Lógica de Programação

Aula 06 - Repetição Enquanto

Apresentação da Aula

Nas aulas anteriores, você conheceu as **estruturas de decisão se...então...senão**, além disso, compreendeu como ocorre o processo de **aninhamento** das **estruturas de decisão** e aprendeu, ainda, sobre a estrutura **escolha**. O uso das **estruturas de decisão** é adequado quando há a necessidade de se estabelecer desvios de fluxo na execução dos comandos de um algoritmo e, para isso, é utilizada uma expressão lógica que deverá ser **verdadeira** ou **falsa**.

Agora, irá conhecer outro grupo de estrutura de controle: as **estruturas de repetição**. Na construção de algoritmos, é comum a necessidade de realizar a repetição de um determinado conjunto de comandos presente em algoritmo, seja por um número específico de vezes, seja por uma quantidade indefinida.

Nesta aula, você identificará as características das estruturas **sequenciais** e **não sequenciais**. Irá, ainda, explorar a estrutura de repetição **enquanto** e a sua classificação.

Objetivos

Conhecer as **Estruturas Sequenciais** e **Estruturas não sequenciais**.

Aprender a definir a sintaxe para o comando **ENQUANTO** em estruturas de repetição com teste condicional no início.

Estruturas Sequenciais e Estruturas não sequenciais

A construção dos algoritmos ocorre sempre por meio de uma sequência de comandos estruturados que podem estar dispostos de duas formas diferentes: como **estruturas sequenciais** ou como **estruturas não sequenciais**.

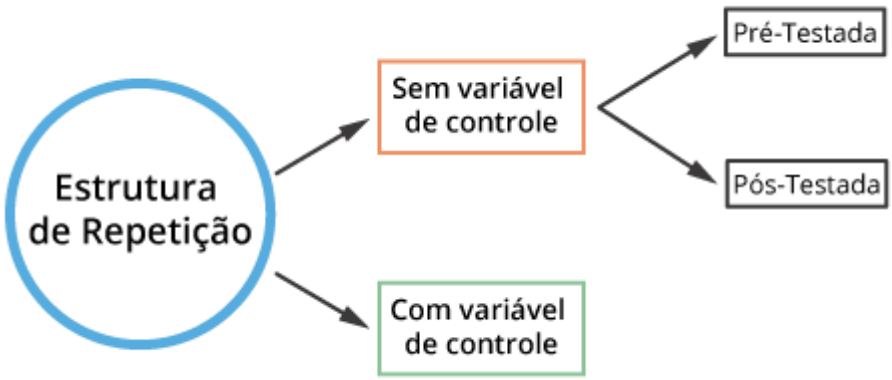
Os algoritmos que utilizam as **estruturas sequenciais** são construídos linearmente. Eles podem ser construídos com ou sem o uso das estruturas de desvio de fluxo **se...então...senão** e **escolha**, por exemplo. A sua execução ocorre sempre do início até o fim, sem a repetição de nenhum bloco de comandos.

Os algoritmos que construímos, até então, foram em **estruturas sequenciais**. Em todos os casos tivemos uma execução do início do nosso código até o fim, mesmo que alguns desses trechos não fossem executados devido ao uso das **estruturas de decisão**.

Já as **estruturas não sequenciais** utilizam comandos de repetição, os quais são responsáveis por executar um conjunto de comandos por repetidas vezes até que uma condição não seja mais satisfeita. Essa condição é responsável por definir quantas vezes esse conjunto de comandos será executado no algoritmo.

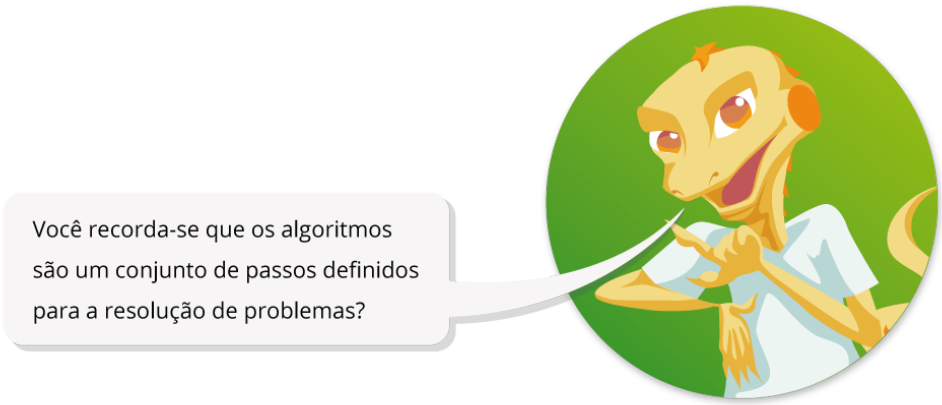
Há dois tipos de estrutura de repetição: as estruturas com **variável de controle** e as **sem variável de controle**, sendo as estruturas sem variável de controle distinguidas, ainda, entre **estruturas pré-testadas** e **estruturas pós-testadas**.

