGICA DE PROGRAMAÇÃO



Comandos sobre Portugol



Portugol
WebStudio

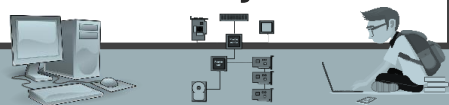
Armazenar informação e imprimir na tela

```
cadeia nome
inteiro idade
escreva("Digite seu nome: ")
leia(nome)
escreva("Digite sua idade: ")
leia(idade)

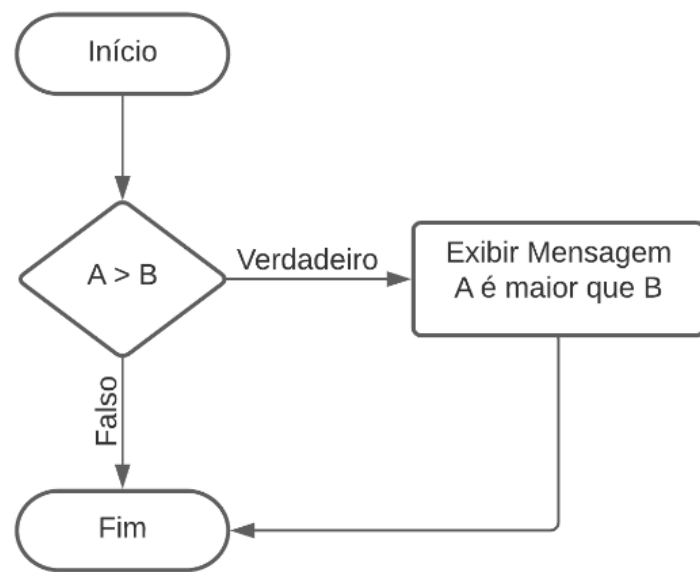
escreva(nome, " possui a idade de ", idade, " anos")
```

```
Digite seu nome: Fulano
Digite sua idade: 20
Fulano possui a idade de 20 anos
```

LÓGICA DE PROGRAMAÇÃO



Estrutura de Controle Comando de decisão



Dentro da programação, um dos pilares é a capacidade do código de fazer escolhas com base em informações.

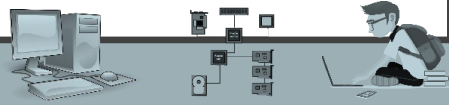
Para viabilizar essas escolhas, utilizamos **estruturas de controle ou comando de decisão**.

Dentre elas, o comando "**SE**" (Portugol), conhecido como "**IF**" em outras linguagens de programação.

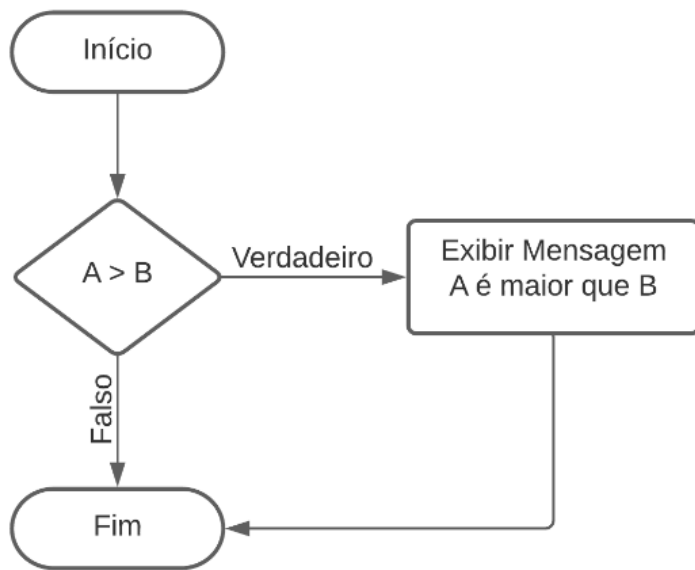
Definição: A estrutura de controle "SE" avalia uma condição **(expressão lógica)**. Se essa condição for verdadeira, o código dentro dessa estrutura é executado.

Caso contrário, pode-se optar por **ignorá-lo** ou **executar um bloco de código alternativo**.

