Estruturas de Repetição

As estruturas de repetição, como o próprio nome já diz, são utilizadas para REPETIR determinados comandos do algoritmo. O número de vezes que esses comandos serão repetidos pode ser especificado pelo programador ou analisados de acordo com uma condição.

Looping com teste lógico no início (**enquanto**... **faca**... **fimenquanto**).

A estrutura enquanto é utilizada para se repetir comandos nos casos em que o número de repetições não é conhecido.

Os comandos vão sendo executados repetidamente até que uma condição específica seja satisfeita. Essa condição que interrompe a repetição dos comandos é conhecida como teste ou flag.

Sua sintaxe é:

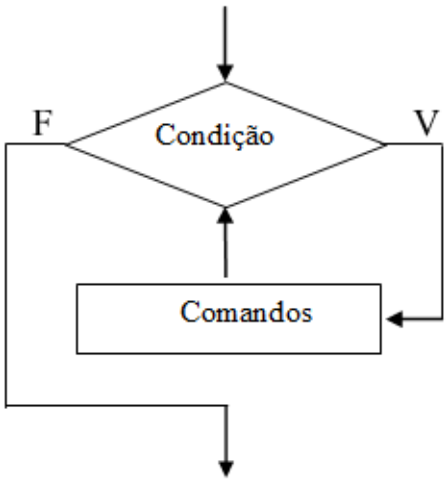
enquanto (condição) **faca**

comando 1

...

comando n

fimenquanto



Problema:
Escreva um algoritmo para calcular a média da idade de um grupo de 5 pessoas.
Saída: Quais os resultados que o algoritmo deve fornecer?
Entrada: Quais os insumos necessários para se obter a saída?
Processamento: Como transformar os insumos na saída de dados?
Neste caso
Saída – média da idade das 5 pessoas
Entrada: a idade das 5 pessoas [id1, id2, id3,... ,id5]
Processamento: media = [id1, id2, id3,... ,id5]/5

```
Var

    id1, id2, id3, id4, id5: inteiro media: real

inicio

    escreva("Digite a idade da pessoa 1:")

    leia(id1)

    escreva("Digite a idade da pessoa 2:")

    leia(id2)

    escreva("Digite a idade da pessoa 3:")

     leia(id3)

    ...

    escreva("Digite a idade da pessoa 5:")

    leia(id5)

    media := (id1+id2+id3+id4+id5)/5

    escreva("A média das idades é ",media)

fimalgoritmo
```

A solução apresentada esta correta, imagina, se ao invés de 5 idades, fosse mais de 1000 idades. Teríamos que declarar 1000 idades, o que torna inviável e teríamos que escrever mil linhas para entrada dessas idades. Observe que existem códigos no programa que se repete, isso é chamado de padrão de comportamento. Padrão de comportamento evidenciam quais estruturas do programa que se repetem. Essas estruturas são aquelas que podemos colocar dentro de um laço de repetição. Então laços de repetições é o ponto focal dessa

aula e serão utilizados para resolver o problema da necessidade de diminuir o código.
Observe o código a seguir:

```
Algoritmo “exemplo1”
var
        idade, c, somaidade: inteiro
        media: real

inicio
        c := 1
        somaidade := 0
        enquanto (c <= 5) faca
                escreva("Digite a idade da pessoa ",c, " : ")
                leia(idade)
                somaidade := somaidade + idade
                c := c + 1
        fimenquanto

        media := (somaidade) / (c - 1)
        escreva("A média das idades é igual a ",media)

fimalgoritmo
```

É preciso saber que a estrutura de repetição permite repetir diversas vezes determinado trechos do código. A estrutura de repetição minimiza o trabalho do desenvolvedor, evitando que digite longos trechos de códigos. Para saber quais trechos de códigos são considerados a repetições pelo uso de estruturas, basta identificar os padrões de comportamento. Então, padrões de comportamentos são aqueles trechos de códigos bastante parecidos.
Quando na sequencia de comandos necessita ser executado repetidas vezes, usamos uma estrutura de repetição. A estrutura de repetição também é conhecida como laço ou loop. Qualquer linguagem de programação possui comandos que implementam a estrutura de repetição.

Comando ENQUANTO permite executar instruções que são repetidas "enquanto" determinada condição for verdadeira.
Sintaxe:

```
enquanto (condição) faca
        instrução1
        ...
        instrução n
fimenquanto
```

Tudo que estiver dentro do bloco será executado enquanto a condição que estiver entre parentes for verdadeira.

