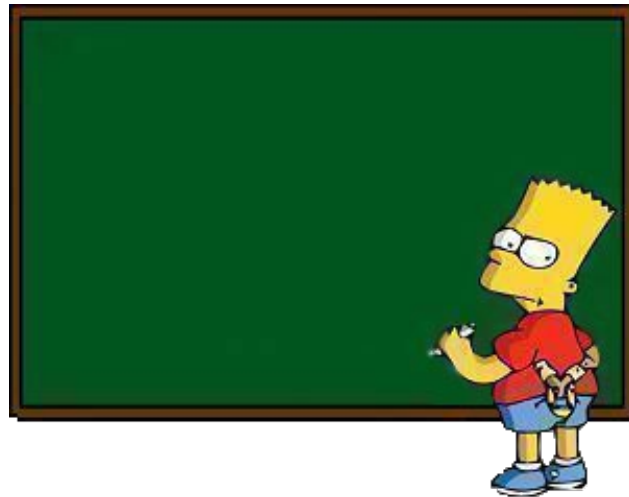

Estruturas de Repetição

— Aula 4 —

Repetir Instruções

- Vamos começar com um estudo de caso simples: Imagine que Bart Simpson tenha aprontado na escola (como sempre). Sua professora o castigou mandando-o que escreva 100 vezes a frase “Serei um bom aluno a partir de agora”; ele se deu mal nessa.
- Bom, nosso objetivo seria tentar ajudar o Bart. Podemos criar um programa para imprimir essa frase 100 vezes? Parece ser algo simples.
- Como você faria? (pense um pouco)

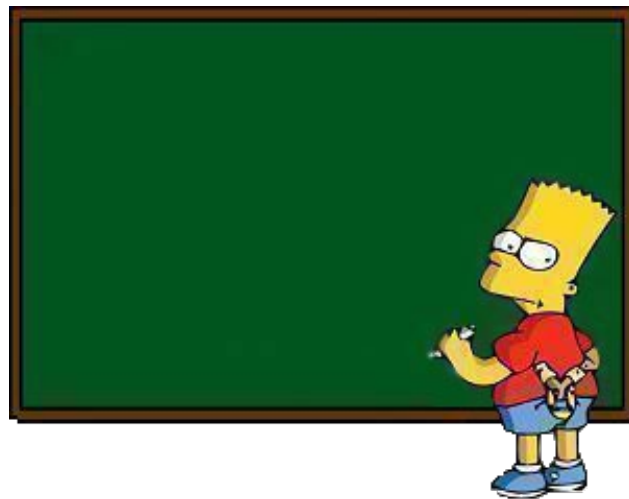


Repetir Instruções

- De início, uma solução poderia ser algo do tipo:

```
escreva("Serei um bom aluno a partir de agora") // #1  
escreva("Serei um bom aluno a partir de agora") // #2  
escreva("Serei um bom aluno a partir de agora") // #3  
//... e repetimos isso, até chegar em 100
```

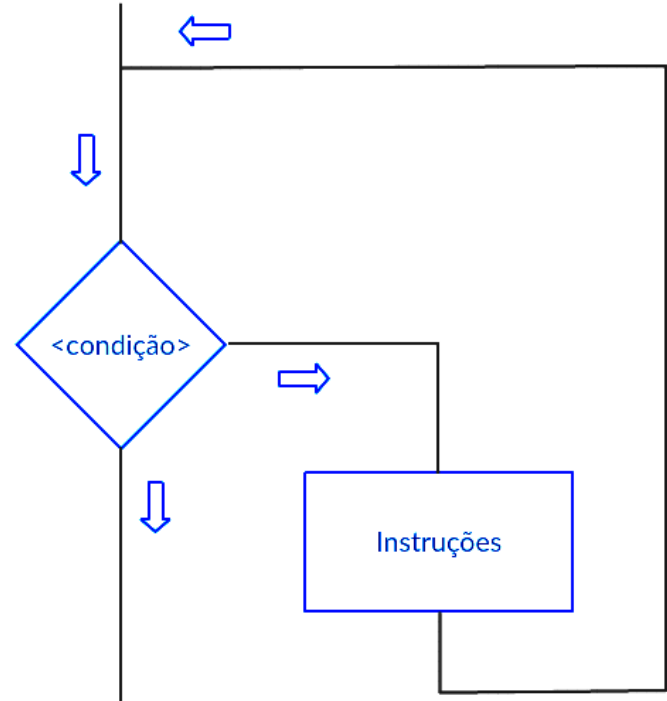
- Claramente, isso seria um pouco repetitivo e daria muito trabalho pra gente.
- Além disso, e se o Bart aprontasse de novo e a professora o castigasse a escrever 200 vezes ao invés de 100? O que faríamos? Escreveríamos linhas e linhas do comando `escreva` toda vez que ele aprontasse?