LÓGICA DE PROGRAMAÇÃO



Estrutura de Controle Comando de decisão



Estrutura de Controle - Simples

Verdadeiro Falso

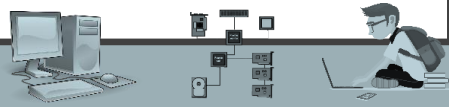
ou

```
se ( expressão lógica ) {
```

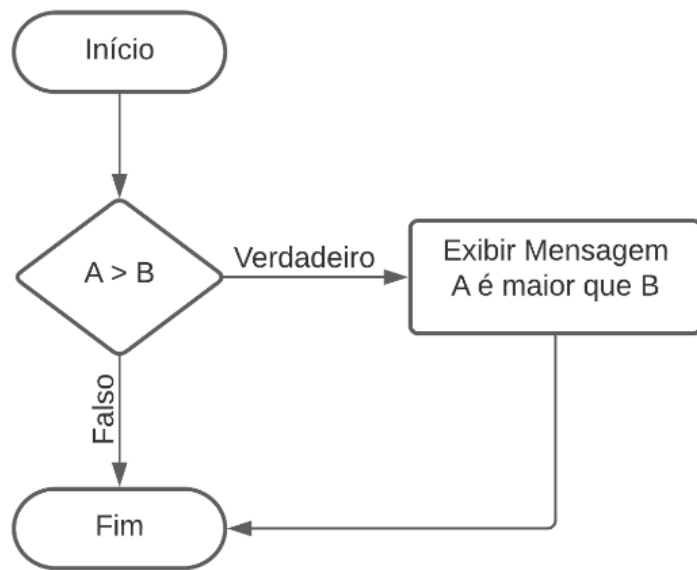
```
    BLOCO DE CÓDIGO
```

```
}
```

LÓGICA DE PROGRAMAÇÃO



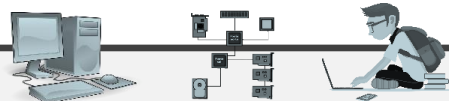
Estrutura de Controle Comando de decisão



Estrutura de Controle - Simples

```
1  programa {
2      funcao inicio() {
3          inteiro valor
4
5          escreva("Informe um valor!")
6          leia(valor)
7
8          se(valor > 10){
9              escreva("o valor informado é maior que 10")
10         }
11     }
12 }
```

LÓGICA DE PROGRAMAÇÃO

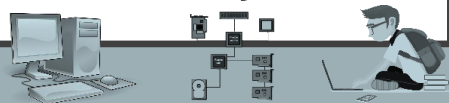


Estrutura de Controle Comando de decisão

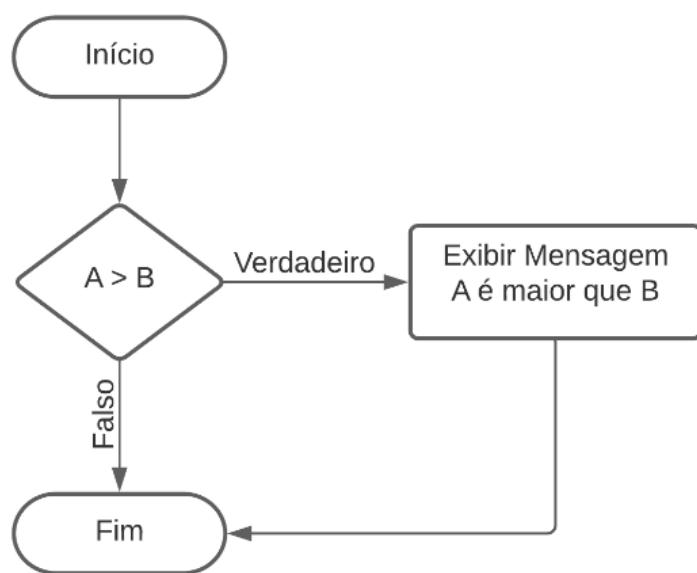
```
programa {  
  funcao inicio() {  
    inteiro valor  
  
    escreva("Informe um valor!")  
    leia(valor)  
  
    se(valor > 10){  
      escreva("o valor informado é maior que 10")  
    }  
  }  
}
```

```
Var  
valor : inteiro  
  
Inicio  
escreval("Informe um valor!")  
leia(valor)  
se (valor > 10) entao  
    escreval("O valor informado é maior que 10")  
fimse  
  
Fimalgoritmo
```


LÓGICA DE PROGRAMAÇÃO



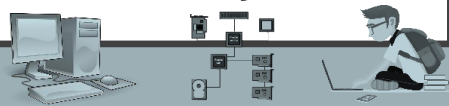
Estrutura de Controle Comando de decisão