Para exemplificar de uma forma simples, vamos analisar o código a seguir para imprimir 5 asterisco ' * ' na tela.

```
Algoritmo "exemplo2"
var

inicio
escreval(" * ")
escreval(" * ")
escreval(" * ")
escreval(" * ")
escreval(" * ")

fimalgoritmo
```

Mas se fossem 1000 asteriscos?
Seria inviável escrever o comando escreva( ) mil vezes.

A solução para esse problema é justamente o uso da estrutura de repetição. Vamos escrever o mesmo código usando o comando ENQUANTO.

```
Algoritmo "exemplo2"

var

c: inteiro //variável de controle inicio

inicio

c := 1 //iniciando a variável de controle

enquanto (c<=1000) faca
        escreval(" * ")
        c := c + 1
fimenquanto

fimalgoritmo
```

Uma observação importante, é sempre, modificar a variável de controle dentro do laço enquanto. Desta forma você vai garantir que em determinado momento a condição se torne FALSO, e assim, o laço será finalizado.
Para facilitar a didática, o exemplo a seguir imprime três " * " em vez de 1000.

```
Algoritmo "exemplo_asterisco"

Var
c: inteiro //variável de controle inicio
c := 1 //iniciando a variável de controle
        enquanto (c<=3) faca
                escreval(" * ")
                c := c + 1 //nessa linha a variável c é incrementada
        fimenquanto
fimalgoritmo
```

c	c <= 3
1	verdadeiro
2	verdadeiro
3	verdadeiro
4	falso

Observe que diferentemente de outros códigos esse programa não é sequencial. Pelos seguintes pontos:
1º) retorna para o ponto anterior para testar a condição.
2º) ele pula instruções, caso em que a condição é falsa.

No próximo exemplo, tem o uso do comando *enquanto* um pouco mais elaborado. Vamos analisar o problema da tabuada.

Escreva um algoritmo para mostrar na tela a tabuada do número n. Sendo n um número de 1 a 10.

Em outras palavras o algoritmo deve mostrar a tabuada do número que o usuário escolher.
Se o número for 3, o resultado será como mostrado a seguir:
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18

3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

Vamos aplicar a técnica para resolução de problemas:
Entrada: um número (n)
Saída: mostrar na tela as 10 linhas com a tabuada de n processamento:

n * 1,
n * 2,
n * 3,
...
n * 9
n * 10

Um exemplo da resolução desse problema, sem uso da estrutura de repetição pode ser visto a seguir:

```
Algoritmo "exemploTabuada"
   var
        n: inteiro

   inicio

         escreva("Digite um número: ")
         leia(n)


         escreval(n, " x 1 = ",n*1)
         escreval(n, " x 2 = ",n*2)
         escreval(n, " x 3 = ",n*3)
         escreval(n, " x 4 = ",n*4)
         escreval(n, " x 5 = ",n*5)
         escreval(n, " x 6 = ",n*6)
         escreval(n, " x 7 = ",n*7)
         escreval(n, " x 8 = ",n*8)
         escreval(n, " x 9 = ",n*9)
         escreval(n, " x 10 = ",n*10)

fimalgoritmo
```

O código é simples, mas pode ser editado para evitar a repetição desnecessária de algumas linhas de código. A repetição desnecessária deixa o código grande e

alguns casos bem mais complexos, inviabilizando o trabalho com as linhas de códigos. Para minimizar a repetição desnecessária do código, vamos identificar os padrões de comportamento e evidenciar as estruturas que se repetem no programa.
Agora que identificamos os padrões de comportamento, vamos elaborar o código do problema utilizando a estrutura de repetição *enquanto*.
O que tem que fazer é basicamente, colocar os padrões de repetições dentro do bloco do comando *enquanto*.
Observem como o código ficou bem menor:

```
Algoritmo “exemploTabuada”
var
        n, c: inteiro
inicio

        escreva(“Digite um número: ”)
        leia(n)
        c := 1
        enquanto (c<=10) faca
                escreval(n, " x ",c, " = ",n*c)
                c := c + 1
        fimenquanto

fimalgoritmo
```

As variáveis que controlam as repetições de laços se chamam variáveis de controle, o que significa dizer que a variável c é uma variável de controle. A variável de controle deve ser sempre iniciada, e isso esta demostrado na linha antes da estrutura enquanto. O bloco do comando enquanto é executado varias vezes, até que a condição seja falsa. É necessário sempre inicializar a variável de controle. É necessário mudar o valor da variável de controle dentro do bloco repetição para não correr o risco de laços infinitos.