Repetir Instruções

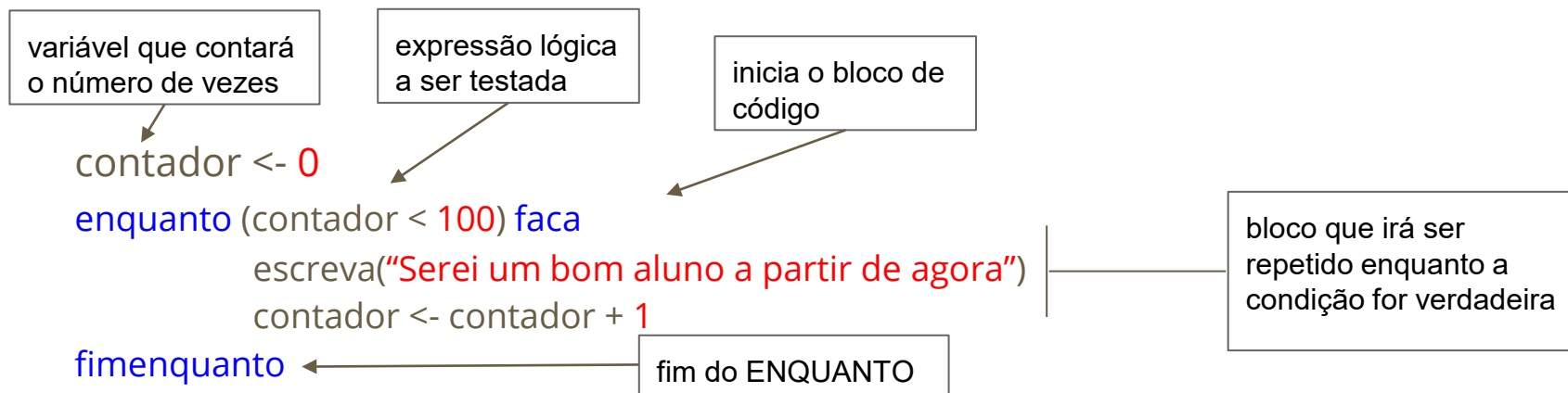
- Será que seria possível fazer algo para que o comando **escreva** seja repetido várias vezes? No caso, enquanto o programa não imprimisse a frase “Serei um bom aluno a partir de agora” 100 vezes, o comando **escreva** continuaria a ser executado.
- A estrutura de repetição **ENQUANTO** serve, para fazer isso. Ela define que, enquanto uma certa condição (que deve ser uma expressão lógica) for verdadeira, um determinado conjunto de instruções deve continuar a ser executado.
- Por exemplo:

```
// Enquanto (Bart não terminar de escrever) faça ele  
//          escrever          até          terminar  
enquanto          (<condicao>)          faca  
    // Instruções
```



Estrutura do ENQUANTO - 1

- Seu funcionamento é tão simples quanto o SE-ENTÃO, só que as instruções dentro do seu bloco de código serão repetidas várias vezes, formando o que chamamos de “loop” (laço) enquanto a expressão testada continuar resultando em **verdadeiro**.



Estrutura do ENQUANTO - 2

- Então, para entender como funciona o exemplo anterior, vamos revisar cada instrução detalhadamente:

contador <- 0

- No caso, subentendemos que uma variável chamada de contador, que será do tipo inteiro, foi declarada e nessa linha atribuímos o valor 0 para ela.
- Usamos essa variável para realizar a contagem da quantidade de repetições do ENQUANTO que já aconteceram. Por isso ela começa com o valor 0 (pois neste momento não houve repetição nenhuma ainda) e será aumentada em uma unidade a cada repetição. Já que queremos que o bloco de código seja repetido 100 vezes; quando essa variável chegar a 100, as repetições iram parar.

Estrutura do ENQUANTO - 3

- Na próxima linha, temos:

`enquanto (contador < 100) faca`

- Aqui, verificamos se a variável contador é menor que 100, como, no começo, ela é igual a 0 (zero), isso resulta em **verdadeiro**, o que significa que o bloco de código do **ENQUANTO** será executado.
- Em outras palavras seria:
Enquanto o valor da variável contador for menor que 100 faça alguma coisa.
- Com isso, podemos observar que, assim como o **SE-ENTAO**, o **ENQUANTO** precisa de um valor lógico **verdadeiro** para executar o seu bloco de instruções.

Estrutura do ENQUANTO - 4

- Dentro do bloco do ENQUANTO, temos:

```
escreva("Serei um bom aluno a partir de agora")  
contador <- contador + 1
```

- Todas as instruções contidas dentro do bloco do ENQUANTO serão repetidas, enquanto a condição for verdadeira.
- Primeiro, colocamos para imprimir a mensagem na tela.
- Na segunda linha, pegamos a variável contador, cujo o valor é, inicialmente, igual a 0 (zero), e atribuímos o valor atual dela mesmo (zero) mais um. Isso aumenta o valor do contador em uma unidade, ou seja, a variável contador para a ser 1 ($0+1=1$).
- Na segunda vez que passar o loop de repetição, será impresso novamente a mensagem e será feita novamente o incremento do contador. Neste caso, o contador passará a ter o valor 2 ($1+1=2$).
- Em todas as próximas vezes, será escrita a mensagem e incrementada a variável contador (3, 4, 5 ... 100), até que contador atinja o valor 100, encerrando a repetição.