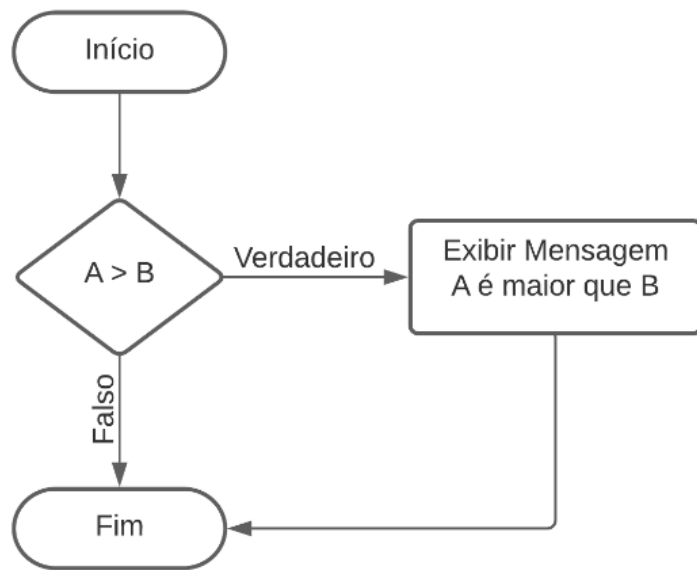
Estrutura de Controle - Composta

```
se ( expressão lógica ) {  
    BLOCO DE CÓDIGO se for VERDADEIRO  
}  
senao {  
    BLOCO DE CÓDIGO se for FALSO  
}
```

LÓGICA DE PROGRAMAÇÃO



Estrutura de Controle Comando de decisão



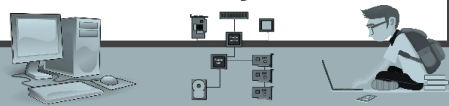
Estrutura de Controle - Composta

```
escreva("Informe um valor!")
leia(valor)

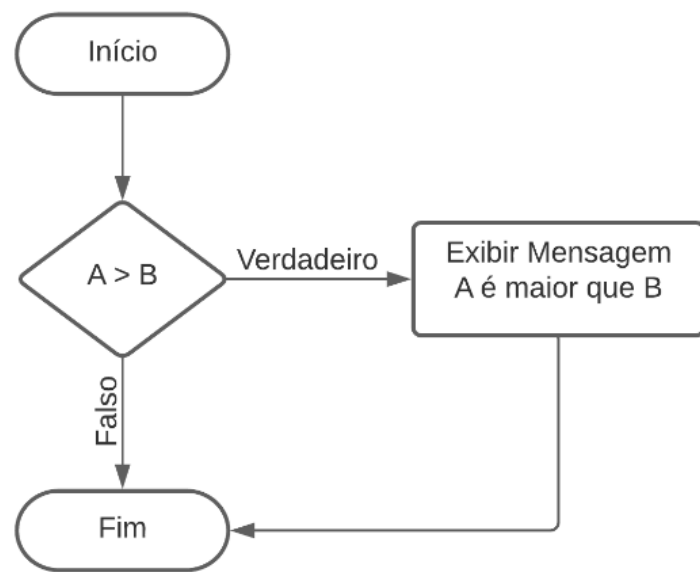
se(valor > 10){
    escreva("o valor informado é maior que 10")
} senao {
    escreva ("o valor informado é menor ou igual a 10")
}
```

Podemos digitar assim

LÓGICA DE PROGRAMAÇÃO



Estrutura de Controle Comando de decisão



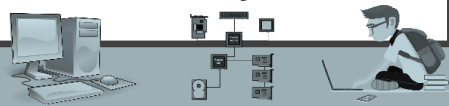
Estrutura de Controle - Composta

```
escreva("Informe um valor!")
leia(valor)

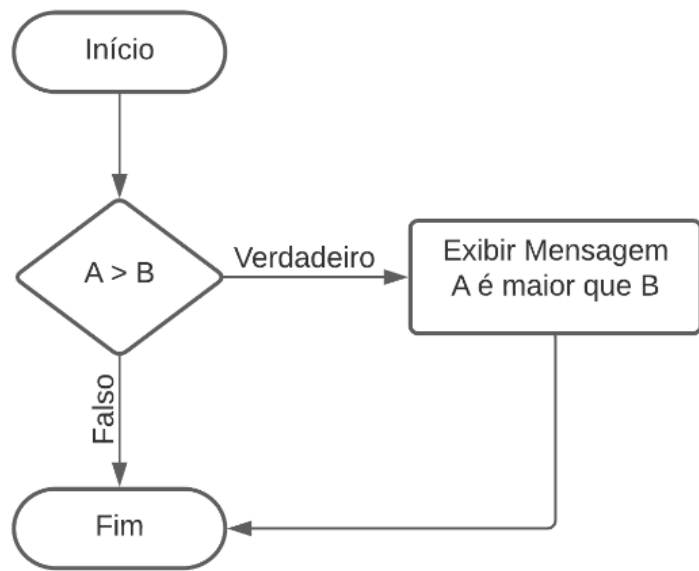
se(valor > 10){
    escreva("o valor informado é maior que 10")
}
senao {
    escreva ("o valor informado é menor ou igual a 10")
}
```

Ou assim...