Figura 01 - Estrutura de repetição



Nesta aula, você irá conhecer e exercitar apenas o uso da **estrutura de repetição enquanto**, que é classificada como **sem variável de controle** e **pré-testada**.

Repetindo...



Esses passos são realizados um após o outro. Até o momento, construímos programas que são executados sempre do início até o fim, sem repetir nenhum dos passos, inclusive com trechos que não foram processados quando utilizadas **estruturas de decisão** para definir fluxos de execução. Porém, em nenhum momento tivemos situações nas quais uma mesma linha de código fosse executada mais do que uma única vez.

As linguagens de programação possuem em seu conjunto de instruções as **estruturas de controle**. Essas estruturas definem a maneira como os algoritmos são executados. Você já conheceu duas **estruturas de controle** (a de decisão **se...então...senão** e a **escolha**). Agora conhecerá uma nova estrutura: a de repetição **enquanto**.

A maioria das linguagens de programação possui as **estruturas de repetição**. Estas são utilizadas para realizar a repetição controlada de blocos de comandos, sendo assim, são empregadas quando é necessário repetir uma mesma operação

(ou conjunto de operações) por uma quantidade de vezes determinada ou enquanto uma condição lógica for **verdadeira**.

O bloco abaixo apresenta a sintaxe do comando **enquanto**. Sempre que a **<condição>** for **verdadeira**, os comandos presentes entre **faça** e **fim** serão executados.

```
1  enquanto <condição> faça
2     <instruções>
3  fim
4
```

Essa **estrutura de repetição** é considerada uma estrutura **sem variável de controle**. Você entenderá melhor o conceito de variável de controle na próxima aula, quando conhecer a estrutura de repetição **para**, que possui essa variável. A estrutura **enquanto** é uma **estrutura pré-testada**, pois, antes de executar o conteúdo entre as palavras **faça** e **fim**, ela verifica se a expressão de **<condição>** é **verdadeira**. Se não for **verdadeira**, o conteúdo presente entre o **faça** e **fim** não será executado nenhuma vez e o programa continuará a partir da linha seguinte ao comando **fim** da estrutura **enquanto**.

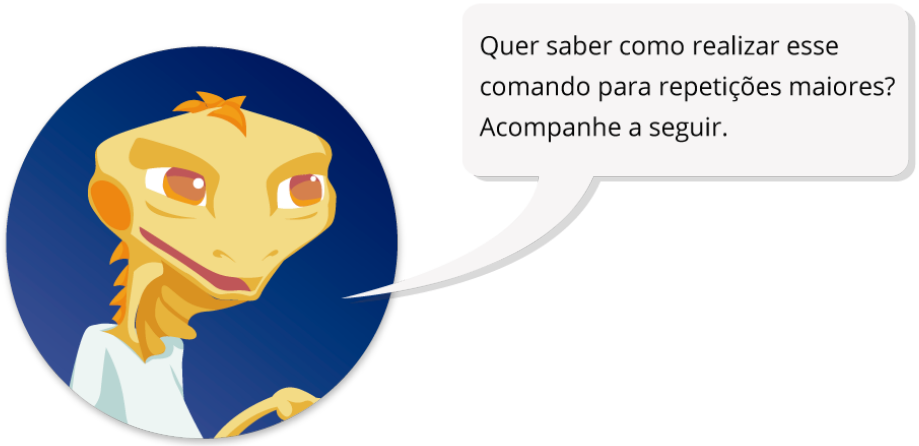
Saiba Mais

Uma outra **estrutura de repetição**, também existente na maioria das linguagens, mas não presente no **Potigol**, chama-se **repita...enquanto**. Essa **estrutura de decisão** é também uma estrutura sem variável de controle, mas, diferentemente da estrutura de repetição **enquanto**, é uma estrutura pós-testar, pois primeiro executa uma única vez o conteúdo presente em sua estrutura e, ao final da primeira execução, avalia a expressão lógica de controle, verificando se a condição é **verdadeira**. Caso a condição seja **verdadeira**, o código presente em sua estrutura é repetido, senão a execução do programa continua a partir da linha seguinte a estrutura.