Estrutura do ENQUANTO

- A última linha é:

FIMENQUANTO

O FIM-ENQUANTO marca o fim do bloco do ENQUANTO. Ele é obrigatório. Não se esqueça dele!

- Chegando nessa linha, o loop volta para a verificação do início:

ENQUANTO (contador < 100) FAÇA

- Nesse momento, a variável contador com seu valor incrementado é verificada mais uma vez. Se a condição ainda for verdadeira (se contador for menor que 100) o bloco de código é executado novamente, senão (contador não é menor que 100) a repetição acaba.
- Perceba que a cada vez que o loop rodar é acrescentado +1 ao contador, ao final é verificado se esse contador chegou a 100, se não o loop continua.
- O loop só vai parar quando o contador chegar a 100. Neste caso, quando a condição for testada, ela não será mais verdadeira e o bloco de instruções do ENQUANTO não será mais executado.

Exemplo Prático

- A ideia é simples: criar um programa que conte até 10. Isso é muito simples quando sabemos utilizar corretamente a estrutura do ENQUANTO.

```
algoritmo "Contar ate 10"  
  
var  
    contador : inteiro  
inicio  
  
    // Contador iniciando em 1  
    contador <- 1  
  
    // Enquanto o contador for menor que 10  
    // repita as instruções abaixo  
    ENQUANTO (contador <= 10) FAÇA  
        escreval(contador)  
        contador <- contador + 1 // Incrementa a var. contador em + 1  
    FIMENQUANTO  
fimalgoritmo
```

No exemplo anterior, o contador começa de zero e o laço se repetia enquanto ele não fosse 100 (isso totalizava 100 repetições). Nesse exemplo, começamos o contador de 1 e, como queremos repetir 10 vezes, a condição do laço é o contador ser menor ou igual a 10 e não só menor (o que totalizaria 10 repetições). Você, programador, é quem define como o laço vai ser "controlado".

Código Completo

VisuAlg

algoritmo "Contador"

var

contador: INTEIRO

inicio

// Contador iniciando em 1
contador <- 1

// Enquanto o contador for menor que 10

// repita as instruções abaixo

ENQUANTO (contador <= 10) FAÇA

 escreval(contador)

 contador <- contador + 1 // Incrementa a var. contador em + 1

FIMENQUANTO

fimalgoritmo

Código Completo

Portugol Studio

```
programa
{
    funcao inicio()
    {
        // Contador iniciando em 1
        inteiro contador = 1

        // Enquanto o contador for menor que 10
        // repita as instruções abaixo
        enquanto (contador <= 10) {
            escreva(contador + "\n")
            contador = contador + 1
        }
    }
}
```

Importante

- O laço de repetição terá sua execução interrompida quando a condição que “controla o laço” for testada e não for verdadeira.
- Entretanto, se dentro do laço a condição que “controla” o laço de repetição torna-se falsa, mas há outras instruções ainda a serem executadas, antes de testar a condição novamente, essas instruções serão executadas normalmente.
- O laço só “para” quando o teste da condição que o controla resulta em FALSO. Esse teste só acontece no começo de cada iteração (repetição). No exemplo abaixo, as instruções após o contador seriam executadas, mesmo quando o contador atingir o valor de 100.

```
contador <- 0
```

```
  enquanto (contador < 100) faça
```

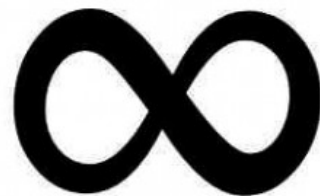
```
    contador <- contador + 1
```

```
      escreva("o valor do contador é " + contador)
```

```
  fimenquanto
```

Loop Infinito

- Uma situação muito comum em programação é a criação de um laço infinito, que corresponde ao momento em que o programa entra em uma estrutura de repetição, mas a condição que controla o laço NUNCA resulta em um valor FALSO.
- Sabendo disso, cuidado ao definir a condição de parada para que, em algum momento, essa condição seja alcançada e o laço pare.
- No exemplo anterior, a condição de parada foi o contador ter o valor maior ou igual que 100. Caso não houvesse o incremento do valor da variável em +1 a cada repetição o valor do contador não mudaria e nunca seria maior ou igual 100, o que acabaria em um loop infinito.