Exercícios de fixação:
1) Construir um programa que fique lendo números, até que seja digitado um número negativo. Após interromper as leituras, escrever quantos números foram lidos.

2) Escrever um programa que leia os valores dos salários de uma empresa. O programa deve interromper a leitura quando for digitado um salário menor ou igual a zero. Após efetuar as leituras, o programa deverá oferecer as seguintes informações.
a. A quantidade de salários;

b. A soma dos salários;
c. A média dos salários.

3) Construir um programa para ler os salários de uma empresa. O programa deve interromper a leitura quando for digitado um salário negativo. Após as leituras escrever:
a) A quantidade de salários lidos;
b) A soma dos salários;
c) A média dos salários;
d) O valor do maior salário;
e) O valor do menor salário.

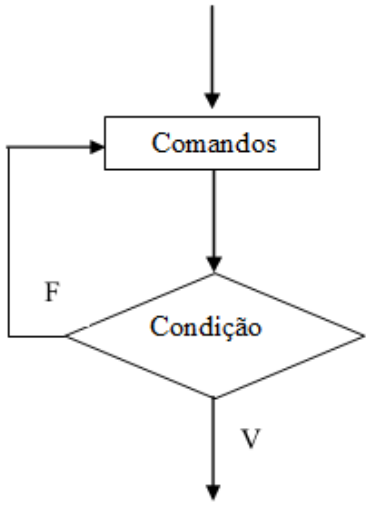
Looping com teste lógico no final (**repita**... **ate**) ou (**faca**... **enquanto**)

Caracteriza-se por uma estrutura que efetua um teste no final de um laço, verificando se é permitido ou não executar novamente o conjunto de comandos no interior do mesmo.
Deve-se existir um contador e esse necessita ser declarado e inicializado fora do laço.

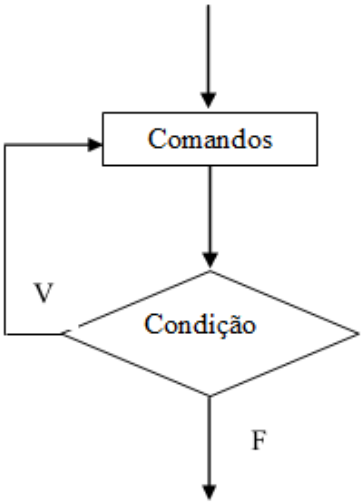
É obrigatório que o contador mude seu valor dentro do laço, caso contrário, entra-se em laço infinito.

A condição do laço faça... enquanto fica no final, após o fechamento do bloco.
As instruções dentro do laço faca... enquanto serão executadas, pelo menos, usa vez.

repita ... ate



faca... enquanto



Vamos analisar o problema da tabuada.
Escreva um algoritmo para mostrar na tela a tabuada do número n. Sendo n um número de 1 a 10.

Em **VISUALG:**

```
Algoritmo "exemploTabuada"

var
        n, c: inteiro inicio
        c  := 1
         escreva("Qual número você deseja gerar a tabuada: ")
         leia(n)

        repita
                escreval(n, " x ", c, " = ",c * n)
                 c := c + 1
        ate(c>10)

fimalgoritmo
```

Em **PORTUGOL WEB STUDIO**

```
inteiro c, n
c = 1
escreva("Qual número você deseja gerar a tabuada: ")
leia(n)

faca{
      escreval(n, " x ", c, " = ",c * n)
      c = c + 1
}enquanto (c<=10);
```

Exemplo no **VISUALG:**

```
var
  num: real
inicio

      repita
            escreva("Digite um número: ")
            leia(num)
            escreva("O quadrado é ",num^2)

      ate(num<0) // até que a condição seja verdadeira

fimalgoritmo
```

Comando **PARA:**

O comando PARA permite executar instruções que são repetidas enquanto determinada condição for verdadeira.
Observe o código da estrutura para visualg:
Sintaxe:
para <variável> **de** <inicio> **ate** <fim> **faca**
bloco de instruções
fimpara

Em **PORTUGOL WEB STUDIO:**

```
para (<variavel = valorinicial>; <variavel = valorfinal>; <variavel><incremento> ) {
        bloco de instruções
}
```

A estrutura para, entre parêntese a variável de controle, condição e incremento da variável de controle.
Tudo que estiver no bloco de instruções, será executado enquanto a condição for verdadeira.
Observe que assim como o comando enquanto, a condição do comando para fica dentro do parêntese. Observe que diferentemente do comando enquanto a variável de controle é inicializada e finalizada dentro do parêntese. No comando enquanto a variável de controle é inicializada antes do comando e modificada dentro do bloco.