Se lhe fosse solicitado construir um algoritmo que realiza a impressão da mensagem "Olá, seja bem-vindo!" cinco vezes, como você o construiria? Eu suspeito que você apresentaria uma solução semelhante à apresentada abaixo:

```
escreva "Olá, seja bem-vindo!"
escreva "Olá, seja bem-vindo!"
escreva "Olá, seja bem-vindo!"
escreva "Olá, seja bem-vindo!"
escreva "Olá, seja bem-vindo!"
```

Para esse caso, que requer uma pequena quantidade de vezes a repetição do comando **escreva**, essa é a solução mais simples. Mas se tivesse sido solicitada a impressão de 30 vezes "Olá, seja bem-vindo!"? ou 100 vezes? Certamente essa solução não seria a mais adequada.



Veja agora a solução para impressão de 30 vezes a mensagem "Olá, seja bem-vindo!" utilizando a estrutura de repetição **enquanto**. Observe com **atenção o comando!**

```
var contador := 1
enquanto contador <= 30 faça    # repete 'enquanto' o 'contador' for menor ou igual que 30
   escreva "Olá, seja bem-vindo!"
   contador := contador + 1       # a cada vez que escreve a mensagem, soma mais 1 no contador.
fim
escreva "Terminou!"

```

No exemplo, na linha 1, é declarada uma variável chamada **contador**, que recebe o valor inicial 1. A linha 2 possui a estrutura de repetição **enquanto**, na qual há a expressão lógica 'contador <= 30'. Se essa expressão lógica for **verdadeira**, os comandos entre as palavras **faça** e **fim** serão repetidos (linhas 3 e 4). Quando a

expressão se tornar **falsa** (o valor da variável **contador** for maior do que 30), os comandos presentes nas linhas 3 e 4 não serão mais executados e o algoritmo continuará após o comando **fim** da linha 5.

A expressão 'contador <= 30' foi utilizada para definir quantas vezes será repetido o conteúdo do comando **enquanto**. Nesse exemplo, se não fosse utilizada uma variável para definir a quantidade de vezes que ele deverá ser repetido e a expressão presente na linha 4, o algoritmo não seria finalizado, pois não encontraria a condição de parada, já que a expressão 'contador <= 30' sempre seria **verdadeira**, pois o valor presente na variável **contador** nunca se modificaria. Isso significa que o conteúdo (os comandos) entre **faça** e **fim** seria executado por infinitas vezes.

Em outras palavras, para que esse **enquanto** seja encerrado (finalizado), é necessário que a expressão 'contador <= 30' torne-se **falsa**. Assim, na linha 5, a cada vez que o conteúdo do comando **enquanto** é repetido, é somado mais 1 ao valor armazenado na variável **contador**. Dessa forma, após 30 repetições, o valor dessa variável será 31 e a expressão 'contador <= 30' deixará de ser **verdadeira**, tornando-se **falsa**. A partir desse momento, o **enquanto** não será mais executado.

Quando a **estrutura de decisão** não for mais executada, o programa continuará a partir da linha seguinte, que, no exemplo, equivale à linha 6.

Em um comando de repetição **enquanto**, a repetição será sempre condicionada a uma expressão lógica. Repare que, quando se utiliza a estrutura de decisão **se...então**, há necessidade, também, de uma expressão lógica. A diferença entre esses comandos é que na **estrutura de decisão** desvia-se o fluxo para um bloco de comandos, já na **estrutura de repetição** é preciso decidir se o conteúdo deverá ser executado (ou repetido).

Nesse sentido, enquanto o valor da variável **contador** for menor ou igual a 30, a expressão 'contador <= 30' causará a repetição do bloco **enquanto**, pois o resultado da expressão é **verdadeiro**. A partir do momento que **contador** possuir um valor maior do que 30, a expressão será **falsa** e a repetição deixará de ocorrer.