Exercício

1. Chico tem 1,50m e cresce 2 centímetros por ano, enquanto Juca tem 1,10m e cresce 3 centímetros por ano. Construir um algoritmo que calcule e imprima quantos anos serão necessários para que Juca seja maior que Chico.
2. Escreva um algoritmo para ler 2 valores e se o segundo valor informado for ZERO, deve ser lido um novo valor, ou seja, para o segundo valor não pode ser aceito o valor zero e imprimir o resultado da divisão do primeiro valor lido pelo segundo valor lido.

Aprenda Mais...

- Curso em Vídeo - Estruturas de Repetição 1 - Curso de Algoritmos #09 - Gustavo Guanabara <<https://youtu.be/U5PnCt58Q68>>
- Jovem Programador - Lógica de Programação com VisualG Estrutura de Repetição - Enquanto - 05 <<https://youtu.be/8-JWuzb-glE>>
- Programando do Zero - Portugal - Utilizando o Enquanto #20 <<https://youtu.be/IIeikUjkveU>>
- Bóson Treinamentos - 13 - Lógica de Programação - Estruturas de Repetição (Loop) - ENQUANTO <<https://youtu.be/6BLB0fBqzlg>>
- Estrutura de repetição ENQUANTO <<http://www.dicasdeprogramacao.com.br/estrutura-de-repeticao-enquanto/>>

Repetir com Interação do Usuário

- Vamos a mais um estudo de caso simples: Por algum motivo precisamos criar um software que some números digitados pelo usuário e **enquanto** o usuário quiser digitar números, o programa deve ir somando todos eles. No momento que ele desejar parar de digitar, o programa deve imprimir na tela a soma de todos os números.
- Primeiro, pensamos em criar um ENQUANTO, para sempre pedir para o usuário ir digitando os números para serem somados:

```
enquanto (<usuário quiser>) faça  
    escreva("Digite um número para ser somado: ")  
    leia(numero)  
    soma <- soma + numero  
fimenquanto
```

Repetir com Interação do Usuário

- Observando o código anterior, vemos que usamos duas variáveis, uma para receber o número do usuário (numero: inteiro) e outra para incrementar a soma dos números (soma: inteiro), então subentendemos que essas variáveis já foram declaradas.
- Então, o código anterior vai sempre pedir para o usuário um número e esse número vai sendo somado a variável soma.
- Só falta a condição de parada, que, como descrito antes, seria até o usuário desejar parar de digitar, para isso podemos simplesmente perguntar pra ele se ele quer continuar digitando ou não.

```
enquanto (resposta = "S") faça
    escreva("Digite um número para ser somado: ")
    leia(numero)
    soma <- soma + numero
    escreva("Deseja continuar digitando? [S/N] ")
    leia(resposta)
fimenquanto
```

Repetir com Interação do Usuário

- O que fizemos foi simplesmente criar uma variável (resposta: caractere), colocamos a condição de parada para que seja se o valor dessa variável for igual a S e no fim perguntamos se o usuário quer continuar, se ele digitar S a verificação vai retornar **verdadeiro**, afinal o valor da resposta vai ser S, e então o loop vai continuar, caso o usuário digite qualquer outro valor o loop para.
- Se deixarmos o código como está no slide anterior, o loop nunca vai iniciar, pois não atribuímos nenhum valor a variável resposta, ou seja ela não é igual a S ainda, portanto a comparação não retorna **verdadeiro** na verificação do **enquanto**. No código abaixo, damos o valor **"S"** pra ela antes do loop, para garantir que se vai entrar no loop, ao menos uma vez.

```
resposta <- "S"
```

```
enquanto (resposta == "S") faça  
    escreva("Digite um número para ser somado: ")  
    leia(numero)  
    soma <- soma + numero  
    escreva("Deseja continuar digitando? [S/N] ")  
    leia(resposta)
```

```
fimenquanto
```

Repetir com Interação do Usuário

- No fim é só imprimir a soma dos números:

algoritmo "Contador"

var

resposta: caractere

numero, soma: inteiro

inicio

```
resposta <- "S" // Inicia com valor S para entrar no loop
```

```
ENQUANTO (resposta = "S") FAÇA
```

```
    escreva("Digite um número para ser somado: ") // pede um numero ao usuário
```

```
    leia(numero) // atribui o numero digitado a variável
```

```
    soma <- soma + numero // soma o numero a variável soma
```

```
    escreva("Deseja continuar digitando? [S/N] ") // pede uma resposta
```