Para exemplificar a funcionalidade do comando para vamos analisar o código para imprimir 5 asteriscos na tela.

```
escreva("*")
escreva("*")
escreva("*")
escreva("*")
escreva("*")
```

Basta digitar o comando escreva asterisco por 5 vezes. Mas se quiser imprimir 1000 asterisco? Seria inviável digitar o comando escreva 1000 vezes.
Para a solução desse problema é justamente a estrutura de repetição. Para apresentar uma das soluções usando a estrutura de repetição, vamos escrever o mesmo código usando o comando PARA.

Usando o **VISUALG**:

```
para i de 1 ate 5 faca
      escreval("*")
fimpara
```

Usando **PORTUGOL WEBSTUDIO:**

```
para (c = 1;c<=5;c = c+1){
      escreval("*")
}
```

A variável c será a variável responsável para controlar o número de repetições do laço, por causa disso é chamada de variável de controle. Sempre iniciar a variável de controle entre parentes. Outro ponto importante

é sempre modificar a variável de controle que em determinado momento a condição se torne falsa e assim finalizar a execução do laço.
Observe que diferente de outros códigos, esse programa não é sequencial pelos seguintes pontos: Ele retorna para as linhas anteriores para testar a condição e ele pula a instrução caso em que a condição é falsa.

Vamos analisar o problema da tabuada.
Escreva um algoritmo para mostrar na tela a tabuada do número n. Sendo n um número de 1 a 10.

Vamos aplicar a técnica para resolução de problemas:

Entrada: um número (n)
Saída: mostrar na tela as 10 linhas com a tabuada de n

Processamento:

n * 1,
n * 2,
n * 3,
...
n * 9
n * 10

Solução em **VISUALG:**

Algoritmo “exemploTabuada”

```
var
    n, c: inteiro
inicio
    escreva("Qual número você deseja gerar a tabuada: ")
    leia(n)

    para c de 1 ate 10 faca
        escreval(n, " x ", c, " = ",c * n)
```

```
    fimpara

fimalgoritmo
```

Solução em **PORTUGOL WEB STUDIO:**

```
inteiro n, c

escreva("Qual número você deseja gerar a tabuada: ")
leia(n)
para (c = 1;c<=10;c = c+1){
    escreva(n, " x ", c, " = ",c * n)
    escreva("\n")
}
```

Comparações entre o comando ENQUANTO com o comando PARA :

1 = inicialização de variáveis

2 = teste de condição

3 = incremento

n, c: inteiro
c := 1 (1)
escreva("Digite um número: ")
leia(n)
(2)
enquanto (c<=10) faca
escreval(n, " x ", c, " = ",c * n)
c := c + 1 (3)
fimenquanto
	n, c: inteiro
	escreva("Digite um número: ")
	leia(n)
	(1) (2) (3)
	para (c = 1; c<=10; c = c+1){
	escreva(n, " x ", c, " = ",c * n)
	escreva("\n")
	}

O comando PARA parece mais “enxuto” pois permite em uma única linha, os três passos 1, 2 e 3.

A estrutura de repetição minimiza o número de linhas no código.

Exercícios de fixação:

1) Construir um programa que leia 10 números e após as leituras escreva:

a) O somatório dos números lidos;

b) O somatório dos números lidos maiores ou iguais a zero;

c) O somatório dos números lidos menores que zero.

2) Faça um programa que mostre a tabuada de um número digitado pelo usuário.

3) Apresentar o total da soma dos cem primeiros números inteiros (1+2+3+4+5+...97+98+100).

4) Elaborar um programa que efetue o cálculo da fatorial do valor 5 e apresente o resultado dessa operação.

Para entender o problema proposto, considere que, do ponto de vista matemático, fatorial é o produto dos números naturais desde 1 até o limite informado, neste caso 5. Assim sendo, a fatorial do valor 5, representada matematicamente como 5!, é a multiplicação de 1 x 2 x 3 x 4 x 5, que resulta no valor 120. Para efetuar essa operação, são necessárias duas variáveis, uma que controla as iterações do laço e outra que calcula a fatorial propriamente dito. Iniciar as variáveis cont e fat, ambas com valor inicial 1.

Tabela de cálculo da 5!			
cont	fat	fat := fat * cont	comentários
1	1	1	valor inicial das variáveis e fatorial
2	1	2	cálculo da fatorial com o contador em 2
3	2	6	cálculo da fatorial com o contador em 3
4	6	24	cálculo da fatorial com o contador em 4
5	24	120	cálculo da fatorial com o contador em 5