Você pode questionar: "Se não tivesse o comando da linha 4, o que aconteceria?". Caso em uma **estrutura de repetição** não seja alterado o valor da variável utilizada na expressão lógica do **enquanto**, o algoritmo repetirá o seu conteúdo por quantidade de vezes indefinida. Assim, acontecerá o que se chama de

***loop* infinito** **(é uma sequência de instruções em um programa de computador repetida infinitamente, em razão de não haver condição de parada ou de a condição existir, mas nunca ser atingida.)**, pois o algoritmo "ficará preso permanentemente na repetição", uma vez que a expressão 'contador <= 30' não será **falsa** em nenhum momento.

Atenção

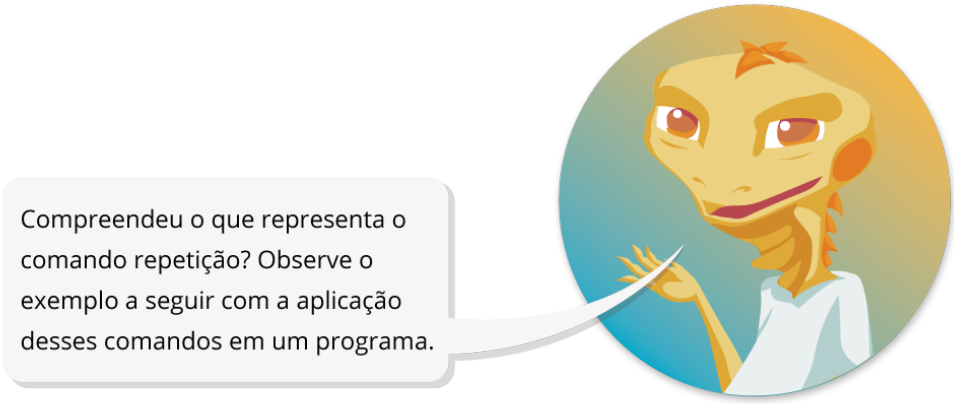
Um algoritmo deve possuir uma finitude, ou seja, ele deve encerrar após executar os seus passos com sucesso. Um algoritmo que não conclui a execução de seu código é considerado incorreto. Por esse motivo, deve-se sempre ficar atento para construir algoritmos que não fiquem executando em ***loop* infinito**.

Estruturas de repetição

São comandos que realizam a repetição de um determinado conjunto de ações (blocos de códigos). Essa repetição tem como objetivo realizar, de forma controlada, a execução de passos que devem ser repetidos, até que uma determinada condição seja **falsa**.

Atenção

O comando repetição **enquanto** também recebe o nome de **comando de iteração**, em razão de ser utilizado para realizar a repetição de um bloco de comando(s), e a palavra "iteração" ser um sinônimo de repetição. Então, não se deve confundir iteração com interação (uma ação recíproca entre dois ou mais elementos).



Exemplo 1 – Repetição Enquanto

Imagine o seguinte problema: criar um programa que simula uma calculadora que recebe uma quantidade indefinida de números inteiros. Essa calculadora apresentará, quando o valor 0 (zero) for informado, o resultado da soma de todos os números digitados.

Observe o código abaixo, que apresenta a solução para o problema descrito.

```
1  var soma := 0
2  var n := 1
3  escreva "Informe um valor e pressione Enter."
4  escreva "Digite 0 (zero) para encerrar o programa."
5  enquanto n <> 0 faça
6     n := leia_inteiro
7     soma := soma + n
8  fim
9
10 escreva "A soma total é {soma}"
11
```

Nesse programa, na linha 1, foi declarada a variável **soma** responsável por armazenar a soma de todos os números digitados. Observe que a variável foi inicializada com o valor 0 (zero). Na linha 2, a variável **n** será responsável por receber

os valores digitados e será também a variável utilizada no teste lógico da estrutura **enquanto**.

Durante a execução do programa, enquanto o valor de **n** for diferente de 0 (zero), as linhas de código 6 e 7 serão executadas repetidas vezes, ou seja, o programa irá receber um novo número inteiro e realizar a sua soma ao valor já armazenado na variável **soma** e, então, armazenar o novo valor (o resultado da soma) na variável **soma**.