LÓGICA DE PROGRAMAÇÃO



Estrutura de Controle Comando de decisão

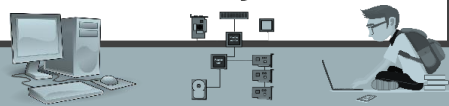


Estrutura de Controle - Composta

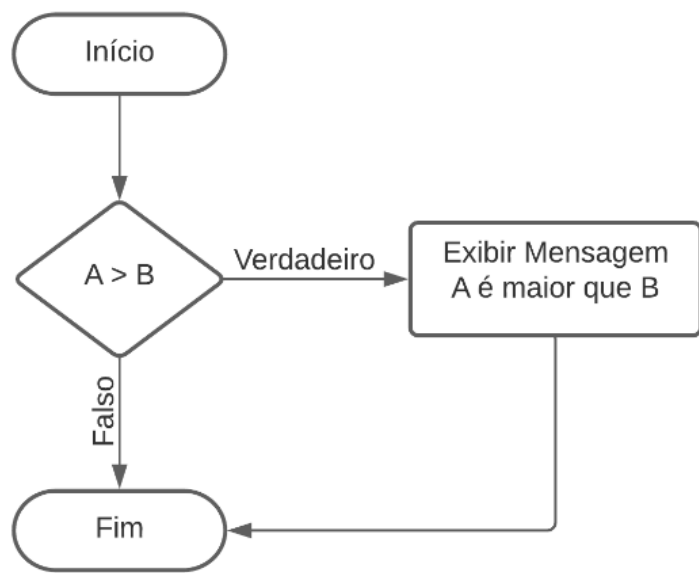
Quando você precisa verificar se uma condição é verdadeira, e se não for, precise verificar se outra condição é verdadeira uma das formas de se fazer esta verificação é utilizando do **se ... senao se;**

```
se ( expressão lógica ) {  
    BLOCO DE CÓDIGO se for VERDADEIRO  
} senao se ( expressão lógica ) {  
    BLOCO DE CÓDIGO se for FALSO  
}
```

LÓGICA DE PROGRAMAÇÃO



Estrutura de Controle Comando de decisão

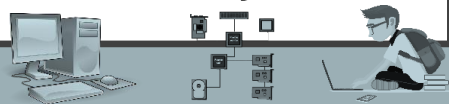


Estrutura de Controle - Composta

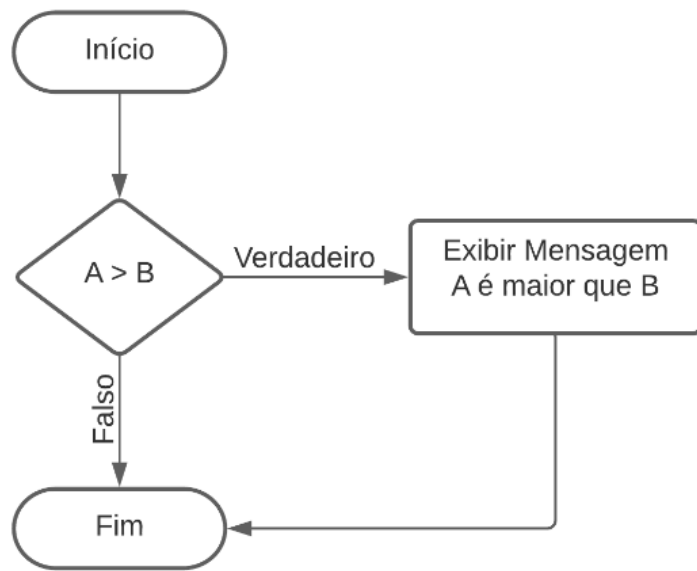
Imagine que você precise verificar a nota da prova de um aluno e dizer se ele foi **muito bem**, **razoável** ou **precisa se empenhar mais** em uma prova. Como fazer isso?

```
se ( expressão lógica ) {  
    BLOCO DE CÓDIGO se for VERDADEIRO  
}  
senao se ( expressão lógica ) {  
    BLOCO DE CÓDIGO  
}  
senao {  
    BLOCO DE CÓDIGO se for Falso  
}
```

LÓGICA DE PROGRAMAÇÃO



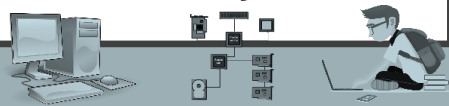
Estrutura de Controle Comando de decisão