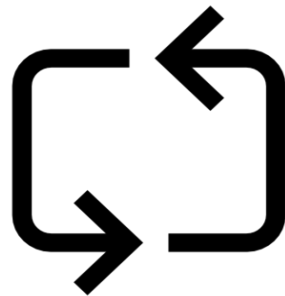
```
    leia(resposta) // atribui o que o usuario digitou a variável
```

```
FIMENQUANTO
```

```
    escreva(soma) // imprime a soma dos números
```

fimalgoritmo

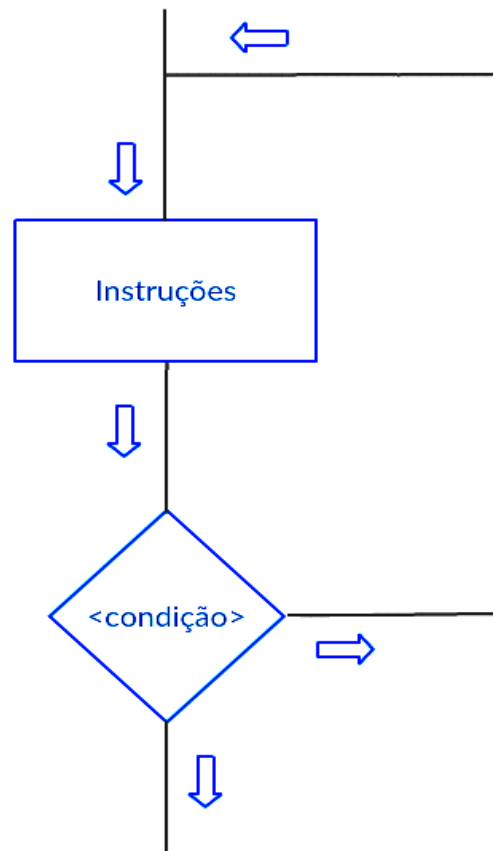
REPITA ATÉ...



- Além do ENQUANTO, existem outras duas estruturas de repetição em Portugal: REPITA-ATÉ e PARA.
- Podemos observar que na estrutura ENQUANTO o loop só vai acontecer se a condição for verdadeira quando for testada pela primeira vez, ou seja o escreva, a soma, o leia e outros comandos que colocarmos no bloco do enquanto só vão ser executados se a expressão testada retornar um **verdadeiro**.
- O funcionamento da estrutura REPITA-ATÉ é bem parecida com a do ENQUANTO, porém a condição é testada somente depois que é executado o seu bloco de código, então os comandos são executados e depois é verificado se a expressão retorna um **verdadeiro**.

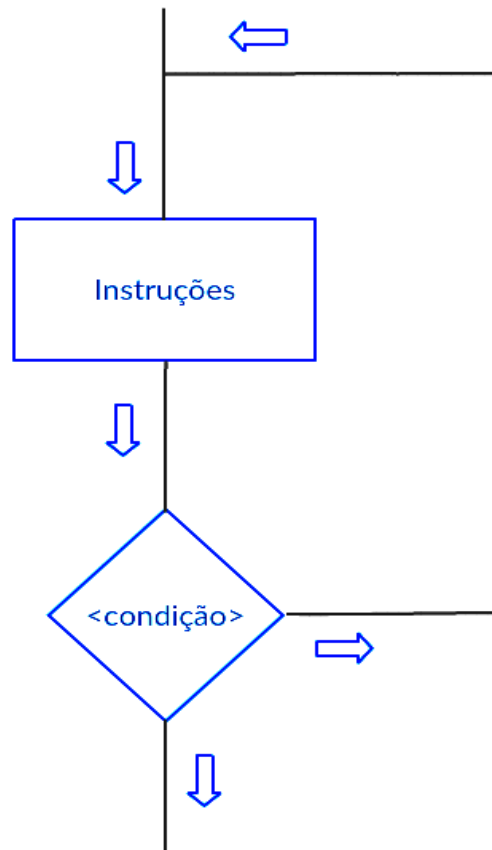
REPITA ATÉ...

- Outra diferença para o ENQUANTO é que, no REPITA-ATÉ, o loop vai ser executado ao menos uma vez, antes da condição que o controla ser testada, pois essa condição fica ao final da estrutura.
- Além disso, o REPITA-ATÉ para a repetição se a condição de parada for verdadeira.



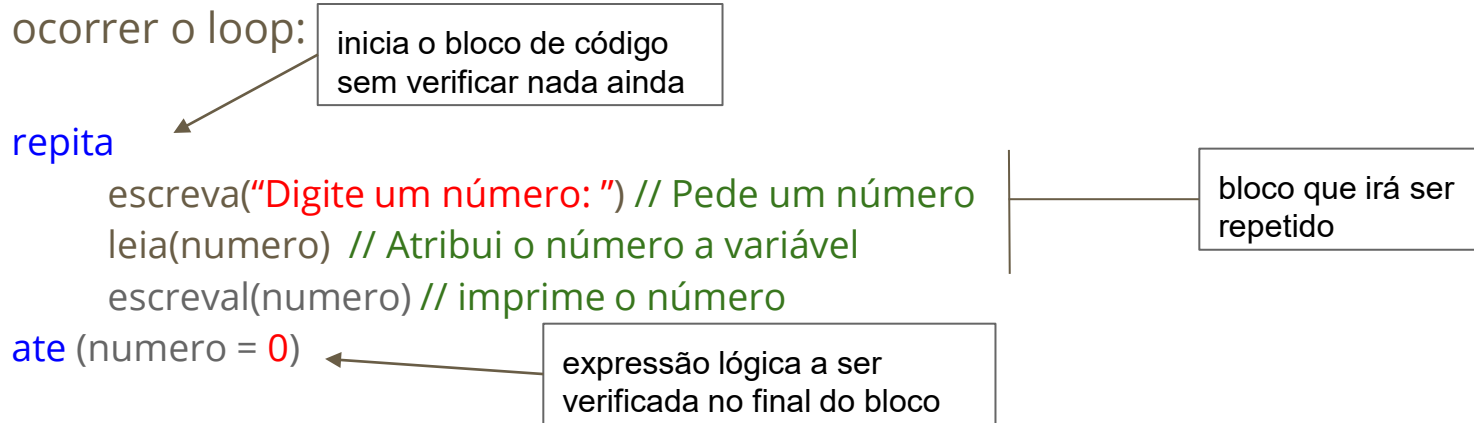
REPITA ATÉ...