Quando o valor informado por meio do comando **leia_inteiro** for 0 (zero), a expressão 'n <> 0' torna-se **falsa**. Se isso ocorrer, as linhas 6 e 7 não serão mais executadas. A partir de então, os comandos após a estrutura de repetição **enquanto** serão executados. No exemplo, o comando da linha 10 apresentará a soma total dos números informados.

Nesse algoritmo, não sabemos a quantidade de números que serão informados e, por esse motivo, ele será executado em uma quantidade de vezes ainda indefinida. Apesar dessa indefinição acerca da quantidade de repetições, o encerramento de execução ocorrerá quando o valor 0 (zero) for informado. Porém, se o valor 0 (zero) não for informado, o programa continuará recebendo uma quantidade de números não determinada.

#FicaDica

Para sistematizar melhor esse esquema, faça outras simulações, modificando os cenários, alterando as variáveis, testando as possibilidades. Assim, você terá mais segurança quando for realizar efetivamente algum comando utilizando essa estrutura!

Exemplo 2 - Repetição Enquanto

Imagine agora o seguinte problema: você e um grupo de amigos decidiram realizar uma atividade no parque e combinaram que tudo gasto pelo grupo seria dividido por igual entre todos os amigos presentes. Você e todos os seus amigos

foram convidados, mas nem todos puderam comparecer no dia. Um dos seus colegas fez um pequeno programa que recebe uma quantidade indefinida de valores (o gasto de cada amigo) e que, ao final, dividirá a soma pela quantidade de amigos presentes. O código abaixo apresenta essa solução:

```
var soma := 0.0
var valor_gasto := 0.0
var qtd_amigos_presentes := 0

escreva "Informe o valor gasto por cada amigo."
escreva "Digite um valor menor que zero para encerrar o programa."

enquanto valor_gasto >= 0 faça
   escreva "Quanto gastou o amigo nº {qtd_amigos_presentes+1}?"
   valor_gasto := leia_numero

   se valor_gasto >= 0 então
      qtd_amigos_presentes := qtd_amigos_presentes+1
      soma := soma + valor_gasto
   fim

fim

escreva "O total gasto foi {soma} reais"
escreva "{qtd_amigos_presentes} amigos estiveram presentes"
escreva "Cada amigo irá pagar {soma / qtd_amigos_presentes}"

```

Nesse exemplo, temos três variáveis que foram declaradas e são utilizadas durante o programa. A primeira variável, **soma**, é utilizada para armazenar a soma de todos os valores informados pelos amigos. A segunda, **valor_gasto**, é utilizada para armazenar o valor informado por cada amigo. Já a terceira, **qtd_amigos_presentes**, serve como um contador de quantos amigos informaram o gasto.

A estrutura de repetição **enquanto** presente nesse exemplo tem como expressão lógica que **enquanto o 'valor_gasto >= 0' (zero)** deve repetir os comandos presentes no bloco de repetição.

Tudo bem até aqui? É importante lembrar que essa estrutura é fundamental para você não perder tempo repetindo infinitas vezes um mesmo comando e até mesmo se perder na estruturação do código ao precisar repetir manualmente cada

trecho... Na próxima aula, você conhecerá os conceitos da estrutura de repetição **para** e continuará os estudos sobre as **estruturas de repetição**.

Até lá! 😉

Resumo

Nesta aula você conheceu a construção de algoritmos relacionados à estrutura de repetição **enquanto**. Também viu que os algoritmos podem ser construídos com **estruturas sequenciais** e **estruturas não sequenciais**. Nas **estruturas sequenciais**, já utilizadas nas aulas anteriores, construiu algoritmos sem a utilização de estrutura de repetição.

Em seguida, ao estudar as **estruturas não sequenciais**, aprendeu que elas dizem respeito a algoritmos que utilizam as estruturas de repetição. Conheceu, também, a estrutura de repetição **sem variável de controle** e **pré-testada** denominada de estrutura de repetição **enquanto**.

Em nossa próxima aula, você conhecerá a estrutura de repetição com variável de controle **para**.

Referências

Linguagem Potigol: Programação para todos. Disponível em: <http://potigol.github.io/>. Acesso em: 23 jul. 2018.

Capítulo 4: Orçamentos, pousos lunares e tratamento de erros. Disponível em: <http://turing.com.br/material/appy/cap4.html#somadora-infinita>. Acesso em: 23 jul. 2018.