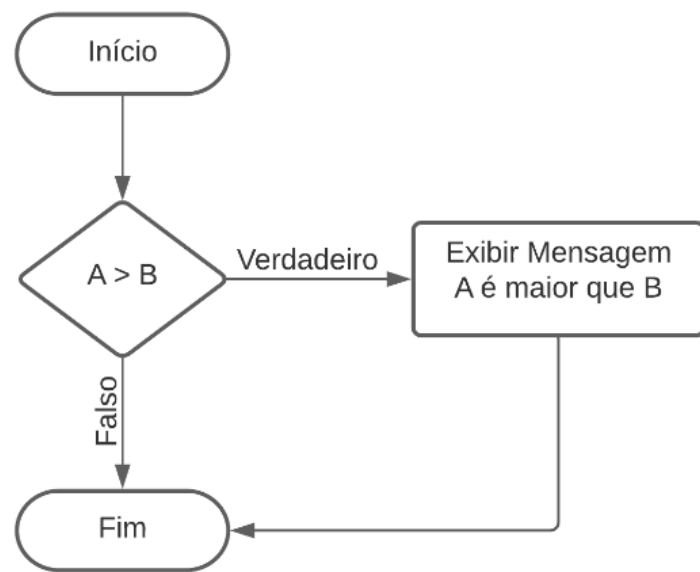
Estrutura de Controle - Composta

```
1  programa {
2      funcao inicio() {
3          real nota
4
5          escreva("Digite a nota do aluno: ")
6          leia(nota)
7
8          se (nota >= 7){
9              escreva("Você foi muito bem na prova!")
10             } senao se (nota > 3 e nota < 7) {
11                 escreva("Sua nota foi razoável")
12             } senao {
13                 escreva("Você precisa se empenhar mais em uma prova")
14             }
15
16         }
17     }
```

LÓGICA DE PROGRAMAÇÃO



Estrutura de Controle Comando de decisão



Praticando

“Vamos imaginar a criação de um programa para calcular a média de notas de um aluno.

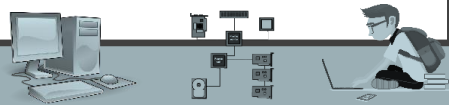
O programa solicitará que o professor digite duas notas. Em seguida, o programa receberá essas notas, calculará a média e informará ao professor o resultado.”

Agora, o professor precisa saber se o aluno foi **aprovado** ou **reprovado** na matéria.

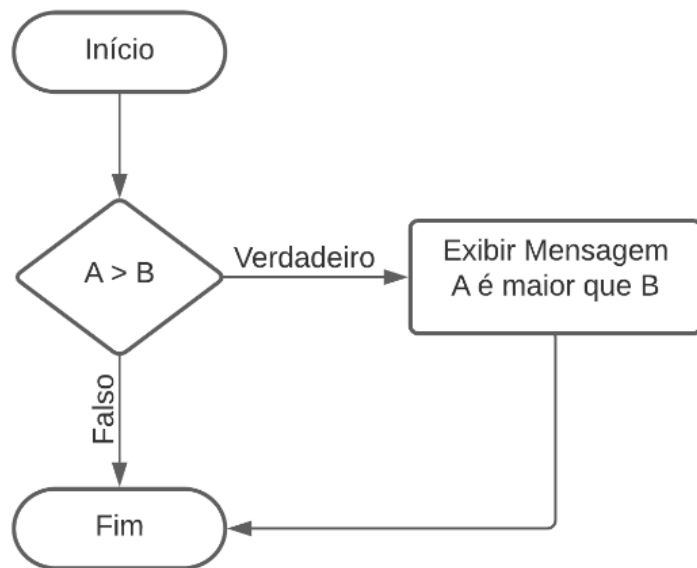
Para isso, usaremos como critério o valor da média **7**.

Se os alunos obtiverem uma média igual ou superior a 7, serão **considerados aprovados**; caso contrário, serão **considerados reprovados**.

LÓGICA DE PROGRAMAÇÃO



Estrutura de Controle Comando de decisão

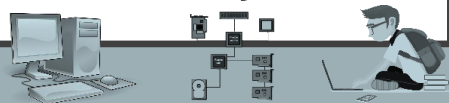


Vamos acessar o site:

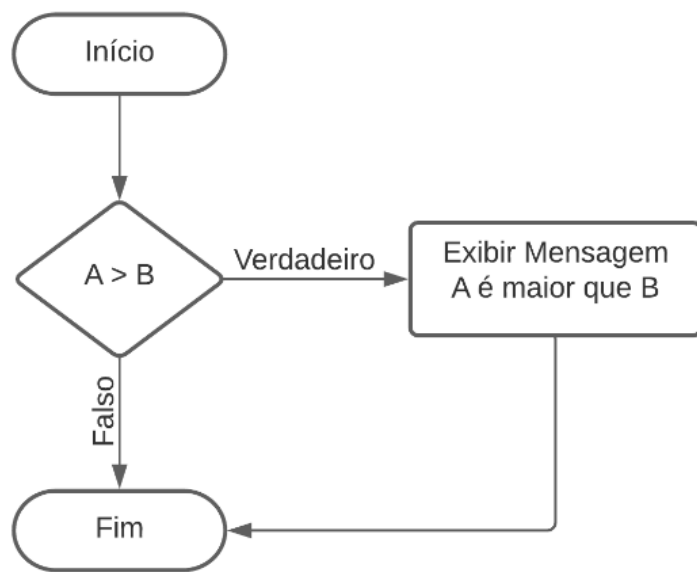
<https://dgadelha.github.io/Portugol-Webstudio/>



LÓGICA DE PROGRAMAÇÃO

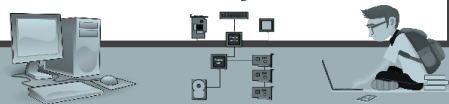


Estrutura de Controle Comando de decisão



```
1  programa {
2      funcao inicio() {
3          real nota_1, nota_2, resultado
4
5          escreva("Digite a primeira nota: ")
6          leia(nota_1)
7
8          escreva("Digite a segunda nota: ")
9          leia(nota_2)
10
11         resultado = (nota_1 + nota_2) / 2
12
13         escreva("A média é: ", resultado)
14         escreva("\n\n")
15
16         se (resultado >= 7){
17             escreva("Parabéns, você está aprovado.")
18         } senao {
19             escreva("Desculpe, você está reprovado.")
20         }
21     }
22 }
23
```

LÓGICA DE PROGRAMAÇÃO



Laço de Repetição Comando de repetição

Em programação, frequentemente precisamos executar uma **ação várias vezes**.

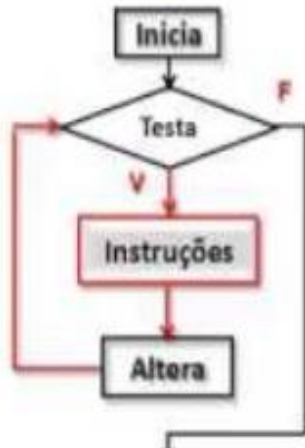
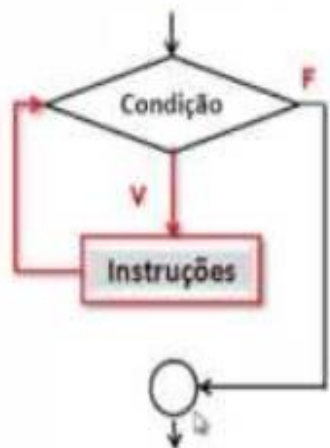
Em vez de repetir o mesmo código múltiplas vezes, utilizamos uma ferramenta chamada "**laço de repetição**".

Este mecanismo nos permite repetir um bloco de código enquanto uma condição for verdadeira ou por um número determinado de vezes.

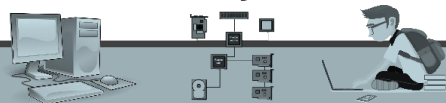
Desse modo, economizamos tempo e evitamos erros, tornando o código mais limpo e eficiente.

Pré-condição
Enquanto

Repetição Contada
Para



LÓGICA DE PROGRAMAÇÃO



Laço de Repetição Comando de repetição

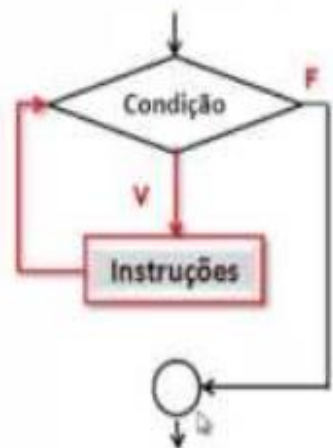
Laço de Repetição - Enquanto

O comando "**enquanto**" serve para avaliar um teste lógico.

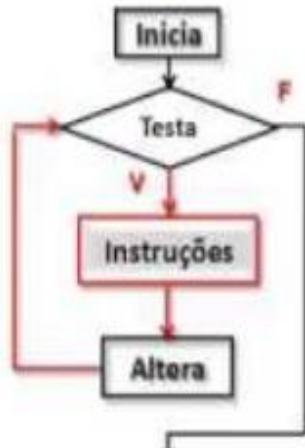
Ele tem como principal função **repetir um conjunto de comandos** enquanto a condição se mantenha verdadeira.

Em outras linguagens é utilizado a expressão "**While**", exemplo da linguagem Java.