- Em outras palavras, se a expressão verificada resultar em um **verdadeiro** a execução sai do loop, ao contrário do ENQUANTO, que a condição para continuar no loop é que a expressão verificada resulte em **verdadeiro**.
- O REPITA-ATÉ mantém o loop funcionando até que uma certa condição seja atendida. Então, a repetição para.



Estrutura do REPITA-ATÉ

- Seu funcionamento é muito parecido com o ENQUANTO, só que todo REPITA-ATE inicia seu bloco de código, antes de testar a expressão que determina se vai ocorrer o loop:



- No caso, a estrutura acima vai executar seu bloco de código pelo menos uma vez. Após realizar a leitura do número, será feito o teste do até. Caso a condição ainda não seja atendida, o bloco continuará se repetindo. Ou seja, é para repetir até a condição ser verdadeira.

Estrutura do REPITA-ATÉ

- Vamos ver como fica o estudo de caso do Contador usando a estrutura REPITA-ATE:

algoritmo "Contador"

var

resposta: caractere

numero, soma: inteiro

inicio

REPITA

 escreva("Digite um número para ser somado: ") // pede um numero ao usuário

 leia(numero) // atribui o numero digitado a variável

 soma <- soma + numero // soma o numero a variável soma

 escreva("Deseja continuar digitando? [S/N] ") // pede uma resposta

 leia(resposta) // atribui o que o usuario digitou a variável

ATE (resposta <> "S")

 escreva(soma) // imprime a soma dos números

fimalgoritmo

Código Completo

VisuAlg

algoritmo "Contador"

var

resposta: caractere

numero, soma: inteiro

inicio

REPITA

 escreva("Digite um número para ser somado: ")

 leia(numero)

 soma <- soma + numero

 escreva("Deseja continuar digitando? [S/N]")

 leia(resposta)

ATE (resposta <> "S")

 escreva(soma)

fimalgoritmo

Código Completo

Portugol Studio

```
programa {  
  
    funcao inicio() {  
        cadeia resposta  
        inteiro numero  
        inteiro soma = 0  
  
        faca {  
            escreva("Digite um número para ser somado: ")  
            leia(numero)  
            soma <- soma + numero  
            escreva("Deseja continuar digitando? [S/N]")  
            leia(resposta)  
        } enquanto (resposta == "S")  
    }  
}
```

Exercício

1. Escreva um algoritmo para ajudar uma empresa em uma pesquisa de qualidade sobre o sabor de seus chocolates, ele deve apresentar na tela as seguintes escolhas para o usuário:

1 - Saboroso

2 - Sabor Normal

3 - Gosto Ruim

O usuário deve digitar um número correspondente à resposta (1, 2, 3), o programa deve ir pedindo para o usuário respostas até que ele digite um número que não corresponda a nenhuma das escolhas. No fim o programa deve imprimir o número de respostas dada e a porcentagem de cada uma das escolhas.

Aprenda Mais...

- Curso em Vídeo - Estruturas de Repetição 2 - Curso de Algoritmos #10 - Gustavo Guanabara
<<https://youtu.be/fP49L1i-HU>>
- JovemProgramadorBR - Lógica de Programação com VisualG Estrutura de Repetição - Repita - 06 <<https://youtu.be/cgfe08eg85o>>
- Hildebrando Ferreira do Nascimento - Aula7 - Laço REPITA - VISUALG
<<https://youtu.be/5JdAfRat8WI>>
- Bóson Treinamentos - 14 - Lógica de Programação - Estruturas de Repetição (Loop) - REPITA ATÉ <<https://youtu.be/PZJmfp42k00>>
- Evandro Júnior - Aula 9 - Estrutura de repetição Repita.. ate <<https://youtu.be/p2a7eV3vFRw>>
- Estrutura de repetição REPITA-ATÉ
<<http://www.dicasdeprogramacao.com.br/estrutura-de-repeticao-repita-ate/>>
- Comandos de Repetição
<<http://www.apoioinformatica.inf.br/produtos/item/14-comandos-de-repeticao>>

Estrutura PARA

- Quando queremos implementar um loop com o número pré-definido de iterações, podemos outra opção é utilizar a estrutura de repetição: **PARA**.
- Vamos voltar com o exemplo do algoritmo para contar até 10:

```
contador <- 0
```

```
    enquanto (contador < 100) faça
```

```
        escreva("Serei um bom aluno a partir de agora")
```

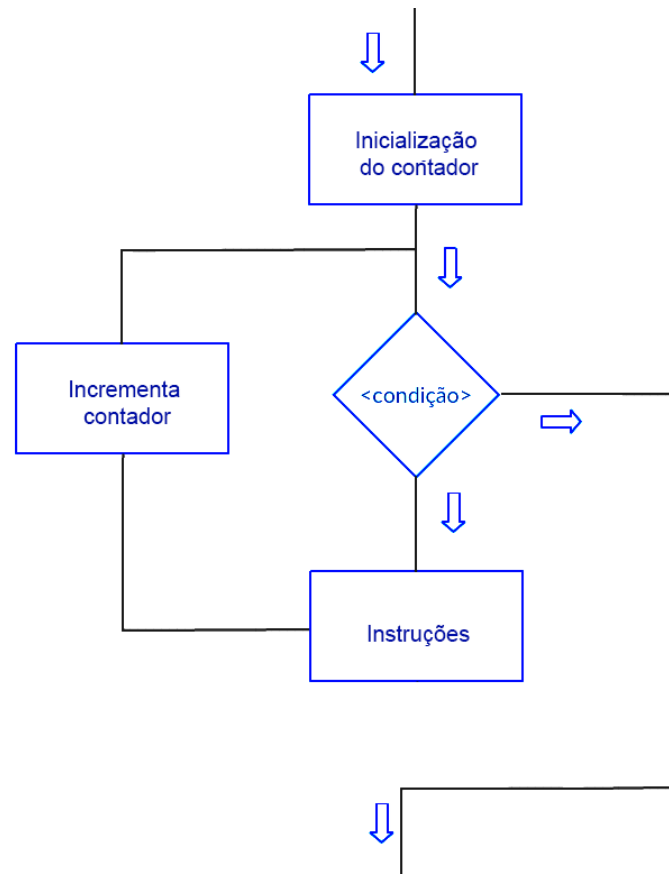
```
        contador <- contador + 1
```

```
    fimenquanto
```

- Para utilizar o enquanto, precisamos criar uma variável "contador" para que ela controle a condição de parada. Essa variável deve ser incrementada a cada repetição do loop.

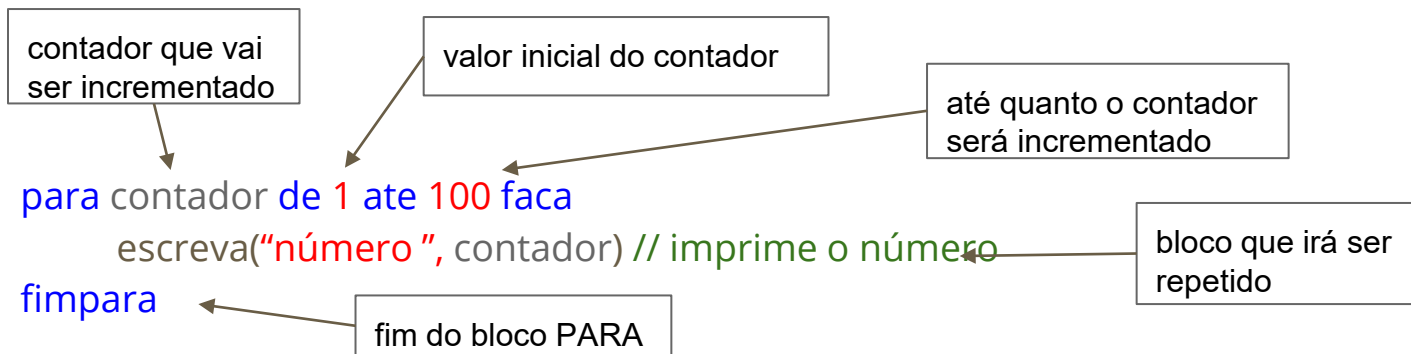
Estrutura PARA

- O **PARA** serve justamente para facilitar esse tipo de iteração,
 - Partindo do pressuposto que sabemos quantas vezes a estrutura tem que se repetir, podemos usar a estrutura **PARA**, sem que haja a necessidade de inicializar ou incrementar um contador, pois a própria estrutura fará isso implicitamente e automaticamente.
- Por exemplo, em um algoritmo que realize a contagem dos números até 100, com a estrutura **PARA**, não precisaríamos criar uma condição de parada, pois ela já existe, que é o próprio loop repetir 100 vezes.