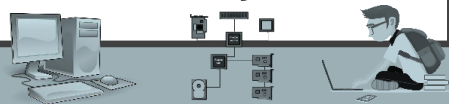
Pré-condição
Enquanto



Repetição Contada
Para



LÓGICA DE PROGRAMAÇÃO



Laço de Repetição Comando de repetição

Laço de Repetição - Enquanto

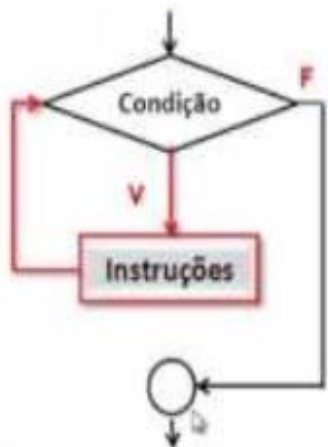
Verdadeiro Falso
 ou

enquanto (condição lógica) {

BLOCO DE CÓDIGO

}

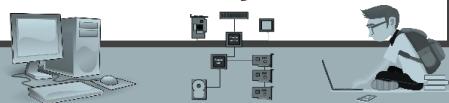
Pré-condição
Enquanto



Repetição Contada
Para

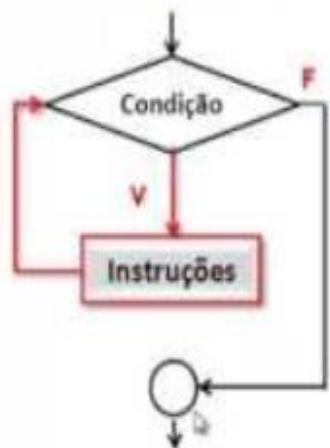


LÓGICA DE PROGRAMAÇÃO

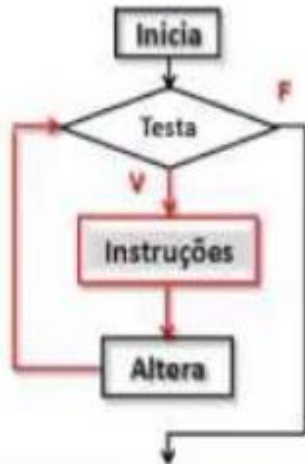


Laço de Repetição Comando de repetição

Pré-condição
Enquanto



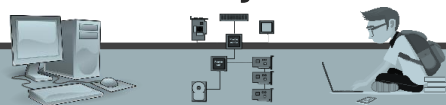
Repetição Contada
Para



Laço de Repetição - Enquanto

```
1  programa {
2      funcao inicio() {
3
4          caracter letra
5
6          enquanto (letra != 's'){
7              escreva("olá, seja bem-vindo...")
8              escreva("\n\n")
9              // Bloco de código
10
11             escreva("Deseja sair do Sistema SIM (s) ou NÃO (n)")
12             leia(letra)
13             limpa()
14         }
15         escreva("Programa finalizado com sucesso")
16     }
17 }
```

LÓGICA DE PROGRAMAÇÃO



Laço de Repetição – Repita ou Para

Na programação, às vezes precisamos repetir ações muitas vezes.

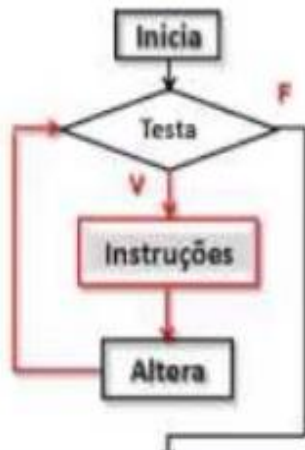
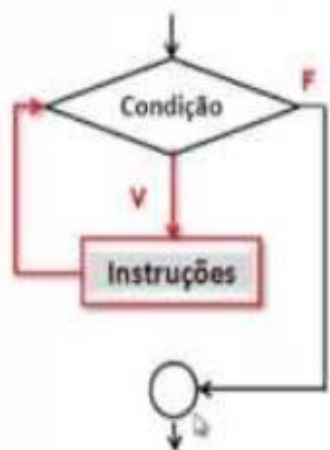
O laço "**para**" é uma ferramenta que permite fazer isso de forma organizada.

Ele possui um contador: **você define onde começa, onde termina** e o **quanto deve adicionar a cada ciclo**.

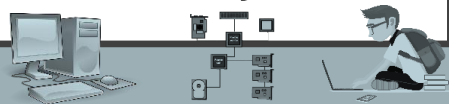
Exemplo: para repetir algo 10 vezes, o laço pode começar em 1 e adicionar 1 a cada vez até chegar a 10.