Estrutura PARA

- A estrutura do **PARA** tem a seguinte sintaxe:



- Basicamente, podemos ler a estrutura acima como: repita o bloco de código de 1 até 100 (i.e., 100 vezes). Em cada repetição, a variável contador é incrementada em + 1, mudando, assim, o valor que estará sendo impresso
- O interessante é que não precisamos implementar nenhum codificar explicitamente o incremento do contador, pois isso é feito automaticamente, quando usamos essa estrutura.

Algoritmo de Tabuada

- Para entender melhor, vamos construir, passo-a-passo, um algoritmo simples de Tabuada: onde o usuário digita um número e o programa deve imprimir esse número multiplicado por números de 1 a 10.

1. Primeiro, vamos declarar as variáveis. Serão três: uma vai ser o contador, a outra que vai representar o número que o usuário vai digitar e, por ultimo, uma que vai ser o resultado da multiplicação em cada repetição do loop.

var

numero, contador: inteiro

mult: real

inicio

.....

Algoritmo de Tabuada

2. Em seguida, pedimos para o usuário digitar um número cuja tabuada iremos apresentar:

inicio

escreva("Digite o número para calcular o tabuada: ")

leia(numero)

.....

3. Organizamos a estrutura do PARA usando a variável contador que vai iniciar com o valor 1 e vai ser incrementada até 10, ou seja o bloco de código vai repetir de 1 até 10 (dez vezes):

para contador de 1 ate 10 faca

.....

Algoritmo de Tabuada

4. Realizamos a multiplicação usando a variável contador, ou seja, em cada repetição a variável mult vai receber o valor da multiplicação da variável numero com o contador (esse que vai ter o valor sempre incrementado em +1).

```
para contador de 1 ate 10 faca  
    mult <- numero * contador
```

5. Por último, imprimimos na tela de o resultado da multiplicação com uma formatação que seja mais legível, já que se trata de uma tabuada, e fechamos a estrutura com a palavra reservada FIMPARA.

```
    para contador de 1 ate 10 faca  
        mult <- numero * contador  
        escreva(numero, " X ", contador, " = ", mult)  
    fimpara  
finalgoritmo
```

Algoritmo de Tabuada

- Ao executar, o programa vai pedir ao usuário um número e vai imprimir algo como:

$$2 \times 1 = 2$$

$$2 \times 2 = 4$$

$$2 \times 3 = 6$$

...

$$2 \times 10 = 20$$

X	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	12	14	16	18	20
3	0	3	6	9	12	15	18	21	24	27	30
4	0	4	8	12	16	20	24	28	32	36	40
5	0	5	10	15	20	25	30	35	40	45	50
6	0	6	12	18	24	30	36	42	48	54	60
7	0	7	14	21	28	35	42	49	56	63	70
8	0	8	16	24	32	40	48	56	64	72	80
9	0	9	18	27	36	45	54	63	72	81	90
10	0	10	20	30	40	50	60	70	80	90	100

Código Completo

VisuAlg

```
algoritmo "Contador"  
var  
    numero, contador: inteiro  
    mult: real  
    inicio  
inicio  
    escreva("Digite o para calcular o tabuada: ")  
    leia(numero)  
  
    PARA contador DE 1 ATE 10 FAÇA  
        mult <- numero * contador  
        escreval(numero, " X ", contador, " = ", mult)  
    FIMPARA  
fimalgoritmo
```

Código Completo

Portugol Studio

```
programa {  
  
    funcao inicio() {  
        inteiro numero, resultado, contador  
  
        escreva("Digite o para calcular o tabuada: ")  
        leia(numero)  
  
        para (contador = 1; contador <= 10; contador++) {  
            resultado = numero * contador  
            escreva (numero, " X ", contador, " = ", resultado , "\n")  
        }  
    }  
}
```

Exercício

1. Escreva um algoritmo para imprimir os números de 1 (inclusive) a 10 (inclusive) em ordem decrescente.
2. Escreva um algoritmo para imprimir os 10 primeiros números inteiros maiores que 100.

Aprenda Mais...

- Curso em Vídeo - Estruturas de Repetição 3 - Curso de Algoritmos #11 - Gustavo Guanabara <<https://www.youtube.com/watch?v=WJQz20i7Cyl>>
- JovemProgramadorBR - Lógica de Programação com VisualG - Estrutura de Repetição - Para - 04
<<https://www.youtube.com/watch?v=lQjGDLSRUdO>>
- Estrutura de repetição PARA
<<http://www.dicasdeprogramacao.com.br/estrutura-de-repeticao-para/>>
- Me Salva - Me Salva! ALP08 - Algoritmos com Iteração - comando "for"
<<https://www.youtube.com/watch?v=YEee8TattRo>>
- Tuto Studio - Visualg Aula 15 - LOOP PARA -
<<https://youtu.be/gUWhyLsNMDc>>