Pré-condição
Enquanto

Repetição Contada
Para



LÓGICA DE PROGRAMAÇÃO



Laço de Repetição – Repita ou Para

Onde o contador começa

Até onde o contador deve ir

Controle do contador

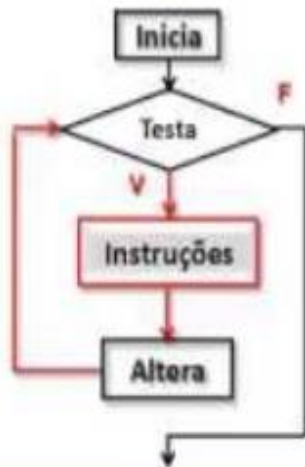
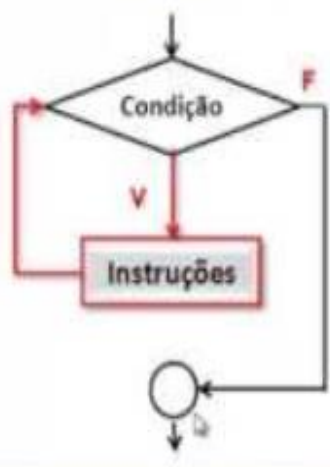
```
para ( parâmetros ) {
```

```
BLOCO DE CÓDIGO
```

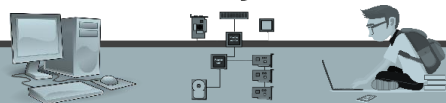
```
}
```

Pré-condição
Enquanto

Repetição Contada
Para



LÓGICA DE PROGRAMAÇÃO



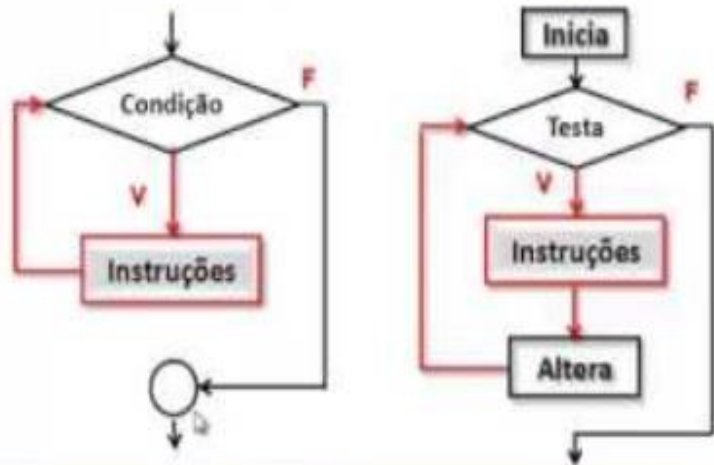
Laço de Repetição – Repita ou Para

Laço de Repetição Comando de repetição

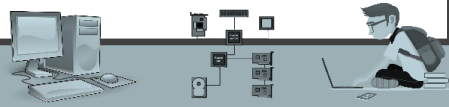
```
1  √ programa {  
2  √   funcao inicio() {  
3        
4      inteiro numero_tabuada  
5        
6  √   para (inteiro c = 1; c <= 10; c++) {  
7       numero_tabuada = c * 3  
8       escreva ("3 x ", c, " = ", numero_tabuada, "\n")  
9       }  
10  }  
11  }  
12
```

Pré-condição
Enquanto

Repetição Contada
Para



LÓGICA DE PROGRAMAÇÃO



Links

```
if( (s0 == false) && (s2 == false)
else
{
    if( s1 == false ) PORTC = 0x00;
    if( s3 == false ) PORTC = 0x00;
    if( s2 == false ) PORTC = 0x00;
}

PORTB = 0x00;
}

int main()
{
    // BIT      76543210
    DDRD      = 0b00000000;
    DDRB      = 0b00100000;
    DDRC      = 0b00000011;
    PORTC     = 0b00000000;
```

Links para estudos

Portugol:

<https://dgadelha.github.io/Portugol-Webstudio/>

Observação: Clique no botão “**Ajuda**” para mais detalhes da documentação.

REFERÊNCIAS

ASCENCIQ Ana Fernanda Gomes; ARAÚJO Graziela Santos de. Estruturas de dados: algoritmos, análise da complexidade e implementações em JAVA e C/C++. São Paulo: Pearson Prentice Hall, 2010.432 p.

Alves, William Pereira. Lógica de programação de computadores. São Paulo: Érica, 2012.

Manzano, J.A.N.G. e Oliveira, J. F. Algoritmos – Lógica ara desenvolvimento de programação de computadores. 26ª ed. São Paulo: Érica, 2012. 328p.

Importante:

Os conteúdos disponibilizados são específicos para este curso/turma, a divulgação ou reprodução do material para outras pessoas/organização não é autorizada.